Univerzitet u Beogradu
Mašinski fakultet

Miloš D. Petrašinović

Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi

doktorska disertacija

Beograd, 2025.

Univerzitet u Beogradu
Mašinski fakultet



Miloš D. Petrašinović

Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi

DOKTORSKA DISERTACIJA

Beograd, 2025.

University of Belgrade
Faculty of Mechanical Engineering



Miloš D. Petrašinović

A system for remote piloting of an aircraft with motion parameters in the feedback loop

DOCTORAL DISSERTATION

Belgrade, 2025.

Mentor:

dr Aleksandar Grbović, redovni profesor,
Univerzitet u Beogradu, Mašinski fakultet

Članovi komisije:

dr Aleksandar Bengin, redovni profesor,
Univerzitet u Beogradu, Mašinski fakultet

dr Nebojša Petrović, redovni profesor,
Univerzitet u Beogradu, Mašinski fakultet

dr Ognjen Peković, redovni profesor,
Univerzitet u Beogradu, Mašinski fakultet

dr Jelena Svorcan, redovni profesor,
Univerzitet u Beogradu, Mašinski fakultet

dr Gordana Kastratović, redovni profesor,
Univerzitet u Beogradu, Saobraćajni fakultet

Datum odbrane:

Svima koji su verovali

Zahvaljujem se pre svega roditeljima, celoj svojoj porodici i saradncima a zatim i svim prijateljima na strpljenju, velikoj podršci i razumevanju tokom čitavog školovanja.

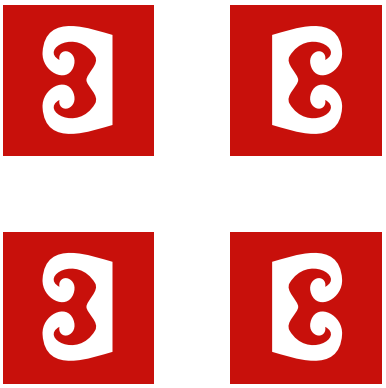
Zahvaljujem se supruzi Milici i sinu Danilu na svoj ljubavi koju su mi pružili.

Ovaj rad ne bi mogao da se realizuje bez podrške mentora Aleksandra Grbovića, kome se zahvaljujem na izdvojenom vremenu i korisnim savetima tokom izrade ove disertacije i čitavih studija.

Za veliku finansijsku podršku koja je bila neophodna za realizaciju ove disertacije se zahvaljujem mojim roditeljima Oliveri i Danilu, bratu Nikoli i sestri Tijani, kao i kompaniji Pink Research & Development Center čiji su osnivači Željko Mitrović, Danilo Petrašinović, Nikola Petrašinović, Miloš Petrašinović, Veljko Petrović i Mihailo Petrović.

Zahvaljujem se svim radnicima kompanije Pink Research & Development Center kao i svim saradnicima na dugogodišnjoj pomoći u realizaciji ove disertacije.

Samo ***Sloga***
Srbina Spasava



Rezime

Naslov doktorske disertacije: Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi

Pored konstruisanja letelica, od samog početka vazduhoplovstva najzahtevnije je bilo ovladati umećem pilotiranja letelicama. Nagli i nezaustavljiv razvoj i upotreba bespilotnih letelica je umeće pilotiranja ponovo vratilo u fokus istraživača i stručnjaka iz ove oblasti i to sa ključnom dilemom da li bespilotne letelice imaju pilote. U ovoj disertaciji je prikazan koncept sistema koji omogućava pilotu daljinski upravljane letelice da ima osećaj prilikom upravljanja kao da se nalazi u samoj letelici. Bespilotna letelica ili dron u svakom trenutku, i po potrebi, može da postane daljinski upravljana letelica. Sa daljinski upravljane letelice se informacija o njenom kretanju, u realnom vremenu, bežičnom komunikacijom, prenosi na simulator leta. Platforma simulatora leta se na osnovu ove informacije na odgovarajući način kreće u odnosu na bazu. U ovom konceptu sistema je primenjen simulator leta sa šest stepeni slobode kretanja čiji je mehanizam zasnovan na Stjuartovoj platformi sa paralelnim kinematičkim lancem i sa obrtnim aktuatorima (6RSS tip mehanizma). Pilot upravlja letelicom sa simulatora leta na osnovu slike sa kamere koja se nalazi na letelici ali i na osnovu kretanja platforme simulatora. Na ovaj način je dobijen zatvoren lanac pilotiranja letelicom sa parametrima kretanja letelice u povratnoj sprezi. Telo pilota se kreće u prostoru i njegov vestibularni sistem registruje kretanje koje je u skladu sa onim što vidi pomoću kamere na letelici i što može da očita sa odgovarajućih instrumenata. Prikazan je razvijen i implementiran algoritam za optimizaciju geometrije mehanizma za potrebe simuliranja leta. Projektovan je kompletan električni sistem letelice sa svim programima koji omogućavaju ovakvo pilotiranje. Sistem je primenjen na više različitih konstrukcionih tipova letelica čime je pokazano da je sistem primenljiv na većinu tipova letelica koji su danas u upotrebi i time se nagoveštava da ovaj sistem može da se iskoristi i za obuku pilota koji neposredno upravljaju letelicama. Rezultat rada je da je omogućeno preciznije i za pilote daleko prirodnije daljinsko pilotiranje letelicama.

Ključne reči: *Simulator leta, Bespilotna letelica, Dron, Rotaciona Stjuartova platforma, Daljinsko pilotiranje, Daljinski upravljanja letelica*
Naučna oblast: Mašinstvo
Uža naučna oblast: Vazduhoplovstvo
UDK broj:

Summary

Doctoral dissertation title: A system for remote piloting of an aircraft with motion parameters in the feedback loop

In addition to aircraft construction, since the very beginning of aviation, mastering the skill of piloting aircraft has been the most demanding. The rapid and unstoppable development and use of unmanned aerial vehicles have brought the skill of piloting back into the focus of researchers and experts in this field, with the key dilemma of whether unmanned aerial vehicles have pilots. This dissertation presents a concept of a system that allows a remotely controlled aircraft pilot to feel flying as if he is inside the aircraft itself. An unmanned aircraft or drone can become a remotely controlled aircraft at any moment and as needed. The information about its movement in real-time is transmitted to a flight simulator through wireless communication. Based on this information, the flight simulator platform moves appropriately relative to its base. This system concept applies a flight simulator with six degrees of freedom, whose mechanism is based on a Stewart platform with a parallel kinematic chain and rotary actuators (6RSS mechanism type). The pilot controls the aircraft from the flight simulator based on the camera feed from the aircraft and the motion of the simulator platform. This creates a closed-loop control system for piloting the aircraft with feedback on the aircraft's motion parameters. The pilot's body moves in space, and his vestibular system registers the motion consistent with what he sees through the camera on the aircraft and what he can read from the relevant instruments. An algorithm for optimizing the mechanism's geometry for flight simulation purposes has been developed and implemented. A complete electrical system for the aircraft has been designed, along with all the necessary programs to enable this type of piloting. The system has been applied to various types of aircraft designs, demonstrating its applicability to most aircraft types used today, suggesting that this system can also be used for training pilots who directly control aircraft. The result of the work is that more precise and far more natural remote aircraft piloting has been made possible for pilots.

Keywords: *Flight simulator, Unmanned aerial vehicle, Drone, Rotary Stewart platform, Remote piloting, Remote control of aircraft*
Scientific field: Mechanical Engineering
Scientific subfield: Aerospace Engineering
UDC number:

Sadržaj

Rezime **xi**

Summary **xiii**

Sadržaj **xv**

Spisak slika **xxiii**

Spisak tabela **xxxiii**

Spisak oznaka i skraćenica **xxxv**

Spisak kodova **xliii**

1 Uvod **1**
1.1 Naučni doprinos i očekivani rezultat disertacije . . . 2
1.2 Struktura disertacije . . . 2

2 Naučna zamisao disertacije **5**
2.1 Osnovni elementi sistema . . . 7
2.2 Povod i tok istraživanja . . . 13

3 Pregled stanja relevantne tehnike i prikaz dosadašnjih relevantnih istraživanja **19**
3.1 Simulator leta i pregled i odabir metode za proračun optimalne geometrije . . . 19
3.2 Bespilotne letelice . . . 22
3.2.1 Istorijski osvrt . . . 24
3.2.2 Primene bespilotnih letelica . . . 26
3.2.3 Klasifikacija bespilotnih letelica . . . 30
3.2.4 Domaće bespilotne letelice . . . 32
3.3 Standardi i važeći zakoni . . . 38
3.4 Alati za trodimenzionalno modeliranje i kompjutersku grafiku . . . 40
3.4.1 FreeCAD . . . 40
3.4.2 CGAL . . . 41
3.4.3 MeshLab . . . 41
3.4.4 Blender . . . 41
3.4.5 MakeHuman . . . 42
3.4.6 CesiumJS . . . 42

3.4.7 Unreal Engine . . . 42
3.4.8 Inkscape . . . 43
3.4.9 GIMP . . . 43
3.5 Kompozitni materijali, 3D štampači i skeneri . . . 43
3.6 Proizvodni procesi i obrada materijala . . . 48
3.7 Štampane ploče za elektroniku . . . 51
3.7.1 Projektovanje štampanih ploča . . . 51
3.7.2 Izrada štampanih ploča . . . 51
3.7.3 Sklapanje . . . 53
3.7.4 Verifikacija . . . 54
3.8 Mikrokontroleri i komunikacije . . . 55
3.9 Bežični prenos podataka . . . 59
3.10 Programiranje . . . 62
3.10.1 C, C++ i Fortran . . . 63
3.10.2 GNU Octave . . . 63
3.10.3 JavaScript . . . 64
3.10.4 OpenModelica . . . 64
3.10.5 Arduino . . . 65
3.10.6 Dokumentacija . . . 65
3.10.7 Verziranje koda . . . 66
3.10.8 Testiranje koda . . . 66
3.11 Manipulacija i obrada video-signala . . . 67
3.11.1 FFmpeg . . . 67
3.11.2 GStreamer . . . 67
3.11.3 Live555 . . . 68
3.11.4 OpenCV . . . 68
3.12 Virtuelna, proširena realnost i haptički uređaji . . . 68
3.13 JSLAB - okruženje za numeričke proračune . . . 70

4 Određivanje geometrijskih parametara i performansi mehanizma simulatora leta za sistem 75
4.1 Tipovi aktuatora . . . 79
4.1.1 Hidraulični cilindri - linearni aktuatori . . . 79
4.1.2 Električni linearni aktuatori . . . 79
4.1.3 Električni obrtni aktuatori . . . 80
4.2 Geometrijski parametri mehanizma . . . 81
4.3 Matrice transformacija koordinata . . . 89
4.4 Inverzni kinematički problem . . . 90
4.5 Direktni kinematički problem . . . 95
4.6 Brzine i ubrzanja mehanizma . . . 104
4.6.1 Brzine mehanizma . . . 104
4.6.2 Ubrzanja mehanizma . . . 105
4.7 Proračun elastičnih deformacija mehanizma . . . 106
4.7.1 Metod konačnih elemenata . . . 106
4.7.1.1 Hamiltonov princip . . . 106
4.7.1.2 Metoda pomeranja . . . 109

4.7.1.3 Spoljašnja opterećenja . . . 111
4.7.1.4 Konturni uslovi . . . 112
4.7.1.5 Metod Lagranževih množioca . . . 112
4.7.2 Element Ojler-Bernulijeva greda . . . 113
4.7.2.1 Osobine elementa . . . 113
4.7.2.2 Diferencijalne jednačine pomeranja . . . 114
4.7.2.3 Jednačine ravnoteže . . . 114
4.7.2.4 Aksijalno naprezanje . . . 115
4.7.2.5 Uvijanje . . . 116
4.7.2.6 Savijanje u xy ravni . . . 117
4.7.2.7 Savijanje u xz ravni . . . 118
4.7.2.8 Matrica krutosti . . . 119
4.7.2.8.1 Aksijalno naprezanje . . . 119
4.7.2.8.2 Uvijanje . . . 121
4.7.2.8.3 Savijanje u xy ravni . . . 122
4.7.2.8.4 Savijanje u xz ravni . . . 125
4.7.2.8.5 Matrica krutosti u lokalnom koordinatnom sistemu elementa . . . 127
4.7.2.8.6 Transformacija matrice krutosti . . . 128
4.7.3 Proračun deformacija . . . 128
4.8 Potrebni obrtni moment aktuatora . . . 131
4.8.1 Inverzni statički problem . . . 133
4.8.2 Inverzni dinamički problem . . . 133
4.9 Indikatori performansi mehanizma . . . 134
4.9.1 Algoritam SpaceSearch . . . 136
4.9.2 Maksimalni ostvarivi parametri . . . 139
4.9.3 Određivanje granične površi oblaka tačaka . . . 140
4.9.4 Radni prostor mehanizma . . . 142
4.10 Fiksni ugao ugradnje sfernih zglobova gornjih poluga . . . 143
4.11 Analitički proračun minimalnog rastojanja između donjih poluga . . . 145
4.12 Analitičko dimenzionisanje poluga . . . 148

5 Proračun optimalne geometrije mehanizma simulatora leta za sistem 151
5.1 Analiza i implementacija algoritma . . . 151
5.2 Funkcije ograničenja i prilagođenosti . . . 154
5.3 Podešavanje postupka proračuna . . . 156
5.4 Postupak proračuna . . . 157
5.5 Rezultati proračuna . . . 158

6 Priprema eksperimentalnog ispitivanja sistema 167
6.1 Simulator leta . . . 167
6.1.1 Umanjeni prototip MP-6DOF-M . . . 167
6.1.1.1 Električni sistem . . . 168
6.1.1.2 3D model konstrukcije i tehnička dokumentacija . . . 170
6.1.1.3 Proizvodnja . . . 173
6.1.1.4 Program za upravljanje . . . 177
6.1.1.5 Komunikacioni protokol PRDC_SCBP . . . 177

6.1.2 Prototip u stvarnoj veličini MP-6DOF-XXX . . . 179
6.1.2.1 Električni sistem . . . 179
6.1.2.1.1 Obrtni aktuator sa apsolutnim enkoderom . . . 180
6.1.2.1.1.1 Inicijalna ispitivanja . . . 181
6.1.2.1.1.2 Elektromotor i reduktor . . . 184
6.1.2.1.1.3 Elektronika Servo Board . . . 187
6.1.2.1.2 Frekventni regulator sa kočionim otpornikom i enkoderom . . . 188
6.1.2.1.2.1 Elektronika Angle Sensor . . . 191
6.1.2.1.3 Elektronika Control Board . . . 193
6.1.2.2 3D model kinematike i novi FreeCAD radni prostor ExternalAPI 194
6.1.2.2.1 Instalacija i upotreba radnog prostora ExternalAPI . . . 195
6.1.2.3 3D model konstrukcije i tehnička dokumentacija . . . 197
6.1.2.4 Proizvodnja . . . 205
6.1.2.5 Program za upravljanje . . . 218
6.1.2.5.1 Programiranje frekventnog regulatora . . . 221
6.1.2.6 Transportovanje simulatora . . . 222
6.1.2.7 Analiza pristupačnosti tehničkog rešenja simulatora leta . . . 223
6.1.3 Program MotionPlatformCI . . . 224
6.1.4 Kontroler MP-6DOF-CTRLR . . . 227
6.1.5 Kontroler MP-6DOF-M-CTRLR . . . 228
6.1.6 Filteri kretanja . . . 229
6.1.7 Značenje kodova statusa . . . 229
6.1.7.1 Status platforme . . . 230
6.1.7.2 Status proračuna . . . 230
6.1.7.3 Status aktuatora . . . 231
6.1.8 Definicija paketa i programa za komunikaciju . . . 231
6.1.8.1 MP-6DOF-XXX Control Board sa Servo Board i kontrolerom misije . . . 231
6.1.8.2 MP-6DOF-XXX Servo Board sa Angle Sensor . . . 236
6.1.8.3 MP-6DOF-M sa kontrolerom misije . . . 236
6.2 Letelice . . . 238
6.2.1 Multikopter SYMA X6 . . . 238
6.2.2 Kvadkopter QFLY-250 . . . 239
6.2.3 Raketoplan LABFLY-RM01 . . . 245
6.2.4 Letelica LABFLY-EM01 . . . 249
6.2.5 Letelica FW-01 . . . 254
6.2.6 Letelica REMILP-01 . . . 259
6.2.7 Autopilot program ArduPilot . . . 262
6.2.7.1 Kompajliranje . . . 264
6.2.7.2 Nova ploča . . . 265
6.2.7.3 Proširenje programa . . . 266
6.2.7.4 SITL . . . 267
6.2.7.5 AP_Periph . . . 267
6.2.7.6 Parametri . . . 269
6.2.7.7 Čitanje dnevnika leta . . . 270

6.2.7.8 MAVLink komunikacioni protokol . . . 271
6.2.7.8.1 Implementacija nove poruke . . . 277
6.2.8 Virtuelne letelice . . . 279
6.2.8.1 FlightGear . . . 279
6.2.8.2 AirSim . . . 281
6.2.8.3 ArduPilot SITL . . . 281
6.2.8.4 VehicleSimulator . . . 282
6.2.9 RTSP server . . . 284
6.2.10 Matematički modeli letelice i pilota za predupravljanje simulatora . . . 284
6.3 Pilotska kabina . . . 287
6.3.1 Interakcija pilota i sistema . . . 288
6.3.2 Prikaz telemetrijskih podataka . . . 300
6.3.3 RTSP klijent . . . 301
6.3.4 Aplikacija FlightControl . . . 302
6.3.5 Električna promena antena . . . 306
6.3.6 Sistem za automatsko usmeravanje antena . . . 307

7 Numerička i eksperimentalna ispitivanja elemenata i celog sistema 309
7.1 Numeričke simulacije mehanizma simulatora leta . . . 309
7.1.1 Vizualizacija simulatora leta pomoću Three.js biblioteke u JSLAB-u . . . 309
7.1.2 Određivanje potrebnog obrtnog momenta pomoću programa OpenModelica 311
7.2 Numerička i eksperimentalna verifikacija rešenja inverznog i direktnog kinematičkog problema . . . 315
7.3 Osnovno ispitivanje koncepta sistema sa umanjenim prototipom simulatora leta 319
7.4 Ispitivanje kašnjenja telemetrije . . . 324
7.4.1 Metodologija ispitivanja . . . 325
7.4.2 Rezultati ispitivanja kašnjenja pri korišćenju TCP-a lokalno . . . 329
7.4.3 Rezultati ispitivanja kašnjenja pri korišćenju TCP-a preko radio modula . 330
7.4.4 Analiza rezultata . . . 331
7.5 Ispitivanje kašnjenja video-snimka . . . 332
7.5.1 Metodologija ispitivanja . . . 332
7.5.2 Rezultati ispitivanja kašnjenja pri korišćenju WebRTC-a . . . 337
7.5.3 Rezultati ispitivanja kašnjenja pri korišćenju RTSP-a . . . 338
7.5.4 Rezultati ispitivanja kašnjenja pri korišćenju RTSP-a i LAN kamere . . . 339
7.5.5 Analiza rezultata . . . 340
7.6 Eksperimentalna validacija celog sistema u stvarnoj veličini . . . 342

8 Diskusija i predlog za nastavak istraživanja 349
8.1 Diskusija . . . 349
8.2 Predlog za nastavak istraživanja . . . 351

9 Zaključak 353

Literatura 361

A JSLAB klasa MP_6RSS_SOLVER 373

B C++ modul SolverMP6RSS 393

C JSLAB klasa MP_6RSS_PLOTTER 417

D Skripta solveInverseKinematics.jsl 431

E Skripta solveForwardKinematics.jsl 433

F Skripta calculateVelocitiesAndAccelerations.jsl 435

G Određivanje matrice krutosti elementa Ojler-Bernulijeva greda 441

H JSLAB klasa MP_6RSS_FEM_SOLVER 445

I Skripta solveInverseStaticFEM.jsl 451

J JSLAB klasa SPACE_SEARCH 455

K Skripta findWorkspaceParallel.jsl 463

L JSLAB funkcija boundary3D 465

M Skripta leversMinDistance.jsl 489

N Metoda checkJointsFixed 495

O Skripta analyticStructuralAnalysis.jsl 497

P JSLAB klasa OPTIM_RCMIGA 501

Q Skripte za primenu postupka proračuna optimalne geometrije 517

R Konfiguracije za MP-6DOF-M i MP-6DOF-XXX 527

S Program za upravljanje MP-6DOF-M 529

T Protokol za komunikaciju PRDC_SCBP 537
T.1 JavaScript implementacija . . . 537
T.2 Octave implementacija . . . 550
T.3 Arduino biblioteka . . . 554

U Pomoćna biblioteka PRDC_PACKET 561

V FreeCAD radni prostor ExternalAPI 565

W Skripta createModelFreeCAD.jsl 579

X Programi za upravljanje MP-6DOF-XXX 583
X.1 Control Board . . . 583
X.2 Servo Board . . . 631
X.3 Angle Sensor . . . 658

Y Skripta flightGearSimulationUDP.jsl 681

Z Skripte ArdupilotLink.jsl i ArdupilotSitlLink.jsl **687**

AA Skripta ServerRTSP.jsl **723**
AA.1 Modul server-rtsp-live555 724
AA.2 Kompajliranje biblioteke LIVE555 740

AB Skripta ClientRTSP.jsl **743**
AB.1 Modul decoder-ffmpeg 765
AB.2 Modul stream-decoder-ffmpeg 775
AB.3 Modul stream-decoder-gst 796

AC Skripta serveOculus.jsl **811**

AD Skripta showModel3D.jsl za vizualizaciju simulatora leta pomoću Three.js **815**

AE Skripte solveInverseDynamicsOpenModelica.jsl za analizu mehanizma pomoću OpenModelica programa **821**

Indeks **835**

Biografija autora **839**

Spisak slika

2.1 Tipična misija za sistem . . . 7
2.2 Prva bespilotna letelica - "Aerial Torpedo" (preuzeto iz [2]) . . . 8
2.3 Osnovni dijagram sistema . . . 8
2.4 Dijagram sistema sa neophodnim elementima . . . 9
2.5 Jedan od prvih poznatih simulator leta - Antoinette simulator (preuzeto iz [3]) 9
2.6 Simulator leta Link Flight Trainer AN-2550-1 u Muzeju vazduhoplovstva u Beogradu . . . 10
2.7 Fotografije simulatora leta kompanije JAT . . . 11
2.8 Moderni simulator leta za avion Boeing 787 (preuzeto iz [6]) . . . 12
2.9 Razvoj mehanizma simulatora leta - od ideje do materijalizacije . . . 13
2.10 Samostalna izrada prvog multikoptera . . . 14
2.11 Upotreba simulatora leta svemirskog broda na setu televizijske serije . . . 14
2.12 Sajam UMEX 2022 u Abu Dabiju . . . 15
2.13 Muzej Steven F. Udvar-Hazy Centar . . . 16
2.14 Prikaz dronova iz serije IKA na vojnom aerodrom pukovnik-pilot Milenko Pavlović u Batajnici . . . 17

3.1 Stjuartova platforma sa obrtnim aktuatorima . . . 20
3.2 Daljinsko upravljanje letelicom . . . 22
3.3 RQ-1A Predator u Muzeju vazduhoplovstva u Beogradu . . . 25
3.4 Kontrolna soba letelice RQ-1A Predator (preuzeto iz [82]) . . . 25
3.5 Primer transporta tereta i ispuštanja tereta sa padobranom . . . 26
3.6 Gašenje požara multikopterom IKA-01 . . . 27
3.7 Gašenje požara multikopterom IKA-20 . . . 27
3.8 Onesposobljavanje osumnjičenog pomoću multikoptera . . . 28
3.9 Pucanje iz pištolja sa multikopterom . . . 28
3.10 Let električnog multikoptera na 6000 metara nadmorske visine . . . 29
3.11 Ispitivanje uticaja ručnog bacača raketa M79 Osa na delove multikoptera . . . 29
3.12 Multikopter za borbu protiv drugih dronova . . . 30
3.13 Klizač - neupravljana letelica (levo) i kvadkopter - daljinski upravljana letelica(desno) . . . 31
3.14 Domaća bespilotna letelica Vrabac (preuzeto iz [83]) . . . 32
3.15 Domaća bespilotna letelica Pegaz 011 (preuzeto iz [84]) . . . 33
3.16 Domaća bespilotna letelica IKA-BOMBARDER . . . 34
3.17 Prikaz ispuštanja avio bombi sa multikoptera . . . 35
3.18 Prikaz slike sa infracrvene kamere prilikom osmatranja mete . . . 36

3.19 Prikaz bombardovanja avio-bombama prečnika 60 mm . . . 36
3.20 Dejstvo avio-bombe prečnika 60 mm posmatrano termalnom kamerom . . . 37
3.21 Poletanje IKA-BOMBARDER-a sa vozila u pokretu . . . 37
3.22 Poletanje IKA-BOMBARDER-a sa plovnog objekta . . . 38
3.23 Formiranje kompozitnog materijala . . . 43
3.24 Primer kompozitnih delova i alata za izradu . . . 44
3.25 Različiti materijali vlakna . . . 45
3.26 Ručno bušenje otvora na kompozitnoj konstrukciji korišćenjem alata . . . 45
3.27 Industrijski štampači FFF tehnologije . . . 46
3.28 Industrijski štampači MSLA tehnologije . . . 46
3.29 Primer metalnih delova dobijenih 3D štampanjem . . . 47
3.30 Primer 3D skeniranja elise za multikopter . . . 48
3.31 CNC mašina za rezanje laserom . . . 49
3.32 Izrada alata na CNC glodalici . . . 49
3.33 Opsecanje kompozitne ploče na CNC ruteru . . . 50
3.34 Primer strukture letelice izrađene od brizgane plastike . . . 50
3.35 Primer strukture letelice izrađene od ekspandiranog polipropilena . . . 51
3.36 Vađenje ploče iz rastvora . . . 52
3.37 Industrijski proizvedene štampane ploče . . . 52
3.38 CNC mašina za izuzimanje i postavljanje komponenti na štampanu ploču . . . 53
3.39 Postupak postavljanja komponenti na štampanu ploču . . . 53
3.40 Verifikacija elektronike sa laboratorijskim napajanjem i termalnom kamerom . . . 54
3.41 Razvojna ploča sa mikrokontrolerom (levo) i programator za mikrokontroler (desno) . . . 55
3.42 Primer elektronike sa mikrokontrolerom . . . 55
3.43 Blok dijagram 2xMIMO radio veze između dva uređaja . . . 60
3.44 Primer analognog FPV sistema . . . 60
3.45 Primer digitalnog FPV sistema . . . 61
3.46 Primer radio modula u letelici . . . 61
3.47 Uređaj Oculus Quest 2 . . . 69
3.48 Logo projekta JSLAB . . . 70
3.49 Glavni prozor programa JSLAB . . . 71
3.50 Prozor uređivača koda programa JSLAB . . . 72
3.51 Kompletan grafički korisnički interfejs programa JSLAB sa nacrtanim grafikom u zasebnom prozoru . . . 74

4.1 3D model modernog simulatora leta . . . 75
4.2 Kinematički model Stjuartove platforma sa obrtnim aktuatorima . . . 76
4.3 Otvoren kinematički lanac . . . 77
4.4 Zatvoreni kinematički lanac . . . 77
4.5 Modeliranje platforme simulatora . . . 78
4.6 Hidraulična platforma (preuzeto iz [123]) . . . 79
4.7 Električni linearni aktuatori (preuzeto iz [124]) . . . 79
4.8 eMotion-1500 Bosch Rexroth (preuzeto iz [125]) . . . 80
4.9 Električni obrtni aktuator Stjuartove platforme . . . 80
4.10 Koordinatni sistemi baze i platforme mehanizma . . . 81

4.11 Geometrijski parametri jedne noge mehanizma . . . 82
4.12 Geometrijski parametri baze . . . 83
4.13 Geometrijski parametri platforme . . . 85
4.14 Koordinatni sistem aktuatora i donje poluge . . . 85
4.15 Koordinatni sistem gornjeg zgloba . . . 87
4.16 Ose za vezu sfernih zglobova . . . 88
4.17 Tri translacije duž x, y i z osa . . . 90
4.18 Tri rotacije za uglove φ, θ i ψ . . . 91
4.19 Uprošćen prikaz geometrije mehanizma . . . 94
4.20 Geometrijska ilustracija Njutnove metode . . . 96
4.21 Vektorska petlja jedne noge mehanizma . . . 97
4.22 Algoritam za rešavanje direktnog kinematičkog problema . . . 100
4.23 Prikaz mehanizma sa među rešenjima . . . 101
4.24 Promena unutrašnjih koordinata . . . 102
4.25 Promena spoljašnjih koordinata . . . 103
4.26 Element tipa Ojler-Bernulijeva greda . . . 113
4.27 Prikaz komponenata napona na elementarnom delu trodimenzionalnog tela . . . 114
4.28 Aksijalno naprezanje elementarnog dela grede . . . 115
4.29 Uvijanje elementarnog dela grede . . . 116
4.30 Savijanje elementarnog dela grede u xy ravni . . . 117
4.31 Savijanje elementarnog dela grede u xz ravni . . . 118
4.32 Funkcije oblika za zatezanje . . . 120
4.33 Funkcije oblika za savijanje u xy ravni . . . 124
4.34 Funkcije oblika za savijanje u xz ravni . . . 126
4.35 Prikaz konačnih elemenata i opterećenja . . . 129
4.36 Prikaz uvećanih elastičnih deformacija konstrukcije . . . 131
4.37 Promena indikatora manipulabilnosti u ravni . . . 134
4.38 Promena indikatora spretnosti u ravni . . . 135
4.39 Promena indikatora maksimalnog potrebnog momenta u ravni . . . 135
4.40 Grafički prikaz algoritma spaceSearch . . . 137
4.41 Pretraga 1D prostora . . . 138
4.42 Pretraga 2D prostora . . . 138
4.43 Promena indikatora maksimalnog ugla propinjanja u ravni . . . 139
4.44 Promena indikatora maksimalnog ugla valjanja u ravni . . . 140
4.45 Spoljna površ radnog prostora odozgo (levo) i odozdo (desno) . . . 142
4.46 Izometrijski prikazi spoljne površi radnog prostora . . . 142
4.47 Karakteristični vektori za određivanje zasićenja sfernih zglobova . . . 143
4.48 Algoritam funkcije disksDistance3D . . . 145
4.49 Rezultat proračuna minimalnog rastojanja između donjih poluga . . . 146
4.50 Statički dijagrami gornje poluge . . . 149
4.51 Statički dijagrami donje poluge . . . 149

5.1 Dijagram toka genetskog algoritma . . . 152
5.2 Referentne tačke za proračun . . . 155
5.3 Promena vrednosti funkcije prilagođenosti u odnosu na generaciju. . . . 158
5.4 Kinematički model proračunate geometrije mehanizma . . . 159

5.5 (**a**) Maksimalni potrebni obrtni moment aktuatora i (**b**) spretnost mehanizma . 160
5.6 (**a**) Maksimalni nagib i (**b**) uglovi nagiba 160
5.7 Prikaz referentnih tačaka u okviru radnog prostora 161
5.8 Render 3D modela mehanizma 162
5.9 Maksimalni potrebni moment aktuatora u referentnim tačkama 162
5.10 Maksimalna radijalna sila u referentnim tačkama 163
5.11 Maksimalna aksijalna sila u referentnim tačkama 163
5.12 Spoljna površ radnog prostora odozgo 164
5.13 Spoljna površ radnog prostora odozdo 164
5.14 Izometrijski prikaz spoljne površi radnog prostora odozgo 165
5.15 Izometrijski prikaz spoljne površi radnog prostora odozdo 165

6.1 Servo motori za umanjeni prototip 168
6.2 Delovi servo motora 169
6.3 Arduino Uno razvojna ploča (levo) i dodatak za nju (desno) 169
6.4 Blok dijagram elektronike umanjenog prototipa 170
6.5 Sferni zglobovi za gornje poluge umanjenog prototipa 170
6.6 Gabaritne mere umanjenog prototipa 171
6.7 3D model sklopa MP-6DOF-M 171
6.8 Razdvojeni delovi sklopa MP-6DOF-M 172
6.9 Crteži delova MP-6DOF-M za rezanje laserom 173
6.10 Delovi MP-6DOF-M dobijeni rezanjem laserom 174
6.11 Sklapanje umanjenog prototipa krajem oktobra 2016. godine 174
6.12 Sferni zglobovi mehanizma umanjenog prototipa 175
6.13 Konačni izgled umanjenog prototipa (fizički i 3D model) 175
6.14 Model pilotske kabine na umanjenom prototipu 176
6.15 Prototip umanjenog prototipa na sajmu tehnike 2018. godine u Beogradu . . . 176
6.16 Blok dijagram elektronike prototipa u stvarnoj veličini 180
6.17 Postavka inicijalnog ispitivanja aktuatora 181
6.18 Nulta verzija elektronike `Servo Board` (levo) i frekventni regulator (desno) . . . 182
6.19 Blok dijagram `Servo Board V0` elektronike 182
6.20 Šest elektronika `Servo Board V1` sa nosačem 183
6.21 Elektronika `Control Board V0` (dole desno) 183
6.22 Elektromotor i pužni reduktor izabranog tipa 184
6.23 Delovi elektromotora iz kataloga proizvođača (preuzeto iz [140]) 184
6.24 Geometrijske karakteristike elektromotora (preuzeto iz [140]) 185
6.25 Geometrijske karakteristike reduktora (preuzeto iz [139]) 186
6.26 Elektronika `Servo Board V2` 187
6.27 Blok dijagram `Servo Board V2` elektronike 187
6.28 Frekventni regulator ENC EN600 188
6.29 Električna šema napajanja prvog frekventnog regulatora 189
6.30 Kočioni otpornici (levo) i enkoder (desno) 190
6.31 Blok dijagram elektronike `Angle Sensor` 191
6.32 Panel sa 10 elektronika `Angle Sensor` 192
6.33 Štampana ploča elektronike `Angle Sensor` 192
6.34 Električna šema elektronike `Angle Sensor` 193

6.35 Elektronika `Control Board V1` . . . 193
6.36 Blok dijagram elektronike `Control Board V1` . . . 194
6.37 Ikonica FreeCAD ExternalAPI radnog prostora . . . 194
6.38 3D model kinematike mehanizma . . . 195
6.39 Struktura direktorijuma FreeCAD ExternalAPI radnog prostora . . . 196
6.40 Zglobni ležaj koji se koristi kao sferni zglob gornje poluge mehanizma . . . 198
6.41 Gabaritne mere prototipa u stvarnoj veličini . . . 198
6.42 3D model sklopa MP-6DOF-XXX . . . 199
6.43 Razdvojeni delovi sklopa MP-6DOF-XXX . . . 199
6.44 Nosači sfernih zglobova . . . 200
6.45 Crteži najznačajnijih delova MP-6DOF-XXX za rezanje laserom . . . 200
6.46 Nosač aktuatora . . . 201
6.47 Crteži preseka dela gornje poluge koji omogućava slobodnu rotaciju jednog kraja u odnosu na drugi . . . 201
6.48 3D model obrtne ose sa sfernim zglobom . . . 202
6.49 Tehnički crtež vratila donjih poluga . . . 202
6.50 3D model vratila donjih poluga . . . 202
6.51 Tehnički crtež sklopa sfernog zgloba . . . 203
6.52 Moguće ugaono pomeranje zgloba u zavisnosti od prečnika osovine . . . 203
6.53 Primer provere čvrstoće donje poluge metodom konačnih elemenata . . . 204
6.54 Model prototipa simulatora leta u stvarnoj veličini pored 3D modela operatera 204
6.55 Elementi konstrukcije prototipa u stvarnoj veličini nakon rezanja laserom . . . 205
6.56 Zavareni nosači aktuatora (levo) i obrađeni zavareni nosač aktuatora (desno) . . . 205
6.57 Pripremljeni elementi od cevi (levo) i zavarivanje elemenata platforme mehanizma (desno) . . . 206
6.58 Zavarivanje elemenata baze . . . 206
6.59 Rezanje elementa za vezu gornjeg sfernog zgloba . . . 207
6.60 Rezanje elementa za vezu donjeg sfernog zgloba . . . 207
6.61 3D štampanje alata za pozicioniranje elemenata . . . 207
6.62 Upotreba 3D štampanog alata za pozicioniranje elemenata na donjoj poluzi . . . 208
6.63 Zavarivanje elemenata za vezu gornjih i donjih sfernih zglobova . . . 208
6.64 Zavarivanje vratila i donje poluge (levo) i elementi gornjih poluga (desno) . . . 208
6.65 Prvo ožičavanje električnog sistema . . . 209
6.66 Prvo programiranje mikrokontrolera . . . 209
6.67 Prvo uključivanje nulte verzije električnog sistema . . . 210
6.68 Potpuno sklopljen i zatim rasklopljen mehanizam prototipa u stvarnoj veličini . 210
6.69 Farbanje delova mehanizma . . . 211
6.70 Prvo potpuno sklapanje prototipa simulatora leta . . . 212
6.71 Novi senzor ugla . . . 212
6.72 Ugrađene poslednje verzije elektronike `Servo Board` . . . 213
6.73 Ugrađena poslednja verzija elektronike `Control Board` . . . 213
6.74 Konektor za električnu vezu platforme i baze . . . 214
6.75 Veza enkodera i elektromotora . . . 214
6.76 Električni panel prototipa . . . 215
6.77 Stepenice za penjanje na platformu prototipa simulatora leta . . . 215
6.78 Finalni izgled pilotske kabine prototipa simulatora leta . . . 216

6.79 Merenje mase prototipa simulatora leta . . . 217
6.80 Prvo javno predstavljanje prototipa simulatora leta . . . 217
6.81 Kalibracija ugla otklona aktuatora . . . 219
6.82 Podizanje simulatora leta za transport paletarom . . . 222
6.83 Kartica aplikacije `MotionPlatformCI` za kontrolu mehanizma na osnovu unutrašnjih koordinata . . . 224
6.84 Kartica aplikacije `MotionPlatformCI` za kontrolu mehanizma na osnovu spoljašnjih koordinata . . . 224
6.85 Kartica za podešavanje aplikacije `MotionPlatformCI` . . . 225
6.86 Dostupna podešavanja za samu aplikaciju `MotionPlatformCI` . . . 225
6.87 Podešavanja u `MotionPlatformCI` za eksterne kontrolere i povezivanje sa letelicom putem `MAVLink`-a . . . 226
6.88 Prikaz panela sa stanjem platforme . . . 226
6.89 Prikaz detaljnijih informacija za parametre stanja . . . 227
6.90 Kontroler inverzne kinematike MP-6DOF-CTRLR . . . 227
6.91 Kontroler direktne kinematike `MP-6DOF-M-CTRLR` . . . 228
6.92 Kontrola oba prototipa mehanizma pomoću `MP-6DOF-M-CTRLR` kontrolera . . . 228
6.93 Blok dijagram klasičnog filtera kretanja . . . 229
6.94 Konfiguracija multikoptera SYMA X6 . . . 238
6.95 Aerotunelsko ispitivanje multikoptera Syma X6 . . . 239
6.96 Kvadkopter QFLY-250 klase 250 . . . 240
6.97 Poletanje kvadkoptera QFLY-250 . . . 240
6.98 Spajanje aktuatora sa pločom za distribuciju električne energije . . . 241
6.99 Montaža motora i elektronike na konstrukciju kvadkoptera . . . 242
6.100 Montaža kontrolera leta na konstrukciju kvadkoptera . . . 242
6.101 Daljinski upravljač FLYSKY FS-i6 . . . 243
6.102 Daljinski upravljač za kvadkopter sa analognim ekranom . . . 243
6.103 Kvadkopter QFLY-250 na sajmu tehnike 2017. godine u Beogradu . . . 244
6.104 Radio-prijemnik i predajnik za bežičnu komunikaciju . . . 244
6.105 Predajnik sa senzorom ubrzanja i orijentacije . . . 245
6.106 3D model letelice LABFLY-RM01 . . . 246
6.107 Horizontalni i vertikalni rep sa komandnim površinama . . . 247
6.108 Otklon krmila pravca . . . 247
6.109 Otklon krmila visine . . . 247
6.110 Otklon krilaca . . . 248
6.111 Prednji deo trupa letelice . . . 248
6.112 Priprema 3D modela nosa letelice za 3D štampanje . . . 248
6.113 Određivanje momenata inercije letelice LABFLY-RM01 . . . 249
6.114 3D model letelice LABFLY-EM01 . . . 250
6.115 Letna ispitivanja letelice LABFLY-EM01 . . . 250
6.116 Konačan izgled letelice LABFLY-EM01 . . . 251
6.117 Primer štampane ploče za elektroniku letelice izrađene foto postupkom . . . 251
6.118 Izrada štampane ploče za elektroniku glodanjem . . . 252
6.119 Predajnik za slanje komandi za upravljanje letelicom . . . 253
6.120 Model letelice *LABFLY-EM01* za prikaz njene elektronike u trupu . . . 253
6.121 Trodimenzionalni model letelice FW-01 . . . 254

6.122 Konačan izgled letelice FW-01 . . . 254
6.123 Realističan prikaz kompletnog sistema sa letelicom FW-01 . . . 255
6.124 Katapult za letelicu . . . 255
6.125 Ispitivanje kretanja kolica katapulta . . . 256
6.126 Letelica FW-01 na katapultu . . . 257
6.127 Katapultiranje letelice FW-01 . . . 257
6.128 Pogled sa kamere na letelici FW-01 . . . 258
6.129 Pogled sa kamere na letelici FW-01 sa unapređenim video-sistemom . . . 258
6.130 Izgled letelice REMILP-01 . . . 259
6.131 Električni sistem letelice REMILP-01 . . . 260
6.132 Let letelice REMILP-01 . . . 261
6.133 Razne letelice zajedno sa prototipom simulatora leta . . . 261
6.134 Primer konstruktivnih tipova multikoptera, HEXA X (levo) i QUAD X (desno) 263
6.135 Prikaz putanje leta na osnovu podataka iz dnevnika leta . . . 270
6.136 Preuzimanje parametra putem MAVLink-a . . . 272
6.137 Preuzimanje svih parametra putem MAVLink-a . . . 273
6.138 Podešavanje parametara putem MAVLink-a . . . 273
6.139 Slanje komande MAVLink porukom COMMAND_LONG . . . 274
6.140 Slanje komande MAVLink porukom COMMAND_INT . . . 274
6.141 Preuzimanje misije putem MAVLink-a . . . 275
6.142 Otpremanje misije putem MAVLink-a . . . 276
6.143 Čitanje fajla putem MAVFTP-a . . . 277
6.144 Tok podataka između JSLAB-a i FlightGear-a . . . 280
6.145 Povezivanje FlightGear programa sa umanjenim prototipom simulatora leta . . . 280
6.146 Prikaz letelice, animacija ispuštanja bombe i eksplozija pomoću Unreal Engine-a 281
6.147 Prikaz instrumenata u JSLAB-u nakon povezivanja sa ArduPilot SITL . . . 282
6.148 Upravljanje virtuelnom letelicom programa VehicleSimulator . . . 283
6.149 Virtuelna soba za IKA-TRAINER . . . 283
6.150 Blok dijagram sa prenosnom funkcijom pilota . . . 285
6.151 Pilotska kabina na platformi simulatora leta . . . 287
6.152 Blok dijagram pilotske kabine simulatora leta `MP-6DOF-XXX` . . . 288
6.153 Pilotska kabina iz ugla pilota . . . 289
6.154 Označeni delovi daljinskog upravljača IKA-CTRL . . . 290
6.155 Primer prikaza za Oculus . . . 291
6.156 Uticaj vertikalnog pomeranja kontrolne palice LJ1 . . . 292
6.157 Uticaj horizontalnog pomeranja kontrolne palice LJ1 . . . 292
6.158 Uticaj vertikalnog pomeranja kontrolne palice RJ1 . . . 293
6.159 Uticaj horizontalnog pomeranja kontrolne palice RJ1 . . . 293
6.160 Uticaj vertikalnog pomeranja kontrolne palice LJ2 . . . 294
6.161 Uticaj horizontalnog pomeranja kontrolne palice LJ2 . . . 294
6.162 Uticaj vertikalnog pomeranja kontrolne palice RJ2 . . . 295
6.163 Uticaj promene položaja prekidača LT1 . . . 295
6.164 Uticaj promene položaja prekidača LT2 . . . 296
6.165 Uticaj promene položaja prekidača RT2 . . . 296
6.166 Uticaj promene položaja prekidača RT1 . . . 297
6.167 Uticaj tastera LS1 . . . 297

6.168 Uticaj tastera LS2 . . . 298
6.169 Uticaj tastera RS1 . . . 298
6.170 Uticaj promene položaja taster RS2 . . . 299
6.171 Uticaj promene položaja prekidača LT3 . . . 299
6.172 Prikaz telemetrijskih podataka sa letelice u JSLAB-u . . . 300
6.173 Početni prozor aplikacije FlightControl za izbor letelice . . . 303
6.174 Aplikacija FlightControl u tabu za let . . . 303
6.175 Prikaz statusa podsistema u FlightControl aplikaciji . . . 304
6.176 Prikaz slike sa kamere i instrumenata nakon povezivanja aplikacije FlightControl sa letelicom . . . 304
6.177 Prikaz kanala daljinskog upravljača u aplikaciji FlightControl . . . 305
6.178 Podešavanje podsistema platforme u FlightControl aplikaciji . . . 306
6.179 Prikaz statusa podsistema platforme u FlightControl aplikaciji . . . 306
6.180 Uređaj Switch Box . . . 307
6.181 Uređaj Automatic Antenna Tracker . . . 307
6.182 Montiranje stativa i AAT uređaja . . . 308

7.1 Vizualizacija simulatora leta pomoću Three.js biblioteke u JSLAB-u . . . 310
7.2 Vizualizacija simulatora leta sa pilotskom kabinom pomoću Three.js biblioteke u JSLAB-u . . . 310
7.3 OpenModelica model mehanizma . . . 311
7.4 Vizualizacija OpenModelica modela mehanizma . . . 312
7.5 Promena potrebnog obrtnog momenta . . . 313
7.6 Promena potrebnog obrtnog momenta sa ograničenim prikazom od -500 Nm do 500 Nm . . . 314
7.7 Podešavanje ugla motora M5 korišćenjem programa MotionPlatformCI . . . 315
7.8 Prikaz rotacije motora M5 za 50°, -50°, 30° i -30° . . . 316
7.9 Prikaz kretanja mehanizma simulatora leta sa upravljanjem samo motora M5 . 316
7.10 Kalibracija ugla servo motora . . . 317
7.11 Skeniranje simulatora leta sa uređajem *3DMakerpro Mole* . . . 318
7.12 Rezultat skeniranja simulatora leta u programima *JMStudio* (levo) i *FreeCAD* (desno) . . . 318
7.13 Blok dijagram za osnovno ispitivanje koncepta sistema . . . 319
7.14 Modul sa senzorom ubrzanja i orijentacije na platformi umanjenog prototipa simulatora leta . . . 319
7.15 Modul sa senzorom ubrzanja i orijentacije na kvadkopteru . . . 320
7.16 Rezultati merenja na kvadkopteru . . . 321
7.17 Rezultati merenja na simulatoru leta . . . 322
7.18 Promena ugla propinjanja kvadkoptera i simulatora tokom eksperimenta . . . 323
7.19 Promena ugla valjanja kvadkoptera i simulatora tokom eksperimenta . . . 323
7.20 Odstupanje ugla propinjanja simulatora u odnosu na kvadkopter . . . 324
7.21 Daljinski upravljač sa tablet računarom Getac F110 G6 (levo) i laptop računar Getac X600 sa digitalnim radio modulom (desno) . . . 325
7.22 Postavka ispitivanja kašnjenja podataka usled korišćenja TCP protkola za telemetriju . . . 325
7.23 Postavka ispitivanja kašnjenja podataka usled korišćenja TCP-a i bežične komunikacije za telemetriju . . . 326

7.24 Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja telemetrije preko TCP-a lokalno . . . 329
7.25 Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja telemetrije preko TCP-a i radio modula . . . 330
7.26 Postavka ispitivanja kašnjenja video-snimka korišćenjem WebRTC-a . . . 333
7.27 Postavka ispitivanja kašnjenja video-snimka korišćenjem RTSP-a . . . 333
7.28 Postavka ispitivanja kašnjenja video-snimka korišćenjem RTSP-a i LAN kamere 334
7.29 Snimanje ekrana tableta `Getac F110 G6` pomoću kamere sa laptopa `Getac X600` 334
7.30 Snimanje ekrana tableta `Getac F110 G6` pomoću `LAN` kamere na letelici . . . 335
7.31 Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka preko `WebRTC`-a . . . 337
7.32 Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka preko `RTSP`-a . . . 338
7.33 Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka preko `RTSP`-a i `LAN` kamere na letelici . . . 339
7.34 Prvo ispitivanje mehanizma u stvarnoj veličini novembra 2019. godine . . . 342
7.35 Ispitivanje prototipa simulatora leta decembra 2019. godine . . . 343
7.36 Translacija platforme duž x ose . . . 343
7.37 Translacija platforme duž y ose . . . 344
7.38 Translacija platforme duž z ose . . . 344
7.39 Rotacija platforme oko x ose . . . 345
7.40 Rotacija platforme oko y ose . . . 345
7.41 Rotacija platforme oko z ose . . . 346
7.42 Ispitivanje prototipa simulatora leta sa 100 kilograma tereta . . . 346
7.43 Validacija kompletnog sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi . . . 347

Spisak tabela

3.1 Tehničke karakteristike bespilotne letelice Vrabac . . . 32
3.2 Tehničke karakteristike bespilotne letelice Pegaz . . . 33
3.3 Osnovne tehničke karakteristike bespilotne letelice IKA-BOMBARDER . . . 34
3.4 Performanse leta bespilotne letelice IKA-BOMBARDER . . . 35
3.5 Karakteristike naoružanja bespilotne letelice IKA-BOMBARDER . . . 35

4.1 Primer konfiguracije mehanizma . . . 89
4.2 Unutrašnje koordinate po iteracijama . . . 102
4.3 Spoljašnje koordinate po iteracijama . . . 103
4.4 Usvojeni parametri za analizu čvrstoće . . . 128
4.5 Čvorovi donjih sfernih zglobova . . . 130
4.6 Čvorovi gornjih sfernih zglobova . . . 130

5.1 Opseg vrednosti geometrijskih parametara . . . 154
5.2 Promenljive funkcije prilagođenosti i funkcije ograničenja . . . 156
5.3 Podešavanja genetskog algoritma . . . 157
5.4 Podešavanja kriterijuma za prekid proračuna . . . 157
5.5 Geometrijski parametri proračunatog mehanizma . . . 159
5.6 Maksimalne vrednosti spoljašnjih koordinata . . . 161
5.7 Minimalne vrednosti spoljašnjih koordinata . . . 161

6.1 Geometrijski parametri prototipa MP-6DOF-M . . . 168
6.2 Spisak delova MP-6DOF-M za rezanje laserom . . . 172
6.3 Tehničke karakteristike elektromotora ELVEM 6T1 90L4 B14 . . . 185
6.4 Vrednosti geometrijskih karakteristika elektromotora ELVEM 6T1 90L4 B14 . . . 186
6.5 Tehničke karakteristike reduktora CHM 90 50 B14 . . . 186
6.6 Vrednosti geometrijskih karakteristika reduktora CHM 90 50 B14 . . . 186
6.7 Tehničke karakteristike frekventnog regulatora ENC EN600-2S0015 . . . 189
6.8 Poruka strukture `packetOutSB` . . . 232
6.9 Poruka strukture `packetInSB` . . . 233
6.10 Poruka ID = 11 strukture `packetInMC` . . . 233
6.11 Poruka ID = 12 strukture `packetInMC` . . . 233
6.12 Poruka ID = 13 strukture `packetInMC` . . . 234
6.13 Poruka ID = 201 strukture `packetInMC` . . . 234
6.14 Poruka ID = 0 strukture `packetOutMC` . . . 234
6.15 Poruka ID = 101 strukture `packetOutMC` . . . 235
6.16 Poruka ID = 102 strukture `packetOutMC` . . . 235

6.17 Poruka ID = 103 strukture packetOutMC . . . 235
6.18 Poruka ID = 104 strukture packetOutMC . . . 235
6.19 Poruka ID = 105 strukture packetOutMC . . . 235
6.20 Poruka ID = 106 strukture packetOutMC . . . 236
6.21 Poruka ID = 107 strukture packetOutMC . . . 236
6.22 Poruka strukture packetInAS . . . 236
6.23 Poruka ID = 1 strukture packetInOut . . . 237
6.24 Poruka ID = 7 strukture packetInOut . . . 237
6.25 Poruka ID = 8 strukture packetInOut . . . 237
6.26 Poruka ID = 9 strukture packetInOut . . . 237
6.27 Specifikacija delova letelice QFLY-250 . . . 241
6.28 Specifikacija delova letelice LABFLY-RM01 . . . 246
6.29 Specifikacija delova letelice REMILP-01 . . . 259

7.1 Izračunata kašnjenja sa slika ekrana tokom ispitivanja kašnjenja telemetrije preko TCP-a lokalno . . . 329
7.2 Izračunata kašnjenja sa slika ekrana tokom ispitivanja kašnjenja telemetrije preko TCP-a i radio modula . . . 331
7.3 Izračunata kašnjenja tokom ispitivanja kašnjenja video-snimka preko WebRTC-a . . . 337
7.4 Izračunata kašnjenja tokom ispitivanja kašnjenja video-snimka preko RTSP-a . . . 338
7.5 Izračunata kašnjenja tokom ispitivanja kašnjenja video-snimka preko RTSP-a i LAN kamere na letelici . . . 340

Spisak oznaka i skraćenica

α	Uglovi otklona donjih poluga
$\mathbf{0}$	Nula matrica
$\boldsymbol{\alpha}^*$	Vektor željenih uglova otklona donjih poluga
$\boldsymbol{\beta_s}$	Vektor uglova između osa za preslikavanje tačaka i x ose
$\boldsymbol{\beta_{jp}}$	Uglovi između osa za vezu zglobova platforme i x_p ose
$\boldsymbol{\beta_{ll}}$	Uglovi između donjih poluga i x_b ose
$\boldsymbol{\delta_b}$	Uglovi između vektora $\boldsymbol{O_bB_i}$ i x_b ose
$\boldsymbol{\delta_p}$	Uglovi između vektora $\boldsymbol{O_pP_i}$ i x_p ose
$\boldsymbol{\epsilon_p}$	Vektor ugaonog ubrzanja platforme
$\boldsymbol{\epsilon_{ul,i}}$	Ugaono ubrzanje gornje poluge
$\boldsymbol{\lambda_b}$	Uglovi odstupanja ugla sfernog zgloba donje poluge i gornje poluge od referentnog
$\boldsymbol{\lambda_p}$	Uglovi odstupanja ugla sfernog zgloba platforme i gornje poluge od referentnog
$\boldsymbol{\lambda}$	Lagranževi množioci
$\boldsymbol{\nu_{ll}}$	Uglovi kojima se definiše referentna orijentacija donjih poluga
$\boldsymbol{\omega_{\alpha_i}}$	Vektor ugaonih brzina aktuatora
$\boldsymbol{\omega_p}$	Ugaona brzina platforme
$\boldsymbol{\omega_{ul,i}}$	Vektor ugaonih brzina gornje poluge
$\boldsymbol{\sigma_l}$	Vektor za promenu smera obrtnih aktuatora
$\boldsymbol{\sigma}$	Tenzor napona
$\boldsymbol{\varepsilon}$	Vektor deformacija
$\boldsymbol{A_i}$	Središte sfernih zglobova donje poluge
$\boldsymbol{a_i}$	Vektor položaja tačke A_i u koordinatnom sistemu baze
$\boldsymbol{A_r}$	Rodrigova matrica transformacije

$\boldsymbol{a_{0i}}$ Vektor položaja tačke A_{0i} u koordinatnom sistemu baze

$\boldsymbol{a_p}$ Vektor translatornog ubrzanja platforme

$\boldsymbol{B_i}$ Koordinate obrtnih zglobova baze

$\boldsymbol{b_i}$ Vektor položaja tačke B_i u koordinatnom sistemu baze

$\boldsymbol{B}$ Matrica deformacija

$\boldsymbol{C}$ Matrica dopunskih ograničenja

$\boldsymbol{c}$ Matrica elastičnih karakteristika materijala

$\boldsymbol{D_e}$ Pomeranja elementa u globalnom koordinatnom sistemu

$\boldsymbol{d_e}$ Vektor pomeranja čvorova elemenata

$\boldsymbol{d_i}$ Vektor pomeranja pojedinačnih čvorova

$\boldsymbol{D_r}$ Redukovan vektor pomeranja

$\boldsymbol{D}$ Vektor pomeranja svih čvorova

$\boldsymbol{F_e}$ Čvorne sile elementa u globalnom koordinatnom sistemu

$\boldsymbol{f_e}$ Totalni vektor čvorne sile

$\boldsymbol{F_r}$ Redukovan vektor čvornih sila

$\boldsymbol{F_x}$ Spoljašnje opterećenje mehanizma

$\boldsymbol{F}$ Vektor svih čvornih sila

$\boldsymbol{i_{jpb}}$ Vektor $\boldsymbol{i_{jp}}$ u koordinatnom sistemu baze

$\boldsymbol{i_{jp}}$ Jedinični vektor ose za vezu sfernog zgloba platforme

$\boldsymbol{i_{ll0}}$ Jedinični vektor ose za vezu sfernog zgloba donje poluge u početnom položaju

$\boldsymbol{i_{ll}}$ Jedinični vektor ose za vezu sfernog zgloba donje poluge

$\boldsymbol{i_m}$ Jedinični vektor obrtne ose aktuatora

$\boldsymbol{i_{ul}}$ Jedinični vektor gornjih poluga

$\boldsymbol{I}$ Jedinična matrica

$\boldsymbol{I}$ Matrica inercije tereta

$\boldsymbol{J_\alpha}$ Jakobijeva matrica unutrašnjih koordinata

$\boldsymbol{J_x}$ Jakobijeva matrica spoljašnjih koordinata

$\boldsymbol{J}$ Jakobijeva matrica mehanizma

$\boldsymbol{K_e}$ Matrica krutosti elementa u globalnom koordinatnom sistemu

$\boldsymbol{k_e}$ Matrica krutosti elementa u lokalnom koordinatnom sistemu elementa

$\boldsymbol{K_r}$	Redukovana matrica krutosti
$\boldsymbol{K}$	Globalna matrica krutosti
$\boldsymbol{L_i}$	Vektor od P_i do B_i
$\boldsymbol{L}$	Operator diferenciranja
$\boldsymbol{M_e}$	Masena matrica elementa u globalnom koordinatnom sistemu
$\boldsymbol{m_e}$	Masena matrica elementa u lokalnom koordinatnom sistemu elementa
$\boldsymbol{M_m}$	Potrebni obrtni momenti aktuatora
$\boldsymbol{M}$	Globalna masena matrica
$\boldsymbol{N}$	Matrica funkcija oblika
$\boldsymbol{P_i}$	Središte sfernih zglobova platforme
$\boldsymbol{p_i}$	Vektor položaja tačke P_i u koordinatnom sistemu platforme
$\boldsymbol{Q}$	Vektor dopunskih čvornih opterećenja
$\boldsymbol{R}$	Matrica rotacije platforme u odnosu na bazu
$\boldsymbol{T}$	Matrica transformacije
$\boldsymbol{T}$	Vektor translacije platforme u odnosu na bazu
$\boldsymbol{U}$	Vektor translatornih pomeranja
$\boldsymbol{v_p}$	Translatorna brzina platforme
$\boldsymbol{x}$	Vektor spoljašnjih koordinata
η	Indikator spretnosti robota
γ_b	Uglovi između vektora $\boldsymbol{O_bB_i}$ i ose za preslikavanje tačaka B_i
γ_p	Ugao između vektora $\boldsymbol{O_pP_i}$ i ose za preslikavanje tačaka P_i
λ_{max}	Maksimalni konični ugao sfernog zgloba
μ_{jp}	Ugao ugradnje ose za vezu sfernog zgloba platforme
μ_{ll}	Ugao ugradnje ose za vezu sfernog zgloba donje poluge
ν	Poasonov koeficijent
$\overline{\boldsymbol{U}}$	Aproksimacija pomeranja
Π	Potencijalna energija
Π_t	Totalna potencijalna energija
ψ	Ugao skretanja
ρ	Gustina tela

τ_{jp} Ugao odstupanja orijentacije gornjih zglobova od referentne orijentacije

τ_{ll} Ugao odstupanja orijentacije donje poluge od referentne

θ Ugao propinjanja

ε Ugao između obrtne ose aktuatora i vertikalne ravni

φ Ugao valjanja

A Površina poprečnog preseka

a Dužina donje poluge od B_i do A_i

A_{lat} Matrica sistema lateralne dinamike

A_{long} Matrica sistema longitudinalne dinamike

B_{lat} Matrica ulaza lateralne dinamike

B_{long} Matrica ulaza longitudinalne dinamike

D Spoljni prečnik kružne cevi

E Modul elastičnosti

f_b Zapreminska sila

f_s Površinska sila

F_A Dozvoljena aksijalna sila reduktora

F_a Aksijalno opterećenje reduktora

F_{Rx} Dozvoljena radijalna sila reduktora za poznati položaj x napadne tačke sile

F_R Dozvoljena radijalna sila reduktora

F_r Radijalno opterećenje reduktora

G Modul klizanja

I_y Aksijalni moment inercije za osu y

I_z Aksijalni moment inercije za osu z

J Polarni moment inercije

k Indeks iteracije

L Lagranžov funkcional

L_e Dužina elementa

l_i Potrebna dužina noge mehanizma

n_d Broj čvorova koji formiraju element

R_b Poluprečnik baze od O_b do B_i

R_p Poluprečnik platforme od O_p do P_i

s Dužina gornje poluge od A_i do P_i

T Kinetička energija

t Debljina zida kružne cevi

u Pomeranje po x osi

V Zapremina tela

v Pomeranje po y osi

V_e Zapremina elementa

W Hamiltonovo dejstvo

w Indikator manipulabilnosti robota

w Pomeranje po z osi

W_f Rad spoljašnjih sila

y_{of} Vrednost funkcije prilagođenosti

z_0 Početna z koordinata platforme

6-RSS 6 Revolute-Spherical-Spherical

6-RUS 6 Rotary-Universal-Spherical

ADC Analog-to-digital converter

AHRS Attitude and Heading Reference System

AM Amplitude Modulation

AR Augmentovana Realnost

ASIC Application-Specific Integrated Circuits

BPSK Binary Phase Shift Keying

CAE Computer Aided Engineering

CAM Computer Aided Manufacturing

CAN Controller Area Network

CGAL The Computational Geometry Algorithms Library

COFDM Coded Orthogonal Frequency Division Multiplexing

CPU Central Processing Unit

DAC Digital-to-analog converter

DC Direct Current

DCM Direction Cosine Matrix

DCV Direktorat civilnog vazduhoplovstva Republike Srbije

DOF Digitalni ortofoto

EKF Extended Kalman Filter

EPP Ekspandirani polipropilen

FEC Forward Error Correction

FFF Fused Filament Fabrication

FM Frequency Modulation

FPGA Field-Programmable Gate Array

FPV First-person view

FR Frekventni Regulator

FSK Frequency-shift keying

geo-ograda Geografska ograda

GIMP GNU Image Manipulation Program

GLB GL Transmission Format Binary file

glTF Graphics Library Transmission Format

HAL Hardware Abstraction Layer

HDL Hardware Description Language

HITL Hardware in the Loop

HTTP Hypertext Transfer Protocol

I2C Inter-Integrated Circuit

IP Internet Protocol

LAN Local-Area Network

MCAD Mechanical Computer Aided Design

MIMO Multiple-In Multiple-Out

MSLA Masked Stereolithography

NASA National Aeronautics and Space Administration

OFDM Orthogonal Frequency Division Multiplexing

PCB Printed Circuit Board

PWM Pulse Width Modulation

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RAM Random-Access Memory

RSSI Received Signal Strength Indicator

RTCP Real-Time Control Protocol

RTP Real-Time Transport Protocol

RTSP Real-Time Streaming Protocol

SBC Single Board Computer

SDR Software-Defined Radio

SITL Software in the Loop

SPI Serial Peripheral Interface

SSL Secure Sockets Layer

SVG Scalable Vector Graphics

TCP Transmission Control Protocol

TLS Transport Layer Security

UART Universal Asynchronous Receiver-Transmitter

UDP User Datagram Protocol

URL Uniform Resource Locator

USB Universal Serial Bus

VR Virtuelna Realnost

WebGL Web Graphics Library

Wi-Fi Wireless Fidelity

WLAN Wireless Local-Area Network

WWW World Wide Web

Spisak kodova

3.1 plot2D.jsl . . . 73
6.1 Primer unije za transformaciju . . . 231
6.2 Struktura packetOutSB . . . 232
6.3 Struktura packetInSB . . . 232
6.4 Struktura packetInMC . . . 233
6.5 Struktura packetOutMC . . . 234
6.6 Struktura packetInAS . . . 236
6.7 Struktura packetInOut . . . 237
6.8 Podešavanje Cygwin okruženja . . . 264
6.9 Preuzimanje ArduPilot koda . . . 264
6.10 Kompajliranje ArduCopter koda za ploču Pixhawk4 . . . 264
6.11 Blink.cpp . . . 264
6.12 wscript za Blink.cpp primer . . . 265
6.13 Komande za kompajliranje Blink.cpp primera . . . 265
6.14 Jedna komanda za kompajliranje Blink.cpp primera . . . 265
6.15 readArdupilotLog.jsl . . . 270
6.16 showFlightPath.jsl . . . 271
6.17 Struktura MAVLink V2 paketa . . . 272
6.18 Definicija nove BASIC_REMILP poruke . . . 278
6.19 Slanje BASIC_REMILP poruke iz UserCode.cpp . . . 278
7.1 checkSolver.jsl . . . 315
7.2 telemetryServer.jsl . . . 326
7.3 telemetryClient.jsl . . . 327
7.4 video2randFrames.jsl . . . 328
7.5 test_videocall_client.jsl . . . 335
7.6 test_videocall_server.jsl . . . 336
A.1 SOLVER_MP_6RSS.jsl . . . 373
B.1 package.json . . . 393
B.2 binding.gyp . . . 393
B.3 solver-mp-6rss.h . . . 394
B.4 solver-mp-6rss.cpp . . . 396
B.5 math-functions.h . . . 409
B.6 print-functions.h . . . 413
C.1 PLOTTER_MP_6RSS.jsl . . . 417
D.1 solveInverseKinematics.jsl . . . 431
E.1 solveForwardKinematics.jsl . . . 433

F.1 calculateVelocitiesAndAccelerations.jsl . . . 435
G.1 tensionShape.jsl . . . 441
G.2 bendingShapeXY.jsl . . . 442
G.3 bendingShapeXZ.jsl . . . 443
H.1 SOLVER_FEM_MP_6RSS.jsl . . . 445
I.1 solveInverseStaticFEM.jsl . . . 451
J.1 SPACE_SEARCH.jsl . . . 455
J.2 example_SpaceSearch.jsl . . . 461
K.1 findWorkspaceParallel.jsl . . . 463
L.1 boundary3D.js . . . 465
L.2 binding.gyp . . . 465
L.3 alpha-shape-3d.h . . . 466
L.4 alpha-shape-3d.cpp . . . 469
M.1 leversMinDistance.jsl . . . 489
N.1 checkJoints() za fiksni ugao ugradnje sfernih zglobova . . . 495
O.1 analyticStructuralAnalysis.jsl . . . 497
P.1 RCMIGA.jsl . . . 501
P.2 example_RCMIGA.jsl . . . 514
Q.1 optimizationRCMIGA.jsl . . . 517
Q.2 optimizationParallelRCMIGA.jsl . . . 521
R.1 MP_6DOF_M.jsl . . . 527
R.2 MP_6DOF_XXX.jsl . . . 528
S.1 MP-6DOF-M_CB.ino . . . 529
T.1 PRDC_SCBP_CODEC.js . . . 537
T.2 PRDC_SCBP.js . . . 541
T.3 PRDC_SCBP_NODE.js . . . 544
T.4 PRDC_SCBP_WEB.js . . . 546
T.5 example_PRDC_SCBP.js . . . 549
T.6 PRDC_SCBP_CLASS.m . . . 550
T.7 example_PRDC_SCBP.m . . . 553
T.8 PRDC_SCBP.h . . . 554
T.9 PRDC_SCBP.cpp . . . 556
T.10 example_SCBP.ino . . . 558
U.1 PRDC_PACKET.js . . . 561
V.1 Init.py . . . 565
V.2 InitGui.py . . . 565
V.3 check_ready.py . . . 566
V.4 resource_utils.py . . . 566
V.5 tcp_server.py . . . 567
V.6 message_parser.py . . . 569
V.7 eapi_prefs.ui . . . 572
V.8 MakeBeams.py . . . 574
V.9 MakeFusion.py . . . 575
V.10 MakeNodes.py . . . 575
V.11 showMsg.py . . . 576
W.1 createModelFreeCAD.jsl . . . 579

X.1 MP-6DOF-XXX_CB.ino . . . 583
X.2 MP-6DOF-XXX_CB config.h . . . 593
X.3 MP-6DOF-XXX_CB build_opt.h . . . 596
X.4 MP-6DOF-XXX_CB BASIC_IO.h . . . 596
X.5 MP-6DOF-XXX_CB BASIC_IO.cpp . . . 597
X.6 MP-6DOF-XXX_CB MECHANISM_6RSS.h . . . 599
X.7 MP-6DOF-XXX_CB MECHANISM_6RSS.cpp . . . 601
X.8 MP-6DOF-XXX_CB MISSION_CONTROLLER.h . . . 607
X.9 MP-6DOF-XXX_CB MISSION_CONTROLLER.cpp . . . 609
X.10 MP-6DOF-XXX_CB MOTION_PLANNER.h . . . 612
X.11 MP-6DOF-XXX_CB MOTION_PLANNER.cpp . . . 613
X.12 MP-6DOF-XXX_CB SERVO_BOARD.h . . . 614
X.13 MP-6DOF-XXX_CB SERVO_BOARD.cpp . . . 615
X.14 MP-6DOF-XXX_CB math_helper.h . . . 619
X.15 MP-6DOF-XXX_CB math_helper.cpp . . . 620
X.16 MP-6DOF-XXX_CB print_helper.h . . . 624
X.17 MP-6DOF-XXX_CB print_helper.cpp . . . 625
X.18 MP-6DOF-XXX_CB debug.h . . . 626
X.19 MP-6DOF-XXX_CB regression_test.h . . . 628
X.20 MP-6DOF-XXX_SB.ino . . . 631
X.21 MP-6DOF-XXX_SB config.h . . . 634
X.22 MP-6DOF-XXX_SB build_opt.h . . . 636
X.23 MP-6DOF-XXX_SB BASIC_IO.h . . . 636
X.24 MP-6DOF-XXX_SB BASIC_IO.cpp . . . 637
X.25 MP-6DOF-XXX_SB ANGLE_CONTROLLER.h . . . 640
X.26 MP-6DOF-XXX_SB ANGLE_CONTROLLER.cpp . . . 641
X.27 MP-6DOF-XXX_SB ANGLE_SENSOR.h . . . 643
X.28 MP-6DOF-XXX_SB ANGLE_SENSOR.cpp . . . 644
X.29 MP-6DOF-XXX_SB CONTROL_BOARD.h . . . 647
X.30 MP-6DOF-XXX_SB CONTROL_BOARD.cpp . . . 648
X.31 MP-6DOF-XXX_SB debug.h . . . 651
X.32 PRDC_VFD_ENC_EN600.h . . . 652
X.33 PRDC_VFD_ENC_EN600.cpp . . . 653
X.34 PRDC_AngleSensor.ino . . . 658
X.35 PRDC_TLE5012B.h . . . 660
X.36 PRDC_TLE5012B.cpp . . . 664
Y.1 flightGearSimulationUDP.jsl . . . 681
Z.1 ArdupilotLink.jsl . . . 687
Z.2 ArdupilotSitlLink.jsl . . . 695
Z.3 PFD.js . . . 704
Z.4 node-mavlink-wrapper package.json . . . 720
Z.5 node-mavlink-wrapper index.js . . . 721
AA.1 serverRTSP.jsl . . . 723
AA.2 server-rtsp-live555 package.json . . . 724
AA.3 server-rtsp-live555 binding.gyp . . . 725
AA.4 server-rtsp-live555.h . . . 726

AA.5 server-rtsp-live555.cpp . . . 729
AB.1 clientRTSP.jsl . . . 743
AB.2 RTSP_DEMUXER.js . . . 747
AB.3 DECODER_JS.js . . . 763
AB.4 decoder-ffmpeg package.json . . . 765
AB.5 decoder-ffmpeg binding.gyp . . . 765
AB.6 decoder-ffmpeg.h . . . 766
AB.7 decoder-ffmpeg.cpp . . . 767
AB.8 stream-decoder-ffmpeg package.json . . . 775
AB.9 stream-decoder-ffmpeg binding.gyp . . . 775
AB.10 stream-decoder-ffmpeg.h . . . 776
AB.11 stream-decoder-ffmpeg.cpp . . . 779
AB.12 stream-decoder-gst package.json . . . 796
AB.13 stream-decoder-gst binding.gyp . . . 796
AB.14 stream-decoder-gst.h . . . 797
AB.15 stream-decoder-gst.cpp . . . 799
AC.1 serveOculus.jsl . . . 811
AC.2 serveOculus index.html . . . 812
AD.1 showModel3D.jsl . . . 815
AE.1 solveInverseDynamicsOpenModelica.jsl . . . 821
AE.2 modelDinamikaParametarski_template.mo . . . 826

1

Uvod

> *"...ali najveća vrednost mog izuma će proizaći iz njegovog uticaja na ratovanje i naoružanje, jer će zbog svoje izvesne i neograničene destruktivnosti težiti da* ***uspostavi i održi trajni mir među narodima****." u patentu* ***Metoda i uređaj za upravljanje mehanizmom pokretnih plovila ili vozila*** *iz 1898. godine [1].*
>
> *– Nikola Tesla*[1]

Od samih početaka vazduhoplovstva, pored projektovanja, razvoja i proizvodnje letelica, veliki izazov bilo je sticanje i unapređivanje umeća pilotiranja letelicama. Nagli razvoj civilnog vazdušnog saobraćaja ubrzo je zahtevao mogućnost instrumentalnog letenja. Piloti su morali da budu u stanju da upravljaju letelicom u uslovima kada je vidljivost iz pilotske kabine toliko smanjena da let više nije bezbedan. Oslanjanje na sofisticirane instrumente i uređaje za automatizovanu kontrolu i navigaciju kompletno je promenilo pilotiranje, jer više nije bilo dovoljno pouzdati se samo u ljudska čula i iskustvo.

Sledeći korak u razvoju vazduhoplovstva bio je potpuno izmeštanje pilota iz letelice i uvođenje bespilotnih letelica, gde se daljinsko upravljanje može posmatrati kao oblik naprednog instrumentalnog letenja. Međutim, za razliku od klasičnog pilotiranja, kada pilot fizički upravlja letelicom i neposredno oseća njeno kretanje, operater (ili pilot) bespilotne letelice obično raspolaže ograničenim vizuelnim informacijama i osnovnom telemetrijom. Nedostatak realističnog čulnog doživljaja i prostorne orijentacije može dovesti do otežanog upravljanja i do povećanja rizika od pojave nepredviđenih situacija tokom leta koje je još teže rešiti daljinskim putem.

Upravo iz ovih razloga, javlja se potreba za naprednim sistemima koji daljinsko pilotiranje čine što sličnijim onom neposrednom, iz kabine letelice. Osnovna motivacija ovog istraživanja jeste potreba da se operateru bespilotne letelice pruža osećaj orijentacije i kretanja koji je uporediv sa onim koji bi imao kada bi se stvarno nalazio u letelici, što može značajno uticati na bezbednost, uspešnost i efikasnost izvršenja misija. Slični sistemi se koriste za obuku pilota i trening, ali ne i za daljinsko upravljanje.

Postavljeni problem koji je potrebno rešiti je multidisciplinaran. On najpre zahteva analizu bespilotnih letelica u cilju razumevanja da li one već imaju neophodne podsisteme koji će omogućiti da se njihovo kretanje saopšti operateru. To pre svega podrazumeva analizu električnog sistema, programa letelice i sistema za bežičnu komunikaciju sa operaterom. Zatim je potrebno razviti robotski mehanizam (što obuhvata i mehaniku i elektroniku i

[1] Nikola Tesla (10. jul 1856. - 7. januar 1943.) je bio srpski i američki pronalazač, inženjer elektrotehnike i mašinstva, najpoznatiji po svom doprinosu u projektovanju modernog sistema napajanja naizmeničnom električnom strujom.

programiranje) koji je u stanju da se kreće slično kao letelica i sa opterećenjem koje u ovom slučaju predstavljaju operater i pilotska kabina. Upravo je pilotska kabina i poslednji značajan element koji je potrebno razviti kako bi se omogućila interakcija između operatera i bespilotne letelice. Kako je jedan od elemenata sistema i čovek kao operater, značajno je razumevanje njegovih čulnih organa i načina na koji doživljava kretanje, ali i razumevanje toga kako i na koji način generiše upravljanje, a na osnovu svih informacija koje prima.

Neke bespilotne letelice (koje nemaju putnike) mogu da budu projektovane i tako da izvršavaju zadatke koji ne bi bili mogući da se pilot nalazi u letelici. Specijalno projektovani filteri kretanja se koriste kada bespilotna letelica treba da radi nešto što operater ne može fizički da podnese ili nešto što mehanizam za omogućavanje kretanja pilota ne može da ostvari. Pilot treba da bude u mogućnosti da kontroliše koliko zapravo oseća kretanje letelice, a u nekim situacijama da ovaj sistem može i da isključi, kako ne bi sam sistem predstavljao ograničenje za misiju letelice.

1.1 Naučni doprinos i očekivani rezultat disertacije

Naučni doprinos ove disertacije je **razvoj mehanizma platforme pristupačnog simulatora leta koji je optimizovan za simuliranje leta i koji zajedno sa ostalim podsistemima može da se primeni pri pilotiranju daljinski upravljanom letelicom u okviru povratne sprege. Ovako formirana povratna sprega sa simulatorom leta omogućava da pilot pri pilotiranju daljinski upravljanom letelicom ima približno isti osećaj kao da se nalazi u samoj letelici.**

Očekivani rezultat ove disertacije je **novi proizvod**, i to kompletan **sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi**, čiji su osnovni elementi simulator leta, pilotska kabina i daljinski upravljana letelica. I pored toga što se pri razvoju bespilotnih letelica uvek teži ka tome da se u maksimalnoj mogućoj meri smanji potreba za operaterom, postoje određene situacije u kojima operater preuzima kontrolu nad bespilotnom letelicom. Zbog toga će naučni doprinos ove disertacije ostati aktuelan i u narednom periodu.

1.2 Struktura disertacije

U poglavlju 1 je dat uvod, istaknut je naučni doprinos disertacije kao i njen očekivani rezultat. U nastavku je u poglavlju 2 data naučna zamisao disertacije sa povodom i tokom istraživanja. U ovom poglavlju je prikazano i idejno rešenje sistema koji je predmet istraživanja.

U poglavlju 3 ove disertacije je dat pregled stanja relevantne tehnike i prikaz dosadašnjih relevantnih istraživanja u vezi sa naučnom zamisli disertacije. Pregled postojeće naučne literature i ostvarenih rezultata, praktičnih, naučnih, tehničkih i drugih dostignuća. Ovaj pregled je prilično obiman zbog multidisciplinarnosti disertacije, pojedinačne analize svih elemenata sistema i svega onoga što je bilo potrebno tokom dugogodišnjeg istraživanja, a u cilju toga da zainteresovanom čitaocu pruži potrebnu transparentnost istraživanja i omogući reproduktivnost. Analizirani su simulatori leta, dat je pregled i izvršen je odabir metode za proračun optimalne geometrije. Za bespilotne letelice je dat istorijski osvrt, zatim njihove primene, klasifikacija kao i prikaz postojećih domaćih bespilotnih letelica. Dat je pregled značajnih standarda i važećih zakona. Zatim su analizirani alati za trodimenzionalno modeliranje i kompjutersku grafiku koji su neophodni za realizaciju disertacije. Prikazani su kompozitni materijali, 3D štampanje i skeniranje kao i drugi proizvodni procesi i obrade

materijala koji se koriste za izradu elemenata sistema. Analizirane su štampane ploče, mikrokontroleri i komunikacije, bežični prenos podataka i programiranje digitalnih procesora. Zatim je urađen pregled dostupnih biblioteka za manipulaciju i obradu video signala kao i analiza mogućnosti upotrebe sistema za virtuelnu i proširenu realnost. Na kraju ovog poglavlja je prikazan program `JSLAB`, kao okruženje za numeričke proračune, koji se pretežno koristi u realizaciji disertacije.

U poglavlju 4 je prikazano određivanje geometrijskih parametara i performansi mehanizma simulatora leta za sistem. Simulator leta je drugi osnovni element sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi koji upravo omogućava ovu povratnu spregu kretanja. Prikazane su različite moguće varijante mehanizma i različiti tipovi aktuatora koji se mogu koristiti. Za usvojeni tip mehanizma su prikazane analitičke i numeričke metode za rešavanje inverznog i direktnog kinematičkog problema kao i određivanje brzine i ubrzanja svih značajnih tačaka mehanizma. Prikazana je implementacija metode konačnih elemenata sa Lagranžovim množiocima za proračun elastičnosti konstrukcije i posledično greške položaja i orijentacije platforme, kao i za dimenzionisanje segmenata mehanizma i aktuatora. Kompletna metoda je detaljno prikazana i na kraju su dati rezultati njene implementacije. Da bi se uradio proračun optimalnih vrednosti geometrijskih parametara, prvo su prikazana rešenja za inverzni statički i dinamički problem u cilju određivanja potrebnih obrtnih momenata aktuatora, a zatim su prikazani i drugi indikatori performansi mehanizma kao što su maksimalni ostvarivi parametri i zapremina radnog prostora. Prikazan je novi algoritam pod nazivom *SpaceSearch* koji predstavlja numerički metod za efikasno određivanje radnog prostora mehanizma.

U poglavlju 5 je dat proračun optimalne geometrije mehanizma simulatora leta za sistem koji se istražuje kako bi sistem imao zadovoljavajuće performanse na kraju. Uređena je analiza i prikazana je implementacija algoritma za proračun optimalnog rešenja. Definisane su funkcije ograničenja i prilagođenosti kojima se vrši pronalaženje rešenja. Zatim su dati parametri kojima se podešava algoritam proračuna, postupak proračuna i na kraju rezultati proračuna.

Konačno, sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi čine simulator leta, pilotska kabina i bespilotna letelica, kao osnovni elementi sistema, i u poglavlju 6 je prikazana njihova integracija za potrebe eksperimentalnog ispitivanja. Prikazani su projektovanje i proizvodnja umanjenog modela kao i modela simulatora leta u stvarnoj veličini. U ovom poglavlju je prikazano i više različitih letelica koje su korišćene tokom istraživanja ovog sistema. Prikazan je neophodan razvoj električnog sistema i programa za integraciju bespilotne letelice, sistema za bežičnu komunikaciju i simulatora leta.

U poglavlju 7 su prikazana sva urađena numerička i eksperimentalna ispitivanja elemenata sistema i na kraju i celog sistema. Prvo su prikazana eksperimentalna ispitivanja na umanjenom prototipu simulatora leta. Parametarsko trodimenzionalno modeliranje i simulacije su korišćene za numeričku verifikaciju rada mehanizma. Eksperimentalno su utvrđene performanse bežične komunikacije kao i kašnjenja usled akvizicije, obrade i slanja podataka. Na kraju je izvršena eksperimentalna validacija celog sistema u stvarnoj veličini.

U poglavlju 8 je data diskusija kao i predlog pravaca za nastavak prikazanog istraživanja.

Poslednje poglavlje, pod brojem 9 predstavlja zaključak u kome je urađen pregled svega onoga što je urađeno u okviru ove disertacije. U ovom poglavlju je zaključeno da li su ostvareni ciljevi disertacije.

2

Naučna zamisao disertacije

Razvoj sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi zahteva rešavanje problema iz različitih oblasti. Pre svega, to su vazduhoplovstvo, a zatim i mehanika robota, pilotiranje vazduhoplovima, bežične komunikacije, akvizicija i obrada video-signala, razvoj senzora i instrumenata, proizvodnja (posebno od kompozitnih materijala), programiranje, izvori energije i propulzija, elektrotehnika, kao i biologija čoveka.

Da bi se ovakav sistem realizovao, neophodno je pomerati pilota tako da ima osećaj kao da se nalazi u samoj letelici, potrebno je osmisliti mehanizam koji omogućava takva kretanja, u realnom vremenu vršiti merenja na samoj letelici i rezultate merenja bežično prenositi do upravljačkog sistema mehanizma, zatim očitati novo upravljanje pilota i komande poslati letelici. Na ovaj način su definisani globalni zahtevi za projektovanje ovog sistema, pri čemu svaki od njih ima dodatne probleme na koje se mora obratiti pažnja.

Zahvaljujući razvoju, pre svega, kompozitnih materijala, mikro-elektro-mehaničkih senzora, mikroprocesora i električnih baterija veće gustine energije danas su široko rasprostranjene veoma pristupačne bespilotne letelice sa električnim pogonom. Cilj je razviti sistem za daljinsko pilotiranje, koji je kompatibilan sa ovakvim letelicama. Takođe, moguće je projektovanje novih različitih daljinski upravljanih letelica koje se prilagođavaju ovom sistemu za daljinsko pilotiranje. Pored aktuatora koji su zaduženi za otklon komandnih površina, letelica ima niz senzora, na osnovu kojih je moguće uspostaviti stabilan let, ali i na osnovu kojih je moguće poslati potrebne parametre kretanja letelice kako bi se ostvarila željena povratna sprega. Za funkcionisanje sistema je neophodno da se sa što manjim kašnjenjem i sa što većom brzinom osvežavanja podataka šalju pre svega trenutno ubrzanje, brzine i orijentacija letelice, ali i ostali podaci.

Primer potrebe za ovakvim sistemom se u poslednje vreme ogleda i u tome što se ljudi koji po prvi put koriste uređaje za daljinsko upravljanje letelicama iz prvog lica (komercijalno dostupni proizvodi prvenstveno namenjeni za zabavu), gledanjem u monitor ili korišćenjem specijalnih naočara, obično žale na neprijatnost, vrtoglavicu i mučninu.

Imajući u vidu sve prethodno, u ovoj disertaciji se polazi od sledećih osnovnih hipoteza:

- Pilotu daljinski upravljane letelice nije dovoljno da vidi sliku sa kamere na letelici kako bi formirao najbolje moguće upravljanje unutar svojih mogućnosti;
- Moguće je u realnom vremenu precizno vršiti merenje ubrzanja i orijentacije daljinski upravljane letelice u prostoru;
- Moguće je uspostaviti bežičnu komunikaciju između pilota i letelice;

- Neizbežno kašnjenje slike i kretanja simulatora leta u odnosu na stvarno kretanje letelice ne utiče na uspešnost pilotiranja;
- Izborom odgovarajućeg mehanizma i optimizacijom njegove geometrije je moguće dobiti adekvatan simulator leta;
- Genetski algoritam je moguće uspešno primeniti za složene nelinearne probleme;
- Moguće je formirati matematički model letelice i pilota kako bi se formiralo predupravljanje za simulator leta;
- Moguće je u realnom vremenu formirati upravljanje za simulator leta na osnovu informacija koje su bežičnom komunikacijom dobijene sa daljinski upravljane letelice.

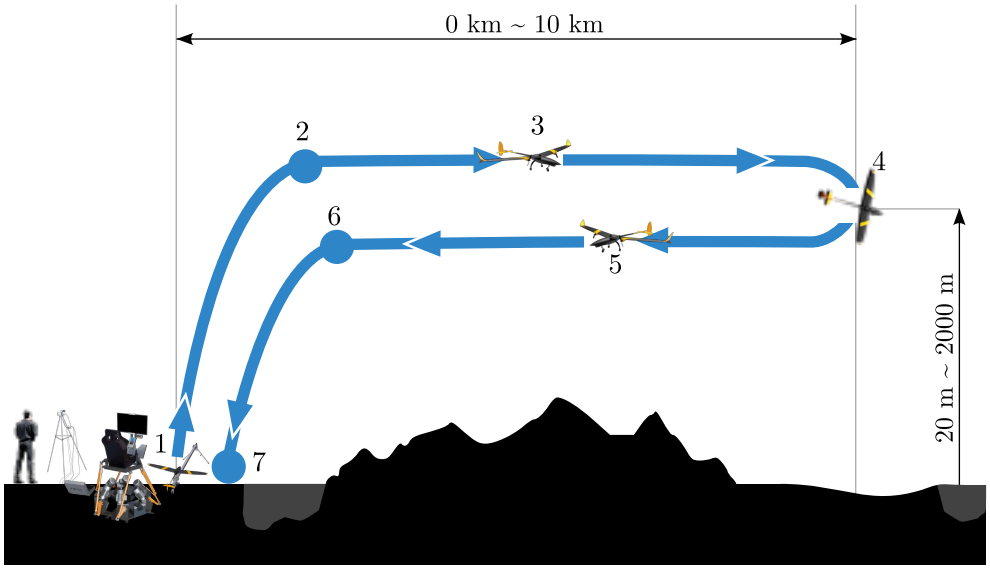
Tokom istraživanja primenjuju se sledeće naučne metode:

- Metoda pregleda postojeće naučne literature i ostvarenih rezultata, praktičnih, naučnih, tehničkih i drugih dostignuća, a u cilju sistematizacije postojećih istraživanja;
- Induktivne i statističke metode koriste se za izvođenje opštih zaključaka koji se odnose na trenutno stanje tehnike simulatora leta, sistema za bežičnu komunikaciju i daljinski upravljanih letelica;
- Preliminarno eksperimentalno istraživanje kojim se utvrđuju osnovne polazne vrednosti parametara i stanje postojećih sistema;
- Analitičke i numeričke metode za rešavanje inverznog i direktnog kinematičkog problema;
- Numeričke metode za određivanje radnog prostora mehanizma;
- Metod konačnih elemenata za proračun elastičnosti konstrukcije i posledično greške položaja i orijentacije platforme, kao i za proveru zamora poluga mehanizma;
- Razvijanje numeričkih modela i odabir odgovarajućih programskih jezika za implementaciju proračuna;
- Parametarsko trodimenzionalno modeliranje i simulacije u cilju verifikacije mehanizma;
- Razvoj i implementacija optimizacionih algoritama za potrebe optimizacije geometrije mehanizma simulatora leta;
- Eksperimentalno istraživanje kojim se utvrđuju performanse mehanizma, bežične komunikacije i daljinski upravljane letelice;
- Metoda komparacije, kojom će se omogućiti poređenje rezultata dobijenih primenom numeričke analize sa vrednostima dobijenim pri eksperimentalnim ispitivanjima.

Pretpostavlja se da su predložene metode potrebne i dovoljne da se potpuno jasno i temeljno predstavi i analizira sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

2.1 Osnovni elementi sistema

Prvi osnovni element sistema je daljinski pilotirana letelica. Na slici 2.1 je prikazana tipična misija za jednu ovakvu letelicu.



Slika 2.1 - Tipična misija za sistem

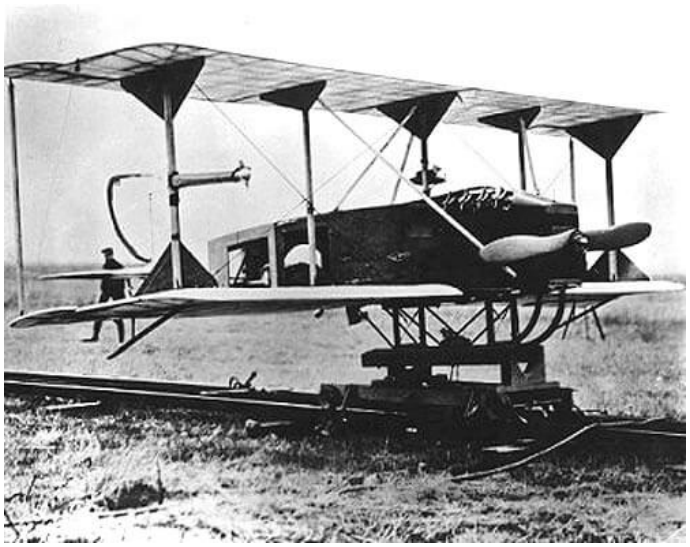
Na prethodnoj slici je označeno sedam ključnih tačaka koje predstavljaju različite faze leta. Ove faze leta se definišu na sledeći način:

- *Tačka 1* - Poletanje koje može biti prosto vertikalno poletanje (u slučaju letelica koje imaju tu mogućnost), poletanje bacanjem iz ruke, poletanje korišćenjem poletne staze, poletanje korišćenjem katapulta, poletanje pomoću dodatnog motora (najčešće raketnog motora) ili poletanje ispuštanjem sa druge letelice.
- *Tačka 2* - Prelazak iz faze poletanja u fazu krstarenja.
- *Tačka 3* - Krstarenje do cilja.
- *Tačka 4* - Izvršavanje zadatka na cilju, npr. izviđanje, fotografisanje, merenje, ispuštanje tereta, upotreba naoružanja ili drugo.
- *Tačka 5* - Krstarenje do mesta sletanja.
- *Tačka 6* - Prelazak iz faze krstarenja u fazu sletanja.
- *Tačka 7* - Sletanje koje može biti prosto vertikalno sletanje (u slučaju letelica koje imaju tu mogućnost), sletanje korišćenjem sletne staze, padobrana ili sletanje u mrežu.

Misija koja treba da se izvrši, kao prethodno prikazana, zapravo predstavlja skup ulaznih podataka u sistem koje direktno prima operater.

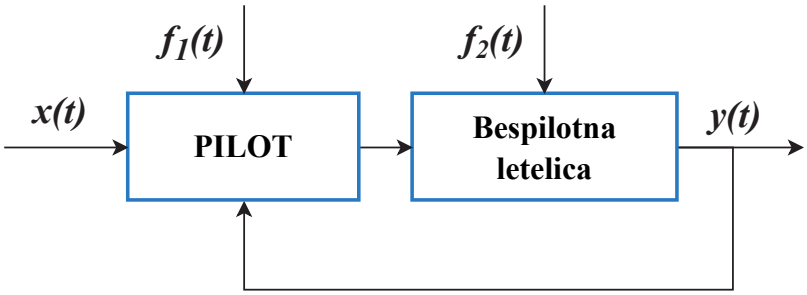
Razvoj bespilotnih letelica je započet samo 13 godina nakon prvog leta aviona, američka mornarica finansirala je projekat razvoja bespilotnog vazdušnog torpeda pod nazivom "Aerial Torpedo" [2] (letelica je prikazana na slici 2.2. Cilj mornarice je bio da se napravi mali, lagani avion koji bi mogao da se lansira bez pilota, da se navodi do mete, i detonira svoju bojevu glavu u tački dovoljno blizu da bude efikasan protiv ratnog broda. Ipak, smatra se da je za

bespilotne letelice i druge sisteme sa daljinskim upravljanjem najzaslužniji *Nikola Tesla* koji je u svom patentu "*Metoda i uređaj za upravljanje mehanizmom pokretnih plovila ili vozila*" iz 1898. godine po prvi put demonstrirao daljinsko upravljanje, u ovom konkretnom slučaju plovila. Već tada je Tesla bio svestan široke primene njegovog patenta što uključuje i primenu u vazduhoplovstvu. Bespilotne letelice će postojati sve dok budu imale određene prednosti u odnosu na letelice sa pilotom.



Slika 2.2 - Prva bespilotna letelica - "Aerial Torpedo" (preuzeto iz [2])

Kao što se može videti sa sledećeg blok dijagrama (slika 2.3), struktura sistema za daljinsko upravljanje bespilotnom letelicom u potpunosti odgovara sistemu u okviru koga se pilot nalazi u letelici što je i osnovni cilj simulirati sistemom.



Slika 2.3 - Osnovni dijagram sistema

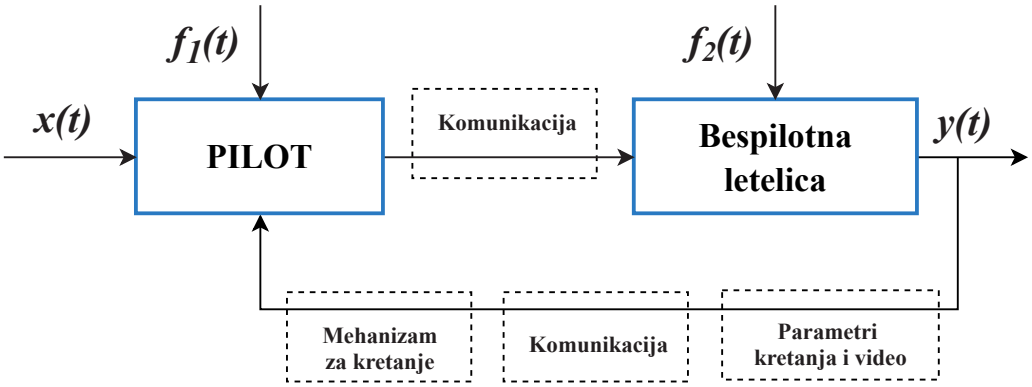
Na ovom blok dijagramu $x(t)$ je vektor ulaznih veličina, $y(t)$ je vektor odgovarajućih izlaznih veličina dok su $f_1(t)$ i $f_2(t)$ moguće poremećajne veličine, kao što su turbulentnost atmosfere, udari vetra, neželjeni otkloni komandi krmila, razni šumovi i slični uticaji.

Nakon izmeštanja pilota iz letelice, poželjno je da ga u određenoj meri zameni odgovarajući uređaj. Automatski uređaji koji u upravljačkom sistemu letelice preuzimaju određene funkcije koje obavlja čovek, kao pilot aviona, zovu se opštim imenom autopiloti. U savremenim letelicama autopiloti obično preuzimaju i razne druge funkcije koje čovek kao pilot letelice ne bi bio u mogućnosti da obavi. Pri pilotiranju od posebnog su značaja brzina i tačnost reagovanja kojom pilot ili autopilot mogu da odgovore na spoljašnje informacije, koje primaju preko odgovarajućih organa, kao što su oči, uši, žiroskopski uređaji i drugi senzori. Ukoliko je potrebno doneti neku odluku na osnovu nepotpunih ili kontradiktornih informacija dobijenih u različitim vremenima i u različitom obliku, čovek kao pilot nenadmašan je u

poređenju sa autopilotom. Ukoliko se pilot ne nalazi u samoj letelici gubi ovu prednost u odnosu na autopilot.

U toku leta čovek, kao pilot neke letelice, istovremeno prima razne informacije preko svojih čulnih organa, analizira ih i upoređuje, donosi odluke o potrebnim merama koje treba preduzeti pri upravljanju i upravlja letelicom tako da vrednosti parametara njenog kretanja što je moguće manje odstupaju od željenih vrednosti. Prema tome, pilot obavlja ustvari funkcije senzora, sabirača, komparatora, računara, pretvarača, pojačala i aktuatora. Iz ovoga sledi da pilot i letelica, kao celina, predstavljaju jedan dinamički sistem sa povratnom spregom.

Sada se na osnovni blok dijagram mogu dodati elementi koji nedostaju i koje je potrebno definisati (blok dijagram prikazan na slici 2.4). Prvi je svakako način komunikacije između pilota i letelice, dok je još uvek nepoznat i način realizovanja povratne sprege jer se pilot ne nalazi u letelici.



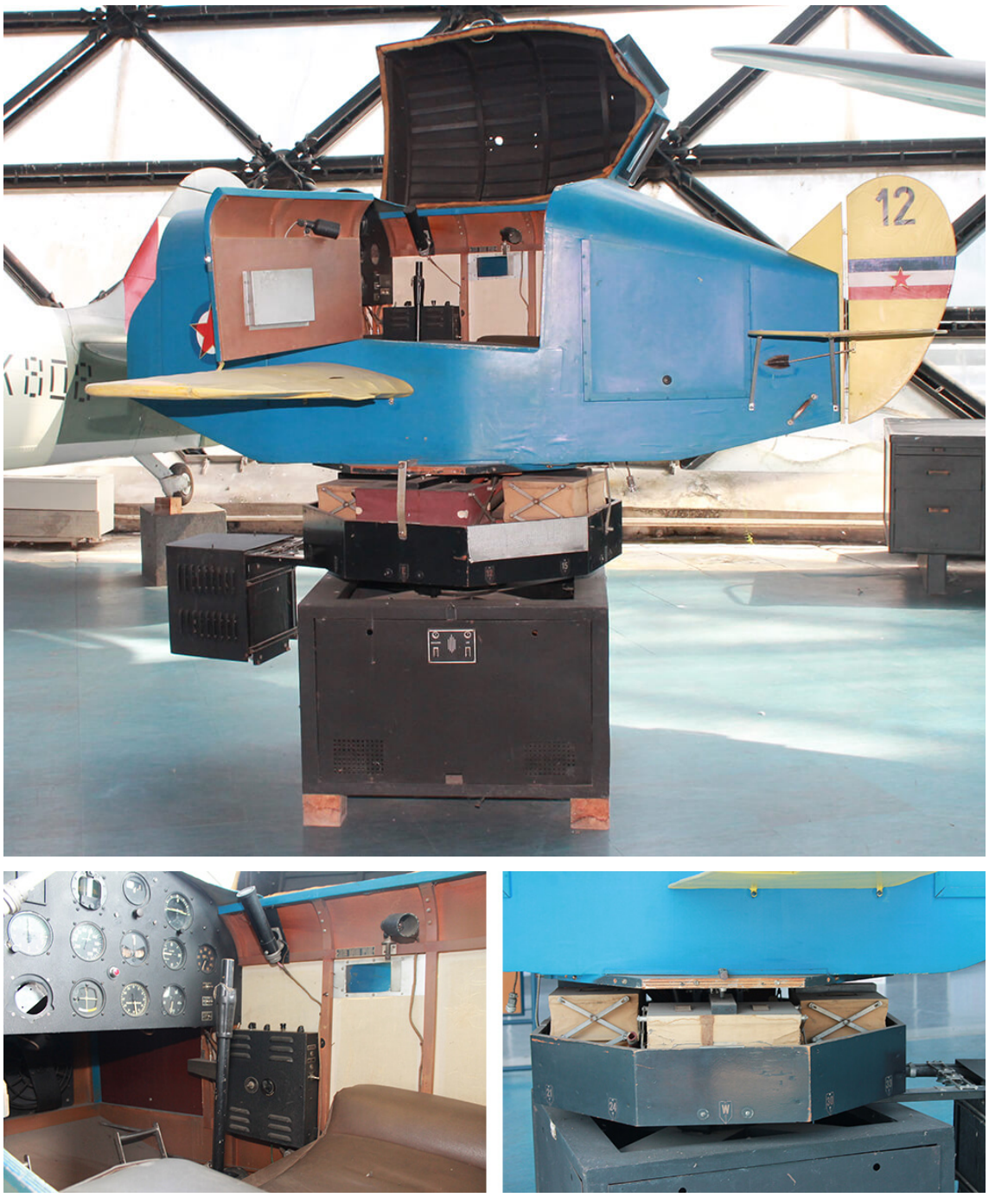
Slika 2.4 - Dijagram sistema sa neophodnim elementima

Do ostalih nepoznatih elemenata sistema se dolazi analizom bespilotnih letelica kao i mehanizama postojećih simulatora leta. Na slici 2.5 je prikazan jedan od prvih mehanizama iz 1910. godine pod nazivom *Antoinette simulator* [3] koji se koristio kako bi piloti vežbali pilotiranje, a da pri tome ne rizikuju svoj život, ideja je bila da pilot na ovom mehanizmu oseća barem približno ono što bi osećao u letelici prilikom leta.



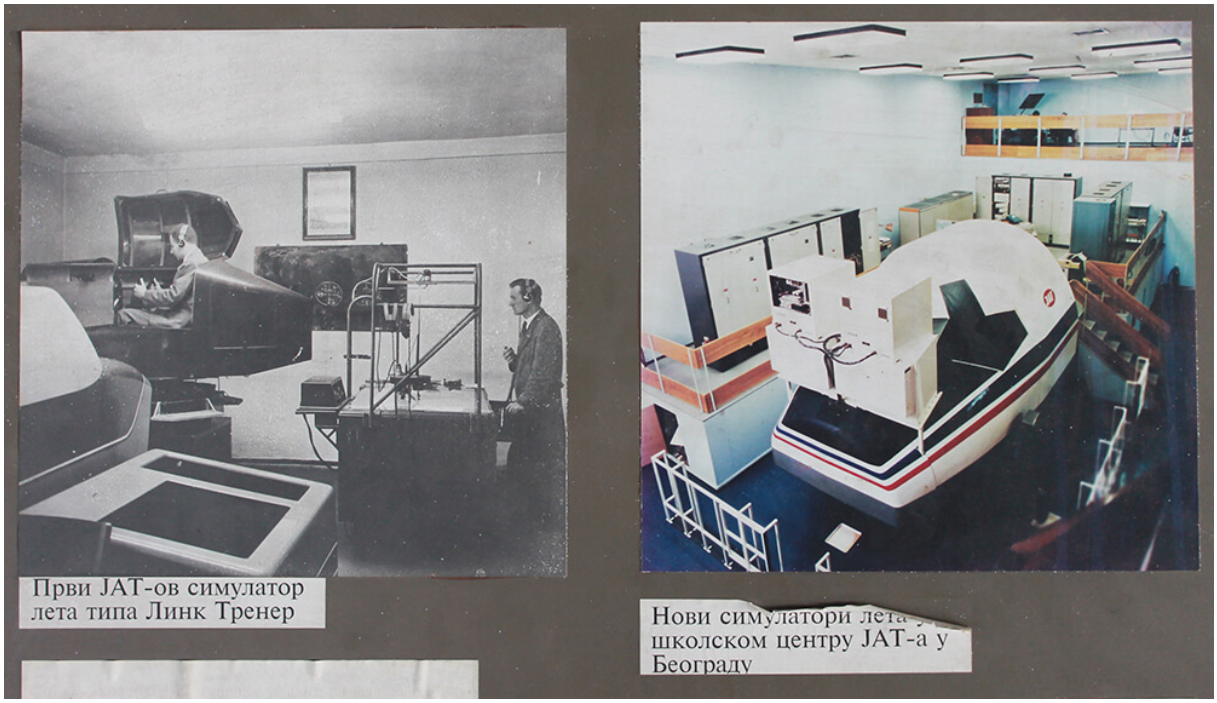
Slika 2.5 - Jedan od prvih poznatih simulator leta - Antoinette simulator (preuzeto iz [3])

U težnji da smanji rizik i troškove obuke, američki konstruktor *Edvin Link* (*Edwin A. Link*) konstruisao je 1930. godine (kada je podneo i patentni zahtev koji mu je odobren 1931. godine [4]) **simulator leta** koji predstavlja pilotsku kabinu postavljenu na specijalni mehanizam tako da može da se pozicionira i orijentiše u zavisnosti od zadate komande na isti način kao prava letelica, pri čemu i svi instrumenti odgovarajuće reaguju. On u svom patentu pod nazivom *"Kombinovani uređaj za obuku studenata avijatičara i za zabavu"* [4] navodi da je jedan od primarnih ciljeva njegovog pronalaska da obezbedi uređaj za obuku studenata avijatičara pomoću kojeg se oni podvrgavaju svim prirodnim osećajima letenja a da se u isto vreme uređaj stavlja pod njegovu kontrolu da upravlja njim tačno kao što se upravlja avionom. Takođe, ističe da korišćenjem njegovog uređaja student može lako i sa savršenom bezbednošću da nauči za sebe različite operacije koje se moraju izvršiti kako bi održao letelicu stabilnom i na odgovarajućoj putanji.



Slika 2.6 - Simulator leta Link Flight Trainer AN-2550-1 u Muzeju vazduhoplovstva u Beogradu

U *Muzeju vazduhoplovstva u Beogradu* se nalazi primerak simulatora leta *Link Flight Trainer AN-2550-1* (prikazan na slici 2.6) koji je dobijen od *Aerokluba "Franjo Kluz"* iz Zemuna 1992. godine. Prvih pet primeraka koji su stigli u Jugoslaviju nabavljeni su 1939. godine za potrebe obuke pilota „*Aeroputa*", a posle Drugog svetskog rata nabavljeni su u dva navrata, prvih 10 komada 1945. godine, a potom još 14 komada 1946. godine, ovi simulatori leta su dobili oznaku *ANT-18* [5]. U *Muzeju vazduhoplovstva* su takođe prikazane fotografije simulatora leta u školskom centru *JAT*-a u Beogradu (slika 2.7).



Slika 2.7 - Fotografije simulatora leta kompanije JAT

Moderni simulator leta služi za praktičnu obuku pilota pri različitim režimima leta. Teorijski, on treba da simulira sve slučajeve leta i sile koje deluju na pilota u toku leta. Od početka razvoja do danas simulatori su postali veoma složeni mehatronički sistemi, ali je njihova osnovna svrha, da omoguće pilotima da osete na ostatku tela ono što vide svojim očima, ostala ista. Kako bi se što bolje simulirao stvarni let potrebno je da pokretni deo simulatora ima najmanje šest upravljanih stepeni slobode kretanja u okviru definisanih granica. U odnosu na početke vazduhoplovstva, letelice su danas daleko složenije sa mnogo više instrumenata tako da su ovaj napredak letelica neizbežno morali da prate simulatori koji su se sve više vezivali za tačno određenu letelicu.

Pored daleko bezbednije obuke pilota za instrumentalno letenje, obuka na ovakvim simulatorima je višestruko jeftinija i brža nego kad bi se izvodila na pravim letelicama. Iz istih razloga i danas simulatori leta imaju nezamenljivu ulogu u obuci i procesu sertifikacije pilota. Danas je najčešći slučaj da je unutrašnjost simulatora prava kabina neke letelice i pored simuliranja kretanja pilot ima priliku da se navikne na stvarni raspored instrumenata i kontrola. Na slici 2.8 je prikazan moderni simulator leta za avion *Boeing 787*. Ulaskom u kabinu koja je identična kabini letelice kojom će zaista upravljati, pilot može nekoliko stotina puta da vežba sletanje na neki virtuelni aerodrom koji je praktično realna kopija stvarnog uz sve zakone fizike koji će dati odgovor da li je uspešno sleteo ili ne. Ponavljanjem velikog broja realnih scenarija koji ga mogu zateći tokom stvarnog leta pilot steče potrebnu sigurnost i samopouzdanje.



Slika 2.8 - Moderni simulator leta za avion Boeing 787 (preuzeto iz [6])

Treba spomenuti i to da simulatori leta mogu biti korisni prilikom projektovanja i razvoja novih letelica, jer omogućavaju kvalitetnije projektovanje kabine, kao i validaciju različitih softverskih rešenja. Iako su simulatori leta nezamenljivi za obuku klasičnih pilota, oni se i dalje ne koriste za obuku ili upravljanje daljinski upravljanim letelicama.

Trenutno se u svetu praktično svaki simulator leta sa šest stepeni slobode kretanja izvodi kao mehanizam poznat kao Stjuartova platforma, koja je u osnovi paralelni mehanizam (mehanizam sa zatvorenim kinematskim lancem). Ovaj mehanizam i njegova geometrija se pažljivo proračunavaju i dimenzionišu se njegovi aktuatori za potrebe simuliranja leta.

Na osnovu predmeta ovog istraživanja jasno je da je potrebno definisati detaljnu proceduru neophodnu za proračun optimalne geometrije mehanizma modernog simulatora leta, kako bi se nakon proračuna dobila mašina sa izuzetnim mogućnostima u pogledu simuliranja leta.

Kao što se vidi na slici 2.9, razvoj mehanizma simulatora leta započinje od zahteva koje taj mehanizam treba da ispuni, a to je u ovom slučaju da ima 6 stepeni slobode kretanja kojim je moguće simulirati let, zatim se vrši izbor kinematskog modela, proračun optimalnih vrednosti njegovih geometrijskih parametara i na kraju materijalizacija. Svaki od ovih koraka je veoma značajan. Loše postavljanje zahteva može rezultirati time da nije moguće praktično simulirati sve ono što je zadatkom definisano, loš odabir kinematskog modela može rezultirati skupim sistemom ili sistemom koji ne može da zadovolji sve postavljane zahteve i na kraju **sve što se zamisli mora da bude proizvodljivo, jer je jedini pravi rezultat mašina koja je materijalizovana i koja kao takva služi čoveku**.

Danas je cena simulatora leta uporediva sa cenom letelica za koju se simulator leta nabavlja. Zbog ovako visokih cena trenutno nije ekonomski opravdano adaptirati postojeće sisteme za primenu u oblasti daljinskog pilotiranja, sa druge strane, zbog postojanja veoma pristupačnih daljinski upravljanih letelica, sistem za njihovo daljinsko upravljanje takođe mora da bude pristupačan. Kao što je u disertaciji i prikazano, tokom čitavog perioda razvoja sistema vođeno je računa da je konačni proizvod proizvodljiv i ekonomski pristupačan.



Slika 2.9 - Razvoj mehanizma simulatora leta - od ideje do materijalizacije

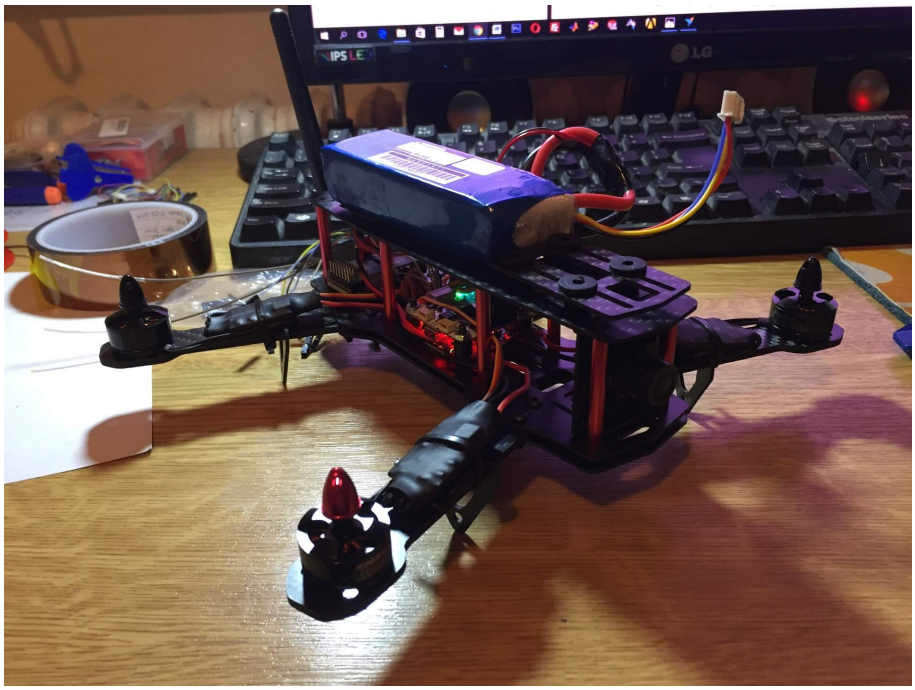
Treći osnovni element sistema, pored bespilotne letelice i simulatora leta, je pilotska kabina koja treba da omogući pilotu interakciju sa bespilotnom letelicom posredstvom bežične komunikacije. Trenutno ne postoje opšte prihvaćena rešenja pilotske kabine za daljinsko upravljanje letelicama sa parametrima kretanja letelice u povratnoj sprezi jer je ovo novi sistem.

2.2 Povod i tok istraživanja

Povod za istraživanje je omogućavanje preciznijeg, kvalitetnijeg i za pilote prirodnijeg daljinskog upravljanja letelicama, omogućavanje upravljanja kada pilot nema letelicu u vidokrugu ili kada izgubi video-signal, smanjenje rizika i troškova obuke i usavršavanja pilota, smanjenje broja udesa koji mogu biti fatalni po letelicu i njene podsisteme, kao i povećanje pouzdanosti izvršavanja misija.

Rad na projektima koji su kasnije rezultirali radom na ovoj disertaciji je započet 2016. godine i 2017. godine.

- Prvi projekat je zapravo samostalna izrada multikoptera sa četiri rotora i prečnikom elisa od 5 inča (prikazan na slici 2.10). Cilj ovog projekta je bio savladavanje tehnologije izrade letelice korišćenjem komercijalno dostupnih komponenti sa tržišta. Po završetku izrade ove letelice ona je postala platforma za dalja istraživanja.



Slika 2.10 - Samostalna izrada prvog multikoptera

- Drugi projekat je komercijalni projekat za pilot epizodu televizijske serije u produkciji jedne od najvećih američkih kompanija u ovoj industriji. Za potrebe ovog projekta je bilo potrebno napraviti simulator leta svemirskog broda, koji je prikazan na slici 2.11, sa jednim stepenom slobode (valjanje). Projekat je realizovan sa dva hidraulična cilindra, hidrauličnim agregatom sa ručnim razvodnikom i konstrukcijom sa jednim stepenom slobode koji je upravljan pomoću hidrauličnih cilindara. U toku istraživanja mehanizama za ovu potrebu se prvi put javila ideja o izradi simulatora leta sa mehanizmom baziranim na *Stjuartovoj platformi*.



Slika 2.11 - Upotreba simulatora leta svemirskog broda na setu televizijske serije

Daljim unapređenjem ova dva projekta i konačno njihovim spajanjem je nastala ideja da se radi na istraživanju u pravcu ove disertacije.

U narednim godinama je konstantno rađeno na stvaranju sistema koji omogućava daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi. Prikupljana su finansijska sredstva za samostalnu realizaciju ovog sistema i paralelno su vršeni mnogi eksperimenti kako bi se utvrdilo da li je moguće realizovati jedan takav sistem sa dostupnom tehnikom i tehnologijom. Do konačnog sistema se došlo nakon nekoliko iteracija.

Veliki deo vremena u godinama koje su obeležile razvoj ovog sistema je utrošeno zbog strateške odluke da se konačno sistem ne oslanja na programe zatvorenog koda kao što su `SOLIDWORKS`, `CATIA V5` ili `MATLAB`. Do ove odluke je došlo zbog izuzetno visoke cene ovih programa i nepoznatog načina na koji su implementirane njihove funkcionalnosti. Proizvođači ovih ali i nekih drugih programa privlače studente i mlade istraživače da koriste njihove programe tako što im daju besplatne verzije ali po isticanju tih licenci studenti najčešće ili kompletno ostaju bez mogućnosti da koriste ove programe ili ih dalje mogu koristiti samo sa određenim ograničenjima. To je u suprotnosti sa potrebom za slobodnim deljenjem znanja i omogućavanjem maksimalne pristupačnosti istraživanja. Nakon ove odluke se prelazi na korišćenje izuzetnih programa otvorenog koda (kao što su `FreeCAD` i `GNU Octave`) kojima je praktično moguće realizovati iste zadatke za koje su prethodno korišćeni popularni programi zatvorenog koda. Rezultat toga je da se svi prikazani programi i postupci u ovoj disertaciji mogu izvršiti bez prethodnog ulaganja finansijskih sredstava u određene programe i licence, što **disertaciju čini potpuno otvorenom i pristupačnom**.



Slika 2.12 - Sajam UMEX 2022 u Abu Dabiju

Jedan od značajnih doprinosa ovoj disertaciji proistekao je iz poseta renomiranim sajmovima i muzejima posvećenim vazduhoplovstvu, posebno je istaknuta poseta *Farnborough International Airshow*-u, što je omogućilo upoznavanje sa najnovijim dostignućima u oblasti vazduhoplovstva, kao i različitim tehnologijama koje se primenjuju u ovoj oblasti. Značajne su i posete vazduhoplovnim izložbama kao što su *ILA Berlin*, *Dubai Airshow*, ali i sajmovima kao što su *UMEX Abu Dabi* (prikazan na slici 2.12), *IDEX Abu Dabi*, *EUROSATORY Pariz* i *Milipol Pariz*.

Pored toga, posete specijalizovanim muzejima kao što su *Muzej vazduhoplovstva u Beogradu*, *Muzej vazduhoplovstva i astronautike na aerodromu Le Burže u Parizu*, *Nacionalni muzej avijacije i astronautike u Vašingtonu kao i Steven F. Udvar-Hazy Centar* na aerodromu *Dales* u Vašingtonu (prikazan na slici 2.13) omogućile su sticanje dubljeg razumevanja istorijskog razvoja i evolucije letelica, što je pružilo vredan kontekst za analizu i interpretaciju prikupljenih podataka.



Slika 2.13 - Muzej Steven F. Udvar-Hazy Centar

Takođe, veoma je značajno i učešće na domaćim sajmovima kao što su *Beogradski sajam tehnike* i *Međunarodni sajam naoružanja i vojne opreme PARTNER*. Iskustvo o upotrebi bespilotnih letelica je stečeno i tokom učešća na raznim prikazima sposobnosti Vojske Srbije (kao što je onaj prikazan na slici 2.14).

Poslednji projekat koji je potrebno istaknuti koji je možda i najviše uticao na razvoj i tok ove disertacije je razvoj i proizvodnja sistema `IKA-BOMBARDER` koji uključuje borbenu i izviđačku bespilotnu letelicu kao i svu neophodnu opremu za daljinsko upravljanje.



Slika 2.14 - Prikaz dronova iz serije IKA na vojnom aerodrom pukovnik-pilot Milenko Pavlović u Batajnici

3

Pregled stanja relevantne tehnike i prikaz dosadašnjih relevantnih istraživanja

"Neverovatno je koliko ne znate o igri koju igrate ceo život".

– Mickey Mantle[1]

U sledećim sekcijama disertacije je dat pregled stanja relevantne tehnike i prikaz dosadašnjih relevantnih istraživanja za pojedinačne delove sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

3.1 Simulator leta i pregled i odabir metode za proračun optimalne geometrije

Među mnogim postojećim mehanizmima, za simulatore letenja sa šest stepeni slobode najčešće se koriste paralelni mehanizmi (mehanizmi sa zatvorenim lancima) zasnovani na *Stjuartovoj platformi*, definisanoj u [7]. Od ovog prvog rada do danas, ovaj mehanizam je popularna istraživačka oblast u robotici sa brojnim istraživačkim radovima objavljenim o problemima inverzne i direktne kinematike [8]–[11], dinamike [8], [12]–[15], singulariteta i procene radnog prostora [8], [16]–[21]. U radu [22] napisanom 1998. godine, autori su predstavili najsavremeniji pregled literature o Stjuartovoj platformi dok su pomenuti radovi i dalje osnova istraživanja. Sličan sveobuhvatni rad napisan je 2016. godine [23].

Čak i nakon toliko godina (skoro šest decenija) istraživanja, i dalje postoji mogućnost pronalaženja novih ili boljih načina praktične upotrebe ovog mehanizma. Osim simulatora letenja, ovaj mehanizam se trenutno koristi, na primer, za mašinske alate [24], kao deo medicinskih instrumenata za precizne operacije [25], za visoko precizno prigušivanje vibracija za koristan teret svemirskih letelica [26], u okviru sistema stabilizacije antene za praćenje satelita [27], za nedestruktivnu inspekciju [28], za montažu avionskih delova [29], za ispitivanje dinamike mikro-elektro-mehaničkog sistema inklinometra [30], za energetski efikasan radni sto za mašinsku obradu [31], rehabilitaciju hoda [32], [33], stabilizaciju sistema za optoelektronske uređaje [34], mehanizam za pristajanje [35] pa čak i za helikopterske plutajuće helidrome [36] dok su dodatne aplikacije prikazane u [37].

Jakobijeva matrica ovog mehanizma analizirana je u [38] i [39] dok je numeričko rešenje direktne kinematike predstavljeno u radu [40]. Dinamička analiza je prikazana u radu [41].

[1] Mickey Mantle (20. oktobar 1931 - 13. avgust 1995) bio je američki profesionalni igrač bejzbola, mnogi smatraju da je jedan od najboljih bejzbol igrača svih vremena.

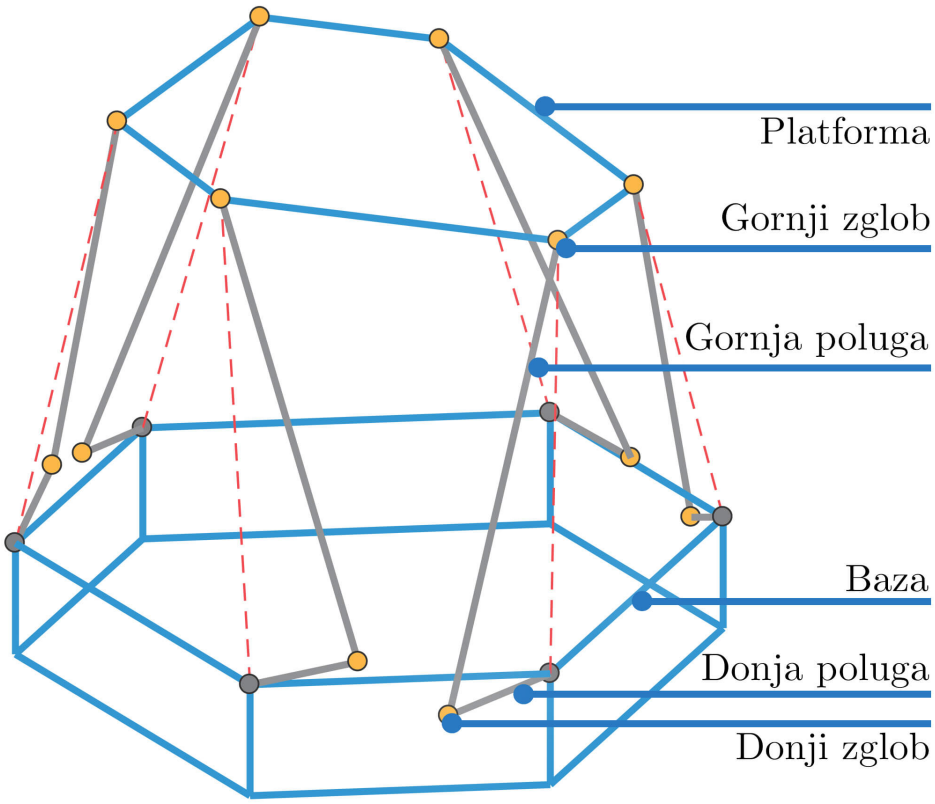
Sistem za praćenje kretanja platforme predstavljen je u radu [42], i može se koristiti za verifikaciju. Postoje i kontrolni sistemi zasnovani na kompjuterskoj viziji, kao što je opisano u [43] koji se takođe mogu koristiti za praćenje i verifikaciju kretanja.

U radu [44] korišćen je softver za kompjutersko projektovanje za vizuelizaciju radnog prostora, dok su u radovima [45], [46] prikazana simulaciona okruženja za ovu vrstu mehanizma.

Primer misije za koju ovaj mehanizam nije moguće optimizovati je dat u [47]. Za simulatore leta koriste se algoritmi za planiranje putanje [48]–[50] i različiti filteri kretanja [51]–[54] kako bi se poboljšale mogućnosti samog mehanizma, ali stepen poboljšanja zavisi od pilota. Rad [55] predstavlja implementaciju procena ponašanja pilota na simulatoru leta, radi daljeg poboljšanja kontrole mehanizma.

Manipulabilnost i spretnost ovog mehanizma su analizirani u radovima [56]–[59]. Za neke aplikacije, indikatori performansi takođe mogu biti i delimična ili potpuna izotropija mehanizma, definisana u [60].

Postoji mnogo različitih tipova geometrijskih interpretacija *Stjuartove platforme*, ali glavni cilj ovog rada je proračun optimalnih parametara geometrije *Stjuartove platforme sa obrtnim aktuatorima*, kao što je prikazano na slici 3.1. Ovaj mehanizam ima šest parova gornjih i donjih poluga (one se zajedno zovu noge), a pomeranje gornje platforme u odnosu na bazu postiže se rotacijom donje poluge oko ose koja prolazi kroz jedan kraj poluge. Ovo je rotacioni zglob između baze i donje poluge koji kontroliše električni motor. Gornja i donja poluga su povezane univerzalnim ili sfernim zglobovima, koji povezuju gornju polugu i platformu. Zbog toga se ovaj tip naziva i `6-RUS`, u skladu sa tipovima spojeva.



Slika 3.1 - Stjuartova platforma sa obrtnim aktuatorima

Dodatni doprinos ovog rada je i to što pored geometrijske definicije mehanizma i algoritma za proračun optimalne geometrije pruža rešenja za inverzni i direktni kinematički problem za Stjuartovu platformu sa obrtnim aktuatorima.

Za kontrolu položaja i orijentacije platforme, deo simulatora koji povezuje bazu i platformu (noga) treba da ima aktuator. Neki od drugih geometrijskih tipova ovog mehanizma koriste šest poluga promenljive dužine (linearni ili prizmatični aktuatori) ili takođe šest pari poluga ali od kojih je jedna poluga fiksne dužine, a druga ima promenljivu dužinu. Najkritičniji aspekti projektovanja mehanizma, pored tipova aktuatora su prostorna konfiguracija (lokacije zglobova) i vrste veza (tipovi zglobova).

Rešenje mehanizma zasnovano na obrtnim aktuatorima, u ovom slučaju elektromotorima sa odgovarajućim reduktorima, usvojeno je zbog svoje jednostavnosti za simulaciju, proizvodnju i održavanje. Ako servo motor sa servo kontrolerom nije pristupačan, standardni trofazni asinhroni motor se može kontrolisati pomoću *frekventnog regulatora* (FR), oba ova proizvoda su standardizovana i široko dostupna. Trenutni ugao vratila reduktora može se meriti enkoderom ili potenciometrom. Mehanizam sa ovim tipom aktuatora je najjednostavniji za umanjivanje i testiranje na tako umanjenom modelu. Čak i pre ulaganja u prvi prototip u punoj veličini, kinematika i algoritmi upravljanja se mogu fizički testirati.

Nakon izbora tipa aktuatora i prostorne konfiguracije, preostali deo projektovanja simulatora je izbor vrednosti unapred definisanih geometrijskih parametara koji u potpunosti definišu mehanizam sa svim ograničenjima. Zbog mnogo različitih kombinacija i njihovog međusobnog uticaja na performanse simulatora, teško je pronaći optimalne vrednosti za ove parametre. U ovom slučaju, sa mnogo parametara, često se moraju koristiti neki algoritmi optimizacije da bi se dobili dovoljno dobri rezultati u željenom vremenu. Za ovakve probleme, algoritmi optimizacije su često numerički i ne daju rešenje u zatvorenom obliku. Ovi algoritmi su tradicionalno iterativni. Problemi ograničene minimizacije mogu se rešiti minimiziranjem funkcije prilagođenosti sukcesivnim približavanjem unutar niza proračuna. Već postoje razumni napori u primeni *genetskog algoritma* [61]–[63] za rešavanje ove vrste problema. Genetski algoritam je pogodan jer ovaj problem ima mnogo lokalnih minimuma, a nije lako pronaći globalni minimum. *Algoritmi iscrpne pretrage* se u ovom slučaju ne biraju zbog računarske snage i vremena neophodnih za proračun. Postoje i noviji *evolucioni* i *bio-inspirisani metaheuristički algoritmi* koji se mogu koristiti za ovu vrstu problema, kao što su oni opisani u [64]–[66].

Realno kodirani mešovito celobrojni genetski algoritam je definisan sa radovima [67]–[70] i sličan algoritam je primenjen i na optimizaciju čitave letelice u [71].

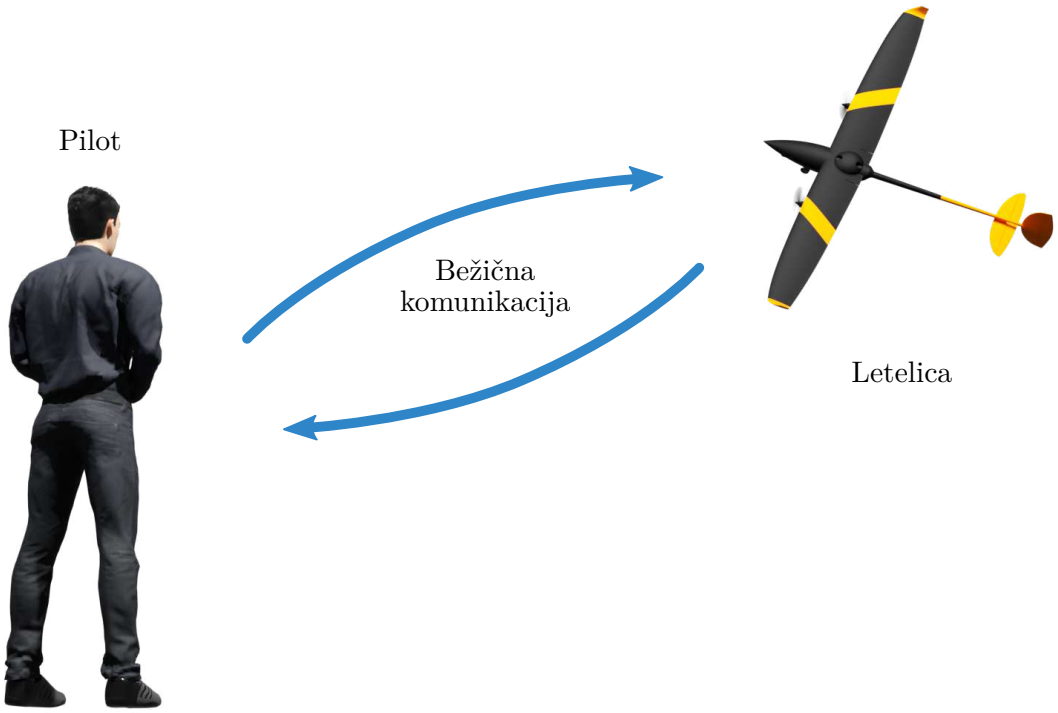
Optimalno projektovanje Stjuartove platforme za određene aplikacije je analizirano u radovima [72]–[74]. Pored toga, u radu [75] je predstavljen metod optimizacije za dobijanje željenog radnog prostora, dok su radovi [76] i [77] fokusirani na optimizaciju Stjuartove platforme za simuliranje leta.

Implementacija funkcije prilagođenosti definiše cilj proračuna. Mnogi indikatori performansi se u ovom slučaju mogu uzeti u obzir prilikom izračunavanja funkcije prilagođenosti, kao što su maksimalna izvodljiva orijentacija platforme, zapremina radnog prostora, potrebni hod zglobova, manipulabilnost, spretnost, singularnosti i fizičke smetnje između segmenata. U ovoj disertaciji se predlažu nova ograničenja, indikatori performansi i funkcija prilagođenosti za simulaciju leta na pristupačnom simulatoru leta sa električnim obrtnim aktuatorima. Prilikom konstruisanja funkcije prilagođenosti i odabira indikatora performansi, cilj je bio da se smanji potrebno vreme računanja i da se za kratko vreme pronađe geometrija mehanizma sa dovoljnim karakteristikama. S druge strane, svrha ograničenja je da se osigura da dobijena geometrija može biti materijalizovana, da je zaista pogodna za simulaciju leta i da nema fizičkih ograničenja na materijalizovanom mehanizmu. Ranije su problemi inverzne i direktne kinematike morali da se rešavaju više puta tokom ove vrste proračuna. Cilj predloženog algoritma je da značajno smanji broj izračunavanja u okviru proračuna.

3.2 Bespilotne letelice

Bespilotne letelice predstavljaju jedan od tri osnovna elementa sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi. Za potrebe ovog sistema letelica mora biti opremljena sistemima za merenje parametara kao što su brzina, ubrzanje i orijentacija, i sistemima za bežično slanje podataka do simulatora leta.

Pod bespilotnom letelicom se, u ovoj disertaciji, smatraju sve letelice bez pilota koji direktno upravlja iz pilotske kabine u letelici. Na osnovu ovoga, bespilotnom letelicom se smatra i sve ono što se naziva još i daljinski upravljanom letelicom, daljinski pilotiranim vazduhoplovom, dronom, bespilotnim multikopterom, letelicom na daljinsko upravljanje, bespilotnim vazduhoplovom, bespilotnim vazdušnim sistemom i slično. U slučaju upravljanih letelica, zbog odsustva pilota u letelici, neophodno je koristiti sistem koji će omogućiti daljinsko upravljanje (sistem kao onaj prikazan na slici 3.2). U slučaju da je upravljana letelica još i autonomna, sistemi za kontrolu i upravljanje koji se nalaze u letelici se programiraju tako da letelica ispunjava specifične zadatke ili niz zadataka bez potrebe za dobijanjem komandi ili korekcija od strane pilota.



Slika 3.2 - Daljinsko upravljanje letelicom

Primena bespilotnih letelica je na početku primarno bila u vojne svrhe, ali se u poslednje vreme koriste i u okviru civilnog vazduhoplovstva, kao i za zabavu. Opšta populacija danas najčešće prepoznaje dronove kao leteće kamere koje mogu praviti video-zapise, što i jeste proizvod koji je najjednostavnije moguće kupiti, odlaskom u prodavnicu potrošačke elektronike. Napredak tehnologije u oblasti mikroelektronike, kompozitnih materijala, električnih baterija i sistema upravljanja omogućio je konstruisanje bespilotnih letelica koje su sposobne da izvedu upravljani let u širokom opsegu visina, brzina i doleta.

Prilikom projektovanja bespilotnih letelica, kao što je i očekivano, postoje određene razlike u odnosu na klasično projektovanje letelica u kojima se nalazi pilot. Osnovna razlika je u tome što nije potrebno smestiti pilota unutar letelice, u letelici ne postoje kontrolni elementi kao što su palice za upravljanje, tasteri, prekidači niti instrumenti, i anvelopa letelice nije ograničena

fizičkim sposobnostima pilota, dok je sa druge strane najčešće neophodno razviti autonoman sistem upravljanja kao i pouzdanu bežičnu komunikaciju.

Kao što je i zaključeno u [78], mnogi piloti napominju da je razlika u upravljanju letelicom iz same letelice i pri daljinskom upravljanju letelice zapravo odsustvo osećaja koji imaju kada se nalaze u samoj letelici. Ovo je posledica samo delimičnog korišćenja senzorskih sposobnosti koje telo čoveka ima. Kao primer piloti navode da pri daljinskom upravljanju pilot ne može da čuje motor, ne može da oseti miris isparenja goriva od curenja, ne može da oseti dim usled vatre i ne može da oseti kretanje, vibracije i ubrzanja (ili usporavanja).

U [79] su date neke od implikacija toga što je letelica bespilotna sa posebnim akcentom na to da je potrebno zadržati željeni nivo pouzdanosti kada pilot više nije u letelici. Napominje se da su bežični sistemi za komunikaciju podložni prekidima u radu i da se za potrebe prenošenja video-signala kvalitet slike koja se prikazuje pilotu degradira.

Pored razlike u projektovanju podrazumeva se da postoje značajne razlike i u procesu sertifikacije i registracije takvih letelica i na kraju razlika i u samoj upotrebi.

Za registraciju civilnih bespilotnih letelica u Republici Srbiji zadužen je *Direktorat civilnog vazduhoplovstva Republike Srbije* (DCV). Ova institucija takođe omogućava proveru znanja iz *Vazduhoplovnih propisa*, polaganje ovog ispita je neophodni uslov za upravljanje bespilotnom letelicom.

Članom 5. *Pravilnika o bespilotnim vazduhoplovima* se definiše da je za upis u *"Evidenciju vazduhoplova"* potrebno dostaviti sledeću dokumentaciju:

- dokaz o plaćenoj carini ako je vazduhoplov proizveden u inostranstvu, odnosno overena pismena izjava vlasnika ako je vazduhoplov proizveden u Republici Srbiji;
- uputstvo proizvođača za korišćenje bespilotnog vazduhoplova, na srpskom ili na engleskom jeziku;
- ugovor o osiguranju od odgovornosti za štetu koja se upotrebom bespilotnog vazduhoplova pričini trećim licima, u skladu sa zakonom kojim se uređuje obavezno osiguranje u saobraćaju.

Za bespilotni vazduhoplov koji pripada kategoriji 4 (obuhvata bespilotne vazduhoplove čija je maksimalna masa na poletanju od 25 kg do 150 kg) podnosilac zahteva je dužan da dostavi Direktoratu tehničku dokumentaciju na osnovu koje može da se utvrdi sposobnost vazduhoplova za bezbedno letenje, kao i da omogući izvođenje demonstracionog leta uz prisustvo ovlašćenog službenog lica Direktorata.

Iz prethodno navedenog se može zaključiti da proizvođač bespilotnih letelica radi upisa njegovih letelica u evidenciju bi trebalo da obezbedi izjavu da je proizveo vazduhoplov, uputstvo za korišćenje na srpskom ili engleskom jeziku i eventualno još tehničku dokumentaciju (čija forma i sadržaj nisu precizno definisani) na osnovu koje može da se utvrdi sposobnost vazduhoplova za bezbedno letenje.

Za vojne letelice je nadležno Ministarstvo odbrane koje i izdaje dozvole za istraživanje, razvoj i proizvodnju. Nakon završetka razvoja je za potrebe uvođenja u naoružanje potrebno izvršiti verifikaciona ispitivanja gotovog proizvoda. Procedura prijavljivanja stranog uzorka sredstva naoružanja i vojne opreme na verifikaciona ispitivanja u opitnom centru, radi donošenja odluke o uvozu, kao i prijavljivanje na verifikaciona ispitivanja uzorka sredstva naoružanja i vojne opreme sa domaćeg tržišta gotovih proizvoda je regulisana zahtevima tačke 2. podtačka 5. iz Uputstva o prijavljivanju ispitivanja opitnim centrima (SVL. Br.20 od 28. jul 2000. godine). Taktički nosilac, u ovom slučaju, uz prijavu treba da dostavi sledeću dokumentaciju (dokumenta iz alineje od 6. do 9. su potrebna, ali nisu i neophodna za ocenu ispunjenja uslova ispitivanja).

1. taktičko-tehničke zahteve;
2. deklarisane taktičko-tehničke karakteristike proizvođača za sredstva naoružanja i vojne opreme;
3. dokumentaciju koja sadrži opis, rukovanje i održavanje;
4. spisak kompleta;
5. sklopni crtež sredstva naoružanja i vojne opreme i njegovih osnovnih delova;
6. rezultate ispitivanja proizvođača;
7. kontrolnu dokumentaciju sa metodama ispitivanja i standardima;
8. imenik (katalog) sastavnih delova;
9. remontnu dokumentaciju sa normativima rezervnih delova.

Kao što je i očekivano, za razliku od civilnih, vojne bespilotne letelice zahtevaju daleko detaljniju dokumentaciju koju proizvođač mora da obezbedi u cilju uvođenja gotovog proizvoda u naoružanje.

3.2.1 Istorijski osvrt

Za početak razvoja bespilotnih letelica se smatra period pred i u toku Prvog svetskog rata. U [80] je navedeno da su se još 1901. godine pojavile ideje da se letelice bez pilota koriste za meteorološka istraživanja kao i za inicijalna ispitivanja eksperimentalnih letelica kao što su letelice na vodonični pogon.

Određeni broj bespilotnih letelica, kao što su *Hjuit-Speri Automatski avion* (*eng. Hewitt-Sperry Automatic Airplane*) i *Kettering Bug*, razvijan je tokom i nakon Prvog svetskog rata. Pretpostavlja se da je *Hjuit-Speri Automatski avion* imao žiroskopski sistem za stabilizaciju i mogao je da preleti skoro 80 kilometara sa brzinom od 145 km/h i bojevom glavom mase 450 kg. Ova bespilotna letelica je bila inspiracija za kasniji razvoj *Kettering Bug* bespilotne letelice, obe letelice su konstruktivno bile dvokrilci sa elisama na prednjem delu trupa i lansirale su se pomoću katapulta sa šinama.

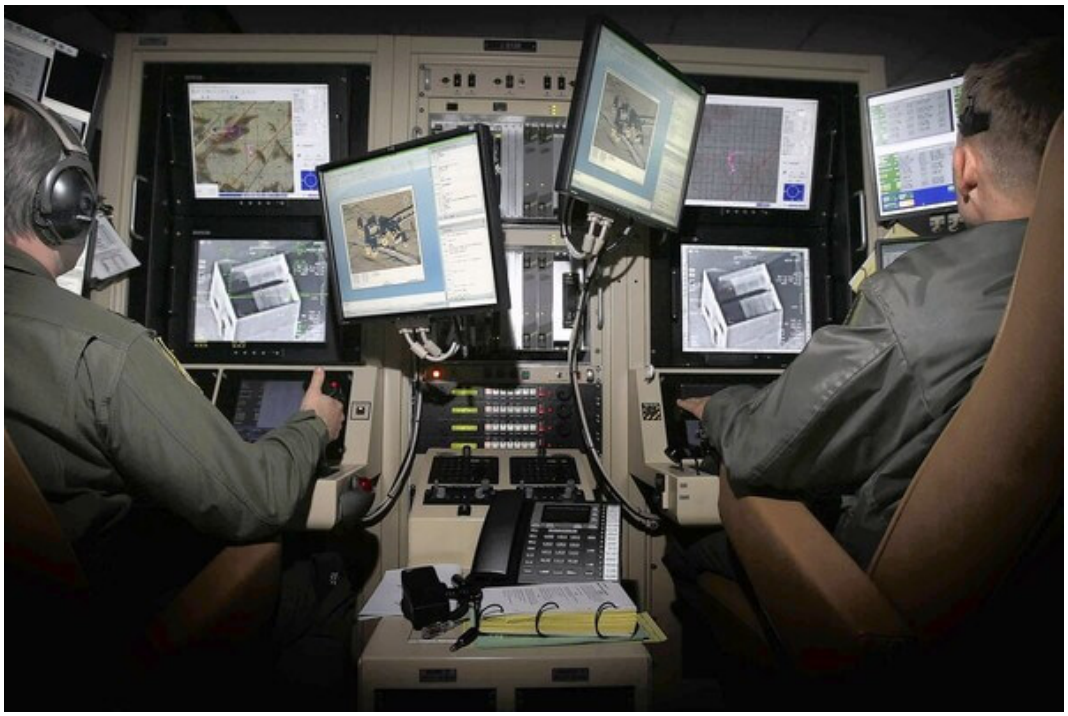
Usled tehnološkog napretka i stvaranja tehničkih preduslova, veći broj bespilotnih letelica je napravljen tokom Drugog svetskog rata. Prevashodno su korišćene kao mete za obuku artiljerijskih jedinica protivvazduhoplovne odbrane, u okviru obuke za ugaono praćenje cilja. Nakon Drugog svetskog rata je započeta ugradnja mlaznih motora. Kada su razvijeni sistemi protivvazduhoplovne odbrane vazduh-zemlja i vazduh-vazduh, bespilotne letelice su korišćene kao mete za obuku u bojevom gađanju. U Vijetnamskom ratu su prvi put značajno korišćene bespilotne letelice i to najviše za obeležavanje ciljeva za dejstvo avijacije.

Neke bespilotne letelice, kao što je *MQ-1 Predator*[81] mogu biti naoružane i izvršavati borbene zadatke. MQ-1 Predator je bespilotna letelica razvijena i proizvedena u SAD i operativno je koristi još nekoliko ratnih vazduhoplovstava zapadnih zemalja. U ranim devedesetim godinama prošlog veka, ova letelica je bila namenjena samo za izviđanje sa uskim pregledom, u smeru leta, pomoću kamera i drugih senzora. Kasnije su integrisana dva nosača za dve rakete ili za drugo alternativno naoružanje. Zbog postavljanja senzora u prednjem delu trupa, motor sa potisnom elisom je ugrađen u zadnji deo trupa.

Iznad mesta Uroševac, 13. maja 1999. godine, oborena je letelica *RQ-1A Predator*, čiji se ostaci (prikazani na slici 3.3) trenutno nalaze u *Muzeju vazduhoplovstva u Beogradu*. Na slici 3.4 je prikazana kontrolna soba jedne ovakve letelice.



Slika 3.3 - RQ-1A Predator u Muzeju vazduhoplovstva u Beogradu



Slika 3.4 - Kontrolna soba letelice RQ-1A Predator (preuzeto iz [82])

Upotreba bespilotnih letelica u vojnim operacijama je često kritikovana, pre svega zbog velike kolateralne štete, stradanja nedužnog civilnog stanovništva i zbog prećutnog kršenja suvereniteta država.

Osim u vojne svrhe, na početku je Nacionalna vazduhoplovna i svemirska administracija (eng. *National Aeronautics and Space Administration*, NASA) Sjedinjenih Američkih Država koristila bespilotne letelice u svrhe aerodinamičkih i strukturalnih ispitivanja, kao i ispitivanja stabilnosti i upravljivosti.

3.2.2 Primene bespilotnih letelica

Do sada su civilne i komercijalne primene bespilotnih letelica najčešće bile vezane za zadatke koji su ili previše opasni ili previše jednostavni i zbog toga neisplativi za primenu letelica sa ljudskom posadom. Neke od poznatih civilnih i komercijalnih primena bespilotnih letelica su:

- *Istraživanja* - na univerzitetima se koriste za različita eksperimentalna istraživanja najčešće u cilju aerodinamičke, strukturalne optimizacije i usavršavanja algoritama upravljanja. Radi se na razvoju novih i efikasnijih bespilotnih letelica, istražuju se načini za skladištenje energije i propulzija.

- *Poljoprivreda* - koriste se za osmatranje poljoprivrednog zemljišta, praćenje stanja i napredovanja useva. Bespilotne letelice omogućavaju relativno lako pravovremeno utvrđivanje potrebe za zaštitom od biljnih bolesti i štetočina, potrebe za navodnjavanjem, prihranom i drugim različitim agrotehničkim meram, ali i utvrđivanje potrebe za posebnim uređenjem zemljišta i njegovim popravkama. Letelica sa globalnim navigacionim satelitskim sistemom (GNSS) za navigaciju i kamerama može brzo obaviti snimanje i mapiranje farme ili područja za proizvodnju, precizno i više puta prikazati stanje površine sa štetočinama, nedostatak vlage u zemljištu pojavu biljnih bolesti i slične operacije na farmama. U poslednje vreme se bespilotne letelice sve više koriste i u okviru precizne poljoprivrede gde se tačno u zavisnosti od trenutnog stanja tretira zemljište, pa su za to letelice opremljene sejalicama i prskalicama.

- *Transport* - Mnoge kompanije razmatraju i istražuju mogućnost dostavljanja različitih proizvoda bespilotnim letelicama u cilju smanjivanja troškova i vremena potrebnog za transport (primer transporta je dat na slici 3.5).



Slika 3.5 - Primer transporta tereta i ispuštanja tereta sa padobranom

- *Industrijska* - neke od primena u industriji su inspekcija zgrada i krovova, aviona, vetroturbina, tornjeva/antena, mostova, gasovoda, naftovoda, dalekovoda ili nadgledanje rada nuklearnih elektrana. Sve češće se koriste i za 3D mapiranje

određenog prostora kao što su okna rudnika. Bespilotne letelice su postale nezamenljiv alat u filmskoj i televizijskoj produkciji.

- *Detekcija požara* - Bespilotne letelice opremljene infracrvenim senzorima nadleću šume radi pravovremene detekcije požara i šalju upozorenje na baznu stanicu sa tačnom lokacijom požara. Gašenje požara multikopterima je prikazano na slikama 3.6 i 3.7.



Slika 3.6 - Gašenje požara multikopterom IKA-01



Slika 3.7 - Gašenje požara multikopterom IKA-20

- *Sprovođenje zakona* - Zamenjuju neisplativu primenu helikoptera sa ljudskom posadom u policijskim jedinicama. Primenjuju se i za mnoge druge operacije, na slici 3.8 je prikazano onesposobljavanje osumnjičenog dok je na slici 3.9 prikazano razbijanje prozora multikopterom.



Slika 3.8 - Onesposobljavanje osumnjičenog pomoću multikoptera



Slika 3.9 - Pucanje iz pištolja sa multikopterom

- *Istraživanje Zemljine površine i atmosfere* - Vrše se precizna merenja gravitacionog polja Zemlje, deformacija njene površine koje mogu biti indikatori prirodnih nepogoda kao što su zemljotresi, klizišta i vulkanske erupcije, koriste se i za merenja zagađenosti vazduha.
- *Potraga i spašavanje* - Bespilotne letelice opremljene kamerama, svetlima i megafonima se koriste za potragu nestalih nakon prirodnih nepogoda i drugih nesreća.
- *Nadzor granica* - Patroliraju granicama u cilju sprečavanja neovlašćenog prelaska granice i krijumčarenja. Letelice sa velikih visina sa dobrim kamerama mogu pokrivati velike površine, na slici 3.10 je prikazan let multikoptera na preko 6000 metara nadmorske visine.



Slika 3.10 - Let električnog multikoptera na 6000 metara nadmorske visine

- *Rekreacija* - često se kupuju ili izrađuju kako bi se koristile za rekreativno letenje, zabavu, snimanje određenih događaja ali i za takmičenja u upravljanju letelicama.

Značajno je da se smatra da su civilne bespilotne letelice proizvedene u najviše primeraka letelice iz serije *Phantom* kineskog proizvođača *Dà-Jiāng Innovations* (DJI). Ovo su letelice koje sam proizvođač promoviše kao leteće kamere i osnovna upotreba im je za fotografisanje i snimanje video-zapisa.

Vojne upotrebe bespilotnih letelica mogu se podeliti na izviđačke i borbene. Borbeni dronovi se naoružavaju sa najraznovrsnijim naoružanjem, od onog koje se i ranije koristilo na vazduhoplovima do nekih potpuno novih rešenja. Kao primer, na slici 3.11 se vidi ispitivanje uticaja ručnog bacača raketa *M79 Osa* na delove multikoptera prilikom lansiranja rakete.



Slika 3.11 - Ispitivanje uticaja ručnog bacača raketa M79 Osa na delove multikoptera

Cilj ispitivanja je bio utvrđivanje da li je ovakvo naoružanje moguće koristiti sa takve letelice i zaključeno je da nije došlo do oštećenja i da treba analizirati i druge moguće ograničavajuće faktore.

Pored svih prikazanih upotreba, bespilotne letelice se koriste i za sprečavanje nedozvoljenog leta drugih bespilotnih letelica, kao što je prikazano na slici 3.12.



Slika 3.12 - Multikopter za borbu protiv drugih dronova

3.2.3 Klasifikacija bespilotnih letelica

Klasifikacija bespilotnih letelica može biti izvršena prema različitim kriterijumima. Klasifikacija se najčešće zasniva na maksimalnoj poletnoj masi, masi korisnog tereta, aerodinamičkoj konfiguraciji letelice, veličini, radijusu dejstva ili nivou autonomije.

Na osnovu aerodinamičke konfiguracije bespilotne letelice se dele na one koje koriste *statički* i *dinamički* princip letanja. Letelice koje koriste dinamički princip letenja se dele na one sa *fiksnim*, *rotacionim* (sa jednim ili više rotora), *mešovitim* (kombinacija fiksnih i rotacionih krila) i *mahajućim* krilima.

Bespilotne letelice, na osnovu načina upravljanja, mogu biti *neupravljane*, *daljinski upravljane od strane pilota* i *autonomne* (primeri neupravljane i daljinski upravljane letelice su dati na slici 3.13).



Slika 3.13 - Klizač - neupravljana letelica (levo) i kvadkopter - daljinski upravljana letelica(desno)

Klasifikacija bespilotnih letelica prema maksimalnoj poletnoj masi može biti sledeća:

1. *Velike* - Maksimalna poletna masa veća od 150 kg. U ovu kategoriju spadaju bespilotne letelice koje mogu prevoziti ljude, teret velikih gabarita i druge teške materijale. Zbog svojih karakteristika, opremljene su najkvalitetnijim senzorima i uređajima za navigaciju, a odlikuju se visokim nivoom pouzdanosti i dugotrajnosti. Koriste se u naprednim industrijskim, vojnim i komercijalnim aplikacijama.

2. *Srednje* - Maksimalna poletna masa manja od 150 kg. Iako manje od velikih letelica, srednje bespilotne letelice se često koriste za zadatke kao što su izviđanje i snimanje, za poljoprivredne operacije i prenos srednjih tereta. Ove letelice pružaju dobar balans između nosivosti i fleksibilnosti u primeni.

3. *Male* - Maksimalna poletna masa manja od 25 kg. Male bespilotne letelice se koriste za profesionalne primene, poput snimanja iz vazduha, inspekcije infrastrukture, nadzora industrijskih postrojenja i prenos manjih tereta. Zbog svojih dimenzija i mase, lako su prenosive, jednostavne za rukovanje i idealne za aplikacije gde je potrebna mobilnost i fleksibilnost.

4. *Mini* - Maksimalna poletna masa manja od 4 kg. Ove bespilotne letelice su popularne u oblastima poput inspekcije zgrada, ekoloških istraživanja, nadzora uskih i teško dostupnih područja, ali i u specijalizovanim industrijama gde su male dimenzije i agilnost ključne. Takođe se koriste za taktičke primene u vojnim i policijskim operacijama, zbog sposobnosti da lako izbegnu detekciju.

5. *Mikro* - Maksimalna poletna masa manja od 900 g. Ove bespilotne letelice su kompaktne, jednostavne za upotrebu i održavanje, te su pristupačne širokom krugu korisnika. Zbog svojih malih dimenzija koriste se za inspekciju malih prostora, istraživačke projekte i edukativne svrhe.

6. *Nano* - Maksimalna poletna masa manja od 250 g. Nano bespilotne letelice su izuzetno male i lagane, te mogu leteti u zatvorenim prostorima. Zbog svojih dimenzija i jednostavne konstrukcije, idealne su za početnike, ali se koriste i za specifične zadatke poput istraživanja, edukacije ili zabave. Zbog male mase i lakoće upravljanja, ove letelice su često prva opcija za korisnike koji ulaze u svet bespilotnih letelica.

3.2.4 Domaće bespilotne letelice

Razvoj i proizvodnja bespilotnih letelica zastupljeni su širom sveta. Praktično su sve države sveta zainteresovane za vojne i civilne primene bespilotnih letelica i zato trenutno postoji mnogo državnih ali i privatnih kompanija koje se bave razvojem i proizvodnjom bespilotnih letelica. Značajno je istražiti mogućnost upotrebe domaćih bespilotnih letelica sa sistemom za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

Vrabac (prikazan na slici 3.14) je bespilotna letelica kratkog dometa namenjena za izviđanje i osmatranje bojnog polja, misije granične patrole, obalske straže, rano otkrivanje požara, praćenje i kontrola važnih saobraćajnih trasa, električnih vodova i naftovoda. Letelica je razvijena od strane *Vojnotehničkog instituta*. Navođenje letelice je potpuno autonomno (programirana putanja, vođenje po azimutu kamere). Opremljen je autopilotom koji obezbeđuje funkcije održavanja visine, brzine i kursa, i funkciju sigurnog povratka kući. Operativni set za tri vojnika uključuje bespilotne letelice (3 komada), zemaljsku stanicu, daljinski video-terminal i transportnu opremu[83]. Tehničke karakteristike letelice su date u tabeli 3.1.



Slika 3.14 - Domaća bespilotna letelica Vrabac (preuzeto iz [83])

Tabela 3.1 - Tehničke karakteristike bespilotne letelice Vrabac

Karakteristika	**Vrednost**
Razmah krila	2.8 m
Dužina	1.94 m
Maksimalna masa pri poletanju	9 kg
Masa tereta	1.5 kg
Visina leta	300 - 500 m
Maksimalna brzina	120 km/h
Brzina krstarenja	61 km/h
Minimalna brzina	35 km/h
Maksimalna brzina penjanja (0 m)	2.5 m/s
Istrajnost leta	1 h +
Akcioni radijus	10 km

Bespilotna letelica velikog doleta nazvana **Pegaz** (prikazan na slici 3.15) namenjena je za dejstvo protiv lako oklopljenih/neoklopljenih vozila i žive sile laserskim i/ili IR/TV vođenim raketama. Letelica je razvijena od strane *Vojnotehničkog instituta*. Pored toga, bespilotna letelica Pegaz namenjena je za dnevne i noćne zadatke izviđanja na operativno-taktičkom nivou, prikupljanje obaveštajnih podataka kao i akviziciju ciljeva i upravljanje artiljerijskom vatrom.

Vođenje je potpuno autonomno (programirana putanja, vođenje po azimutu kamere). Poseduje i komandovani mod vođenja (direktno ili preko autopilota brzine, visine i kursa), kao i mod za bezbedan povratak[84]. Tehničke karakteristike letelice su date u tabeli 3.2. Operativni komplet čine:

- Bespilotna letelica sa senzorima - 3 kom.
- Zemaljska stanica za kontrolu i vođenje - 1 kom.
- Zemaljska stanica za retranslaciju - 1 kom.
- Daljinski video-terminal - 3 kom.
- Motorno vozilo za transport kompleta - 3 kom.



Slika 3.15 - Domaća bespilotna letelica Pegaz 011 (preuzeto iz [84])

Tabela 3.2 - Tehničke karakteristike bespilotne letelice Pegaz

Karakteristika	**Vrednost**
Motor	dvocilindrični bokser motor
Snaga	32 KW (43 KS)
Elisa	drvena, dvokraka potisna
Razmah krila	6.34 m
Površina krila	4.24 m
Dužina	5.395 m
Masa prazne letelice	120 kg
Masa opreme misije	30-55 kg
Maksimalna poletna masa	230 kg
Maksimalna brzina	210 km/h
Brzina krstarenja	130 - 150 km/h
Operativna visina	3000 m
Istrajnost leta	8 h +
Domet	50 km +
Poletanje	automatsko sa piste/katapult
Sletanje	automatsko na pistu/padobran

IKA-BOMBARDER (prikazan na slici 3.16) je vojni dron sa obrtnim krilima projektovan i proizveden od strane kompanije PR-DC (*Pink Research & Development Center iz Šimanovaca*) u heksakopter konfiguraciji. Konstrukcija letelice je napravljena od kompozitnih materijala, uglavnom od karbonskih vlakana i epoksidne smole, što omogućava let u trajanju od 30 minuta sa 20 kilograma tereta, što je jedinstveno na tržištu. Let može biti potpuno autonoman i može nositi 12 avio-bombi prečnika 60mm (kao što je Krušik M73) koje se mogu precizno ispuštati jedna po jedna i pritom neutralisati različite ciljeve, i danju i noću[85]. Za upravljanje letelicom se koristi daljinski upravljač **IKA-CTRL**. Osnovne tehničke karakteristike letelice su date u tabeli 3.3, performanse su date u tabeli 3.4, dok su karakteristike naoružanja date u 3.5.



Slika 3.16 - Domaća bespilotna letelica IKA-BOMBARDER

Tabela 3.3 - Osnovne tehničke karakteristike bespilotne letelice IKA-BOMBARDER

Karakteristika	**Vrednost**
Pogon i izvor napajanja	BLDC električni motori i zamenjiva litijumska baterija od 4.3 kWh
Prečnik/korak elise	812.8 mm / 279.4 mm (32" / 11")
Maksimalna snaga svakog motora	5.7 kW (za 24 kgf pri 4600 rpm)
Dimenzije	2490 mm x 2400 mm x 670 mm
Dimenzije transportnog pakovanja	1200 mm x 1100 mm x 850 mm
Materijal konstrukcije	Polimer ojačan karbonskim vlaknima
Maksimalna poletna masa	70 kg
Optimalna masa tereta	20 kg
Radni temperaturni opseg (sa baterijom)	od −5 do 50 °C
Temperatura skladištenja (bez baterije)	od −40 do 70 °C
Daljinski upravljač	IKA-CTRL sa prilagođenom aplikacijom FlightControl, istovremena upotreba više upravljača (modovi kontrole i nadzora, odvojena kontrola naoružanja)
Oprema	3-osna EO/IR stabilizovana kamera

Tabela 3.4 - Performanse leta bespilotne letelice IKA-BOMBARDER

Karakteristika	Vrednost
Radijus misije	od 5 km do 15 km
Operativna visina	od 150 m do 500 m
Istrajnost leta	od 20 min do 40 min
Maksimalni pređeni put	do 30 km
Plafon leta	7000 m
Maksimalna brzina	90 km/h
Brzina krstarenja	60 km/h
Brzina penjanja	10 m/s
Otpornost na vetar	8 m/s

Tabela 3.5 - Karakteristike naoružanja bespilotne letelice IKA-BOMBARDER

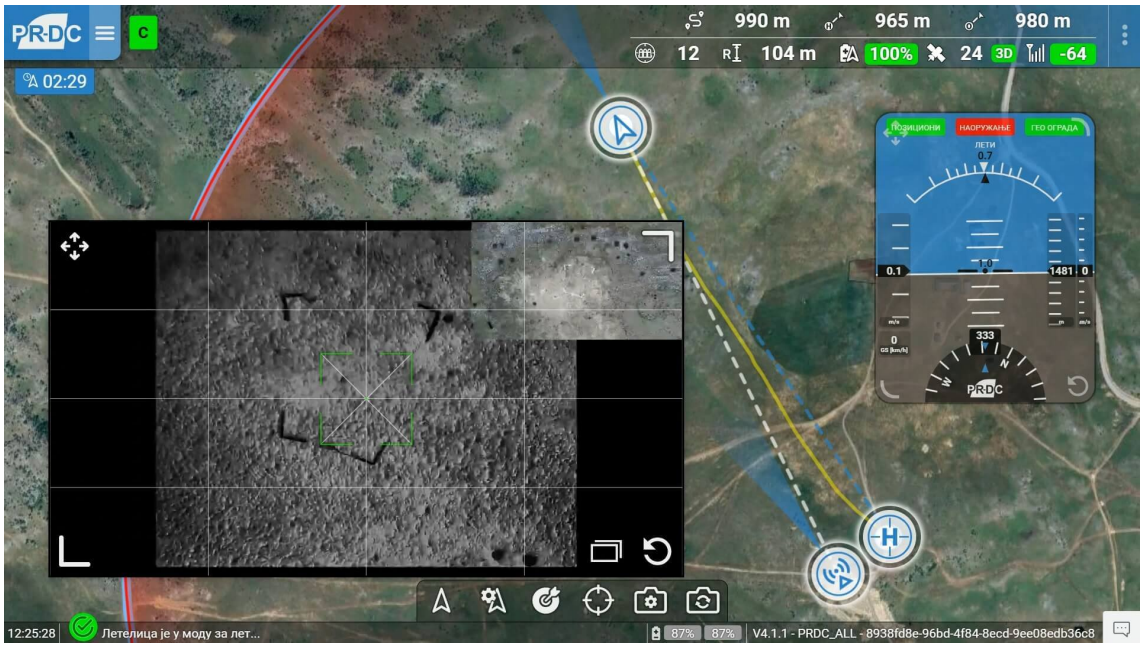
Karakteristika	Vrednost
Tip i količina	12 avio-bombi prečnika 60 mm
Dužina bombe sa upaljačem	286 mm (11.3")
Tip upaljača	udarni, superbrza reakcija, posebno dizajniran za upotrebu na dronovima
Sigurnost upaljača	dva sigurnosna pina (za transport i za ispuštanje)
Masa jedne bombe sa upaljačem	1350 g
Eksplozivno punjenje	trotil (TNT)
Masa eksplozivnog punjenja	250 g
Radijus eksplozije (1 penetracija / m^2)	10 m
Bezbedan temperaturni opseg rada	-30 °C do +50 °C
Visina ispuštanja iznad zemlje	$>$ 150 m

Na slici 3.17 je prikazano precizno ispuštanje avio bombi sa multikoptera pomoću dva mehanizma za ispuštanje. Visina ispuštanja je najčešće između 100 metara i 500 metara što je adekvatna udaljenost i za kamere kojima se prethodno osmatra meta.



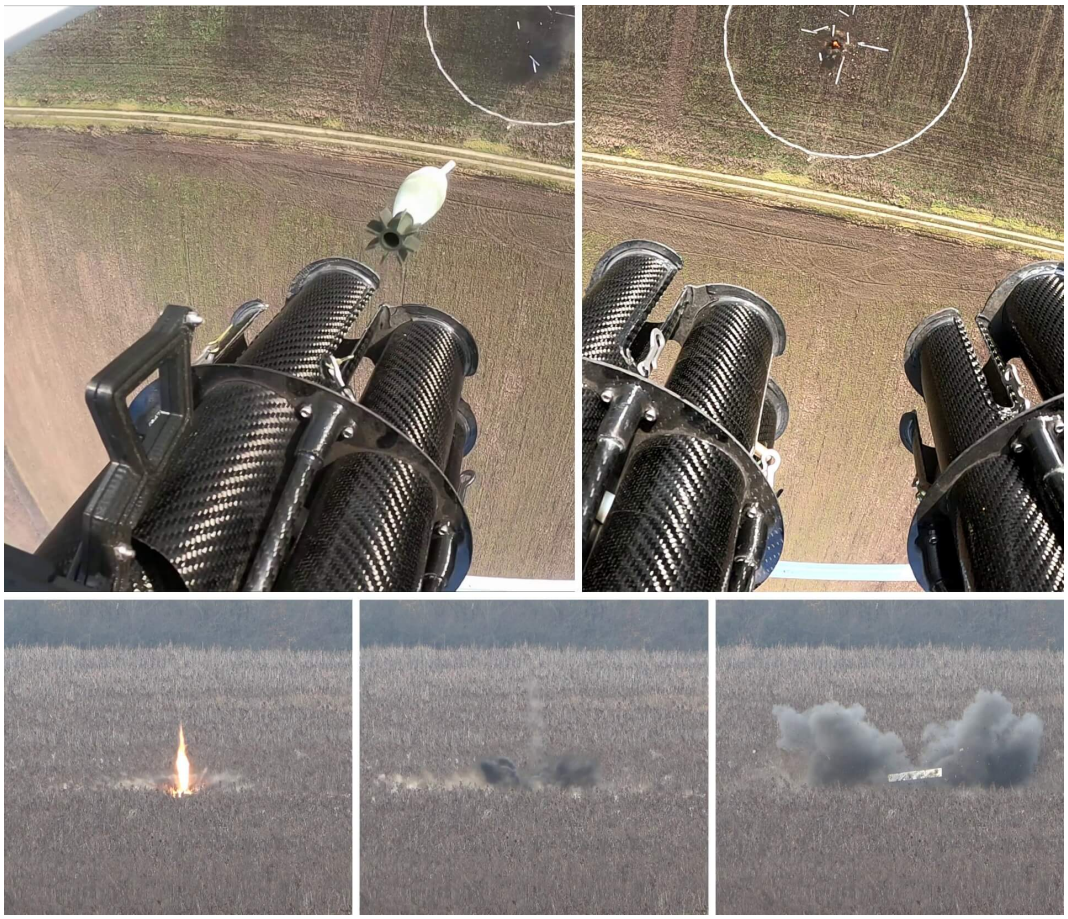
Slika 3.17 - Prikaz ispuštanja avio bombi sa multikoptera

Na slici 3.18 je prikazan ekran daljinskog upravljača bespilotne letelice sa prikazom sa infracrvene kamere na letelici tokom osmatranja mete. Zahvaljujući razlici u toploti se jasno mogu videti oivičenja mete.



Slika 3.18 - Prikaz slike sa infracrvene kamere prilikom osmatranja mete

Samo dejstvo avio-bombi prečnika 60 mm je prikazano na slici 3.19.



Slika 3.19 - Prikaz bombardovanja avio-bombama prečnika 60 mm

Operater tokom dejstva može gledati termalnu kameru na kojoj se veoma jasno vidi mesto eksplozije i eventualno izvršiti korekciju. Dejstvo ove avio-bombe posmatrano termalnom kamerom je prikazano na slici 3.20.



Slika 3.20 - Dejstvo avio-bombe prečnika 60 mm posmatrano termalnom kamerom

Na slici 3.21 je prikazano kretanje vozila sa letelicom IKA-BOMBARDER na kutiji za transport a nakon toga i poletanje letelice dok je vozilo u pokretu.



Slika 3.21 - Poletanje IKA-BOMBARDER-a sa vozila u pokretu

Letelica IKA-BOMBARDER se može koristiti i sa vodenih površina, na slici 3.22 je prikazano poletanje letelice sa plovnog objekta koji je neposredno pre toga isplovio iz vode, zatim otvorio poklopac i ugrađenim mehanizmom podigao letelicu tako da izađe iz samog objekta.



Slika 3.22 - Poletanje IKA-BOMBARDER-a sa plovnog objekta

3.3 Standardi i važeći zakoni

Upotreba bespilotnih letelica u Republici Srbiji je predmet stroge regulative kako bi se obezbedila sigurnost, efikasnost i usklađenost sa međunarodnim standardima. Zakonodavni okvir koji reguliše ovu oblast obuhvata zakone i pravilnike koji definišu uslove za registraciju, upotrebu i kontrolu bespilotnih letelica, kako u civilne, tako i u vojne svrhe. Ovi propisi su ključni za osiguravanje bezbednog korišćenja bespilotnih letelica, sprečavanje nesreća i nezakonitih aktivnosti, te pružanje jasnih smernica operatorima i proizvođačima. Treba voditi računa o tome da je trenutni nivo integracije bespilotnih letelica u postojeći vazdušni prostor još uvek na niskom nivou. Ovo je konzervativan pristup koji osigurava nesmetano obavljanje komercijalnog avio-saobraćaja.

U Republici Srbiji je upotreba bespilotnih letelica regulisana sledećim zakonom i pravilnikom [86]:

- *ZAKON O VAZDUŠNOM SAOBRAĆAJU* ("Sl. glasnik RS", br. 73/2010, 57/2011, 93/2012, 45/2015, 66/2015 - dr. zakon, 83/2018 i 9/2020)
- *PRAVILNIK O BESPILOTNIM VAZDUHOPLOVIMA* ("Sl. glasnik RS", br. 1/2020).

Za proizvodnju bespilotnih letelica koje su namenjene vojnoj upotrebi, neophodno je dobiti odgovarajuću dozvolu za proizvodnju naoružanja i vojne opreme koja se izdaje na određeni period, specifično za ovu oblast. Ovaj proces obuhvata stroge regulative i zahteva usklađenost sa posebnim kriterijumima kako bi se osiguralo da proizvođači ispunjavaju sve potrebne uslove za bezbednost i efikasnost proizvodnje. Ova regulativa je definisana sledećim zakonom:

- *ZAKON O PROIZVODNJI NAORUŽANJA I VOJNE OPREME* ("Sl. glasnik RS", br. 36/2018)

Što se tiče bežičnih komunikacija, u Republici Srbiji postoji regulatorno telo za elektronske komunikacije i poštanske usluge (*RATEL* [87]). Bežične komunikacije su uređene na osnovu sledećeg zakona:

- *ZAKON O ELEKTRONSKIM KOMUNIKACIJAMA* ("Sl. glasnik RS", br. 35/2023)

Jedna od obaveza proizvođača je da vodi kontrolu kvaliteta (u Republici Srbiji postoji *Vojna kontrola kvaliteta* koja vrši sertifikaciju i vrši kontrolu usaglašenosti sa određenim standardima [88], s druge strane *Tehnički opitni centar* je ustanova ovlašćena za sprovođenje završnih, verifikacionih i homologacionih ispitivanja sredstava naoružanja i vojne opreme i metrološko obezbeđenje sistema odbrane [89]), integrisani menadžment sistem proizvođača u ovoj industriji mogu da čine sledeći standardi:

- *SRPS ISO 9001:2015* - Sistem menadžmenta kvalitetom – zahtevi;
- *SORS 9000:2021* - Zahtevi za obezbeđenje kvaliteta u procesu realizacije proizvoda (ovaj vojni standard je usaglašen sa NATO standardom `AQAP 2110`);
- *SORS 9423:2018* - Obezbeđenje kvaliteta u procesu realizacije proizvoda u vazduhoplovnoj, svemirskoj i odbrambenoj industriji (`SRPS EN 9100:2018` – Sistem menadžmenta kvalitetom – Zahtevi za vazduhoplovstvo, svemirske i odbrambene organizacije, identičan sa `EN 9100:2018`).

Neki opšti vojni standardi su primenljivi i na bespilotne letelice i simulatore leta, kao što su:

- *MIL-STD-810H* - Uticaji sredine;
- *MIL-STD-461G* - Kontrola elektromagnetnih smetnji;
- *SORS 5706* - Ispitivanje postojanosti sklopova i uređaja;
- *RTCA DO-254* - Smernice za projektovanje elektronike za vazduhoplove;
- *RTCA DO-178C* - Zahtevi za programe za vazduhoplove.

Testiranje na uticaje sredine je ključno jer ispituje kako sredstva funkcionišu tokom celog životnog veka, uključujući transport i skladištenje. Sredstva se testiraju u klima komorama na različite temperature, sa i bez vlage, kao i na vibracije, udare, uticaj slane magle, plesni i druge faktore.

Standardi za bespilotne letelice najčešće imaju specifične namene za određene kategorije letelica. Specifični standardi za bespilotne letelice uključuju:

- *NATO AEP-83* - definiše zahteve za tehničku ispravnost lakih bespilotnih letelica (bespilotne letelice sa fiksnim krilima i maksimalne poletne mase do 150 kg).
- *NATO AEP-80* - definiše zahteve za tehničku ispravnost bespilotnih letelica sa rotacionim krilima (bespilotni helikopteri maksimalne poletne mase od 150 kg do 3175 kg).
- *NATO AEP-89* (NACRT) - definiše zahteve za tehničku ispravnost lakih bespilotnih letelica sa vertikalnim poletanjem i sletanjem (bespilotne letelice maksimalne poletne mase do 150 kg).
- *STANAG 4671* - definiše zahteve za tehničku ispravnost bespilotnih letelica (bespilotne letelice sa fiksnim krilima i maksimalne poletne mase od 150 kg do 20 000 kg).
- *STANAG 4702* - postavlja zahteve za ispravnost bespilotnih letelica sa rotacionim krilima - poziva se na *NATO AEP-80*.

- *STANAG 4703* - definiše zahteve za tehničku ispravnost lakih bespilotnih letelica (sa fiksnim krilima) - poziva se na *NATO AEP-83*.
- *STANAG 4738* (NACRT) - postavlja zahteve za ispravnost lakih bespilotnih letelica sa vertikalnim poletanjem i sletanjem - poziva se na *NATO AEP-89*.
- *CS-LUAS* - daje preporuke za sertifikaciju lakih bespilotnih letelica sa fiksnim krilima i maksimalne poletne mase do 750 kg.
- *CS-LURS* - smernice za sertifikaciju lakih bespilotnih letelica sa rotacionim krilima i maksimalne poletne mase do 750 kg.

Ovi standardi obuhvataju sve aspekte projektovanja, proizvodnje i operativne upotrebe bespilotnih letelica, omogućavajući visoke nivoe bezbednosti, pouzdanosti i performansi.

3.4 Alati za trodimenzionalno modeliranje i kompjutersku grafiku

Savremeno projektovanje je gotovo nezamislivo bez alata za trodimenzionalno modeliranje i kompjutersku grafiku. Kao što je prikazano u nastavku, tih alata može biti mnogo i mogu biti prilično raznovrsni. Pri izboru alata se svakako polazilo od toga šta je potrebno uraditi, ali je prednost uvek davana projektima otvorenog koda. Nekim projektima je i dat doprinos u vidu ispravljanja grešaka, prijavljivanja grešaka kao i izradi primera i uputstava, a sve to kao posledica rada na ovoj disertaciji.

3.4.1 FreeCAD

Programski paket `FreeCAD` [90] je besplatni program otvorenog koda za parametarsko 3D modeliranje. Trenutno je najrazvijeniji, najčešće korišćen, aktuelan, besplatan program otvorenog koda za 3D modeliranje. Pored njega, postoje mnogi napredni programi za 3D modeliranje ali su zatvorenog koda i izuzetno skupi. U knjizi [91] je detaljno prikazano parametarsko modeliranje u jednom programu zatvorenog koda.

`FreeCAD` je prvenstveno `MCAD` (Mechanical Computer Aided Design) program, što znači da sadrži skup metoda i alata kojima se inženjer služi tokom projektovanja mehaničkih delova i sklopova na računaru. Pod time se podrazumeva da program omogućava:

- stvaranje geometrijskog modela,
- dimenzionisanje modela,
- unos potrebnih promena na modelu,
- čuvanje/arhiviranje modela kako bi drugi korisnici mogli da ga koriste,
- stvaranje sklopova,
- izrada prateće dokumentacije,
- pripremu modela za analizu (CAE) i simulaciju proizvodnog procesa (CAM).

Kombinacijom ljudske imaginacije i modernih softverskih alata moguće je konstruisati sistem po definisanim zahtevima. Za potrebe mašinske industrije, program omogućava modeliranje linijskih, ravanskih i prostornih tela veoma složene geometrije, uključujući i elemente od limova. Osim modeliranja delova, program omogućava i formiranje sklopova. Izuzetno je značajno što je pomoću ovakvog programa moguće vršiti vizuelnu inspekciju sklopova, moguće je proveriti da li dolazi do kolizije među elementima strukture kao i da li je kinematski moguće ostvariti željena kretanja mehanizama.

Na osnovu izrađenih 3D modela je dalje moguće uraditi kompletnu tehničku dokumentaciju koja je potrebna za izradu delova, od osnovnih 2D tehničkih crteža do programiranja numerički upravljanih mašina-alatki kao što su CNC glodalice (3D putanje), 3D štampači (3D putanje) ili CNC laseri za rezanje ili graviranje (2D vektorski crteži) i CNC mašina za rezanje tekstila (takođe 2D vektorski crteži u procesu izrade delova od kompozitnih materijala).

Program omogućava stvaranje geometrije pomoću programskog jezika `Python`, što omogućava brzo implementiranje neophodnih funkcionalnosti koje u tom trenutku nisu dostupne ili automatizaciju određenih postupaka za kreiranje modela. Tako je moguće generisati niz različitih modela na osnovu promene samo nekoliko ulaznih parametara (katalog modela). Veoma je bitno i to što se koristi baš programski jezik `Python` jer je on široko rasprostranjen, veliki broj inženjera ga koristi i postoji velika količina koda koja je već napisana.

3.4.2 CGAL

Biblioteka *The Computational Geometry Algorithms Library* (`CGAL`) je softverski projekat otvorenog koda koji omogućava lak pristup efikasnim i pouzdanim geometrijskim algoritmima u obliku `C++` biblioteke[92].

Integracijom ove biblioteke u neki program se omogućava korišćenje algoritama za triangulaciju, voronojeve dijagrame, alfa oblike, bulove operacije nad poligonima i poliedrima, generisanje površinskih i zapreminskih mreža i drugo. Biblioteka je takođe, veoma značajna kada je iz oblaka tačaka potrebno generisati 3D modele. Ovo je bitno za procese proračuna u diskretnim tačkama ili za obradu podataka sa senzora kao što su radari i lidari.

3.4.3 MeshLab

Programski paket otvorenog koda `MeshLab` je namenjen za obradu i uređivanje 3D trouglastih mreža [93]. Pruža skup alata za uređivanje, čišćenje, popravku, proveru, renderovanje, teksturiranje i pretvaranje mreža. Nudi funkcije za obradu neobrađenih podataka proizvedenih alatima/uređajima za 3D skeniranje i za pripremu modela za 3D štampanje.

Program `MeshLab` je veoma značajan kada postoji potreba za brzom analizom 3D mreža ili oblaka tačaka jer pruža izuzetno brzo učitavanje podataka i njihovu vizualizaciju. Takođe, ovim programom je moguće brzo obraditi postojeće modele (npr. obrisati delove modela, izvršiti njihovo skaliranje ili pojednostavljivanje). Potreba za ovakvim programom je naročito porasla sa pojavom 3D štampača i skenera koji isključivo koriste 3D modele formirane pomoću 3D mreža.

3.4.4 Blender

Program `Blender` je besplatni alat za 3D kompjutersku grafiku otvorenog koda koji se prvenstveno koristi za izradu animiranih filmova, vizuelnih efekata i realističnih prikaza 3D

modela [94]. Program omogućava kompletan 3D radni tok — modeliranje, rigovanje modela, animaciju, simulaciju, renderovanje, komponovanje i praćenje pokreta, čak i uređivanje videa i kreiranje igara.

Program `Blender` je veoma značajan jer omogućava izradu realističnih prikaza 3D modela za različite prezentacije, promotivne materijale ili prosto za verifikaciju konačnog izgleda pre ulaganja u stvarnu izradu. Takođe, program omogućava rad sa 3D modelima različitih formata što omogućava pripremu 3D modela za korišćenje u okviru drugih programa, u okviru pripreme se mogu dodatno menjati teksture ili praviti animacije.

3.4.5 MakeHuman

`MakeHuman` [95] je besplatni program otvorenog koda za 3D kompjutersku grafiku namenjen za izradu fotorealističnih 3D modela humanoida. Razvila ga je zajednica programera i umetnika zainteresovanih za 3D modeliranje likova za video-igre i druge primene. Ova zajednica je omogućila da se u programu nalazi mnogo opcija koje omogućavaju da se kreira lik tačno određenog izgleda i sa određenom odećom i drugim rekvizitima.

Program omogućava kreiranje modela čoveka različitih veličina (moguće je podešavati veličinu tela npr. visinu kao i masu tela, broj godina i drugo) kao i promenu položaja tela, tako da je moguće napraviti čoveka određene veličine i u nekom određenom položaju. Ovo je izuzetno značajno za virtuelno ispitivanje ergonomije. Moguće je virtuelnog čoveka pomerati i odrediti moguća kretanja njegovog tela i pomoću drugih programa kao što je `Blender` moguće je odrediti recimo zonu vidljivosti. Takođe, ovim modelima je lako moguće unaprediti fotorealistične prikaze npr. moguće je dodati pilote, kabinsko osoblje i putnike u okviru izrade fotorealističnog prikaza aviona.

3.4.6 CesiumJS

`CesiumJS` je `JavaScript` biblioteka za kreiranje 3D globusa i 2D mapa u internet pretraživaču. Koristi `WebGL` (Web Graphics Library) tehnologiju za hardverski ubrzanu grafiku, a radi na više platformi, u više pretraživača i namenjen je za vizuelizaciju dinamičkih podataka. Program je zasnovan na otvorenim formatima i projektovan je za robusnu interoperabilnost i skaliranje za ogromne skupove podataka [96].

Značaj programa je u tome što omogućava pozicioniranje kamere i objekata na tačnoj lokaciji na 3D globusu (npr. na osnovu geografske širine, geografske dužine i nadmorske visine), omogućava da se u prozor učitaju direktno 3D modeli i to sa animacijama npr. iz `glTF` (Graphics Library Transmission Format) ili `GLB` (GL Transmission Format Binary file) formata. Biblioteka omogućava učitavanje fotorealističnih 3D pločica mape sveta, učitavanje digitalne visine terena ili digitalnih ortofotoa (DOF). Pomoću samo ove biblioteke i odgovarajućih podataka je potpuno moguće prikazati, i to fotorealistično, putanju leta letelice u odnosu na Zemlju.

3.4.7 Unreal Engine

`Unreal Engine` je 3D platforma prvenstveno namenjena za razvoj video-igara koju je razvila kompanija *Epic Games* [97]. Od svog nastanka pa do danas, ova platforma je postala standard za razvoj ne samo igara već i animiranih filmova, realistični prikaz modela u realnom vremenu, razvoj virtuelnih okruženja za virtuelnu realnost (`VR`) ili augmentovanu realnost (`AR`) ili prikaz arhitektonskih rešenja.

Ova platforma je značajna jer omogućava jednostavan prikaz virtuelnih modela u fotorealističnom obliku i to sa renderovanjem u realnom vremenu, moguće je veoma precizno simulirati let letelice u odnosu na Zemlju i iskoristiti postojeće okruženje za generisanje podataka za senzore letelice. Pored toga, program omogućava povezivanje sa specijalnim `VR` i `AR` naočarima, generisanje slike za naočare i istovremeno isčitavanje njihove orijentacije i položaja, što se može izuzetno primeniti za npr. formiranje virtuelne pilotske kabine.

3.4.8 Inkscape

`Inkscape` je program za jednostavno kreiranje i uređivanje grafičkih vektora [98]. Ovo je besplatni program otvorenog koda koji ima kompletan skup alata za dizajn koji omogućavaju rad sa vektorskim grafičkim datotekama u formatu `SVG` (Scalable Vector Graphics).

Jedna od veoma bitnih funkcionalnosti programa je što omogućava generisanje vektorske slike na osnovu rasterske slike, tako što se prate linije na slici (alatka `Trace Bitmap`). Program omogućava npr. uređivanje vektorskih grafika u cilju njihovog štampanja ili prezentovanja, izradu ilustracija kao i izradu 2D crteža za CNC rezanje ili graviranje laserom.

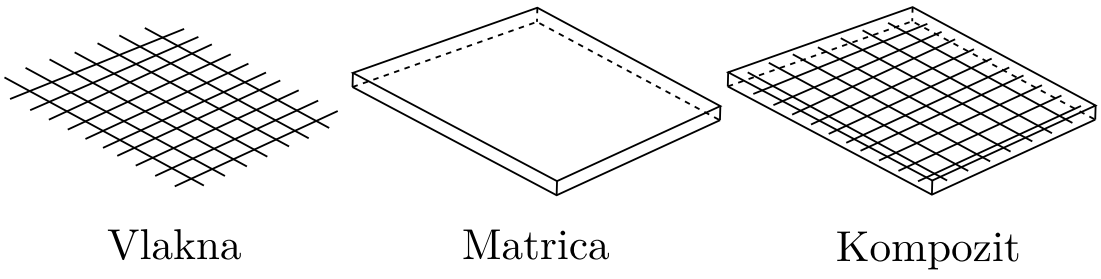
3.4.9 GIMP

`GNU Image Manipulation Program`, poznat po svojoj skraćenici `GIMP`, je besplatan program otvorenog koda za rad sa rasterskom grafikom. Program se koristi za manipulaciju i uređivanje slika, crtanje u slobodnoj formi kao i za transkodiranje između različitih formata datoteka [99].

Program je značajan jer omogućava uklanjanje pozadina sa fotografija, promenu njihove veličine i orijentacije kao i mnoge druge korekcije na samoj fotografiji.

3.5 Kompozitni materijali, 3D štampači i skeneri

Mnogi materijali su efektivno kompoziti. Ovo se posebno odnosi na prirodne biološke materijale, koji se često sastoje od najmanje dve komponente. U mnogim slučajevima, prisutna je čvrsta i kruta komponenta, često u izduženom obliku (vlakno), ugrađena u mekšu i usaglašeniju komponentu (matricu) [100]. Na slici 3.23 je ilustrovano prethodno opisano: vlakna i matrica zajedno formiraju kompozit.



Slika 3.23 - Formiranje kompozitnog materijala

Upotreba kompozitnih materijala nastavlja dalje da se širi. Tehnologija izrade delova od kompozitnih materijala kao i sami materijali konstantno napreduju. Pojavom veoma pristupačnih 3D štampača pojavili su se i novi materijali, najčešće različiti plastični materijali sa određenim primesama. Sa druge strane, napredovanjem dizajna sve više se koriste kompleksni oblici koje je teško izraditi klasičnim metodama.



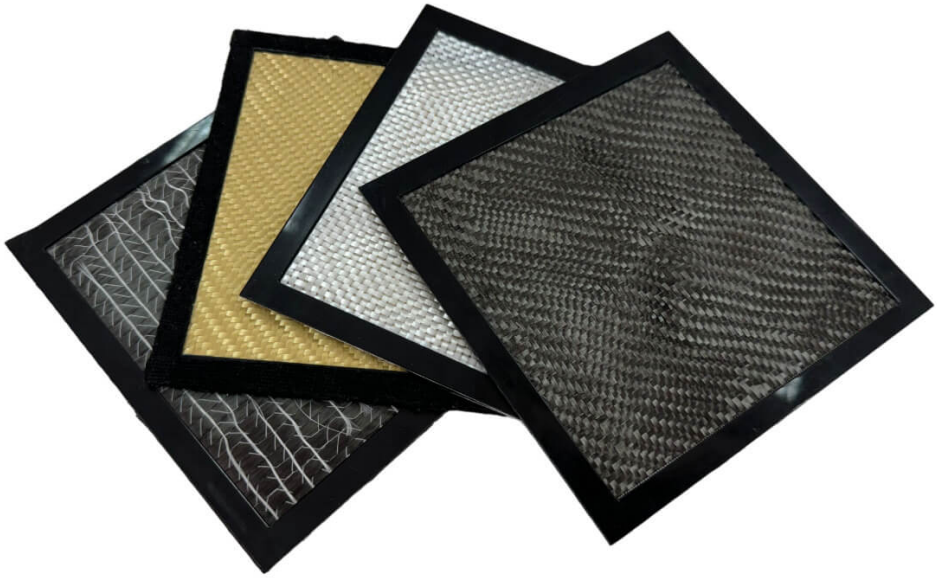
Slika 3.24 - Primer kompozitnih delova i alata za izradu

Danas, kompoziti predstavljaju jednu od najvažnijih klasa inženjerskih materijala, odmah iza čelika po značaju u industriji i domenu primene. Za ovaj veliki prodor kompozitnih materijala postoji nekoliko razloga od kojih je jedan da nude jedinstvene kombinacije krutosti, čvrstoće, žilavosti, lakoće i na sve to otpornost na koroziju. Kompozitni materijali ostavljaju mogućnost projektantima da prilagode strukturu materijala tako da ona odgovara uslovima rada dela. Na slici 3.24 je prikazano nekoliko primera delova izrađenih od kompozitnih materijala kao i alati pomoću kojih su ovi delovi izrađeni.

Za proizvodnju bespilotnih letelica često se koriste najkvalitetnija vlakna i matrice kako bi se postigla maksimalna izdržljivost, čvrstoća i smanjenje težine. Najčešće korišćeni materijali u ovoj industriji su karbonska vlakna i epoksidna smola. Karbonska vlakna su poznata po svojoj izuzetnoj čvrstoći i lakoći, što ih čini idealnim za konstrukciju bespilotnih letelica koje zahtevaju visoke performanse. Epoksidna smola, s druge strane, obezbeđuje odličnu vezu između vlakana i dodatnu čvrstoću strukture.

Za određene delove letelica, pored karbonskih vlakana, koriste se i *staklena* ili *aramidna* vlakna (različiti materijali vlakna su prikazani na slici 3.25). Staklena vlakna pružaju dobru otpornost na habanje i hemijsku stabilnost, dok aramidna vlakna, kao što je *Kevlar*, nude izuzetnu otpornost na udarce i visoku čvrstoću.

U postupku izrade kompozitnih delova koriste se ili suva vlakna koja se oblikuju i kasnije impregniraju smolom, ili prethodno impregnirana vlakna sa smolom, poznata kao *prepreg* materijali. Prepreg materijali omogućavaju precizniju kontrolu sadržaja smole i ravnomerniju distribuciju kroz materijal, što rezultira superiornim mehaničkim svojstvima.



Slika 3.25 - Različiti materijali vlakna

Alati koji se koriste u proizvodnji kompozitnih delova takođe igraju ključnu ulogu u kvalitetu finalnog proizvoda. Ovi alati se izrađuju od različitih materijala, uključujući alatni čelik, aluminijum i specijalne alatne smole. U poslednje vreme, sve više se koriste alati izrađeni 3D štampanjem od visoko otpornih smola, što omogućava brzu i ekonomičnu proizvodnju složenih geometrija.

Izbor materijala za izradu alata zavisi od broja potrebnih delova, oblika delova i mogućnosti obrade. Procesi kao što su glodanje, livenje, kovanje i 3D štampanje koriste se za proizvodnju ovih alata. Hrapavost površine alata je izuzetno bitna za kvalitet finalnih delova, pa se često poslednja obrada sastoji od poliranja radne površine kako bi se postigla što manja hrapavost.

Nakon proizvodnje pojedinačnih delova prelazi se na njihovu obradu, odstranjuju se viškovi, režu se delovi na potrebne dužine ili se opsecaju (ručno ili pomoću CNC mašina), otvaraju se otvori (na slici 3.26 je prikazano ručno bušenje otvora na kompozitnoj konstrukciji korišćenjem alata) i drugo. Delovi se zatim mogu međusobno spajati lepljenjem i može se vršiti farbanje ili lakiranje spoljnih površina. Pored ovoga mogu se vršiti i druge obrade koje se već koriste u procesima izrade delova od metala ili plastike.



Slika 3.26 - Ručno bušenje otvora na kompozitnoj konstrukciji korišćenjem alata

Za izradu nekih delova ili, kao što je već spomenuto, alata je moguće koristiti, sada široko dostupne, *3D štampače*. Postoji više tipova 3D štampača koji rade sa različitim materijalima.

Mnogi delovi tokom pravljenja prototipova mogu biti izrađeni na 3D štampačima koji rade na principu *Fused Filament Fabrication* (FFF) tehnologije, što podrazumeva da se trodimenzionalni model izrađuje nanošenjem i spajanjem slojeva materijala. Termoplastika se topi grejačem i kroz mlaznicu, određenog prečnika, otvora se nanosi na podlogu sloj po sloj uz izbor odgovarajućeg protoka plastike kroz mlaznicu. Ovo je možda trenutno i najzastupljenija tehnologija 3D štampanja delova od plastike. Druga tehnologija od značaja je stereolitografija (MSLA, Masked Stereolithography) za izradu delova od specijalnih smola.

Na sledećim slikama (3.27 i 3.28) su prikazani industrijski 3D štampači FFF i MSLA tehnologije. Mnoge kompanije danas imaju čitavu farmu 3D štampača koji neprestano rade.



Slika 3.27 - Industrijski štampači FFF tehnologije



Slika 3.28 - Industrijski štampači MSLA tehnologije

Pretvaranje 3D modela u format koji 3D štampači koriste za stvaranje objekata je moguće uraditi u specijalizovanim programima kao što je `PrusaSlicer` [101]. Ovo je besplatan program otvorenog koda koji omogućava pretvaranje 3D modela u formate koje razumeju i `FFF` i `MSLA` 3D štampači. Program je baziran na starijem programu pod nazivom `Slic3r` [102] koji je u ovoj verziji dalje proširen i unapređen.

Osim 3D štampanja delova od plastike i različitih smola, danas su sve dostupniji i procesi 3D štampanja od različitih metala kao što su aluminijum, čelik, nerđajući čelik ili titanijum. Na sledećoj slici (slika 3.29) su prikazani delovi od metala dobijeni 3D štampanjem. Različite tehnologije su predviđene za delove različitih dimenzija i mogu ostvariti različitu tačnost kao što se može videti i na slici. Element na slici je deo zgloba koji predstavlja vezu ruke multikoptera i trupa.



Slika 3.29 - Primer metalnih delova dobijenih 3D štampanjem

Sa druge strane, kada su potrebni plastični delovi u serijskoj proizvodnji proizvoda, zbog efikasnosti i mehaničkih karakteristika delova, najčešće se koriste procesi kao što je brizganje plastike. Brizganje se ne koristi često u procesu istraživanja i razvoja jer je zbog potrebnih alata i mašina skup proces, ali postoje i određene mogućnosti za ručno brizganje sa jeftinim alatima što ovaj proces približava istraživačima.

Osim što se 3D modeli mogu formirati korišćenjem odgovarajućih programskih paketa, sve je popularnije korišćenje 3D skenera za formiranje modela koji se potom mogu koristiti u najrazličitije svrhe. 3D skeniranje je postupak prikupljanja trodimenzionalnih informacija o obliku i dimenzijama objekata ili okruženja, pri čemu se informacije čuvaju u vidu 3D modela. Ova tehnologija omogućava preciznu i detaljnu analizu fizičkih struktura, bilo da su u pitanju sitni mehanički delovi ili kompletne građevine ili geografske površine. Primena 3D skeniranja je raznovrsna, od industrijske metrologije, preko arhitektonskog planiranja, geodetskih istraživanja, pa sve do uvođenja kompleksnih modela u virtuelnu stvarnost i sisteme za simuliranje.

Postoji više različitih tehnologija 3D skeniranja, koje se međusobno razlikuju po principu rada, tačnosti, brzini i prilagođenosti različitim uslovima. Najčešće korišćeni pristupi obuhvataju lasersko skeniranje, strukturisano svetlo, fotogrametriju i hibridne metode koje kombinuju više tehnologija. Odabir odgovarajućeg uređaja zavisi od niza faktora, uključujući zahtevani nivo tačnosti, veličinu i složenost objekta ili terena, brzinu skeniranja, kao i uslove pod kojima se obavlja merenje (spoljašnji ili unutrašnji prostor, nivo osvetljenja, prisustvo reflektujućih ili transparentnih površina). Bitni parametri pri izboru uključuju rezoluciju

(gustinu tačaka u 3D oblaku), preciznost (odstupanje između izmerenih i realnih dimenzija), domet (maksimalna udaljenost na kojoj sistem može uspešno detektovati površinu), kao i mobilnost i autonomiju sistema.

Laserski skeneri se ističu po visokoj rezoluciji i velikom dometu, dok strukturisano svetlo omogućava izuzetno detaljne skenove objekata manjih dimenzija. Fotogrametrija, sa druge strane, koristi digitalne fotografije snimljene iz različitih uglova, a potom programski rekonstruiše 3D model. Ovakvi sistemi mogu biti montirani i na bespilotne letelice, što omogućava automatsko prikupljanje podataka i modeliranje širokih geografskih površina.

Na slici 3.30 je prikazan postupak skeniranja elise za multikopter pomoću uređaja *Mole* proizvođača *3DMakerpro*, na računaru u pozadini se vidi i program *JMStudio* kojim se vrši skeniranje. Nakon skeniranja se može koristiti program `MeshLab` za dalju obradu.

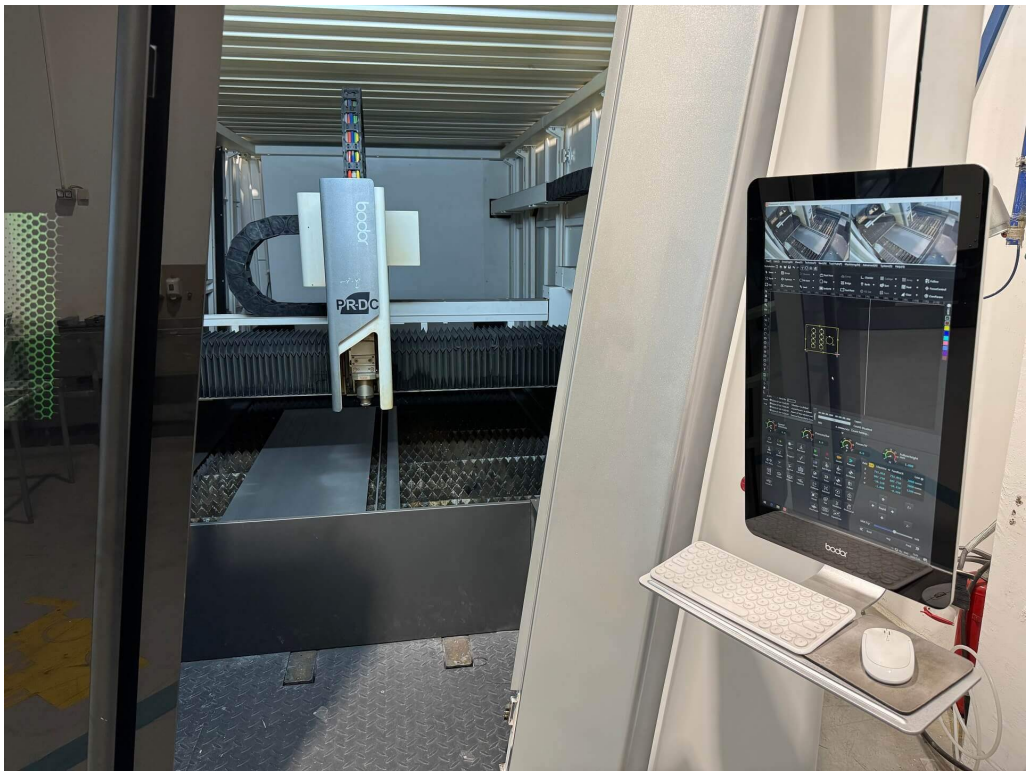


Slika 3.30 - Primer 3D skeniranja elise za multikopter

3.6 Proizvodni procesi i obrada materijala

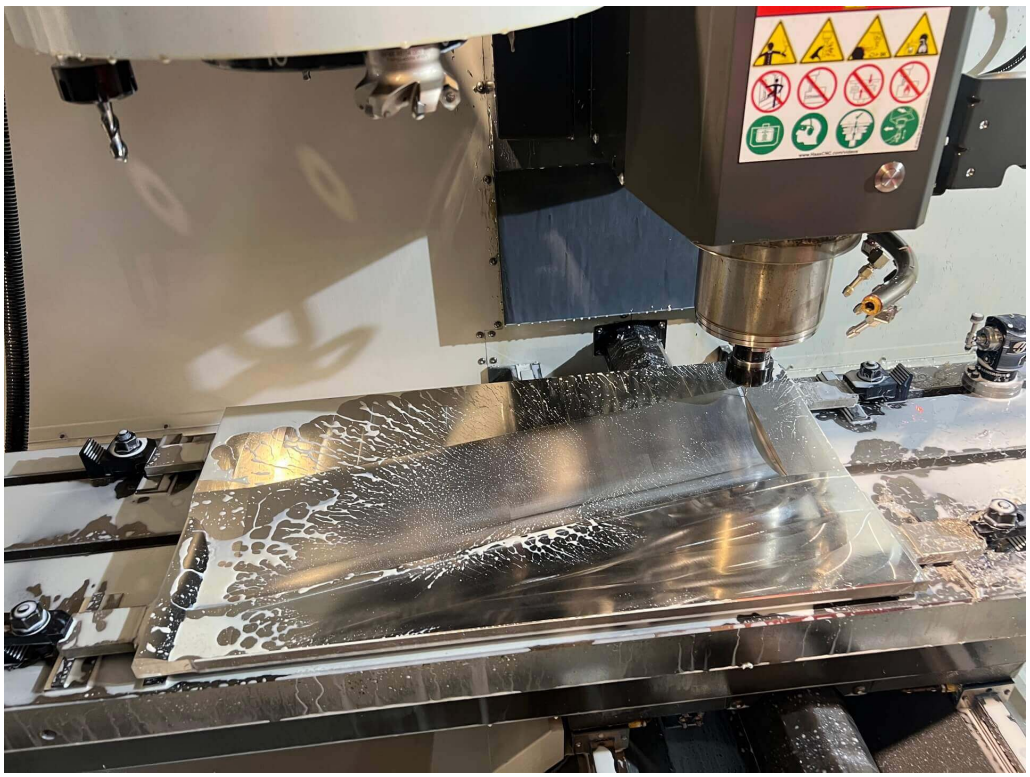
Da bi se napravio proizvod kao što je bespilotna letelica ili simulator leta, neophodno je ovladati nizom različitih mašinskih obrada, i to različitih materijala.

Za izradu konstrukcije od metala, CNC mašine za rezanje laserom (kao ona prikazana na slici 3.31) omogućile su daleko bržu, efikasniju i precizniju izradu. Konstrukcije se od samog početka projektuju tako da mogu jednostavno da se izrađuju, koriste se pločasti materijali koji se režu laserom i zatim zavaruju postupcima, kao što su `MIG`, `MAG` ili `TIG`. Karakteristično za projektovanje je da se delovi izrađuju sa oblicima koji u isto vreme omogućavaju i lako pozicioniranje tokom zavarivanja, što omogućava precizne konačne sklopove.



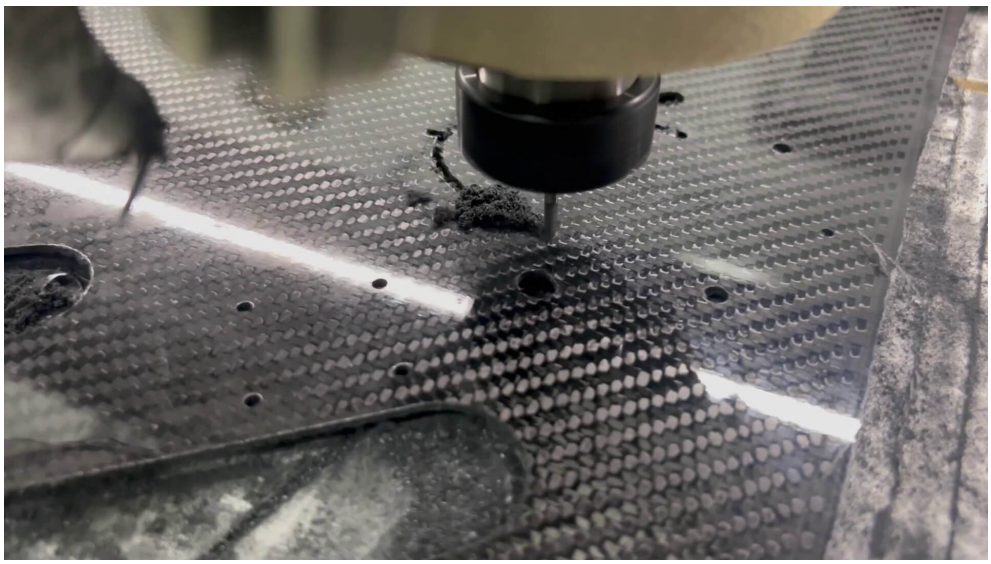
Slika 3.31 - CNC mašina za rezanje laserom

Nezaobilazna mašinska obrada za potrebe izrade kalupa za delove od kompozitnih materijala je CNC glodanje. U postupku izrade alata se najčešće koriste CNC glodalice sa tri ili pet osa. Na slici 3.32 prikazana je obrada alata od aluminijuma za polukrilo bespilotne letelice. Pored izrade alata na ovim mašinama se izrađuju i drugi elementi od metala za letelice kao što su na primer kućišta za elektroniku (kućište autopilota, radio modula i drugih elemenata).



Slika 3.32 - Izrada alata na CNC glodalici

Za razliku od CNC glodalica, koje su veoma robusne mašine projektovane da izdržavaju ekstremne radne uslove pri obradi metala, CNC ruteri su mašine-alatke projektovane za glodanje mekših materijala, kao što su drvo ili plastike. Na slici 3.33 je prikazan CNC ruter kako vrši opsecanje kompozitne ploče. Ruteri se često koriste za obradu delova od kompozitnih materijala, kao i za obradu alatnih plastika za potrebe izrade alata.



Slika 3.33 - Opsecanje kompozitne ploče na CNC ruteru

Delovi od plastičnih materijala koji se proizvode u serijama najčešće se izrađuju procesom brizganja, koji omogućava masovnu proizvodnju uz visok nivo preciznosti i ponovljivosti. Ova tehnologija je idealna za izradu složenih oblika i omogućava smanjenje troškova po jedinici proizvoda pri velikim serijama. Primer strukture bespilotne letelice izrađene od brizgane plastike prikazan je na slici 3.34.



Slika 3.34 - Primer strukture letelice izrađene od brizgane plastike

Za brzu izradu ultralakih konstrukcija jednostavnog oblika koriste se i ekspandirani plastični materijali, kao što je ekspandirani polipropilen (EPP). EPP se odlikuje izuzetnom elastičnošću i otpornošću na udarce, što ga čini idealnim za konstrukciju letelica koje zahtevaju nisku masu i visoku izdržljivost. Primer strukture bespilotne letelice izrađene od ekspandiranog polipropilena prikazan je na slici 3.35.



Slika 3.35 - Primer strukture letelice izrađene od ekspandiranog polipropilena

3.7 Štampane ploče za elektroniku

Kao što je kasnije prikazano u ovoj disertaciji, kompletan sistem se bazira na digitalnoj elektronici. Da bi se napravio takav sistem neophodno je projektovati štampanu ploču (PCB - *Printed Circuit Board*) sa odgovarajućim komponentama, ovladati izradom štampanih ploča, i na kraju njihovim sklapanjem.

3.7.1 Projektovanje štampanih ploča

Program KiCad je besplatan alat otvorenog koda namenjen za izradu šema i izradu tehničke dokumentacije za izradu štampanih ploča [103]. Pored toga, program ima alate za simulacije i neke inženjerske proračune.

U programu je veoma jednostavno definisati šemu komponenti, kao i izgled njihovog pakovanja, kako bi se formirala štampana ploča. Kada su definisane sve komponente, i u pogledu šeme i u pogledu fizičkog pakovanja, moguće je preći na međusobno povezivanje komponenti vodovima u skladu sa šemom. Cilj je formirati štampanu ploču koja zadovoljava sve zahteve (i u pogledu uslova rada) i maksimalno smanjiti potrebno ožičenje u konačnom proizvodu.

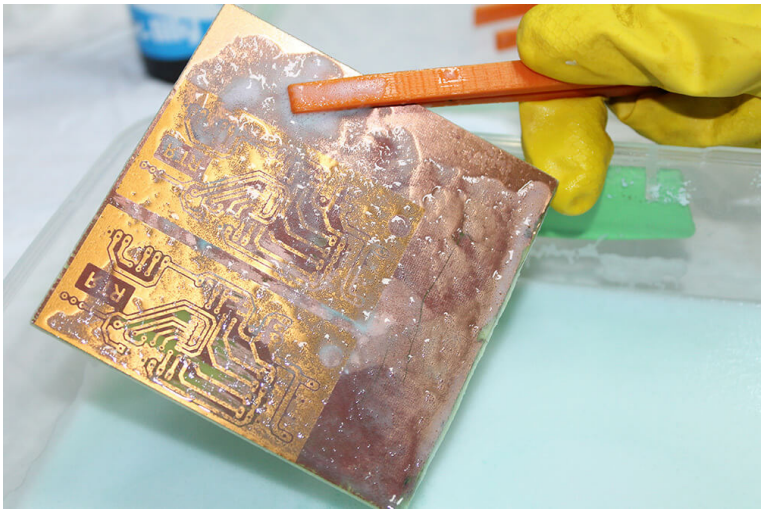
3.7.2 Izrada štampanih ploča

Štampane ploče se, pre svega, razlikuju po broju bakarnih slojeva kao i njihovom nosaču. Najčešći nosač je staklolit, ali postoje i drugi kao što su na primer savitljivi nosači za štampane ploče koje treba da budu fleksibilne. Postoje različite debljine bakarnih slojeva kao i različite

debljine njihovih nosača. Finalni oblik štampane ploče može biti slobodno definisan i najčešće se dobija opsecanjem. Pored opsecanja, često se koriste bušenje otvora i međusobno spajanje različitih slojeva vijama.

Nakon što je projektovana štampana ploča, nju je potrebno i izraditi. Osnovni postupak izrade štampane ploče je foto postupak. U okviru najjednostavnijeg foto postupka za izradu jednoslojne štampane ploče se na paus papiru odštampa šema vodova, zatim se ploča, sa slojem bakra i fotoosetljivim premazom osvetljava određeno vreme. Fotoosetljiv sloj u zonama iza odštampanih vodova nije osvetljen istim intenzitetom kao ostatak sloja, zato se nakon stavljanja u rastvor foto razvijača (*Natrijum hidroksid, NaOH*) ne uklanja lak u zonama koje su manje osvetljene.

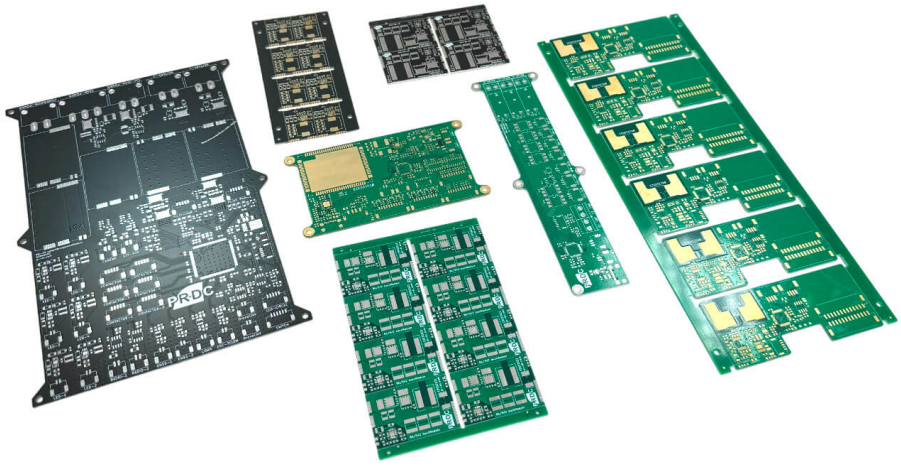
Nakon što je skinut lak, ploča se može staviti u nemetalnu posudu u kojoj je rastvor vodonik peroksida (H_2O_2) i hlorovodonične kiseline (HCl) koji nagriza bakar u zonama na kojima nema laka i na ploči ostavlja samo željene bakarne vodove.



Slika 3.36 - Vađenje ploče iz rastvora

Nakon ovog procesa, preostaje još finalna obrada koja podrazumeva vađenje ploče iz rastvora (prikazano na slici 3.36), ispiranje, uklanjanje preostalog laka, isecanje ploče i bušenje otvora za komponente. Na ploču je na kraju moguće dodati i zaštitne slojeve, kao i slojeve sa različitim oznakama koji se takođe osvetljavaju.

Na slici 3.37 su prikazane različite štampane ploče koje su izrađene naprednim industrijskim postupkom koji je vrlo sličan prethodno prikazanom.



Slika 3.37 - Industrijski proizvedene štampane ploče

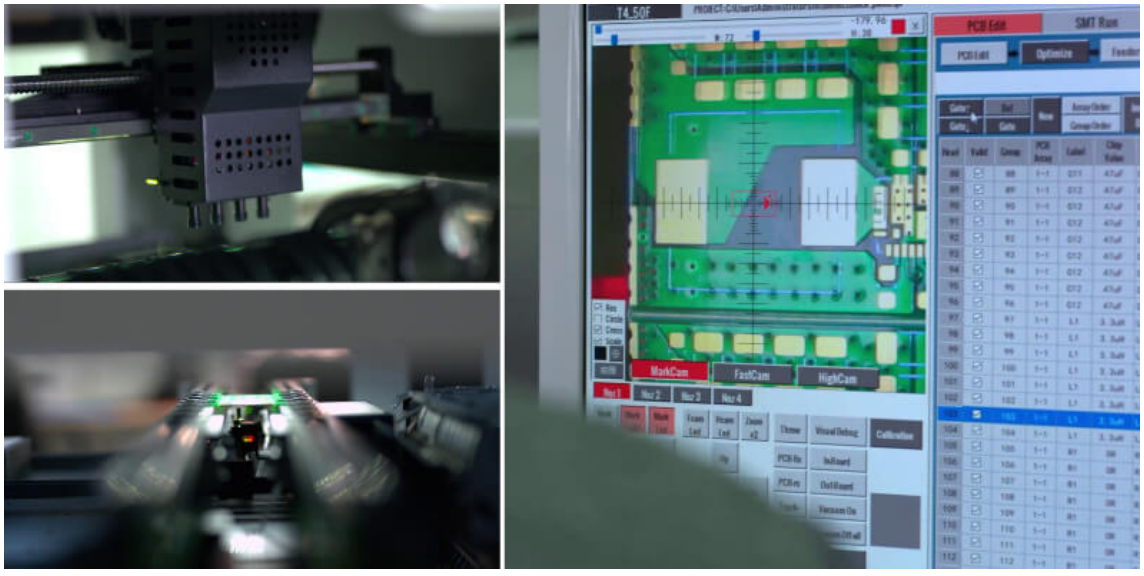
3.7.3 Sklapanje

Nakon što su dostupne štampane ploče i sve komponente koje je potrebno spojiti sa štampanom pločom, može se preći na sklapanje. Sklapanje može biti ručno, poluautomatsko ili automatsko. Na slici 3.38 je prikazana CNC mašina za automatsko izuzimanje i postavljanje komponenti na štampanu ploču. Operater na mašini postavlja komponente u za to predviđene nosače, zatim u mašinu unosi tehničku dokumentaciju za štampanu ploču i potom mašina automatski uzima komponente, koriguje njihov položaj na osnovu kamere i precizno ih postavlja na za njih određeno mesto.



Slika 3.38 - CNC mašina za izuzimanje i postavljanje komponenti na štampanu ploču

Na slici 3.39 su prikazane vakuumske hvataljke kojima se prenose komponente, šine kojima se pomeraju štampane ploče, kao i program putem koga operater kontroliše rad mašine.



Slika 3.39 - Postupak postavljanja komponenti na štampanu ploču

Postoje različiti načini kako se komponente mogu montirati na štampanu ploču i u skladu sa tim postoje različite metode za lemljenje komponenti. Pre nego što se štampana ploča ubaci u mašinu za izuzimanje i postavljanje prvo se nanosi pasta za lemljenje pomoću specijalnih šablona za nanošenje paste, tako da se pasta nanosi samo na delove štampane ploče gde se leme komponente. Nakon što su postavljene sve komponente štampana ploča se prenosi u peć za lemljenje. Peć se programira tako da temperaturni profil lemljenja zadovoljava tehničku dokumentaciju i da ne dođe do oštećenja komponenti.

Komponente koje se montiraju na površinu štampane ploče je moguće i ručno zalemiti, tako se najčešće leme komponente koje svojim pinovima prolaze kroz štampanu ploču. Na kraju kada je sklopljena elektronika, kada su zalemljene sve komponente na štampanu ploču, preostalo je još samo da se nanese zaštitni sloj.

3.7.4 Verifikacija

Osnovna verifikacija se vrši instrumentima kao što je multimetar kojim se prvo proverava da li su određeni vodovi u kontaktu. Proverava se da li postoji kratak spoj na napajanju, a zatim i da li su signali na odgovarajući način sprovedeni.

Zatim se na ploču dovodi napajanje preko laboratorijskog napajanja i vrše se dalja testiranja. Na početku se može dovesti napajanje niskog napona, koji i u slučaju postojanja greške neće oštetiti većinu komponenti, i vršiti nadzor elektronike pomoću termalne kamere (prikazano na slici 3.40). Ukoliko postoji greška na štampanoj ploči ona će se najčešće pokazati kao izvor toplote.



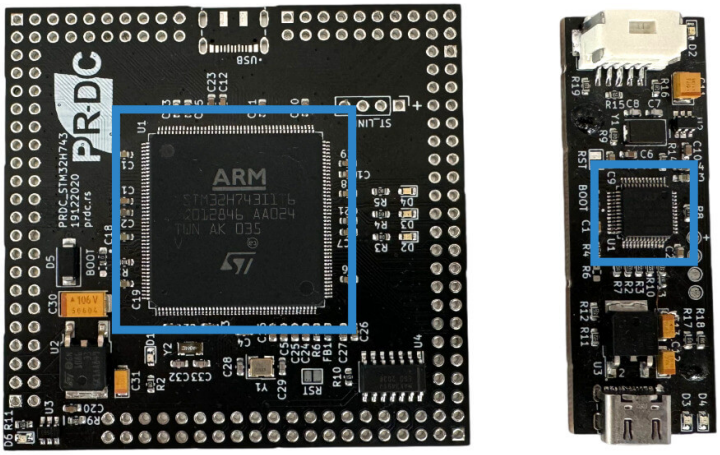
Slika 3.40 - Verifikacija elektronike sa laboratorijskim napajanjem i termalnom kamerom

Štampane ploče za serijsku proizvodnju se projektuju tako da se ostavljaju dostupne tačke na kojima se vrši automatizovano testiranje. Izrađuju se posebni alati sa pinovima kojima se dodiruju tačke za testiranje i pokreće se automatski program koji proverava signale. Štampane ploče za elektroniku se mogu testirati na vibracije i udare na vibro stolovima, kao i na povišene i snižene temperature sa i bez vlage u laboratorijskim klima komorama.

3.8 Mikrokontroleri i komunikacije

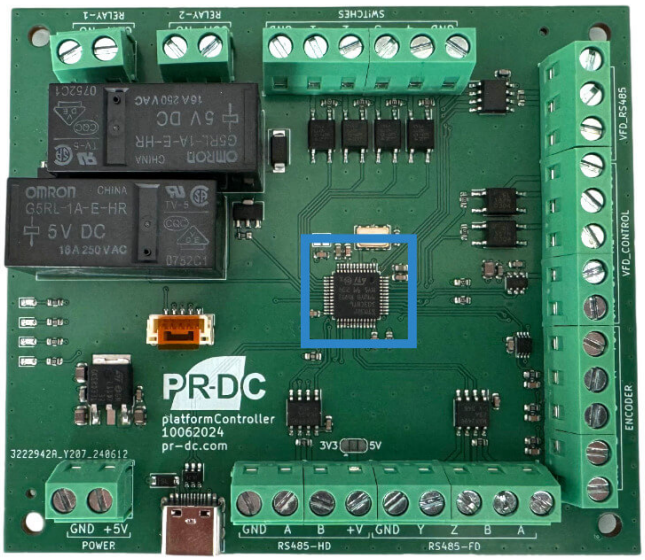
Mikrokontroler je elektronsko integrisano kolo koje sadrži procesorsku jedinicu, memoriju i različite periferne uređaje, i danas predstavlja nezamenljivu komponentu gotovo svake mašine.

Na slici 3.41 su prikazane razvojna ploča, bazirana na mikrokontroleru iz familije `STM32` kompanije *STMicroelectronics*, na levoj strani, i desno pored nje, programator koji na sebi takođe ima mikrokontroler. Programator se koristi za upisivanje novog programa u memoriju mikrokontrolera, kao i za praćenje rada mikrokontrolera.



Slika 3.41 - Razvojna ploča sa mikrokontrolerom (levo) i programator za mikrokontroler (desno)

Za razliku od razvojne ploče, na kojoj su skoro svi ili baš svi pinovi mikrokontrolera izvučeni na veće pinove kako bi im se lakše pristupilo tokom razvoja proizvoda i testiranja, na slici 3.42 je prikazana elektronika sa mikrokontrolerom u centru koja ima specijalnu namenu i zato na štampanoj ploči ima mnoge druge komponente koje uz mikrokontroler omogućavaju ostvarivanje željene funkcionalnosti.



Slika 3.42 - Primer elektronike sa mikrokontrolerom

Pored mikrokontrolera, na štampanoj ploči se najčešće nalazi barem još i regulator napona za njega i druge periferne uređaje, kao i neke pasivne komponente kao što su kondenzatori i

otpornici. Na prethodnoj slici je moguće videti još i USB konektor, terminale pomoću kojih se povezuju žice, različita integrisana kola i dva releja.

Mikrokontroler je izuzetno značajan jer jednostavno omogućava da se napisanim programom upravljaju mehanički sklopovi, kontrolišući njihove komponente, poput motora, na osnovu unetih instrukcija i podataka kao i na osnovu podataka dobijenih sa senzora, što omogućava automatizaciju i precizno upravljanje različitim uređajima. Mikrokontroleri mogu biti povezani sa složenijim računarima direktno ili preko interneta i tako postaju deo mnogo većeg sistema. Najčešće mikrokontroler na osnovu određenih ulaznih informacija (prekidači ili tasteri na digitalnim ulazima, različiti analogni senzori na analognim ulazima ili podaci dobijeni preko komunikacija npr. preko USB-a od računara) pokreće izvršavanje nekog dela programa koji može kontrolisati digitalne izlaze (npr. uključivanje ili isključivanje uređaja) ili aktuatore (npr. servo motore za otklon krmila visine).

Za komunikaciju sa senzorima i aktuatorima, za komunikaciju između više mikrokontrolera ili drugih integrisanih kola, za komunikaciju između mikrokontrolera i računara ili između više računara se najčešće koriste:

- *UART* (Universal Asynchronous Receiver/Transmitter) - serijski komunikacioni protokol koji omogućava asinhronu komunikaciju između dva uređaja. Koristi se za jednostavnu i efikasnu razmenu podataka preko najmanje tri žice od kojih su dve žice TX (transmit - slanje) i RX (receive - primanje) za signal i treća je zajednička referenca GND. Zbog svoje jednostavnosti, UART se široko koristi sa mikrokontrolerima, GNSS modulima i različitim senzorima. Podaci se šalju redom bit po bit bez potrebe za sinhronizacijom putem takta signala, što olakšava implementaciju ali može dovesti do problema sa greškama u prenosu pri većim brzinama ili dužim linijama.

- *USB* (Universal Serial Bus) - univerzalni serijski komunikacioni protokol koji se najčešće koristi za povezivanje računara i periferijskih uređaja. Omogućava visoke brzine prenosa podataka i istovremeno napajanje uređaja, čineći ga izuzetno praktičnim za širok spektar primena. Ovaj protokol je standard u industriji za povezivanje i komunikaciju. USB kao i UART može da se implementira sa najmanje tri žice (D+, D- i GND).

- *SPI* (Serial Peripheral Interface) - sinhroni serijski komunikacioni protokol koji omogućava brzu razmenu podataka između mikrokontrolera i perifernih uređaja kao što su uglavnom senzori, memorije i ekrani. Koristi najmanje pet žica, zajednička referenca GND i četiri osnovne linije za signal: SCK (clock - takt), MOSI (master out, slave in - kontroler izlaz, uređaj ulaz), MISO (master in, slave out - kontroler ulaz, uređaj izlaz) i SS (slave select, izbor uređaja). SPI omogućava pun-dupleks komunikaciju, što znači da se podaci mogu istovremeno slati i primati. Zbog svoje brzine i efikasnosti, SPI je idealan za aplikacije gde je brzina prenosa podataka kritična.

- *I2C* (Inter-Integrated Circuit) - sinhroni serijski komunikacioni protokol koji se koristi za povezivanje mikrokontrolera sa perifernim uređajima kao što su uglavnom senzori, memorijski čipovi i ekrani. Koristi tri žice, zajednička referenca GND i dve osnovne linije za signal: SDA (data line, podaci) i SCL (clock line, takt), preko kojih se vrši komunikacija između kontrolera i uređaja. I2C podržava adresiranje uređaja na istoj liniji, što omogućava jednostavno povezivanje više uređaja sa jednim kontrolerom. Protokol funkcioniše po principu pisanja i čitanja registara na određenim adresama. Ovaj protokol je popularan zbog svoje jednostavnosti i efikasnosti, ali je sporiji u poređenju sa SPI protokolom.

- *CAN* (Controller Area Network) - robustan serijski komunikacioni protokol projektovan za povezivanje različitih elektronskih jedinica u vozilima i industrijskim sistemima. Omogućava višemrežnu komunikaciju bez centralnog uređaja, gde svaki uređaj može slati i primati poruke. `CAN` koristi diferencijalni signal (tri žice: `H`, `L` i `GND`), što ga čini otpornim na elektromagnetne smetnje i pogodnim za upotrebu u zahtevnim okruženjima. Ovaj protokol je poznat po svojoj sposobnosti detekcije i ispravljanja grešaka, što ga čini izuzetno pouzdanim za kritične aplikacije.

- *RS485 Pun-dupleks* - serijski komunikacioni protokol koji omogućava punu-dupleks komunikaciju, što znači da podaci mogu biti poslati i primljeni istovremeno. Koristi diferencijalni signal (pet žica: `A`, `B`, `Z`, `Y` i `GND`), što mu omogućava pouzdan prenos podataka na velike udaljenosti i u prisustvu elektromagnetnih smetnji. `RS485 pun-dupleks` je idealan za industrijske mreže i aplikacije gde je potrebna visoka pouzdanost i brzina prenosa podataka sa istovremenim slanjem i primanjem podataka. Ovaj protokol podržava povezivanje više uređaja na istu mrežu, omogućavajući efikasnu višemrežnu komunikaciju.

- *RS485 Polu-dupleks* - serijski komunikacioni protokol sličan sa `RS485 pun-dupleks`, ali podržava samo polu-dupleks komunikaciju, što znači da se podaci mogu ili slati ili primati u bilo kom trenutku, ali ne istovremeno. Uređaj mora da vodi računa o tome kada se primaju podaci, a kada se šalju, tako da prilikom slanja mora da vodi računa da pređe u mod prijema nakon što su svi podaci poslati, ili može doći do gubitka podataka. Ovaj protokol je često korišćen u industrijskim sistemima i aplikacijama gde je potreban jednostavan i robustan komunikacioni sistem. `RS485 polu-dupleks` takođe koristi diferencijalni signal (tri žice: `A`, `B`, i `GND`), što mu daje otpornost na smetnje i omogućava pouzdan prenos podataka na velike udaljenosti, često do nekoliko kilometara.

- *PWM* (Pulse Width Modulation) - digitalni signal koji se najčešće koristi za kontrolu snage, pozicioniranja i brzine električnih uređaja, poput motora, LED-ova i ventilatora. Signal se sastoji od niza impulsa sa promenljivom širinom, gde se širina impulsa (duty cycle) koristi za kontrolu količine isporučene energije. Zbog svoje efikasnosti i jednostavnosti implementacije (samo dve žice: `GND` i `S`), `PWM` je široko korišćen u elektronici i automatizaciji.

- *IP* (Internet Protocol) - osnovni protokol za prenos podataka na internetu i mnogim drugim mrežama. Ovaj protokol definiše strukturu i način prenosa podataka, odnosno paketa, između različitih uređaja u mreži. Svaki uređaj koji koristi `IP` mora imati jedinstvenu `IP` adresu, koja se koristi za identifikaciju i adresiranje tokom prenosa podataka. `IP` adrese mogu biti u obliku `IPv4` (32-bitne adrese, npr. `192.168.1.1`) ili `IPv6` (128-bitne adrese, npr. `2001:0db8:85a3:0000:0000:8a2e:0370:7334`). `IP` pruža osnovne metode za slanje i primanje paketa, kao što su fragmentacija i ponovno sastavljanje paketa, rutiranje, detekcija i ispravljanje grešaka. Međutim, `IP` ne obezbeđuje pouzdanost prenosa podataka, već to ostavlja protokolima višeg nivoa kao što je `TCP`.

- *TCP* (Transmission Control Protocol) - jedan od osnovnih protokola za `IP` koji osigurava pouzdanu i direktnu komunikaciju između uređaja u mreži. `TCP` pruža mehanizme za segmentaciju, kontrolu toka, kontrolu grešaka i ponovno sastavljanje podataka, što ga čini ključnim za pouzdan prenos podataka. Funkcioniše tako što uspostavlja vezu između pošiljaoca i primaoca, koristeći trostruko rukovanje za uspostavljanje i prekidanje veze. Tokom prenosa podataka, `TCP` garantuje da će svi paketi podataka stići do odredišta u

ispravnom redosledu i bez grešaka, koristeći mehanizme kao što su provera kontrolnog zbira i potvrda prijema.

- *UDP* (User Datagram Protocol) - jedan od osnovnih protokola za IP koji omogućava brzu i jednostavnu komunikaciju između uređaja u mreži. Za razliku od TCP-a, UDP je protokol bez uspostavljanja veze i ne garantuje pouzdanost prenosa podataka. UDP šalje pakete pod nazivom datagrami, bez provere da li su stigli do odredišta ili u kojem redosledu, što ga čini bržim ali manje pouzdanim od TCP-a. Pored toga, UDP omogućava da se jedan paket šalje na više adresa istovremeno.

- *HTTP* (Hypertext Transfer Protocol) - osnovni protokol za razmenu informacija na www-u (World Wide Web). HTTP omogućava komunikaciju između web servera i klijenata, poput web pretraživača, koristeći model zahteva i odgovora. Klijent šalje HTTP zahtev serveru tražeći određeni resurs, kao što je web stranica ili datoteka, a server odgovara s odgovarajućim sadržajem. HTTP je protokol bez stanja, što znači da svaki zahtev i odgovor su nezavisni i ne sadrže informacije o prethodnim interakcijama. Protokol koristi URL (Uniform Resource Locator) za identifikaciju resursa na serveru i podržava različite metode zahteva, kao što su GET, POST, PUT, DELETE, koje definišu radnje koje se izvršavaju nad resursom. Sigurna verzija HTTP-a, poznata kao HTTPS, koristi SSL/TLS (Secure Sockets Layer / Transport Layer Security) enkripciju za zaštitu podataka tokom prenosa, osiguravajući poverljivost i integritet informacija.

- *Ethernet* - standard za mrežno povezivanje koji omogućava komunikaciju između uređaja u lokalnim mrežama (LAN, Local-Area Network)). Ethernet je zasnovan na standardu IEEE 802.3 i koristi kablove (najčešće uvrnute parice ili optičke kablove) za prenos podataka. Karakterističan je po svojoj sposobnosti za prenos velike količine podataka velikim brzinama, što ga čini idealnim za širok spektar aplikacija, od kućnih mreža do industrijskih sistema. Ethernet koristi koncept okvira (frame) za prenos podataka, koji uključuje MAC adresu pošiljaoca i primaoca, kao i kontrolni mehanizam za detekciju grešaka.

- *Bluetooth* - bežična tehnologija za razmenu podataka na kratkim udaljenostima između uređaja koristeći radio talase frekvencije oko 2.4 GHz. Bluetooth je namenjen za povezivanje uređaja kao što su pametni telefoni, računari, bežične slušalice, tastature, miševi, i drugi periferni uređaji. Bluetooth koristi različite profile kako bi podržao specifične tipove aplikacija, kao što su prenos zvuka, datoteka, i razmena podataka sa senzorima. Bluetooth takođe podržava sigurnosne funkcije kao što su uparivanje i enkripcija podataka, što povećava sigurnost komunikacije između uređaja.

- *Wi-Fi* (Wireless Fidelity) - bežična tehnologija za povezivanje uređaja u lokalnim mrežama (WLAN, Wireless Local-Area Network) i omogućavanje pristupa internetu bez potrebe za fizičkim kablovima. Wi-Fi koristi radio frekvencijski spektar, obično u opsezima oko 2.4 GHz i 5 GHz, za prenos podataka. Ovaj standard, zasnovan na IEEE 802.11 specifikacijama omogućava povezivanje uređaja kao što su pametni telefoni, računari, tableti, i pametni uređaji u mrežu, pružajući slobodu kretanja bez ograničenja kablovima.

Osim prethodno prikazanih komunikacija mikrokontroleri u sebi najčešće imaju i konverter analognog signala u digitalni (ADC - Analog-to-digital converter) dok neki imaju i konverter digitalnog signala u analogni (DAC - Digital-to-analog converter), što znači da pinovi mikrokontrolera mogu biti digitalni ulazi ili izlazi, odnosno analogni ulazi ili izlazi. Digitalni

izlazi se najčešće koriste za uključivanje i isključivanje, dok se digitalni ulazi koriste za detektovanje da li je nešto isključeno ili uključeno. Analogni izlazi se obično koriste za slanje neke informacije kao što je npr. željena frekvencija na frekventnom regulatoru, dok se analogni ulazi koriste za merenja kao što je npr. merenje položaja potenciometra. Mikrokontroleri imaju još i tajmere koji omogućavaju izvršavanje određenih radnji u zadatom vremenskom trenutku.

U ovoj disertaciji su prikazani mnogi senzori, aktuatori, kao i primene ulaza i izlaza mikrokontrolera. Pored toga, da bi se omogućio rad ovako kompleksnog sistema iskorišćene su sve prethodno opisane komunikacije. Kao primer, za kontrolu i akviziciju podataka sa različitih kamera se koriste `Ethernet`, `IP`, `TCP`, `UDP` i `HTTP`.

3.9 Bežični prenos podataka

Za sistem kao što je daljinsko pilotiranje letelicom je ključan element bežični prenos podataka od operatera ka letelici ali i od letelice ka operateru. Suština bežičnog prenosa podataka je u tome da ne postoji fizička veza između dva uređaja već se podaci prenose npr. putem elektromagnetnih talasa. U zavisnosti od tehnologije, snage uređaja i frekvencijskog opsega mogu se ostvariti komunikacije na različitim udaljenostima i sa različitim brzinama protoka.

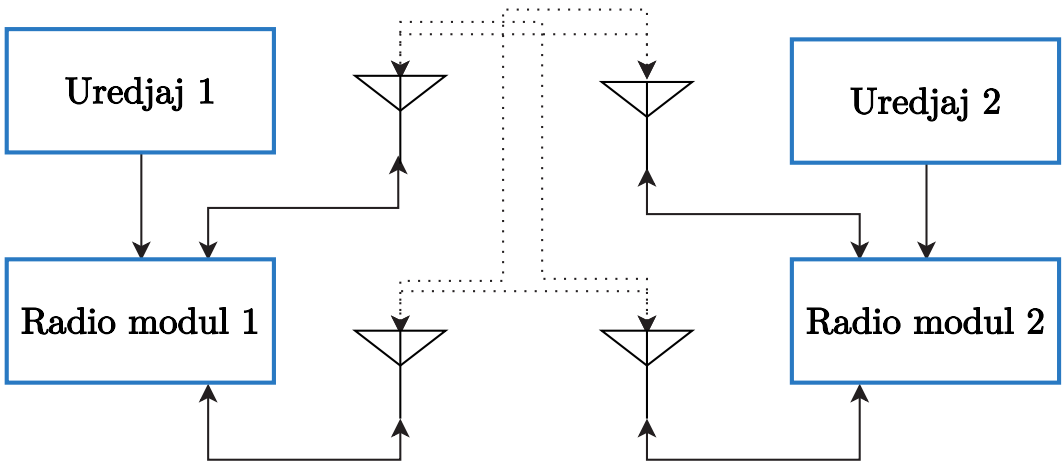
Na tržištu je moguće kupiti gotove radio module koji omogućavaju digitalnu radio vezu između dva uređaja. Neki od najistaknutijih proizvođača radio modula su *Persistent Systems*, *Silvus*, *DTC*, *Doodle Labs*, *Microhard*, *TaiSync* i drugi. Ovi radio moduli najčešće implementiraju `IP` tako da je zapravo moguće dve fizički odvojene mreže bežično povezati. Radio moduli najčešće imaju jedan frekventni opseg u kome rade (npr. od `2400` MHz do `2500` MHz), ali mogu imati mogućnost i softverskog izbora frekventnog opsega. Izbor frekventnog opsega, snage radio modula i druge karakteristike može zavisiti od važećih zakona na teritoriji upotrebe ali i od određenog tipa upotrebe.

Da bi se željena informacija ugradila na izabranu noseću frekvenciju koriste se različite modulacije signala. Od osnovnih modulacija kao što su amplitudna (`AM - Amplitude Modulation`) ili frekventna modulacija (`FM - Frequency Modulation`) pa sve do modulacija koje su danas češće u upotrebi kao što su `BPSK` (`Binary Phase Shift Keying`), `QPSK` (`Quadrature Phase Shift Keying`), `QAM` (`Quadrature Amplitude Modulation`) ili `FSK` (`Frequency-shift keying`). Većina radio modula danas koristi `OFDM` (`Orthogonal Frequency Division Multiplexing`) tehniku multipleksiranja koja deli frekvencijski opseg na više ortogonalnih podkanala (multipleksiranje), pri čemu svaki podkanal prenosi deo informacija. Kada se na `OFDM` doda još i `FEC` (`Forward Error Correction`) algoritam za ispravljanje grešaka, dobija se `COFDM` (`Coded Orthogonal Frequency Division Multiplexing`) tehnika, ovo čini komunikaciju još robusnijom i otpornijom na smetnje.

U uslovima povećanih smetnji (ili ometanja) se može koristiti tehnika prenosa naizmenično na različitim frekvencijama (`Frequency Hopping`) u kratkim vremenskim intervalima, frekvencije se menjaju prema unapred određenom nizu.

Da bi se povećala otpornost na smetnje kao i da bi se povećala brzina prenosa informacija (kapacitet) moguće je koristiti sistem sa više ulaznih i više izlaznih antena (`MIMO - Multiple Input Multiple Output`)[104]. Na slici 3.43 je prikazan blok dijagram sistema koji koristi `2xMIMO` radio vezu, što znači da ima po dve antene na oba radio modula.

`MIMO` omogućava istovremeni prenos različitih podataka preko istog frekvencijskog opsega, povećavajući propusnost bez potrebe za dodatnim spektrom. Ovo se postiže korišćenjem tehnika kao što su prostorna multipleksija (gde se različiti nizovi podataka šalju kroz različite antene) i diversifikacija (gde se isti podaci šalju kroz više antena radi poboljšanja pouzdanosti).



Slika 3.43 - Blok dijagram 2xMIMO radio veze između dva uređaja

Pored svega prethodnog, jačina signala (koju kvantifikujemo pomoću indikatora - `RSSI`, `Received Signal Strength Indicator`) zavisi i od tipa, položaja i orijentacije antena kao i od antenskih kablova koji se koriste za povezivanje antene sa radio modulom.

Za daljinski upravljane sisteme je veoma bitno praćenje jačine signala kako bi se na vreme izbegao gubitak veze. Za upravljanje sistemima je veoma bitno da je kašnjenje koje mora da postoji svedeno na minimum. Za sisteme koji se upravljaju na osnovu slike sa kamere koja se nalazi na sistemu je takođe bitno da je kašnjenje slike minimalno ali je takođe bitno da je slika što veće rezolucije i da je dostupno što više slika po sekundi.

I danas je za sisteme koji se upravljaju daljinski u realnom vremenu popularno korišćenje analognih sistema za prenos slike. Na slici 3.44 je prikazan analogni sistem za upravljanje iz prvog lica (`FPV - First-person view`). Ovo su sistemi sa izuzetno malim kašnjenjem i niskom cenom, ali sa druge strane, niskom rezolucijom zbog ograničenog protoka analognog sistema.



Slika 3.44 - Primer analognog FPV sistema

I analogni i digitalni `FPV` sistem čine naočare sa ekranom, prijemnik sa antenama, predajnik sa antenama i kamera. Čak i u okviru analognih sistema su monitori koji se koriste zapravo digitalni i vrši se konverzija analognog signala u digitalni signal koji se prikazuje na monitoru.

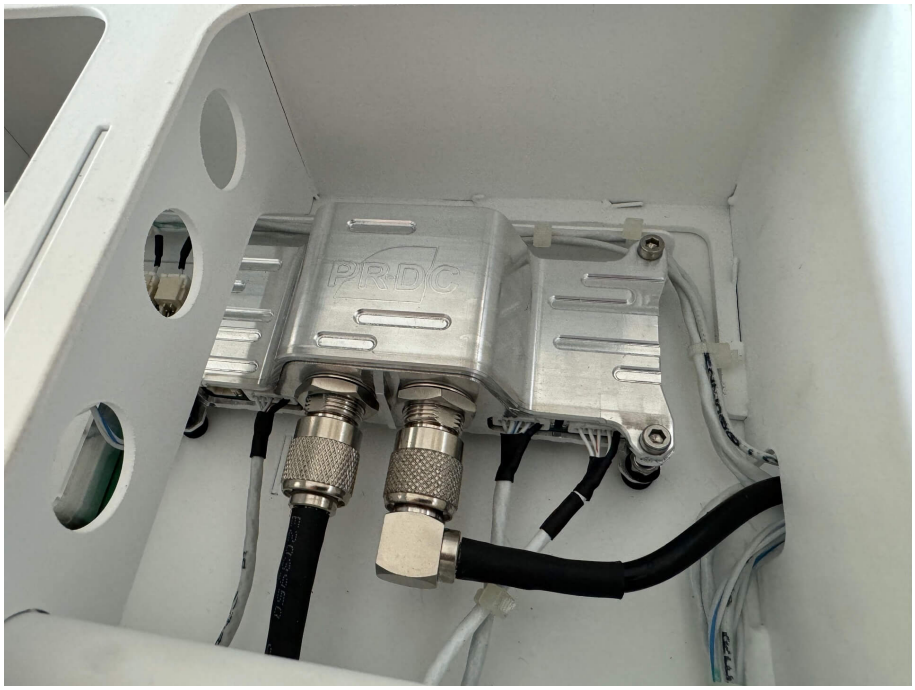
Na slici 3.45 je prikazan primer digitalnog `FPV` sistema. Razlika u odnosu na analogni sistem je što kod digitalnog sistema postoji enkodiranje i dekodiranje video-signala kako bi

se uz ograničeni protok prenosila slika visoke rezolucije (npr. rezolucija od 1280 piksela po širini i 720 piksela po visini) i to 30 ili 60 puta u sekundi. Od izbora rezolucije slike, tipa enkodera i podešavanja enkodera zavisi potreban protok bežične komunikacije. Zbog potrebe za procesorima koji su u stanju da vrše enkodiranje i dekodiranje video-signala ovi sistemi su značajno skuplji od analognih sistema.



Slika 3.45 - Primer digitalnog FPV sistema

Na slici 3.46 je prikazan primer digitalnog radio modula u aluminijumskom kućištu koji je ugrađen u letelicu.



Slika 3.46 - Primer radio modula u letelici

Zbog konstantog razvoja procesora opšte namene, specijalnih integrisanih kola za ovu namenu (ASIC - Application-Specific Integrated Circuits) kao i digitalnih integrisanih kola opšte namene (FPGA - Field-Programmable Gate Array), digitalni sistemi su sve pristupačniji. Danas se FPGA čipovi često koriste i za brze operacije obrade slike, što bi u slučaju sistema za daljinsko upravljanje letelicama omogućilo prekrivanje slike sa instrumentima i drugom simbolikom. Prednost FPGA u odnosu na ASIC je to što je FPGA moguće reprogramirati korišćenjem jezika kao što je HDL (Hardware Description Language). Na osnovu slične prednosti su sve popularniji softverski definisani radio sistemi (SDR - Software-Defined Radio) koji umesto hardverskih komponenti, kao što su modulatori, demodulatori, ili filteri koriste procesor i programe.

Pored već spomenutih proizvođača radio modula koji se baziraju na zatvorenoj tehnologiji, postoje i otvoreni projekti sa istim ili sličnim ciljem kao što su OpenHD[105], Ruby FPV[106], WFB-NG[107] ili ELRS[108]. Projekat OpenHD se bavi softverom koji omogućava bežičnu komunikaciju dugog dometa na osnovu Wi-Fi kartica baš kao i Ruby FPV[106] i WFB-NG. Ovi projekti iskorišćavaju mogućnost pojedinih Wi-Fi kartica da funkcionišu u modu za nadgledanje (monitor) mreže i direktno injektiranje (inject) paketa. Sirovi (raw) Wi-Fi paketi se šalju zaobilazeći standardne mrežne protokole i procedure koje se obično koriste za komunikaciju u Wi-Fi mrežama. Ova tehnika omogućava korisniku da generiše i šalje prilagođene pakete, uključujući pakete koji ne prate uobičajena pravila i strukture Wi-Fi komunikacije. Ove Wi-Fi kartice se najčešće povezuju na računare integrisane na jednoj ploči (SBC - Single Board Computer) koji pokreću Linux operativni sistem. Ovi računari sve komponente, uključujući procesor (CPU - Central Processing Unit), memoriju (RAM - Random-Access Memory), ulazno-izlazne portove, i druge funkcionalnosti, integrišu na jednoj ploči. Ovaj dizajn se razlikuje od tradicionalnih računarskih sistema koji često imaju odvojene komponente za procesor, memoriju, i skladištenje podataka, ali im u isto vreme ovo omogućava da su daleko manjih dimenzija i kao takve ih je moguće koristiti u situacijama kada je zahtevano minimalno zauzimanje prostora i minimalna masa.

Takođe otvoreni, projekat OpenIPC[109] ima za cilj razvoj operativnog sistema za IP kamere (koje već imaju SBC). Kombinacijom ovog projekta sa prethodno spomenutim je moguće formirati digitalni FPV sistem.

3.10 Programiranje

Programiranjem se postiže povezivanje elektronike i mehanike, omogućavajući time razvoj složenih mehatroničkih sistema koji kombinuju elektronske komponente sa mehaničkim delovima. Ovi složeni mehatronički sistemi mogu biti i letelice ili simulatori leta. Programiranje nije samo pisanje koda, već je veza koja omogućava komunikaciju i sinhronizaciju između elektronskih i mehaničkih elemenata.

Programiranje omogućava preciznu kontrolu elektronskih komponenti, kao što su mikrokontroleri, senzori i aktuatori, koji su ključni za upravljanje mehaničkim sistemima. Kroz pisanje koda, moguće je definisati ponašanje sistema uključujući pokretanje motora, upravljanje ventilima, čitanje senzorskih podataka i još mnogo toga. Senzori prikupljaju podatke iz mehaničkih sistema, kao što su položaj, brzina, sila i temperatura. Programi omogućavaju obradu ovih podataka u realnom vremenu, što je ključno za donošenje odluka i kontrolu sistema.

Sa druge strane, korišćenje određenih programskih alata omogućava simulaciju mehaničkih sistema i njihove interakcije sa elektronskim komponentama. Simulacije se kreću od najjednostavnijih, koje se oslanjaju na mnoge idealizacije prirode, do veoma složenih

proračuna koji mogu zahtevati ogromne računarske resurse. Ovo omogućava istraživačima da testiraju i optimizuju dizajn pre nego što se naprave fizički prototipi, što štedi vreme i resurse.

Programiranje omogućava kreiranje interfejsa za komunikaciju između različitih komponenti sistema. Kroz protokole i standarde kao što su `I2C`, `SPI` i `UART` programi omogućavaju razmenu informacija između senzora, kontrolera i izvršnih elemenata, osiguravajući da svi delovi sistema rade sinhronizovano.

Kroz programiranje, istraživači i inženjeri mogu razvijati napredne sisteme koji su precizni, efikasni i prilagodljivi, što je ključno za inovacije u mnogim oblastima, uključujući robotiku, automatizaciju, industrijsku kontrolu i mnoge druge. Na njima je da za svaki problem izaberu programski jezik koji je adekvatan, efikasan i aktuelan. Programski jezik treba da inženjeru omogući najbržu moguću implementaciju zamišljenog proračuna, brzo izvršavanje programa, jednostavnu analizu rezultata i interoperabilnost u cilju razmene podataka sa drugim inženjerima i njihovim radnim okruženjima. Aktuelni programski jezik je onaj sa velikim brojem ljudi i kompanija koje učestvuju u njegovom razvoju ili ga koriste, što je garant napredovanja programskog jezika kroz vreme.

Inženjer treba da bude minimalno opterećen sintaksom i vođenjem računa o za njega trivijalnim stvarima i pritom da ima maksimalnu slobodu. Količina prethodno napisanog koda i biblioteka koje je moguće integrisati mogu biti jedan od glavnih aduta pri izboru adekvatnog programskog jezika. Izuzetno je bitno koliko je dobra dokumentacija za programski jezik, koliko ima zvaničnih primera za funkcionalnosti i da li je on otvorenog koda. Programer mora biti spreman da menja stare navike, koje su nekad stečene i nakon decenijskog iskustva, jer u programiranju prosto sve jako brzo napreduje i ono što danas izgleda gotovo nemoguće za implementaciju već sutra može biti veoma jednostavno. Danas postoji sve više modela veštačke inteligencije koji omogućavaju pomoć tokom programiranja.

3.10.1 C, C++ i Fortran

`C`, `C++` i `Fortran` su programski jezici visokih performansi i opšte namene, koji se često koriste za numeričke proračune, simulacije i obradu podataka i dalje se aktivno održavaju. Ovi jezici spadaju u grupu kompajliranih jezika i idealni su za razvoj sistema koji zahtevaju visoku efikasnost i brzinu izvršavanja. Izvorni kod napisan u ovim programskim jezicima se obrađuje preprocesorom, zatim se kompajlerom i linkerom preprocesirani izvorni kod prevodi sve do izvršnog koda. Zbog performansi ovi programski jezici se najčešće koriste za programiranje mikrokontrolera koji imaju velika ograničenja u pogledu memorije. Ovi programski jezici postoje već decenijama, a obilje literature, među kojom se ističu [110]–[112], često se koristi kao referenca za najbolje prakse prilikom programiranja.

Interpretirani jezici najčešće pružaju mogućnost integracije sa kompajliranim jezicima kao što su ovi, pa je tako moguće napisati funkciju koja poziva `C++` biblioteku `Eigen`[113] za veoma efikasne operacije nad matricama, kompajlirati je, pokrenuti je sa odgovarajućim argumentima i rezultat vratiti u interpretirani jezik i na taj način drastično povećati njegove performanse.

3.10.2 GNU Octave

`GNU Octave` je programski paket koji sadrži programski jezik visokog nivoa, prvenstveno namenjen numeričkim proračunima. Aktivno se razvija i održava od strane zajednice dobrovoljaca. Program pomaže prilikom numeričkog rešavanja linearnih i nelinearnih problema. Pored ovoga, program je predviđen za matrična izračunavanja, rešavanje sistema linearnih jednačina, iscrtavanje grafika funkcija i slično. Mogućnosti programa se proširuju programima koji su pisani u programskom jeziku `GNU Octave` koji mogu da se implementiraju

zajedno sa drugim programskim jezicima kao što su C/C++, Fortran i Java. Ovaj programski jezik je dinamički i interpretirani programski jezik. Pored numeričkih izračunavanja postoji paket symbolic za simbolička izračunavanja (implementiran na osnovu sympy biblioteke programskog jezika Python). Programski jezik GNU Octave je pretežno kompatibilan sa programskim jezikom Matlab koji je zatvorenog koda.

Jednostavan grafički prikaz podataka u okviru GNU Octave-a je jedna od najznačajnijih njegovih karakteristika za inženjere. Grafički ili vizuelni prikaz podataka se može podeliti na dvodimenzionalne i trodimenzionalne grafike. Zbog toga se programski jezici GNU Octave i Matlab često koriste na univerzitetima.

3.10.3 JavaScript

JavaScript je dinamičan i interpretiran programski jezik slične sintakse kao C/C++ koji je namenjen za izradu interaktivnih web aplikacija. Zahvaljujući tome što većina kompanija na svetu koje se bave programiranjem ili učestvuju u razvoju ili koristi JavaScript u svojim proizvodima, ovaj programski jezik je postao opšti programski jezik visokog nivoa sa ogromnim brojem korisnika.

Za razvoj programskog jezika je zadužena ECMA International organizacija i u razvoju učestvuju kompanije kao što su *Mozilla*, *Google*, *Microsoft*, *Apple* i *Facebook*. Na osnovu *Google* V8 JavaScript engine-a je napravljen i projekat Node.js sa ciljem da se ovaj programski jezik ne koristi samo na strani klijenta na web-u već i na serverskoj strani ili za desktop aplikacije.

Projekti Electron, Node.js i jQuery su neki od projekata koje održava *OpenJS* fondacija (čiji su članovi takođe neke od najvećih kompanija). Electron omogućava izradu desktop aplikacija na osnovu Node.js i Chromium projekata a korišćenjem JavaScript jezika zajedno sa HTML-om i CSS-om. Sa druge strane, Apache Cordova projekat omogućava izradu mobilnih aplikacija takođe korišćenjem JavaScript-a, HTML-a i CSS-a. Svi prethodno spomenuti projekti su projekti otvorenog koda.

Danas je JavaScript jedan od najbržih interpretiranih jezika sa neverovatnim mogućnostima, zajedno sa HTML-om i CSS-om je nezamenljiv za izradu grafičkih korisničkih interfejsa (kao generalizacija osnovne namene, izrade interaktivnih web aplikacija). Postoje mnoge biblioteke kao što je Math.js za matematičke operacije, Three.js za rad sa 3D modelima i vizualizaciju, Plotly.js za izradu grafika ili Leaflet za prikaz mapa. WebAssembly omogućava da se programi pisani u programskim jezicima kao što su C/C++ ili Rust izvršavaju zajedno sa JavaScript-om, pored toga u okviru Node.js je moguće koristiti Node-API koji omogućava pozivanje C++ funkcija.

3.10.4 OpenModelica

OpenModelica je otvorena platforma za modeliranje i simulaciju kompleksnih sistema koristeći Modelica jezik. Modelica je objektno orijentisani jezik za modeliranje sajber-fizičkih sistema. Koristi se za razvoj i analizu dinamičkih sistema, omogućavajući istraživačima da simuliraju ponašanje sistema pri različitim uslovima.

Ovaj alat je posebno koristan za rad na modelima fizičkih sistema. Spajanjem predefinisanih blokova koji se vode matematičkim jednačinama, jednostavno je formirati kompleksne fizičke sisteme kao što su roboti sa mnogim segmentima i zglobovima. Nakon proračuna je jednostavno i analizirati njihova stanja. OpenModelica omogućava lako povezivanje sa drugim programima pa je tako moguće pokretati simulacije iz programa pisanih u GNU Octave-u ili JavaScript-u. Na taj način je moguće implementirati slična rešenja kao u Matlab-u koristeći Simulink. Ova integracija sa drugim programskim jezicima omogućava pokretanje niza simulacija sa drugačijim

vrednostima određenih parametara u simulaciji i pomoću optimizacionih algoritama određivati optimalne vrednosti tih parametara.

3.10.5 Arduino

Arduino je platforma koja omogućava razvoj i testiranje prototipova elektronike kroz jednostavno programiranje mikrokontrolera. Platforma uključuje posebno projektovane štampane ploče sa mikrokontrolerima, kompletno razvojno okruženje za pisanje, kompajliranje i učitavanje koda na mikrokontrolere i jezgro koje čine biblioteke za jednostavnije programiranje. Veoma brzo je moguće iskoristiti Arduino i programirati mikrokontroler za prikupljanje podataka sa senzora, kontrolu aktuatora i interakciju sa različitim hardverskim komponentama, što je ključno za brze eksperimente i praktične primene istraživanja.

Za programiranje se koristi programski jezik C++ iokvir koji definiše Sketch fajl sa ekstenzijom ino, sa predefinisanom strukturom programa. Uvedena su određena pojednostavljenja koja omogućavaju brže učenje i razumevanje procesa programiranja mikrokontrolera. Arduino daje skup biblioteka i osnovnih funkcionalnosti koje olakšavaju interakciju sa hardverom. Ovo uključuje funkcije za rad sa pinovima, tajmerima, serijskom komunikacijom, itd.

Arduino_Core_STM32 je implementacija Arduino jezgra za STM32 mikrokontrolere, omogućavajući korisnicima da koriste Arduino biblioteke za programiranje STM32 ploča. STM32 mikrokontroleri su bazirani na ARM Cortex-M jezgrama i pružaju visok nivo performansi i fleksibilnosti. Zajedno sa Arduino-om je moguće koristiti i ARM CMSIS skup standardizovanih softverskih interfejsa za ARM Cortex-M mikrokontrolere. CMSIS je razvijen od strane ARM-a kako bi olakšao razvoj softvera za njihove procesore. Tako je npr. moguće koristiti CMSIS-DSP biblioteku za matričan račun na mikrokontroleru.

3.10.6 Dokumentacija

Pisanje dokumentacije je važan deo programiranja jer omogućava da se napisani kod detaljnije objasni dodavanjem komentara na odgovarajuće linije koda. Dokumentacija olakšava razumevanje i korišćenje razvijenog softvera, omogućava lakšu saradnju među programerima, inženjerima i istraživačima i osigurava da se znanje prenosi i čuva za buduće potrebe. U najvećem broju slučajeva komentari omogućavaju i da onaj koji je prethodno napisao kod može da se vrati i podseti se zašto je taj kod baš tako napisan. U okviru komentara je moguće napisati i planove za buduće izmene.

Dobro napisana dokumentacija može značajno poboljšati produktivnost i kvalitet projekta, omogućavajući programerima i korisnicima da efikasno rade sa softverom. Pisanje dokumentacije tokom programiranja, značajno doprinosi uspehu i dugoročnoj održivosti projekta. Dokumentacija služi kao sredstvo za komunikaciju među članovima tima, pruža uputstva za korisnike, olakšava održavanje koda i pomaže novim članovima tima da se brzo uključe u rad. Kvalitetno napisana dokumentacija omogućava programerima da razumeju kako sistem funkcioniše, smanjuje vreme potrebno za rešavanje problema i povećava efikasnost timskog rada.

Jedan od najefikasnijih načina za kreiranje visokokvalitetne tehničke dokumentacije je korišćenje LaTeX-a. LaTeX je moćan alat za pripremu dokumenata, posebno cenjen u akademskim i inženjerskim krugovima zbog svoje sposobnosti da proizvodi profesionalno formatirane dokumente. LaTeX omogućava jednostavno upravljanje referencama, automatsko generisanje sadržaja, popisa figura i tabela, kao i elegantno formatiranje složenih matematičkih izraza, što ga čini idealnim za pisanje tehničke dokumentacije.

Korišćenjem LaTeX-a, programeri mogu kreirati dokumente koji ne samo da izgledaju profesionalno, već su i veoma funkcionalni. Na primer, LaTeX omogućava automatizaciju mnogih aspekata formatiranja, što smanjuje mogućnost grešaka i omogućava autorima da se fokusiraju na sadržaj dokumenta. Takođe, LaTeX-ova fleksibilnost omogućava prilagođavanje stilova dokumenta specifičnim potrebama projekta, obezbeđujući konzistentnost i lakoću čitanja.

U praksi, dokumentacija može uključivati tehničke detalje o implementaciji koda, korisničke priručnike, API reference i komentare unutar koda. Sve ove komponente su kritične za održavanje softvera i omogućavanje njegovog daljeg razvoja.

3.10.7 Verziranje koda

Verziranje koda, putem alata kao što je Git ili Mercurial, omogućava praćenje promena u kodu, kolaboraciju među timovima i upravljanje različitim verzijama softvera. Ovo je esencijalno za organizaciju i kontrolu tokom razvoja složenih i dugoročnih projekata, obezbeđujući transparentnost i kontinuitet rada. Verziranje je praktično neizbežno kada na istom kodu treba da radi najmanje dve osobe ili kada se kod razvija mesecima i godinama, kada postoji potreba za različitim verzijama jer se testira nova implementacija dela koda ili slično.

Sistemi za verziranje omogućavaju programerima da prate sve promene u kodu jer se čuva istorija razvoja projekta što omogućava i vraćanje na prethodne verzije i rad na različitim granama projekta bez međusobnog ometanja. Glavne funkcionalnosti Git-a uključuju grananje (branching), koje omogućava paralelni rad na različitim funkcionalnostima ili ispravkama grešaka, i spajanje (merging), koje kombinuje promene iz različitih grana.

Alati kao što su *GitHub*, *GitLab* ili *Bitbucket* pružaju dodatne funkcionalnosti za upravljanje projektima, kao što su praćenje problema, recenziranje koda i automatizacija procesa integracije i isporučivanja.

3.10.8 Testiranje koda

Testiranje koda je kritičan proces koji osigurava ispravnost, pouzdanost i performanse razvijenog softvera. Kroz različite vrste testiranja, uključujući jedinično, integraciono i sistemsko testiranje, moguće je identifikovati i otkloniti greške, poboljšati kvalitet koda i osigurati da softver ispunjava sve postavljene zahteve i očekivanja.

- *Jedinično testiranje* (unit testing) fokusira se na testiranje najmanjih delova koda, poput funkcija ili metoda, kako bi se osiguralo da svaki deo radi ispravno. Ove testove obično pišu sami programeri tokom ili odmah nakon razvoja koda. Alati i okviri za automatizaciju jediničnih testova omogućavaju programerima da brzo pokreću testove i dobiju povratne informacije o ispravnosti koda.

- *Integracijsko testiranje* (integration testing) proverava kako različiti moduli ili komponente sistema rade zajedno. Cilj je identifikacija problema koji se mogu pojaviti kada se različiti delovi sistema kombinuju. Integracijsko testiranje osigurava da interfejsi između modula funkcionišu pravilno i da sistem kao celina ispunjava svoje zahteve.

- *Funkcionalno testiranje* (functional testing) fokusira se na verifikaciju funkcionalnosti sistema kao celina prema specifikacijama i zahtevima. Ovaj tip testiranja simulira stvarne korisničke scenarije kako bi se osiguralo da softver ispravno obavlja svoje funkcije.

- *Testiranje performansi* (performance testing) meri kako sistem radi pod određenim opterećenjem. Ovaj tip testiranja je ključan za identifikaciju uskih grla, problema sa skalabilnošću i za optimizaciju resursa.

3.11 Manipulacija i obrada video-signala

Manipulacija i obrada video-signala su ključni zadaci u mnogim aplikacijama, a pogotovo za daljinsko upravljanje u realnom vremenu. Pored toga, ključno je za sigurnosne sisteme, video-nadzor, obradu medijskih sadržaja i druge. Ovi procesi često uključuju dekodiranje, kodiranje, filtriranje, analizu i prikazivanje video-signala. `IP` kamere često koriste `RTSP` (*Real-Time Streaming Protocol*) za prenos video-signala preko mreže, omogućavajući daljinski pristup i kontrolu, zbog toga je često potrebno implementirati metode za čitanje video-signala preko `RTSP`-a i zatim njegovo prikazivanje korisniku. Postoji niz različitih biblioteka koje omogućavaju manipulaciju i obradu video-signala, pa i onog koji se prenosi putem `RTSP`-a.

3.11.1 FFmpeg

`FFmpeg` je moćan softverski paket koji se koristi za manipulaciju i obradu video i audio zapisa. Podržava širok spektar formata i omogućava dekodiranje, kodiranje, transkodiranje, streamovanje, filtriranje i reprodukciju video-zapisa. `FFmpeg` je alat otvorenog koda i često se koristi kao biblioteka u razvoju aplikacija koje zahtevaju obradu multimedijalnih sadržaja. Njegova fleksibilnost i moćne funkcionalnosti čine ga idealnim za razne zadatke obrade video-signala. Zbog podrške širokog spektra formata idealan je u uslovima kada je potrebno konvertovati neke fajlove iz jednog formata u drugi i pritom izvršiti eventualno neophodne korekcije i dodati informacije koje fale. Primer ispravke je u slučajevima kada sistem nije završio sa pisanjem datoteke što se može dogoditi npr. u slučajevima gubitka napajanja jer postoje sistemi kod kojih se zahteva snimanje sve do krajnjih granica rada sistema.

Ovaj paket omogućava jednostavno testiranje i pomaže tokom razvoja jer omogućava da se urade brze probe (za svega par sekundi je moguće dobiti sliku sa kamere ili podatke o samom izvoru signala kao što su format piksela, veličina slike, broj slika po sekundi, potreban protok i slično) i to direktno iz terminala operativnog sistema. Kasnije se iste funkcionalnosti mogu ostvariti tako što se `FFmpeg` koristi kao biblioteka.

3.11.2 GStreamer

`GStreamer` je multimedijalni framework otvorenog koda koji omogućava kreiranje aplikacija koje obrađuju audio i video sadržaje. Pruža infrastrukturu za građenje multimedijalnih tokova (pipelines) koji mogu uključivati dekodiranje, kodiranje, filtriranje, analiziranje i prenos video-signala. `GStreamer` je vrlo fleksibilan i skalabilan, podržava veliki broj formata i protokola, te se široko koristi u industriji za razne aplikacije, uključujući video-nadzor, medijske servere i aplikacije za obradu multimedijalnih sadržaja.

Najčešće se koristi u situacijama kada je potreban prenos video-signala u realnom vremenu (sa podrazumevanim parametrima se video-signal prenosi sa minimalnim kašnjenjem). Kao i `FFmpeg`, moguće je jednostavno raditi testiranja direktno iz terminala operativnog sistema i dobijanje raznih podataka o izvoru signala.

3.11.3 Live555

Live555 je biblioteka otvorenog koda koja pruža podršku za RTSP (*Real-Time Streaming Protocol*), RTP (*Real-Time Transport Protocol*) i RTCP (*Real-Time Control Protocol*). Live555 omogućava strimovanje multimedijalnih sadržaja preko mreže i često se koristi u aplikacijama koje zahtevaju prenos video-signala u realnom vremenu, kao što su IP kamere i video-konferencijski sistemi. Biblioteka je dizajnirana da bude jednostavna za integraciju i omogućava kreiranje prilagođenih rešenja za strimovanje video i audio sadržaja.

3.11.4 OpenCV

OpenCV je biblioteka otvorenog koda za računarsku viziju, što podrazumeva analizu i obradu slike i video-zapisa. OpenCV pruža moćne alate za prepoznavanje objekata, praćenje objekata, detekciju pokreta, obradu slika, rekonstrukciju 3D modela i druge zadatke iz oblasti računarskog vida. Koristi se u raznim aplikacijama, uključujući sigurnosne sisteme, autonomna vozila, industrijsku automatizaciju i analizu sportskih događaja. OpenCV je kompatibilan sa raznim programskim jezicima, uključujući C++, JavaScript, Python i Java, što ga čini veoma fleksibilnim za integraciju u različite projekte.

Biblioteka omogućava i akviziciju slike sa kamere, preko mreže putem različitih protokola ali ono što je značajnije je to što pruža pristup različitim algoritmima koji se mogu jednostavno potom koristiti. Korišćenjem ove biblioteke moguće je pročitati sliku sa kamere, uraditi korekciju slike, detektovati objekat, označiti objekat direktno na slici, prikazati sliku korisniku i zatim pratiti kretanje objekta sa stalnim označavanjem njegovog položaja, pritom je moguće slati podatke za recimo upravljanje orijentacijom kamere tako da objekat konstantno bude u sredini slike. Program se može koristiti za određivanje dubine na osnovu slika i na osnovu tih podataka vršiti izbegavanje prepreka, može da se koristi za upoređivanje slika u cilju određivanja translatorne brzine kretanja (na osnovu toga je moguće održavati letelicu na zadatoj poziciji) ili za detekciju markera i određivanje njegovog položaja npr. za precizno sletanja.

3.12 Virtuelna, proširena realnost i haptički uređaji

U poslednjih nekoliko godina, tehnologije proširene ili augmentovane realnosti (AR) i virtuelne realnosti (VR) doživele su značajan razvoj i primenu u različitim oblastima, uključujući i one sa bespilotnim letelicama i u vazduhoplovstvu uopšte.

AR uređaji omogućavaju korisnicima interakciju sa digitalnim sadržajem koji je integrisan u stvarni svet, dok VR uređaji pružaju potpuno uranjanje u virtuelno okruženje. Glavna razlika između AR i VR uređaja leži u nivou uranjanja u digitalni sadržaj. VR uređaji, poput Oculus Quest 2 (koji je prikazan na slici 3.47), pružaju potpunu izolaciju od spoljnog sveta, stvarajući potpuno virtuelno okruženje koje korisnik doživljava kroz naočare ili kacigu sa ekranom i sa kojim vrši interakciju preko različitih haptičkih kontrolera ili direktno pomoću ruku. S druge strane, AR uređaji, kao što su Microsoft HoloLens ili mobilne aplikacije, koje koriste na primer WebXR standard, kombinuju digitalne elemente sa stvarnim okruženjem korisnika, omogućavajući interakciju sa oba sveta simultano. AR i VR uređaji omogućavaju da čovek vidi i čuje ali su potrebni dodatni haptički uređaji kako bi i osećao virtuelne objekte i kretanje. Dobro je poznato da je osećaj dodira neizbežan za razumevanje stvarnog sveta[114]. Upotreba povratne sile za unapređenje interakcije između računara i čoveka je često diskutovana.



Slika 3.47 - Uređaj Oculus Quest 2

Haptički interfejs je uređaj za povratnu informaciju koji generiše osećaj na koži i mišićima, uključujući osećaj dodira, hrapavosti, težine, toplote, vibracija, kretanja, krutosti i drugog. Može se reći da je u okviru sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi mehanizam simulatora leta zapravo haptički uređaj kojim pilot oseća kretanje letelice.

U knjizi [114] su detaljno obrađeni neurofiziološki i psihološki aspekti haptičkih uređaja kao osnov za njihovo kreiranje, u [115] je prikazan razvoj haptičkih uređaja sa analizom njihovih električnih sistema i dati su primeri. Knjiga [116] analizira upravo upotrebu haptičkih uređaja zajedno sa AR i VR tehnologijama uključujući i teleoperacije.

I AR i VR tehnologije se mogu koristiti za različite vidove obuke. Jednako kako se mogu vršiti obuke operatera za bespilotne letelice, tako se mogu vršiti i obuke tehničara koji održavaju ili čak proizvode letelice. Ovi uređaji se mogu koristiti i direktno za upravljanje letelicama jer pružaju više mogućnosti za komunikaciju sa operaterom. Umesto najčešće jednog ekrana i niza fizičkih tastera, prekidača i upravljačkih palica, korišćenjem virtuelne realnosti postaje dostupan praktično neograničen virtuelni prostor kroz koji korisnik može da se kreće. Zbog dostupnog prostora je korisnički grafički interfejs moguće bolje organizovati tako da se kontrole, instrumenti i podaci prikazuju u skladu sa značajem za operatera i bez prevelike gustine podataka. Operater do kontrola ili informacija koje su mu manje značajne može doći prostim okretanjem glave, što praktično odgovara slučaju kada se pilot nalazi u pilotskoj kabini letelice.

Slično kao i virtuelna realnost, proširena realnost se takođe može iskoristiti kako bi se operateru na bolji način prezentovali podaci. Nakon analize okruženja se u stvarni svet mogu dodati virtuelni objekti koji na primer mogu pomoći operateru da se orijentiše u prostoru. Najčešće se koriste poznati markeri koje kada prepozna sistem zamenjuje sa odgovarajućim virtuelnim modelima.

Razvojem uređaja za AR i VR se postiglo da su već danas komercijalno dostupni uređaji koji imaju dovoljnu procesorsku snagu, dovoljnu rezoluciju slike, trajanje baterije, stabilnu i sveobuhvatnu softversku podršku i dovoljnu ergonomiju i udobnost za duži period korišćenja i za uspešnu praktičnu upotrebu.

Za razvoj pilotske kabine je od ključnog značaja dobro poznavanje mogućnosti ljudskog tela, biomehaničke osnove su detaljno opisane u knjizi [117]. Treba izabrati odgovarajući položaj za telo operatera tako da može da izdrži opterećenja tokom kretanja simulatora i u isto vreme da operater može da upravlja letelicom putem kontrolnih palica, tastera i prekidača.

U radovima kao što su [118]–[120] je analizirana upotreba haptičkih uređaja u okviru pilotske kabine, pravih letelica i simulatora leta. Prikazana je upotreba programabilnih ultrasoničnih aktuatora za ostvarivanje osećaja dodira kojim bi se zamenili stvarni tasteri, prekidači, volani ili kontrolne palice. Ovo je značajno jer je moguće programom, gotovo trenutno, promeniti raspored tastera unutar pilotske kabine. Istražuje se upotreba kontrolnih palica i pedala sa haptičkom povratnom spregom što bi simuliralo direktno upravljanje kontrolnim površinama pri kome se sa povećanjem otklona komandnih površina povećava i potrebna sila na palici. Cilj je omogućiti da pilot oseti povećanje potrebne sile na palici ali samo u meri u kojoj ne dolazi do zamora samog pilota.

3.13 JSLAB - okruženje za numeričke proračune

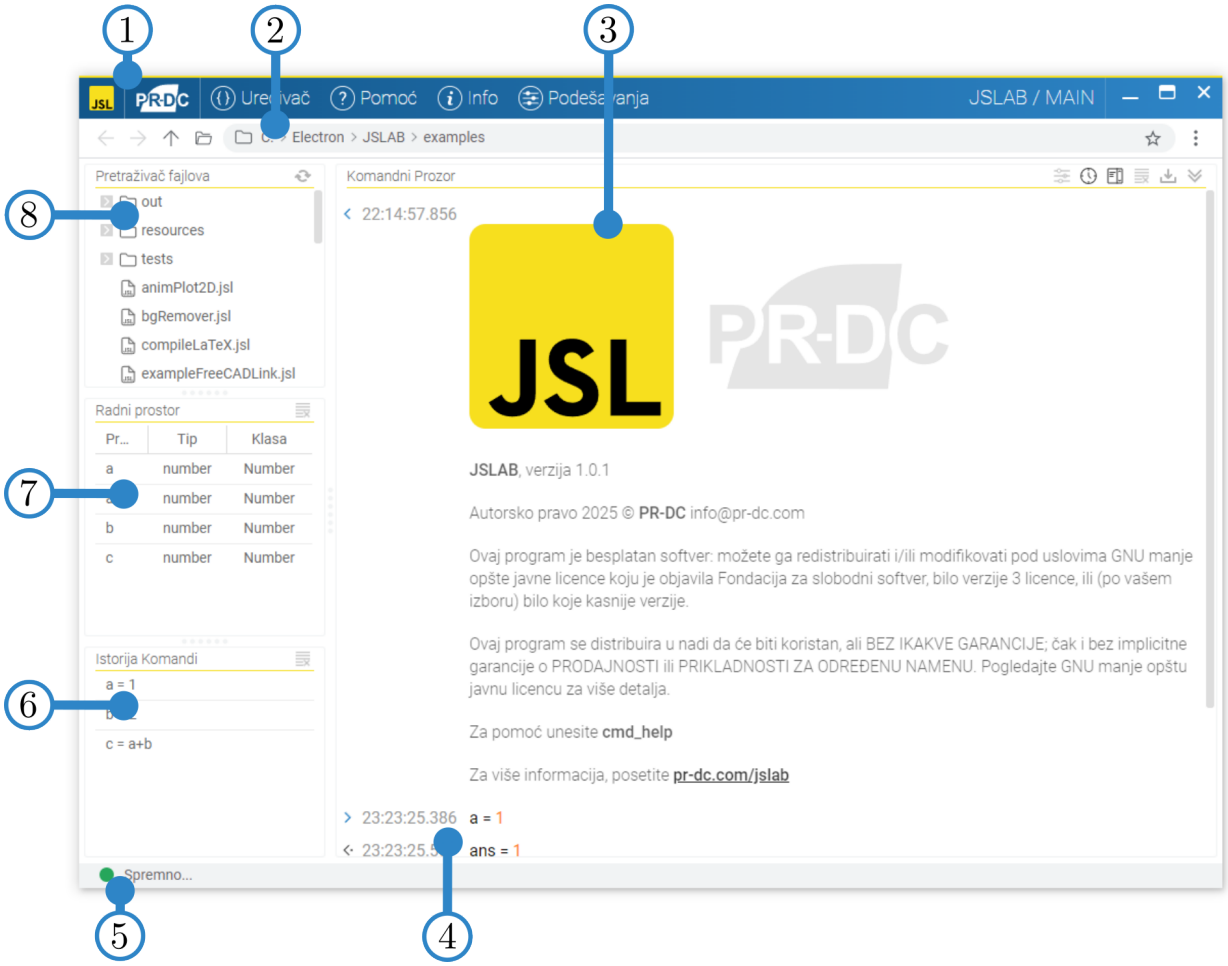
JSLAB je JavaScript laboratorijsko okruženje otvorenog koda za naučna i inženjerska izračunavanja, vizuelizaciju podataka ili bilo koju drugu operaciju na računaru[121]. Izvorni kod kao i prethodno kompajlirane verzije programa su dostupne na GitHub-u. Inspirisan GNU Octave-om i Matlab-om, ali sa svim prednostima JavaScript-a (neverovatno brz, sa impresivnom količinom dostupnih primera, koji su razvile neke od najvećih softverskih kompanija na svetu, i sa najvećim brojem aktivnih programera i softverskih inženjera). Logo projekta je prikazan na slici 3.48.



Slika 3.48 - Logo projekta JSLAB

Program je nastao zbog potrebe da se proračuni u ovoj disertaciji izvrše u programskom jeziku koji će omogućiti da se taj kod ponovo može koristiti i u kasnijim fazama projekta. Programski jezik JavaScript je izabran i zbog svih stvari prethodno navedenih za ovaj jezik. Radi se o veoma brzom, dinamičkom i interpretiranom jeziku (što omogućava brze promene koda i njegovo izvršavanje). Takođe, radi se o aktuelnom jeziku, sa mnogo dostupnih biblioteka, velikom bazom postojećeg koda, podrškom od najvećih svetskih softverskih kompanija i sa

mogućnošću kreiranja desktop i mobilnih aplikacija. Na slici 3.49 je prikazan glavni prozor JSLAB programa.



Slika 3.49 - Glavni prozor programa JSLAB

U okviru glavnog prozora postoje razne prečice koje su definisane u prozoru za pomoć, kao što je npr. prečica `[CTRL]+[H]` kojom se i otvara prozor za pomoć.

Na prethodnoj slici su brojevima označeni sledeći elementi glavnog prozora:

1. *Zaglavlje glavnog prozora* - meni sa opcijama za otvaranje prozora uređivača koda, prikaz pomoći, prikaz informacija o programu i za otvaranje podešavanja programa (kao što je podešavanje jezika, do sada su implementirani srpski jezik na latinici i ćirilici i engleski jezik).

2. *Navigacija za radni direktorijum* - implementirane su ikonice za kretanje kroz foldere i za otvaranje prozora za izbor određenog foldera. Osim trenutnog radnog foldera moguće je sačuvati druge putanje koje će se koristiti prilikom pokretanja skripti.

3. *Komandni prozor* - u centralnom desnom panelu, prozor u okviru koga se prikazuju poruke od radnog prostora nakon izvršenja koda.

4. *Polje za unos komandi* - u donjem delu desnog centralnog panela, koristi se za slanje komandi radnom prostoru. U polju za unos komandi se automatski dobijaju predlozi za završavanje započetog unosa komande i to na osnovu trenutno aktivnog radnog prostora. U polju za unos komande je moguće i kretanje kroz istoriju korišćenih komandi. Nakon unosa komande i pritiska tastera `[ENTER]` se komanda izvršava u okviru radnog prostora.

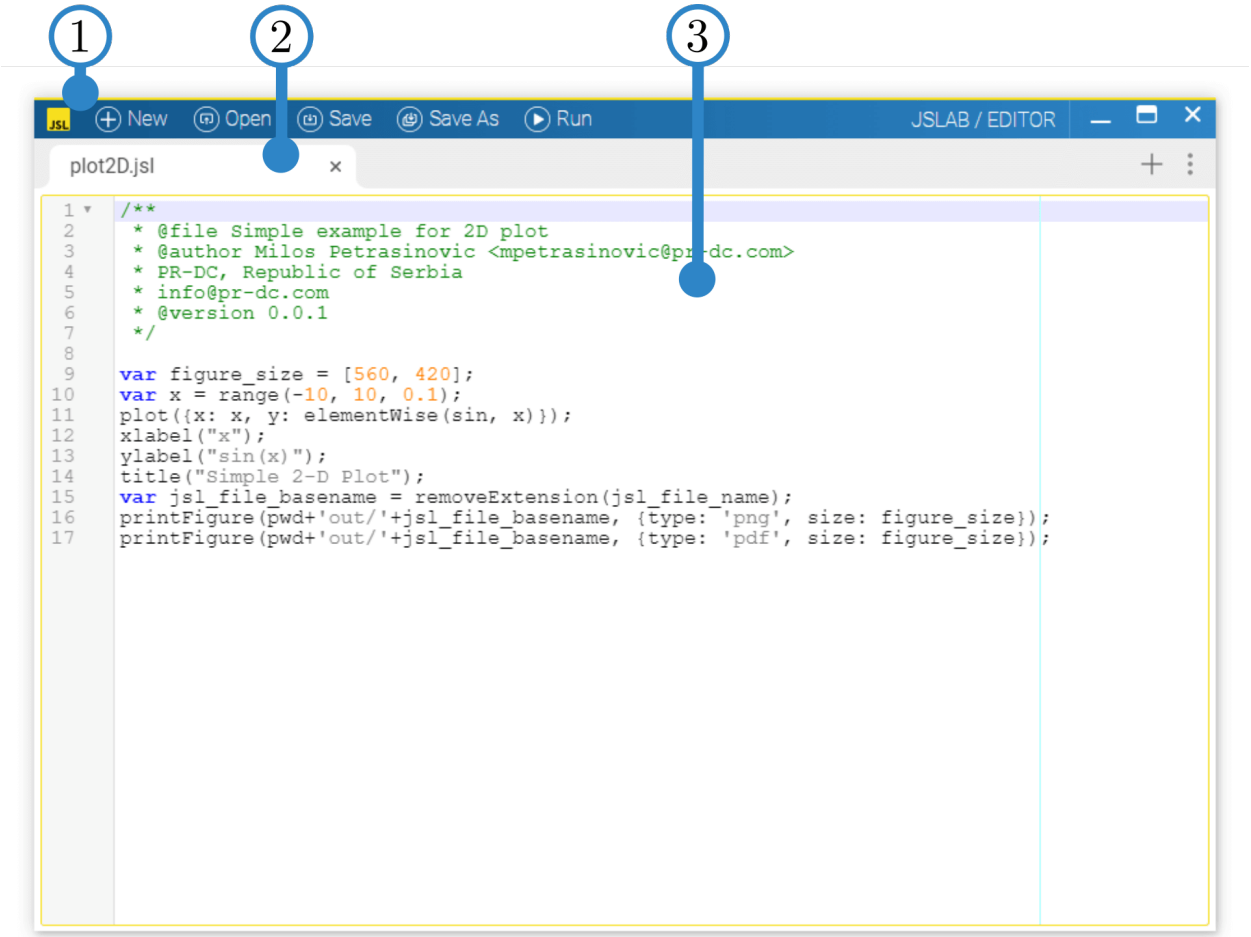
5. *Statusna traka* - traka u donjem delu prozora na kojoj se ispisuje trenutno stanje radnog prostora, a u donjem levom ćošku se nalazi ikonica koja menja boju u zavisnosti od tog stanja, pritiskom na ovu ikonicu se prikazuje oblačić u kome se ispisuje šta je trenutno u upotrebi u okviru radnog prostora.

6. *Istorija komandi* - donji deo levog centralnog panela, omogućava praćenje redosleda izvršavanja komandi i jednostavno ponovno izvršavanje neke komande (duplim pritiskom na komandu).

7. *Promenljive radnog prostora* - centralni deo levog centralnog panela, prikazuju se ime promenljive, tip i klasa za svaku aktivnu globalnu promenljivu u radnom prostoru koja je kreirana od strane korisnika.

8. *Pretraživač fajlova* - gornji deo levog centralnog panela, prikazuju se folderi i fajlovi i moguće je njihovo direktno otvaranje u okviru uređivača.

Komandi prozor je koristan kada postoji manji broj komandi koje se unose, ali kada je potrebno izvršiti nešto složenije tada se koriste skripte.



Slika 3.50 - Prozor uređivača koda programa JSLAB

Za skripte programa `JSLAB` se koristi ekstenzija `.jsl` kako bi se razlikovala od standardne ekstenzije `.js` za `JavaScript`. Skripte se pokreću unosom putanje do skripte kao jedini argument funkcije `run`. Za pisanje ali i izmenu koda u skripti je najbolje koristiti ugrađeni uređivač koda (može se otvoriti komandom `editor` ili funkcijom `edit()`). Prozor uređivača

koda je prikazan u okviru slike 3.50. Omogućeno je otvaranje više skripti u isto vreme, čuvanje skripti kao i njihovo direktno pokretanje u radnom prostoru.

Na prethodnoj slici su brojevima označeni sledeći elementi prozora uređivača:

1. *Zaglavlje prozora uređivača* - na samom vrhu prozora, meni sa opcijama za kreiranje novih skripti, otvaranje postojećih, čuvanje otvorenih skripti i direktno pokretanje skripti u radnom prostoru.

2. *Kartice skripti* - ispod zaglavlja prozora, služi za promenu aktivne skripte koja se trenutno uređuje kao i za otvaranje i zatvaranje skripti.

3. *Polje uređivača teksta skripte* - polje u okviru koga se nalazi kod skripte, tokom unosa se automatski dobijaju predlozi za završavanje započetog unosa komande i to na osnovu trenutno aktivnog radnog prostora, isto kao i prilikom unosa komandi u komandnom prozoru. Polje za unos teksta ima većinu naprednih opcija za rad sa kodom, vrši se provera koda i korisnik dobija sugestije, moguće je određene blokove koda kondenzovati, trenutna aktivna linija koda je naglašena itd.

Uređivač ima i druge napredne opcije za unos koda skripte, moguće je pretraživati kod kao i vršiti izmenu teksta sa drugim tekstom (oblačić se otvara pritiskom na `[CTRL]+[F]`).

Najznačajnije je da se kod izvršava na dva načina ili unosom u polje za unos komandi u okviru komandnog prozora ili putem pokretanja skripte sa funkcijom `run()` ili iz prozora uređivača.

U okviru listinga 3.1 je dat kod kojim se crta grafik funkcije, kao primer formiranja grafika u odvojenom prozoru. Korišćene funkcije `range`, `plot`, `elementWise`, `xlabel`, `ylabel` i `title` nisu standardne funkcije programskog jezika `JavaScript` već su kao i još mnoge druge funkcije definisane u okviru `JSLAB`-a.

```
1 var x = range(-10, 10, 0.1);
2 plot({x: x, y: elementWise(sin, x)});
3 xlabel("x");
4 ylabel("sin(x)");
5 title("Simple 2-D Plot");
```

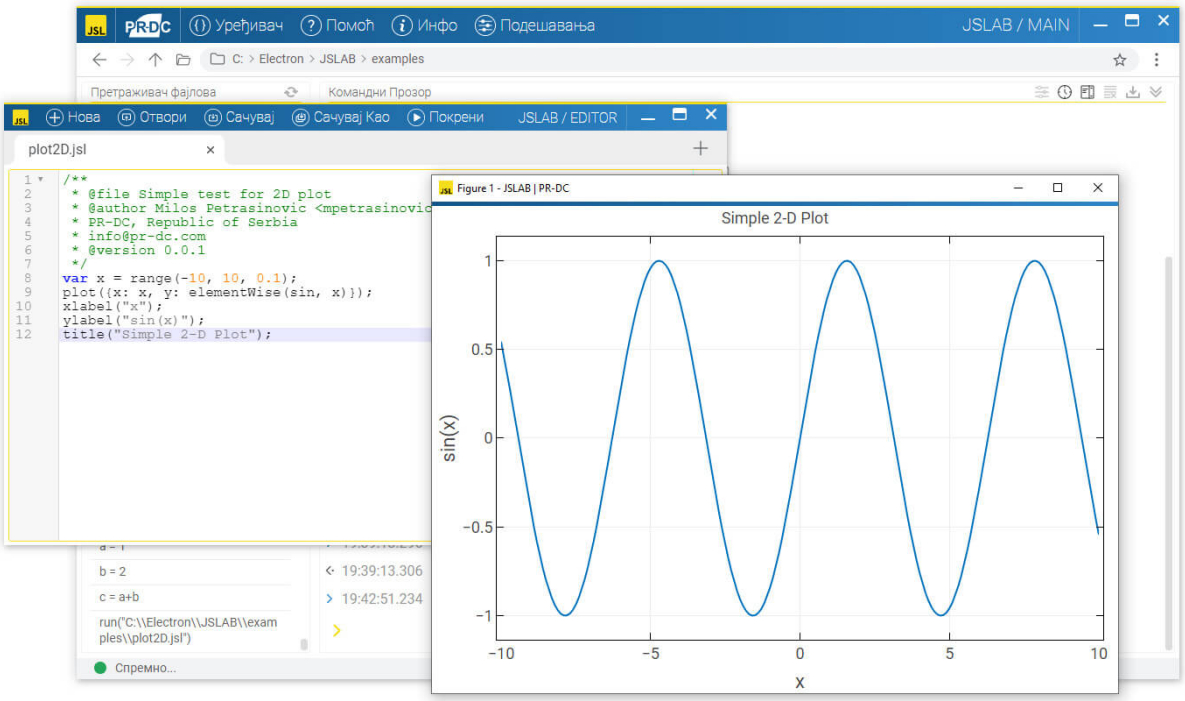
Listing 3.1 - `plot2D.jsl`

Na slici 3.51 je prikazan prozor sa grafikom, odnosno ono što se dobija nakon izvršavanja prethodno prikazane skripte. Na grafiku su označene ose, definisan je naslov i prikazana je kriva koja odgovara zatraženoj. Ovaj prozor grafika omogućava i druge opcije kao što je čuvanje samog grafika i to u vektorskom ili rasterskom obliku i u različitim formatima. Moguće je sačuvati grafik u `svg`, `pdf`, `png`, `jpg`, `webp` ili `html` formatu. Pored čuvanja grafika moguće je vršiti i zumiranje, transliranje ili fitovanje podataka u dostupan prostor prozora. Prilikom promene veličine prozora se automatski menja veličina i oblik grafika.

Prozori grafika su samo specijalne varijante dodatnih prozora koji se mogu otvarati iz radnog prostora. Funkcijom `openWindowBlank()` se otvara novi dodatni prazan prozor u kojem je bez ograničenja moguće definisati grafički korisnički interfejs pomoću `HTML`-a i `CSS`-a. Upravo ovde dolazi do izražaja i programski jezik *JavaScript* koji je i nastao iz potrebe za upravljanjem ovakvim prozorima u okviru internet pregledača.

Kao što je već spomenuto, `JavaScript` ima sintaksu koja je slična jezicima `C/C++` i pored toga zaista može da poziva kod napisan u programskom jeziku `C++` kada postoji potreba za još bržim izvršavanjem koda. `JSLAB` je implementiran korišćenjem projekta `Electron` što znači da sve što je implementirano u ovom programu tokom razvoja kasnije može da se potpuno isto primeni kada je potrebno izdvojiti kod u zasebnu desktop aplikaciju koja opet može biti

bazirana na projektu `Electron`. Kada već postoji kod u programskom jeziku `C++` moguće ga je odmah koristiti ali i kada je potrebno prevođenje iz `JavaScript`-a to je relativno jednostavno.



Slika 3.51 - Kompletan grafički korisnički interfejs programa JSLAB sa nacrtanim grafikom u zasebnom prozoru

Zajedno sa programom `JSLAB` se isporučuje i niz primera za različite mogućnosti programa, od već prikazanog formiranja grafika, preko komunikacije sa `USB` uređajima, do toga kako mogu da se kontrolišu drugi programi kao što su `FreeCAD` i `OpenModelica`. Baš ova mogućnost za kontrolisanje ova dva programa otvara neverovatne mogućnosti za inženjere koji se bave optimizacijom mehaničkih mehanizama i robotikom. Skripta može da pokrene `FreeCAD` program i zatim dinamički crta 3D model na osnovu rezultata nekog proračuna ili na osnovu unetih parametara, zatim da takav 3D model sačuva na disk. Ovo omogućava izradu celog kataloga modela na osnovu određenih parametara i to izuzetno brzo i bez potrebe za dodatnim mešanjem korisnika. Ovakvi 3D modeli se mogu zatim direktno koristiti zajedno sa `OpenModelica` programom. U okviru skripte se mogu menjati parametri za simulaciju, pokrenuti proračun i zatim dobiti rezultate proračuna, odnosno simulacije. Ukoliko se u skripti definiše algoritam za optimizaciju moguće je odrediti optimalne parametre na osnovu uzastopnog pokretanja simulacija i to sve automatski.

Funkcije `checkArduino`, `compileArduino` i `uploadArduino` se zajedno sa programom `arduino-cli` mogu koristiti za kompajliranje *Arduino* programa za mikrokontrolere. Ove funkcije se oslanjaju na konfiguracioni fajl kojim se podešava proces kompajliranja i upisivanja programa u memoriju mikrokontrolera (unosi se tip mikrokontrolera i podešavanja kao što su nivo optimizacije).

Osim desktop verzije programa planira se i web verzija koja će se izvršavati direktno u internet pregledaču. Što će pored postojećih grafika, koji se mogu već otvoriti na webu, omogućiti mnogo jednostavnije prezentovanje rezultata proračuna.

4

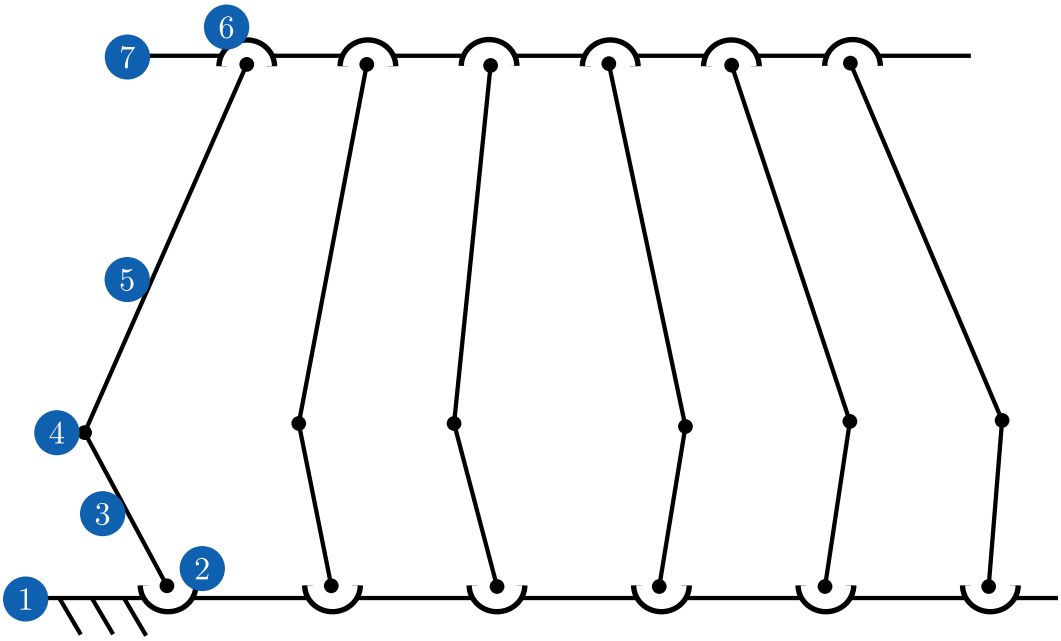
Određivanje geometrijskih parametara i performansi mehanizma simulatora leta za sistem

Simulator leta je drugi osnovni element sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi, kojim se ostvaruje povratna sprega kretanja. Da bi se razvio pristupačan moderni simulator leta (prikazan na slici 4.1) za ovaj sistem ključno je dobro proračunati optimalne parametre njegove geometrije za simuliranje leta. Cilj je da se nakon proračuna optimalnih vrednosti parametara mehanizma dobije mašina koja ima izuzetne mogućnosti u pogledu simuliranja leta, da bude konstruktivno jednostavna, a energetski efikasna. Pre procesa proračuna optimalnih vrednosti geometrijskih parametara je potrebno odrediti vrednosti geometrijskih parametara kojim se jednoznačno definiše geometrija mehanizma.



Slika 4.1 - 3D model modernog simulatora leta

Za mehanizam simulatora leta je izabran paralelni mehanizam koji se naziva *Stjuartova platforma* [7]. Ovo je mehanizam koji se trenutno u svetu koristi za praktično svaki simulator leta sa šest stepeni slobode kretanja. Na slici 4.2 (kao i na slici 3.1) je prikazan jedan od tipova i to Stjuartova platforma sa obrtnim aktuatorima, sa označenim elementima mehanizma.

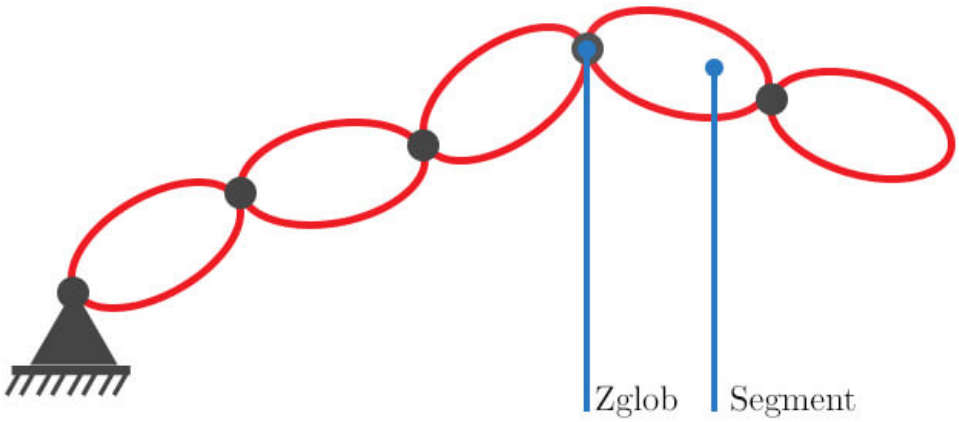


Slika 4.2 - Kinematički model Stjuartove platforma sa obrtnim aktuatorima

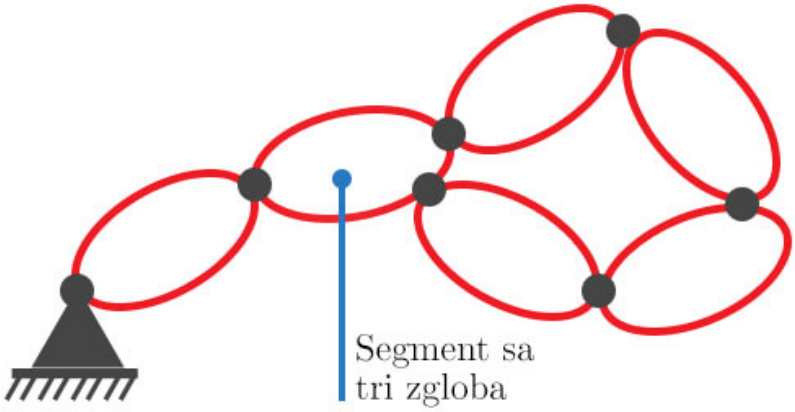
Ovakvi mehanizmi se kategorišu na osnovu prostorne konfiguracije i korišćenih zglobova. Platforma (7) prikazanog mehanizma je povezana sa šest gornjih poluga (5) preko gornjih zglobova (6) koji su po tipu sferni zglobovi. Sve gornje poluge su povezane sa donjim polugama (3) preko donjih zglobova (4) koji su takođe po tipu sferni zglobovi. Sve donje poluge su vezane za obrtne aktuatore (2) koji su na bazi mehanizma (1). Na osnovu tipa zglobova koji se koriste ovaj mehanizam se naziva još i `6-RUS` (na eng. Revolute, Universal i Spherical) ili `6-RSS` (na eng. Revolute, Spherical i Spherical). Prilikom rotacije donjih poluga dolazi do pomeranja gornjih poluga i istovremeno do promene pozicije i orijentacije platforme, što željena primena zahteva. Ovo je paralelni mehanizam jer poluge i platforma obrazuju *zatvoreni kinematički lanac*.

Pod mehanizmom se podrazumeva sistem krutih tela čija su kretanja u uzajamnoj vezi. Mehanizmi čine osnovu svih mašina. Kod mehanizama se jedna vrsta kretanja pretvara u drugu dok se kod mašina jedna vrsta energije pretvara u drugu. Svaki mehanizam se sastoji iz pokretnih članova (segmenata) i nepokretnih članova. Kinematički par obrazuju dva člana čija su kretanja u uzajamnoj vezi. Iz kinematike krutog tela je poznato da slobodno apsolutno kruto telo ima šest stepeni slobode kretanja, odnosno da može da vrši 6 nezavisnih kretanja, 3 translacije i 3 rotacije.

Niz tela, odnosno članova ili segmenata, povezani kinematičkim parovima obrazuju kinematički lanac. Kinematički lanci se dele na proste i složene, dok oba mogu biti otvoreni ili zatvoreni. Jedan segment prostog kinematičkog lanca ne može biti povezan sa više od dva segmenta, dok kod složenog kinematičkog lanca jedan segment može biti povezan sa više od dva segmenta. Otvoreni kinematički lanci imaju segmente koji su povezani samo sa još jednim segmentom, dok su kod zatvorenih kinematičkih lanaca svi segmenti povezani sa bar još dva segmenta [122], kao što je prikazano na slikama 4.3 i 4.4.



Slika 4.3 - Otvoren kinematički lanac



Slika 4.4 - Zatvoreni kinematički lanac

Pomeranja u zglobovima se u robotici najčešće nazivaju generalisanim ili unutrašnjim koordinatama, a spoljašnje koordinate su one koje opisuju poziciju i orijentaciju članova (segmenata) u odnosu na nepokretni koordinatni sistem. Promenom unutrašnjih koordinata menjaju se spoljašnje koordinate, odnosno menja se i pozicija i orijentacija. Određivanje spoljašnjih koordinata, odnosno pozicije i orijentacije članova za zadate unutrašnje koordinate, odnosno pomeranja u zglobovima se naziva *direktni problem kinematike*. Određivanje unutrašnjih koordinata, odnosno pomeranja u zglobovima, za zadate spoljašnje koordinate, odnosno zadatu poziciju i orijentaciju članova se naziva *inverzni problem kinematike*.

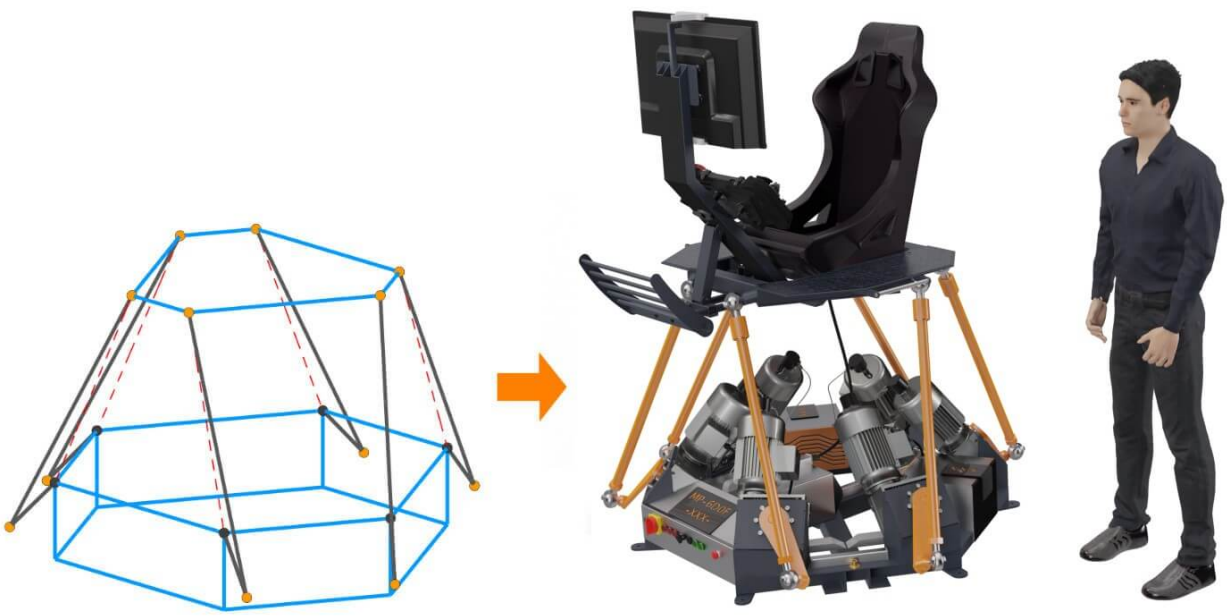
S obzirom na značaj i kompleksnost, rešavanje inverzne kinematike se mora posmatrati kao matematički, ali i kao inženjerski problem. Sistem je rešiv ako je za zadatu poziciju i orijentaciju moguće odrediti promenljive u zglobovima algoritmom koji daje sva rešenja. Izborom tipa kinematičke konfiguracije, broja stepeni slobode, parametara segmenata, ali i konstruktivnim rešenjima se definišu oblik i dimenzije radnog prostora i utiče na rešivost inverzne kinematike [122].

Potrebno je proveriti da li su sva rešenja inverzne kinematike ostvariva, i ako matematički postoje rešenja pitanje je da li su sva fizički upotrebljiva s obzirom da su rotacije u zglobovima često ograničene i da je moguća interferencija između segmenata.

U ovoj disertaciji je prikazana postavka *direktnog kinematičkog problema* robotskog mehanizma sa šest stepeni slobode kretanja za koji nije izveden izraz koji bi predstavljao

rešenje u zatvorenoj formi. Rešavanjem ovog problema se dobija mnogo u pogledu mogućnosti samog robota, ali od rešenja se zahteva kako preciznost tako i *rad u realnom vremenu u okviru sistema upravljanja.* Direktni kinematički problem se rešava korišćenjem *iterativne Njutnove metode za rešavanje sistema nelinearnih jednačina.*

Kada su određeni svi geometrijski parametri kojima se jednoznačno definiše mehanizam i kada su proverene sve performanse samog mehanizma, što se bazira na rešavanju inverznog i direktnog kinematičkog problema, potrebno je, kao što je prikazano na slici 4.5, materijalizovati mehanizam.



Slika 4.5 - Modeliranje platforme simulatora

Svakom segmentu mehanizma je potrebno dodeliti odgovarajući materijalizovani oblik. Na primer, ukoliko se izabere da se gornja poluga izrađuje od čelične cevi kružnog poprečnog preseka tada je još potrebno i odrediti spoljašnji prečnik i debljinu zida te cevi. Ovaj izbor je moguće izvršiti dimenzionisanjem na osnovu maksimalne dozvoljene elastične deformacije i odgovarajućeg pomeranja.

Na osnovu proračuna elastičnih deformacija konstrukcije mehanizma simulatora leta u okviru procesa njegovog projektovanja moguće je pravilno dimenzionisanje same konstrukcije, ali i aktuatora mehanizma. Pod pravilnim dimenzionisanjem se podrazumeva da su sve nužne deformacije elastične, odnosno da će se nakon prestanka dejstva spoljašnjeg opterećenja konstrukcija vratiti u prvobitno stanje. Za proračun neželjenih pomeranja konstrukcije je izabran *metod konačnih elemenata* kao opšte prihvaćen industrijski standard za ovakvu vrstu proračuna.

U slučaju robotskih sistema za pozicioniranje je neophodno smanjiti elastične deformacije jer svakoj deformaciji konstrukcije odgovara određeno pomeranje koje direktno predstavlja grešku pozicioniranja, ukoliko se u okviru sistema upravljanja ona na odgovarajući način ne kompenzuje.

Zbog zavisnosti opterećenja svakog elementa mehanizma od pozicije i orijentacije platforme, neophodno je, pre proračuna metodom konačnih elemenata, rešiti inverzni kinematički problem za taj konkretni slučaj i izabrati tip elementa kojim se modelira konstrukcija.

4.1 Tipovi aktuatora

Razmatrana su tri moguća tipa aktuatora za ovaj mehanizam i to: hidraulični cilindri, električni linearni aktuatori i električni obrtni aktuatori.

Prva dva moguća tipa aktuatora omogućavaju vezivanje platforme i baze pomoću 6 poluga promenljivih dužina, dok se kod trećeg tipa promena te dužine ostvaruje rotacijom dve poluge (gornje i donje) odnosno sa ukupno 12 pokretnih poluga konstantnih dužina.

4.1.1 Hidraulični cilindri - linearni aktuatori

Promenu dužine poluge moguće je ostvariti hidrauličkim cilindrom, na čijim krajevima se nalaze odgovarajući zglobovi.



Slika 4.6 - Hidraulična platforma (preuzeto iz [123])

Prednost ovih aktuatora (prikazanih na slici 4.6) je velika nosivost i mogućnost servo upravljanja, zbog čega su i danas nezamenljivi za simulatore velikih aviona koji zahtevaju mnogo različite opreme u kabini pa je potrebna nosivost od nekoliko tona. Nedostaci ovog rešenja su potreba za posebnom jedinicom za napajanje odnosno hidrauličkim agregatom, potreba za specijalnim održavanjem hidrauličkog sistema visokog pritiska i komplikovanija šema upravljanja.

4.1.2 Električni linearni aktuatori

Promenu dužine poluge je takođe moguće ostvariti električnim linearnim aktuatorom (prikazanim na slici 4.7), na čijim krajevima se nalaze odgovarajući zglobovi.



Slika 4.7 - Električni linearni aktuatori (preuzeto iz [124])

Na slici 4.8 je prikazana Stjuartova platforma sa električnim linearnim aktuatorima.

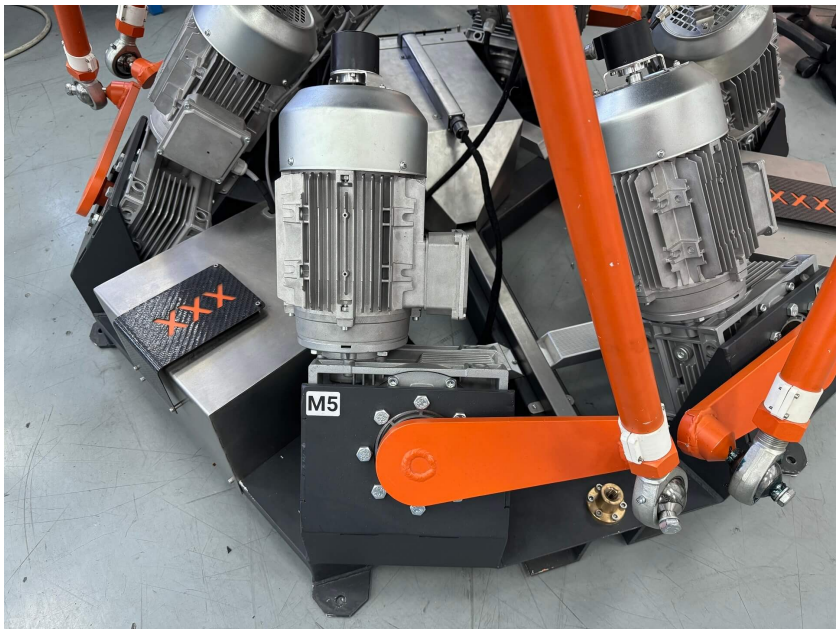


Slika 4.8 - eMotion-1500 Bosch Rexroth (preuzeto iz [125])

Ovo moguće rešenje koristi aktuatore koji se sastoje od elektromotora sa enkoderom, prenosnika i navojnog para koji pretvara rotaciono u linearno kretanje. Prednost ovih aktuatora je jednostavno upravljanje pozicijom i brzinom, i mogućnost kupovine kao gotovog sklopa sa potrebnim karakteristikama. Nedostaci ovog rešenja su manja nosivost, gubici i habanje u prenosnicima i složenija izrada, što povećava troškove.

4.1.3 Električni obrtni aktuatori

Kod ovog tipa aktuatora se promena dužine ostvaruje rotacijom dve poluge konstantnih dužina. Prva poluga je sa jedne strane direktno vezana na vratilo reduktora motora, koji se nalazi na bazi, dok je na drugom kraju vezana zglobno za drugu polugu koja je takođe zglobno vezana za pokretnu platformu.



Slika 4.9 - Električni obrtni aktuator Stjuartove platforme

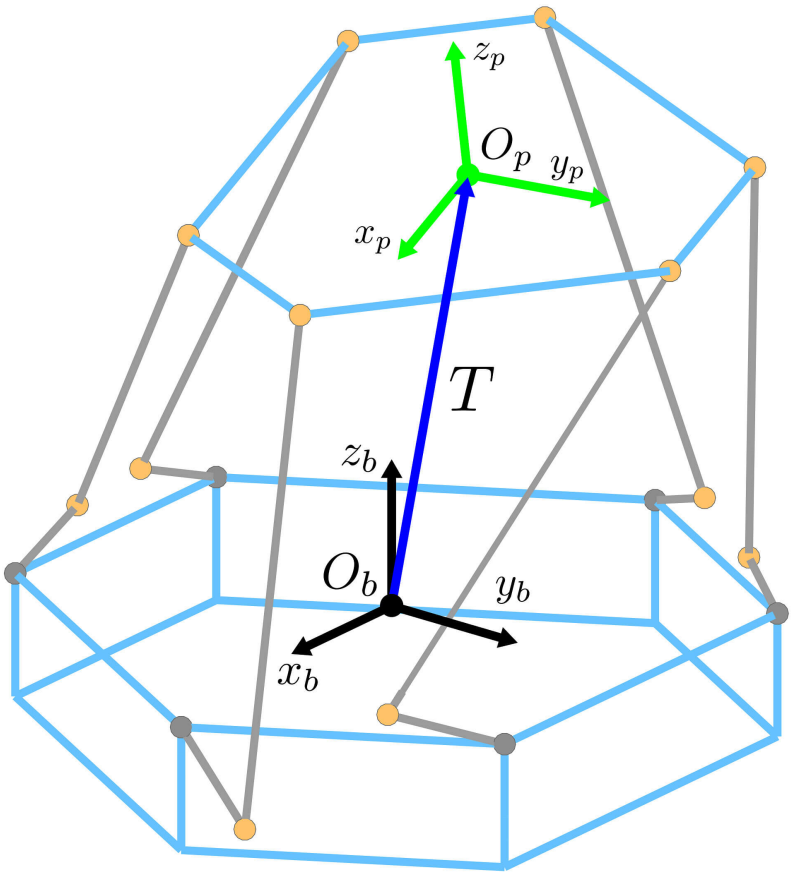
Ovo moguće rešenje aktuatora (prikazano na slici 4.9) se koristi u okviru ove disertacije pre svega zbog jednostavnosti sistema za izradu, održavanje i simuliranje. Ovakvi aktuatori su standardizovani i veoma dostupni, upravljaju se frekventnim regulatorima koji su takođe standardizovani i dostupni kao gotovi proizvodi. Ovaj tip obrtnih električnih aktuatora se mnogo više koristi u industriji od linearnih električnih aktuatora. Nedostatak ovog rešenja je potreba za reduktorom sa visokim prenosnim odnosom i manja nosivost.

4.2 Geometrijski parametri mehanizma

Kako bi se analiziralo da li je izabrani mehanizam moguće iskoristiti za simuliranje leta, prvo je potrebno definisati njegovu geometriju. Geometrija se definiše određivanjem svih karakterističnih vektora položaja tačaka mehanizma.

Potrebno je odrediti minimalni broj parametara koji intuitivno i jednoznačno određuju konfiguraciju mehanizma simulatora leta. Pod konfiguracijom mehanizma se misli na skup podataka koji su potrebni kako bi se on jednoznačno geometrijski opisao. U nastavku se usvaja ukupno *11 promenljivih parametara* koji mogu u potpunosti i jednoznačno da definišu geometriju mehanizma.

Na početku je potrebno definisati koordinatne sisteme koji se koriste prilikom analize mehanizma. Prvi koordinatni sistem je *nepokretni ortogonalni koordinatni sistem* $O_b\, x_b\, y_b\, z_b$ *vezan za bazu mehanizma*, dok je drugi *pokretni ortogonalni koordinatni sistem* $O_p\, x_p\, y_p\, z_p$ *vezan za platformu mehanizma.*

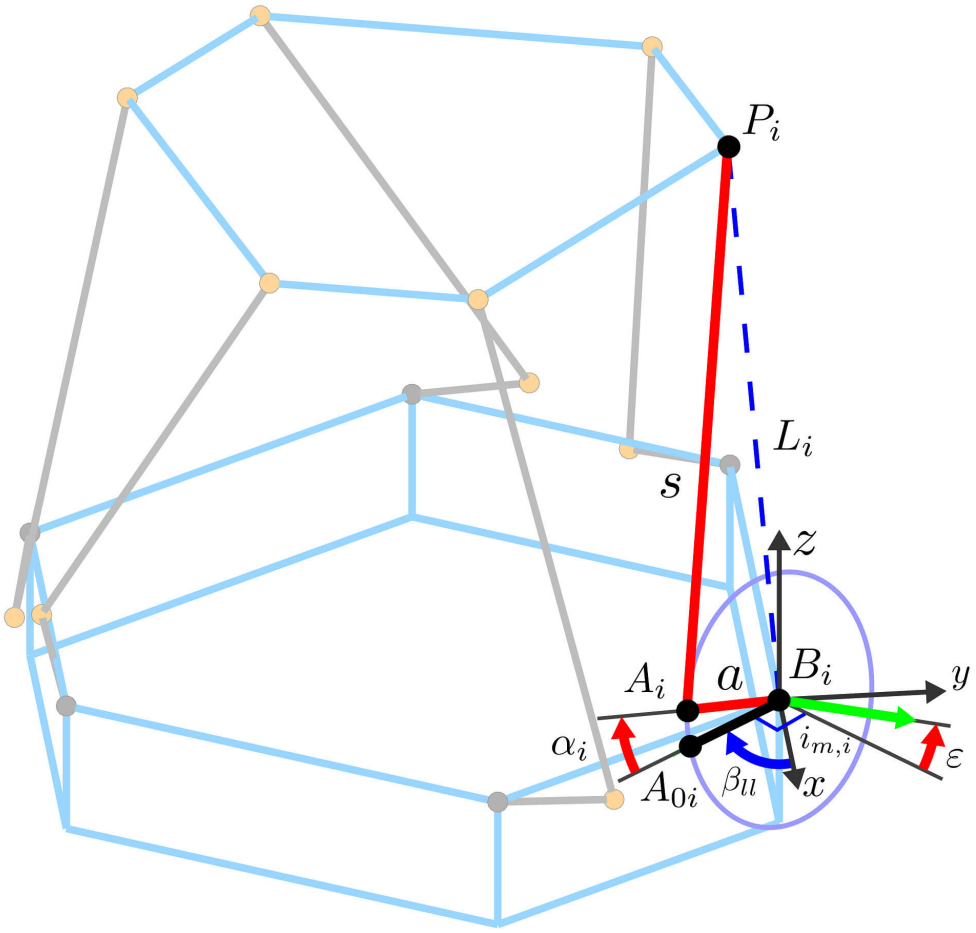


Slika 4.10 - Koordinatni sistemi baze i platforme mehanizma

Na slici 4.10 je još prikazan i vektor $\boldsymbol{T}$ (čiji elementi su dati u 4.1) koji predstavlja vektor translacije koordinatnog sistema platforme u odnosu na koordinatni sistem baze.

$$\boldsymbol{T} = \begin{bmatrix} x_{b,O_p} & y_{b,O_p} & z_{b,O_p} \end{bmatrix}^T \tag{4.1}$$

Na slici 4.11 je prikazana jedna *noga mehanizma* od ukupno šest, iz tog razloga je $i = 1, 2, \ldots, 6$. Pod nogom mehanizma se podrazumeva jedan aktuator, jedna donja i jedna gornja poluga koje su međusobno povezane donjim sfernim zglobom.



Slika 4.11 - Geometrijski parametri jedne noge mehanizma

Prva tri geometrijska parametra koja su prikazana na prethodnoj slici su:

- a - Dužina donje poluge od B_i do A_i,
- s - Dužina gornje poluge od A_i do P_i,
- ε - Ugao između obrtne ose aktuatora i horizontalne ravni.

Vektor $\boldsymbol{p_i}$ je vektor položaja tačke P_i u koordinatnom sistemu platforme (elementi vektora su prikazani u 4.2).

$$\boldsymbol{p_i} = \begin{bmatrix} x_{p,P_i} & y_{p,P_i} & z_{p,P_i} \end{bmatrix}^T = \begin{bmatrix} x_{P_i} & y_{P_i} & z_{P_i} \end{bmatrix}^T \tag{4.2}$$

Vektor $\boldsymbol{b_i}$ je vektor položaja tačke B_i u koordinatnom sistemu baze (elementi vektora su prikazani u 4.3).

$$\boldsymbol{b_i} = \begin{bmatrix} x_{b,B_i} & y_{b,B_i} & z_{b,B_i} \end{bmatrix}^T = \begin{bmatrix} x_{B_i} & y_{B_i} & z_{B_i} \end{bmatrix}^T \tag{4.3}$$

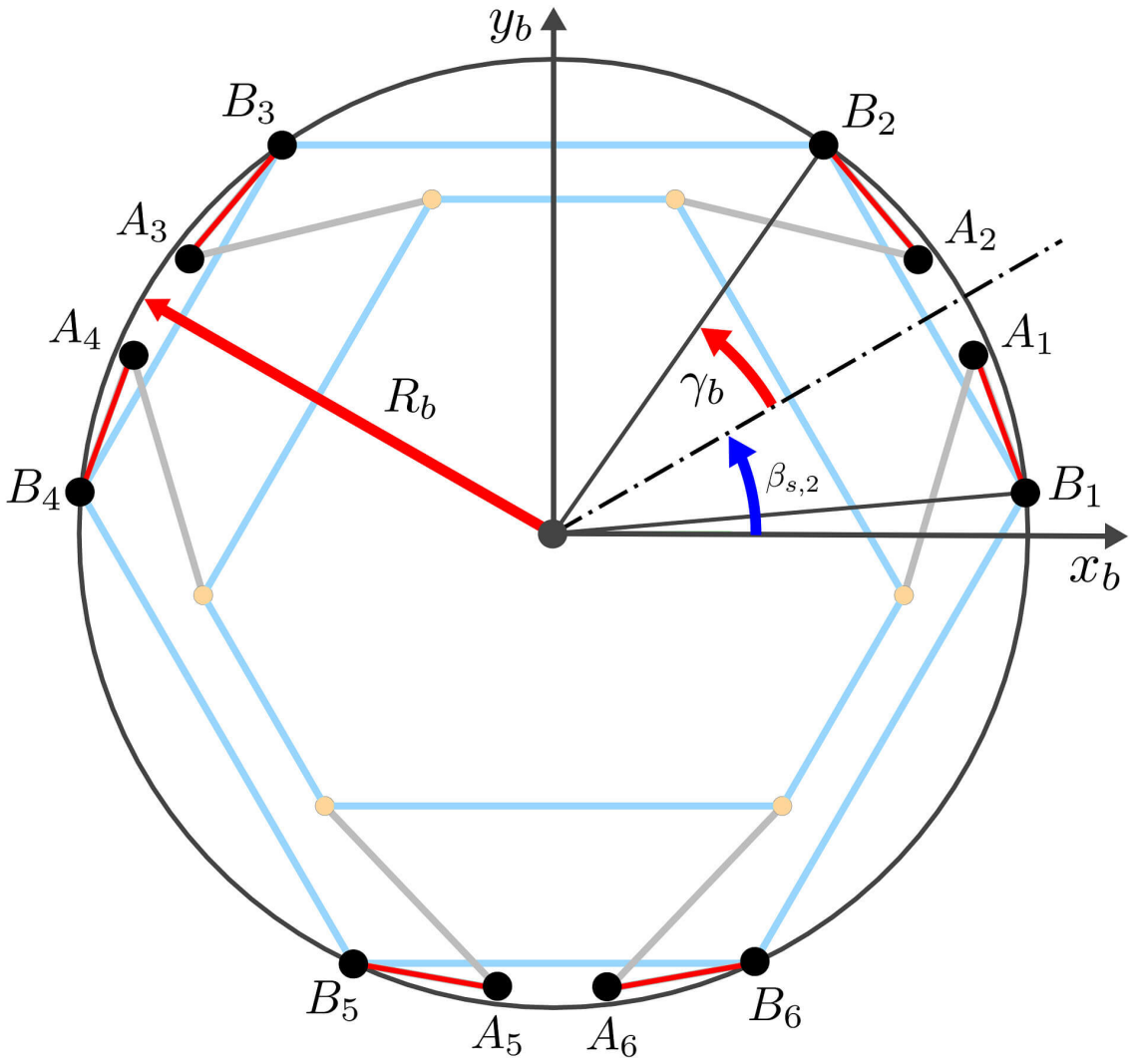
Vektor $\boldsymbol{a_i}$ je vektor položaja tačke A_i u koordinatnom sistemu baze (elementi vektora su prikazani u 4.4).

$$\boldsymbol{a_i} = \begin{bmatrix} x_{b,A_i} & y_{b,A_i} & z_{b,A_i} \end{bmatrix}^T = \begin{bmatrix} x_{A_i} & y_{A_i} & z_{A_i} \end{bmatrix}^T \tag{4.4}$$

Aktuatorom se vrši pomeranje donje poluge, čime se menja dužina noge. U ovom slučaju je dužina noge rastojanje između gornjeg sfernog zgloba, odnosno tačke P_i, i obrtnog aktuatora, odnosno tačke B_i. Ovu dužinu je moguće odrediti korišćenjem *Euklidske norme vektora* tako da je:

$$l_i = \|\boldsymbol{L_i}\|_2 \tag{4.5}$$

Pomeranje donje poluge predstavlja rotaciju aktuatora za ugao α_i oko ose aktuatora koja je definisana jediničnim vektorom ose $\boldsymbol{i_{m,i}}$. Ugao α_i se meri od početnog vektora $\boldsymbol{B_iA_i}$ koji leži u horizontalnoj ravni. Na slici 4.11 su kružnicom predstavljeni svi mogući položaji tačke A_i i vektor ose $\boldsymbol{i_{m,i}}$ (zeleni vektor).



Slika 4.12 - Geometrijski parametri baze

Kada su u potpunosti definisane gornje i donje poluge, potrebno je još definisati i platformu i bazu mehanizma. Koordinate tačaka B_i i P_i moraju biti određene u odgovarajućim referentnim sistemima kako bi se u potpunosti definisali baza i platforma mehanizma. Intuitivan način za

to je korišćenje polarnih koordinata, pri čemu se primećuje da tačke B_i i P_i leže na krugovima sa konstantnim prečnikom i da postoje tri ose simetrije između ovih tačaka (tačka B_1 može se preslikati na tačku B_2, dok je osa simetrije pod određenim uglom u odnosu na x osu globalnog referentnog sistema).

Kao što je prikazano na slici 4.12, osa simetrije između tačaka B_1 i B_2 je definisana uglom $\beta_{s,i}$. Ovaj ugao ima konstantnu vrednost za sve posmatrane geometrije, a njegova svrha je da definiše tri segmenta za tačke B_i. Osa simetrije tačaka B_1 i B_2 je rotirana za 30° u odnosu na x-osu, zatim je osa simetrije tačaka B_3 i B_4 rotirana za 150°, a osa simetrije tačaka B_5 i B_6 je rotirana za $-90°$. Ove tri ose simetrije čine raspodelu tačaka ravnomernom. Ugao $\beta_{s,i}$ nije geometrijski parametar za proračun, ali je neophodan za određivanje koordinata tačaka B_i sa ostalim parametrima.

$$\boldsymbol{\beta_s} = \begin{bmatrix} \frac{1}{6}\pi & \frac{1}{6}\pi & \frac{5}{6}\pi & \frac{5}{6}\pi & -\frac{1}{2}\pi & -\frac{1}{2}\pi \end{bmatrix} \tag{4.6}$$

Kako tačka B_i može da se nalazi levo ili desno u odnosu na osu simetrije dok apsolutna vrednost ugla γ_b ostaje ista, uvodi se dodatni vektor kojim se definiše predznak za ugao.

$$\boldsymbol{\sigma_l} = \begin{bmatrix} -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix} \tag{4.7}$$

Prilikom definisanja položaja tačaka B_i se koriste još dva geometrijska parametra koja su prikazana na prethodnoj slici (slika 4.12) i to su:

- R_b - Poluprečnik baze, rastojanje od O_b do B_i,
- γ_b - Uglovi između vektora $\boldsymbol{O_bB_i}$ i ose za preslikavanje tačaka B_i.

$$\boldsymbol{\delta_b} = \boldsymbol{\beta_s} + \gamma_b \boldsymbol{\sigma_l} \tag{4.8}$$

$$x_{B_i} = R_b \cos \delta_{b,i} \tag{4.9}$$
$$y_{B_i} = R_b \sin \delta_{b,i} \tag{4.10}$$
$$z_{B_i} = 0 \tag{4.11}$$

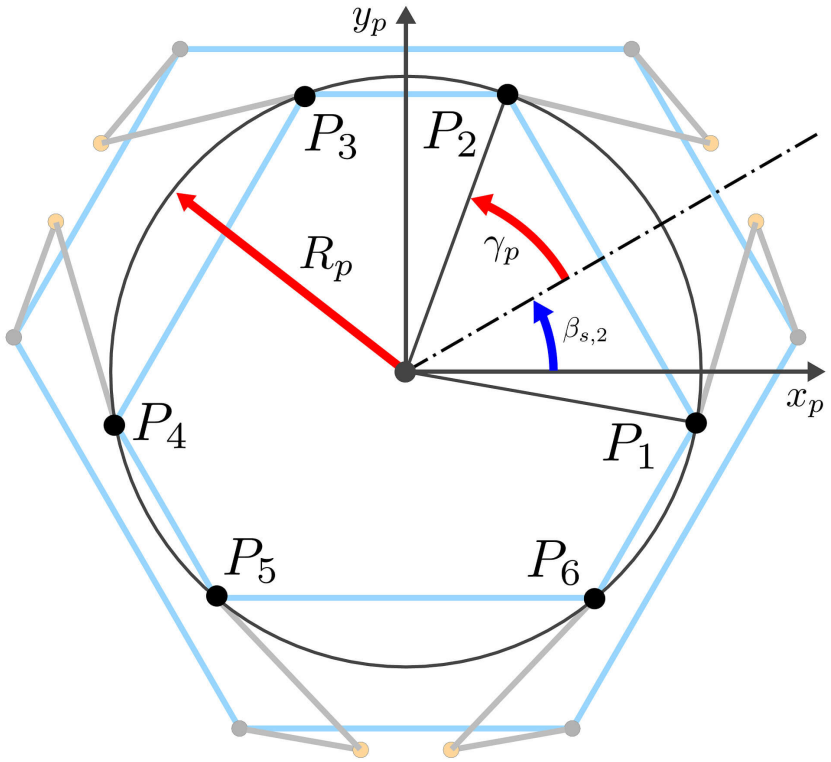
Slično kao kod tačaka B_i, položaj svih šest tačaka P_i u odnosu na referentni sistem platforme definisan je sa dva promenljiva geometrijska parametra. Takođe kao u slučaju baze, ponovo postoje tri ose simetrije između tačaka.

Prilikom definisanja položaja tačaka P_i se koriste još dva geometrijska parametra koja su prikazana na slici 4.13 i to su:

- R_p - Poluprečnik platforme, rastojanje od O_p do P_i,
- γ_p - Ugao između vektora $\boldsymbol{O_pP_i}$ i ose za preslikavanje tačaka P_i.

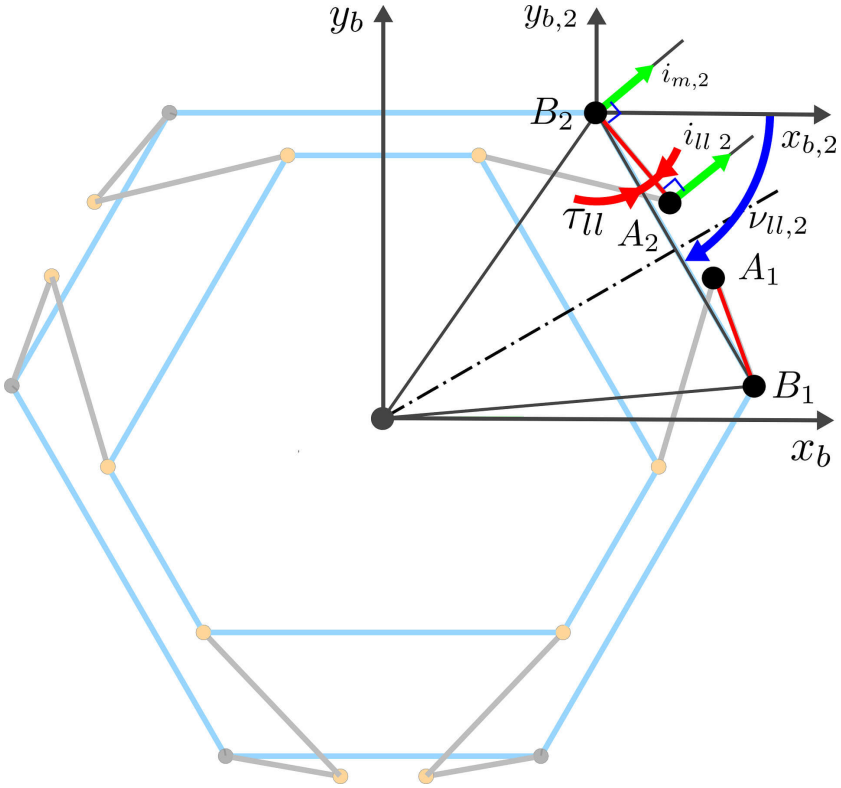
$$\boldsymbol{\delta_p} = \boldsymbol{\beta_s} + \gamma_p \boldsymbol{\sigma_l} \tag{4.12}$$

$$x_{P,i} = R_p \cos \delta_{p,i} \tag{4.13}$$
$$y_{P,i} = R_p \sin \delta_{p,i} \tag{4.14}$$
$$z_{P,i} = 0 \tag{4.15}$$



Slika 4.13 - Geometrijski parametri platforme

Sada je neophodno definisati položaj tačaka A_i, koji zavisi od prethodno pomenutog vektora, $\boldsymbol{i_{m,i}}$, koji predstavlja jedinični vektor ose rotacije aktuatora.



Slika 4.14 - Koordinatni sistem aktuatora i donje poluge

Kao što je prikazano na slici 4.14, osa rotacije aktuatora ne mora biti normalna u odnosu na liniju koja prolazi kroz tačke B_1 i B_2, ali je ova linija referentna za definisanje orijentacije

donjih poluga u horizontalnoj ravni (kada su donje poluge u početnom položaju). Referentna linija definisana je uglom $\boldsymbol{\nu_{ll,i}}$, koji se meri od x ose globalnog referentnog sistema. Vrednost ovog ugla se definiše vektorom $\boldsymbol{\nu_{ll}}$ za svaku donju polugu.

$$\boldsymbol{\nu_{ll}} = \boldsymbol{\beta_s} - \frac{\pi}{2}\boldsymbol{\sigma_l} \tag{4.16}$$

Za tačno definisanje orijentacije donjih poluga se koristi sledeći geometrijski parametar:

- τ_{ll} - Ugao odstupanja orijentacije donje poluge od referentne.

Zatim, ugao između donje poluge i x_b ose dat je sledećom jednačinom:

$$\boldsymbol{\beta_{ll}} = \boldsymbol{\nu_{ll}} + \tau_{ll}\,\boldsymbol{\sigma_l} \tag{4.17}$$

Projekcija jediničnog vektora ose rotacije aktuatora u horizontalnoj ravni je normalna na donju polugu. Na osnovu toga i već definisanog ugla nagiba ε, jedinični vektor ose rotacije aktuatora je:

$$\boldsymbol{i_{m,i}} = -\sigma_{l,i}\begin{bmatrix}\cos(\beta_{ll,i} + \frac{\pi}{2}\,\sigma_{l,i})\,\cos\varepsilon \\ \sin(\beta_{ll,i} + \frac{\pi}{2}\,\sigma_{l,i})\,\cos\varepsilon \\ \sin\varepsilon\end{bmatrix} \tag{4.18}$$

Kada su analizirane i geometrijske karakteristike baze, na osnovu onih već definisanih za donju polugu mogu se dobiti izrazi za određivanje koordinata tačke A_{0i}, odnosno tačaka A_i u početnom položaju u odnosu na koordinatni sistem baze.

Tačka A_{0i} (prikazana na slici 4.11) je referentna tačka koja predstavlja tačku A_i u početnom položaju. Vektor $\boldsymbol{a_{0i}}$ je vektor položaja tačke A_{0i} u koordinatnom sistemu baze (elementi vektora su prikazani u 4.19).

$$\boldsymbol{a_{0i}} = \begin{bmatrix}x_{b,A_{0i}} & y_{b,A_{0i}} & z_{b,A_{0i}}\end{bmatrix}^T = \begin{bmatrix}x_{A_{0i}} & y_{A_{0i}} & z_{A_{0i}}\end{bmatrix}^T \tag{4.19}$$

$$x_{A_{0i}} = a\,\cos\beta_{ll,i} + x_{B,i} \tag{4.20}$$

$$y_{A_{0i}} = a\,\sin\beta_{ll,i} + y_{B,i} \tag{4.21}$$

$$z_{A_{0i}} = z_{B,i} \tag{4.22}$$

Pored geometrijskih parametara, koji se usvajaju na početku proračuna, za određivanje položaja tačke A_i ostaje još promenljiva α_i kao nepoznata. Određivanje uglova $\boldsymbol{\alpha}$ za željenu poziciju i orijentaciju platforme predstavlja geometrijski metod za rešavanje inverznog kinematičkog problema. Kako je poznat jedinični vektor ose rotacije pomoću *Rodrigovog obrasca* je moguće napisati izraz za izračunavanje položaja tačake A_i.

$$\boldsymbol{a_i} = \boldsymbol{b_i} + \boldsymbol{B_iA_{0i}} + (1-\cos\alpha_i)\,\boldsymbol{Rm1_{B_iA_i}} + (\sin\alpha_i)\,\boldsymbol{Rm2_{B_iA_i}} \tag{4.23}$$

$$\boldsymbol{Rm2_{B_iA_i}} = \boldsymbol{i_{m,i}} \times \boldsymbol{B_iA_{0i}} \tag{4.24}$$

$$\boldsymbol{Rm1_{B_iA_i}} = \boldsymbol{i_{m,i}} \times \boldsymbol{Rm2_{BA}} \tag{4.25}$$

Sa svim prethodno definisanim tačkama, geometrija mehanizma je potpuno definisana. Ipak, neophodno je definisati još dva jedinična vektora za svaku nogu kako bi se mogla modelirati ograničenja sfernih zglobova. Ovi jedinični vektori sfernih zglobova su ključni za

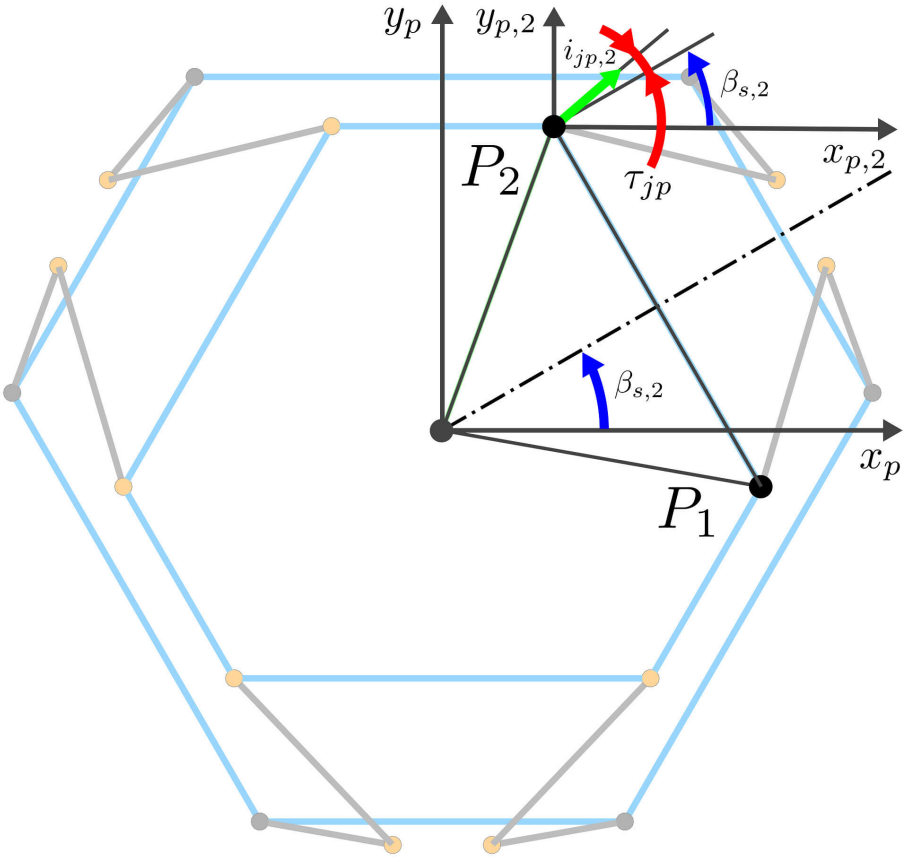
proračun, jer se njihova orijentacija koristi za izračunavanje fizičkih ograničenja ovog mehanizma. Ovo fizičko ograničenje se implementira kao nelinearno ograničenje jer pri određenom maksimalnom uglu zgloba dolazi do kontakta između različitih delova zgloba, nakon čega se dalji pokret onemogućava. Očekuje se da algoritam za proračun optimalnih parametara prilagodi orijentaciju ovih vektora kako bi se dobio najveći mogući radni prostor.

Na slici 4.15 prikazana je projekcija jediničnog vektora ose gornjeg zgloba $\boldsymbol{i_{jp,i}}$ u ravni definisanoj sa x_p i y_p. Za tačno definisanje orijentacije ovog vektora u ovoj ravni se koristi sledeći geometrijski parametar:

- τ_{jp} - Ugao odstupanja orijentacije gornjih zglobova od referentne orijentacije.

Orijentacija jediničnog vektora ose gornjeg zgloba u ovoj ravni (ugao između projekcije vektora i x_p ose) definiše se na sledeći način:

$$\boldsymbol{\beta_{jp}} = \boldsymbol{\beta_s} + \tau_{jp}\,\boldsymbol{\sigma_l} \tag{4.26}$$



Slika 4.15 - Koordinatni sistem gornjeg zgloba

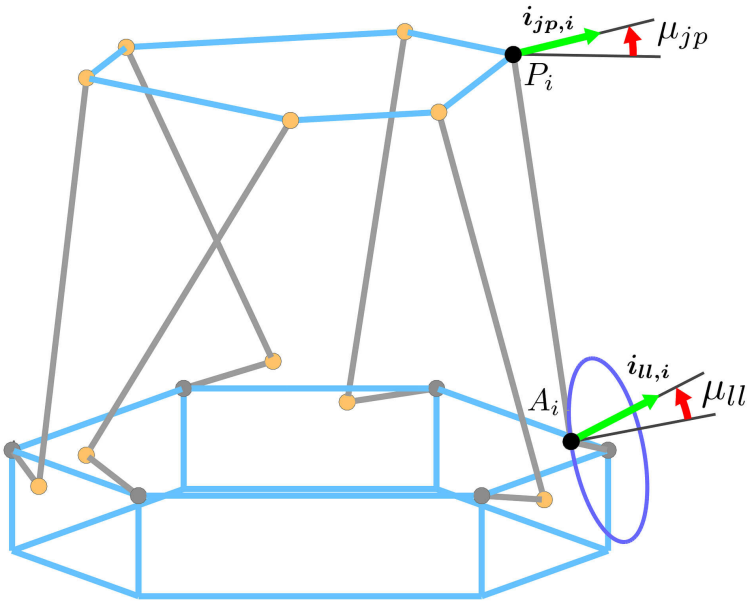
Prethodno analizirani jedinični vektor koristi se za izračunavanje fizičkog ograničenja sfernog zgloba koji povezuje platformu i gornju polugu. Poslednji jedinični vektor je $\boldsymbol{i_{ll}}$, koji definiše osu sfernog zgloba koji povezuje gornju i donju polugu. Ovaj jedinični vektor je već prikazan na slici 4.14. U početnom položaju, njegova projekcija na horizontalnu ravan ima isti pravac kao i projekcija ose rotacije aktuatora.

Jedinični vektori zglobova ponovo su prikazani na slici 4.16; njihova orijentacija u ravnima platforme i baze već je razmotrena, ali se uvode još dva, poslednja geometrijska parametra za potpunu orijentaciju.

- μ_{jp} - Ugao ugradnje ose za vezu sfernog zgloba platforme.
- μ_{ll} - Ugao ugradnje ose za vezu sfernog zgloba donje poluge.

Vektori su nagnuti u odnosu na odgovarajuće ravni pod uglovima μ_{jp} i μ_{ll}. Isti princip je već korišćen za aktuatore, te se dobija sledeća jednačina:

$$\boldsymbol{i_{jp,i}} = \begin{bmatrix} \cos\beta_{jp,i}\,\cos\mu_{jp} & \sin\beta_{jp,i}\,\cos\mu_{jp} & \sin\mu_{jp} \end{bmatrix}^T \tag{4.27}$$



Slika 4.16 - Ose za vezu sfernih zglobova

Pošto je jedan deo zgloba između donje i gornje poluge pričvršćen za donju polugu, on se rotira zajedno sa donjom polugom, te je neophodno definisati referentni vektor ose u početnom položaju $\boldsymbol{i_{ll0,i}}$ sa:

$$\boldsymbol{i_{ll0,i}} = \begin{bmatrix} \cos(\beta_{ll,i} + \frac{\pi}{2}\,\sigma_{l,i})\,\cos(\varepsilon + \mu_{ll}) \\ \sin(\beta_{ll,i} + \frac{\pi}{2}\,\sigma_{l,i})\,\cos(\varepsilon + \mu_{ll}) \\ \sin(\varepsilon + \mu_{ll}) \end{bmatrix} \tag{4.28}$$

Kako se ovaj vektor rotira zajedno sa donjom polugom, potrebno je ponovo iskoristiti *Rodrigov obrazac* na osnovu čega se dobija sledeći izraz za određivanje vektora.

$$\boldsymbol{i_{ll,i}} = \boldsymbol{i_{ll0,i}} + (1 - \cos\alpha_i)\,\boldsymbol{Rm1_{i_{ll}}} + (\sin\alpha_i)\,\boldsymbol{Rm2_{i_{ll}}} \tag{4.29}$$

$$\boldsymbol{Rm2_{i_{ll}}} = \boldsymbol{i_{m,i}} \times \boldsymbol{i_{ll0,i}},\ \boldsymbol{Rm1_{i_{ll}}} = \boldsymbol{i_{m,i}} \times \boldsymbol{Rm2_{i_{ll}}} \tag{4.30}$$

Za proračun optimalne geometrije, jedanaest geometrijskih parametara koji određuju geometriju mehanizma za simulatore letenja su precizno definisani. Oni nedvosmisleno opisuju geometriju platforme. Treba napomenuti da se geometrija može definisati i na druge načine. Ipak, pošto osim simetrije između zglobova (npr. preslikavanje tačke B_1 u tačku B_2) i rasporeda na kružnici, nisu nametnuta nikakva druga ograničenja, može se dobiti bilo koja upotrebljiva konfiguracija. Neophodno je samo jednom implementirati prethodno opisani algoritam za izračunavanje položaja i vektora osa na osnovu geometrijskih parametara. Zatim je jednostavno promeniti 11 brojeva kako bi se dobila drugačija geometrija mehanizma.

Primer vrednosti za sve geometrijske parametre dat je u tabeli 4.1.

Tabela 4.1 - Primer konfiguracije mehanizma

Parametar	Vrednost
a	230 mm
s	1110 mm
R_p	520 mm
R_b	840 mm
γ_p	7°
γ_b	35°
ε	-39°
τ_{ll}	-122°
τ_{jp}	-12°
μ_{ll}	-18°
μ_{jp}	-1°

Sve prethodno prikazane jednačine je neophodno iskoristiti kako bi se izračunali odgovarajući položaji tačaka. Ove jednačine su implementirane u `JSLAB`-u, ali i u programskim jezicima `C` i `C++`, određena implementacija se koristi u zavisnosti od zahteva po pitanju brzine izvršavanja programa i jednostavnosti implementacije.

Iz prethodno prikazanih jednačina se može zaključiti da postoji potreba za čestim izračunavanjem vektorskog proizvoda u trodimenzionalnom prostoru i to svaki put za šest pari vektora, za to je napisana funkcija `cross3D()`. Zbog toga što je ovu funkciju potrebno pozivati i nekoliko miliona puta u okviru proračuna, napisana je ista funkcija u programskom jeziku `C` kako bi se dobilo izvršavanje programa koje može da bude značajno brže.

Za definisanje geometrijskih parametara mehanizma na početku svakog programa za analiziranje se pokreće metoda `getGeometry()`. U okviru ove metode se koristi promenljiva `MP` koja je objekat klase `MP_6RSS_MECHANISM`. Sama metoda `getGeometry()` se izvršava nad objektom klase `MP_6RSS_SOLVER` koja ima metode za sva potrebna izračunavanja (kompletan kod dat je u listingu u prilogu A). U prilogu B je dat kompletan kod modula kojim se definiše `C++` klasa `SolverMP6RSS` koja takođe uključuje i funkciju `getGeometry()`.

Ovakvo, objektno orijentisano, programiranje veoma pojednostavljuje sam kod, posebno kada je potrebno definisati mnoštvo različitih promenljivih i sa njima pozivati različite funkcije. Pored toga, na ovaj način se jednostavno može formirati više objekata, za više mehanizama, sa različitim svojstvima i vršiti dalje analize.

4.3 Matrice transformacija koordinata

Osim vektora translacije koordinata $\boldsymbol{T}$ (čiji elementi su dati u 4.1), potrebno je odrediti i elemente matrice rotacije $\boldsymbol{R}$ koji su funkcije koje zavise od tri *Ojlerova ugla*, koji određuju orijentaciju platforme mehanizma u odnosu na koordinatni sistem baze. Kako bi se pojednostavio prikaz, kondenzovani oblici zapisa trigonometrijskih funkcija `sinus` i `kosinus`, *"s"* i *"c"* su korišćeni da se definiše matrica rotacije na sledeći način:

$$\boldsymbol{R} = \begin{bmatrix} c_\psi\, c_\theta & -s_\psi\, c_\varphi + c_\psi\, s_\theta\, s_\varphi & s_\psi\, s_\varphi + c_\psi\, s_\theta\, c_\varphi \\ s_\psi\, c_\theta & c_\psi\, c_\varphi + s_\psi\, s_\theta\, s_\varphi & -c_\psi\, s_\varphi + s_\psi\, s_\theta\, c_\varphi \\ -s_\theta & c_\theta\, s_\varphi & c_\theta\, c_\varphi \end{bmatrix} \tag{4.31}$$

Prethodna matrica se dobija množenjem matrica za rotaciju oko određenih osa i to na sledeći način:

$$\boldsymbol{R} = \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{R_x}(\varphi) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\varphi & -s_\varphi \\ 0 & s_\varphi & c_\varphi \end{bmatrix} \tag{4.32}$$

Elementi matrice rotacije određuju se metodom `rotMat()` klase `MP_6RSS_SOLVER`. Prilikom implementacije korišćen je nešto efikasniji metod za definisanje ove matrice. Pošto što je ovo zaista esencijalna funkcija za proračun, funkcija je napisana i u `C` programskom jeziku (u okviru `C++` klase `SolverMP6RSS`).

Prilikom definisanja geometrijskih parametara ranije je korišćen *Rodrigov obrazac* u sledećem obliku:

$$\boldsymbol{r_1} = \boldsymbol{r_0} + (1 - \cos\alpha_i)\,\boldsymbol{e} \times (\boldsymbol{e} \times \boldsymbol{r_0}) + (\sin\alpha_i)\,\boldsymbol{e} \times \boldsymbol{r_0} \tag{4.33}$$

Ovaj obrazac je moguće napisati i u matričnom obliku tako da se transformacija vektora $\boldsymbol{r_0}$ u $\boldsymbol{r_1}$ svodi na *Rodrigovu matricu* koja se označava sa $\boldsymbol{A_r}$ [126].

$$\boldsymbol{r_1} = \boldsymbol{A_r}\,\boldsymbol{r_0} \tag{4.34}$$

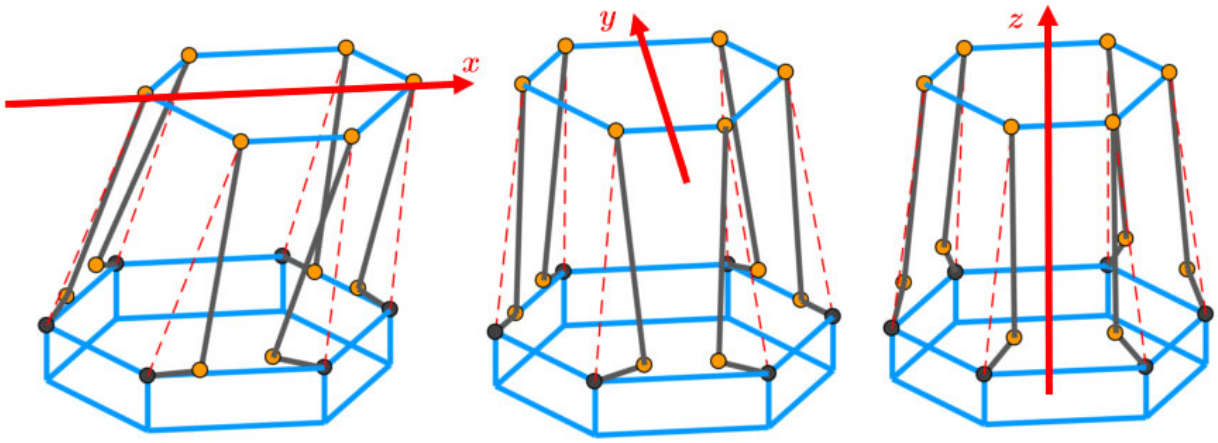
$$\boldsymbol{A_r} = \boldsymbol{I} + (1 - \cos\alpha_i)\,\boldsymbol{e}^{d^2} + (\sin\alpha_i)\,\boldsymbol{e}^{d} \tag{4.35}$$

4.4 Inverzni kinematički problem

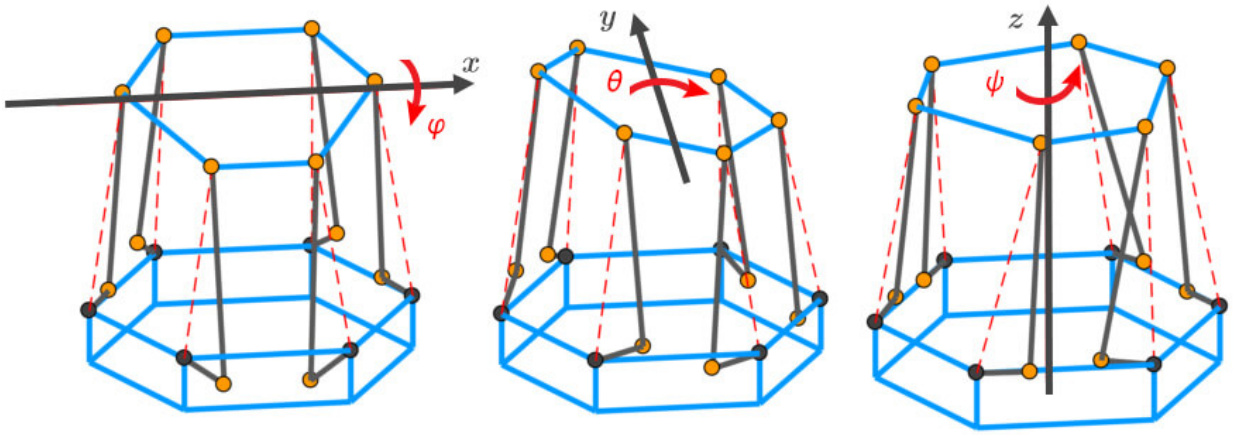
Upravljačke veličine su definisane vektorom $\boldsymbol{x}$ koji je dat u okviru sledeće jednačine i koji predstavlja *vektor spoljašnjih koordinata pokretne platforme mehanizma u odnosu na nepokretni koordinatni sistem baze.*

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & \varphi & \theta & \psi \end{bmatrix}^T \tag{4.36}$$

Na slici 4.17 se mogu videti tri translacije duž x, y i z osa koje predstavljaju prva tri stepena slobode platforme u odnosu na bazu simulatora.



Slika 4.17 - Tri translacije duž x, y i z osa



Slika 4.18 - Tri rotacije za uglove φ, θ i ψ

Na slici 4.18 se mogu videti tri rotacije oko x, y i z osa za uglove φ, θ i ψ. Ovi uglovi se nazivaju *Ojlerovi uglovi* i predstavljaju druga tri stepena slobode platforme u odnosu na bazu simulatora. U vazduhoplovstvu ovi uglovi najčešće imaju sledeće nazive:

- φ - Ugao valjanja,
- θ - Ugao propinjanja,
- ψ - Ugao skretanja.

Slaganjem svih mogućih kretanja prikazanih na prethodnim slikama se dobija šest nezavisnih stepena slobode. To znači da platforma u odnosu na bazu može da ostvari zadatu poziciju i orijentaciju.

Izračunavanje potrebnih rotacija obrtnih aktuatora i donjih poluga za dati položaj i orijentaciju platforme zahteva rešavanje inverznog kinematičkog problema. Najintuitivniji način rešavanja ovog tipa problema često se naziva *geometrijski metod*. Upotrebom vektora translacije $\boldsymbol{T}$ i matrice rotacije $\boldsymbol{R}$, moguće je dobiti vektor položaja tačke P_i u odnosu na globalni referentni sistem.

Neophodna dužina noge (udaljenost između tačaka B_i i P_i) može se izračunati pomoću Euklidove norme vektora $\boldsymbol{L_i}$ prikazanog na slici 4.11, kao $l_i = \|\boldsymbol{L_i}\|_2$. Nakon izračunavanja vektora $\boldsymbol{L_i}$, sledeći korak je pronalaženje ugla rotacije α_i za obrtni aktuator. Ovaj ugao definiše rotaciju oko ose aktuatora, dok je ova osa definisana jediničnim vektorom $\boldsymbol{i_{m,i}}$. Svrha pronalaženja ugla α_i je da se definiše položaj tačke A_i, koja obezbeđuje ispravan položaj gornjih i donjih poluga za definisanu dužinu noge. Ugao α_i meri se od početnog vektora $\boldsymbol{B_iA_i}$, koji leži u horizontalnoj ravni. Osa rotacije aktuatora i krug koji predstavlja sve moguće položaje tačke A_i takođe su prikazani na slici 4.11. Za uvedene referentne sisteme, sledeća relacija određuje vektor i-te noge [8].

$$\boldsymbol{L_i} = \boldsymbol{T} + \boldsymbol{R}\boldsymbol{p_i} - \boldsymbol{b_i} \tag{4.37}$$

U okviru vektora $\boldsymbol{T}$ se definiše z_b koordinata platforme koja predstavlja globalnu koordinatu platforme. Ovaj način definisanja translacije nije toliko intuitivan za potrebe upravljanja mehanizmom jer je radni prostor takav da platforma nikad ne može da dostigne nultu koordinatu. Zato se definiše početno odstupanje koordinate z_b veličinom z_0 i zatim se može relativno u odnosu na ovu koordinatu definisati kretanje platforme duž z ose. Zato se sada vektor translacije u zavisnosti od upravljačkih veličina definiše na sledeći način:

$$\boldsymbol{T} = \begin{bmatrix} x & y & z_0 + z \end{bmatrix}^T \tag{4.38}$$

Da bi se došlo do neke vrednosti z_0 potrebno je rešiti direktni kinematički problem za neke željene uglove $\boldsymbol{\alpha}$. Ovo je moguće jednostavno uraditi za slučaj kada su uglovi $\boldsymbol{\alpha}$ jednaki nuli i dobiti da je:

$$z_0 = \sqrt{s^2 - (x_{P_1} - x_{A_1})^2 - (y_{P_1} - y_{A_1})^2} + z_{A_1} \tag{4.39}$$

Kako je već pokazano da su u jednačinama 4.20, 4.21 i 4.22 nepoznati samo uglovi $\boldsymbol{\alpha}$, potrebno je na neki način iz ovih jednačina izraziti uglove kako bi se dobilo rešenje inverznog kinematičkog problema. Kada je $\boldsymbol{L_i}$ određeno korišćenjem jednačine 4.37, poznate su sve tri stranice trougla koji se obrazuje sa l_i, s i a.

$$\begin{aligned} a^2 &= (x_{Ai} - x_{Bi})^2 + (y_{Ai} - y_{Bi})^2 + (z_{Ai} - z_{Bi})^2 \\ &= ({x_{Ai}}^2 + {y_{Ai}}^2 + {z_{Ai}}^2) + ({x_{Bi}}^2 + {y_{Bi}}^2 + {z_{Bi}}^2) - 2(x_{Ai}x_{Bi} + y_{Ai}y_{Bi} + z_{Ai}z_{Bi}) \end{aligned} \tag{4.40}$$

$$\begin{aligned} l_i^2 &= (x_{Pi} - x_{Bi})^2 + (y_{Pi} - y_{Bi})^2 + (z_{Pi} - z_{Bi})^2 \\ &= ({x_{Pi}}^2 + {y_{Pi}}^2 + {z_{Pi}}^2) + ({x_{Bi}}^2 + {y_{Bi}}^2 + {z_{Bi}}^2) - 2(x_{Pi}x_{Bi} + y_{Pi}y_{Bi} + z_{Pi}z_{Bi}) \end{aligned} \tag{4.41}$$

$$\begin{aligned} s^2 &= (x_{Pi} - x_{Ai})^2 + (y_{Pi} - y_{Ai})^2 + (z_{Pi} - z_{Ai})^2 \\ &= ({x_{Pi}}^2 + {y_{Pi}}^2 + {z_{Pi}}^2) + ({x_{Ai}}^2 + {y_{Ai}}^2 + {z_{Ai}}^2) - 2(x_{Pi}x_{Ai} + y_{Pi}y_{Ai} + z_{Pi}z_{Ai}) \end{aligned} \tag{4.42}$$

Kombinacijom prethodne tri jednačine, gde se prve dve uvrste u treću, dobija se da je:

$$\begin{aligned} l_i^2 + a^2 - s^2 = {} & 2({x_{Bi}}^2 + {y_{Bi}}^2 + {z_{Bi}}^2) \\ & + 2x_{Ai}(x_{Pi} - x_{Bi}) + 2y_{Ai}(y_{Pi} - y_{Bi}) + 2z_{Ai}(z_{Pi} - z_{Bi}) \\ & - 2(x_{Pi}x_{Bi} + y_{Pi}y_{Bi} + z_{Pi}z_{Bi}) \end{aligned} \tag{4.43}$$

Kada se u prethodnu jednačinu uvrste koordinate iz jednačine 4.23 sledi da je:

$$\begin{aligned} l_i^2 + a^2 - s^2 = {} & \left(B_iA_{0ix} + (1 - \cos\alpha_i)\, Rm1_{B_iA_ix} + (\sin\alpha_i)\, Rm2_{B_iA_ix}\right) 2(x_{Pi} - x_{Bi}) \\ & + \left(B_iA_{0iy} + (1 - \cos\alpha_i)\, Rm1_{B_iA_iy} + (\sin\alpha_i)\, Rm2_{B_iA_iy}\right) 2(y_{Pi} - y_{Bi}) \\ & + \left(B_iA_{0iz} + (1 - \cos\alpha_i)\, Rm1_{B_iA_iz} + (\sin\alpha_i)\, Rm2_{B_iA_iz}\right) 2(z_{Pi} - z_{Bi}) \end{aligned} \tag{4.44}$$

Nakon sređivanja prethodne jednačine se dobija da je:

$$\boldsymbol{D} = \boldsymbol{E}\sin\boldsymbol{\alpha} + \boldsymbol{F}\cos\boldsymbol{\alpha} \tag{4.45}$$

Gde su $\boldsymbol{D}$, $\boldsymbol{E}$ i $\boldsymbol{F}$:

$$\boldsymbol{D} = \boldsymbol{l}^2 + a^2 - s^2 - 2(\boldsymbol{Rm1_{BA}} + \boldsymbol{BA_0})\,\boldsymbol{BP} \tag{4.46}$$

$$\boldsymbol{E} = 2\boldsymbol{Rm2_{BA}} \cdot \boldsymbol{BP} \tag{4.47}$$

$$\boldsymbol{F} = -2\boldsymbol{Rm1_{BA}} \cdot \boldsymbol{BP} \tag{4.48}$$

Jednačinu 4.45 treba svesti na oblik iz koga mogu da se izraze uglovi α_i. Desnu stranu jednačine treba napisati u obliku sa samo jednom trigonometrijskom funkcijom slično kao u [127]. Na osnovu adicione formule

$$\sin(\varphi+\theta) = \sin\varphi\cos\theta + \cos\varphi\sin\theta \tag{4.49}$$

se dobija da je

$$R\sin(\varphi+\theta) = R\sin\varphi\cos\theta + R\cos\varphi\sin\theta \tag{4.50}$$

Ukoliko se uvedu sledeće smene

$$R\sin\varphi = B \tag{4.51}$$

$$R\cos\varphi = A \tag{4.52}$$

dobija se da je

$$\begin{aligned} A\sin\theta + B\cos\theta &= R\sin(\varphi+\theta) \\ &= \sqrt{B^2+A^2}\sin\left(\varphi+\operatorname{atan2}(B,A)\right) \end{aligned} \tag{4.53}$$

Na osnovu analogije sa prethodnim sledi da je:

$$\boldsymbol{D} = \sqrt{\boldsymbol{E}^2+\boldsymbol{F}^2}\sin\left(\boldsymbol{\alpha}+\operatorname{atan2}(\boldsymbol{F},\boldsymbol{E})\right) \tag{4.54}$$

Nakon što se sada izrazi ugao α_i, dobija se:

$$\boldsymbol{\alpha} = \sin^{-1}\left(\frac{\boldsymbol{D}}{\sqrt{\boldsymbol{E}^2+\boldsymbol{F}^2}}\right) - \operatorname{atan2}(\boldsymbol{F},\boldsymbol{E}) \tag{4.55}$$

$$\operatorname{atan2}(F_i,E_i) = \begin{cases} \tan^{-1}\left(\dfrac{F_i}{E_i}\right), & E_i > 0, \\ \tan^{-1}\left(\dfrac{F_i}{E_i}\right)+\pi, & E_i < 0 \wedge F_i \geq 0, \\ \tan^{-1}\left(\dfrac{F_i}{E_i}\right)-\pi, & E_i < 0 \wedge F_i < 0, \\ +\dfrac{\pi}{2}, & E_i = 0 \wedge F_i > 0, \\ -\dfrac{\pi}{2}, & E_i = 0 \wedge F_i < 0, \\ \mathrm{NaN}, & E_i = 0 \wedge F_i = 0. \end{cases} \tag{4.56}$$

Sa dobijenom jednačinom 4.55, inverzni kinematički problem je rešen. Ukoliko ova jednačina ima realno rešenje za sve uglove α_i željeni položaj i orijentaciju platforme u odnosu na bazu simulatora moguće je ostvariti, ukoliko ne postoje druga fizička ograničenja.

Ovaj vektor uglova $\boldsymbol{\alpha}$ se nakon određivanja geometrijskih parametara i definisanja upravljačkih veličina određuje pomoću metode `getAlpha()` klase `MP_6RSS_SOLVER`.

U procesu projektovanja mehanizma poput ovog za praktičnu upotrebu kao simulator letenja, moraju se uzeti u obzir fizička ograničenja kinematičkih lanaca, kao što su mogućnost sudaranja poluga i ograničenja zglobova [17]. Kad god neki mehanizam ima sferne ili univerzalne zglobove, treba razmotriti njihove ugaone limite [63]. Radni opseg mora biti definisan za svaki zglob mehanizma. Najčešće dostupan sferni zglob naziva se zglob za kraj

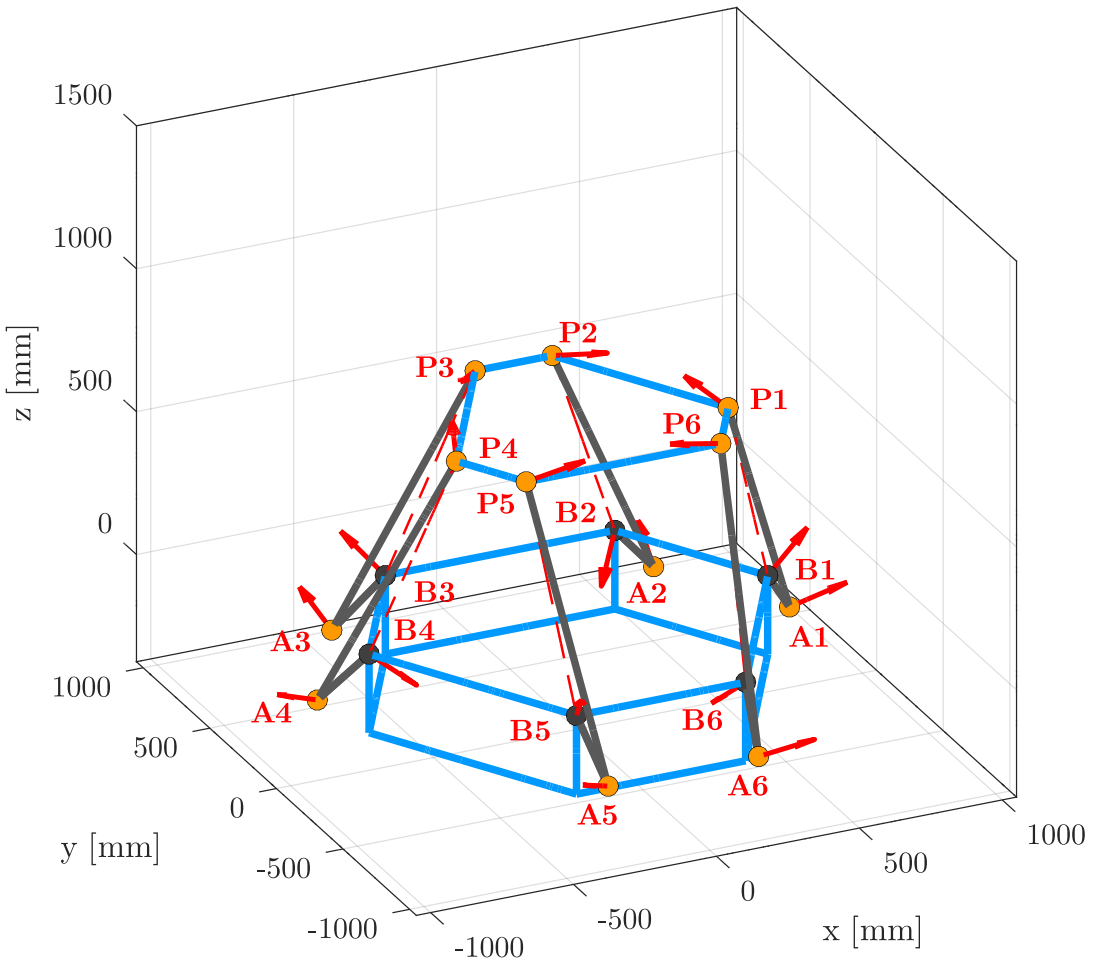
osovine. Nominalni položaj poluga za ovaj zglob je onaj u kojem su poluge međusobno ortogonalne. Na osnovu toga, ako je moguća rotacija između krajeva gornje poluge, ograničenja gornjih i donjih zglobova mogu se definisati na sledeći način:

$$\lambda_{p,i} = \left| \cos^{-1}\left(\frac{\boldsymbol{P_i A_i} \cdot \boldsymbol{i_{jpb,i}}}{s} \right) - 90° \right| < \lambda_{max}, \quad \boldsymbol{i_{jpb,i}} = \boldsymbol{R}\, \boldsymbol{i_{jp,i}} \tag{4.57}$$

$$\lambda_{b,i} = \left| \cos^{-1}\left(\frac{\boldsymbol{P_i A_i} \cdot \boldsymbol{i_{ll,i}}}{s} \right) - 90° \right| < \lambda_{max} \tag{4.58}$$

Maksimalni ugao konusa za ovu vrstu zgloba je λ_{max}, i potrebno ga je unapred odrediti za specifični zglob (ugao se najčešće kreće i do 35°). Provera da li uglovi u zglobovima premašuju dozvoljene je implementirana pomoću metode `checkJoints()` klase `MP_6RSS_SOLVER`.

Pošto se ovde analizira robotski mehanizam koji je potrebno upravljati u realnom vremenu, veoma je značajno smanjiti vreme koje je potrebno za proračun potrebnih uglova $\boldsymbol{\alpha}$ i sigurnosnih provera kao što je ova prikazana za uglove u zglobovima. Kod napisan u `JSLAB`-u se veoma lako prevodi u `C++` a sa druge strane omogućava se značajno brži rad na razvoju algoritma i programa nego pri direktnom korišćenju `C++` programskog jezika. Kompletan kod napisan u `JSLAB`-u je lako isporučiti na gotovo sve uređaje kao aplikacije ali je moguće napraviti i aplikaciju kojoj se pristupa preko interneta ili lokalno iz internet pregledača. Za izradu aplikacija se najčešće koriste `Electron` i `Cordova`. Ovako napisan kod prilikom projektovanja se odmah može ponovo iskoristiti i za upravljanje pravim mehanizmom.



Slika 4.19 - Uprošćen prikaz geometrije mehanizma

U okviru priloga C je data klasa `MP_6RSS_PLOTTER` kojom se vrši formiranje prikaza mehanizma, dok je u okviru priloga D data skripta `solveInverseKinematics.jsl` koja objedinjuje sve prethodno prikazano. Na slici 4.19 je prikazana uprošćena geometrija mehanizma sa značajnim jediničnim vektorima (prikaz dobijen metodama klase `MP_6RSS_PLOTTER`). Ovakav uprošćen prikaz je veoma značajan tokom analize jer omogućava jednostavnu vizuelnu inspekciju mehanizma definisanog određenim skupom parametara.

4.5 Direktni kinematički problem

Za rešavanje direktnog kinematičkog problema je potrebno rešiti sistem nelinearnih jednačina. Pored toga što su nelinearne, ove jednačine su i transcendentne. Transcendentne funkcije su analitičke funkcije koje ne mogu biti izražene na osnovu konačnog niza algebarskih operacija sabiranja, množenja i korenovanja.

Za rešavanje sistema nelinearnih jednačina se može iskoristiti Njutnov metod kao jedan od najstarijih poznatih metoda za iterativno rešavanje nelinearnih jednačina funkcije jedne promenljive[128]. Ovaj metod se izvorno primenjuje na rešavanje jednačine sledećeg oblika:

$$f(x) = 0 \tag{4.59}$$

gde je $f(x)$ nelinearna funkcija definisana na intervalu $[a, b]$. Pretpostavlja se da je f neprekidna na intervalu $[a, b]$ i da u njemu ima samo jednu nulu α, ne uzimajući u obzir višestrukosti.

Ovaj metod se može opisati pomoću sledećeg iterativnog procesa (gde je x^k vrednost x u k-toj iteraciji):

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}, \quad k \in \mathbb{N}_0 \tag{4.60}$$

gde je x^0 pogodno izabrana početna vrednost. Kao što se vidi, Njutnov metod može se primeniti na rešavanje jednačine 4.59 jedino u slučaju kada je funkcija f diferencijabilna u okolini rešenja jednačine.

Njutnov metod je moguće izvesti na osnovu *Tejlorovog razvoja funkcije*. Razvojem u Tejlorov polinom funkcije f u tački x koja je u okolini tačke α, sledi da je:

$$f(\alpha) = f(x) + f'(x)(\alpha - x) + \omega(\alpha - x)(\alpha - x) \tag{4.61}$$

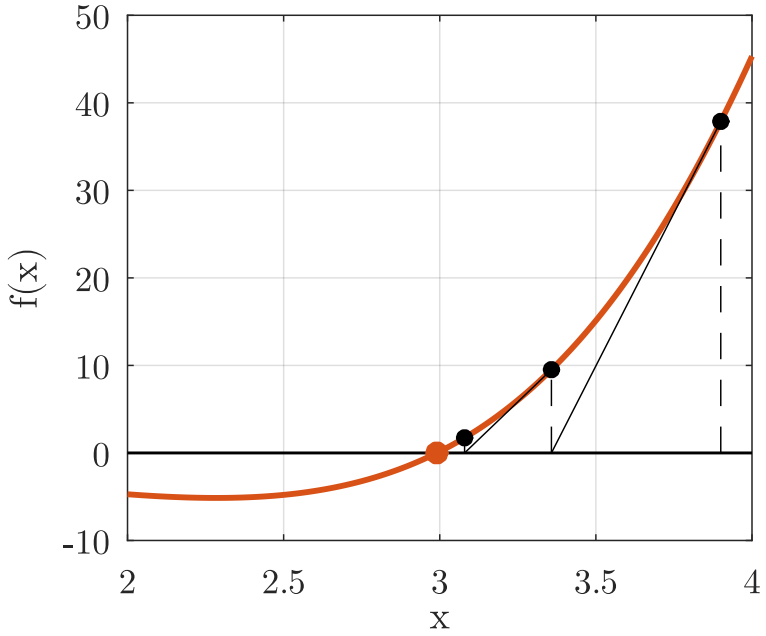
Ako se iskoristi to što je $f(\alpha) = 0$ i zanemari član uz funkciju ω, može da se izrazi α tako da je:

$$\alpha \approx x - \frac{f(x)}{f'(x)} \tag{4.62}$$

Kako se na ovaj način dobija samo aproksimacija rešenja jednačine, postupak se treba ponoviti sa novom vrednošću x. Na taj način se dobija Njutnov iterativni proces geometrijski ilustrovan na slici 4.20. Ovaj proces ima i jednostavnu geometrijsku interpretaciju. Ukoliko se posmatra jednačina:

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} \Rightarrow 0 = f(x^k) + f'(x^k)(x^{k+1} - x^k) \tag{4.63}$$

jednostavno može da se zaključi da je njeno rešenje, tačka x^{k+1}, tačka u kojoj tangenta na grafik funkcije f, u tački $(x^k, f(x^k))$, seče x-osu.



Slika 4.20 - Geometrijska ilustracija Njutnove metode

Za sisteme nelinearnih jednačina trenutno ne postoji metoda koja sigurno uvek dovodi do rešenja. U slučaju nelinearne jednačine jedne realne promenljive *metod bisekcije* uvek dovodi do rešenja, ako funkcija u rešenju menja znak. Njutnov metod za sistem nelinearnih jednačina ili *Metod Njutn-Kantoroviča* je jedan od najstarijih metoda za iterativno rešavanje sistema nelinearnih jednačina[128]. Ovaj metod se dobija uopštavanjem Njutnove metode na sistem nelinearnih jednačina oblika:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{4.64}$$

gde su $f_i : \mathbb{R}^n \to \mathbb{R}$ zadate nelinearne funkcije. Sistem se može napisati i u matričnom obliku:

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0} \tag{4.65}$$

gde je $\boldsymbol{F}(\boldsymbol{x}) = \left[f_1, f_2, \ldots, f_n\right]^T : \mathbb{R}^n \to \mathbb{R}^n$ vektor funkcija, $\boldsymbol{x} = \left[x_1, x_2, \ldots, x_n\right]^T$ vektor nezavisno promenljivih i $\boldsymbol{0}$ nula vektor iz $\mathbb{R}^n$.

Zapis je sličan onom korišćenom prilikom rešavanja jedne jednačine, analogno tom, iterativni proces se sada može formulisati na sledeći način:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - [\boldsymbol{J}(\boldsymbol{x}^k)]^{-1}\boldsymbol{F}(\boldsymbol{x}^k), \quad k \in \mathbb{N}_0 \tag{4.66}$$

gde je $\boldsymbol{J}$ *Jakobijeva matrica* definisana sa:

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \frac{\partial f_2(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_2(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\boldsymbol{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_n(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\boldsymbol{x})}{\partial x_n} \end{bmatrix} = \left[\frac{\partial f_i(\boldsymbol{x})}{\partial x_j}\right]_{i,j=1,2,\ldots,n} \tag{4.67}$$

Vektor $\boldsymbol{x}^0$ je sada vektor početnih vrednosti, koji treba tako izabrati da metod konvergira. Uvodi se veličina $\boldsymbol{d}^{k+1}$ koja se naziva korekcija metode.

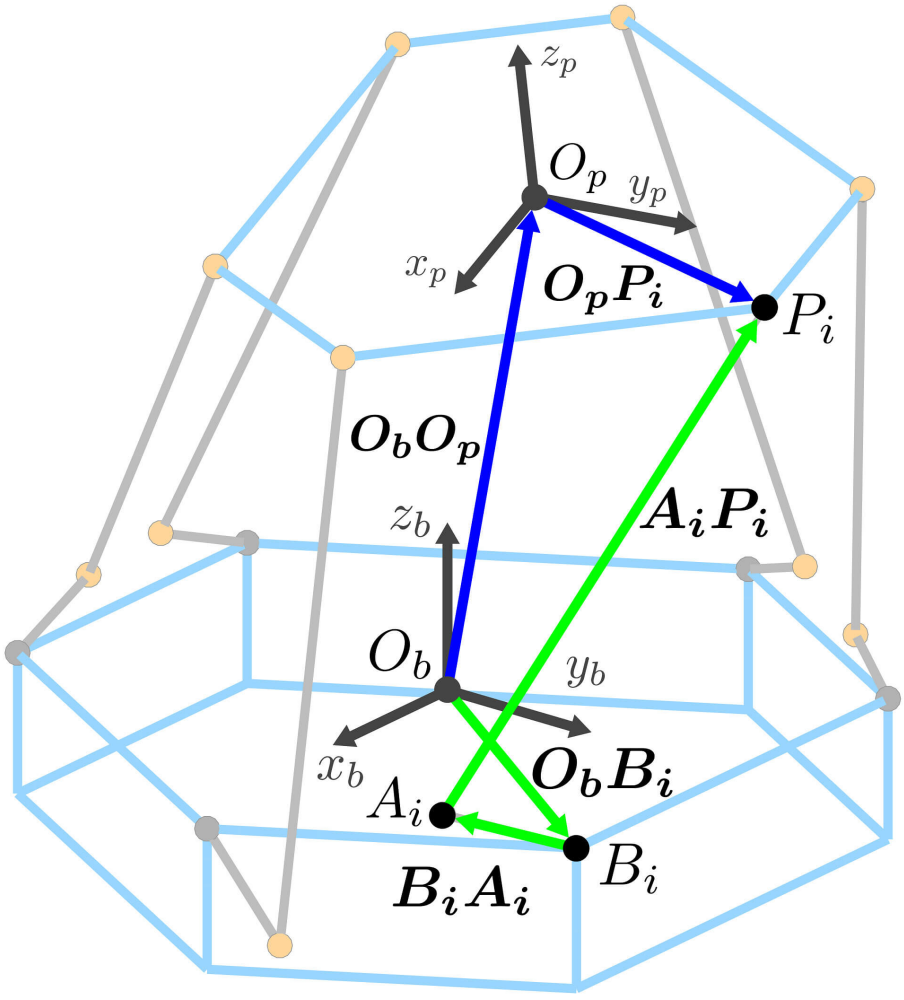
$$\boldsymbol{d}^{k+1} = -[\boldsymbol{J}(\boldsymbol{x}^k)]^{-1}\boldsymbol{F}(\boldsymbol{x}^k), \quad k \in \mathbb{N}_0 \tag{4.68}$$

U slučaju funkcije jedne promenljive Jakobijeva matrica se svodi na matricu prvog reda i to na $\boldsymbol{J}(x) = [f'(x)]$, pa inverzna matrica iznosi $[\frac{1}{f'(x)}]$. Na osnovu ovoga potvrđuje se potpuna analogija između metoda.

Kada je rešavana jednačina f, da bi se obezbedila konvergencija metode, zahtevalo se da izvod funkcije bude različit od nule u okolini rešenja. Kada se rešava sistem jednačina, zahteva se da je Jakobijeva matrica regularna u okolini rešenja. U suprotnom se ne može garantovati konvergencija metode.

Pokazano je da je potrebno poznavati Jakobijevu matricu sistema kako bi se iterativnim putem došlo do rešenja sistema. Jakobijeva matrica se u robotici koristi za mapiranje promene unutrašnjih koordinata u promenu spoljašnjih koordinata po vremenu, odnosno predstavlja vezu između brzina u zglobovima i brzina platforme. Na osnovu principa virtualnog rada, može se pokazati da je Jakobijeva matrica direktna veza između potrebnih momenata aktuatora i spoljašnjeg opterećenja platforme.

U slučaju problema inverzne kinematike, spoljašnje koordinate su poznate, ali ako je potrebno odrediti spoljašnje koordinate na osnovu poznatih orijentacija donjih poluga, to bi zahtevalo rešavanje direktnog kinematičkog problema. Direktna kinematika ovog mehanizma je veoma složena i uložen je veliki trud u njeno rešavanje, kako analitički, tako i numerički.



Slika 4.21 - Vektorska petlja jedne noge mehanizma

Kao što se vidi na slici 4.21, za svaku nogu mehanizma se može napisati vektorska jednačina zatvorene petlje na sledeći način:

$$\boldsymbol{O_bB_i} + \boldsymbol{B_iA_i} + \boldsymbol{A_iP_i} = \boldsymbol{O_bO_p} + \boldsymbol{O_pP_i} \tag{4.69}$$

Na osnovu već uvedenih oznaka u okviru jednačina 4.2 i 4.3 i činjenice da vektor položaja koordinatnog početka koordinatog sistema platforme u odnosu na koordinatni sistem baze predstavlja vektor translacije platforme, prethodna jednačina postaje:

$$\boldsymbol{b_i} + \boldsymbol{B_iA_i} + \boldsymbol{A_iP_i} = \boldsymbol{T} + \boldsymbol{O_pP_i} \tag{4.70}$$

Diferenciranjem jednačine 4.70 po vremenu se dobija:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{b_i}) + \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{B_iA_i}) + \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{A_iP_i}) = \boldsymbol{v_p} + \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{O_pP_i}) \tag{4.71}$$

Da bi se dalje sređivao izraz potrebno je primeniti *Ojlerov obrazac* za brzinu tačke, koji glasi:

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{\omega} \times \boldsymbol{r} \tag{4.72}$$

Korišćenjem Ojlerovog obrazca jednačina 4.71 postaje:

$$\boldsymbol{\omega_{\alpha,i}} \times \boldsymbol{B_iA_i} + \boldsymbol{\omega_{ul,i}} \times \boldsymbol{A_iP_i} = \boldsymbol{v_p} + \boldsymbol{\omega_p} \times \boldsymbol{O_pP_i} \tag{4.73}$$

gde je $\boldsymbol{v_p}$ translatorna brzina platforme, $\boldsymbol{\omega_p}$, $\boldsymbol{\omega_{ul,i}}$ i $\boldsymbol{\omega_{\alpha,i}}$ su ugaone brzine aktuatora (odnosno donjih poluga), gornjih poluga i platforme. Na osnovu toga što je dužina gornjih poluga konstanta, može se uvesti jedinični vektor gornje poluga tako da je:

$$\boldsymbol{A_iP_i} = s\boldsymbol{i_{ul,i}} \tag{4.74}$$

Kada se jednačina 4.73 pomnoži sa $\boldsymbol{i_{ul,i}}$, jediničnim vektorom gornje poluge, i korišćenjem osobina mešovitog proizvoda dobije se da je:

$$\boldsymbol{i_{ul,i}}\left(\boldsymbol{\omega_{\alpha,i}} \times \boldsymbol{B_iA_i}\right) + \boldsymbol{\omega_{ul,i}}\left(s\boldsymbol{i_{ul,i}} \times \boldsymbol{i_{ul,i}}\right) = \boldsymbol{i_{ul,i}}\,\boldsymbol{v_p} + \left(\boldsymbol{O_pP_i} \times \boldsymbol{i_{ul,i}}\right)\boldsymbol{\omega_p} \tag{4.75}$$

Vektorski proizvod vektora sa samim sobom je jednak nuli, na osnovu čega se gubi drugi član sa leve strane prethodne jednačine. Vektor ugaone brzine donje poluge se može izraziti preko jediničnog vektora obrtne ose aktuatora i intenziteta ugaone brzine, pored toga neophodno je uzeti u obzir i smer ugaone brzine, sledi da je:

$$\boldsymbol{\omega_{\alpha,i}} = \dot{\alpha}_i\,\boldsymbol{i_{m,i}} \tag{4.76}$$

Ukoliko se prethodno izraženo iskoristi, jednačina 4.75 postaje:

$$\boldsymbol{i_{ul,i}}\left(\boldsymbol{i_{m,i}} \times \boldsymbol{B_iA_i}\right)\,\dot{\alpha}_i = \boldsymbol{i_{ul,i}}\,\boldsymbol{v_p} + \left(\boldsymbol{O_pP_i} \times \boldsymbol{i_{ul,i}}\right)\boldsymbol{\omega_p} \tag{4.77}$$

Na osnovu prethodne jednačine, Jakobijeve matrice spoljašnjih koordinata $\boldsymbol{J_x}$ i unutrašnjih koordinata $\boldsymbol{J_\alpha}$ definišu se kao:

$$\boldsymbol{J_\alpha}\dot{\boldsymbol{\alpha}} = \boldsymbol{J_x}\dot{\boldsymbol{x}} \tag{4.78}$$

$$\boldsymbol{J_x} = \begin{bmatrix} \boldsymbol{i_{ul,1}} & \boldsymbol{O_pP_1} \times \boldsymbol{i_{ul,1}} \\ \vdots & \vdots \\ \boldsymbol{i_{ul,6}} & \boldsymbol{O_pP_6} \times \boldsymbol{i_{ul,6}} \end{bmatrix} \tag{4.79}$$

$$\boldsymbol{J_\alpha} = \begin{bmatrix} \boldsymbol{i_{ul,1}} \cdot \left(\boldsymbol{i_{m,1}} \times \boldsymbol{B_1 A_1}\right) & & \\ & \ddots & \\ & & \boldsymbol{i_{ul,6}} \cdot \left(\boldsymbol{i_{m,6}} \times \boldsymbol{B_6 A_6}\right) \end{bmatrix} \tag{4.80}$$

Vektori spoljašnjih brzina $\dot{\boldsymbol{x}}$ i unutrašnjih brzina $\dot{\boldsymbol{\alpha}}$ su:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} v_{px} & v_{py} & v_{pz} & \omega_{px} & \omega_{py} & \omega_{pz} \end{bmatrix}^T \tag{4.81}$$

$$\dot{\boldsymbol{\alpha}} = \begin{bmatrix} \dot{\alpha}_1 & \cdots & \dot{\alpha}_6 \end{bmatrix}^T \tag{4.82}$$

Na kraju, Jakobijeva matrica mehanizma se dobija na sledeći način:

$$\boldsymbol{J} = \boldsymbol{J_x}^{-1} \boldsymbol{J_\alpha} \tag{4.83}$$

Za dalju analizu je značajna i inverzna Jakobijeva matrica.

$$\boldsymbol{J}^{-1} = \boldsymbol{J_\alpha}^{-1} \boldsymbol{J_x} \tag{4.84}$$

Na osnovu prethodne jednačine, određivanje Jakobijeve matrice je implementirano u okviru metode `getJacobian()` klase `MP_6RSS_SOLVER`.

Kada je određena Jakobijeva matrica moguće je formulisati Njutnov metod za određivanje spoljašnjih koordinata, dobija se da je:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \boldsymbol{J}^{-1}(\boldsymbol{x}^k)\boldsymbol{F}(\boldsymbol{x}^k) \tag{4.85}$$

gde je $\boldsymbol{F}$ vektor šest nelinearnih funkcija, po jedna za svaki aktuator. Na osnovu rešenja inverznog kinematičkog problema i uvođenjem $\boldsymbol{\alpha}^*$ kao vektora željenih uglova otklona donjih poluga, može se napisati da je:

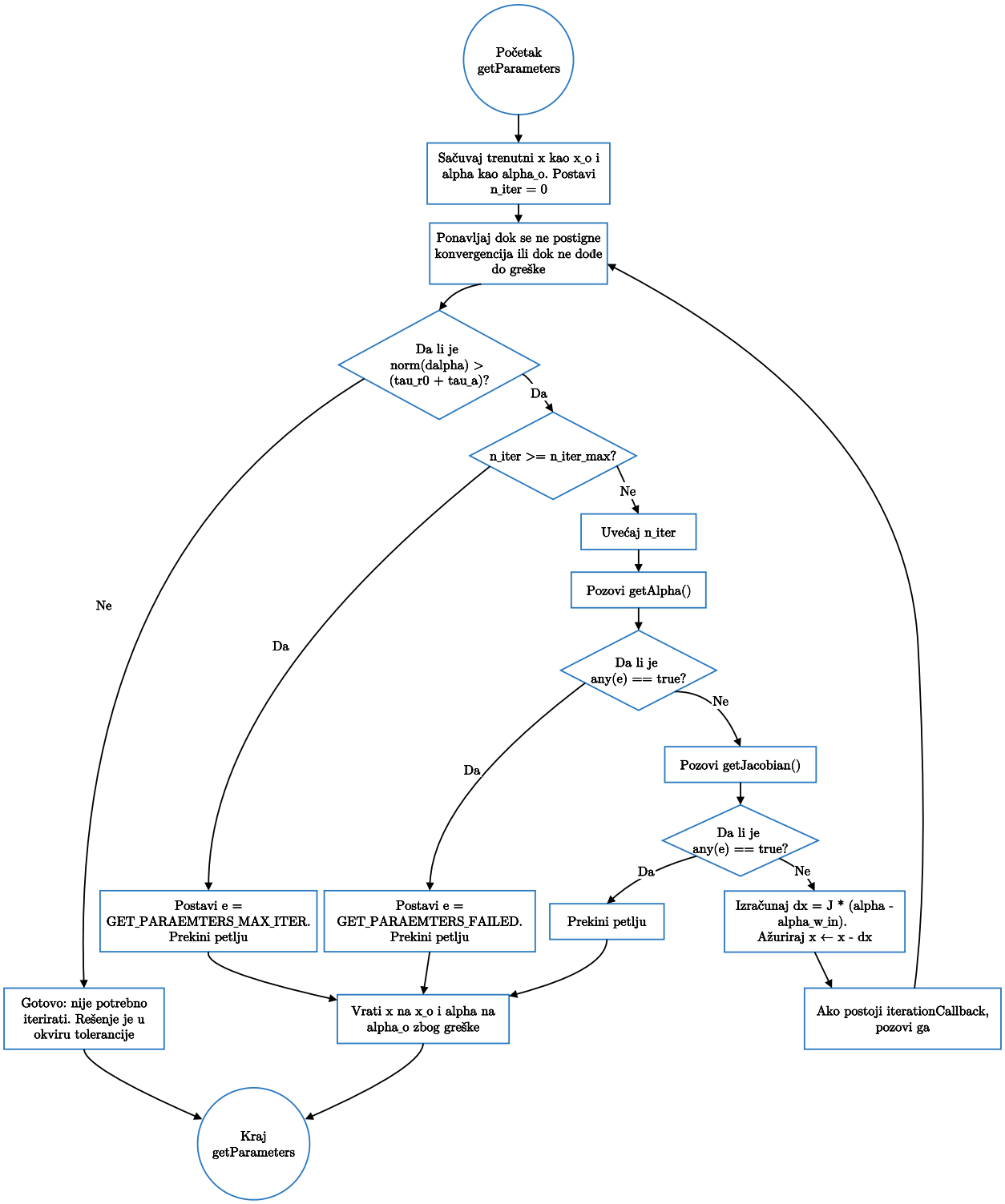
$$\boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) & \cdots & f_6(\boldsymbol{x}) \end{bmatrix}^T \tag{4.86}$$

$$f_i(\boldsymbol{x}) = \sin^{-1}\left(\frac{D_i}{\sqrt{E_i^2 + F_i^2}}\right) - \operatorname{atan2}\left(F_i, E_i\right) - \alpha_i^* \tag{4.87}$$

Treba uočiti da je $\boldsymbol{J}^{-1}(\boldsymbol{x}^k)$ u ovom slučaju jednako sa $\boldsymbol{J}$ iz jednačine 4.83. Ako se jednačina 4.85 napiše preko rešenja inverznog kinematičkog problema i vektora željenih uglova otklona donjih poluga dobija se da je:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \boldsymbol{J}\left(\boldsymbol{\alpha}(\boldsymbol{x}^k) - \boldsymbol{\alpha}^*\right) \tag{4.88}$$

Na ovaj način dobija se jednostavno iterativno numeričko rešenje problema direktne kinematike Stjuartove platforme sa obrtnim aktuatorima. Dobra aproksimacija (sa greškom manjom od definisanog praga [40]) spoljašnjih koordinata dobija se iz jednačine 4.88 nakon samo nekoliko iteracija.



Slika 4.22 - Algoritam za rešavanje direktnog kinematičkog problema

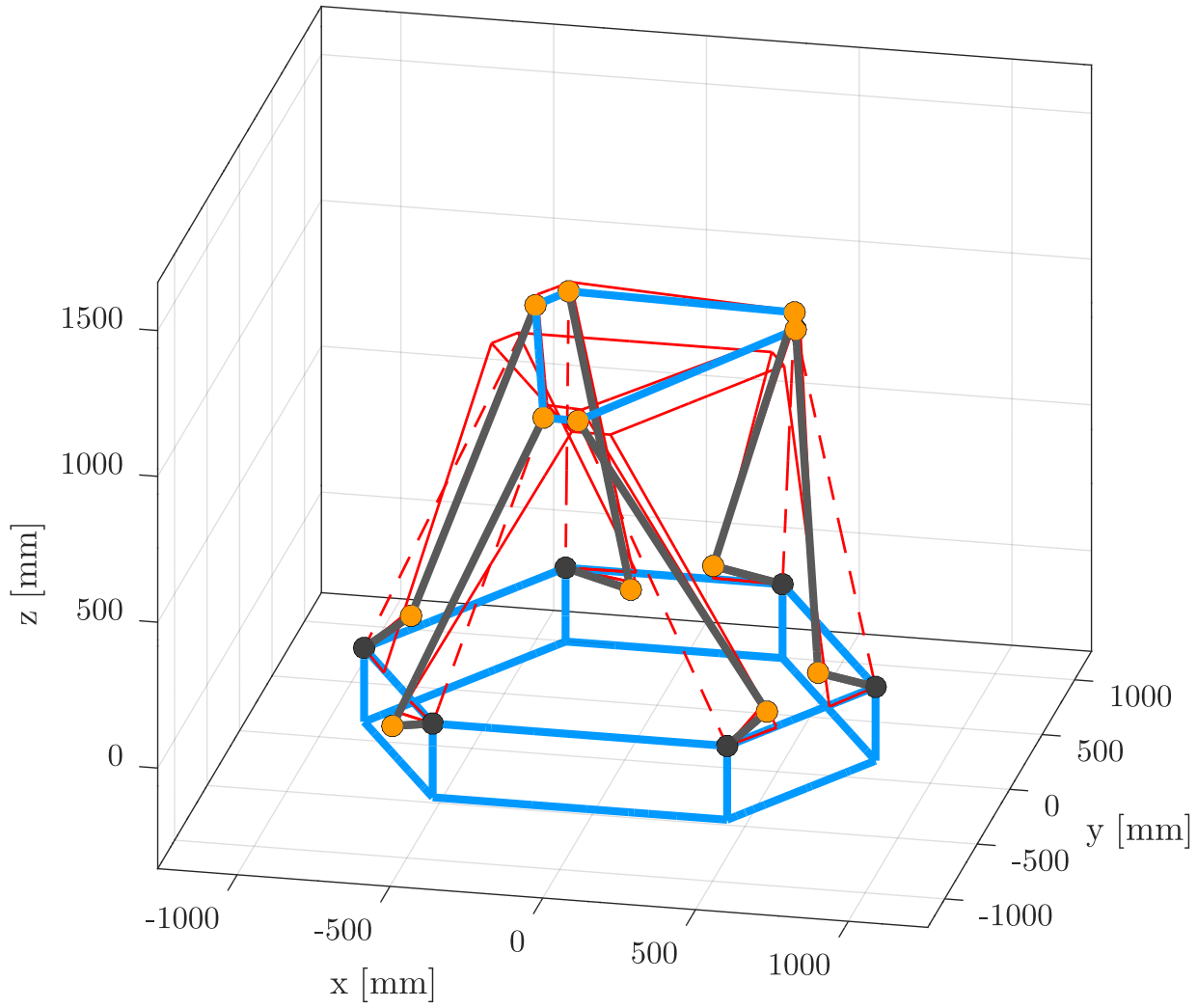
Prethodna jednačina i Njutnov metod su implementirani u okviru metode `getParameters()` klase `MP_6RSS_SOLVER` (prema algoritmu prikazanom na slici 4.22). Pored željenih vrednosti uglova otklona donjih poluga, metoda koristi i unete dozvoljene relativne i apsolutne greške na osnovu čega se prekida iterativni postupak za rešavanje direktne kinematike. Tokom upravljanja se za početni položaj uzima prethodni poznati položaj ili trenutno izmereni položaj što značajno smanjuje broj potrebnih iteracija.

U okviru priloga E je data skripta `solveForwardKinematics.jsl` kojom se sve prethodno prikazane metode povezuju u proces rešavanja direktnog kinematičkog problema.

U sledećim tabelama (tabele 4.2 i 4.3) i u okviru grafika (slike 4.24 i 4.25) su prikazani rezultati proračuna direktne kinematike po iteracijama za konfiguraciju iz tabele 4.1. Prikazane su promene i unutrašnjih i spoljašnjih koordinata iz nultog početnog položaja za vektor željenih uglova donjih poluga:

$$\boldsymbol{\alpha}^* = \begin{bmatrix} 30 & 20 & -30 & 55 & -25 & 15 \end{bmatrix}^T \tag{4.89}$$

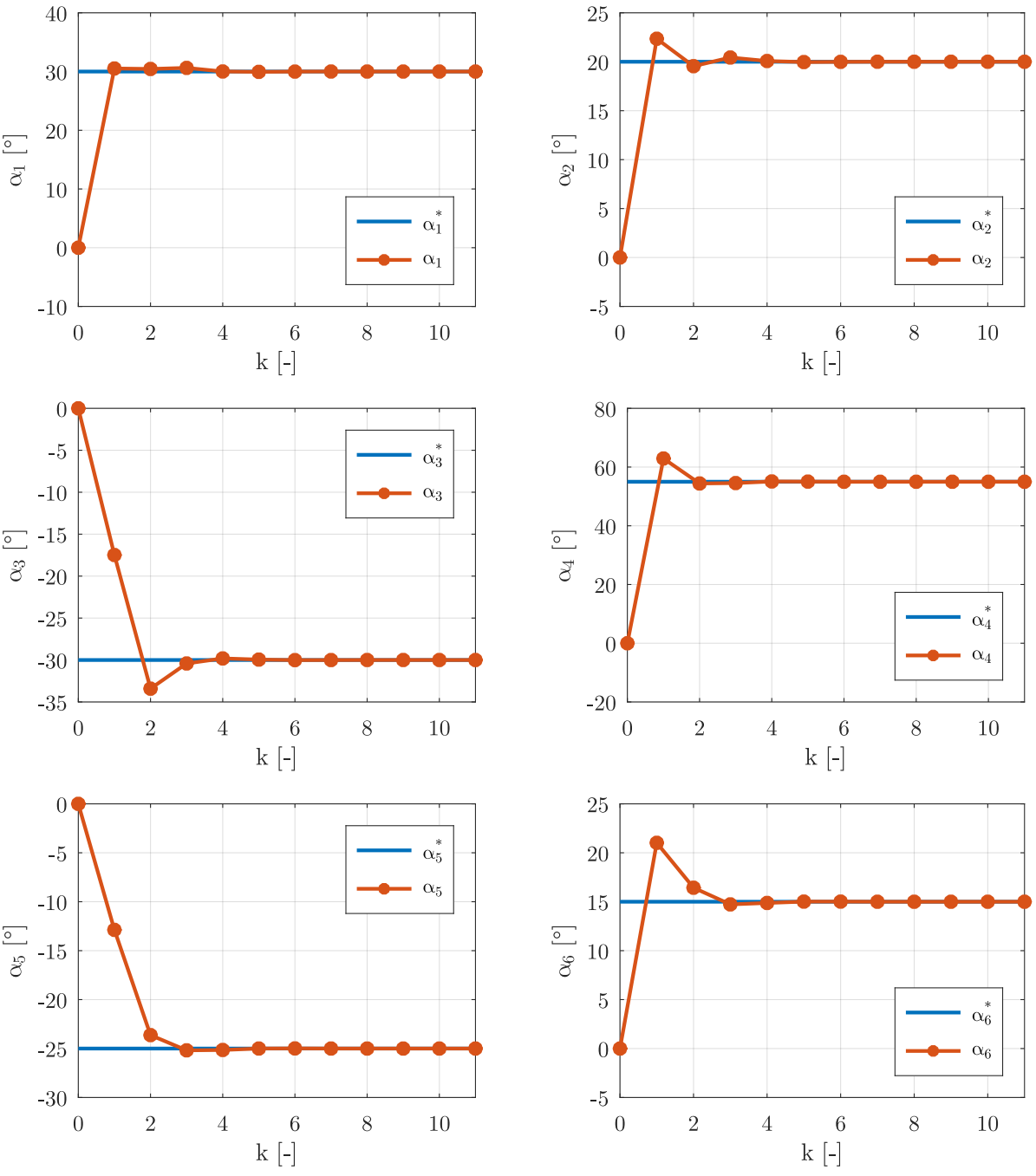
Na slici 4.23 je prikazan dobijeni položaj i orijentacija mehanizma na osnovu položaja donjih poluga, sa prikazom među-rešenja.



Slika 4.23 - Prikaz mehanizma sa među rešenjima

Tabela 4.2 - Unutrašnje koordinate po iteracijama

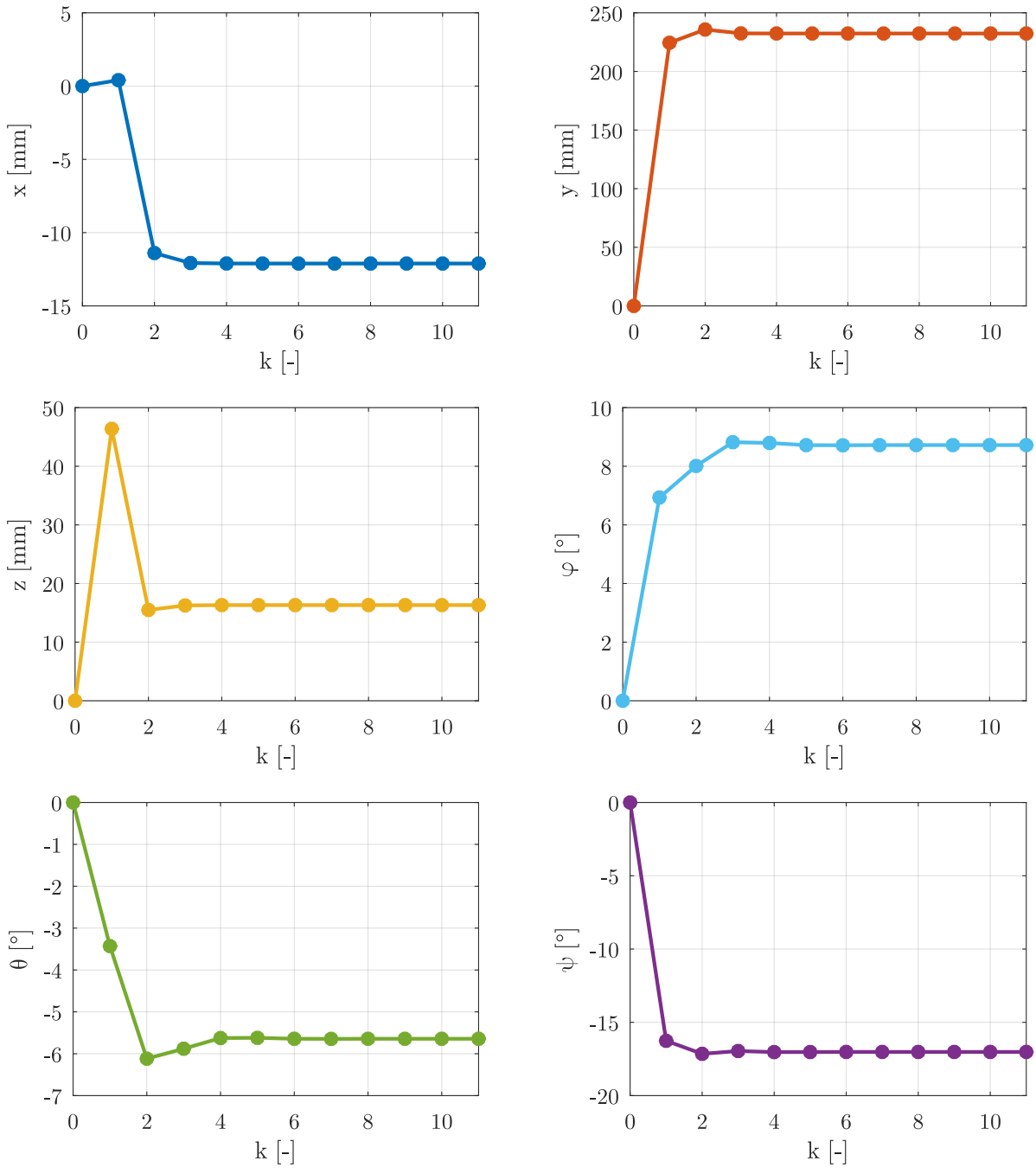
k	α_1 [°]	α_2 [°]	α_3 [°]	α_4 [°]	α_5 [°]	α_6 [°]
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	30.5190	22.3510	-17.4829	62.9146	-12.8751	21.0196
2	30.4386	19.5486	-33.4141	54.4137	-23.6279	16.4304
3	30.6205	20.4315	-30.4118	54.5506	-25.1830	14.7282
4	30.0109	20.0759	-29.8107	55.0961	-25.1469	14.8746
5	29.9457	19.9686	-29.9500	55.0501	-24.9985	15.0123
6	29.9941	19.9905	-30.0121	54.9964	-24.9870	15.0121
7	30.0042	20.0019	-30.0055	54.9953	-24.9989	15.0000
8	30.0009	20.0010	-29.9995	54.9999	-25.0010	14.9989
9	29.9997	19.9999	-29.9995	55.0004	-25.0002	14.9999
10	29.9999	19.9999	-30.0000	55.0000	-24.9999	15.0001
11	30.0000	20.0000	-30.0000	55.0000	-25.0000	15.0000



Slika 4.24 - Promena unutrašnjih koordinata

Tabela 4.3 - Spoljašnje koordinate po iteracijama

k	x [mm]	y [mm]	z [mm]	ψ [°]	θ [°]	φ [°]
0	0	0	0	0	0	0
1	0.4034	224.4288	46.3878	-16.2598	-3.4294	6.9329
2	-11.3930	235.7242	15.4899	-17.1489	-6.1178	8.0107
3	-12.0741	232.4458	16.2592	-16.9583	-5.8805	8.8198
4	-12.1049	232.3570	16.3180	-17.0307	-5.6250	8.7956
5	-12.1081	232.3499	16.3180	-17.0279	-5.6207	8.7199
6	-12.1081	232.3498	16.3181	-17.0204	-5.6425	8.7150
7	-12.1081	232.3497	16.3181	-17.0199	-5.6449	8.7211
8	-12.1081	232.3497	16.3181	-17.0206	-5.6432	8.7221
9	-12.1081	232.3497	16.3181	-17.0207	-5.6429	8.7217
10	-12.1081	232.3497	16.3181	-17.0206	-5.6430	8.7216
11	-12.1081	232.3497	16.3181	-17.0206	-5.6430	8.7216



Slika 4.25 - Promena spoljašnjih koordinata

4.6 Brzine i ubrzanja mehanizma

Određivanje brzina i ubrzanja mehanizma je neophodno za rešavanje problema dinamike i u nastavku prikazanog određivanja potrebnog obrtnog momenta aktuatora.

4.6.1 Brzine mehanizma

Kada je poznata ugaona brzina motora $\boldsymbol{\omega_\alpha}$ i kada je poznato da su tačke B_i nepomične, moguće je odrediti vektor brzine tačaka A_i, na osnovu:

$$\boldsymbol{v_{A_i}} = \boldsymbol{v_{B_i}} + \boldsymbol{\omega_{\alpha,i}} \times (\boldsymbol{b_i} - \boldsymbol{a_i}) = a \left|\boldsymbol{\omega_{\alpha,i}}\right| (\boldsymbol{i_{m,i}} \times \boldsymbol{i_{ll,i}}) \tag{4.90}$$

Kada je Jakobijeva matrica poznata, na osnovu poznatih ugaonih brzina motora, moguće je odrediti brzine platforme na osnovu:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{\alpha}} \tag{4.91}$$

Vektor brzina platforme je definisan pomoću translatornih i ugaonih komponenti tako da je:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v_p} & \boldsymbol{\omega_p} \end{bmatrix}^T \tag{4.92}$$

Kada su poznati vektori translatorne i ugaone brzine platforme moguće je odrediti brzine tačaka P_i, na osnovu:

$$\boldsymbol{v_{P_i}} = \boldsymbol{v_p} + \boldsymbol{\omega_p} \times \boldsymbol{p_i} \tag{4.93}$$

Potrebno je voditi računa da je vektor $\boldsymbol{p_i}$ definisan u lokalnom koordinatnom sistemu platforme. Na osnovu prethodnog izraza moguće je izračunati i brzinu bilo koje druge tačke na platformi ukoliko je poznat njen vektor položaja u koordinatnom sistemu platforme.

Sa druge strane, brzinu tačke P_i moguće je izraziti i pomoću:

$$\boldsymbol{v_{P_i}} = \boldsymbol{v_{A_i}} + \boldsymbol{\omega_{ul,i}} \times \boldsymbol{A_iP_i} \tag{4.94}$$

Nakon izjednačavanja izraza 4.93 i 4.94, moguće je daljim sređivanjem dobiti izraz za određivanje $\boldsymbol{\omega_{ul,i}}$ kao ugaone brzine gornjih poluga.

$$\boldsymbol{v_p} + \boldsymbol{\omega_p} \times \boldsymbol{p_i} = \boldsymbol{v_{A_i}} + \boldsymbol{\omega_{ul,i}} \times \boldsymbol{A_iP_i} \tag{4.95}$$

Grupisanjem i vektorskim množenjem obe strane izraza sa $\boldsymbol{A_iP_i}$ se dobija:

$$(\boldsymbol{\omega_{ul,i}} \times \boldsymbol{A_iP_i}) \times \boldsymbol{A_iP_i} = (\boldsymbol{v_p} - \boldsymbol{v_{A_i}} + \boldsymbol{\omega_p} \times \boldsymbol{p_i}) \times \boldsymbol{A_iP_i} \tag{4.96}$$

Na osnovu identiteta za trostruki vektorski proizvod se može napisati da je:

$$(\boldsymbol{A_iP_i} \cdot \boldsymbol{\omega_{ul,i}})\, \boldsymbol{A_iP_i} - (\boldsymbol{A_iP_i} \cdot \boldsymbol{A_iP_i})\, \boldsymbol{\omega_{ul,i}} = (\boldsymbol{v_p} - \boldsymbol{v_{A_i}} + \boldsymbol{\omega_p} \times \boldsymbol{p_i}) \times \boldsymbol{A_iP_i} \tag{4.97}$$

S obzirom na ortogonalnost vektora $\boldsymbol{\omega_{ul,i}}$ i $\boldsymbol{A_iP_i}$ ($\boldsymbol{A_iP_i} \cdot \boldsymbol{\omega_{ul,i}} = 0$) sledi da je izraz za određivanje ugaone brzine gornjih poluga:

$$\boldsymbol{\omega_{ul,i}} = \frac{-1}{\left|\boldsymbol{A_iP_i}\right|^2} (\boldsymbol{v_p} - \boldsymbol{v_{A_i}} + \boldsymbol{\omega_p} \times \boldsymbol{p_i}) \times \boldsymbol{A_iP_i} \tag{4.98}$$

4.6.2 Ubrzanja mehanizma

Kada su poznate ugaona brzina i ugaono ubrzanje motora, moguće je odrediti ubrzanje tačaka A_i na osnovu:

$$\begin{aligned} \boldsymbol{a}_{\boldsymbol{A}_i} &= \boldsymbol{a}_{\boldsymbol{B}_i} + \boldsymbol{\epsilon}_{\boldsymbol{\alpha}_i} \times (\boldsymbol{b}_i - \boldsymbol{a}_i) + \boldsymbol{\omega}_{\boldsymbol{\alpha},i} \times (\boldsymbol{\omega}_{\boldsymbol{\alpha},i} \times (\boldsymbol{b}_i - \boldsymbol{a}_i)) \\ &= a\, \boldsymbol{\epsilon}_{\boldsymbol{\alpha}_i} \times \boldsymbol{i}_{\boldsymbol{ll},i} + a\, \boldsymbol{\omega}_{\boldsymbol{\alpha},i} \times (\boldsymbol{\omega}_{\boldsymbol{\alpha},i} \times \boldsymbol{i}_{\boldsymbol{ll},i}) \end{aligned} \tag{4.99}$$

Da bi se odredilo translatorno i ugaono ubrzanje platforme na osnovu poznatih brzina moguće je diferencirati jednačinu 4.78 i dobiti sledeće:

$$\dot{\boldsymbol{J}}_{\boldsymbol{\alpha}} \dot{\boldsymbol{\alpha}} + \boldsymbol{J}_{\boldsymbol{\alpha}} \ddot{\boldsymbol{\alpha}} = \dot{\boldsymbol{J}}_{\boldsymbol{x}} \dot{\boldsymbol{x}} + \boldsymbol{J}_{\boldsymbol{x}} \ddot{\boldsymbol{x}} \tag{4.100}$$

Na osnovu prethodne jednačine moguće je izraziti vektor ubrzanja platforme tako da je:

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{x}}^{-1} \left(\dot{\boldsymbol{J}}_{\boldsymbol{\alpha}} \dot{\boldsymbol{\alpha}} + \boldsymbol{J}_{\boldsymbol{\alpha}} \ddot{\boldsymbol{\alpha}} - \dot{\boldsymbol{J}}_{\boldsymbol{x}} \dot{\boldsymbol{x}} \right) \tag{4.101}$$

U prethodnoj jednačini nisu još definisani samo $\dot{\boldsymbol{J}}_{\boldsymbol{\alpha}}$ i $\dot{\boldsymbol{J}}_{\boldsymbol{x}}$ koji se dobijaju diferenciranjem po vremenu $\boldsymbol{J}_{\boldsymbol{\alpha}}$ i $\boldsymbol{J}_{\boldsymbol{x}}$, tako da su:

$$\dot{\boldsymbol{J}}_{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{\omega}_{\boldsymbol{ul},1} \times \boldsymbol{i}_{\boldsymbol{ul},1} & (\boldsymbol{\omega}_p \times \boldsymbol{O_pP_1}) \times \boldsymbol{i}_{\boldsymbol{ul},1} + \boldsymbol{O_pP_1} \times (\boldsymbol{\omega}_{\boldsymbol{ul},1} \times \boldsymbol{i}_{\boldsymbol{ul},1}) \\ \vdots & \vdots \\ \boldsymbol{\omega}_{\boldsymbol{ul},6} \times \boldsymbol{i}_{\boldsymbol{ul},6} & (\boldsymbol{\omega}_p \times \boldsymbol{O_pP_6}) \times \boldsymbol{i}_{\boldsymbol{ul},6} + \boldsymbol{O_pP_6} \times (\boldsymbol{\omega}_{\boldsymbol{ul},6} \times \boldsymbol{i}_{\boldsymbol{ul},6}) \end{bmatrix} \tag{4.102}$$

$$\dot{\boldsymbol{J}}_{\boldsymbol{\alpha}} = \operatorname{diag} \left(\begin{bmatrix} (\boldsymbol{\omega}_{\boldsymbol{ul},1} \times \boldsymbol{i}_{\boldsymbol{ul},1}) \cdot \left(\boldsymbol{i}_{\boldsymbol{m},1} \times \boldsymbol{B_1A_1} \right) + \boldsymbol{i}_{\boldsymbol{ul},1} \cdot \left(\boldsymbol{i}_{\boldsymbol{m},1} \times \left(\omega_{\alpha,1}\, \boldsymbol{i}_{\boldsymbol{m},1} \times \boldsymbol{B_1A_1} \right) \right) \\ \vdots \\ (\boldsymbol{\omega}_{\boldsymbol{ul},6} \times \boldsymbol{i}_{\boldsymbol{ul},6}) \cdot \left(\boldsymbol{i}_{\boldsymbol{m},6} \times \boldsymbol{B_6A_6} \right) + \boldsymbol{i}_{\boldsymbol{ul},6} \cdot \left(\boldsymbol{i}_{\boldsymbol{m},6} \times \left(\omega_{\alpha,6}\, \boldsymbol{i}_{\boldsymbol{m},6} \times \boldsymbol{B_6A_6} \right) \right) \end{bmatrix} \right) \tag{4.103}$$

Vektor ubrzanja platforme je definisan pomoću translatornih i ugaonih komponenti tako da je:

$$\ddot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{a}_p & \boldsymbol{\epsilon}_p \end{bmatrix}^T \tag{4.104}$$

Kada su poznati vektori translatornog i ugaonog ubrzanja platforme moguće je odrediti ubrzanja tačaka P_i, na osnovu:

$$\boldsymbol{a}_{\boldsymbol{P}_i} = \boldsymbol{a}_p + \boldsymbol{\epsilon}_p \times \boldsymbol{p}_i + \boldsymbol{\omega}_p \times (\boldsymbol{\omega}_p \times \boldsymbol{p}_i) \tag{4.105}$$

Sa druge strane, ubrzanje tačke P_i je moguće izraziti i pomoću:

$$\boldsymbol{a}_{\boldsymbol{P}_i} = \boldsymbol{a}_{\boldsymbol{A}_i} + \boldsymbol{\epsilon}_{\boldsymbol{ul},i} \times \boldsymbol{A_iP_i} + \boldsymbol{\omega}_{\boldsymbol{ul},i} \times (\boldsymbol{\omega}_{\boldsymbol{ul},i} \times \boldsymbol{A_iP_i}) \tag{4.106}$$

Nakon izjednačavanja izraza 4.105 i 4.106 moguće je daljim sređivanjem dobiti izraz za određivanje $\boldsymbol{\epsilon}_{\boldsymbol{ul},i}$ kao ugaonog ubrzanja gornjih poluga.

$$\boldsymbol{a}_p + \boldsymbol{\epsilon}_p \times \boldsymbol{p}_i + \boldsymbol{\omega}_p \times (\boldsymbol{\omega}_p \times \boldsymbol{p}_i) = \boldsymbol{a}_{\boldsymbol{A}_i} + \boldsymbol{\epsilon}_{\boldsymbol{ul},i} \times \boldsymbol{A_iP_i} + \boldsymbol{\omega}_{\boldsymbol{ul},i} \times (\boldsymbol{\omega}_{\boldsymbol{ul},i} \times \boldsymbol{A_iP_i}) \tag{4.107}$$

Grupisanjem i vektorskim množenjem obe strane izraza sa $\boldsymbol{A_iP_i}$ se dobija:

$$(\boldsymbol{\epsilon}_{ul,i} \times \boldsymbol{A_iP_i}) \times \boldsymbol{A_iP_i} = (\boldsymbol{a_p} - \boldsymbol{a_{A_i}} + \boldsymbol{\epsilon_p} \times \boldsymbol{p_i} + \boldsymbol{\omega_p} \times (\boldsymbol{\omega_p} \times \boldsymbol{p_i}) - \boldsymbol{\omega}_{ul,i} \times (\boldsymbol{\omega}_{ul,i} \times \boldsymbol{A_iP_i})) \times \boldsymbol{A_iP_i} \tag{4.108}$$

Na osnovu identiteta za trostruki vektorski proizvod se može napisati da je:

$$(\boldsymbol{A_iP_i} \cdot \boldsymbol{\epsilon}_{ul,i})\,\boldsymbol{A_iP_i} - (\boldsymbol{A_iP_i} \cdot \boldsymbol{A_iP_i})\,\boldsymbol{\epsilon}_{ul,i} = (\boldsymbol{a_p} - \boldsymbol{a_{A_i}} + \boldsymbol{\epsilon_p} \times \boldsymbol{p_i} + \boldsymbol{\omega_p} \times (\boldsymbol{\omega_p} \times \boldsymbol{p_i}) - \boldsymbol{\omega}_{ul,i} \times (\boldsymbol{\omega}_{ul,i} \times \boldsymbol{A_iP_i})) \times \boldsymbol{A_iP_i} \tag{4.109}$$

S obzirom na ortogonalnost vektora $\boldsymbol{\epsilon}_{ul,i}$ i $\boldsymbol{A_iP_i}$ ($\boldsymbol{A_iP_i} \cdot \boldsymbol{\epsilon}_{ul,i} = 0$) sledi da je izraz za određivanje ugaonog ubrzanja gornjih poluga:

$$\boldsymbol{\epsilon}_{ul,i} = \frac{-1}{|\boldsymbol{A_iP_i}|^2}(\boldsymbol{a_p} - \boldsymbol{a_{A_i}} + \boldsymbol{\epsilon_p} \times \boldsymbol{p_i} + \boldsymbol{\omega_p} \times (\boldsymbol{\omega_p} \times \boldsymbol{p_i}) - \boldsymbol{\omega}_{ul,i} \times (\boldsymbol{\omega}_{ul,i} \times \boldsymbol{A_iP_i})) \times \boldsymbol{A_iP_i} \tag{4.110}$$

U prilogu F je data skripta `calculateVelocitiesAndAccelerations.jsl` u kojoj su implementirane sve prethodne jednačine na jednom primeru mehanizma.

4.7 Proračun elastičnih deformacija mehanizma

Na početku jednog ovakvog proračuna u okviru *strukturalne analize* neophodno je napraviti geometrijski model za proračun. Zbog toga što ovo nije statična konstrukcija već se radi o robotskom mehanizmu, geometrija je izmenljiva i proračun čvrstoće je potrebno uraditi za kritičan slučaj u okviru radnog prostora mehanizma.

4.7.1 Metod konačnih elemenata

Metod konačnih elemenata predstavlja jednu od metoda *numeričke analize* kojom se rešavaju problemi mehanike kontinuma [129]. Zasniva se na fizičkoj *diskretizaciji posmatranog domena* konačnim elementima, kako bi se od beskonačno mnogo stepeni slobode kretanja tačaka u posmatranom domenu i sistema parcijalnih diferencijalnih jednačina sistem sveo na proračunski model koji ima konačne dimenzije, fizičke karakteristike, konačan broj stepena slobode kretanja i jednostavnije jednačine koje moraju da se reše.

Tako se metod konačnih elemenata ogleda u svođenju sistema parcijalnih diferencijalnih jednačina u sistem aproksimacionih jednačina za svaki element čijim sastavljanjem se formira sistem koji pokriva ceo domen problema i njegovim rešavanjem se dobija celo polje pomeranja.

4.7.1.1 Hamiltonov princip

Hamiltonov princip je *varijacioni princip* koji za kretanje sistema glasi:

Hamiltonovo dejstvo, pri kretanju holonomnog sistema na koji deluju konzervativne sile, na stvarnoj trajektoriji ima stacionarnu vrednost u poređenju sa kretanjem po zaobilaznoj putanji.

Sistem je *holonoman* ako je izložen delovanju konačnih ili integrabilnih diferencijalnih veza koje se nazivaju još i holonomne veze. *Konzervativne sile* su potencijalne sile čija potencijalna

energija ne zavisi eksplicitno od vremena pa dolazi do održanja mehaničke energije. Rad ovih sila ne zavisi od pređenog puta ili oblika putanje već samo od početnog i krajnjeg položaja. Na osnovu ove definicije za Hamiltonovo dejstvo se može napisati:

$$\delta W = \delta \int_{t_1}^{t_2} L \mathrm{d}t = 0 \tag{4.111}$$

$$L = \Pi - T - W_f \tag{4.112}$$

gde su:

- L - Lagranžov funkcional,
- T - Kinetička energija,
- Π - Potencijalna energija,
- W_f - Rad spoljašnjih sila.

Kinetička energija se definiše na sledeći način:

$$T = \frac{1}{2} \int_V \rho \dot{\boldsymbol{U}}^T \dot{\boldsymbol{U}} \mathrm{d}V \tag{4.113}$$

gde je $\boldsymbol{U}$ vektor pomeranja, ρ gustina i V zapremina.

$$\dot{\boldsymbol{U}} = \frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}t}, \quad \boldsymbol{U} = \begin{bmatrix} u & v & w \end{bmatrix}^T \tag{4.114}$$

Potencijalna energija se definiše na sledeći način:

$$\Pi = \frac{1}{2} \int_V \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} \, \mathrm{d}V = \frac{1}{2} \int_V \boldsymbol{\varepsilon}^T \boldsymbol{c} \, \boldsymbol{\varepsilon} \, \mathrm{d}V \tag{4.115}$$

gde je $\boldsymbol{\sigma}$ tenzor napona, $\boldsymbol{\varepsilon}$ vektor deformacija i $\boldsymbol{c}$ matrica kojom se definišu elastične karakteristike materijala.

Prilikom dejstva spoljašnjih sila koje teže da deformišu telo njima se suprotstavljaju odgovarajuće *unutrašnje sile* koje predstavljaju rezultat uzajamnog dejstva između delića tela. Kada se neko opterećeno telo fiktivno preseče, tada unutrašnje sile postaju zamišljene spoljašnje sile koje predstavljaju uticaj uklonjenog dela tela na deo koji se posmatra. *Napon* predstavlja odnos unutrašnje sile i površine, to je pojam vezan za određenu tačku i ravan koja prolazi kroz tu tačku (ravan presecanja tela).

Da bi se odredilo stanje napona u posmatranoj tački neophodno je poznavati vektore napona za tri međusobno upravne ravni koje prolaze kroz tu tačku. U ravni preseka se definišu dva *napona smicanja* dok se u ravni normalnoj na ravan preseka definiše *normalni napon.*

Prethodno korišćena zavisnost data zasebno u okviru jednačine 4.116 predstavlja *konstitutivne jednačine* koje daju zavisnost između napona i deformacija u materijalu. Ova zavisnost se naziva još i *Generalisan Hukov zakon.*

$$\boldsymbol{\sigma} = \boldsymbol{c} \, \boldsymbol{\varepsilon} \tag{4.116}$$

Prethodna matrična jednačina u razvijenom obliku glasi:

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{yz} \\ \sigma_{xz} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ & & c_{33} & c_{34} & c_{35} & c_{36} \\ & & & c_{44} & c_{45} & c_{46} \\ & & & & c_{55} & c_{56} \\ \text{Sim.} & & & & & c_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} \tag{4.117}$$

gde su:

- σ_{xx}, σ_{yy}, σ_{zz} - Normalni naponi,
- σ_{yz}, σ_{xz}, σ_{xy} - Naponi smicanja,
- ε_{xx}, ε_{yy}, ε_{zz} - Aksijalne deformacije
- γ_{yz}, γ_{xz}, γ_{xy} - Deformacije klizanja.

Za napone smicanja važi da je:

$$\sigma_{yz} = \sigma_{zy}, \quad \sigma_{xz} = \sigma_{zx}, \quad \sigma_{xy} = \sigma_{yx} \tag{4.118}$$

Tri karakteristike izotropnih materijala čija međusobna zavisnost je data sledećom jednačinom su modul elastičnosti E, modul klizanja G i Poasonov koeficijent ν.

$$G = \frac{E}{2(1+\nu)} \tag{4.119}$$

Modul elastičnosti materijala je konstantna fizička karakteristika materijala koja se određuje eksperimentalnim putem. *Modul klizanja* je takođe konstantna fizička karakteristika materijala koja povezuje napon smicanja i odgovarajuću ugaonu deformaciju. *Poasonov koeficijent* je konstantna bezdimenziona fizička karakteristika materijala koja predstavlja odnos između poprečne i podužne deformacije.

Rad spoljašnjih sila je definisan na sledeći način:

$$W_f = \int_V \boldsymbol{U}^T f_b \, \mathrm{d}V + \int_{S_f} \boldsymbol{U}^T f_s \, \mathrm{d}S_f \tag{4.120}$$

gde su f_b i f_s zapreminske i površinske sile.

Zapreminske sile deluju na svaku gradivnu česticu tela i srazmerne su masi tog tela, dok površinske sile deluju na pojedinačne delove površine spoljašnjeg omotača tela. *Površinskim silama* je moguće predstaviti kontaktno dejstvo između tela i sredine koja ga okružuje.

U slučaju rešavanja problema iz statike se Hamiltonov princip svodi na *princip minimalne totalne potencijalne energije* što je moguće napisati u sledećem obliku:

$$\delta \Pi_t = 0 \tag{4.121}$$

Totalna potencijalna energija Π_t se određuje na osnovu razlike potencijalne energije i rada spoljašnjih sila.

$$\Pi_t = \Pi - W_f \tag{4.122}$$

4.7.1.2 Metoda pomeranja

U okviru *metode pomeranja* kao jedne od metoda za formulaciju metode konačnih elemenata su osnovne nepoznate pomeranja u čvorovima. U lokalnom koordinatnom sistemu vezanom za element se vrši polinomijalna interpolacija pomeranja korišćenjem pomeranja u čvorovima na sledeći način:

$$\overline{\boldsymbol{U}}(x,y,z) = \sum_{i=1}^{n_d} \boldsymbol{N_i}(x,y,z)\, \boldsymbol{d}_i = \boldsymbol{N}(x,y,z)\, \boldsymbol{d}_e \tag{4.123}$$

gde je $\overline{U}$ aproksimacija pomeranja, n_d broj čvorova koji formiraju element, d_i vektor pomeranja čvorova čija dimenzija zavisi od broja stepena slobode čvorova (n_f) i d_e vektor pomeranja čvorova elemenata.

Matrica funkcija oblika $\boldsymbol{N}$ se formira na sledeći način:

$$\boldsymbol{N}(x,y,z) = \begin{bmatrix} \boldsymbol{N}_1(x,y,z) & \boldsymbol{N}_2(x,y,z) & \cdots & \boldsymbol{N}_{n_d}(x,y,z) \end{bmatrix} \tag{4.124}$$

gde su $\boldsymbol{N}_i$ dijagonalne podmatrice funkcija oblika za komponente pomeranja.

Funkcijama oblika se pretpostavlja oblik rešenja tako da budu zadovoljene polazne parcijalne diferencijalne jednačine i granični uslovi. Korišćenjem funkcija oblika se mogu interpolirati vrednosti pomeranja po elementu na osnovu vrednosti u čvorovima.

Deformacije predstavljaju promenu pomeranja po jedinici dužine i na osnovu toga se mogu izračunati na sledeći način:

$$\begin{aligned} \varepsilon_{xx} &= \frac{\partial u}{\partial x}, \quad \varepsilon_{yy} = \frac{\partial v}{\partial y}, \quad \varepsilon_{zz} = \frac{\partial w}{\partial z} \\ \gamma_{xy} &= 2\varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \gamma_{xz} &= 2\varepsilon_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ \gamma_{yz} &= 2\varepsilon_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{aligned} \tag{4.125}$$

Iz prethodnih jednačina za deformacije je moguće definisati $\boldsymbol{L}$ kao *operator diferenciranja* pomeranja na sledeći način:

$$\boldsymbol{\varepsilon} = \boldsymbol{L}\boldsymbol{U} = \begin{bmatrix} \partial/\partial x & 0 & 0 \\ 0 & \partial/\partial y & 0 \\ 0 & 0 & \partial/\partial z \\ 0 & \partial/\partial z & \partial/\partial y \\ \partial/\partial z & 0 & \partial/\partial x \\ \partial/\partial y & \partial/\partial x & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.126}$$

Na osnovu jednačina 4.123 i 4.126 sledi da je:

$$\boldsymbol{\varepsilon} = \boldsymbol{L}\, \boldsymbol{N}\, \boldsymbol{d}_e = \boldsymbol{B}\, \boldsymbol{d_e} \tag{4.127}$$

U prethodnoj jednačini je uvedena *matrica deformacija* $\boldsymbol{B}$ tako da je:

$$\boldsymbol{B} = \boldsymbol{L}\boldsymbol{N} \tag{4.128}$$

Sada se korišćenjem prethodnih relacija može potencijalna energija elementa napisati u sledećem obliku:

$$\Pi = \frac{1}{2}\int_{V_e} \boldsymbol{\varepsilon}^T\, \boldsymbol{c}\, \boldsymbol{\varepsilon}\, \mathrm{d}V_e = \frac{1}{2}\int_{V_e} \boldsymbol{d}_e^T\, \boldsymbol{B}^T\, \boldsymbol{c}\, \boldsymbol{B}\, \boldsymbol{d}_e\, \mathrm{d}V_e = \frac{1}{2}\boldsymbol{d}_e^T\, \boldsymbol{k}_e\, \boldsymbol{d}_e \tag{4.129}$$

gde je $\boldsymbol{k_e}$ matrica krutosti elementa u lokalnom koordinatnom sistemu elementa i definisana je jednačinom 4.130.

$$\boldsymbol{k}_e = \int_{V_e} \boldsymbol{B}^T\, \boldsymbol{c}\, \boldsymbol{B}\, \mathrm{d}V_e \tag{4.130}$$

Kinetička energija se može napisati u sledećem obliku:

$$T = \frac{1}{2}\int_{V_e} \rho\, \dot{\boldsymbol{U}}^T\, \dot{\boldsymbol{U}} \mathrm{d}V_e = \frac{1}{2}\int_{V_e} \rho\, \dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{N}^T\, \boldsymbol{N}\, \dot{\boldsymbol{d}}_e \mathrm{d}V_e = \frac{1}{2}\dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \dot{\boldsymbol{d}}_e \tag{4.131}$$

gde je $\boldsymbol{m_e}$ masena matrica elementa u lokalnom koordinatnom sistemu elementa i definisana je jednačinom 4.132.

$$\boldsymbol{m}_e = \int_{V_e} \boldsymbol{N}^T\, \boldsymbol{N}\, \mathrm{d}V_e \tag{4.132}$$

I još se rad spoljašnjih sila može napisati u sledećem obliku:

$$W_f = \int_{V_e} \boldsymbol{U}^T\, f_b\, \mathrm{d}V_e + \int_{S_{f,e}} \boldsymbol{U}^T\, f_s\, \mathrm{d}S_{f,e} = \int_{V_e} \boldsymbol{d}_e^T\, \boldsymbol{N}^T\, f_b\, \mathrm{d}V_e + \int_{S_{f,e}} \boldsymbol{d}_e^T\, \boldsymbol{N}^T\, f_s\, \mathrm{d}S_{f,e} \tag{4.133}$$

Definišu se sile u čvorovima koje su ekvivalentne zapreminskim i površinskim silama u sledećem obliku:

$$\boldsymbol{F_{b,e}} = \int_{V_e} \boldsymbol{N}^T\, \boldsymbol{f_{b,e}}\, \mathrm{d}V_e, \quad \boldsymbol{F_{S,e}} = \int_{S_{f,e}} \boldsymbol{N}^T\, \boldsymbol{f_s}\, \mathrm{d}S_{f,e} \tag{4.134}$$

Izraz za rad spoljašnjih sila u zavisnosti od totalnog vektor čvorne sile $\boldsymbol{f_e}$ je dat u sledećoj jednačini.

$$W_f = \boldsymbol{d}_e^T\, \boldsymbol{F_{b,e}} + \boldsymbol{d}_e^T\, \boldsymbol{F_{S,e}} = \boldsymbol{d}_e^T\, \boldsymbol{f_e} \tag{4.135}$$

Nakon zamene jednačina 4.129, 4.131 i 4.135 u jednačinu 4.112, za Lagranžev funkcional se tada dobija da je:

$$L = \frac{1}{2}\boldsymbol{d}_e^T\, \boldsymbol{k}_e\, \boldsymbol{d}_e - \frac{1}{2}\dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \dot{\boldsymbol{d}}_e - \boldsymbol{d}_e^T\, \boldsymbol{f}_e \tag{4.136}$$

Primenom Hamiltonovg principa za prethodnu jednačinu se dobija:

$$\begin{gathered}\delta \int_{t_1}^{t_2} \left(\frac{1}{2}\boldsymbol{d}_e^T\, \boldsymbol{k}_e\, \boldsymbol{d}_e - \frac{1}{2}\dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \dot{\boldsymbol{d}}_e - \boldsymbol{d}_e^T\, \boldsymbol{f}_e\right) \mathrm{d}t = \\ \int_{t_1}^{t_2} \left(\delta\, \boldsymbol{d}_e^T\, \boldsymbol{k}_e\, \boldsymbol{d}_e - \delta\, \dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \dot{\boldsymbol{d}}_e - \delta\, \boldsymbol{d}_e^T\, \boldsymbol{f}_e\right) \mathrm{d}t = 0\end{gathered} \tag{4.137}$$

Na osnovu pravila parcijalne integracije se može napisati sledeće:

$$\int_{t_1}^{t_2} \delta\, \dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \dot{\boldsymbol{d}}_e \mathrm{d}t = \delta\, \boldsymbol{d}_e^T\, \boldsymbol{m}_e\, \dot{\boldsymbol{d}}_e|_{t_1}^{t_2} - \int_{t_1}^{t_2} \delta\, \dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \ddot{\boldsymbol{d}}_e \mathrm{d}t = -\int_{t_1}^{t_2} \delta\, \dot{\boldsymbol{d}}_e^{\,T}\, \boldsymbol{m}_e\, \ddot{\boldsymbol{d}}_e \mathrm{d}t \tag{4.138}$$

U prethodnom izrazu je, na osnovu toga što u početnom i krajnjem stanju varijacija koordinata ne postoji, iskorišćeno da je:

$$\delta\, \boldsymbol{d_e}|_{t_1} = \delta\, \boldsymbol{d_e}|_{t_2} = 0 \tag{4.139}$$

Na osnovu ovoga jednačina 4.137 postaje:

$$\delta \left(\boldsymbol{m_e}\, \ddot{\boldsymbol{d}}_{\boldsymbol{e}} + \boldsymbol{k_e}\, \boldsymbol{d_e} - \boldsymbol{f_e} \right) = 0 \tag{4.140}$$

Iz čega sledi da je:

$$\boldsymbol{m_e}\, \ddot{\boldsymbol{d}}_{\boldsymbol{e}} + \boldsymbol{k_e}\, \boldsymbol{d_e} = \boldsymbol{f_e} \tag{4.141}$$

Jednačina 4.141 predstavlja *jednačinu konačnih elemenata za jedan element u lokalnom koordinatnom sistemu elementa.*

Kada se odredi matrica transformacije $\boldsymbol{T}$ iz lokalnog u globalni koordinatni sistem jednostavno se dobija sledeća jednačina koja predstavlja *jednačinu konačnih elemenata za jedan element u globalnom koordinatnom sistemu*:

$$\boldsymbol{M_e}\, \ddot{\boldsymbol{D}}_{\boldsymbol{e}} + \boldsymbol{K_e}\, \boldsymbol{D_e} = \boldsymbol{F_e} \tag{4.142}$$

gde su:

$$\begin{aligned} \boldsymbol{D_e} &= \boldsymbol{T}^T\, \boldsymbol{d_e}, \quad \boldsymbol{F_e} = \boldsymbol{T}^T\, \boldsymbol{f_e} \\ \boldsymbol{K_e} &= \boldsymbol{T}^T\, \boldsymbol{k_e}\, \boldsymbol{T}, \quad \boldsymbol{M_e} = \boldsymbol{T}^T\, \boldsymbol{m_e}\, \boldsymbol{T} \end{aligned} \tag{4.143}$$

Nakon transformacije koordinata i spajanja elemenata se može napisati *jednačina konačnih elemenata za sve elemente u globalnom koordinatnom sistemu* kao:

$$\boldsymbol{M}\, \ddot{\boldsymbol{D}} + \boldsymbol{K}\, \boldsymbol{D} = \boldsymbol{F} \tag{4.144}$$

gde su $\boldsymbol{M}$ i $\boldsymbol{K}$ globalna masena i globalna matrica krutosti, $\boldsymbol{D}$ je vektor pomeranja u svim čvorovima i $\boldsymbol{F}$ je vektor svih čvornih opterećenja. Prilikom formiranja jednačina za sve elemente se sabira uticaj svih elementa povezanih u odgovarajućem čvoru. *U slučaju statike* jednačina 4.144 postaje:

$$\boldsymbol{K}\, \boldsymbol{D} = \boldsymbol{F} \tag{4.145}$$

Polje pomeranja $\boldsymbol{U}$ koje je izraženo pomeranjima u čvorovima, korišćenjem funkcija oblika se izražava po elementima. Dobijanje jednačina konačnih elemenata se nakon izbora odgovarajućeg tipa elementa svodi na određivanje matrice $\boldsymbol{N}$ i potom $\boldsymbol{B}$ za taj tip elementa.

4.7.1.3 Spoljašnja opterećenja

Spoljašnja opterećenja se mogu uvesti samo u čvorovima elemenata i to u okviru vektora $\boldsymbol{F}$. Za *element tipa greda* koji je konstruisan između *čvora 1* i *čvora 2* ovaj vektor ima sledeći oblik:

$$\boldsymbol{F_e} = \begin{bmatrix} F_{x1} & F_{y1} & F_{z1} & M_{x1} & M_{y1} & M_{z1} & F_{x2} & F_{y2} & F_{z2} & M_{x2} & M_{y2} & M_{z2} \end{bmatrix}^T \tag{4.146}$$

Sve komponente ovog vektora koje nemaju definisano spoljašnje opterećenje treba da imaju nultu vrednost prilikom rešavanja sistema jednačina konačnih elemenata.

4.7.1.4 Konturni uslovi

Globalna matrica krutosti $\boldsymbol{K}$ je obično singularna jer nisu zadati konturni uslovi kojima se uvode ograničenja kretanja tela. Da bi se sprečilo slobodno kretanje tela neophodno je uvesti ograničenja u vidu sprečavanja odgovarajućih komponenata pomeranja u čvorovima elemenata. Ograničenja se uvode u zavisnosti od samog postavljenog problema tako da što je moguće vernije opišu postavljeni problem.

Za *element tipa greda* koji je konstruisan između *čvora 1* i *čvora 2* vektor pomeranja $\boldsymbol{D_e}$ ima sledeći oblik:

$$\boldsymbol{D_e} = \begin{bmatrix} u_1 & v_1 & w_1 & \theta_{x1} & \theta_{y1} & \theta_{z1} & u_2 & v_2 & w_2 & \theta_{x2} & \theta_{y2} & \theta_{z2} \end{bmatrix}^T \tag{4.147}$$

Ograničenja se mogu uvesti uklanjanjem redova i kolona iz jednačine 4.145 koje odgovaraju komponentama pomeranja koja se sprečavaju, čime se dobija redukovana matrica krutosti $\boldsymbol{K_r}$, kao i redukovana pomeranja $\boldsymbol{D_r}$ i čvorna opterećenja $\boldsymbol{F_r}$.

$$\boldsymbol{K_r}\,\boldsymbol{D_r} = \boldsymbol{F_r} \tag{4.148}$$

Nakon toga, ako je problem pravilno definisan, matrica krutosti postaje pozitivno definitna i simetrična, što ujedno znači i da je rešavanjem sada redukovane jednačine 4.148 moguće odrediti pomeranja $\boldsymbol{D_r}$ i potom konstruisati sada potpuno definisan vektor pomeranja $\boldsymbol{D}$ koji je moguće vratiti u jednačinu 4.145 i odrediti reakcije veza iz $\boldsymbol{F}$.

4.7.1.5 Metod Lagranževih množioca

U okviru *Metode Lagranževih množioca* se uvode dodatne promenljive $\boldsymbol{\lambda}$ koje se nazivaju *Lagranževi množioci.*

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_m \end{bmatrix}^T \tag{4.149}$$

Ovaj metod je moguće iskoristiti u slučajevima kada je potrebno definisati ograničenja kao što su linearno zavisna pomeranja određenih čvorova[130]. Na ovaj način je moguće zadati da određeni čvorovi imaju ista translatorna pomeranja što bi u tom slučaju značilo da se u tom čvoru nalazi sferni zglob i da se prenose sile, ali se ne prenose momenti. Ove zavisnosti se definišu u okviru matrice $\boldsymbol{C}$ kojom se proširuje matrica krutosti. Sa vektorom $\boldsymbol{Q}$ se proširuje vektor čvornih opterećenja dok se nepoznate $\boldsymbol{\lambda}$ dodaju vektoru pomeranja $\boldsymbol{D}$.

Sistem definisan u 4.145 se sada proširuje i pritom mora da bude zadovoljeno da je:

$$\boldsymbol{\lambda}^T\,[\boldsymbol{C}\boldsymbol{D} - \boldsymbol{Q}] = 0 \tag{4.150}$$

Kada se proširi Lagranžov funkcional iz 4.112 i za slučaj statike on postaje:

$$\Pi_p = \frac{1}{2}\boldsymbol{D}^T\,\boldsymbol{K}\,\boldsymbol{D} - \boldsymbol{D}^T\,\boldsymbol{F} + \boldsymbol{\lambda}^T\,[\boldsymbol{C}\boldsymbol{D} - \boldsymbol{Q}] \tag{4.151}$$

Primenom Hamiltonovg principa za prethodnu jednačinu se dobija:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{C}^T \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{Q} \end{bmatrix} \tag{4.152}$$

Ovo je novi sistem jednačina za koji se definišu spoljašnja opterećenja i konturni uslovi, zatim se sa novom redukovanom matricom krutosti određuju nepoznate promenljive. Redukovana

matrica krutosti nije više pozitivno definitna zbog nula na dijagonali, povećan je broj nepoznatih ali sva uvedena ograničenja moraju uvek da budu zadovoljena.

Ukoliko je potrebno formirati sferni zglob između elementa 1 i elementa 2 tipa greda prvo se formira matrica $\boldsymbol{C}$. Element 1 je definisan čvorom 1 i čvorom 2 dok je element 2 definisan čvorom 3 i čvorom . Za *povezivanje čvorova 2 i 3 sfernim zglobom matrica* $\boldsymbol{C}$ treba da ima sledeće vrednosti:

$$\begin{aligned} \boldsymbol{C}(1:3,[1:6,10:12,15:24]) &= \mathbf{0_{3,18}} \\ \boldsymbol{C}(1:3,[7:9,13:15]) &= \begin{bmatrix} \boldsymbol{I_3} & -\boldsymbol{I_3} \end{bmatrix} \end{aligned} \tag{4.153}$$

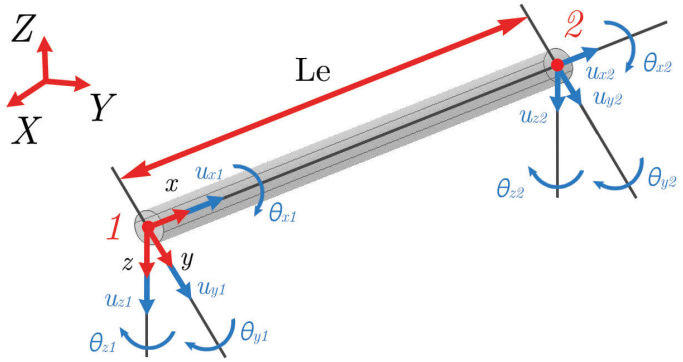
Odnosno u sistem jednačina su dodate jednačine:

$$\begin{aligned} u_{x2} &= u_{x3} \\ u_{y2} &= u_{y3} \\ u_{z2} &= u_{z3} \end{aligned} \tag{4.154}$$

4.7.2 Element Ojler-Bernulijeva greda

4.7.2.1 Osobine elementa

Greda je u stvarnosti trodimenzionalno telo, ali se aproksimira i analizira *linijskim nosećim elementom*. To je telo čija je jedna dimenzija (dužina) znatno veća od druge dve (širine i visine). Ovo telo nastaje tako što se proizvoljna figura kreće duž neke prave ili krive linije tako da joj težište sve vreme leži na toj liniji koja se iz tog razloga naziva težišna linija. Razlika između grede i, takođe linijskog nosećeg elementa, štapa je u tome što greda može da bude opterećena još i upravno na svoju podužnu osu.



Slika 4.26 - Element tipa Ojler-Bernulijeva greda

Na osnovu *Ojler-Bernulijeve teorije za tanke grede* se pretpostavlja da ravni poprečni preseci, koji su normalni na centralnu podužnu osu grede u nedeformisanom slučaju, ostaju ravni i normalni u odnosu na deformisanu centralnu osu nakon delovanja spoljašnjeg opterećenja. Na osnovu toga se zaključuje da su:

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = 0 \tag{4.155}$$

$$\gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = 0 \tag{4.156}$$

Iz čega potom sledi da se *naponi smicanja ne uzimaju u obzir* pa se preporučuje da se ovaj tip elementa koristi kada je *dužina elementa barem 10 puta veća od ostale dve dimenzije*.

Za definisanje grede je potrebno definisati karakteristike njenog poprečnog preseka, koji u opštem slučaju može biti promenljiv duž grede. Karakteristike poprečnog preseka od značaja za dalju analizu u ovom radu su:

- A - Površina poprečnog preseka,
- I_y - Aksijalni moment inercije za osu y,
- I_z - Aksijalni moment inercije za osu z,
- J - Polarni moment inercije.

U odnosu na površinu poprečnog preseka koja je fizička veličina dobijena presecanjem grede i merenjem površine dobijene figure, momenti inercije predstavljaju zamišljene geometrijske karakteristike [[131]].

4.7.2.2 Diferencijalne jednačine pomeranja

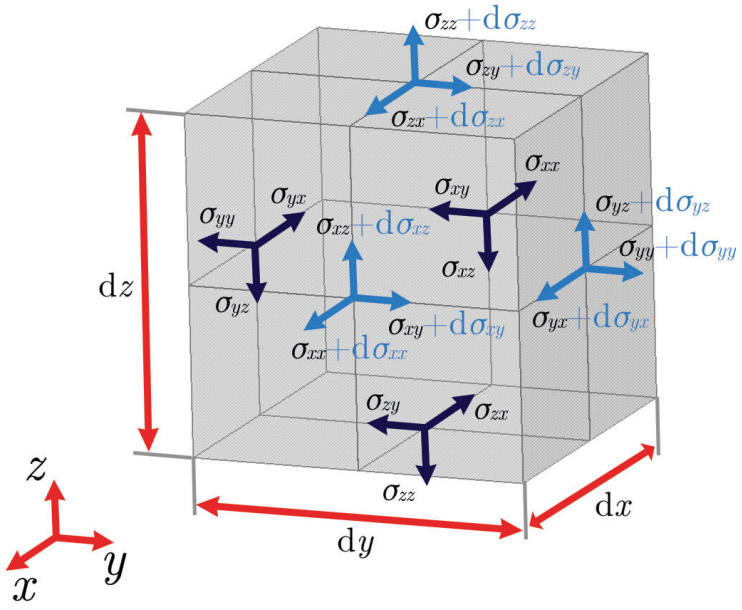
4.7.2.3 Jednačine ravnoteže

Posmatranjem ravnoteže, u opštem slučaju, za *beskonačno mali deo trodimenzionalnog tela* (slike 4.26 i 4.27) je moguće napisati jednačine ravnoteže u sledećem obliku:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} + f_x = \rho\,\ddot{u} \tag{4.157}$$

$$\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z} + f_y = \rho\,\ddot{v} \tag{4.158}$$

$$\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + f_z = \rho\,\ddot{w} \tag{4.159}$$



Slika 4.27 - Prikaz komponenata napona na elementarnom delu trodimenzionalnog tela

Jednačine 4.157, 4.158 i 4.159 se u matričnom obliku mogu napisati na sledeći način:

$$\boldsymbol{L}^T \boldsymbol{c}\, \boldsymbol{L}\, \boldsymbol{U} + \boldsymbol{f_b} = \rho\, \ddot{\boldsymbol{U}} \tag{4.160}$$

4.7.2.4 Aksijalno naprezanje

Za slučaj *aksijalnog naprezanja* grede od svih komponenata napona ostaje samo σ_{xx} komponenta normalnog napona duž x ose pa se jednačina 4.157 svodi na:

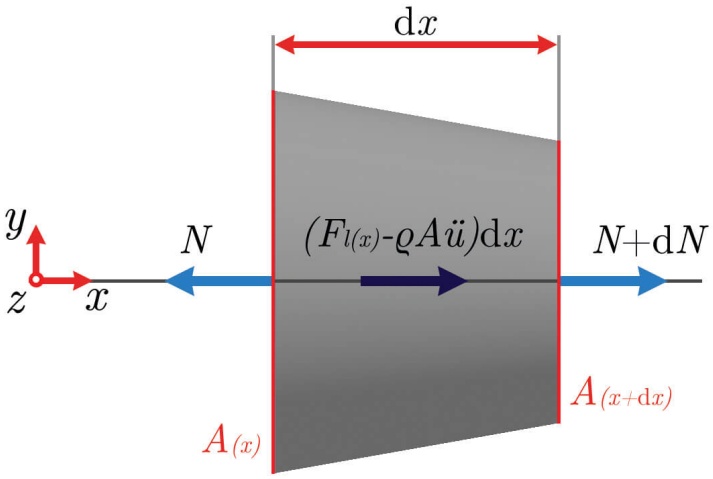
$$\frac{\partial \sigma_{xx}}{\partial x} + f_x = \rho\, \ddot{u} \tag{4.161}$$

Na osnovu veze napona i deformacija i veze deformacija i pomeranja se dobija da je:

$$E\, \frac{\partial \varepsilon_{xx}}{\partial x} + f_x = E\, \frac{\partial^2 u}{\partial x^2} + f_x = \rho\, \ddot{u} \tag{4.162}$$

Sa slike 4.28 se može postaviti jednačina ravnoteže sila duž ose x za elementarni deo grede i dobija se sledeća jednačina:

$$-N + F_l(x)\mathrm{d}x + N + \mathrm{d}N - \rho\, A\, \ddot{u}\mathrm{d}x = 0 \tag{4.163}$$



Slika 4.28 - Aksijalno naprezanje elementarnog dela grede

Na osnovu izraza za aksijalnu silu N i jednačine ravnoteže 4.162 se dobija da je:

$$N = \sigma_{xx}\, A = E\, A\, \varepsilon_{xx} = E\, A\, \frac{\partial u}{\partial x} \tag{4.164}$$

$$\frac{\mathrm{d}N}{\mathrm{d}x} + F_l(x) = \frac{\mathrm{d}}{\mathrm{d}x}\left(E\, A\, \frac{\partial u}{\partial x}\right) + F_l(x) = \rho\, \ddot{u} \tag{4.165}$$

Konačna parcijalna diferencijalna jednačina za slučaj aksijalnog naprezanja i bez uticaja inercijalne sile glasi:

$$E\, A\, \frac{\partial^2 u}{\partial x^2} + F_l(x) = 0 \tag{4.166}$$

4.7.2.5 Uvijanje

Za slučaj *uvijanja* grede se na osnovu veze napona i deformacija i veze deformacija i pomeranja dobija da je:

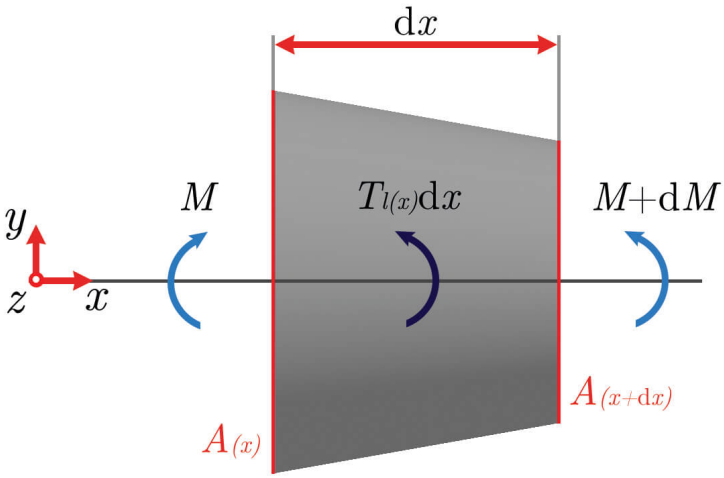
$$\sigma_{yz} = G\,\gamma_{yz} = G\,\sqrt{y^2+z^2}\,\frac{\partial\theta_x}{\partial x} = G\,r\,\frac{\partial\theta_x}{\partial x} \tag{4.167}$$

Veza između ugla klizanja i ugla uvijanja je:

$$\gamma_{yz} = r\,\frac{\partial\theta_x}{\partial x} \tag{4.168}$$

Sa slike 4.29 se može postaviti jednačina ravnoteže momenata oko ose x za elementarni deo grede i dobija se sledeća jednačina:

$$-M_x + T_l(x)\mathrm{d}x + M_x + \mathrm{d}M_x = 0 \tag{4.169}$$



Slika 4.29 - Uvijanje elementarnog dela grede

Karakteristika poprečnog preseka koja je važna za analizu uvijanja je polarni moment inercije J, koji se dobija na osnovu:

$$J = \int_A r^2\,\mathrm{d}A \tag{4.170}$$

Na osnovu izraza za moment oko ose x i jednačine ravnoteže 4.169 se dobija da je:

$$M_x = \int_A r\,\sigma_{yz}\,\mathrm{d}A = G\,\frac{\partial\theta_x}{\partial x}\int_A r^2\,\mathrm{d}A = G\,J\,\frac{\partial\theta_x}{\partial x} \tag{4.171}$$

$$\frac{\mathrm{d}M_x}{\mathrm{d}x} + T_l(x) = \frac{\mathrm{d}}{\mathrm{d}x}\left(G\,J\,\frac{\partial\theta_x}{\partial x}\right) + T_l(x) = 0 \tag{4.172}$$

Konačna parcijalna diferencijalna jednačina za slučaj uvijanja glasi:

$$G\,J\,\frac{\partial^2\theta_x}{\partial x^2} + T_l(x) = 0 \tag{4.173}$$

4.7.2.6 Savijanje u xy ravni

Za slučaj *savijanja* grede u xy ravni se na osnovu pretpostavke 4.155 dobija da je:

$$u = -y\,\frac{\partial v}{\partial x} \tag{4.174}$$

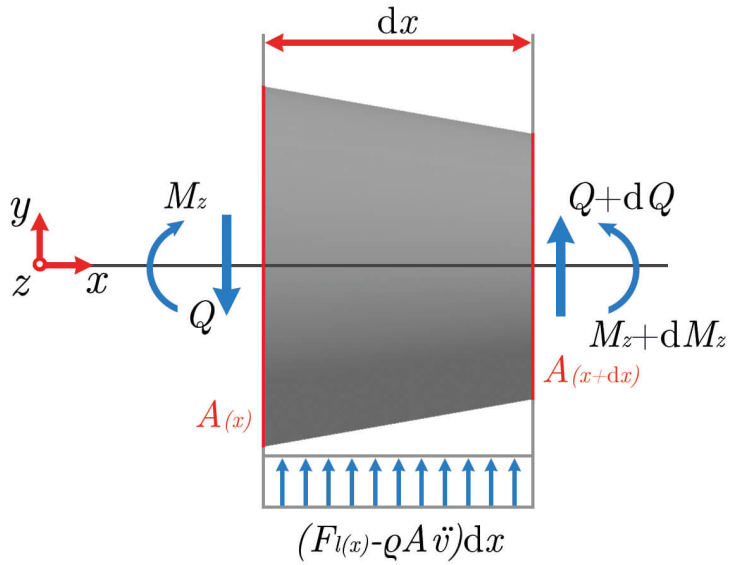
Veza između ugla rotacije poprečnog preseka θ_z i pomeranja v je:

$$\theta_z = \frac{\partial v}{\partial x} \tag{4.175}$$

Sa slike 4.30 se mogu postaviti jednačine ravnoteže sila duž ose y i momenata oko ose z za elementarni deo grede i dobijaju se sledeće jednačine:

$$\left(F_l(x) - \rho\,A\,\ddot{v}\right)\mathrm{d}x + Q - Q - \mathrm{d}Q = 0 \tag{4.176}$$

$$-M_z + M_z + \mathrm{d}M_z + Q\mathrm{d}x - \left(F_l(x) - \rho\,A\,\ddot{v}\right)\mathrm{d}x\frac{\mathrm{d}x}{2} = 0 \tag{4.177}$$



Slika 4.30 - Savijanje elementarnog dela grede u xy ravni

Iz ravnoteže sila se dobija da je:

$$\frac{\mathrm{d}Q}{\mathrm{d}x} + F_l(x) = \rho\,A\,\ddot{v} \tag{4.178}$$

Iz ravnoteže momenata se dobija da je:

$$\frac{\mathrm{d}M_z}{\mathrm{d}x} = -Q \tag{4.179}$$

Na osnovu prethodne dve jednačine sledi da je:

$$\frac{\mathrm{d}^2M_z}{\mathrm{d}x^2} + \rho\,A\,\ddot{v} = F_l(x) \tag{4.180}$$

Karakteristika poprečnog preseka koja je važna za analizu savijanja u ravni xy je aksijalni moment inercije I_z, koji se dobija na osnovu:

$$I_z = \int_A y^2 \, \mathrm{d}A \tag{4.181}$$

Na osnovu veze napona i deformacija i izraza za moment oko ose z se dobija da je:

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} = -y \frac{\partial \theta_z}{\partial x} = -y \frac{\partial^2 v}{\partial x^2} \tag{4.182}$$

$$M_z = -\int_A y \, \sigma_{xx} \mathrm{d}A = -E \int_A y \, \varepsilon_{xx} \, \mathrm{d}A = E \frac{\partial^2 v}{\partial x^2} \int_A y^2 \, \mathrm{d}A = E \, I_z \frac{\partial^2 v}{\partial x^2} \tag{4.183}$$

Konačna parcijalna diferencijalna jednačina za slučaj savijanja u xy ravni glasi:

$$E \, I_z \frac{\partial^4 v}{\partial x^4} + \rho \, A \, \ddot{v} = F_l(x) \tag{4.184}$$

U slučaju statike se ova jednačina svodi na:

$$E \, I_z \frac{\partial^4 v}{\partial x^4} = F_l(x) \tag{4.185}$$

4.7.2.7 Savijanje u xz ravni

Za slučaj *savijanja* grede u xz ravni se na osnovu pretpostavke 4.156 dobija da je:

$$u = -z \frac{\partial w}{\partial x} \tag{4.186}$$

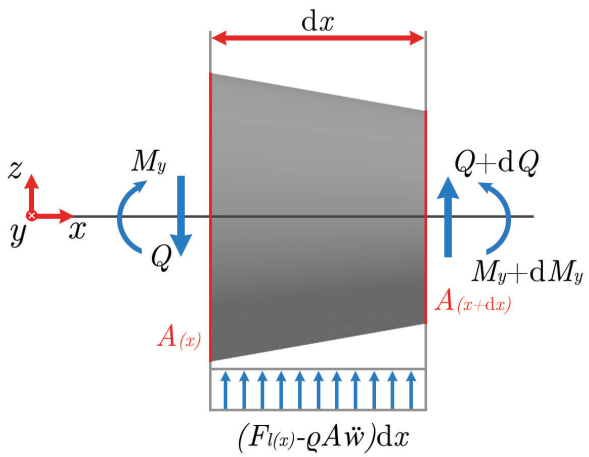
Veza između ugla rotacije poprečnog preseka θ_y i pomeranja w je:

$$\theta_y = -\frac{\partial w}{\partial x} \tag{4.187}$$

Sa slike 4.31 se mogu postaviti jednačine ravnoteže sila duž ose y i momenata oko ose z za elementarni deo grede i dobijaju se sledeće jednačine:

$$\left(F_l(x) - \rho \, A \, \ddot{w}\right) \mathrm{d}x + Q - Q - \mathrm{d}Q = 0 \tag{4.188}$$

$$-M_y + M_y + \mathrm{d}M_y + Q\mathrm{d}x - \left(F_l(x) - \rho \, A \, \ddot{w}\right) \mathrm{d}x \frac{\mathrm{d}x}{2} = 0 \tag{4.189}$$



Slika 4.31 - Savijanje elementarnog dela grede u xz ravni

Iz ravnoteže sila se dobija da je:

$$\frac{\mathrm{d}Q}{\mathrm{d}x} + F_l(x) = \rho\, A\, \ddot{w} \tag{4.190}$$

Iz ravnoteže momenata se dobija da je:

$$\frac{\mathrm{d}M_y}{\mathrm{d}x} = -Q \tag{4.191}$$

Na osnovu prethodne dve jednačine sledi da je:

$$\frac{\mathrm{d}^2 M_y}{\mathrm{d}x^2} + \rho\, A\, \ddot{w} = F_l(x) \tag{4.192}$$

Karakteristika poprečnog preseka koja je važna za analizu savijanja u ravni xz je aksijalni moment inercije I_y, koji se dobija na osnovu:

$$I_y = \int_A z^2\, \mathrm{d}A \tag{4.193}$$

Na osnovu veze napona i deformacija i izraza za moment oko ose y se dobija da je:

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} = z\,\frac{\partial \theta_y}{\partial x} = -z\frac{\partial^2 w}{\partial x^2} \tag{4.194}$$

$$M_y = \int_A z\,\sigma_{xx}\mathrm{d}A = E\int_A z\,\varepsilon_{xx}\,\mathrm{d}A = -E\frac{\partial^2 w}{\partial x^2}\int_A z^2\,\mathrm{d}A = -E\,I_y\,\frac{\partial^2 w}{\partial x^2} \tag{4.195}$$

Konačna parcijalna diferencijalna jednačina za slučaj savijanja u xz ravni glasi:

$$E\,I_y\,\frac{\partial^4 w}{\partial x^4} - \rho\, A\, \ddot{w} = -F_l(x) \tag{4.196}$$

U slučaju statike se ova jednačina svodi na:

$$E\,I_y\,\frac{\partial^4 w}{\partial x^4} = -F_l(x) \tag{4.197}$$

4.7.2.8 Matrica krutosti

Kada su poznate diferencijalne jednačine 4.166, 4.173, 4.185 i 4.197 ponašanja grede prilikom aksijalnog naprezanja, uvijanja, savijanja u xy i savijanja u xz ravni moguće je definisati funkcije oblika i potom matricu krutosti elementa.

4.7.2.8.1 Aksijalno naprezanje

U slučaju *aksijalnog naprezanja* grede oba čvora elementa imaju po jedan stepen slobode i to aksijalna pomeranja u_1 i u_2.

$$\overline{u}(x) = \boldsymbol{N}(x)\boldsymbol{d_e} = \boldsymbol{N}(x)\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.198}$$

Aproksimacija pomeranja u se prvo definiše preko vektora baznih polinoma $\boldsymbol{p}$ i vektora nepoznatih $\boldsymbol{\alpha}$ koji su u ovom slučaju dvodimenzionalni.

$$\overline{u}(x) = \alpha_0 + \alpha_1 x = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \boldsymbol{p}^T \boldsymbol{\alpha} \tag{4.199}$$

Matricom $\boldsymbol{P}$ se određuje veza između nepoznatih koeficijenata $\boldsymbol{\alpha}$ i pomeranja elemenata, tako da je:

$$\boldsymbol{d_e} = \boldsymbol{P}\alpha \tag{4.200}$$

Elemente matrice $\boldsymbol{P}$ je moguće odrediti na osnovu vrednosti pomeranja u čvorovima, na osnovu čega je:

$$\begin{aligned} u_1 &= \overline{u}(0) = \alpha_0 \\ u_2 &= \overline{u}(L_e) = \alpha_0 + \alpha_1 L_e \end{aligned} \tag{4.201}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \boldsymbol{P}\,\boldsymbol{\alpha} = \begin{bmatrix} 1 & 0 \\ 1 & L_e \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \tag{4.202}$$

Na osnovu jednačina 4.198, 4.199 i 4.200 je aproksimaciju pomeranja moguće izraziti kao:

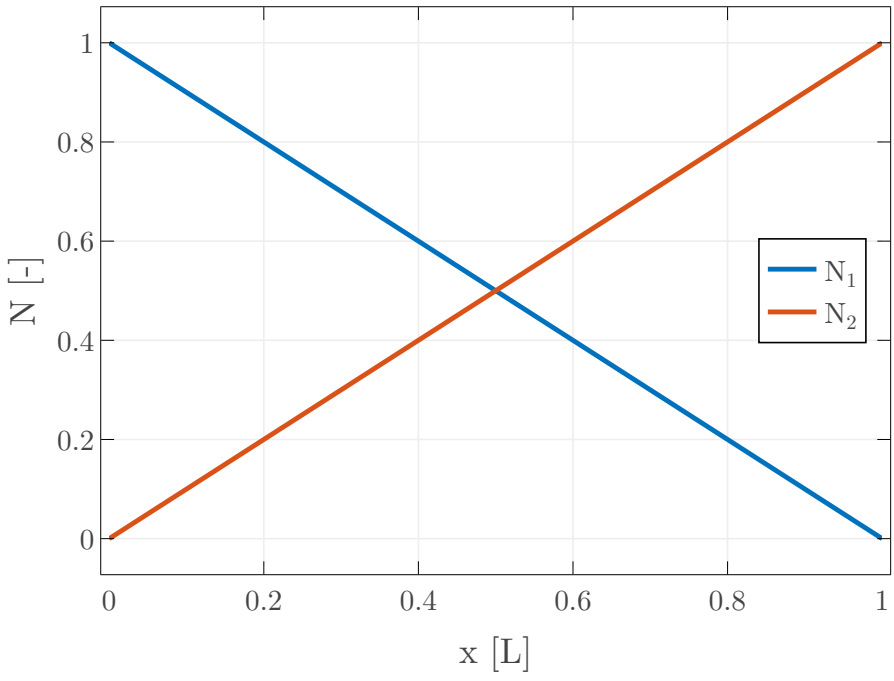
$$\overline{u}(x) = \boldsymbol{p}^T \boldsymbol{P}^{-1} \boldsymbol{d_e} = \boldsymbol{N}(x)\boldsymbol{d_e} \tag{4.203}$$

Iz prethodne jednačine se vidi da je potrebno odrediti inverz matrice $\boldsymbol{P}$ kako bi se odredile funkcije oblika.

$$\boldsymbol{P}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L_e} & \frac{1}{L_e} \end{bmatrix} \tag{4.204}$$

$$\boldsymbol{N}(x) = \boldsymbol{p}^T \boldsymbol{P}^{-1} = \begin{bmatrix} N_1 & N_2 \end{bmatrix} = \begin{bmatrix} 1 - \frac{x}{L_e} & \frac{x}{L_e} \end{bmatrix} \tag{4.205}$$

U okviru jednačine 4.205 su date funkcije oblika N_1 i N_2, dijagram ovih funkcija je dat na slici 4.32. Ovaj dijagram se dobija izvršavanjem skripte koja je data u okviru priloga G.



Slika 4.32 - Funkcije oblika za zatezanje

Na osnovu dobijenih funkcija oblika se vidi da se pomeranje u po elementu menja linearno, pa se u ovom slučaju element naziva još i *linearni element*.

$$u_x = u_1 + \frac{u_2 - u_1}{L_e} x \tag{4.206}$$

Sledeći korak je izraziti matricu deformacija na osnovu veze između deformacija i pomeranja .

$$\varepsilon_{xx} = \frac{\partial}{\partial x} (\boldsymbol{N}\, \boldsymbol{de}) = \boldsymbol{B}\, \boldsymbol{de} \tag{4.207}$$

$$\boldsymbol{B} = \frac{\partial \boldsymbol{N}}{\partial x} = \begin{bmatrix} -\frac{1}{L_e} & \frac{1}{L_e} \end{bmatrix} \tag{4.208}$$

Kada su poznate matrica deformacija i elastične karakteristike materijala za ovaj slučaj opterećenja, može se odrediti matrica krutosti elementa.

$$\boldsymbol{k_e} = \int_{V_e} \boldsymbol{B}^T\, \boldsymbol{c}\, \boldsymbol{B}\, \mathrm{d}V_e = \int_A \mathrm{d}A \int_0^{L_e} \begin{bmatrix} -\frac{1}{L_e} \\ \frac{1}{L_e} \end{bmatrix} E \begin{bmatrix} -\frac{1}{L_e} & \frac{1}{L_e} \end{bmatrix} \mathrm{d}x \tag{4.209}$$

Na kraju se dobije da je matrica krutosti elementa u lokalnom koordinatnom sistemu elementa *za slučaj aksijalnog naprezanja*:

$$\boldsymbol{k_e} = \frac{A\, E}{L_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.210}$$

4.7.2.8.2 Uvijanje

U slučaju *uvijanja* grede, kao i u prethodnom slučaju, oba čvora elementa imaju po jedan stepen slobode i to ugaona pomeranja θ_{x1} i θ_{x2}.

$$\overline{\theta_x}(x) = \boldsymbol{N}(x)\boldsymbol{d_e} = \boldsymbol{N}(x) \begin{bmatrix} \theta_{x1} \\ \theta_{x2} \end{bmatrix} \tag{4.211}$$

Aproksimacija pomeranja θ_x se prvo definiše preko vektora baznih polinoma $\boldsymbol{p}$ i vektora nepoznatih $\boldsymbol{\alpha}$ koji su u ovom slučaju dvodimenzionalni.

$$\overline{\theta_x}(x) = \alpha_0 + \alpha_1\, x = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \boldsymbol{p}^T\, \boldsymbol{\alpha} \tag{4.212}$$

Matricom $\boldsymbol{P}$ se određuje veza između nepoznatih koeficijenata $\boldsymbol{\alpha}$ i pomeranja elemenata, tako da je:

$$\boldsymbol{d_e} = \boldsymbol{P}\alpha \tag{4.213}$$

Elemente matrice $\boldsymbol{P}$ je moguće odrediti na osnovu vrednosti pomeranja u čvorovima, na osnovu čega je:

$$\begin{aligned} \theta_{x1} &= \overline{\theta_x}(0) = \alpha_0 \\ \theta_{x2} &= \overline{\theta_x}(L_e) = \alpha_0 + \alpha_1\, L_e \end{aligned} \tag{4.214}$$

$$\begin{bmatrix}\theta_{x1}\\ \theta_{x2}\end{bmatrix} = \boldsymbol{P}\,\boldsymbol{\alpha} = \begin{bmatrix}1 & 0\\ 1 & L_e\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix} \tag{4.215}$$

Na osnovu jednačina 4.211, 4.212 i 4.213 je aproksimaciju pomeranja moguće izraziti kao:

$$\overline{\theta_x}(x) = \boldsymbol{p}^T\,\boldsymbol{P}^{-1}\,\boldsymbol{d_e} = \boldsymbol{N}(x)\boldsymbol{d_e} \tag{4.216}$$

Iz prethodne jednačine se vidi da je potrebno odrediti inverz matrice $\boldsymbol{P}$ kako bi se odredile funkcije oblika.

$$\boldsymbol{P}^{-1} = \begin{bmatrix}1 & 0\\ -\frac{1}{L_e} & \frac{1}{L_e}\end{bmatrix} \tag{4.217}$$

$$\boldsymbol{N}(x) = \boldsymbol{p}^T\,\boldsymbol{P}^{-1} = \begin{bmatrix}N_1 & N_2\end{bmatrix} = \begin{bmatrix}1-\frac{x}{L_e} & \frac{x}{L_e}\end{bmatrix} \tag{4.218}$$

U okviru jednačine 4.218 su date funkcije oblika N_1 i N_2, one su iste kao za prethodni slučaj i dijagram ovih funkcija odgovara onom datom na slici 4.32.

Na osnovu dobijenih funkcija oblika se vidi da se pomeranje θ_x po elementu menja linearno, pa se u ovom slučaju element naziva još i *linearni element*.

$$\theta_x = \theta_{x1} + \frac{\theta_{x2}-\theta_{x1}}{L_e}x \tag{4.219}$$

Sledeći korak je na osnovu veze između deformacija i pomeranja izraziti matricu deformacija.

$$\gamma_{yz} = \gamma_{zy} = r\frac{\partial\theta_x}{\partial x} = r\,\frac{\partial}{\partial x}\left(\boldsymbol{N}\,\boldsymbol{de}\right) = \boldsymbol{B}\,\boldsymbol{de} \tag{4.220}$$

$$\boldsymbol{B} = r\,\frac{\partial\boldsymbol{N}}{\partial x}\begin{bmatrix}-\frac{1}{L_e} & \frac{1}{L_e}\end{bmatrix} \tag{4.221}$$

Kada su poznate matrica deformacija i elastične karakteristike materijala za ovaj slučaj opterećenja, može se odrediti matrica krutosti elementa.

$$\boldsymbol{k_e} = \int_{V_e}\boldsymbol{B}^T\,\boldsymbol{c}\,\boldsymbol{B}\,\mathrm{d}V_e = \int_A r^2\,\mathrm{d}A\int_0^{L_e}\begin{bmatrix}-\frac{1}{L_e}\\ \frac{1}{L_e}\end{bmatrix}G\begin{bmatrix}-\frac{1}{L_e} & \frac{1}{L_e}\end{bmatrix}\mathrm{d}x \tag{4.222}$$

Na kraju se dobije da je matrica krutosti elementa u lokalnom koordinatnom sistemu elementa *za slučaj uvijanja*:

$$\boldsymbol{k_e} = \frac{G\,J}{L_e}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} \tag{4.223}$$

4.7.2.8.3 Savijanje u xy ravni

U slučaju *savijanja* grede u xy ravni, oba čvora elementa imaju po dva stepena slobode i to translatorna pomeranja v_1 i v_2 i ugaona pomeranja θ_{z1} i θ_{z2}.

$$\overline{v}(x) = \boldsymbol{N}(x)\boldsymbol{d_e} = \boldsymbol{N}(x)\begin{bmatrix}v_1\\ \theta_{z1}\\ v_2\\ \theta_{z2}\end{bmatrix} \tag{4.224}$$

Aproksimacija pomeranja u se prvo definiše preko vektora baznih polinoma $\boldsymbol{p}$ i vektora nepoznatih $\boldsymbol{\alpha}$ koji su u ovom slučaju četvorodimenzionalni.

$$\overline{v}(x) = \alpha_0 + \alpha_1\, x + \alpha_2\, x^2 + \alpha_3\, x^3 = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \boldsymbol{p}^T\, \boldsymbol{\alpha} \tag{4.225}$$

Matricom $\boldsymbol{P}$ se određuje veza između nepoznatih koeficijenata $\boldsymbol{\alpha}$ i pomeranja elemenata, tako da je:

$$\boldsymbol{d_e} = \boldsymbol{P}\alpha \tag{4.226}$$

Elemente matrice $\boldsymbol{P}$ je moguće odrediti na osnovu vrednosti pomeranja u čvorovima, na osnovu čega je:

$$\begin{aligned} v_1 &= \overline{v}(0) = \alpha_0 \\ \theta_{z1} &= \left.\frac{\mathrm{d}\overline{v}}{\mathrm{d}x}\right|_{x=0} = \alpha_1 \\ v_2 &= \overline{v}(L_e) = \alpha_0 + \alpha_1\, L_e + \alpha_2\, L_e^2 + \alpha_3\, L_e^3 \\ \theta_{z2} &= \left.\frac{\mathrm{d}\overline{v}}{\mathrm{d}x}\right|_{x=L_e} = \alpha_1 + 2\alpha_2\, L_e + 3\alpha_3\, L_e^2 \end{aligned} \tag{4.227}$$

$$\begin{bmatrix} v_1 \\ \theta_{z1} \\ v_2 \\ \theta_{z2} \end{bmatrix} = \boldsymbol{P}\,\boldsymbol{\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & L_e & L_e^2 & L_e^3 \\ 0 & 1 & 2L_e & 3L_e^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{4.228}$$

Na osnovu jednačina 4.224, 4.225 i 4.226 je aproksimaciju pomeranja moguće izraziti kao:

$$\overline{v}(x) = \boldsymbol{p}^T\, \boldsymbol{P}^{-1}\, \boldsymbol{d_e} = \boldsymbol{N}(x)\boldsymbol{d_e} \tag{4.229}$$

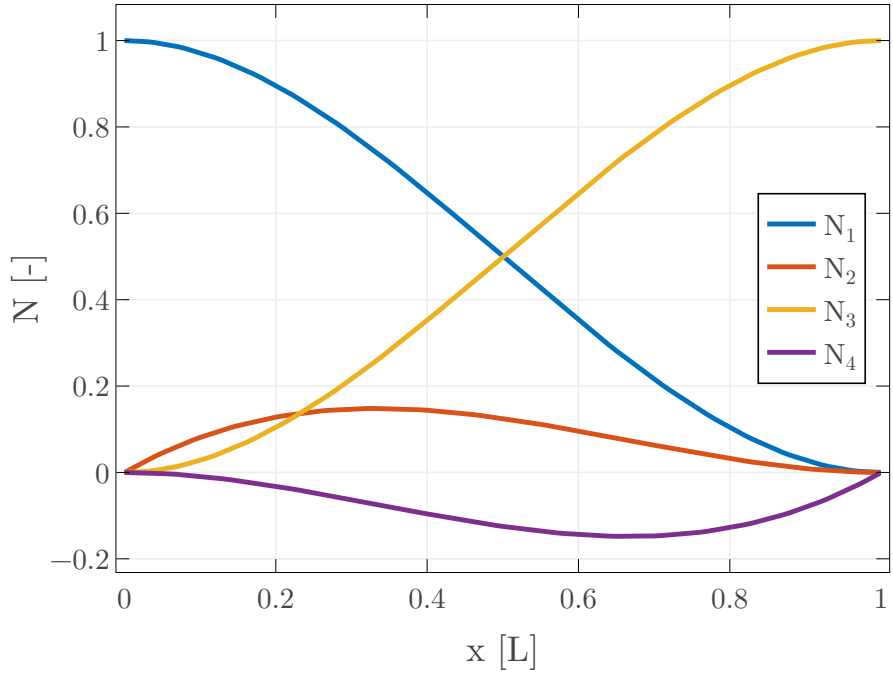
Iz prethodne jednačine se vidi da je potrebno odrediti inverz matrice $\boldsymbol{P}$ kako bi se odredile funkcije oblika.

$$\boldsymbol{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L_e^2} & -\frac{2}{L_e} & \frac{3}{L_e^2} & -\frac{1}{L_e} \\ \frac{2}{L_e^3} & \frac{1}{L_e^2} & -\frac{2}{L_e^3} & \frac{1}{L_e^2} \end{bmatrix} \tag{4.230}$$

$$\boldsymbol{N}(x) = \boldsymbol{p}^T\, \boldsymbol{P}^{-1} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \tag{4.231}$$

$$
\begin{aligned}
N_1 &= 1 - \frac{3}{L_e^2}\,x^2 + \frac{2}{L_e^3}\,x^3, \quad \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} = -\frac{6}{L_e^2} + \frac{12}{L_e^3}\,x \\
N_2 &= x - \frac{2}{L_e}\,x^2 + \frac{1}{L_e^2}\,x^3, \quad \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} = -\frac{4}{L_e} + \frac{6}{L_e^2}\,x \\
N_3 &= \frac{3}{L_e^2}\,x^2 - \frac{2}{L_e^3}\,x^3, \quad \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} = \frac{6}{L_e^2} - \frac{12}{L_e^3}\,x \\
N_4 &= -\frac{1}{L_e}\,x^2 + \frac{1}{L_e^2}\,x^3, \quad \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} = -\frac{2}{L_e} + \frac{6}{L_e^2}\,x
\end{aligned}
\tag{4.232}
$$

U okviru jednačine 4.232 su date funkcije oblika N_1, N_2, N_3 i N_4 kao i drugi izvodi funkcija oblika po promenljivoj x, dijagram ovih funkcija je dat na slici 4.33. Ovaj dijagram se dobija izvršavanjem skripte koja je data u okviru priloga G.



Slika 4.33 - Funkcije oblika za savijanje u xy ravni

Sledeći korak je na osnovu veze između deformacija i pomeranja izraziti matricu deformacija.

$$
\varepsilon_{xx} = -y\,\frac{\partial^2 v}{\partial x^2} = -y\,\frac{\partial^2}{\partial x^2}\left(\boldsymbol{N}\,\boldsymbol{de}\right) = \boldsymbol{B}\,\boldsymbol{de} \tag{4.233}
$$

$$
\boldsymbol{B} = -y\,\frac{\partial^2 \boldsymbol{N}}{\partial x^2} = -y\begin{bmatrix} \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} \end{bmatrix} \tag{4.234}
$$

Kada su poznate matrica deformacija i elastične karakteristike materijala za ovaj slučaj opterećenja, može se odrediti matrica krutosti elementa.

$$
\boldsymbol{k_e} = \int_{V_e} \boldsymbol{B}^T\,\boldsymbol{c}\,\boldsymbol{B}\,\mathrm{d}V_e = \int_A y^2\,\mathrm{d}A \int_0^{L_e} \begin{bmatrix} \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} \\ \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} \\ \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} \\ \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} \end{bmatrix} E \begin{bmatrix} \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} \end{bmatrix} \mathrm{d}x \tag{4.235}
$$

Na kraju se dobije da je matrica krutosti elementa u lokalnom koordinatnom sistemu elementa *za slučaj savijanja u xy ravni*:

$$\boldsymbol{k_e} = E\,I_z \begin{bmatrix} \frac{12}{L_e^3} & \frac{6}{L_e^2} & -\frac{12}{L_e^3} & \frac{6}{L_e^2} \\ \frac{6}{L_e^2} & \frac{4}{L_e} & -\frac{6}{L_e^2} & \frac{2}{L_e} \\ -\frac{12}{L_e^3} & -\frac{6}{L_e^2} & \frac{12}{L_e^3} & -\frac{6}{L_e^2} \\ \frac{6}{L_e^2} & \frac{2}{L_e} & -\frac{6}{L_e^2} & \frac{4}{L_e} \end{bmatrix} \tag{4.236}$$

4.7.2.8.4 Savijanje u xz ravni

U slučaju *savijanja* grede u xz ravni, oba čvora elementa imaju po dva stepena slobode i to translatorna pomeranja w_1 i w_2 i ugaona pomeranja θ_{y1} i θ_{y2}.

$$\overline{w}(x) = \boldsymbol{N}(x)\boldsymbol{d_e} = \boldsymbol{N}(x) \begin{bmatrix} w_1 \\ \theta_{y1} \\ w_2 \\ \theta_{y2} \end{bmatrix} \tag{4.237}$$

Aproksimacija pomeranja w se prvo definiše preko vektora baznih polinoma $\boldsymbol{p}$ i vektora nepoznatih $\boldsymbol{\alpha}$ koji su u ovom slučaju četvorodimenzionalni.

$$\overline{w}(x) = \alpha_0 + \alpha_1\,x + \alpha_2\,x^2 + \alpha_3\,x^3 = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \boldsymbol{p}^T\,\boldsymbol{\alpha} \tag{4.238}$$

Matricom $\boldsymbol{P}$ se određuje veza između nepoznatih koeficijenata $\boldsymbol{\alpha}$ i pomeranja elemenata, tako da je:

$$\boldsymbol{d_e} = \boldsymbol{P}\alpha \tag{4.239}$$

Elemente matrice $\boldsymbol{P}$ je moguće odrediti na osnovu vrednosti pomeranja u čvorovima, na osnovu čega je:

$$\begin{aligned} w_1 &= \overline{w}(0) = \alpha_0 \\ \theta_{y1} &= -\left.\frac{\mathrm{d}\overline{w}}{\mathrm{d}x}\right|_{x=0} = -\alpha_1 \\ w_2 &= \overline{w}(L_e) = \alpha_0 + \alpha_1\,L_e + \alpha_2\,L_e^2 + \alpha_3\,L_e^3 \\ \theta_{y2} &= -\left.\frac{\mathrm{d}\overline{w}}{\mathrm{d}x}\right|_{x=L_e} = -\alpha_1 - 2\alpha_2\,L_e - 3\alpha_3\,L_e^2 \end{aligned} \tag{4.240}$$

$$\begin{bmatrix} w_1 \\ \theta_{y1} \\ w_2 \\ \theta_{y2} \end{bmatrix} = \boldsymbol{P}\,\boldsymbol{\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & L_e & L_e^2 & L_e^3 \\ 0 & -1 & -2L_e & -3L_e^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{4.241}$$

Na osnovu jednačina 4.237, 4.238 i 4.239 je aproksimaciju pomeranja moguće izraziti kao:

$$\overline{w}(x) = \boldsymbol{p}^T \boldsymbol{P}^{-1} \boldsymbol{d_e} = \boldsymbol{N}(x)\boldsymbol{d_e} \tag{4.242}$$

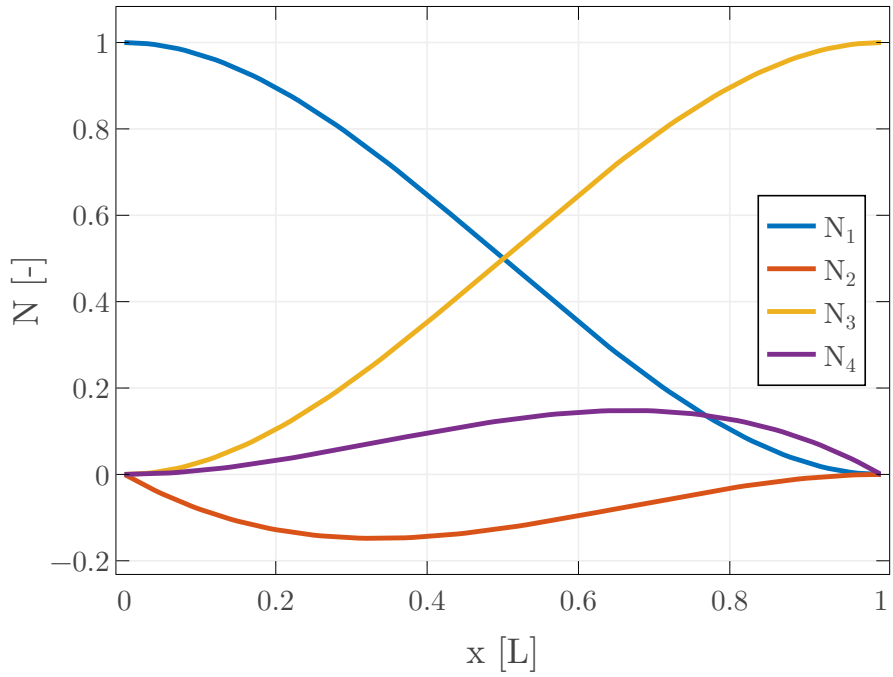
Iz prethodne jednačine se vidi da je potrebno odrediti inverz matrice $\boldsymbol{P}$ kako bi se odredile funkcije oblika.

$$\boldsymbol{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ -\frac{3}{L_e^2} & \frac{2}{L_e} & \frac{3}{L_e^2} & \frac{1}{L_e} \\ \frac{2}{L_e^3} & -\frac{1}{L_e^2} & -\frac{2}{L_e^3} & -\frac{1}{L_e^2} \end{bmatrix} \tag{4.243}$$

$$\boldsymbol{N}(x) = \boldsymbol{p}^T \boldsymbol{P}^{-1} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \tag{4.244}$$

$$\begin{aligned} N_1 &= 1 - \frac{3}{L_e^2}x^2 + \frac{2}{L_e^3}x^3, \quad \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} = -\frac{6}{L_e^2} + \frac{12}{L_e^3}x \\ N_2 &= -x + \frac{2}{L_e}x^2 - \frac{1}{L_e^2}x^3, \quad \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} = \frac{4}{L_e} - \frac{6}{L_e^2}x \\ N_3 &= \frac{3}{L_e^2}x^2 - \frac{2}{L_e^3}x^3, \quad \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} = \frac{6}{L_e^2} - \frac{12}{L_e^3}x \\ N_4 &= \frac{1}{L_e}x^2 - \frac{1}{L_e^2}x^3, \quad \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} = \frac{2}{L_e} - \frac{6}{L_e^2}x \end{aligned} \tag{4.245}$$

U okviru jednačine 4.245 su date funkcije oblika N_1, N_2, N_3 i N_4 kao i drugi izvodi funkcija oblika po promenljivoj x, dijagram ovih funkcija je dat na slici 4.34. Ovaj dijagram se dobija izvršavanjem skripte koja je data u okviru priloga G.



Slika 4.34 - Funkcije oblika za savijanje u xz ravni

Sledeći korak je na osnovu veze između deformacija i pomeranja izraziti matricu deformacija.

$$\varepsilon_{xx} = -z\frac{\partial^2 w}{\partial x^2} = -z\frac{\partial^2}{\partial x^2}(\boldsymbol{N}\,\boldsymbol{de}) = \boldsymbol{B}\,\boldsymbol{de} \tag{4.246}$$

$$\boldsymbol{B} = -z\,\frac{\partial^2 \boldsymbol{N}}{\partial x^2} = -z\,\begin{bmatrix} \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} \end{bmatrix} \tag{4.247}$$

Kada su poznate matrica deformacija i elastične karakteristike materijala za ovaj slučaj opterećenja, može se odrediti matrica krutosti elementa.

$$\boldsymbol{k_e} = \int_{V_e} \boldsymbol{B}^T\,\boldsymbol{c}\,\boldsymbol{B}\,\mathrm{d}V_e = \int_A z^2\,\mathrm{d}A \int_0^{L_e} \begin{bmatrix} \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} \\ \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} \\ \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} \\ \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} \end{bmatrix} E \begin{bmatrix} \frac{\mathrm{d}^2 N_1}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_2}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_3}{\mathrm{d}x^2} & \frac{\mathrm{d}^2 N_4}{\mathrm{d}x^2} \end{bmatrix} \mathrm{d}x \tag{4.248}$$

Na kraju se dobije da je matrica krutosti elementa u lokalnom koordinatnom sistemu elementa *za slučaj savijanja u xz ravni*:

$$\boldsymbol{k_e} = E\,I_y \begin{bmatrix} \frac{12}{L_e^3} & -\frac{6}{L_e^2} & -\frac{12}{L_e^3} & -\frac{6}{L_e^2} \\ -\frac{6}{L_e^2} & \frac{4}{L_e} & \frac{6}{L_e^2} & \frac{2}{L_e} \\ -\frac{12}{L_e^3} & \frac{6}{L_e^2} & \frac{12}{L_e^3} & \frac{6}{L_e^2} \\ -\frac{6}{L_e^2} & \frac{2}{L_e} & \frac{6}{L_e^2} & \frac{4}{L_e} \end{bmatrix} \tag{4.249}$$

4.7.2.8.5 Matrica krutosti u lokalnom koordinatnom sistemu elementa

Kada se matrice krutosti za pojedinačne slučajeve, koje su date u okviru jednačina 4.210, 4.223, 4.236 i 4.249, na odgovarajući način slože dobija se *matrica krutosti elementa tipa Ojler-Bernulijeva greda* u lokalnom koordinatnom sistemu elementa i sa sledeće tri jednačine se potpuno definiše jednačina konačnih elemenata za jedan element u slučaju statike.

$$\boldsymbol{k_e} = \begin{bmatrix} \frac{E\,A}{L_e} & 0 & 0 & 0 & 0 & 0 & -\frac{E\,A}{L_e} & 0 & 0 & 0 & 0 & 0 \\ & \frac{12\,E\,I_z}{L_e^3} & 0 & 0 & 0 & \frac{6\,E\,I_z}{L_e^2} & 0 & -\frac{12\,E\,I_z}{L_e^3} & 0 & 0 & 0 & \frac{6\,E\,I_z}{L_e^2} \\ & & \frac{12\,E\,I_y}{L_e^3} & 0 & -\frac{6\,E\,I_y}{L_e^2} & 0 & 0 & 0 & -\frac{12\,E\,I_y}{L_e^3} & 0 & -\frac{6\,E\,I_y}{L_e^2} & 0 \\ & & & \frac{G\,J}{L_e} & 0 & 0 & 0 & 0 & 0 & -\frac{G\,J}{L_e} & 0 & 0 \\ & & & & \frac{4\,E\,I_y}{L_e} & 0 & 0 & 0 & \frac{6\,E\,I_y}{L_e^2} & 0 & \frac{2\,E\,I_y}{L_e} & 0 \\ & & & & & \frac{4\,E\,I_z}{L_e} & 0 & -\frac{6\,E\,I_z}{L_e^2} & 0 & 0 & 0 & \frac{2\,E\,I_z}{L_e} \\ & & & & & & \frac{E\,A}{L_e} & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \frac{12\,E\,I_z}{L_e^3} & 0 & 0 & 0 & \frac{-6\,E\,I_z}{L_e^2} \\ & & & & & & & & \frac{12\,E\,I_y}{L_e^3} & 0 & \frac{6\,E\,I_y}{L_e^2} & 0 \\ & & & & & & & & & \frac{G\,J}{L_e} & 0 & 0 \\ & & & & & & & & & & \frac{4\,E\,I_y}{L_e} & 0 \\ \text{Sim.} & & & & & & & & & & & \frac{4\,E\,I_z}{L_e} \end{bmatrix} \tag{4.250}$$

$$\boldsymbol{d_e} = \begin{bmatrix} u_i & v_i & w_i & \theta_{xi} & \theta_{yi} & \theta_{zi} & u_j & v_j & w_j & \theta_{xj} & \theta_{yj} & \theta_{zj} \end{bmatrix}^T \tag{4.251}$$

$$\boldsymbol{f_e} = \begin{bmatrix} f_{xi} & f_{yi} & f_{zi} & m_{xi} & m_{yi} & m_{zi} & f_{xj} & f_{yj} & f_{zj} & m_{xj} & m_{yj} & m_{zj} \end{bmatrix}^T \tag{4.252}$$

4.7.2.8.6 Transformacija matrice krutosti

Transformaciju matrice krutosti iz lokalnog u globalni koordinatni sistem je moguće definisati na sledeći način:

$$\boldsymbol{K_e} = \boldsymbol{T}^T \boldsymbol{k_e} \boldsymbol{T} \tag{4.253}$$

Gde je *matrica transformacije* $\boldsymbol{T}$ definisana na sledeći način:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{r} & 0 & 0 & 0 \\ 0 & \boldsymbol{r} & 0 & 0 \\ 0 & 0 & \boldsymbol{r} & 0 \\ 0 & 0 & 0 & \boldsymbol{r} \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} C_{Xx} & C_{Yx} & C_{Zx} \\ C_{Xy} & C_{Yy} & C_{Zy} \\ C_{Xz} & C_{Yz} & C_{Zz} \end{bmatrix} \tag{4.254}$$

$$\begin{aligned} C_{Xx} &= \cos(x,X), \quad C_{Yx} = \cos(x,Y), \quad C_{Zx} = \cos(x,Z) \\ C_{Xy} &= \cos(y,X), \quad C_{Yy} = \cos(y,Y), \quad C_{Zy} = \cos(y,Z) \\ C_{Xz} &= \cos(z,X), \quad C_{Yz} = \cos(z,Y), \quad C_{Zz} = \cos(z,Z) \end{aligned} \tag{4.255}$$

4.7.3 Proračun deformacija

Do sada je na osnovu Hamiltonovog principa izvedena jednačina konačnih elemenata, prikazana je mogućnost korišćenja Lagranževih množioca za formiranje sfernih zglobova, izvedeno je sve što je neophodno za korišćenje elementa tipa Ojler-Bernulijeva greda i definisana je geometrija mehanizma uvođenjem geometrijskih parametara i rešavanjem inverznog kinematičkog problema. Zahvaljujući tome sada je moguće posmatrati mehanizam kao sistem elastičnih tela što je na druge načine već analizirano [12].

Dobijen je sistem algebarskih jednačina kojim se aproksimira rešenje sistema parcijalnih diferencijalnih jednačina i sledeći korak je implementacija svega navedenog u okviru jednog procesa kojim je moguće izvršiti kontrolisanu materijalizaciju kinematičkog modela mehanizma simulatora leta. Za kriterijum prilikom dimenzionisanja segmenata mehanizma je usvojen *kriterijum maksimalnih dozvoljenih pomeranja*, odnosno kriterijum maksimalnih dozvoljenih deformacija konstrukcije.

Zbog jednostavnijeg definisanja a potom i prikaza samog postupka, usvojeno je da svi konačni elementi imaju isti poprečni presek pa se segmenti mehanizma izrađuju od cevi kružnog poprečnog preseka i *usvojeni materijal je običan konstrukcioni čelik*. Poprečni presek je definisan spoljnim prečnikom D i debljinom zida t. U tabeli 4.4 su prikazane vrednosti svih parametara koji su još potrebni za proračun čvrstoće.

Tabela 4.4 - Usvojeni parametri za analizu čvrstoće

Parametar	Vrednost
m	250 kg
$\boldsymbol{C_{in}}$	[0, 0, 500] mm
E	210 GPa
ν	0.3
D	50 mm
t	2 mm
λ_{max}	35°

Spoljašnje opterećenje je definisano sa masom tereta m i centrom inercije $\boldsymbol{C_{in}}$ čije vrednosti su prikazane u prethodnoj tabeli. U centru inercije je definisan prvi čvor prvog elementa i zato je moguće inercijalno opterećenje uvesti samo kao silu u tom čvoru, što se u okviru proračuna čvrstoće metodom konačnih elemenata izvodi na sledeći način:

$$\boldsymbol{F}(3) = -mg \tag{4.256}$$

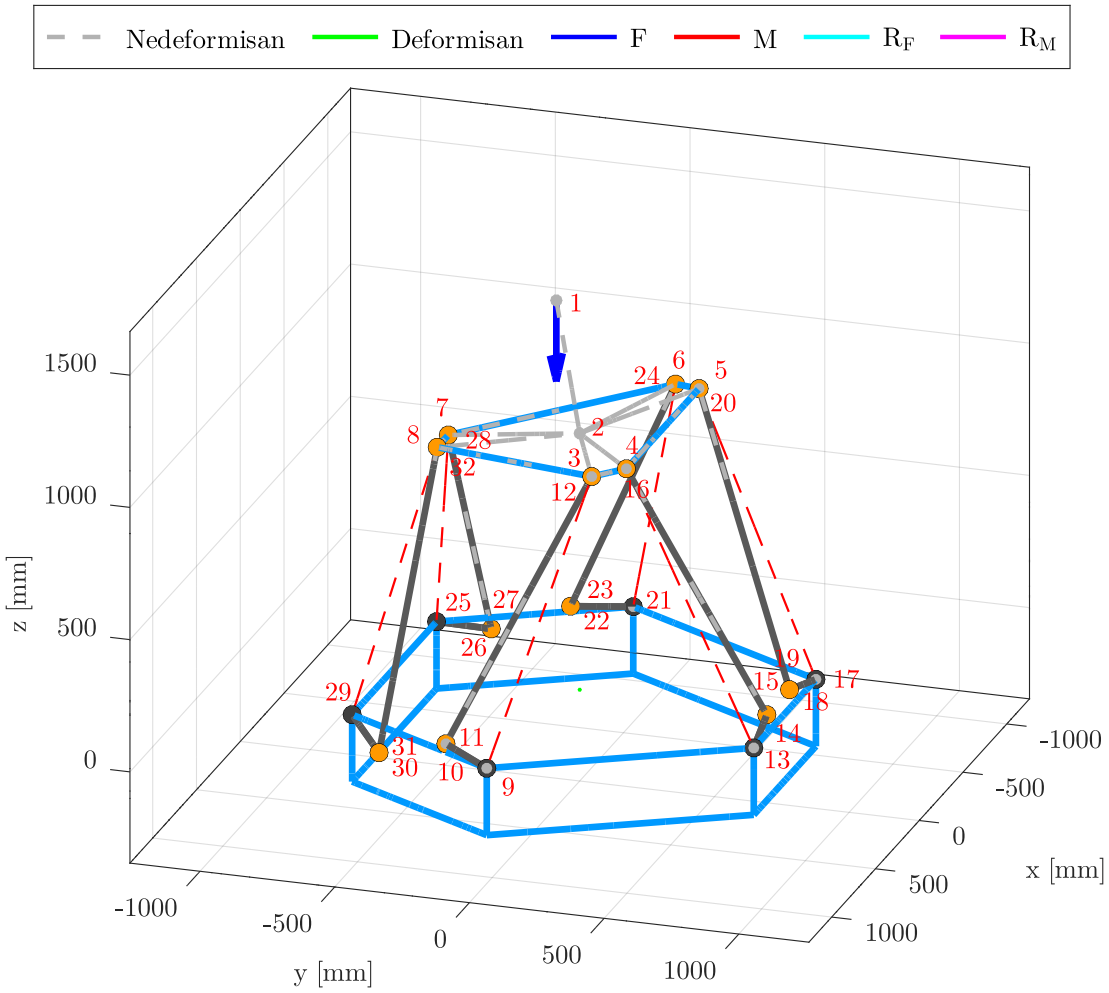
Treba napomenuti da je ovaj proces iterativan i da se vrednosti za dimenzije poprečnog preseka prvo pretpostavljaju i zatim menjaju u skladu sa dobijenim rezultatima. Na osnovu usvojenih vrednosti se, u slučaju kružnog poprečnog preseka cevi, računaju karakteristike poprečnog preseka sledećim izrazima:

$$A = \left(D^2 - (D - 2\,t)^2\right)\,\frac{\pi}{4} \tag{4.257}$$

$$I_y = I_z = \left(D^4 - (D - 2\,t)^4\right)\,\frac{\pi}{64} \tag{4.258}$$

$$J = \left(D^4 - (D - 2\,t)^4\right)\,\frac{\pi}{32} \tag{4.259}$$

Na slici 4.35 je prikazan kinematički model mehanizma simulatora leta, ali ovog puta sa konačnim elementima i spoljašnjim opterećenjem. Svi konačni elementi, kojih je ukupno 25, su definisani sa *32 čvora* (svi su označeni na slici) sa ukupno 192 stepena slobode.



Slika 4.35 - Prikaz konačnih elemenata i opterećenja

Gornje i donje poluge su definisane sa po jednim elementom, dok je platforma definisana sa preostalih 13 elemenata. Da bi se primenom metode Lagranževih množioca definisali sferni zglobovi na njihovom mestu je jedan čvor podeljen na dva čvora koji se fizički nalaze na istom mestu i za koje se potom definiše da imaju jednaka pomeranja. U sledećim tabelama (tabele 4.5 i 4.6) su dati *čvorovi koji se spajaju i formiraju donje i gornje sferne zglobove*.

Tabela 4.5 - Čvorovi donjih sfernih zglobova

Prvi čvor	10	14	18	22	26	30
Drugi čvor	11	15	19	23	27	31

Tabela 4.6 - Čvorovi gornjih sfernih zglobova

Prvi čvor	3	4	5	6	7	8
Drugi čvor	12	16	20	24	28	32

Da bi proračun bio potpuno definisan ostalo je još samo da se definišu *konturni uslovi*. To je urađeno tako što je usvojeno da se na mestima čvorova 9, 13, 17, 21, 25 i 29 nalaze uklještenja. To podrazumeva da su ovim čvorovima oduzeti svi stepeni slobode kretanja, odnosno da nema mogućih pomeranja.

Sledi implementacija samog procesa proračuna u programskom paketu `JSLAB` (isto je prethodno urađeno u programskom paketu `GNU Octave` i `Matlab` na osnovu [132] i pomoću [133]). Napisana je klasa `MP_6RSS_FEM_SOLVER` (data u prilogu H), prvo se metodom `construct()` na osnovu ulaznih argumenata određuju sve potrebne početne vrednosti kako bi se formulisao metod konačnih elemenata. Definišu se matrice za implementaciju Lagranževih množioca, određuju se karakteristike poprečnog preseka i materijala i na kraju se poziva metoda za određivanje globalne matrice krutosti.

Metodom za određivanje globalne matrice krutosti `stiffnessBeam3D()`, klase `MP_6RSS_FEM_SOLVER`, se formira prvo matrica krutosti elementa tipa Ojler-Bernulijeva greda za pojedinačne elemente na osnovu jednačina koje su prikazane u ovom radu. U svakom ciklusu petlje kojom se prolazi kroz sve definisane elemente se uticaj pojedinačnih elemenata transformiše iz lokalnog u globalni koordinatni sistem i zatim dodaje u okviru globalne matrice krutosti. Nakon poslednjeg ciklusa petlje dobija se matrica krutosti svih elemenata u globalnom koordinatnom sistemu.

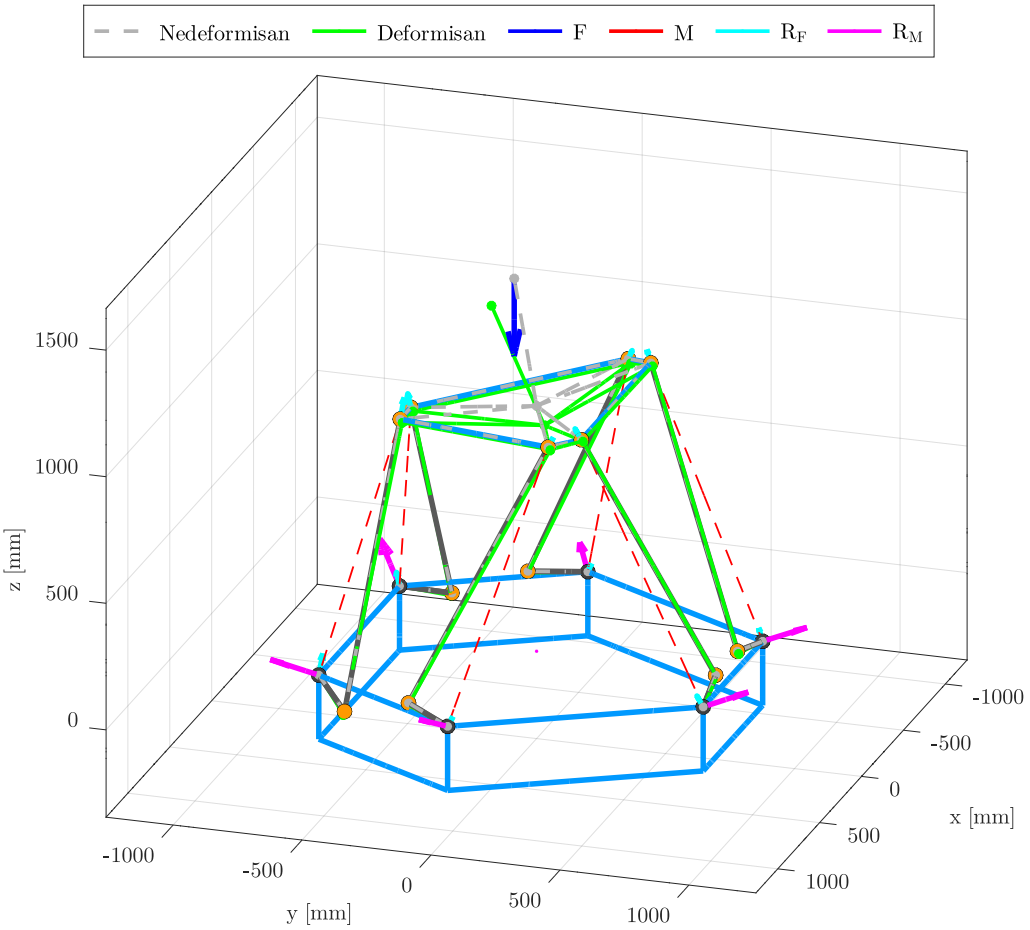
Metodom `getReactions()` klase `MP_6RSS_FEM_SOLVER` se rešava jednačina konačnih elemenata za slučaj statike. Na osnovu određene matrice krutosti se prvo formira proširena matrica krutosti zbog sfernih zglobova, zatim se na osnovu konturnih uslova definiše redukovan sistem jednačina konačnih elemenata. *Rešavanjem ovog sistema jednačina se dobijaju tražena pomeranja.* Vraćanjem dobijenih pomeranja u jednačinu konačnih elemenata se dobijaju nepoznate reakcije veza.

Moguće je zaključiti da u ovom slučaju reakcije veza predstavljaju opterećenja aktuatora koji se nalaze na mestu baš tih uklještenih čvorova, ova činjenica omogućava da se na ovaj način odredi i potreban moment obrtnih aktuatora.

Metodom `getTorques()`, klase `MP_6RSS_FEM_SOLVER`, se na osnovu sledeće jednačine određuje potrebni obrtni moment aktuatora kao projekcije dobijene momentne rekacije veze na obrtnu osu aktuatora (skalarni proizvod vektora momentne reakcije veze i jediničnog vektora obrtne ose aktuatora).

$$M_{m,i} = \boldsymbol{M_i} \cdot \boldsymbol{i_{m,i}} \tag{4.260}$$

U okviru priloga I je dat listing i objašnjena je skripta `solveInverseStaticFEM.jsl`, kojom se sve prethodno prikazane funkcije povezuju u proces proračuna čvrstoće ovog mehanizma. Na slici 4.36 je sada prikazan i deformisan oblik konstrukcije, pri čemu su *pomeranja uvećana 100 puta* kako bi se video karakter deformacije konstrukcije, i vektori reakcija veza.



Slika 4.36 - Prikaz uvećanih elastičnih deformacija konstrukcije

Treba napomenuti da je prikazan proračun samo za jednu vrednost vektora upravljačkih veličina i to za $\boldsymbol{x} = [0, 0, 0, 0, 0, 10]^T$ i da se ovakav proračun vrši za sve kritične slučajeve u okviru radnog prostora mehanizma. Na kraju se na osnovu analize rezultata zaključuje da su izabrane cevi, kojim se materijalizuju segmenti mehanizma, odgovarajuće po osnovu kriterijuma da su sva pomeranja manja od 2 mm, što je prethodno usvojeno.

4.8 Potrebni obrtni moment aktuatora

Na osnovu Jakobijeve matrice definisane u jednačini 4.83 je moguće odrediti potrebni moment aktuatora ukoliko je poznato opterećenje mehanizma i ukoliko se zanemare inercijalne karakteristike poluga i donjih sfernih zglobova. Potrebni moment aktuatora je moguće odrediti na osnovu jednačine

$$\boldsymbol{M_m} = \boldsymbol{J}^T \boldsymbol{F_x} \tag{4.261}$$

gde je $\boldsymbol{F_x}$ spoljašnje opterećenje mehanizma. Prethodna jednačina je dobijena na osnovu principa virtualnog rada jer na osnovu jednačine 4.78 važi da je:

$$\delta \boldsymbol{x} = \boldsymbol{J} \delta \boldsymbol{\alpha} \tag{4.262}$$

Ukoliko se izrazi rad spoljašnjeg opterećenja $\boldsymbol{F_x}$ i iskoristi prethodna jednakost dobija se da je:

$$\boldsymbol{F_x}^T \delta \boldsymbol{x} = \boldsymbol{F_x}^T \boldsymbol{J} \delta \boldsymbol{\alpha} \tag{4.263}$$

Prethodno napisani rad mora biti jednak radu aktuatora, odnosno važi:

$$\boldsymbol{M_m}^T \delta \boldsymbol{\alpha} = \boldsymbol{F_x}^T \boldsymbol{J} \delta \boldsymbol{\alpha} \tag{4.264}$$

Ukoliko se prethodna jednačina napiše u obliku

$$\boldsymbol{M_m}^T \delta \boldsymbol{\alpha} = \left(\boldsymbol{J}^T \boldsymbol{F_x} \right)^T \delta \boldsymbol{\alpha} \tag{4.265}$$

vidi se da se na kraju dobija jednačina 4.261.

Osim prethodno prikazanog postupka, moguće je iskoristiti i činjenicu da se opterećenje sa platforme na donje poluge prenosi preko sfernih zglobova i na osnovu toga odrediti silu. Ovaj postupak je implementiran u okviru metode `getForces()` klase `MP_6RSS_SOLVER`.

$$\boldsymbol{F} = \left(\boldsymbol{J_x}^T \right)^{-1} \boldsymbol{F_x} \tag{4.266}$$

Prethodno je moguće ponoviti, ali u koordinatnom sistemu platforme.

$$\boldsymbol{F} = \left(\boldsymbol{J_{xp}}^T \right)^{-1} \boldsymbol{F_{xp}} \tag{4.267}$$

$$\boldsymbol{F_{xp}} = \left[\boldsymbol{R}^{-1} (m\boldsymbol{g}) \quad \boldsymbol{C_{inp}} \times \left(\boldsymbol{R}^{-1} (m\boldsymbol{g}) \right) \right]^T \tag{4.268}$$

$$\boldsymbol{J_{xp}} = \left[\boldsymbol{i_{Fm}} \quad \boldsymbol{p} \times \boldsymbol{i_{Fm}} \right]^T \tag{4.269}$$

$$\boldsymbol{i_{Fm}} = \boldsymbol{R}^{-1} \frac{\boldsymbol{AP}}{s} \tag{4.270}$$

Jedna od karakteristika matrice rotacije jeste da je inverz matrice rotacije jednak transponovanoj matrici rotacije. Ova karakteristika je iskorišćena u metodi `getForcesPlatform()` (koja je implementirana u okviru klase `MP_6RSS_SOLVER`) jer je potrebna transformacija koordinata iz koordinatnog sistema platforme u koordinatni sistem baze. Ovom metodom je implementirano određivanje sila u gornjim sfernim zglobovima u koordinatnom sistemu platforme.

Kada je određena sila na mestu donjeg sfernog zgloba, potrebno je odrediti moment na mestu aktuatora, a zatim i komponentu oko ose obrtanja aktuatora što je implementirano u metodi `getTorquesByForce()` klase `MP_6RSS_SOLVER`.

$$\boldsymbol{M} = \boldsymbol{BA} \times F \tag{4.271}$$

$$\boldsymbol{M_m} = \boldsymbol{M} \cdot \boldsymbol{i_m} \tag{4.272}$$

Kao što je već objašnjeno, sastavni deo aktuatora je prenosnik snage (reduktor) koji je osim potrebnog momenta neophodno dimenzionisati i u odnosu na radijalnu i aksijalnu silu. Prvo se metodom `getActuatorAllowedLoad()` klase `MP_6RSS_SOLVER` izračunavaju dozvoljene radijalne i aksijalne sile za poznati položaj napadne tačke sile na osnovu sledećih jednačina.

$$F_{Rx} = F_R \frac{a}{b+x} \tag{4.273}$$

$$F_A = \frac{1}{5} \boldsymbol{F_R} \tag{4.274}$$

Zatim se određuju i konkretne vrednosti za određeno opterećenje i to metodom `getActuatorLoad()` klase `MP_6RSS_SOLVER` prema sledećim jednačinama. Uslovi su da je $F_a < F_A$ i da je $F_r < F_{Rx}$.

$$F_a = \|\boldsymbol{F} \cdot \boldsymbol{i_m}\| \tag{4.275}$$

$$F_r = \|\boldsymbol{F} - F_a \boldsymbol{i_m}\| \tag{4.276}$$

4.8.1 Inverzni statički problem

U slučaju statike spoljašnje opterećenje se može napisati u obliku

$$\boldsymbol{F_x} = \begin{bmatrix} m\boldsymbol{g} & (\boldsymbol{C_{in}} - \boldsymbol{O_p}) \times m\boldsymbol{g} \end{bmatrix}^T \tag{4.277}$$

gde je $\boldsymbol{C_{in}}$ vektor položaja centra inercije u koordinatnom sistemu baze. Na osnovu svega prethodnog je moguće napisati metodu `getTorques()` klase `MP_6RSS_SOLVER`, za izračunavanje potrebnog momenta aktuatora.

Sada se mogu iskoristiti prethodno definisane jednačine, kao što je 4.261, i odrediti potrebne obrtne momente aktuatora i doći do rešenja inverznog statičkog problema.

4.8.2 Inverzni dinamički problem

U slučaju dinamike spoljašnje opterećenje se na osnovu slučaja statike i brzina i ubrzanja, prikazanih u 4.6, može napisati u obliku

$$\boldsymbol{F_x} = \begin{bmatrix} \boldsymbol{F} & (\boldsymbol{C_{in}} - \boldsymbol{O_p}) \times \boldsymbol{F} - (\boldsymbol{I}\ \boldsymbol{\epsilon_p} + \boldsymbol{\omega_p} \times (\boldsymbol{I}\ \boldsymbol{\omega_p})) \end{bmatrix}^T \tag{4.278}$$

gde je $\boldsymbol{I}$ matrica inercije tereta , dok je $\boldsymbol{F}$:

$$\boldsymbol{F} = m\,(\boldsymbol{g} - \boldsymbol{a_{C_{in}}}) \tag{4.279}$$

Ubrzanje tačke C_{in} ($\boldsymbol{a_{C_{in}}}$) je moguće izraziti preko translatornog i ugaonog ubrzanja platforme kao i ugaone brzine platforme, na osnovu:

$$\boldsymbol{a_{C_{in}}} = \boldsymbol{a_p} + \boldsymbol{\epsilon_p} \times \left(\boldsymbol{C_{in}} - \boldsymbol{O_p}\right) + \boldsymbol{\omega_p} \times (\boldsymbol{\omega_p} \times \left(\boldsymbol{C_{in}} - \boldsymbol{O_p}\right)) \tag{4.280}$$

Sada se mogu iskoristiti prethodno definisane jednačine, kao što je 4.261, i odrediti potrebne obrtne momente aktuatora i doći do rešenja inverznog dinamičkog problema. U poglavlju 7.1.2 je urađena uporedna analiza rezultata određivanja obrtnog momenta u slučaju inverznog dinamičkog problema korišćenjem ovde prikazanog rešenja i korišćenjem programskog paketa `OpenModelica`.

4.9 Indikatori performansi mehanizma

Prvi uslov proračuna optimalnih parametara je da postoji određeni parametar na osnovu koga je moguće izmeriti uspešnost pojedinačnih rešenja i nakon toga uporediti različita rešenja. Potrebno je konstruisati funkciju čija vrednost će biti mera uspešnosti rešenja. Kada se radi o složenim inženjerskim problemima, ovu funkciju nije uvek lako formirati. Potrebno je odrediti parametre koji će predstavljati indikatore performansi, u ovom slučaju, mehanizma simulatora leta.

Kao što je prethodno napomenuto, ovi indikatori moraju da omoguće upoređivanje rešenja i iz tog razloga moraju biti zasnovani na normalizovanim veličinama. U ovom slučaju je neophodno normalizovati Jakobijevu matricu jer elementi imaju različite merne jedinice. Određeni elementi se dele sa poluprečnikom platforme ili sa dužinom donje poluge kako bi se izvršila normalizacija.

$$\boldsymbol{J_{xn}} = \begin{bmatrix} \boldsymbol{i_{ul}} & \boldsymbol{O_pP_i} \times \boldsymbol{i_{ul}}/R_p \end{bmatrix} \tag{4.281}$$

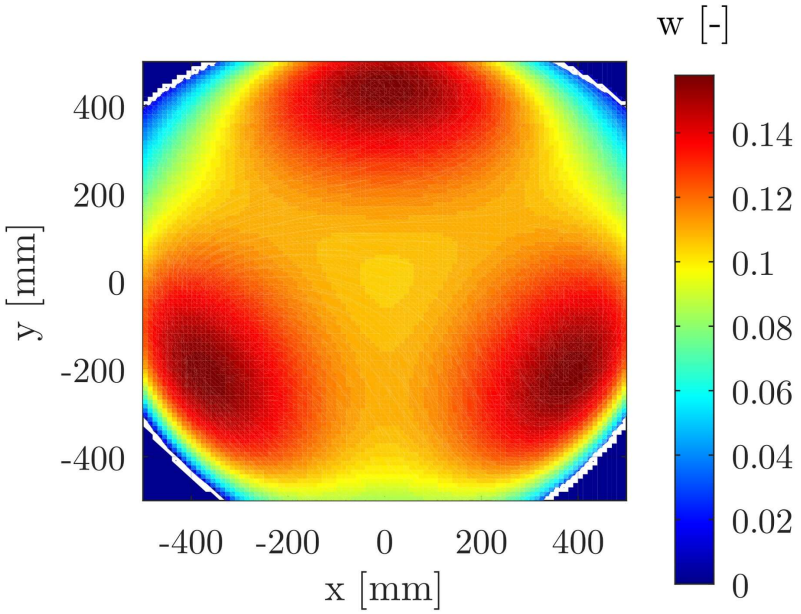
$$\boldsymbol{J_{\alpha n}} = \boldsymbol{J_{\alpha}}/a \tag{4.282}$$

Na osnovu prethodne dve jednačine je normalizovana Jakobijeva matrica (metoda `getJacobianNormalized()`, klase `MP_6RSS_SOLVER`, se koristi za određivanje ove matrice):

$$\boldsymbol{J_n} = \boldsymbol{J_{xn}^{-1}}\boldsymbol{J_{\alpha n}} \tag{4.283}$$

U radu [56] se definiše *manipulabilnost robota* kao apsolutna vrednost determinante Jakobijeve matrice, koja je u ovom slučaju prethodno normalizovana (metoda `getManipulability()` klase `MP_6RSS_SOLVER` se koristi za određivanje ovog indikatora).

$$w = \left|\det\left(\boldsymbol{J_n}\right)\right| \tag{4.284}$$



Slika 4.37 - Promena indikatora manipulabilnosti u ravni

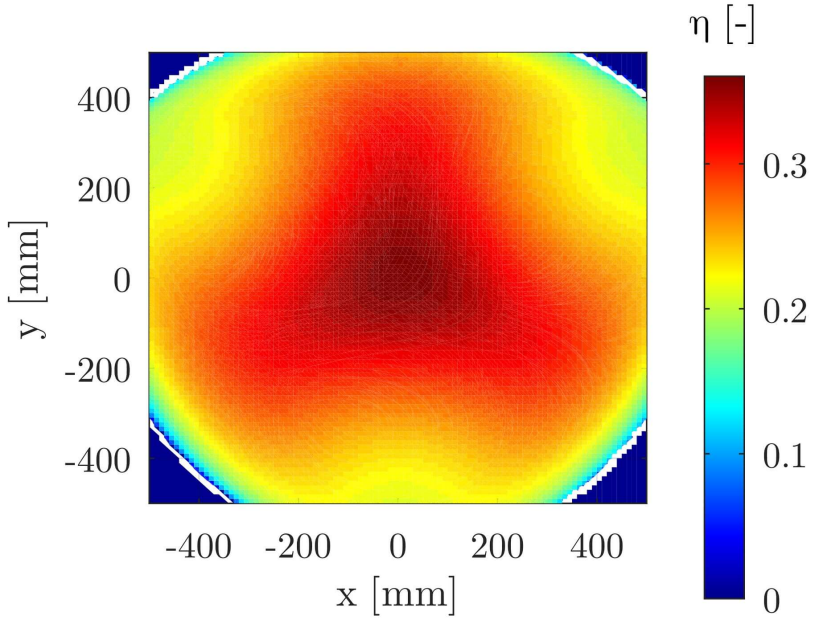
Na prethodnom grafiku (slika 4.37) je prikazan primer vrednosti indikatora manipulabilnosti u zavisnosti od koordinata `x` i `y`.

U radu [38] se definiše *spretnost robota* kao recipročna vrednost kondicionog broja Jakobijeve matrice, koja je u ovom slučaju prethodno normalizovana (metoda

`getDexterity()` klase `MP_6RSS_SOLVER` se koristi za određivanje ovog indikatora). Kondicioni broj matrice je proizvod euklidske norme te matrice i euklidske norme inverzne matrice.

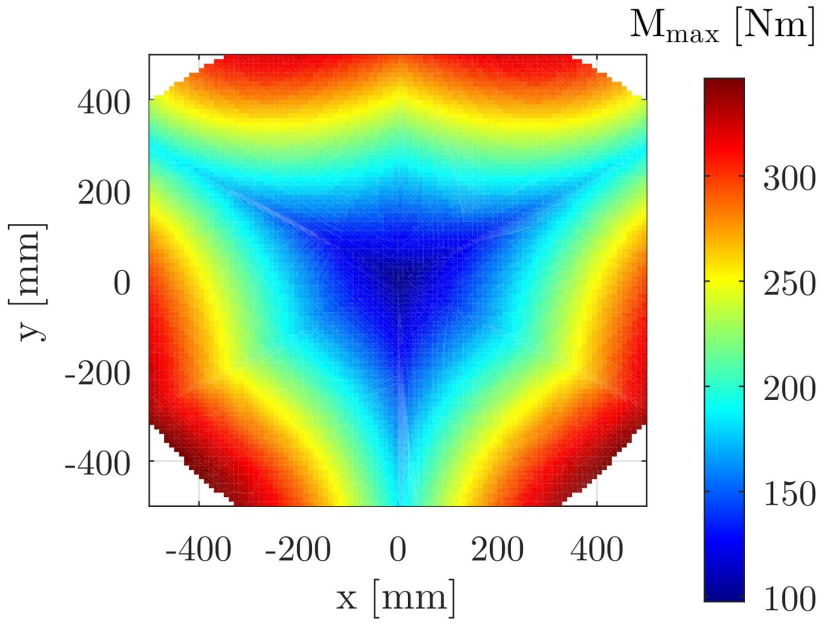
$$\eta = \frac{1}{k} = \frac{1}{\text{cond}(\boldsymbol{J_n})} = \frac{1}{\|\boldsymbol{J_n}\|_2 \cdot \left\|\boldsymbol{J_n^{-1}}\right\|_2} \tag{4.285}$$

Na sledećem grafiku (slika 4.38) je prikazan primer vrednosti indikatora spretnosti u zavisnosti od koordinata `x` i `y`.



Slika 4.38 - Promena indikatora spretnosti u ravni

Na osnovu funkcije `getTorques()` klase `MP_6RSS_SOLVER` je moguće odrediti potrebne momente aktuatora (nakon čega i maksimalni potrebni obrtni moment). Primer promene indikatora maksimalnog potrebnog momenta u jednoj ravni je dat na slici 4.39.



Slika 4.39 - Promena indikatora maksimalnog potrebnog momenta u ravni

Zbog potrebe da mehanizam simulira let letelice, poželjno je da mehanizam može da postigne što veće uglove propinjanja i valjanja. Ukoliko se menja samo jedan ugao a sve ostale spoljašnje koordinate fiksiraju može se odrediti maksimalni ugao za koji postoji rešenje inverznog kinematičkog problema. Ipak osim maksimalnih ostvarivih parametara, mehanizam treba da ima i zadovoljavajući radni prostor, kao još jedan indikator performansi.

4.9.1 Algoritam SpaceSearch

Maksimalne ostvarive parametre i radni prostor mehanizma moguće je odrediti jednostavnom diskretizacijom prostora i zatim proverom da li postoji rešenje inverznog kinematičkog problema za takav položaj i orijentaciju ili ne. Da li postoji rešenje inverznog kinematičkog problema ili ne se implementira kao generalizovana *funkcija uslova* koja može vratiti samo vrednosti `true` ili `false` (ili njima odgovarajuće `1` i `0`). Ovaj postupak može biti vremenski veoma zahtevan ukoliko je potrebno postići relativno dobru preciznost. Može se primetiti da ukoliko je potrebno odrediti samo granice na kojima dolazi do promene rešivosti ovog problema da se provera nepotrebno ponavlja u nekim regionima, što je moguće izbeći i skratiti vreme proračuna.

Algoritam `SpaceSearch`, implementiran u klasi `SPACE_SEARCH` (kompletno data u prilogu J), osmišljen je za efikasno istraživanje i analizu N-dimenzionalnih prostora kroz rekurzivnu deobu na manje hiperkocke. Ovaj metod je posebno efikasan za identifikaciju oblasti unutar višedimenzionalnog prostora koje zadovoljavaju specifičan uslov (definisan u okviru funkcije uslova), što je ključno u različitim oblastima kao što su računarska geometrija, optimizacija i simulacija.

U svojoj osnovi, algoritam počinje definisanjem granica N-dimenzionalnog prostora, predstavljenih kao niz u kojem svaki element sadrži minimalne i maksimalne vrednosti za neku dimenziju. Prostor se inicijalno deli na veće hiperkocke na osnovu ovih granica i početnih faktora podele obezbeđenih za svaku dimenziju. Početne veličine koraka za svaku dimenziju se izračunavaju deljenjem opsega svake dimenzije odgovarajućim faktorom podele. Ove veličine koraka određuju dimenzije elemenata (*hiperkocki*) na početnom nivou dubine.

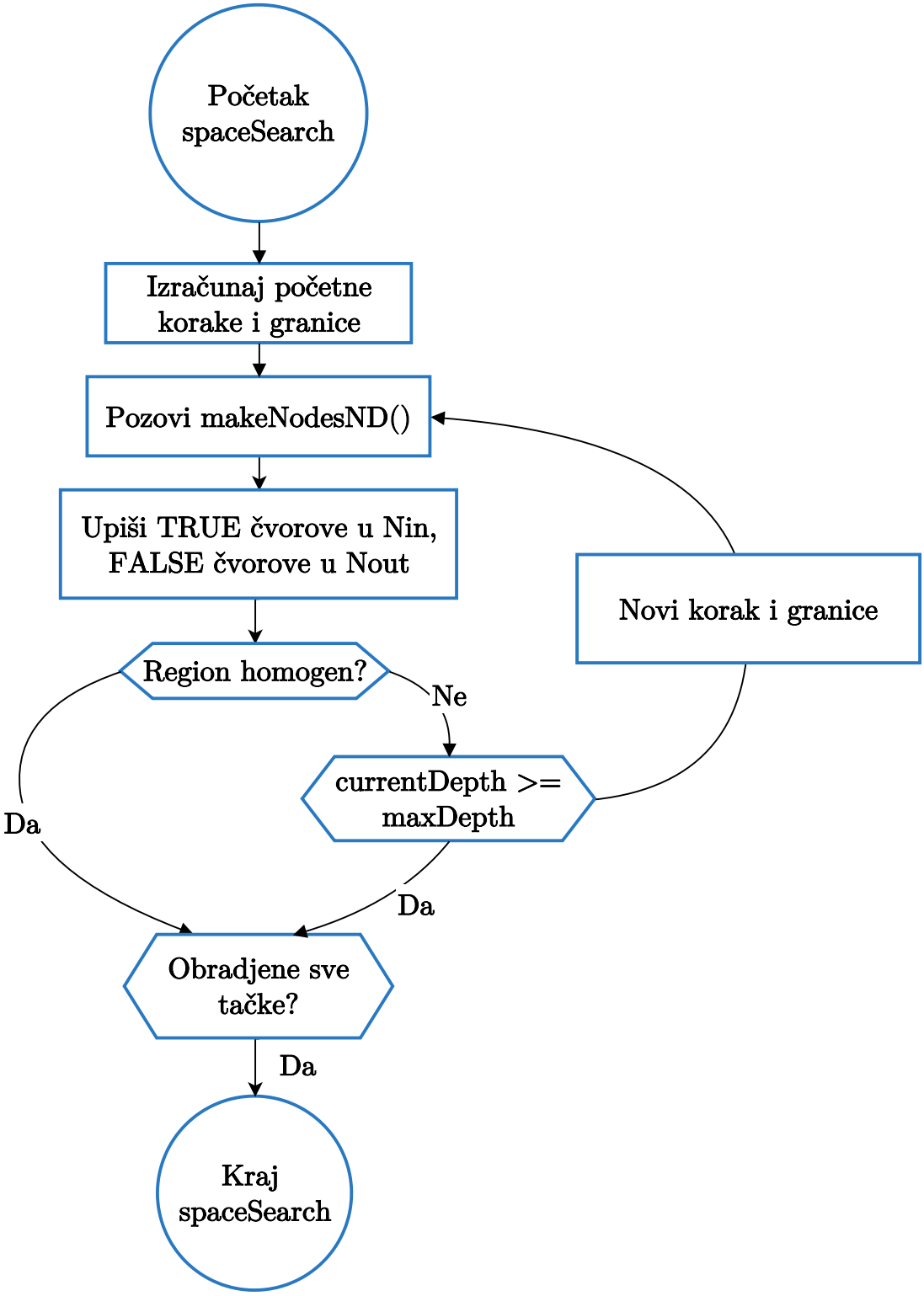
Algoritam zatim izračunava funkciju uslova, koju definiše korisnik, u čvorovima ovih početnih hiperkocki. Čvorne tačke su ključne jer predstavljaju ekstreme hiperkocke u svakoj dimenziji, a izračunavanjem funkcije uslova u ovim tačkama pomaže da se utvrdi da li cela hiperkocka zadovoljava uslov ili ne. Ako sve čvorne tačke daju isti rezultat, bilo da sve zadovoljavaju uslov ili nijedna ne zadovoljava, hiperkocka se smatra homogenom u odnosu na uslov, i dalja podela za taj region nije potrebna.

Međutim, ako čvorne tačke daju mešovite rezultate, tj. neke zadovoljavaju uslov, a druge ne, to ukazuje na to da hiperkocka sadrži granicu između regiona koji zadovoljavaju uslov i onih koji ga ne zadovoljavaju. U takvim slučajevima, algoritam rekurzivno deli hiperkocku na manje hiperkocke koristeći finije faktore podele za sledeći nivo dubine. Veličine koraka se ažuriraju deljenjem prethodnih veličina koraka novim faktorima podele, što efektivno povećava preciznost pretrage u oblastima od interesa.

Ova rekurzivna podela se nastavlja sve dok se ne dostigne maksimalna dubina ili dok sve preostale hiperkocke ne postanu homogene u odnosu na funkciju uslova. Fokusirajući računsku snagu na regione gde se promena zadovoljenja uslova javlja, algoritam efikasno identifikuje interesantne granice unutar N-dimenzionalnog prostora.

Jedna od bitnih karakteristika algoritma je sposobnost da beleži tačke koje zadovoljavaju i ne zadovoljavaju funkciju uslova. Održava dva niza koji akumuliraju koordinate tačaka. Ova dva niza tačaka su značajna za rekonstrukciju interesantnih regiona unutar prostora i mogu se koristiti za dalju analizu ili vizualizaciju.

Na slici 4.40 je dat grafički prikaz algoritma `spaceSearch`.

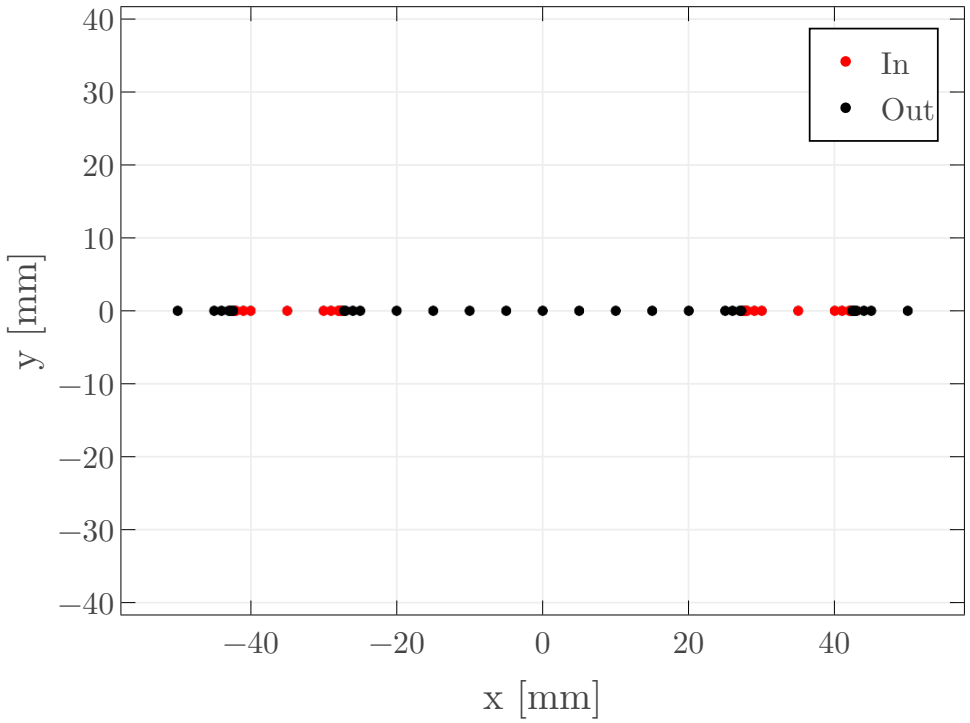


Slika 4.40 - Grafički prikaz algoritma spaceSearch

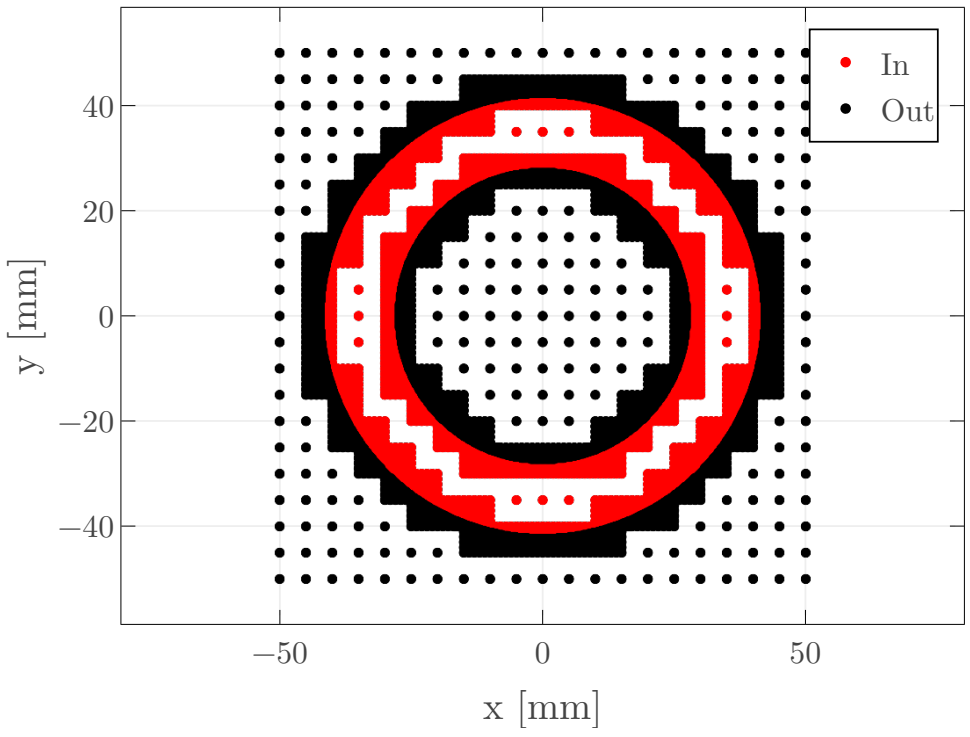
Da bi se omogućila efikasna računanja, naročito u visoko-dimenzionalnim prostorima, algoritam koristi nekoliko pomoćnih metoda. Jedna od njih generiše sve moguće pomeraje čvorova za hiperkocku u N *dimenzija*, što je ključno za identifikaciju čvornih tačaka tokom svakog rekurzivnog poziva. Druga metoda pronalazi indekse ovih čvornih tačaka unutar strukture mreže, omogućavajući algoritmu da pristupi i proceni potrebne tačke bez redundantnih izračunavanja.

Pored toga, algoritam je osmišljen tako da podržava paralelizaciju izvršavanja. Može da podeli prostor pretrage na manje intervale pogodne za paralelno izvršavanje, koristeći više procesora za istovremenu procenu funkcije uslova u različitim regionima prostora. Ova paralelizacija značajno smanjuje vreme obrade, čineći algoritam skalabilnim i praktičnim za velike probleme.

Na slikama 4.41 i 4.42 je prikazan rezultat rada algoritma za `1D` i `2D` slučaj. Nepoznate granice se lako uočavaju jer je oko njih izvršena preciznija pretraga.



Slika 4.41 - Pretraga 1D prostora



Slika 4.42 - Pretraga 2D prostora

U početnim fazama, algoritam prikazuje veličine početnih i najmanjih elemenata nakon svih deoba, pružajući uvid u postignuti nivo granularnosti. Izračunavanjem dimenzija ovih elemenata, korisnici mogu razumeti koliko se prostor fino deli, što može pomoći pri donošenju odluka o dubini rekurzije i faktorima podele.

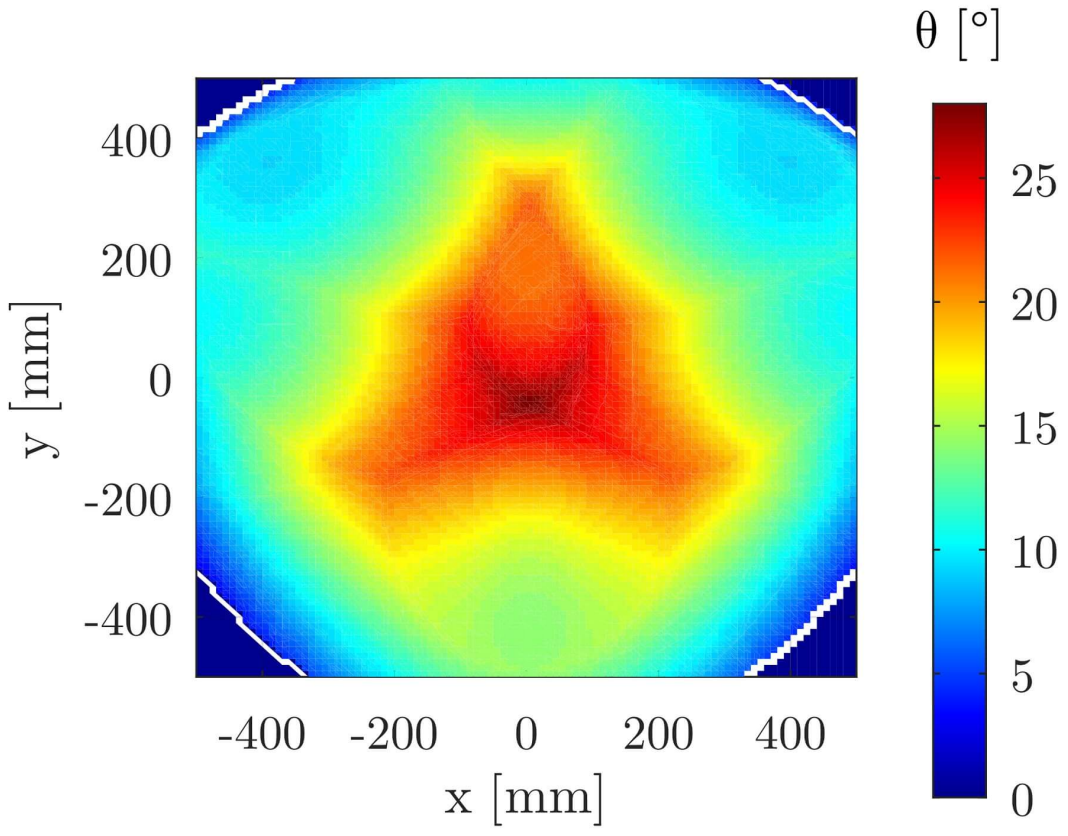
Rekurzivna priroda algoritma omogućava mu da adaptivno sužava prostor za pretragu, pružajući detaljne informacije o oblastima od interesa bez iscrpnog evaluiranja celog prostora.

Algoritam `SpaceSearch` nudi robusan i efikasan pristup za istraživanje *N-dimenzionalnih* prostora u cilju identifikacije oblasti koje zadovoljavaju specifične uslove. Njegova strategija rekurzivne podele, u kombinaciji sa sposobnošću za paralelnu obradu, čini ga moćnim alatom za aplikacije koje zahtevaju preciznu identifikaciju granica unutar složenih prostora. Implementacija algoritma osigurava da se računarski resursi efikasno koriste, pružajući balans između performansi i tačnosti koji je ključan za napredne računarske zadatke.

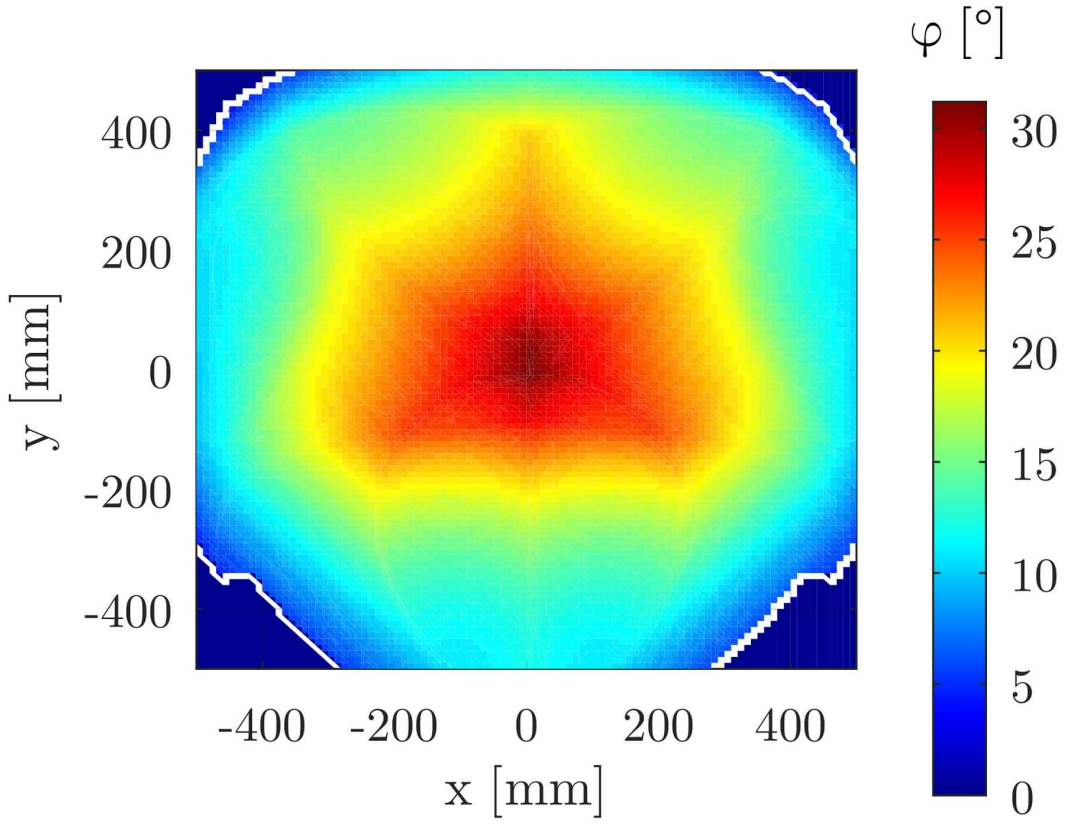
4.9.2 Maksimalni ostvarivi parametri

U odnosu na metod inkrementalne promene parametara sa konstantnim korakom sve dok se ne odredi maksimalna ostvariva vrednost, implementiran je značajno efikasniji metod koji koristi prethodno objašnjen algoritam pod nazivom `SpaceSearch`. Pretraga se vrši za sve parametre osim z u `2D` prostoru zajedno sa z dok se za z radi proračun u `1D` prostoru. Što znači da se kao rezultat dobija maksimalna vrednost parametra kao i z koordinata na kojoj je moguće ostvariti ovu vrednost. Za ovaj postupak se koriste metode `findMaxParameter()` i `getAllMaxParameters()` klase `MP_6RSS_SOLVER`.

Na sledećim slikama (slike 4.43 i 4.44) su prikazani primeri promene indikatora maksimalnih uglova propinjanja i valjanja u zavisnosti od koordinata `x` i `y`.



Slika 4.43 - Promena indikatora maksimalnog ugla propinjanja u ravni



Slika 4.44 - Promena indikatora maksimalnog ugla valjanja u ravni

4.9.3 Određivanje granične površi oblaka tačaka

Određivanje granica oblaka (skupa) tačaka u tri dimenzije je izazovan problem koji zahteva efikasne algoritme i optimizovanu implementaciju zbog velikog broja tačaka. Dostupna `C++` biblioteka *The Computational Geometry Algorithms Library* (`CGAL`) pruža niz algoritama za računsko-geometrijske probleme, uključujući *alfa oblike* (*Alpha Shapes*), *Delunijevu triangulaciju* i druge koji se mogu iskoristiti za određivanje granične površi oblaka tačaka.

Granična površ se određuje korišćenjem metode pod nazivom *Alfa oblici* (*eng. Alpha Shapes*) i to osnovnom verzijom ove metode koja se bazira na *Deloneovoj triangulaciji* tačaka [134]. Nakon triangulacije se dobija konveksna površ koja obuhvata sve tačke oblaka. Cilj je odrediti poluprečnik sfere (parametar α po kom metod nosi ime) koji daje najpribližniji konkavan oblik oblaka stvarnom obliku. U okviru ove metode se sferama bombarduje model dobijen nakon triangulacije i uklanjaju stranice kroz koje sfera može da prođe.

Funkcija `boundary3D()`, koja implementira prethodno opisani postupak, je integrisana u program `JSLAB`. Ona je ovde izdvojena i prikazana zbog značaja za ovaj rad. Takođe, on je značajna za različite primene u računskoj geometriji, obradi slike i 3D modeliranju. Za ovaj rad je izuzetno značajna jer od oblaka tačaka omogućava dobijanje granica radnog prostora i izračunavanje zapremine dobijenog tela.

Argumenti funkcije `boundary3D()` su skup tačaka i opcioni argument `shrink`, koji kontroliše stepen kontrakcije granice. Funkcija vraća niz trouglova (definisanih čvorovima) koji predstavljaju površinu granične površine i zapreminu generisane oblasti.

Implementacija funkcije `boundary3D()` zasniva se na nekoliko ključnih algoritama:

- *Alfa oblici*

  Alfa oblici su generalizacija koncepta konveksnog omotača koja omogućava kontrolu

detalja granice skupa tačaka. Parametar α određuje koliko "čvrsto" granica prati oblik tačaka:

  - Za vrlo male vrednosti α, granica će obuhvatiti samo guste regione tačaka.
  - Za velike vrednosti α, granica će biti blizu konveksnog omotača skupa.

U implementaciji se koristi `CGAL::Alpha_shape_3` klasa za generisanje *3D alfa oblika*.

Kritična vrednost α_{crit} je vrednost za koju *alfa oblik* ima određena svojstva, kao što je postojanje jednog regiona. Funkcija `getCriticalAlpha('one-region')` vraća ovu vrednost, koja se zatim koristi za prilagođavanje parametra α.

Spektar α vrednosti je niz svih mogućih α za dati skup tačaka koje rezultuju promenom topologije *alfa oblika*. Funkcija `getAlphaSpectrum()` vraća ovaj niz, koji se koristi za izbor odgovarajuće vrednosti α u skladu sa željenim stepenom kontrakcije (`shrink` parametar).

- *Delunijeva triangulacija*

  Delunijeva triangulacija u trodimenzionalnom prostoru predstavlja metodu razlaganja skupa tačaka u prostoru na tetraedre, tako da nijedna tačka iz tog skupa ne leži unutar opisane sfere bilo kog tetraedra u triangulaciji. Ova karakteristika osigurava da su elementi mreže što bliže regularnim oblicima, što je ključno za precizne numeričke proračune. *Delunijeva triangulacija* u 3D prostoru se veoma često koristi u oblastima kao što su računska geometrija, kompjuterska grafika i inženjerske simulacije, gde je potrebno efikasno i tačno modeliranje trodimenzionalnih objekata i prostora.

  Delunijeva triangulacija se koristi kao osnov za formiranje *alfa oblika*. U `CGAL`-u, ovo je podržano kroz `CGAL::Delaunay_triangulation_3` klasu.

- *Pojednostavljenje oblika*

  Nakon generisanja početne granice, može biti potrebno pojednostaviti mrežu kako bi se smanjila kompleksnost. Za to se može koristiti neka `CGAL`-ova funkcija za pojednostavljenje površinskih mreža, kao što je `edge_collapse` i relevantni prediktori zaustavljanja.

Implementacija funkcije `boundary3D()` se bazira na novoj klasi `AlphaShape3D` koja obuhvata sledeće metode:

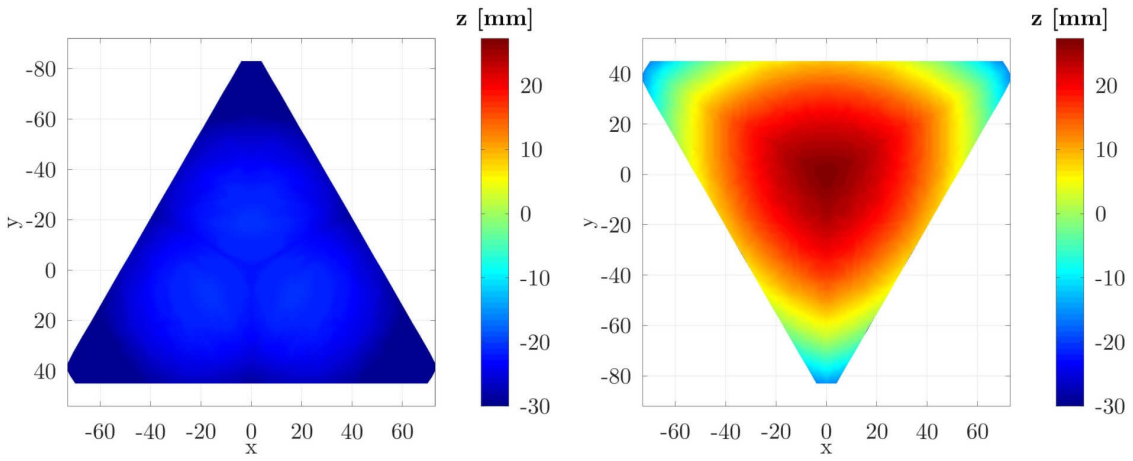
- `newShape()`: Kreira novu *Alpha Shape* strukturu na osnovu ulaznih tačaka.
- `setAlpha()`: Postavlja vrednost α i regeneriše površinsku mrežu.
- `getVolume()` i `getSurfaceArea()`: Vraćaju zapreminu i površinu generisane granice.
- `getBoundaryFacets()`: Vraća niz trouglova koji čine granicu, što je neophodno za prikaz.
- `getSimplifiedShape()`: Pojednostavljuje površinsku mrežu radi smanjenja kompleksnosti.

U okviru priloga L su dati listinzi kodova kojima je implementirana ova funkcija. Kombinovanjem programskih jezika `C++` i `JavaScript` je dobijeno mnogo u pogledu preciznosti i fleksibilnosti.

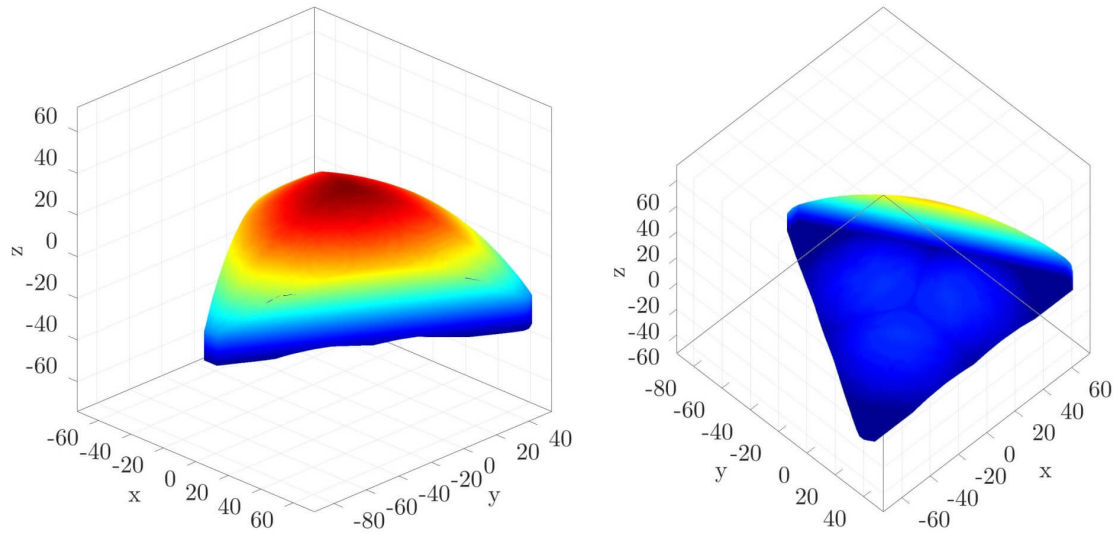
4.9.4 Radni prostor mehanizma

Radni prostor prikazanog mehanizma je u `6D` prostoru i njega je moguće približno odrediti prikazanim algoritmom pod nazivom `SpaceSearch`. Za potrebe proračuna optimalnih parametara je reprezentativno koristiti radni prostor mehanizma sa konstantnom nultom orijentacijom (stepeni slobode φ, θ i ψ su nulti), u ovom slučaju se radi o 3D prostoru koji je moguće i grafički prikazati kao trodimenzionalnu figuru. Za određivanje radnog prostora sa nultom orijentacijom se koristi skripta `findWorkspaceParallel.jsl` data u prilogu K. U ovoj skripti se koristi paralelizacija proračuna kako bi se smanjilo potrebno vreme izvršavanja i iskoristili svi dostupni resursi procesora računara.

U poglavlju 4.9.3 je detaljno opisano kako je moguće od oblaka tačaka dobiti njihovu graničnu površ, u ovom slučaju radni prostor mehanizma. Na sledećim graficima (slike 4.45 i 4.46) je prikazan primer spoljne površi radnog prostora.



Slika 4.45 - Spoljna površ radnog prostora odozgo (levo) i odozdo (desno)

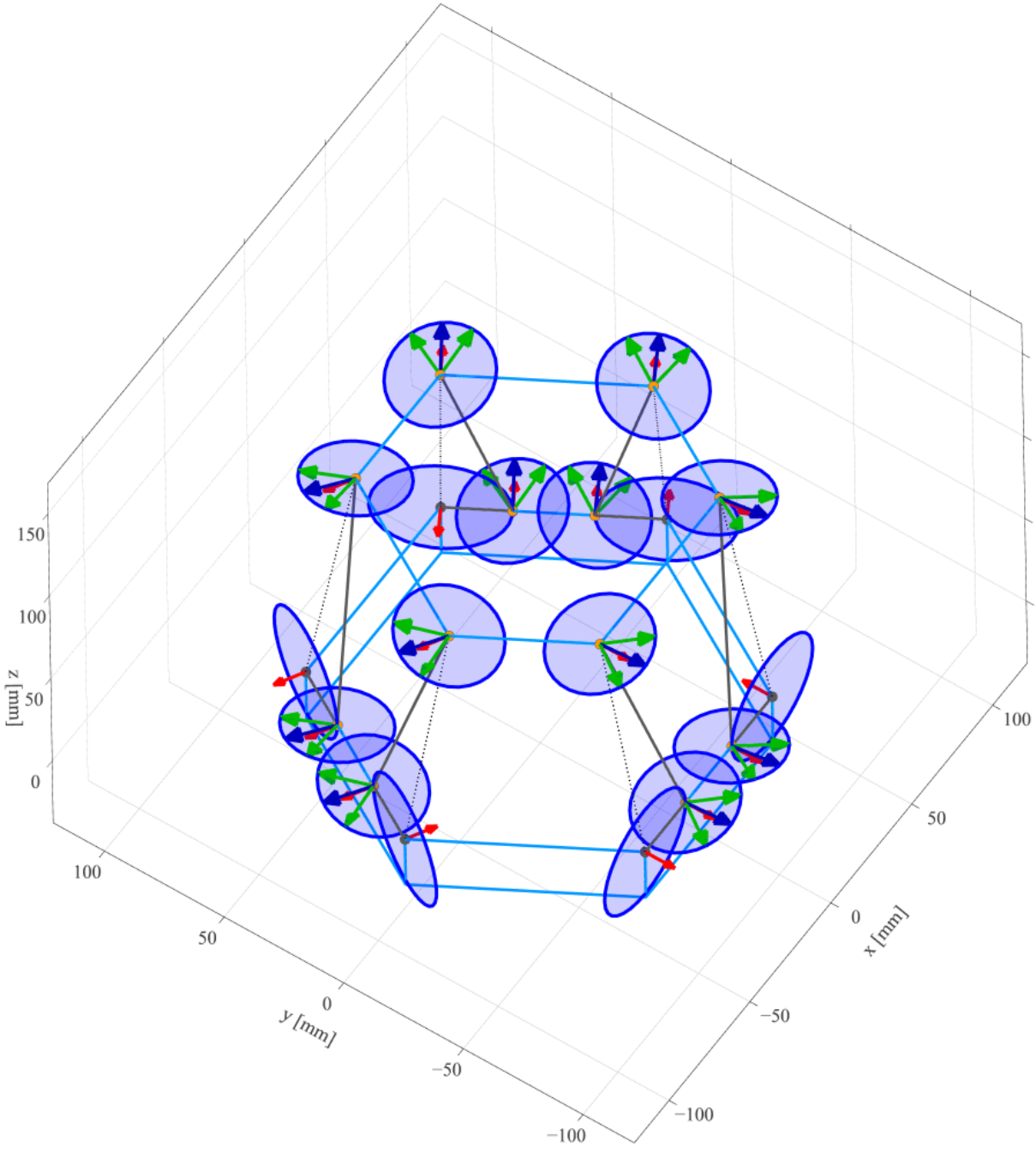


Slika 4.46 - Izometrijski prikazi spoljne površi radnog prostora

Nakon definisanja spoljne (granične) površi određujuju se i njena površina ali i zapremina radnog prostora koja takođe može biti jedan od indikatora performansi mehanizma. Zahvaljujući korišćenom algoritmu ovaj metod je nekoliko desetina puta brži u odnosu na prostu diskretizaciju prostora i proveru u svakoj tački sa zadržavanjem mogućnosti za paralelizaciju proračuna.

4.10 Fiksni ugao ugradnje sfernih zglobova gornjih poluga

Pored 11 geometrijskih parametara definisanih u potpoglavlju 4.2, moguće je uvesti još jedan parametar i to μ_{ul} kao ugao ugradnje zglobova gornje poluge. Ovaj parametar se određuje kao apsolutni ugao za koji je jedan sferni zglob (gornji ili donji) prilikom ugradnje zakrenut u odnosu na drugi. Korišćenje fiksnog ugla ugradnje značajno pojednostavljuje samu konstrukciju gornje poluge jer više nije potrebno obezbediti slobodnu rotaciju jednog kraja poluge u odnosu na drugi ali je zato neophodno izabrati optimalan ugao koji obezbeđuje dovoljnu slobodu kretanja mehanizmu. Na slici 4.47 su zelenom bojom prikazani vektori koji predstavljaju limite za orijentaciju sfernih zglobova gornjih poluga. Unutar granica sfernih zglobova, gornje poluge mogu slobodno da se rotiraju oko svoje podužne ose.



Slika 4.47 - Karakteristični vektori za određivanje zasićenja sfernih zglobova

Provera da li dolazi do zasićenja slobode kretanja u sfernim zglobovima podrazumeva sledeći postupak:

- Projektovati vektore $\boldsymbol{i_{jpb}}$ i $\boldsymbol{i_{ll}}$ na ravni normalne u odnosu na vektore $\boldsymbol{i_{ul}}$ (ravni su na slici prikazane transparentnim diskovima).

$$\boldsymbol{i_{jpb,p}} = \boldsymbol{i_{jpb}} - \Big((\boldsymbol{i_{jpb}} \cdot \boldsymbol{i_{ul}})\, \boldsymbol{i_{ul}} \Big) \tag{4.286}$$

$$\boldsymbol{i_{ll,p}} = \boldsymbol{i_{ll}} - \Big((\boldsymbol{i_{ll}} \cdot \boldsymbol{i_{ul}})\, \boldsymbol{i_{ul}} \Big) \tag{4.287}$$

- Normalizovati projekcije (dobijeni vektori plave boje na slici).

$$\boldsymbol{i_{jpb,p}} = \frac{\boldsymbol{i_{jpb,p}}}{\|\boldsymbol{i_{jpb,p}}\|} \quad \boldsymbol{i_{ll,p}} = \frac{\boldsymbol{i_{ll,p}}}{\|\boldsymbol{i_{ll,p}}\|} \tag{4.288}$$

- Odrediti uglove $\boldsymbol{\lambda_{pj}}$ i $\boldsymbol{\lambda_{bj}}$ između $\boldsymbol{i_{ul}}$ i vektora $\boldsymbol{i_{jpb}}$ i $\boldsymbol{i_{ll}}$.

$$\boldsymbol{\lambda_{pj}} = \Big| \arccos\Big(\boldsymbol{i_{ul}} \cdot \boldsymbol{i_{jpb}} \Big) - 90^\circ \Big| \tag{4.289}$$

$$\boldsymbol{\lambda_{bj}} = \Big| \arccos\Big(\boldsymbol{i_{ul}} \cdot \boldsymbol{i_{ll}} \Big) - 90^\circ \Big| \tag{4.290}$$

- Odrediti maksimalne dozvoljene uglove $\boldsymbol{\mu_{pj,max}}$ i $\boldsymbol{\mu_{bj,max}}$ na osnovu λ_{max} i $\boldsymbol{\lambda_{pj}}$ i $\boldsymbol{\lambda_{bj}}$ (na slici su zelenom bojom prikazani vektori definisani ovim uglovima).

$$\boldsymbol{\mu_{pj,max}} = \arccos\left(\frac{\cos(\lambda_{max})}{\cos(\boldsymbol{\lambda_{pj}})} \right) \tag{4.291}$$

$$\boldsymbol{\mu_{bj,max}} = \arccos\left(\frac{\cos(\lambda_{max})}{\cos(\boldsymbol{\lambda_{bj}})} \right) \tag{4.292}$$

- Na osnovu sledećih jednačina potom odrediti optimalne uglove $\boldsymbol{\mu_{ul,opt}}$ i zatim minimalne $\boldsymbol{\mu_{ul,min}}$ i maksimalne $\boldsymbol{\mu_{ul,max}}$ granice.

$$\boldsymbol{\mu_{ul,opt}} = \boldsymbol{\sigma_l} \cdot \operatorname{atan2}\left(\sum \boldsymbol{i_{jpb,p}} \times \boldsymbol{i_{ll,p}}, \sum \boldsymbol{i_{jpb,p}} \cdot \boldsymbol{i_{ll,p}} \right) \tag{4.293}$$

$$\boldsymbol{\mu_{ul,min}} = \boldsymbol{\mu_{ul,opt}} - \boldsymbol{\mu_{pj,max}} - \boldsymbol{\mu_{bj,max}} \tag{4.294}$$

$$\boldsymbol{\mu_{ul,max}} = \boldsymbol{\mu_{ul,opt}} + \boldsymbol{\mu_{pj,max}} + \boldsymbol{\mu_{bj,max}} \tag{4.295}$$

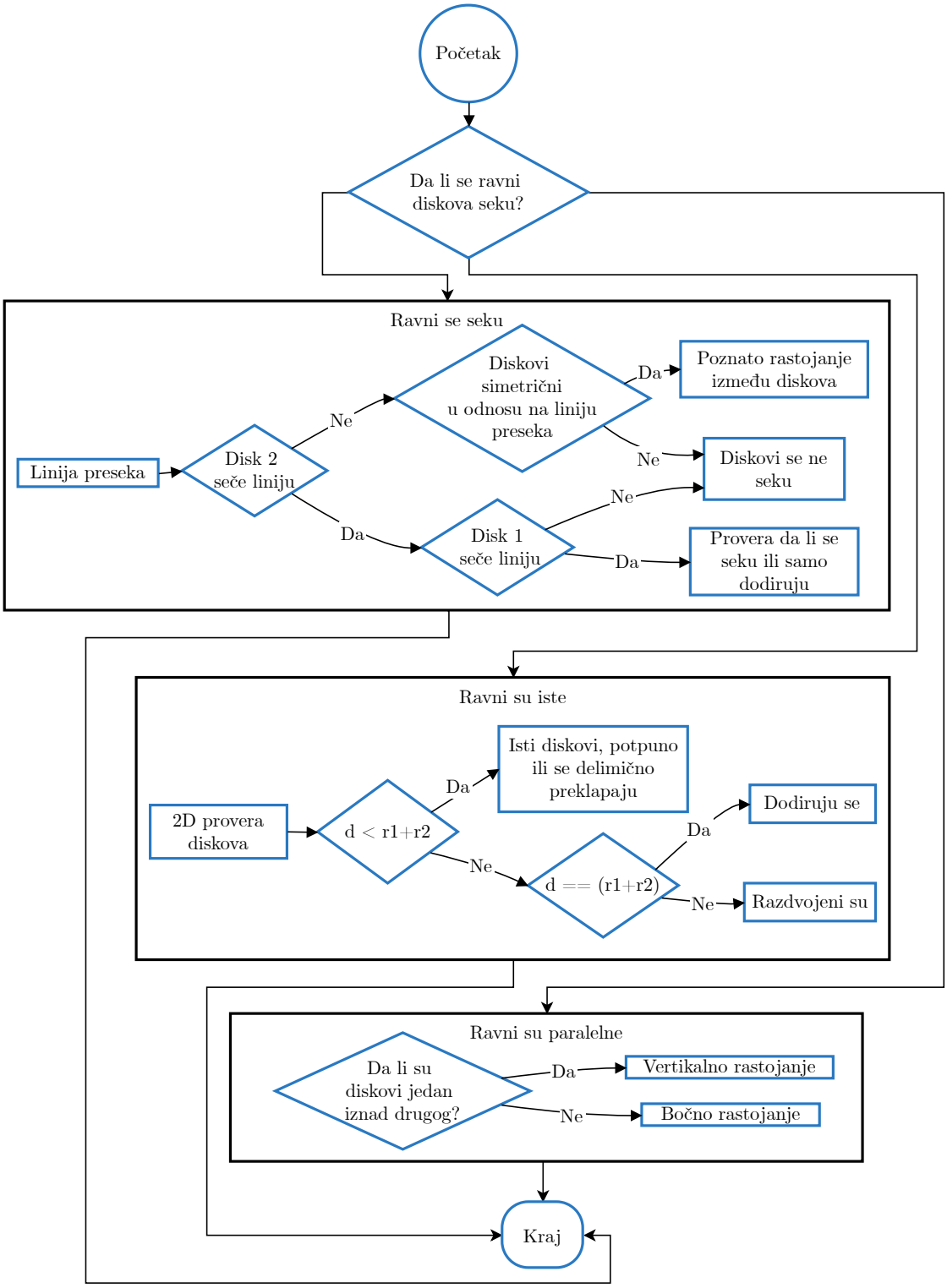
Ograničenje za mehanizam je potom moguće definisati sa:

$$\boldsymbol{\mu_{ul,min}} < \mu_{ul} \ \wedge \ \boldsymbol{\mu_{ul,max}} > \mu_{ul} \tag{4.296}$$

U prilogu N je data metoda `checkJointsFixed()` (modifikovana metoda `checkJoints()`) ali ovog puta uzima u obzir i fiksni ugao ugradnje sfernih zglobova gornjih poluga.

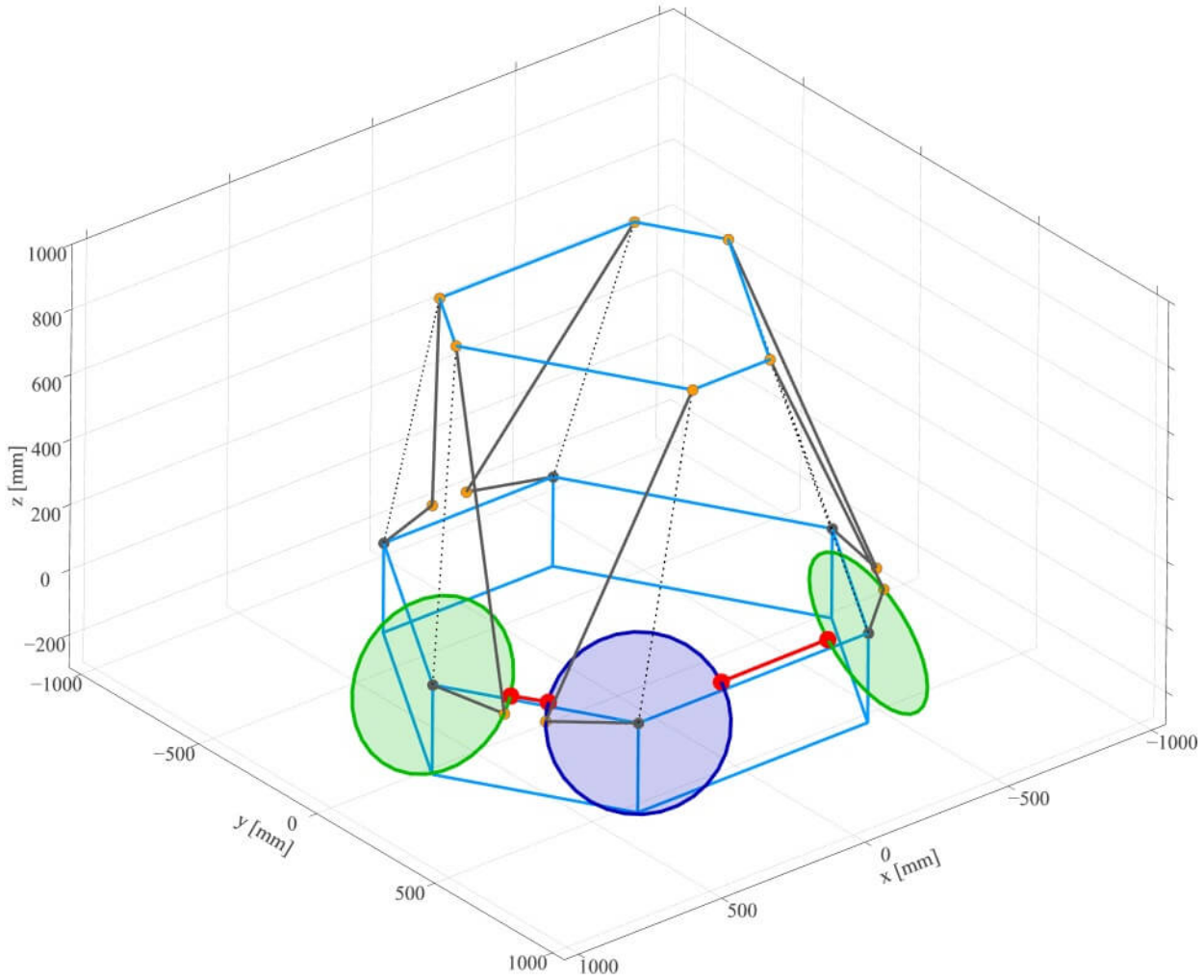
4.11 Analitički proračun minimalnog rastojanja između donjih poluga

Prilikom projektovanja mehanizma je potrebno osigurati da neće doći do sudaranja između poluga mehanizma. Analizira se međusobni položaj diskova koje definiše rotacija donjih poluga oko ose obrtanja aktuatora. Proverava se da li se seku, tangencijalno dodiruju ili se uopšte ne seku, da li su međusobno paralelni ili koliko je rastojanje između njih.



Slika 4.48 - Algoritam funkcije disksDistance3D

U prilogu M je data skripta `leversMinDistance.jsl` u kojoj je definisana i funkcija `disksDistance3D()` koja na osnovu algoritma prikazanog na slici 4.48 određuje minimalno rastojanje između dva diska u trodimenzionalnom prostoru. Konačan rezultat, kada se ova funkcija primeni na analizirani mehanizam, je prikazan na slici 4.49.



Slika 4.49 - Rezultat proračuna minimalnog rastojanja između donjih poluga

Argumenti funkcije `disksDistance3D` su tačke $\boldsymbol{O_1}$ i $\boldsymbol{O_2}$ kao središta diskova, zatim vektori $\boldsymbol{n_1}$ i $\boldsymbol{n_2}$ kojim je definisana orijentacija diskova u odnosu na središta i prečnici diskova r_1 i r_2. Prvo se analizira međusobni odnos ravni u kojima leže diskovi. Ravni diskova su definisane sa $\boldsymbol{O_1}$, $\boldsymbol{O_2}$, $\boldsymbol{n_1}$ i $\boldsymbol{n_2}$. Određuje se da li se ravni seku, da li su zapravo ravni iste ili su paralelne. Ova provera se vrši na osnovu vektorskog proizvoda normalnih vektora ravni.

$$\|\boldsymbol{n}_1 \times \boldsymbol{n}_2\| = \|\boldsymbol{V}\| \begin{cases} = 0, & \text{ravni su ili paralelne ili iste,} \\ > 0, & \text{ravni seku i obrazuju pravu,} \end{cases} \tag{4.297}$$

Ravni su iste ako važi da je:

$$\left\|\boldsymbol{n}_1 \cdot (\boldsymbol{P}_1 - \boldsymbol{P}_2)\right\| = 0. \tag{4.298}$$

Ukoliko postoji presek dve ravni, onda je jedinični vektor te prave:

$$\boldsymbol{i} = \frac{\boldsymbol{V}}{\|\boldsymbol{V}\|}. \tag{4.299}$$

Da bi prava preseka bila potpuno definisana, dovoljno je još odrediti i jednu tačku kroz koju ova prava prolazi. To je moguće uraditi rešavanjem sledećeg linearnog sistema, a na osnovu toga što tačka zapravo pripada i jednoj i drugoj ravni:

$$\boldsymbol{n}_1 \cdot \boldsymbol{X} = \boldsymbol{n}_1 \cdot \boldsymbol{P}_1 \tag{4.300}$$

$$\boldsymbol{n}_2 \cdot \boldsymbol{X} = \boldsymbol{n}_2 \cdot \boldsymbol{P}_2 \tag{4.301}$$

Ova procedura je implementirana u okviru `JSLAB` funkcije `planesIntersection()`.

Za određivanje položaja diska u odnosu na pravu preseka je potrebno implementirati i proračun rastojanja tačke (središta diska) od prave. Rastojanje tačke $\boldsymbol{P}$ od prave definisane sa tačkom $\boldsymbol{A}$ i vektorom $\boldsymbol{i}$ se određuje tako što se najpre odredi projekcija tačke na pravu i to na osnovu:

$$\boldsymbol{P}_1 = \boldsymbol{A} + \left((\boldsymbol{P} - \boldsymbol{A}) \cdot \boldsymbol{i}\right) \boldsymbol{i} \tag{4.302}$$

Tada je rastojanje tačke P do prave moguće izračunati pomoću sledećeg izraza:

$$d = \|\boldsymbol{P} - \boldsymbol{P}_1\|. \tag{4.303}$$

Ova procedura je takođe implementirana u okviru `JSLAB`-a i to funkcijom `pointLineDistance()`.

Na osnovu rastojanja d je moguće analizirati da li postoji presek ove prave i kružnice diska.

$$d = \|\boldsymbol{O} - \boldsymbol{P}_1\| \begin{cases} < r, & \text{dve tačke preseka,} \\ = r & \text{dodiruju se u tački,} \\ > r, & \text{nema preseka.} \end{cases} \tag{4.304}$$

Ako važi da je $d < r$, dve tačke preseka na pravoj su:

$$\boldsymbol{P}_1 \pm h\,\boldsymbol{i} \tag{4.305}$$

gde je

$$h = \sqrt{r^2 - d^2}. \tag{4.306}$$

Na osnovu toga kako je definisana geometrija mehanizma se može zaključiti da se ravni diskova ili seku ili su iste. Kada se ravni diskova seku tada su diskovi sigurno simetrično postavljeni u odnosu na pravu preseka njihovih ravni. U ovom slučaju su projekcije središta diskova $\boldsymbol{O_1}$ i $\boldsymbol{O_2}$ tačke $\boldsymbol{P_1}$ i $\boldsymbol{P_2}$ zapravo jednake ($\boldsymbol{P_1} = \boldsymbol{P_2}$). Ukoliko diskovi ne seku pravu preseka ravni, rastojanje između diskova se određuje na osnovu toga što je:

$$\boldsymbol{P_{c1}} = \boldsymbol{O_1} + \frac{\boldsymbol{P_1} - \boldsymbol{O_1}}{\|\boldsymbol{P_1} - \boldsymbol{O_1}\|} r_1 \tag{4.307}$$

$$\boldsymbol{P_{c2}} = \boldsymbol{O_2} + \frac{\boldsymbol{P_2} - \boldsymbol{O_2}}{\|\boldsymbol{P_2} - \boldsymbol{O_2}\|} r_2 \tag{4.308}$$

$$L_{min} = \|\boldsymbol{P_{c1}} - \boldsymbol{P_{c2}}\| \tag{4.309}$$

Kada su diskovi u istoj ravni onda je određivanje rastojanja između diskova značajno jednostavnije i tada je:

$$L_{min} = \|\boldsymbol{O_2} - \boldsymbol{O_1}\| - (r_1 + r_2) \tag{4.310}$$

4.12 Analitičko dimenzionisanje poluga

Prethodno je u poglavlju 4.7 prikazan proračun elastičnih deformacija na osnovu metode konačnih elemenata. U ovom poglavlju se prikazuje analitičko dimenzionisanje poluga Stjuartove platforme sa obrtnim aktuatorima. Pošto se u tačkama A_i i P_i nalaze sferni zglobovi i ukoliko se uticaj trenja zanemari, može se zaključiti da se na gornju polugu prenosi samo aksijalna sila.

Na osnovu hipoteze najvećeg normalnog napona kao jedne od hipoteza o slomu materijala, za ravno stanje napona (za donje poluge) je uporedni (ekvivalentni) napon[135]:

$$\sigma_e = \frac{\sigma}{2} \pm \frac{1}{2}\sqrt{\sigma^2 + 4\tau^2} \tag{4.311}$$

Za slučaj donje poluge se normalni i tangencijalni napon mogu odrediti na osnovu:

$$\sigma = \frac{N}{A} \pm \frac{M_x}{W_x} = \frac{N}{A} \pm \frac{a \cdot T}{W_x} \tag{4.312}$$

$$\tau = \frac{T \cdot S_x}{I_x \cdot t} = \frac{T \cdot S_x}{I_x \cdot (D - d)} \tag{4.313}$$

Gornja poluga je opterećena samo na pritisak pa se zbog toga proverava da li će doći do izvijanja. Prvo se računa vitkost poluge sa:

$$\lambda = \frac{L}{i_2} = \frac{s}{i_2} \tag{4.314}$$

Granična vitkost je definisana sa:

$$\lambda_p = \pi\sqrt{\frac{E}{\sigma_{max}}} \tag{4.315}$$

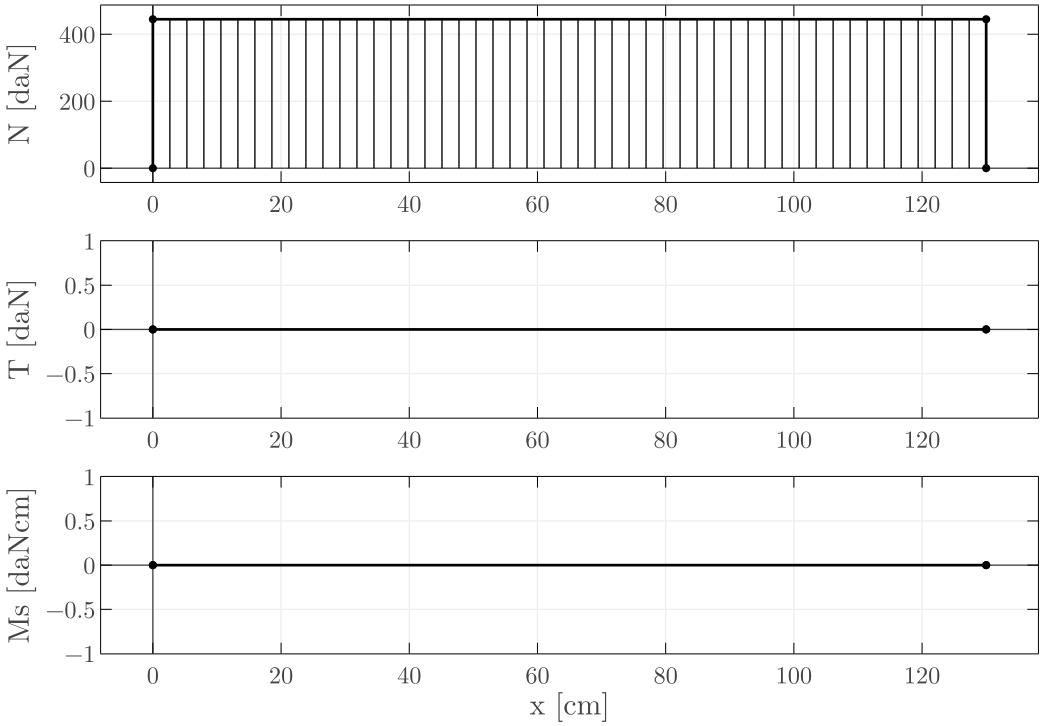
Ukoliko je $\lambda > \lambda_p$, može se koristiti Ojlerova formula za kritični napon u oblasti elastičnog izvijanja i to:

$$\sigma_{kr} = \frac{\pi^2 E}{\lambda^2} \tag{4.316}$$

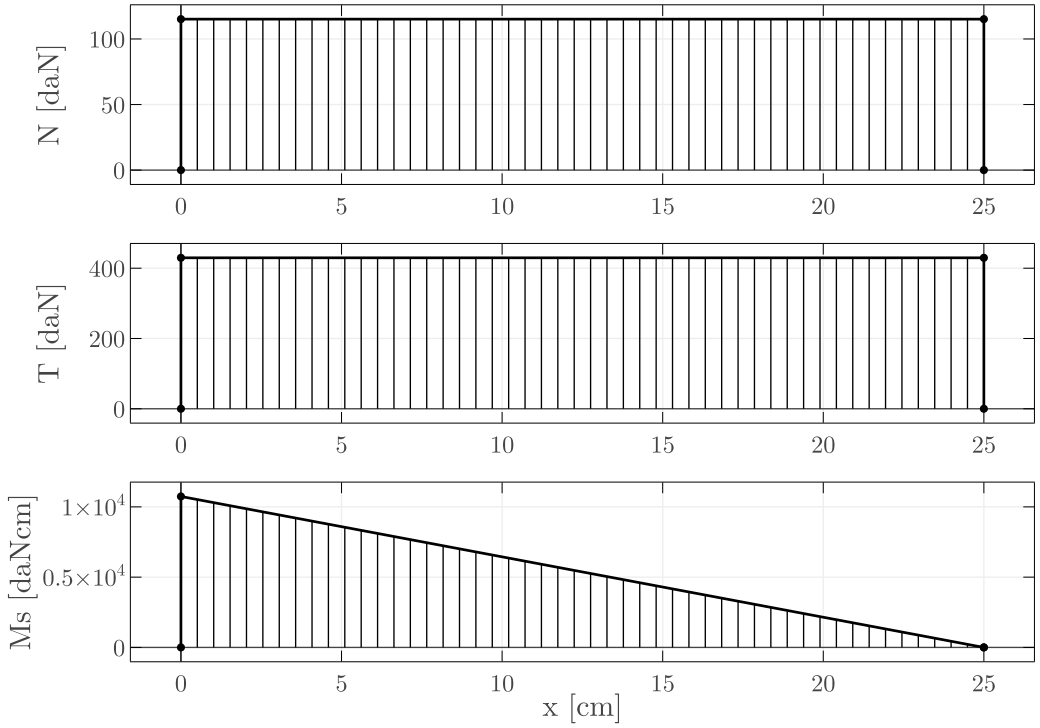
Tada se izračunava normalni napon $\sigma = \frac{N}{A}$ i upoređuje sa σ_{kr}.

U prilogu O je data skripta `analyticStructuralAnalysis.jsl` kojom se na osnovu prethodnih jednačina i sa usvojenim podacima za poprečni presek (usvojene su cevi prstenastog poprečnog preseka) proverava čvrstoća poluga. Za materijal je usvojen običan konstrukcioni čelik opšte namene. Ovaj proračun može se takođe koristiti u procesu proračuna optimalnih geometrijskih parametara mehanizma.

Na slikama 4.50 i 4.51 su prikazani primeri statičkih dijagrama gornjih i donjih poluga, prikazane su promene aksijalne sile, transverzalne sile i momenta savijanja duž poluga. Ovi dijagrami su nacrtani pomoću JSLAB funkcije `plotBeamDiagrams()`.



Slika 4.50 - Statički dijagrami gornje poluge



Slika 4.51 - Statički dijagrami donje poluge

5

Proračun optimalne geometrije mehanizma simulatora leta za sistem

"Prema delu Čarlsa Darvina pod naslovom - O poreklu vrsta, ne opstaju najjači pripadnici jedne vrste, niti najinteligentniji, već oni koji se najbolje prilagođavaju promenama."

– Leon C. Megginson[1]

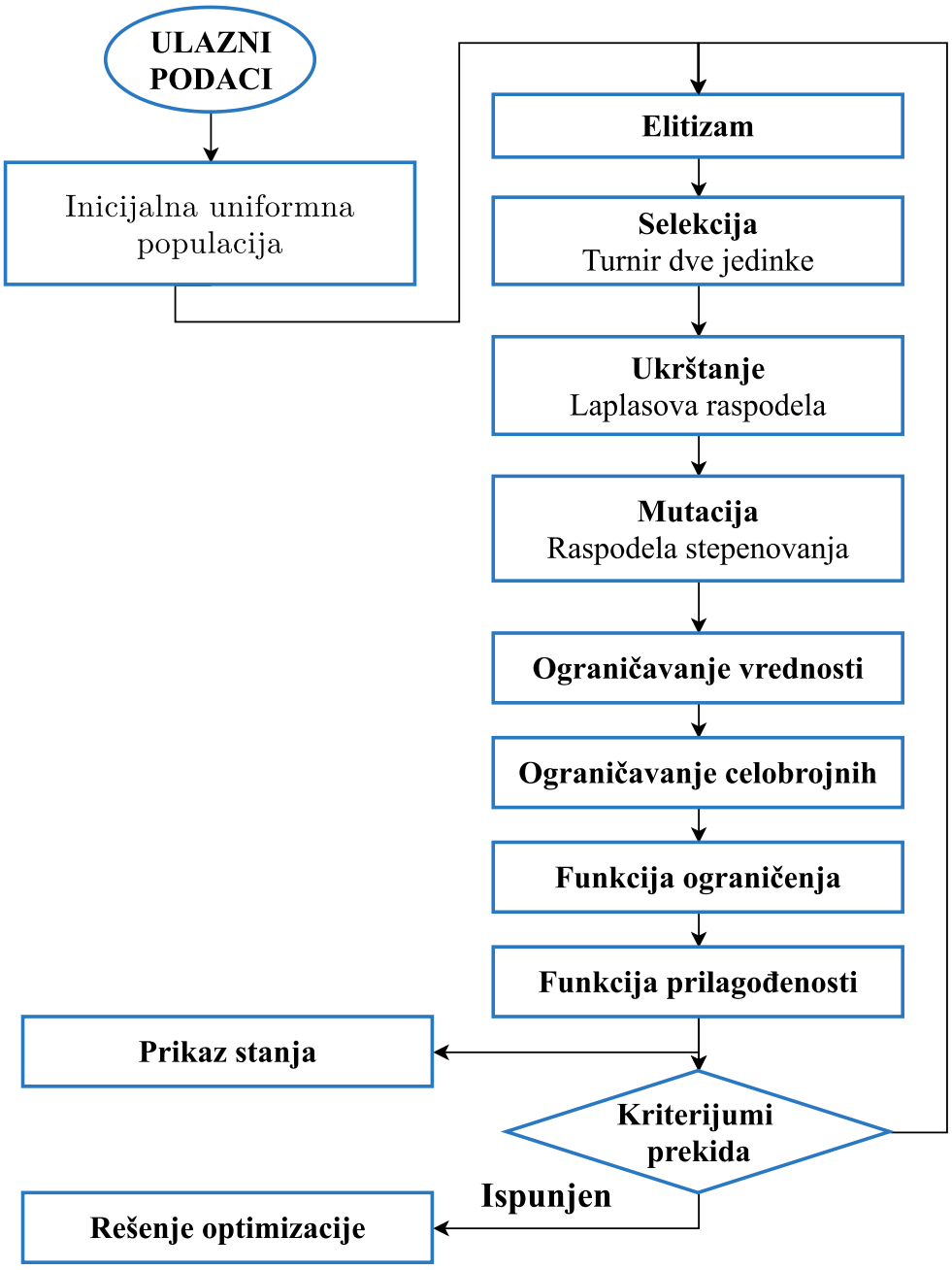
Performanse sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi direktno zavise od performansi simulatora leta. Zbog toga je veoma bitno izvršiti proračun optimalne geometrije mehanizma simulatora leta za ovaj sistem. Zbog velikog broja kombinacija vrednosti geometrijskih parametara kojima se jednoznačno definiše mehanizam, neophodno je uposliti neki od algoritama za pronalaženje optimalnih vrednosti u relativno kratkom vremenskom periodu. Za probleme kao što je ovaj, algoritmi su često numerički i ne daju rešenje u zatvorenoj formi. Ovi algoritmi su tradicionalno iterativni, a ograničeni problemi minimizacije se mogu rešavati minimiziranjem funkcije prilagođenosti sukcesivnim približavanjem u nizu proračuna [136]. Baš na ovaj način funkcioniše i *genetski algoritam* koji se koristi u okviru ove disertacije. Ovaj algoritam pripada klasi evolucionih algoritama i do optimalnog rešenja se dolazi oponašanjem procesa prirodne selekcije [137].

Cilj proračuna definiše se implementacijom funkcije prilagođenosti, čiji minimum predstavlja optimalno rešenje. Ova funkcija se naziva još i funkcija cilja ili funkcija troška. Ovaj algoritam je izabran za ovaj problem jer postoji veliki broj lokalnih minimuma i teško je pronaći globalni minimum. Postoji mnogo indikatora performansi ovog mehanizma koje je moguće uzeti u obzir prilikom izračunavanja funkcije prilagođenosti, kao što su maksimalni uglovi valjanja i propinjanja, zapremina radnog prostora, potrebna sloboda u zglobovima, spretnost mehanizma, singulariteti i fizička interferencija između segmenata mehanizma.

5.1 Analiza i implementacija algoritma

Genetski Algoritmi (GA) predstavljaju jednu od tehnika optimizacije i pretrage zasnovane na genetici i prirodnoj selekciji po uzoru na ideju *Darvinove teorije o poreklu vrsta i prirodnoj evoluciji.*

[1]Leon C. Meginson (eng. *Leon C. Megginson*, 26. jul 1921 - 22. februar 2010) je bio američki profesor menadžmenta i marketinga na Državnom fakultetu Luizijane (*eng.* Louisiana State University, LSU).



Slika 5.1 - Dijagram toka genetskog algoritma

Na prethodnoj slici (slika 5.1) je prikazan uprošćen dijagram toka programa za *realno kodirani mešovito celobrojni genetski algoritam sa ograničenjima* definisan u [67]. Baš ovaj tip genetskog algoritma je u potpunosti implementiran u programskom paketu `JSLAB` i dat u prilogu P. Algoritam je implementiran u okviru klase `OPTIM_RCMIGA` jer je to najprikladnije za kasniju eksploataciju.

Genetski algoritmi se primenjuju i daju dobre rezultate u području obučavanja neuronskih mreža, pri traženju najkraćeg puta, problemu trgovačkog putnika, problemu raspoređivanja procesa, određivanja parametara sistema, itd[138]. GA je heuristička[2] metoda koja oponaša veoma složeni prirodni evolucioni proces pretraživanja prostora rešenja.

[2]Heuristika je nauka o metodama i principima pronalaženja novog. Obuhvata metode i tehnike rešavanja problema, učenja i otkrivanja koji su bazirani na iskustvu. Heurističke metode se koriste da ubrzaju proces pronalaženja dovoljno dobrog rešenja u situacijama kada sprovođenje detaljnog istraživanja nije praktično.

Analogija evolucije kao prirodnog procesa i genetskog algoritma kao metode optimizacije ogleda se u procesu selekcije i genetskim operatorima. Mehanizam odabira nad nekom vrstom živih bića u procesu evolucije čine okolina i prirodni uslovi. U GA ključ selekcije je funkcija prilagođenosti, koja na odgovarajući način predstavlja problem koji se rešava. Jedno rešenje funkcije prilagođenosti je jedna jedinka, selekcijom se vrši odabir dobrih jedinki koje se prenose u sledeću populaciju, a manipulacijom genetskog materijala se stvaraju nove jedinke. Takav ciklus reprodukcije i manipulacije genetskim materijalom jedinki ponavlja se sve dok nije zadovoljen neki od kriterijuma prekida procesa evolucije.

Kao prvo, genetski algoritam je izabran zbog velikog broja promenljivih (u ovom slučaju 11 ili 12 geometrijskih parametara) i veoma nelinearne prirode sa mnogim diskontinuitetima i velikim brojem lokalnih minimuma. Ovaj tip GA je izabran jer tehnologija izrade zahteva da geometrijski parametri nisu kontinualne vrednosti već postoji neka tačnost sa kojom je moguće izraditi delove. Iz ovoga proizilazi i objašnjenje za izraz *mešovito celobrojni* u nazivu, prosto je neke promenljive moguće definisati tako da uzimaju samo celobrojne vrednosti. Pored toga, *realno kodirani* definiše da se koriste realni brojevi za predstavljanje jedinki.

Kako se ovde analizira jedan pravi inženjerski problem, neizbežno postoje ograničenja kojima se dodatno sužava oblast zadovoljavajućih rešenja. Zbog potrebe za definisanjem ograničenja se koristi *metod penalizacije* definisan u radu [68]. Ovaj metod se ogleda u evaluaciji nelinearnih uslova i sabiranjem vrednosti nelinearnih uslova sa vrednošću funkcije prilagođenosti. Treba napomenuti još samo da se vrednost za penalizaciju na osnovu evaluacije funkcije ograničenja dobija normalizacijom vrednosti funkcije ograničenja.

Kako bi se osiguralo da iz generacije u generaciju nepromenjene prelaze najbolje jedinke, implementiran je *elitizam*. I pored toga što ostaju nepromenjene ove jedinke zajedno sa ostalim učestvuju u procesu selekcije tako da mogu biti izabrane za *ukrštanje* i *mutaciju*, odnosno primenu genetskih operatora nad njima.

Da bi se uopšte započeo proračun prvo se vrši generisanje populacije za prvu generaciju na osnovu uniformne raspodele metodom `creationMixedUniform()`. Od ove inicijalne populacije se kao i za svaku sledeću generaciju traži da su vrednosti parametara u okviru definisanih granica (metod `checkBounds()`) i ukoliko postoje celobrojni parametri od njih se zahteva da imaju celobrojne vrednosti. Ovo ograničavanje na celobrojne vrednosti se vrši metodom `integerRestriction()`.

Kada postoji inicijalna populacija moguće je započeti proračun. Prvo se odvajaju elitne jedinke a zatim se vrši selekcija. U ovom slučaju se koristi selekcija bazirana na turniru između dve jedinke, implementirana u okviru metode `binaryTournamentSelection()`.

Na ovako izabrane jedinke se zatim primenjuje genetski operator ukrštanja i to ukrštanje na osnovu Laplasove raspodele. Ovo ukrštanje definisano u radu [69] je implementirano u okviru metode `laplaceMixedCrossover()`. Ova metoda ima tri parametra za podešavanje i njihove vrednosti su preuzete iz istog rada i iznose:

$$a = 0,\ b_{real} = 0.15,\ b_{int} = 0.35 \tag{5.1}$$

Zatim se primenjuje genetski operator mutacije i to mutiranje na osnovu raspodele stepenovanja. Ovo mutiranje definisano u radu [70] je implementirano u okviru metode `powerMixedMutation()`. Ova metoda ima dva parametra za podešavanje i njihove vrednosti su preuzete iz istog rada i iznose:

$$p_{real} = 10,\ p_{int} = 4 \tag{5.2}$$

Zbog toga što proračun može trajati i duži vremenski period, implementirane su funkcije za prikaz toka proračuna, tako da se na kraju evaluacije svake generacije prikazuju podaci na

osnovu kojih se može doći do zaključaka o toku samog proračuna.

Kada su implementirani svi genetski operatori i omogućena evaluacija funkcije ograničenja i funkcije prilagođenosti, koje se prosleđuju klasi koja ih zatim samo koristi bez ikakve promene, preostaje da se definišu provere kriterijuma za prekid procesa. Ovi kriterijumi su implementirani u okviru metode `stoppingCriteria()` i ima ih ukupno sedam. Ukoliko je neki od ovih kriterijuma ispunjen, dolazi do prekida proračunske petlje.

5.2 Funkcije ograničenja i prilagođenosti

Kada je u potpunosti implementiran genetski algoritam, prelazi se na formulisanje problema koji je neophodno rešiti. Inženjerski problemi u okviru projektovanja mogu imati najrazličitija ograničenja kao što je ograničenje tehnologije proizvodnje. U okviru proračuna optimalne geometrije je neophodno sva ograničenja na neki način uzeti u obzir.

Prva ograničenja parametara su zapravo granice prostora pretrage. Ova ograničenja se ne definišu u okviru funkcije ograničenja već predstavljaju parametar genetskog algoritma. Tako se za svaku promenljivu tačno definišu koje su to donje i gornje granične vrednosti.

Tabela 5.1 - Opseg vrednosti geometrijskih parametara

Parametar	Opseg vrednosti	Korak
a	od 100 mm do 400 mm	10 mm
s	od 600 mm do 1000 mm	10 mm
R_p	od 400 mm do 800 mm	10 mm
R_b	od 700 mm do 1500 mm	10 mm
γ_p	od 5° do 55°	1°
γ_b	od 5° do 55°	1°
ε	od -90° do 90°	1°
τ_{ll}	od -180° do 180°	1°
τ_{jp}	od -90° do 90°	1°
μ_{ll}	od -90° do 90°	1°
μ_{jp}	od -90° do 90°	1°

U prethodnoj tabeli (tabela 5.1) su prikazane granične vrednosti za parametre kao i korak promene vrednosti u okviru granica. U ovom konkretnom slučaju je izabrano da se sve promenljive baziraju na celobrojnim vrednostima jedinki, što ne znači da su i same vrednosti parametara celobrojne već se na osnovu mapiranja celobrojnim promenljivim dodeljuju vrednosti parametara. Na ovaj način se formira niz mogućih vrednosti umesto kontinualnog prostora.

Sledeći korak je definisanje *funkcije ograničenja*. Prvo ograničenje predstavlja minimalno rastojanje između poluga mehanizma, definisano sledećom nejednačinom:

$$\min\left(\left\|\boldsymbol{D_i D_j}\right\|_2\right) > r_{min},\ i \neq j \tag{5.3}$$

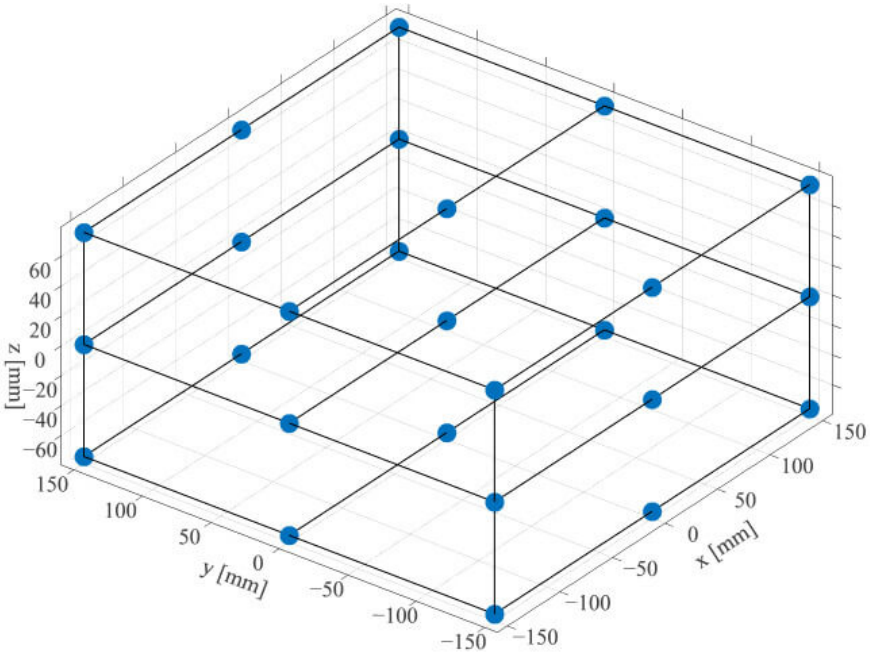
Ovo ograničenje je implementirano metodom `checkIntersectionNumerical()` klase `MP_6RSS_SOLVER`, i na taj način je osigurano da rešenja u okviru kojih dolazi do interferencije između delova mehanizma neće biti uzimana u razmatranje.

Sledeća ograničenja, kao što je ono dato u narednoj nejednačini, osiguravaju da je moguće koristiti izabrani aktuator. Proveravaju se maksimalni moment, maksimalne radijalne i aksijalne sile koje predstavljaju opterećenje reduktora.

$$|M_{mi}| < M_{max} \tag{5.4}$$

Poslednje ograničenje koje se uzima u obzir u okviru funkcije ograničenja (implementirana u okviru metode `constraints()` klase `MP_6RSS_SOLVER`) je potrebni maksimalni ugao u sfernim zglobovima. Sva ova ograničenja su prethodno analizirana u okviru disertacije.

Za potrebe formulisanja funkcije prilagođenosti su određeni indikatori performansi mehanizma. Umesto evaluacije ovih parametara u svim tačkama radnog prostora izabrano je **27 referentnih tačaka**.



Slika 5.2 - Referentne tačke za proračun

Na slici 5.2 su prikazane referentne tačke za proračun, ove tačke se dobijaju kao čvorovi mreže konstruisane na osnovu sledeća tri vektora.

$$\begin{aligned} \boldsymbol{x_g} &= \Big[-150, 0, 150\Big]^T \\ \boldsymbol{y_g} &= \Big[-150, 0, 150\Big]^T \\ \boldsymbol{z_g} &= \Big[-75, 0, 75\Big]^T \end{aligned} \tag{5.5}$$

Za izračunavanje vrednosti funkcije prilagođenosti se koriste tri indikatora performansi i to indikator spretnosti, kao i indikatori maksimalnih vrednosti uglova propinjanja i valjanja. Za integraciju navedena tri indikatora se koriste težinski koeficijenti i vrednosti zasićenja za parametre.

$$y_{of} = 100 - \frac{100}{27}\sum_{j=1}^{27}\left(b_1\eta_j + b_2\frac{\theta_{j,max} - \theta_{j,min}}{2\theta_{max}} + b_3\frac{\varphi_{j,max} - \varphi_{j,min}}{2\varphi_{max}}\right) \tag{5.6}$$

Zasićenje maksimalnih vrednosti parametara definiše granice preko kojih nije potrebno ići, to je definisano sa:

$$\theta_{j,max} = \begin{cases} \theta_{\max}, & \theta_{j,\max} > \theta_{\max}, \\ \theta_{j,\max}, & \theta_{j,\max} \leq \theta_{\max}. \end{cases} \tag{5.7}$$

$$\theta_{j,min} = \begin{cases} -\theta_{\max}, & \theta_{j,\min} < -\theta_{\max}, \\ \theta_{j,\min}, & \theta_{j,\min} \geq -\theta_{\max}. \end{cases} \tag{5.8}$$

$$\varphi_{j,max} = \begin{cases} \varphi_{\max}, & \varphi_{j,\max} > \varphi_{\max}, \\ \varphi_{j,\max}, & \varphi_{j,\max} \leq \varphi_{\max}. \end{cases} \tag{5.9}$$

$$\varphi_{j,min} = \begin{cases} -\varphi_{\max}, & \varphi_{j,\min} < -\varphi_{\max}, \\ \varphi_{j,\min}, & \varphi_{j,\min} \geq -\varphi_{\max}. \end{cases} \tag{5.10}$$

Funkcija prilagođenosti je implementirana metodom pod nazivom `fitness()` klase `MP_6RSS_SOLVER`.

Kako je već napomenuto, u okviru algoritma se koriste celobrojne vrednosti za jedinke, i to vrednosti od 0 do broja mogućih vrednosti smanjenog za jedan, za svaki parametar. Ove celobrojne vrednosti se potom mapiraju u moguće vrednosti za geometrijske parametre funkcijom kao što je metoda `mapping()` klase `MP_6RSS_SOLVER`.

5.3 Podešavanje postupka proračuna

Nakon implementacije algoritma i formulisanja problema, prelazi se na podešavanje samog postupka proračuna. U tabeli 5.2 su prikazane usvojene vrednosti za sve neophodne promenljive koje se koriste u okviru funkcije prilagođenosti i funkcije ograničenja.

Tabela 5.2 - Promenljive funkcije prilagođenosti i funkcije ograničenja

Promenljiva	**Vrednost**
Vektor težinskih koeficijenata	$\boldsymbol{b} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \end{bmatrix}^T$
Granične vrednosti uglova	$\theta_{max} = 35°$, $\varphi_{max} = 35°$, $\lambda_{max} = 35°$
Karakteristike reduktora	$x_R = 110$ mm, $a_R = 162$ mm, $b_R = 122$ mm
Spoljašnje opterećenje	$m = 150$ kg, $\boldsymbol{C_{in}} = \begin{bmatrix} 0 & 0 & 300 \end{bmatrix}^T$ mm
Raspoloživi moment i dozvoljena sila	$M_{max} = 200$ Nm, $Fr_R = 5000$ N
Rastojanje između donjih poluga	$dr = 100$ mm

U okviru tabele 5.3 su prikazane korišćene vrednosti za parametre kojim se podešava genetski algoritam. Može se zaključiti da je usvojena prilično velika populacija, što je logično imajući u vidu broj parametara i nelinearnih uslova.

Tabela 5.3 - Podešavanja genetskog algoritma

Parametar	Vrednost
Veličina populacije	5000
Broj elitnih jedinki	500
Udeo za ukrštanje	0.3

U okviru tabele 5.4 su prikazane vrednosti koje su usvojene za kriterijume prekida proračuna, prvi put kad je ispunjen neki od ovih uslova dolazi do prekida.

Tabela 5.4 - Podešavanja kriterijuma za prekid proračuna

Kriterijum	Vrednost
Maksimalni broj generacija	1000
Maksimalni broj generacija bez promene	200
Maksimalno vreme	12 h
Maksimalno vreme bez promene	1 h
Željena vrednost prilagođenosti	0
Tolerancija prilagođenosti	10^{-9}

5.4 Postupak proračuna

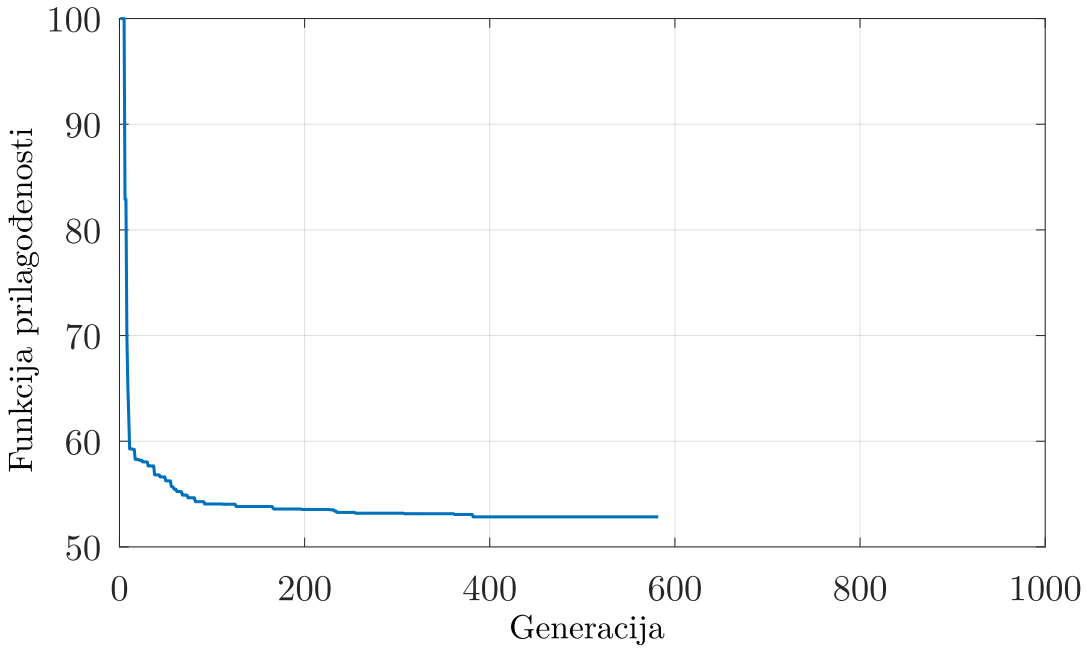
Skriptom `optimizationRCMIGA.jsl`, koja je u celosti data u okviru listinga Q.1 u prilogu Q, se vrši prethodno definisan postupak proračuna optimalne geometrije mehanizma simulatora leta, na osnovu klase `OPTIM_RCMIGA` koja je data u okviru priloga P. Nakon završetka proračuna, kada je zadovoljen jedan od uslova završetka proračuna, prikazuje se dobijeno rešenje.

Formiran je objekat `OP` koji skladišti sve potrebne parametre za proračun kao svoja svojstva. Parametri su zapravo sve promenljive prikazane u tabeli 5.2, sva podešavanja algoritma prikazana u tabeli 5.3 i svi kriterijumi prekida proračuna definisani u tabeli 5.4.

Zbog velikog broja jedinki i relativno zahtevnog proračuna u okviru funkcije prilagođenosti, implementiran je isti postupak ali paralelizovan tako da se iskoristi maksimum dostupnih računarskih resursa. Ovo je implementirano u skripti `optimizationParallelRCMIGA.jsl`, koja je u celosti data u okviru listinga Q.2 u prilogu Q, zahvaljujući ugrađenoj funkcionalnosti `JSLAB` programa da paralelizuje `for` petlju. Umesto što se uzima vrednost (jedinka) iz niza (populacija) i izračunava blok koda (evaluacija funkcije prilagođenosti) za tu vrednost sekvencijalno, niz se deli na broj dostupnih procesorskih jezgara i pokreću se zasebni procesi na svakom i izračunavaju se vrednosti za delove niza paralelno. U zavisnosti od broja dostupnih jezgara procesora, skripta `optimizationParallelRCMIGA.jsl` može se višestruko brže izvršavati nego `optimizationRCMIGA.jsl`.

Proračun se pokreće pokretanjem jedne od prethodno prikazanih skripti. Tokom postupka i nakon svake generacije se u komandnom prozoru programa `JSLAB` prikazuju trenutni rezultati i tako je moguće pratiti sam postupak i dobiti informaciju o potrebnom vremenu za izvršavanje proračuna za jednu generaciju, na osnovu čega se može pretpostaviti ukupno potrebno vreme za

proračun. U ovom slučaju je do prekida proračuna došlo jer je dostignut kriterijum zaustavljanja i to maksimalni broj novih generacija bez promene vrednosti funkcije prilagođenosti. Kao što je prikazano na slici 5.3, proračun je zaustavljen pre nego što je dostignuta 600. generacija, sa najboljom vrednošću funkcije prilagođenosti od 52.842. Algoritam je pronašao rešenje koje zadovoljava sva ograničenja u šestoj generaciji i dostigao vrednost funkcije prilagođenosti od 55 nakon prvih 100 generacija.



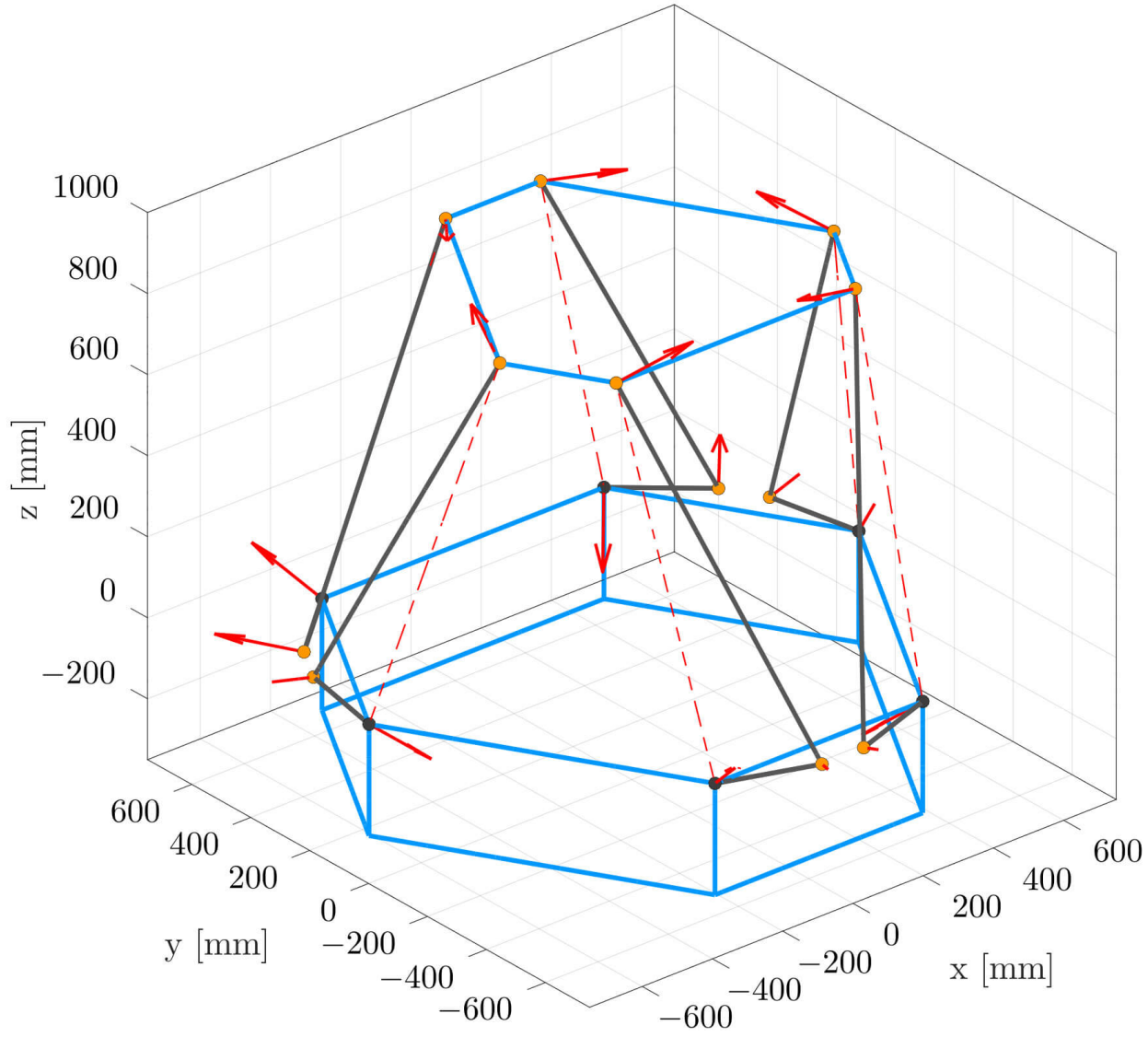
Slika 5.3 - Promena vrednosti funkcije prilagođenosti u odnosu na generaciju.

Proračun je počeo bez ikakve pretpostavke ili poznatog rešenja. U prvoj generaciji, vrednosti geometrijskih parametara za svaku jedinku u populaciji se generišu korišćenjem slučajne uniformne raspodele. U prvih pet generacija, nijedna od 5000 jedinki u populaciji nije zadovoljila postavljena nelinearna ograničenja. U šestoj generaciji, prve jedinke su zadovoljile sva ograničenja i izračunata je validna vrednost funkcije prilagođenosti. Ova potraga za jedinkama sposobnim da zadovolje sva ograničenja moguća je zahvaljujući implementaciji operacija ukrštanja i mutacije. Kasnije su ove funkcije, zajedno sa elitizmom i selekcijom, omogućile konvergenciju ka konačnom rezultatu, kao što je prikazano na slici 5.3.

5.5 Rezultati proračuna

Jedan od rezultata ove disertacije je algoritam sposoban da dizajnira mehanizme sa dovoljnim karakteristikama za simulaciju leta, uzimajući u obzir sva nametnuta ograničenja. Polazeći od vrednosti geometrijskih parametara koji se generišu na osnovu slučajne uniformne distribucije sa celobrojnim vrednostima unutar donjih i gornjih granica (kasnije preslikanih na realne vrednosti geometrijskih parametara), algoritam je dobio rešenje koje se može proizvesti i koristiti za simulaciju leta.

Kinematički model proračunate geometrije mehanizma prikazan je na slici 5.4, a verifikovano je da ova geometrija mehanizma ima vrednosti pokazatelja performansi iste kao što se izračunavaju tokom procesa proračuna.



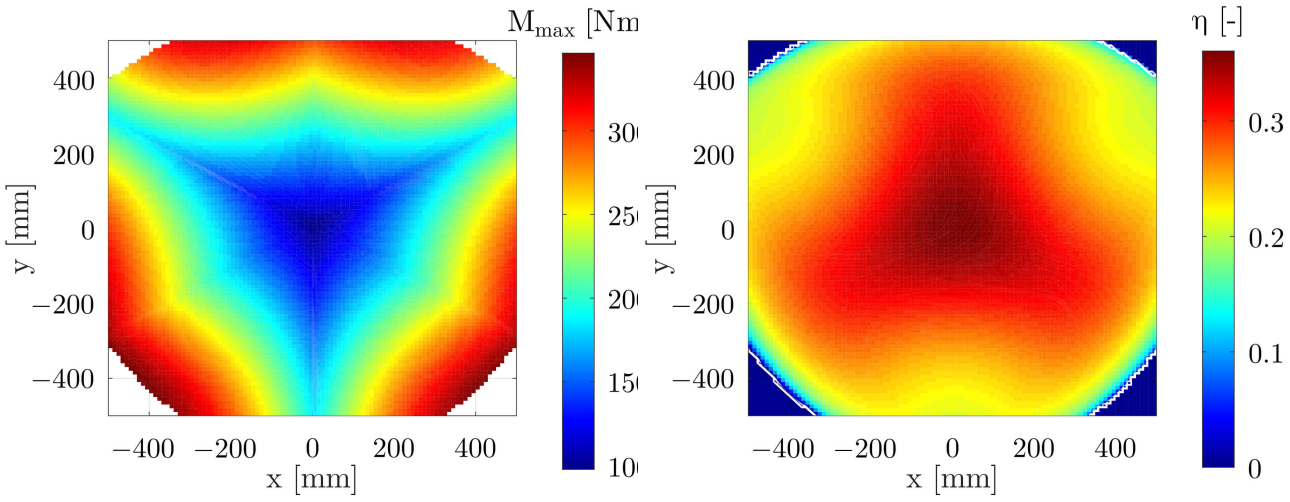
Slika 5.4 - Kinematički model proračunate geometrije mehanizma

Dobijene vrednosti geometrijskih parametara za konačno rešenje proračuna date su u tabeli 5.5. Ovo je optimalno rešenje dobijeno za vrednosti koje su usvojene i prikazane u tabeli 5.3, kao i za izabranu implementaciju funkcije prilagođenosti (jednačina 5.6).

Tabela 5.5 - Geometrijski parametri proračunatog mehanizma

Parametar	**Vrednost**
a	250 mm
s	1000 mm
R_p	490 mm
R_b	700 mm
γ_p	44°
γ_b	25°
ε	39°
τ_{ll}	19°
τ_{jp}	-81°
μ_{ll}	-31°
μ_{jp}	14°

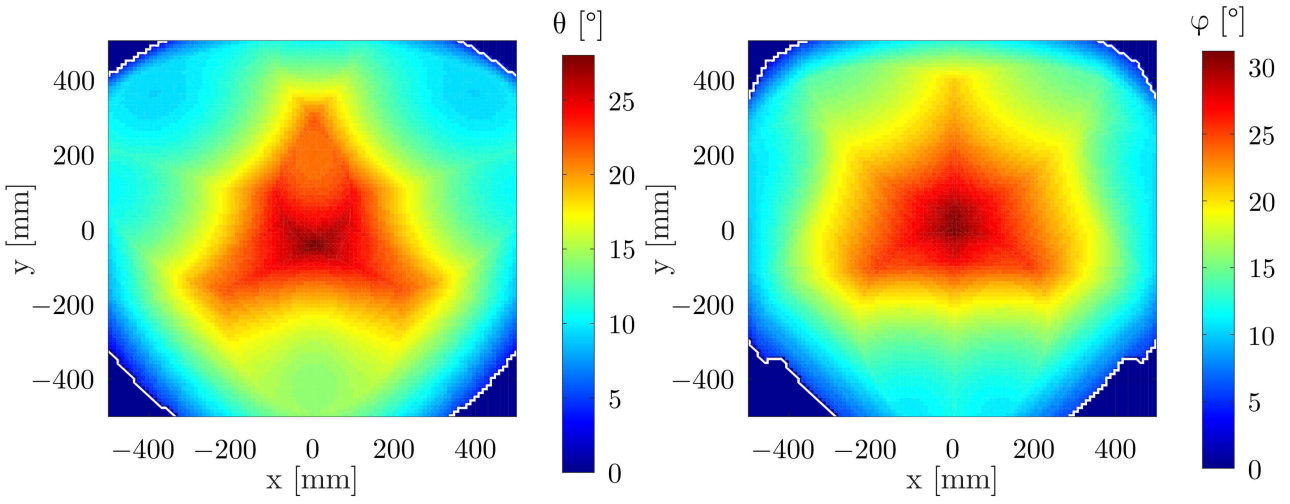
Slika 5.5a prikazuje maksimalni potrebni obrtni moment aktuatora u ravni O_{xy} kada sve druge spoljne koordinate imaju nultu vrednost. Slika 5.5b prikazuje vrednost indikatora spretnosti u istoj ravni.



(**a**) (**b**)

Slika 5.5 - (**a**) Maksimalni potrebni obrtni moment aktuatora i (**b**) spretnost mehanizma

Slike 5.6a i 5.6b prikazuju maksimalno dostižne vrednosti uglova propinjanja i valjanja u ravni O_{xy} kada sve druge spoljne koordinate imaju nultu vrednost. Očekivano, mehanizam ima najbolje performanse u blizini početne pozicije, a njegove mogućnosti se smanjuju promenom vrednosti bilo koje spoljašnje koordinate.



(**a**) (**b**)

Slika 5.6 - (**a**) Maksimalni nagib i (**b**) uglovi nagiba

Sledeće dve tabele (tabele 5.6 i 5.7) prikazuju maksimalne i minimalne vrednosti svake spoljašnje koordinate. Ove vrednosti predstavljaju maksimalne i minimalne vrednosti određene spoljašnje koordinate koje platforma može da dostigne (postoji rešenje inverzne kinematike). Tokom ovog proračuna, samo je koordinata z bila promenljiva, a sve ostale spoljašnje koordinate su imale nultu vrednost. Ovi rezultati pokazuju mogućnosti dobijenog mehanizma.

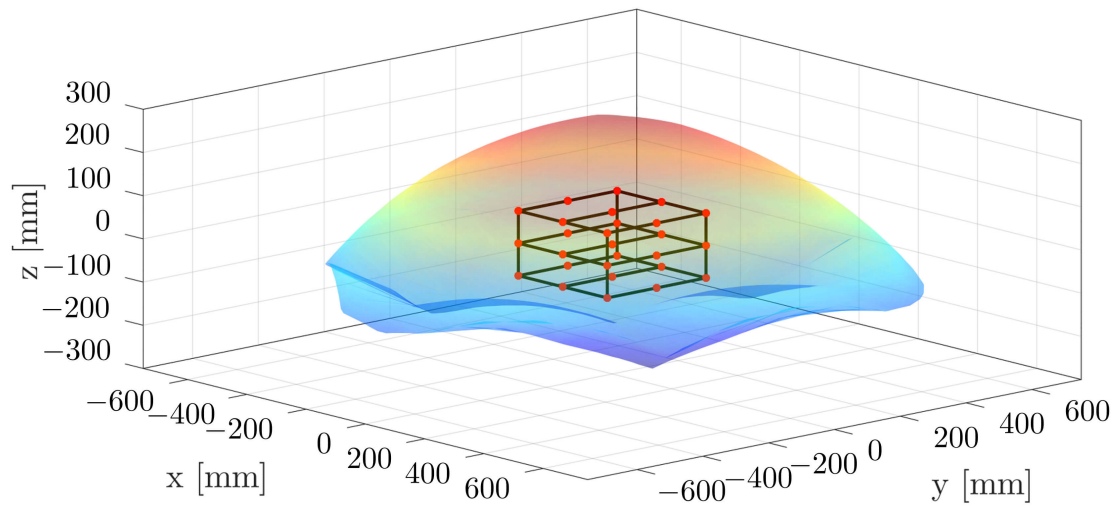
Tabela 5.6 - Maksimalne vrednosti spoljašnjih koordinata

Spoljašnje koordinate	Vrednost	Koordinata z
x	640 mm	−38 mm
y	670 mm	−91 mm
z	290 mm	290 mm
ψ	67.5°	43 mm
θ	31.5°	33 mm
φ	38.5°	−6 mm

Tabela 5.7 - Minimalne vrednosti spoljašnjih koordinata

Spoljašnje koordinate	Vrednost	Koordinata z
x	−640 mm	−38 mm
y	−750 mm	−89 mm
z	−180 mm	−180 mm
ψ	−67.5°	43 mm
θ	−31.5°	33 mm
φ	−35°	78 mm

Na slici 5.7 prikazan je radni prostor mehanizma sa konstantnom orijentacijom i to za nulte uglove. Na ovoj slici su ujedno i prikazane referentne tačke koje su korišćene za evaluaciju indikatora performansi.



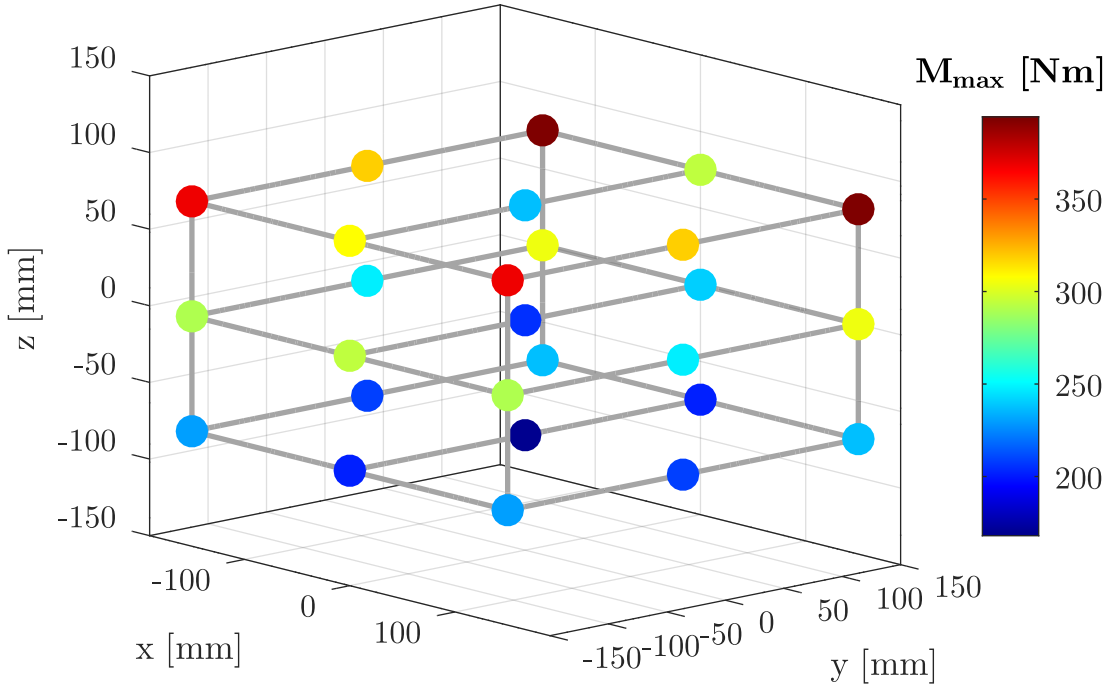
Slika 5.7 - Prikaz referentnih tačaka u okviru radnog prostora

Da bi se dobili svi prethodno prikazani podaci, na osnovu prethodno prikazanih jednačina i programa datih u prilozima, koristi se konfiguracija data u prilogu R. Nakon što je određen kinematski model, urađena je materijalizacija kako bi se na kraju dobio trodimenzionalni model mehanizma simulatora leta čiji realističan prikaz je dat u okviru slike 5.8.

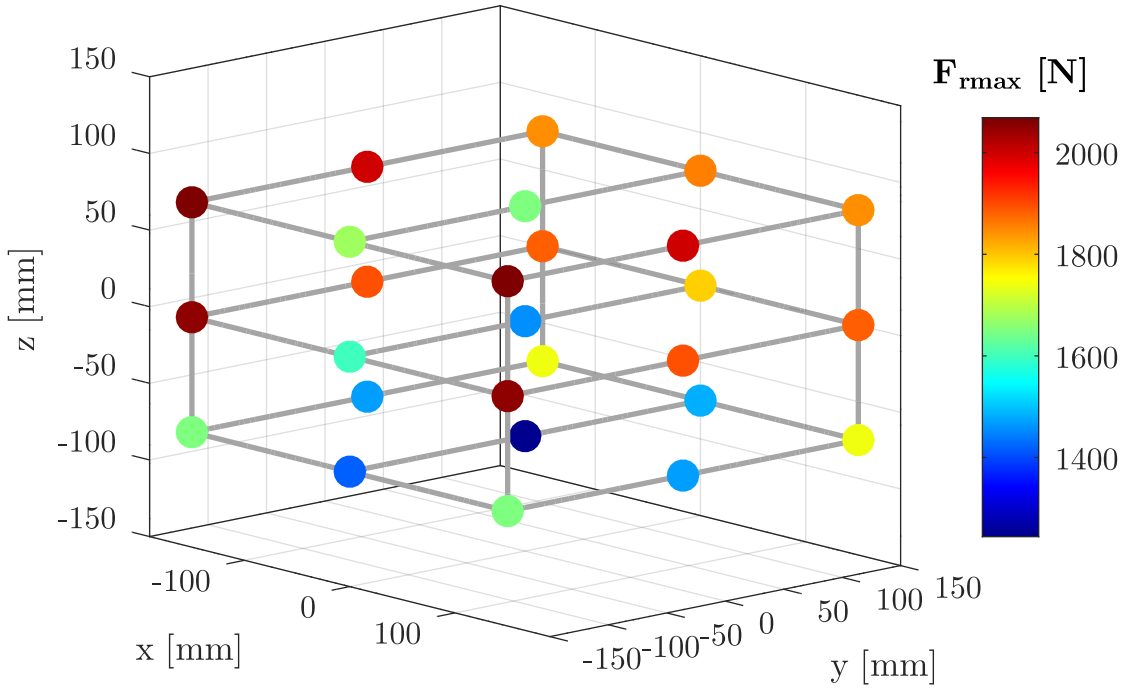


Slika 5.8 - Render 3D modela mehanizma

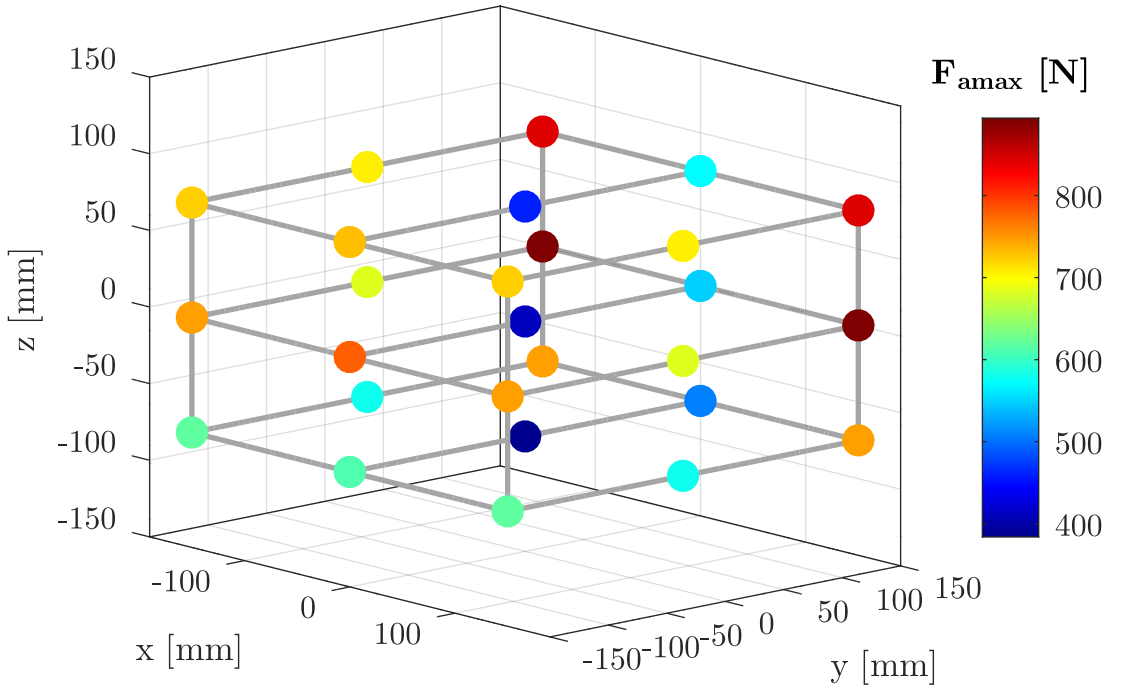
Zatim su na slikama 5.9, 5.10 i 5.11 prikazane maksimalne vrednosti opterećenja aktuatora u referentnim tačkama.



Slika 5.9 - Maksimalni potrebni moment aktuatora u referentnim tačkama

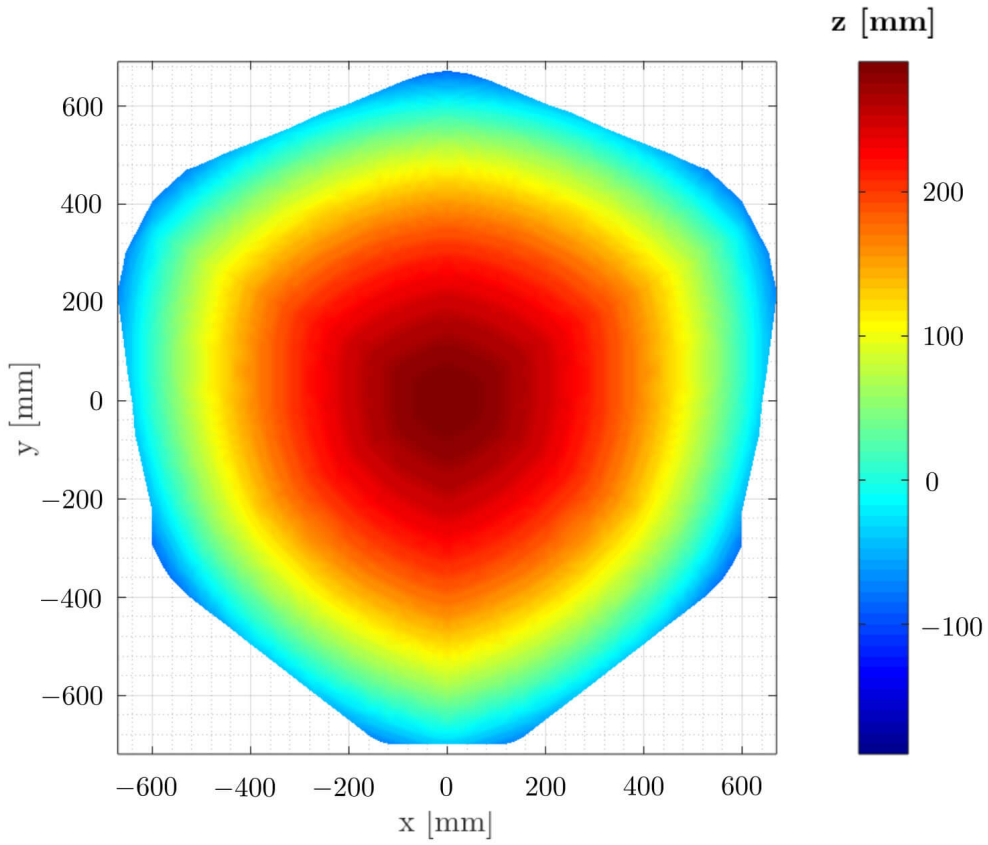


Slika 5.10 - Maksimalna radijalna sila u referentnim tačkama

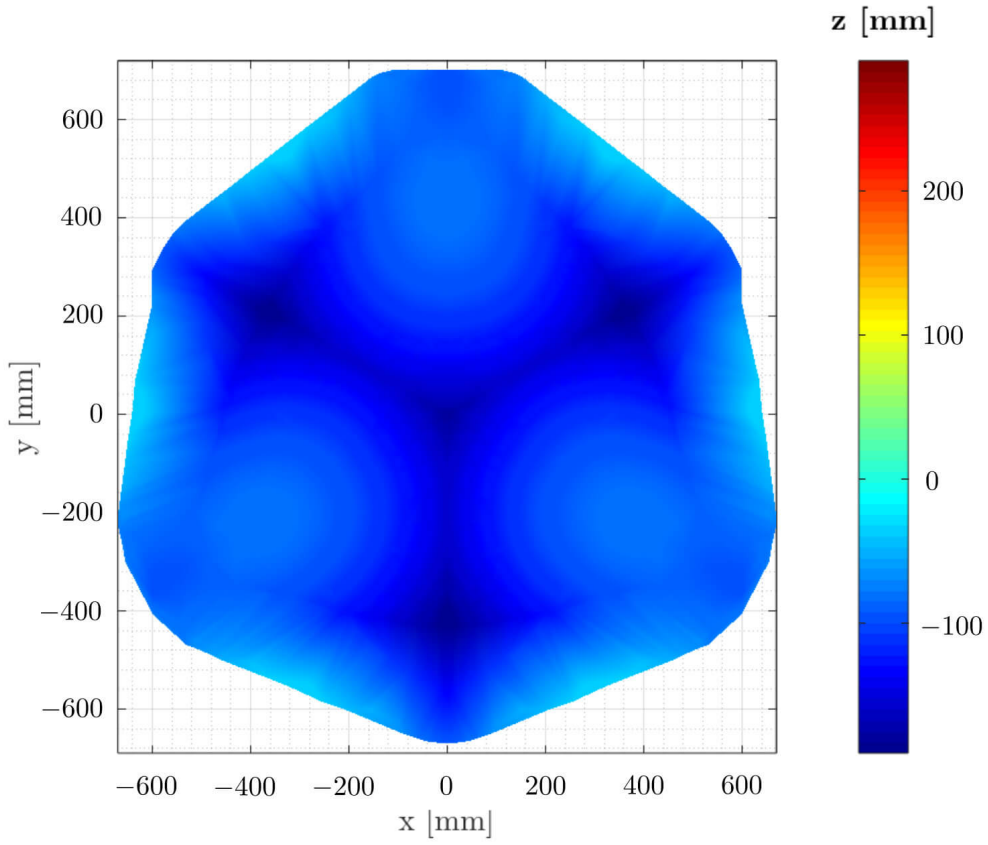


Slika 5.11 - Maksimalna aksijalna sila u referentnim tačkama

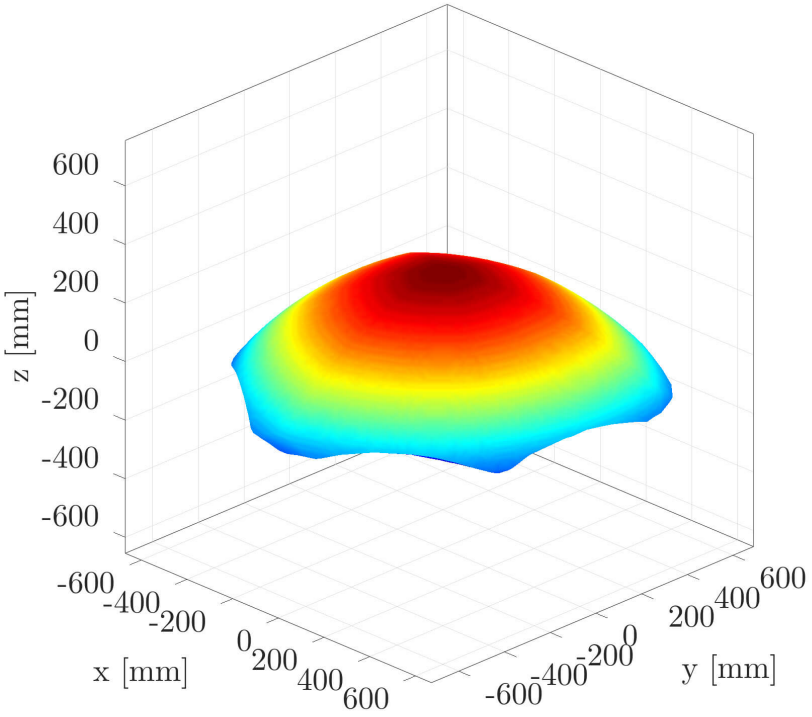
Na poslednje četiri slike (5.12, 5.13, 5.14 i 5.15) u ovom poglavlju detaljnije je prikazan radni prostor mehanizma sa konstantnom orijentacijom, kao što je već napomenuto.



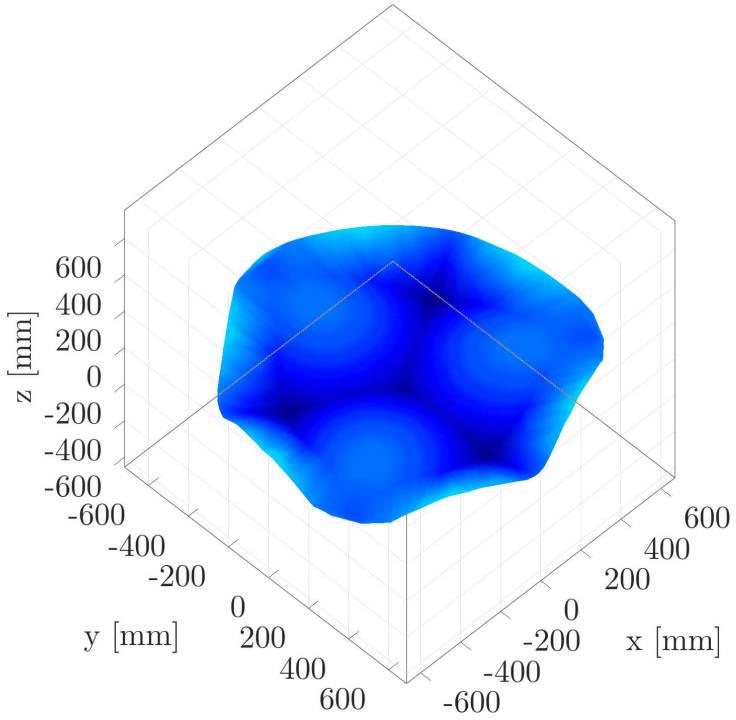
Slika 5.12 - Spoljna površ radnog prostora odozgo



Slika 5.13 - Spoljna površ radnog prostora odozdo



Slika 5.14 - Izometrijski prikaz spoljne površi radnog prostora odozgo



Slika 5.15 - Izometrijski prikaz spoljne površi radnog prostora odozdo

Treba napomenuti da se podešavanja proračuna i implementacija funkcije prilagođenosti mogu veoma lako prilagoditi za druge željene performanse, kao i za potpuno druge aplikacije.

6

Priprema eksperimentalnog ispitivanja sistema

*"Gotovo je, kada je **gotovo**."*

– John Carmack[1]

Nakon što su u prethodnim poglavljima disertacije zasebno analizirani mehanizam simulatora leta, bespilotne letelice i sistemi za daljinsko pilotiranje, u ovom poglavlju disertacije se analizira njihova integracija u cilju dobijanja sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi koji može da se eksperimentalno ispita. Rezultat ove integracije je novi proizvod.

6.1 Simulator leta

Za potrebe realizacije ove disertacije su izrađena dva prototipa simulatora leta i to umanjeni prototip sa oznakom `MP-6DOF-M` kao i prvi prototip u stvarnoj veličini sa oznakom `MP-6DOF-XXX`. U ovom poglavlju su detaljno opisani procesi projektovanja, proizvodnje i programiranja oba prototipa. U sledećem poglavlju su data i numerička i eksperimentalna ispitivanja ovih prototipova.

6.1.1 Umanjeni prototip MP-6DOF-M

Kako bi se funkcionalnost sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi proverila i eksperimentalno ispitala, pored računarskih simulacija koje su urađene, napravljen je umanjeni fizički model konceptualnog rešenja simulatora leta. U tabeli 6.1 je data konfiguracija modela (konfiguracioni fajl je dat u prilogu R), svi podaci kojima se jednoznačno geometrijski opisuje mehanizam simulatora leta. U trenutku projektovanja ovog prototipa je pri izboru vrednosti geometrijskih parametara dat prioritet jednostavnosti izrade i u smislu konstrukcije i u smislu raspoloživih električnih komponenti.

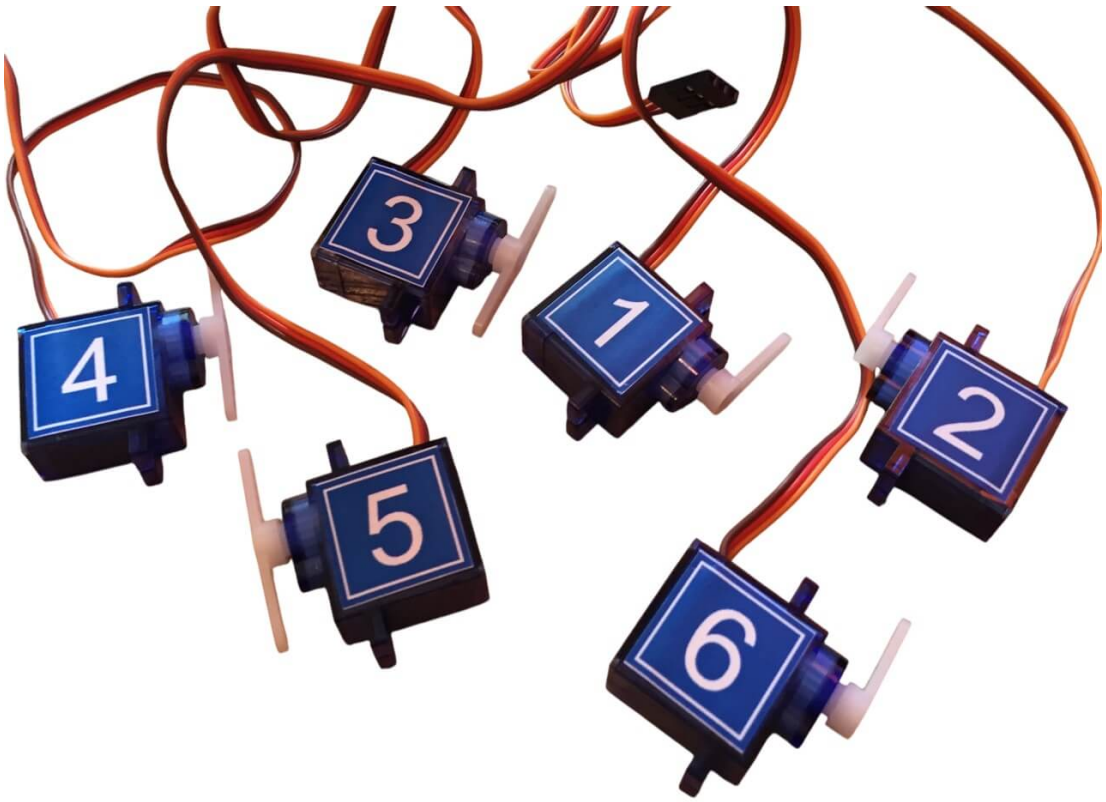
[1] John Carmack (21. avgust 1970) je američki programer video-igara. Suosnivač je kompanije *id Software* i bio je vodeći programer igara kao što su Doom i Quake.

Tabela 6.1 - Geometrijski parametri prototipa MP-6DOF-M

Parametar	Vrednost
a	25 mm
s	112 mm
R_p	63.5 mm
R_b	81 mm
γ_p	35.5°
γ_b	29°
ε	0°
τ_{ll}	0°
τ_{jp}	60°
μ_{ll}	0°
μ_{jp}	0°

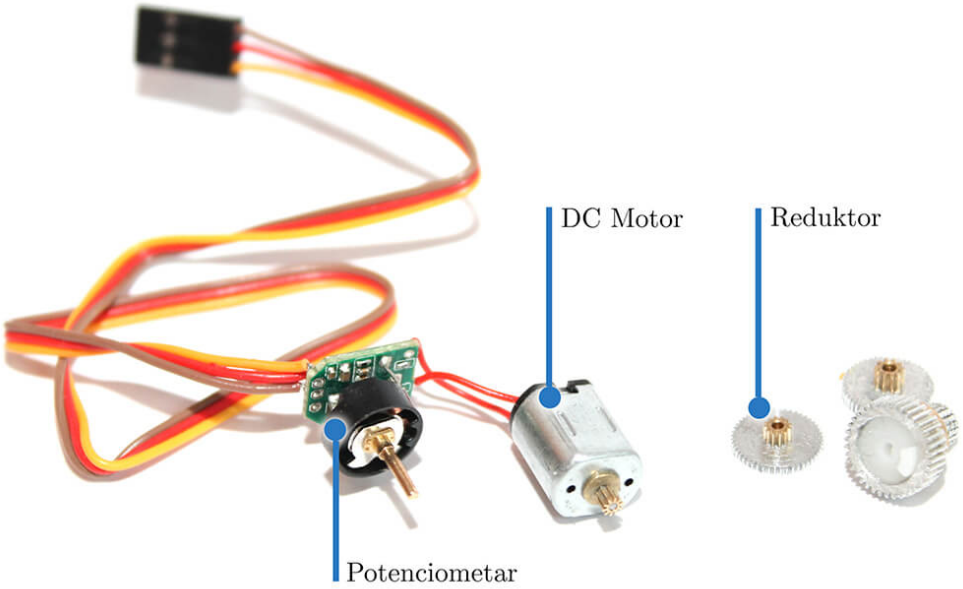
6.1.1.1 Električni sistem

Za šest obrtnih aktuatora koji su neophodni za izradu ovog mehanizma su izabrani električni servo motori. Ovi motori (prikazani na slici 6.1) su razvijeni za potrebe vazduhoplovnog modelarstva i za izradu različitih prototipova i obično se vrši njihova kategorizacija na osnovu mase. Pored mase, prilikom izbora ovog tipa servo motora se vodi računa o brzini, obrtnom momentu i tipu materijala (zupčanici reduktora mogu biti plastični ili metalni).



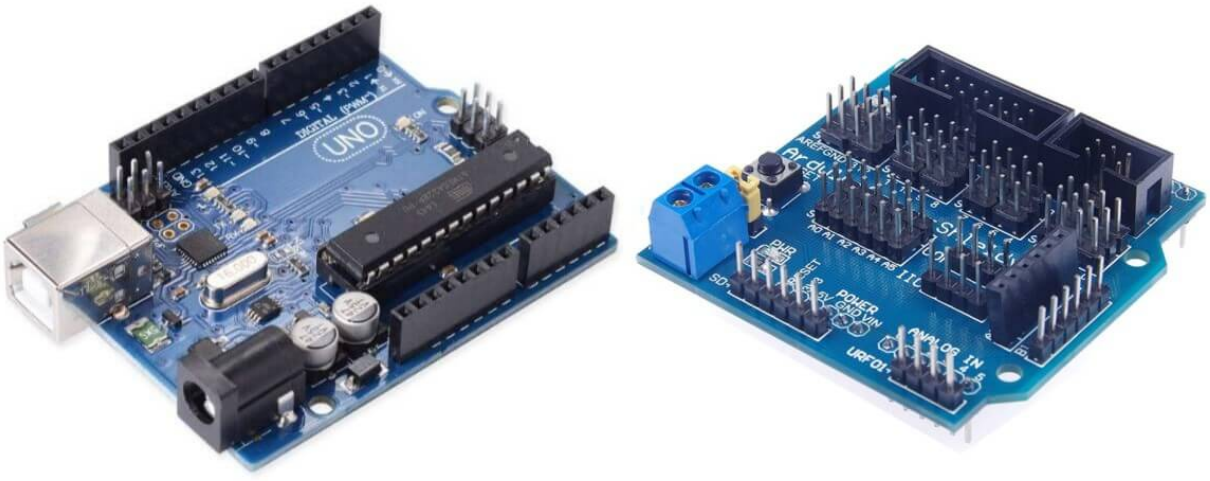
Slika 6.1 - Servo motori za umanjeni prototip

Tipičan servo motor za radio upravljane modele (RC) se sastoji iz motora jednosmerne struje (DC, Direct Current), potenciometra, reduktora i kontrolera (kao što se vidi na slici 6.2).



Slika 6.2 - Delovi servo motora

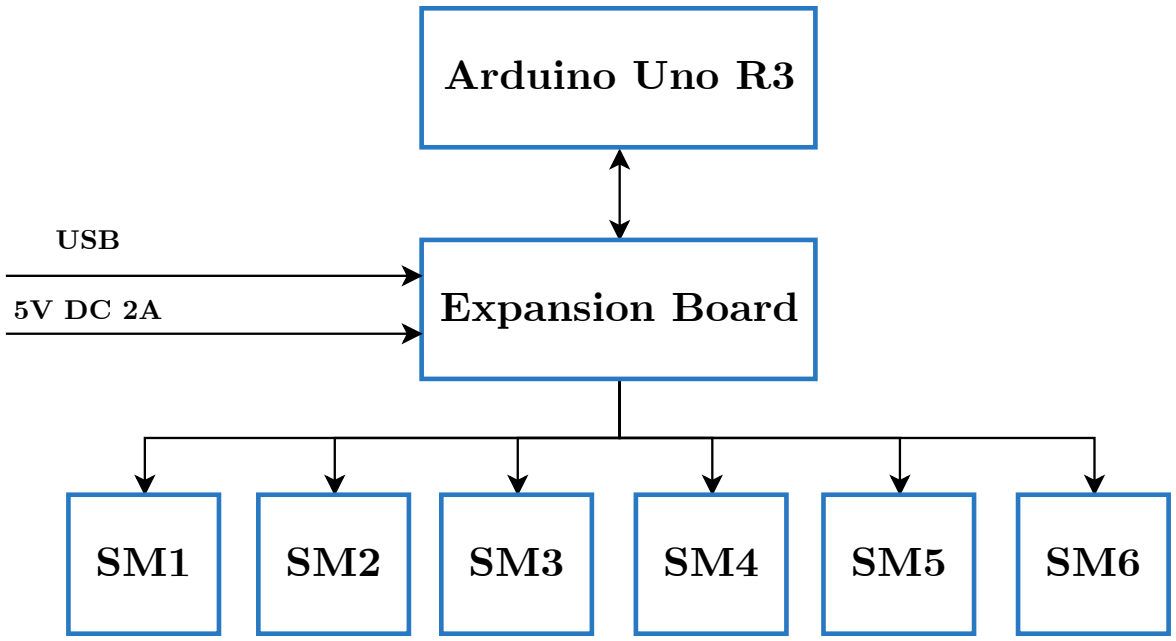
Servo motor funkcioniše na principu zatvorenog sistema upravljanja u kome se merenje ostvarenog ugla vrši potenciometrom, dok se zupčastim reduktorom obezbeđuje odgovarajući obrtni moment. Kako bi se ostvarila željena rotacija, informacija o željenom uglu mora da bude pretvorena u signal odgovarajućeg tipa kako bi upravljački sistem servo motora razumeo komandu. Ovaj tip motora se najčešće kontroliše pomoću modulacije širine pulsa u signalu (`PWM signal`). To je digitalni signal koji servo motor analizira i potom se ponaša u skladu sa dobijenom komandom. Za ovo pretvaranje informacije o željenom uglu u odgovarajući signal je korišćen mikrokontroler `ATmega328` kompanije `Microchip` na razvojnoj ploči `Arduino Uno R3` (prikazana na slici 6.3 levo).



Slika 6.3 - Arduino Uno razvojna ploča (levo) i dodatak za nju (desno)

Za distribuciju energije i povezivanje servo motora sa razvojnom pločom se koristi dodatna štampana ploča sa odgovarajućim konektorima koja je prikazana desno na slici 6.3.

Na slici 6.4 je prikazan konačan blok dijagram elektronike umanjenog prototipa simulatora leta. Sa ovog dijagrama se vidi da se ovako napravljen uređaj povezuje sa računarom preko `USB`-a (`tip A` konektor) i da se napaja sa jednosmernom strujom sa naponom od `5V` i jačine struje od `2A` (`2.1 x 5.5` mm konektor).



Slika 6.4 - Blok dijagram elektronike umanjenog prototipa

6.1.1.2 3D model konstrukcije i tehnička dokumentacija

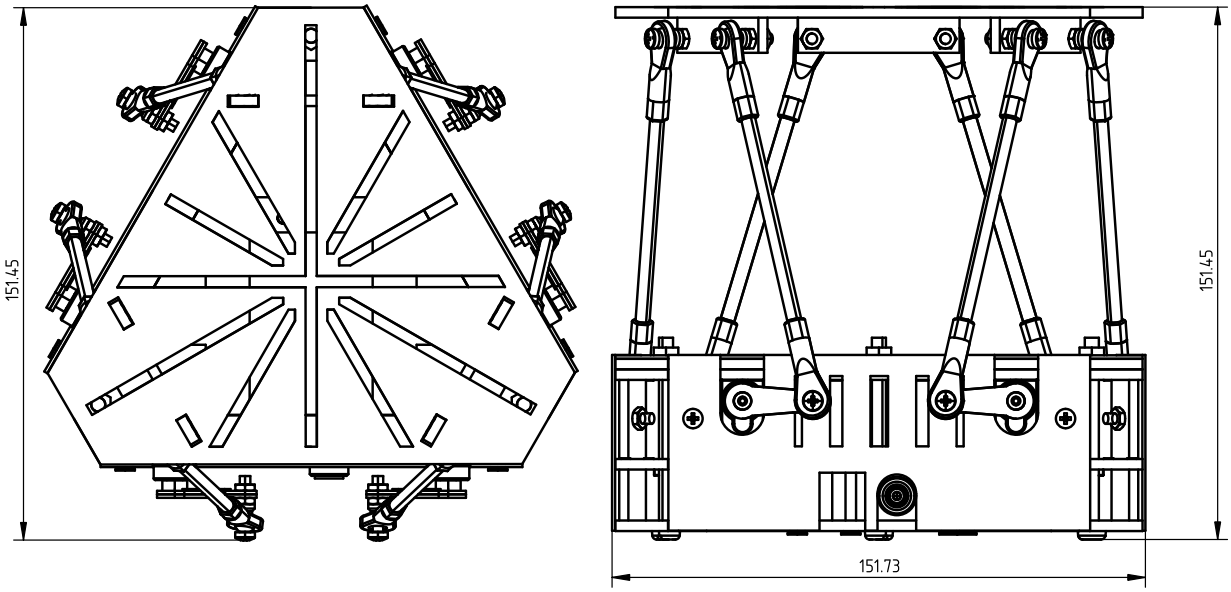
Projektovanje konstrukcije započinje analizom raspoloživih tehnologija za proizvodnju. U trenutku projektovanja umanjenog prototipa nije bila dostupna tehnologija 3D štampanja pa je doneta odluka da se izrada konstrukcije bazira na elementima koji se dobijaju rezanjem na laseru.

Pored elektronike, ključan element koji je bilo potrebno nabaviti pre nego što se prešlo na samu realizaciju projekta su bili sferni zglobovi za gornje poluge umanjenog prototipa (ovi zglobovi su prikazani na slici 6.5). Ove zglobove je teško moguće zameniti bilo kojim drugim elementom a bez njih nije moguće ostvariti potrebna kretanja mehanizma.



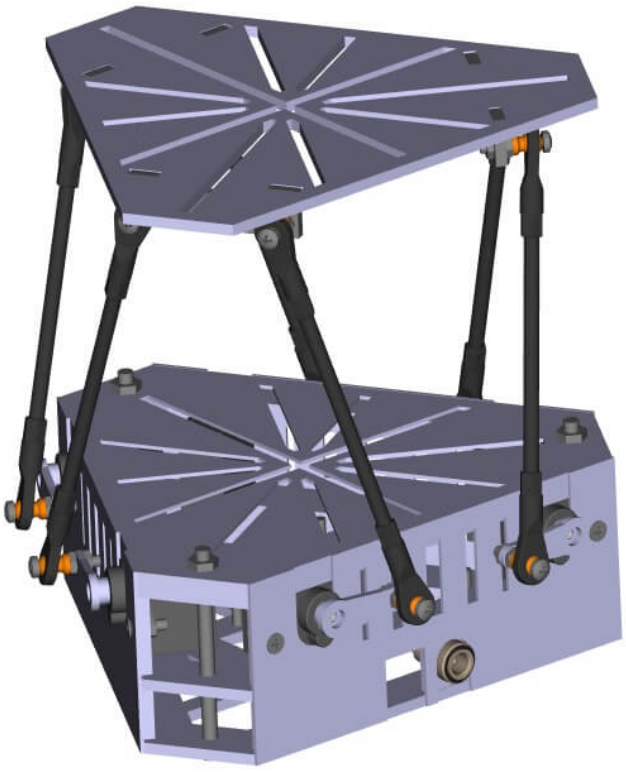
Slika 6.5 - Sferni zglobovi za gornje poluge umanjenog prototipa

Formiranje 3D modela je započeto formiranjem modela za elemente koji se kupuju (kao što su servo motori, konektori, elektronika i sferni zglobovi). Onda je moguće preći na formiranje geometrije na osnovu geometrijskih parametara koji su prethodno usvojeni. Treba napomenuti da su u ovom slučaju usvojeni prilično jednostavni parametri za 3D modeliranje. Na slici 6.6 su prikazane gabaritne mere konačnog 3D modela.

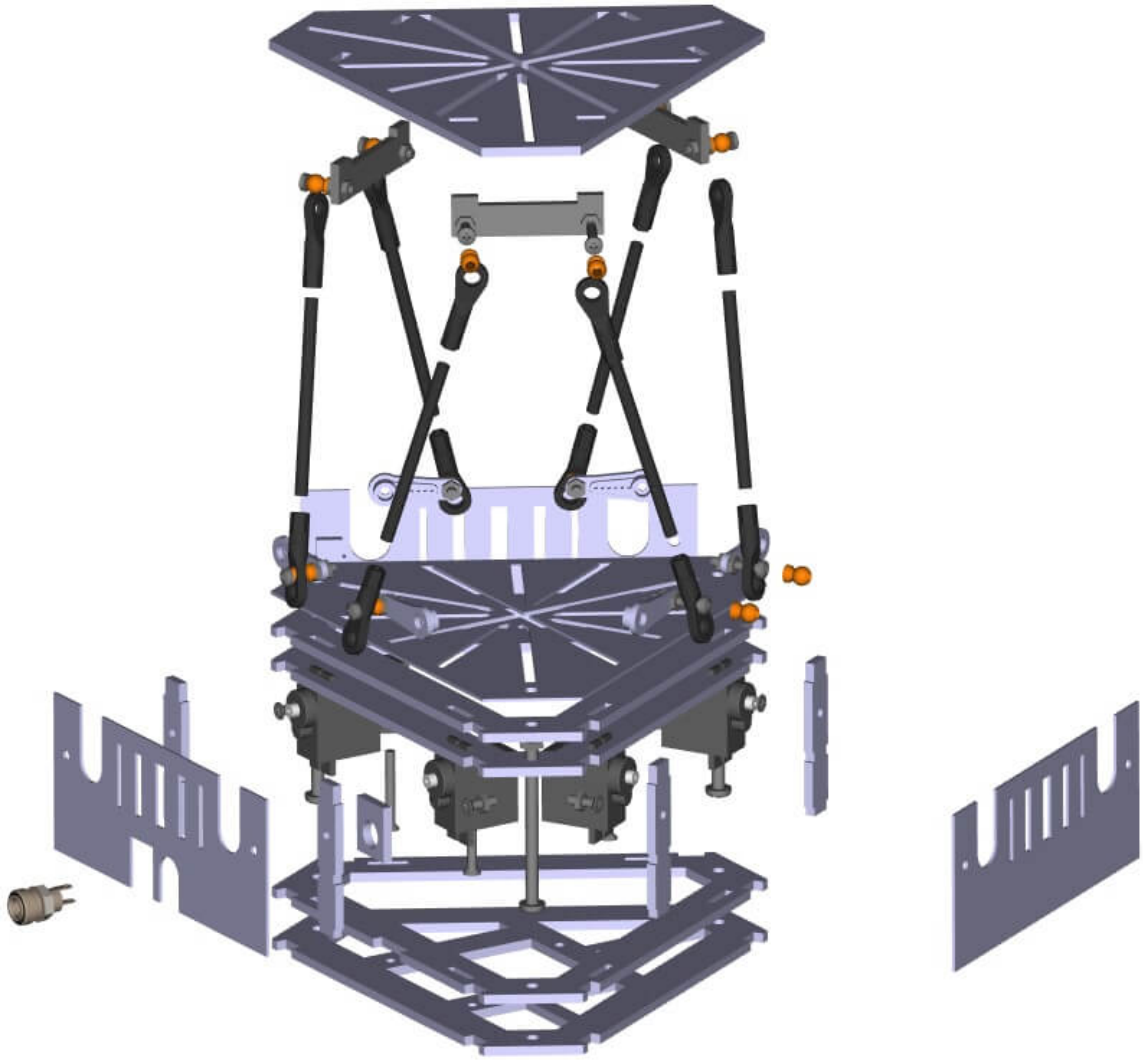


Slika 6.6 - Gabaritne mere umanjenog prototipa

Na slici 6.7 je prikazan 3D model umanjenog prototipa `MP-6DOF-M` u programu *FreeCAD*. Da bi se bolje prikazao ovaj sklop, na slici 6.8 je prikazan model sklopa sa razdvojenim delovima.



Slika 6.7 - 3D model sklopa MP-6DOF-M

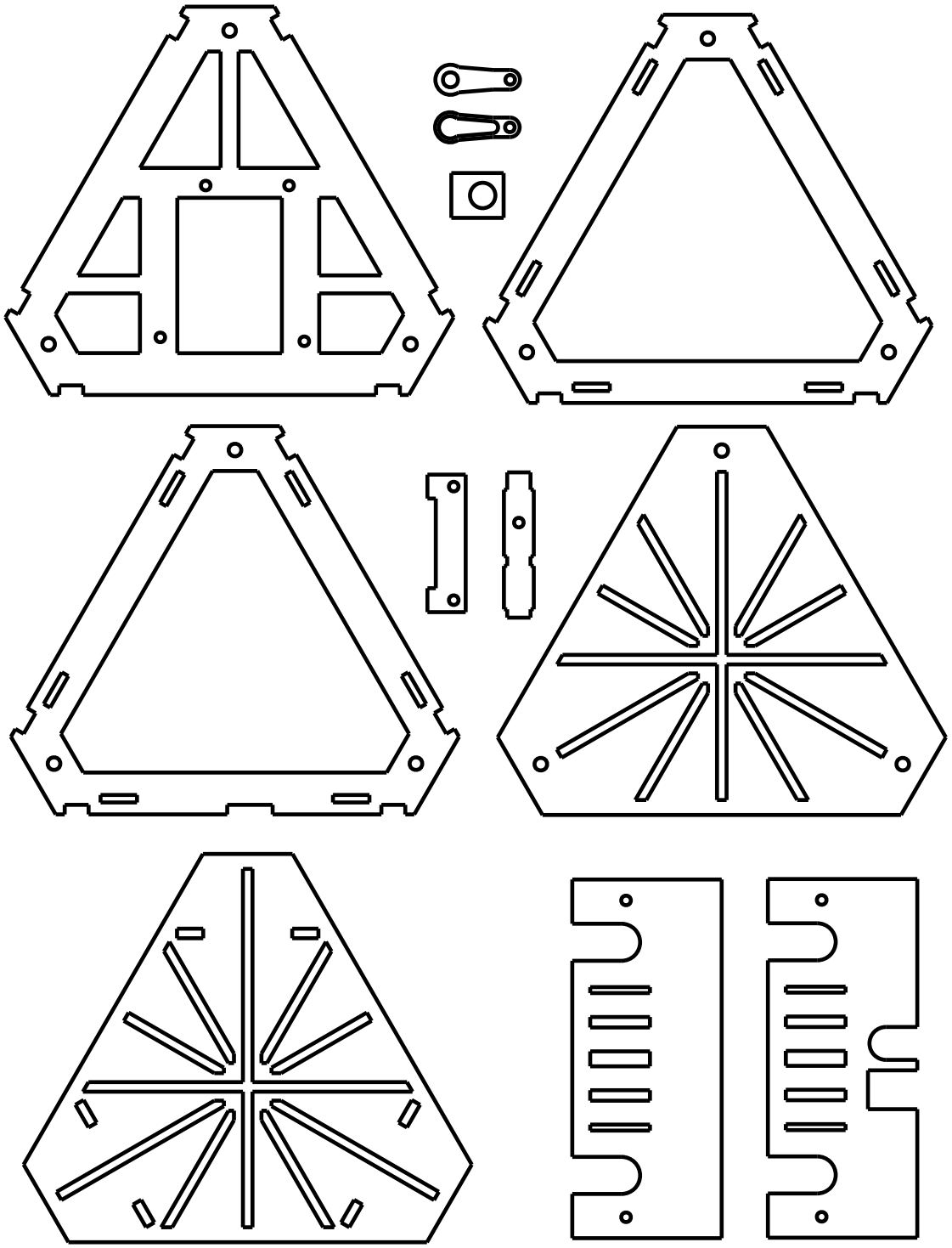


Slika 6.8 - Razdvojeni delovi sklopa MP-6DOF-M

U tabeli 6.2 je dat spisak a na slici 6.9 su prikazani tehnički crteži za rezanje delova laserom.

Tabela 6.2 - Spisak delova MP-6DOF-M za rezanje laserom

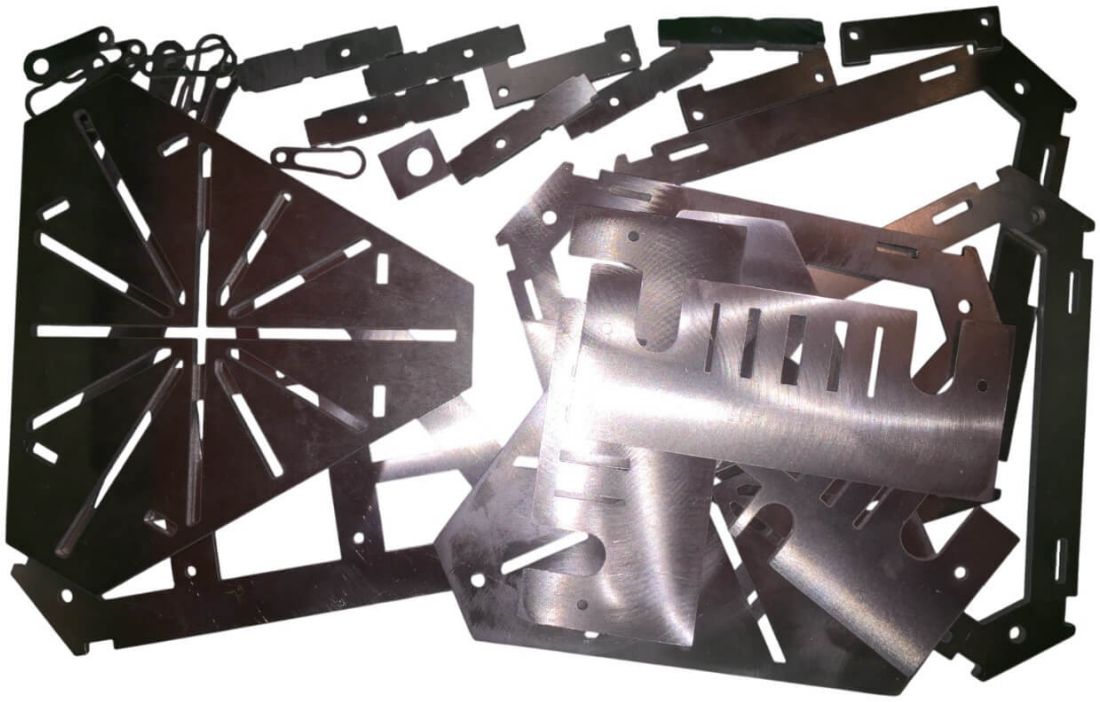
Br. crteža	Količina	Debljina	Materijal
10001	1	3 mm	AlMg3
10002	1	3 mm	AlMg3
10003	2	3 mm	AlMg3
10004	1	1 mm	AlMg3
10005	10	1 mm	AlMg3
10006	1	3 mm	AlMg3
10007	5	3 mm	AlMg3
10009	10	3 mm	AlMg3
10010	2	3 mm	AlMg3
10011	2	1 mm	AlMg3
10012	1	1 mm	AlMg3
10013	10	1 mm	AlMg3



Slika 6.9 - Crteži delova MP-6DOF-M za rezanje laserom

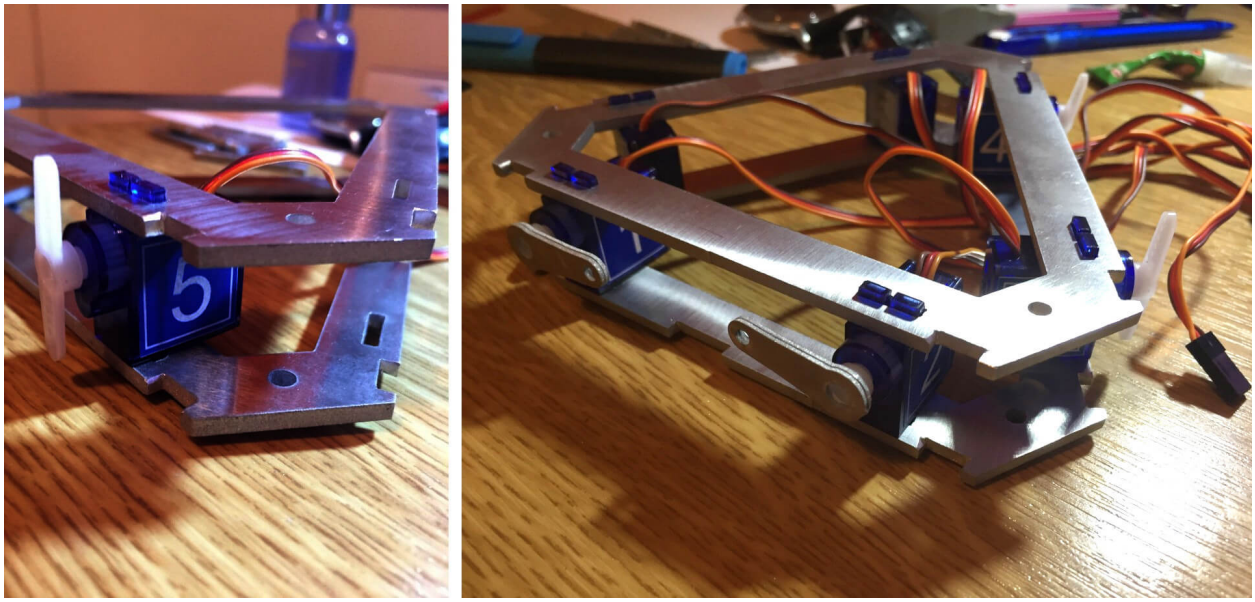
6.1.1.3 Proizvodnja

Proizvodnja umanjenog prototipa `MP-6DOF-M` je započeta krajem oktobra 2016. godine, rezanjem delova od aluminijuma na CNC laseru. Korišćeni su pločasti materijali različitih debljina. Na slici 6.10 su prikazani svi delovi nakon rezanja.



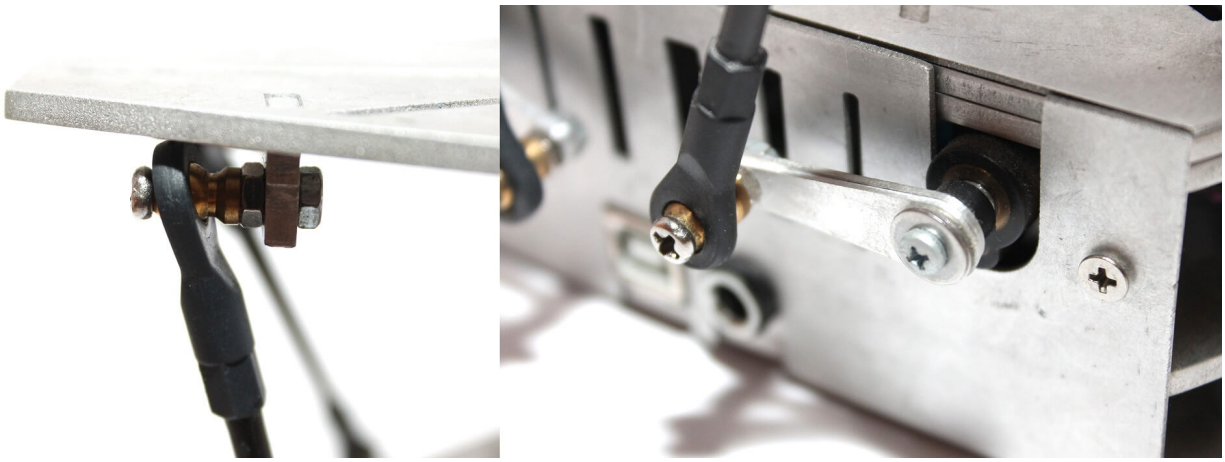
Slika 6.10 - Delovi MP-6DOF-M dobijeni rezanjem laserom

Prilikom projektovanja je kompletna konstrukcija zamišljena tako da se sklapa sa minimalnim brojem vijaka i navrtki i bez zavarivanja ili lemljenja, što bi u kasnijim fazama projekta moglo da bude značajno ukoliko se ovakav uređaj bude koristio kao školsko ili univerzitetsko učilo. Na slici 6.11 je prikazano sklapanje. Svih šest servo motora imaju svoje uležištenje i cela baza se spaja sa tri dugačka vijka koji prolaze kroz celu bazu.



Slika 6.11 - Sklapanje umanjenog prototipa krajem oktobra 2016. godine

Gornje poluge su izrađene od kompozitnog materijala koji se sastoji od epoksidne smole sa ugljeničnim vlaknima. Poluge i sferni zglobovi se spajaju lepljenjem. Korišćeni sferni zglobovi (prikazani na slici 6.12) imaju otvore (ose za vezu) koje omogućavaju rastavljivu vezu između gornjih poluga sa ostatkom mehanizma korišćenjem `M3` vijaka.



Slika 6.12 - Sferni zglobovi mehanizma umanjenog prototipa

Konačno, na slici 6.13 su prikazani i finalni izgled 3D modela, ali još bitnije, i finalni izgled umanjenog prototipa koji je proizveden za potrebe ove disertacije.



Slika 6.13 - Konačni izgled umanjenog prototipa (fizički i 3D model)

Gornja platforma je napravljena tako da je moguće jednostavno vezivati različite terete na nju. Tokom istraživanja su rađeni različiti prikazi i u skladu sa tim su vezivani različiti eksperimenti, primer za to je dat na slici 6.14.



Slika 6.14 - Model pilotske kabine na umanjenom prototipu

Simulator leta sa prethodne slike prvi put je prikazan u novembru 2016. godine, u okviru prezentacije na temu *"SIMULATOR LETA AVIONA SA 6 STEPENI SLOBODE"* iz predmeta *Mehatronika*, kod profesora Aleksandra Vega, na Mašinskom fakultetu Univerziteta u Beogradu.

Umanjeni prototip je prvi put javno prikazan na 61. međunarodnom sajmu tehnike i tehničkih dostignuća u Beogradu, maja 2017. godine, na štandu Mašinskog fakulteta Univerziteta u Beogradu. Na istom sajmu je prikazan i 2018. godine zajedno sa bespilotnom letelicom i prezentovan je rad sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi (slika 6.15).



Slika 6.15 - Prototip umanjenog prototipa na sajmu tehnike 2018. godine u Beogradu

6.1.1.4 Program za upravljanje

Jedan od ključnih razloga za izbor Arduino Uno razvojne platforme u okviru realizacije električnog podsistema umanjenog prototipa MP-6DOF-M jeste njena široka rasprostranjenost i podrška u zajednici. Arduino platforma omogućava jednostavnu i brzu primenu AVR mikrokontrolera (ATmega328), obezbeđeno je sve neophodno za lako programiranje i implementaciju različitih funkcionalnosti (ugrađeni USB interfejs za programiranje i napajanje). Pored toga, dostupnost velikog broja softverskih biblioteka (na primer za generisanje PWM signala i komunikaciju preko serijskog porta) značajno ubrzava razvoj i testiranje. Uopšteno gledano, ova platforma omogućava veoma lak upliv u svet elektronike i programiranja (naročito fizičkih sklopova).

Arduino platforma je popularna u edukativnim sredinama, što je važan aspekt kada se razmatra potencijalna primena ovog prototipa u okviru nastave. Zbog pomenute rasprostranjenosti lako je pronaći odgovarajuću dokumentaciju, primere koda i gotove biblioteke, što omogućava kraće vreme za implementaciju i olakšava naknadnu izmenu ili unapređenje sistema. Konačno, Arduino razvojno okruženje (IDE) dodatno olakšava proces programiranja mikrokontrolera, pa tako doprinosi ukupnoj jednostavnosti integracije između programskog dela i hardvera. U ovom slučaju se za programiranje koristi JSLAB okruženje i za otpremanje programa se koristi funkcija uploadArduino().

U prilogu S je dat program MP-6DOF-M_CB.ino koji se pokreće na mikrokontroleru umanjenog protitipa. Zadatak ovog programa je da na osnovu instrukcija, koje dobija preko serijske komunikacije, istovremeno upravlja svim servo motorima. Veliki značaj je posvećen ispravnosti komunikacije jer loše instrukcije servo motorima mogu prouzrokovati mehanička oštećenja prototipa. Zbog toga je implementiran specijalni binarni komunikacioni protokol PRDC_SCBP koji je detaljno opisan u poglavlju 6.1.1.5 i čija je implementacija data u prilogu T. U sekciji 6.1.7 ove disertacije su data značenja kodova statusa, a u sekciji 6.1.8 definicije paketa i poruka koje se koriste za komunikaciju.

Zbog praktičnosti za upotrebu je odlučeno da se kompletan algoritam za planiranje kretanja i rešavanje kinematičkih problema zapravo izvršava na računaru koji mikrokontroleru samo izdaje komande za željene uglove otklona donjih poluga preko USB-a.

6.1.1.5 Komunikacioni protokol PRDC_SCBP

Komunikacioni protokol PRDC_SCBP (*Serial Communication Binary Protocol*) je dizajniran za serijsku komunikaciju između uređaja, omogućavajući pouzdanu razmenu podataka uz mehanizme za obeležavanje početka i kraja paketa, enkodiranje specijalnih znakova i proveru integriteta podataka putem kontrolne sume. U prilogu T su date implementacije u programskim jezicima JavaScript i Octave, kao i Arduino biblioteka.

Serijska komunikacija zahteva efikasne protokole koji mogu da obezbede pouzdan prenos podataka čak i u prisustvu smetnji. Protokol PRDC_SCBP je osmišljen da odgovori na ove zahteve kroz jednostavnu, ali robusnu implementaciju koja koristi standardne kontrolne znakove i mehanizme za detekciju grešaka. Karakteristike protokola su:

- *Struktura paketa*

  Svaki paket podataka u PRDC_SCBP protokolu ima jasno definisanu strukturu:

  - *Start marker (M1):* Označava početak paketa. ASCII vrednost je 60 ('<').
  - *Podaci:* Niz bajtova koji predstavljaju korisničke podatke.
  - *Kontrolna suma (CKS):* Dva bajta koja služe za proveru integriteta podataka.

– *Stop marker (M2):* Označava kraj paketa. `ASCII` vrednost je `62` (`'>'`).

Specijalni znakovi unutar podataka, koji mogu biti interpretirani kao kontrolni znakovi (`M1`, `M2`, `M3`), enkodiraju se posebnim mehanizmom kako bi se izbegla zabuna.

- *Enkodiranje specijalnih znakova*

  Da bi se obezbedila ispravna interpretacija podataka, specijalni znakovi unutar korisničkih podataka se enkodiraju koristeći *marker za enkodiranje M3*, čija je `ASCII` vrednost `95` (`'_'`).

  Enkodiranje se vrši na sledeći način:

  – *M1 (60):* Zamenjuje se sa sekvencom `M3`, `1`.
  – *M2 (62):* Zamenjuje se sa sekvencom `M3`, `2`.
  – *M3 (95):* Zamenjuje se sa sekvencom `M3`, `3`.

  Ovaj postupak osigurava da kontrolni znakovi unutar podataka ne budu pogrešno protumačeni kao *start* ili *stop markeri* paketa.

- *Kontrolna suma*

  Za proveru integriteta podataka koristi se dvobajtna kontrolna suma, koja se izračunava pomoću *Fletcher-ovog algoritma.*

- *Inicijalizacija Protokola*

  Pre početka komunikacije, potrebno je inicijalizovati protokol pozivanjem metode `begin()`, koja postavlja početne vrednosti kontrolnih znakova i resetuje interne promenljive.

- *Slanje paketa*

  Za slanje podataka koristi se metoda `sendPacket(data_out, N_out)`, koja obavlja sledeće korake:

  1. *Izračunavanje kontrolne sume:* Poziva se metoda `getCKS()` sa korisničkim podacima.
  2. *Enkodiranje podataka i kontrolne sume:* Specijalni znakovi se enkodiraju u skladu sa već opisanim mehanizmom.
  3. *Formiranje izlaznog bafera: Start marker* (`M1`) se dodaje na početak, zatim enkodirani podaci i kontrolna suma, a na kraju *stop marker* (`M2`).
  4. *Vraćanje izlaznog niza:* Rezultujući niz bajtova spreman je za slanje preko serijskog interfejsa.

- *Prijem paketa*

  Prijem podataka i rekonstrukcija originalnih podataka obavlja se metodom `readPacket(N_in)`, koja:

  1. *Čita ulazni bafer:* Bajtovi se čitaju sekvencijalno iz `buffer_in`.
  2. *Detektuje start i stop markere:* Početak i kraj paketa se identifikuju pomoću `M1` i `M2`.
  3. *Dekodira podatke:* Enkodirani specijalni znakovi se vraćaju u originalne vrednosti.

4. *Proverava kontrolnu sumu:* Izračunava se kontrolna suma primljenih podataka i upoređuje sa primljenom.
5. *Vraća rezultat:* Ako je kontrolna suma validna, vraća se niz originalnih podataka; u suprotnom, paket se odbacuje.

- *Rukovanje greškama*

  U slučaju detekcije neispravnog paketa ili greške u komunikaciji:

  – *Odbacivanje podataka:* Neispravni bajtovi se skladište u `discarded_buffer`.
  – *Pozivanje callback funkcije:* Kada `discarded_buffer` dostigne maksimalnu dužinu, poziva se `discardedCallback()` metoda za dalje rukovanje odbačenim podacima.
  – *Resetovanje stanja:* Interni brojači i flagovi se resetuju kako bi se protokol vratio u početno stanje.

Protokol `PRDC_SCBP` predstavlja efikasno rešenje za serijsku komunikaciju, obezbeđujući:

- *Pouzdanost:* Korišćenjem kontrolnih suma, kontrolnih znakova i enkodiranja specijalnih znakova.
- *Jednostavnost implementacije:* Jasna struktura paketa i mehanizmi obrade.
- *Fleksibilnost:* Mogućnost prilagođavanja za različite veličine paketa i brzine prenosa.

Ova implementacija omogućava stabilnu komunikaciju između uređaja u različitim uslovima rada, što je ključno za sisteme koji zahtevaju visok nivo pouzdanosti.

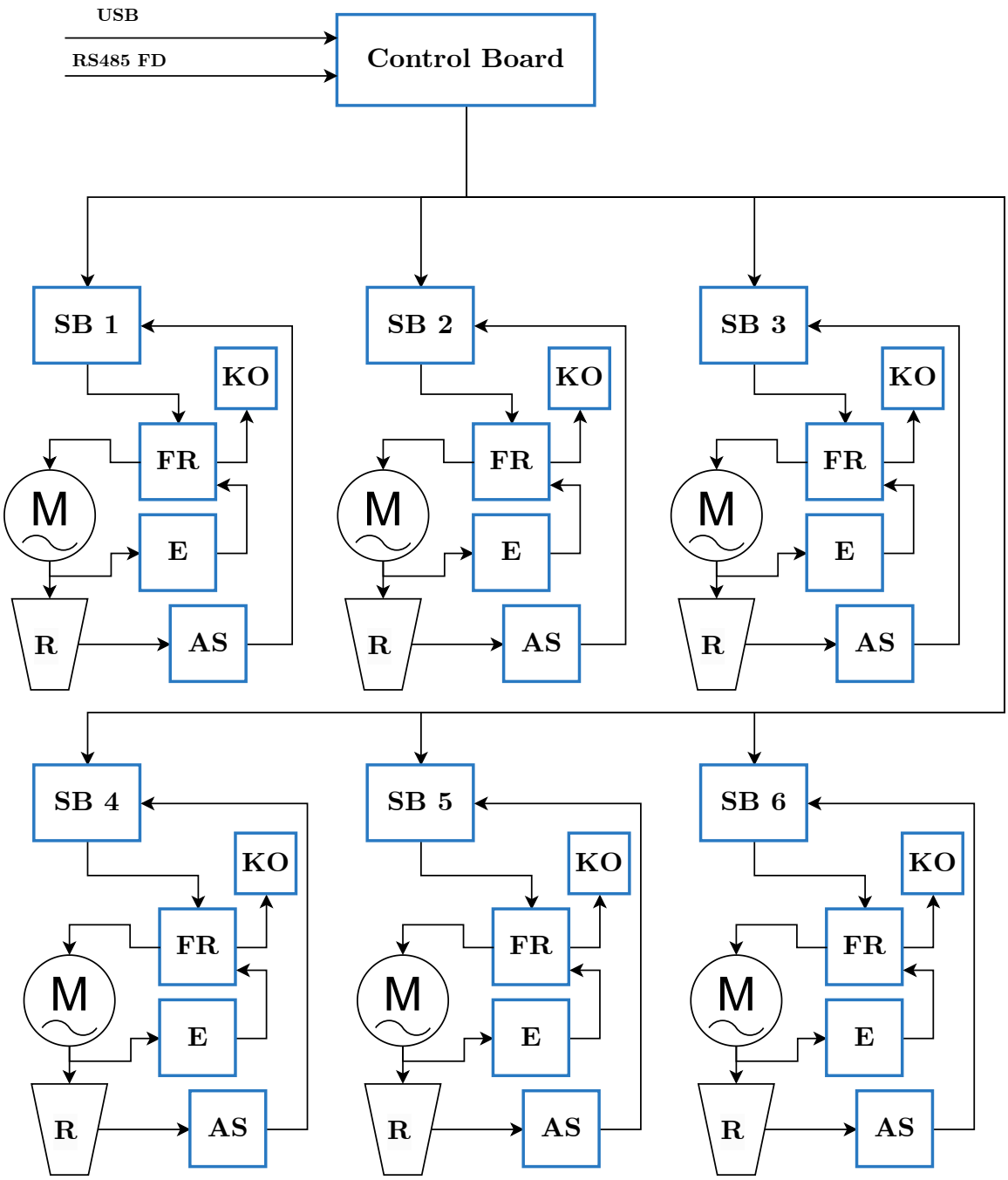
6.1.2 Prototip u stvarnoj veličini MP-6DOF-XXX

U poglavlju 5 je prikazan proračun optimalne geometrije mehanizma simulatora leta i rezultati proračuna u vidu vrednosti geometrijskih parametara su dati u tabeli 5.5 (konfiguracioni fajl je dat u prilogu R). Ovo je konačna konfiguracija prototipa u stvarnoj veličini oznake `MP-6DOF-XXX`.

Izrada prototipa u stvarnoj veličini je usledila nakon što je sve što je bilo moguće uraditi na umanjenom modelu već urađeno i nakon što su prikupljena finansijska sredstva za izradu. U maju 2019. godine su započeta prva testiranja sa aktuatorima dok je konstrukcija prototipa izrađena u novembru iste godine. Cilj izrade prototipa u stvarnoj veličini jeste realno testiranje sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

6.1.2.1 Električni sistem

Za razliku od konstrukcije koja se nije menjala od početnog rešenja, električni sistem je prošao kroz nekoliko iteracija dok nisu postignute optimalne performanse i pouzdan rad. Blok dijagram konačnog električnog sistema prototipa u stvarnoj veličini je dat na slici 6.16. Generalno gledano ne postoje značajne promene u odnosu na električni sistem umanjenog prototipa. Kao i u slučaju umenjenog prototipa, električni sistem se sastoji iz ploče zadužene za kontrolu i šest obrtnih aktuatora.



Slika 6.16 - Blok dijagram elektronike prototipa u stvarnoj veličini

Sa blok dijagrama se vidi da se ovog puta ne koriste integrisana rešenja kao što su servo motori već se aktuatori sastoje iz nekoliko komponenti, od kojih su neke specijalno projektovane i izrađene za potrebe izrade ovog prototipa. U odnosu na nabavku gotovih rešenja ovakav pristup pruža znatno veću slobodu prilikom integracije aktuatora u ostatak sistema.

Za izradu svih električnih šema i projekata štampanih ploča je korišćen programski paket *KiCad*.

6.1.2.1.1 Obrtni aktuator sa apsolutnim enkoderom

Prilikom projektovanja je za obrtne aktuatore izabrano da se koriste standardni asinhroni (indukcioni) elektromotori sa kavezom koji se upravljaju pomoću frekventnog regulatora. Indukcioni elektromotori i frekventni regulatori su najrasprostranjeniji u industriji i zato su

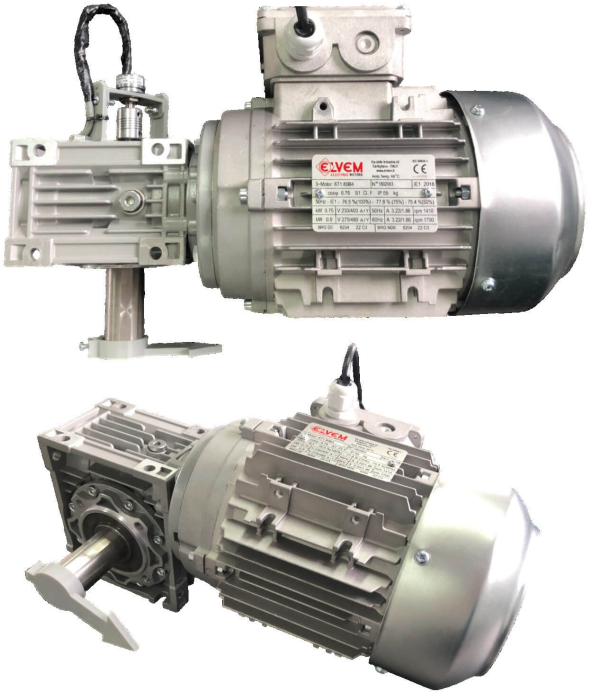
veoma pouzdani i pristupačni. Ugao donje poluge je za potrebe upravljanja moguće meriti različitim tipovima senzora ugla kao što su enkoderi, potenciometri ili magnetni beskontaktni senzori ugla zasnovani na *Holovom efektu*.

Sa blok dijagrama elektronike sistema se vidi da se svaki aktuator sastoji iz:

- **SB** - Elektronika `Servo Board` - zadužena za komunikaciju sa glavnom elektronikom za kontrolu (`Control Board`) i generisanje odgovarajućeg upravljanja za elektromotor.
- **FR** - Frekventni regulator - zadužen za kontrolu ugaone brzine elektromotora.
- **KO** - Kočioni otpornik - zadužen za potrošnju generisane električne energije prilikom kočenja.
- **M** - Elektromotor - zadužen za pretvaranje električne u mehaničku energiju.
- **E** - Enkoder - zadužen za merenje ugaone brzine elektromotora u povratnoj sprezi upravljanja frekventnog regulatora.
- **R** - Reduktor - zadužen za redukciju ugaone brzine elektromotora i prenos snage na donju polugu mehanizma.
- **AS** - Senzor ugla sa `Angle Sensor` elektronikom - zadužen za merenje trenutnog ugla otklona donje poluge mehanizma za potrebe generisanja upravljanja od strane elektronike `Servo Board`.

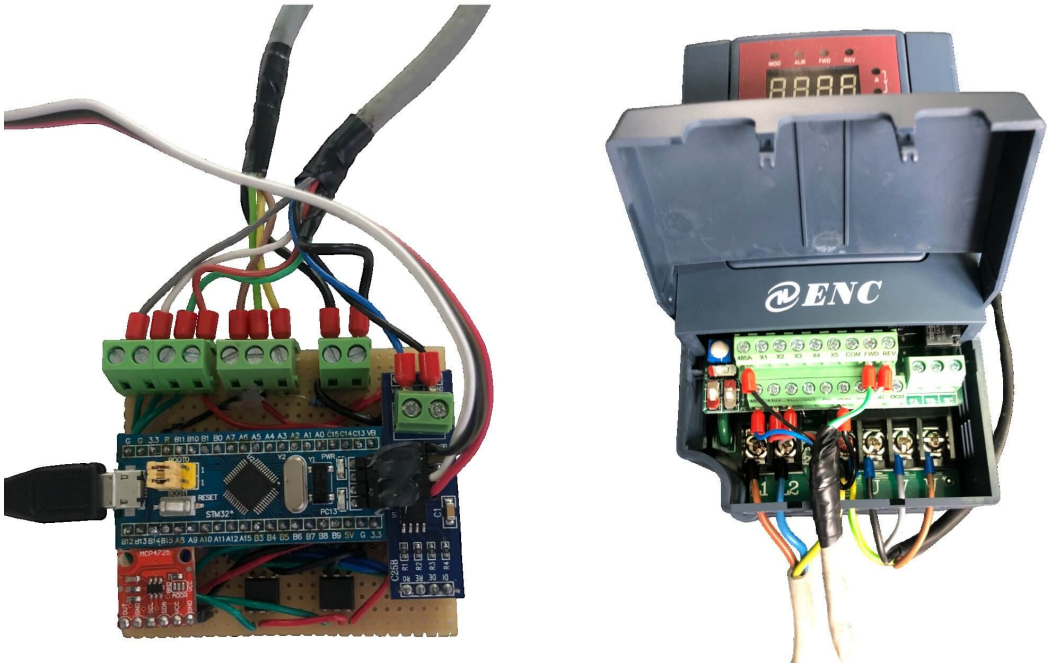
6.1.2.1.1.1 Inicijalna ispitivanja

Prvo su urađena ispitivanja sa elektronikom napravljenom od pojedinačnih modula i sa nabavljenim apsolutnim senzorom ugla baziranim na *Holovom efektu* koji daje analogni izlaz. Postavka inicijalnog ispitivanja je prikazana na slici 6.17.



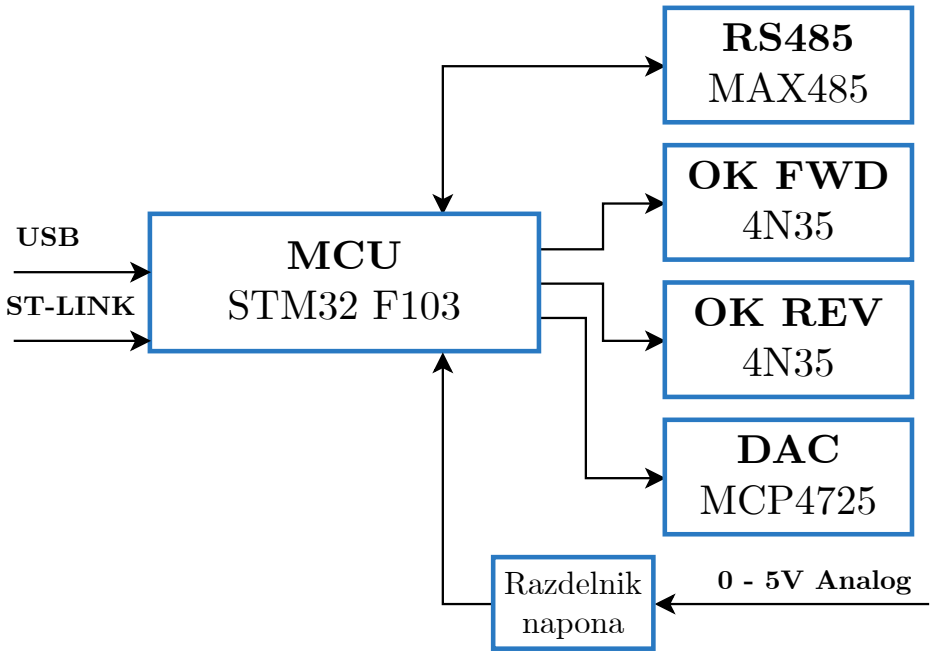
Slika 6.17 - Postavka inicijalnog ispitivanja aktuatora

Na slici 6.18 je prikazana nulta verzija elektronike `Servo Board` za kontrolu ugla aktuatora i frekventni regulator koji se koristi za upravljanje elektromotorom. Ova elektronika čita analogni izlaz sa senzora ugla i upoređuje ga sa željenim koji dobija preko `USB`-a.



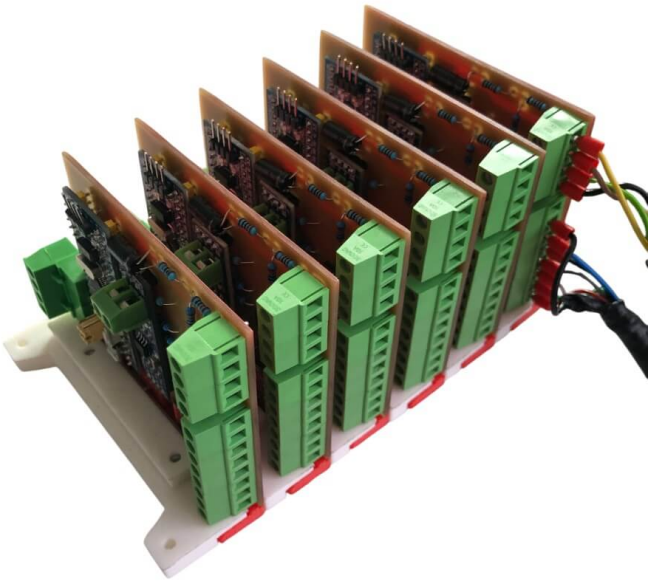
Slika 6.18 - Nulta verzija elektronike `Servo Board` (levo) i frekventni regulator (desno)

Blok dijagram na slici 6.19 prikazuje sve delove elektronike `Servo Board V0`. Integrisano kolo `MAX485` se koristi za `Polu-dupleks RS485` komunikaciju sa frekventnim regulatorom (putem ove komunikacije se proverava stanje frekventnog regulatora), optokaplerima `4N35` za `FWD` i `REV` se podešava smer obrtanja elektromotora, dok se sa digitalno-analognim konverterom `MCP4725` podešava ugaona brzina. Izlazni signal korišćenog senzora ugla je od `0` do `5V` pa se koristi razdelnik napona kako bi se ovaj opseg skalirao i doveo unutar opsega od `0` do `3.3V` što je prihvatljivo za korišćeni mikrokontroler (`MCU`) `STM32 F103`.



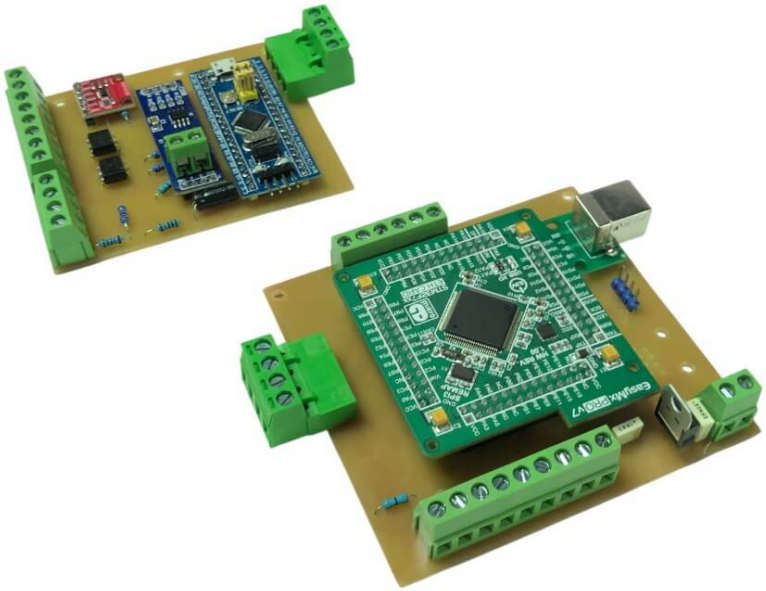
Slika 6.19 - Blok dijagram `Servo Board V0` elektronike

Nakon što je uspešno testiran aktuator sa `Servo Board V0` elektronikom napravljeno je šest `Servo Board V1` elektronika, sa jedinom izmenom što je izrađena specijalna štampana ploča i time što je dodat `UART` putem koga se šalje željena vrednost ugla. Na slici 6.20 su prikazane sve elektronike i njihov nosač.



Slika 6.20 - Šest elektronika `Servo Board V1` sa nosačem

Da bi se testiralo svih šest `Servo Board V1` elektronika, napravljena je i nulta verzija elektronike `Control Board` koja je bazirana na mikrokontroleru serije `STM32 H7` (prikazana dole desno na slici 6.21). Osim velike brzine i memorije ovaj mikrokontroler omogućava i osam zasebnih `UART` komunikacija, od kojih se šest koristi za komunikaciju sa `Servo Board V1` elektronikama.

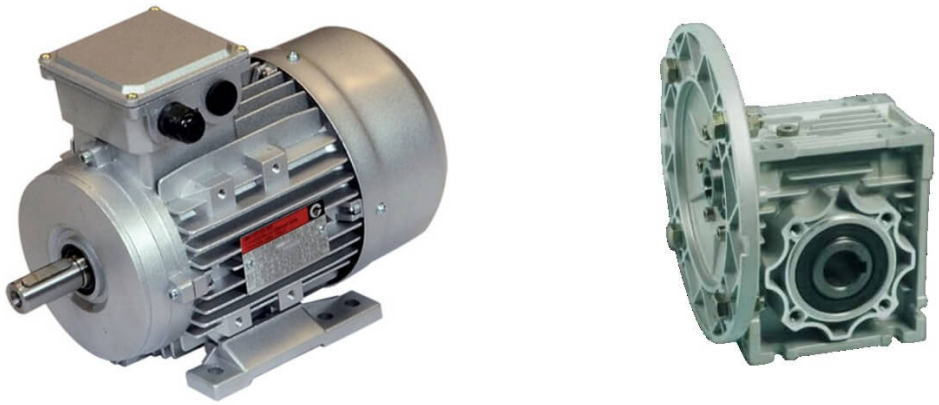


Slika 6.21 - Elektronika `Control Board V0` (dole desno)

Nakon povezivanja se pokazalo da je moguće ostvariti jedan ovakav sistem sa sedam mikrokontrolera i njihovim međusobnim komunikacijama, odnosno da je moguće upravljati sa šest aktuatora istovremeno sa izabranim komponentama.

6.1.2.1.1.2 Elektromotor i reduktor

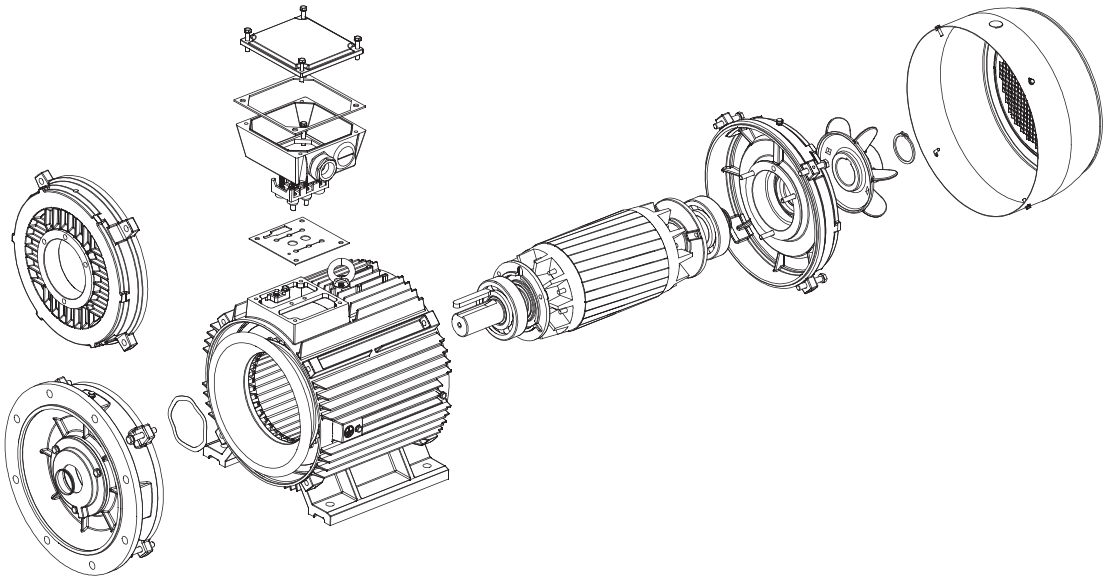
Nakon analize mehanizma simulatora leta, odlučeno je da se koriste elektromotori snage `1.5 kW` sa `1410` obrtaja u minuti, zajedno sa pužnim reduktorom prenosnog odnosa `1:50`. Izabrani su elektromotor kompanije *ELVEM* oznake `6T1 90L4 B14` i reduktor kompanije *Chiaravalli* oznake `CHM 90 50 B14` koji su prikazani na slici 6.22. I elektromotor i reduktor su proizvedeni u Italiji. Na osnovu kataloga oba proizvođača [139], [140] je zaključeno da je izabrani asinhroni elektromotor naizmenične trofazne struje u potpunosti kompatibilan sa navedenim pužnim reduktorom.



Slika 6.22 - Elektromotor i pužni reduktor izabranog tipa

Ovaj asinhroni elektromotor sastoji se od kućišta sa kutijom za priključivanje struje, statora i rotora, kako je prikazano na slici 6.23. Kućište je napravljeno od livenog aluminijuma i ima rebra koja poboljšavaju hlađenje celog motora. Na kućištu se takođe nalazi kutija za priključak električne energije, čija je svrha da omogući dovođenje struje iz izvora napajanja u stator.

Stator je cilindričnog oblika i izrađen je od više slojeva feromagnetnih limova koji su međusobno izolovani, a u njegovoj unutrašnjosti nalaze se žlebovi u koje su postavljeni bakarni namotaji. Rotor je izveden kao kavez pričvršćen na vratilo elektromotora. Kavez se sastoji od metalnih šipki koje su povezane prstenovima na oba kraja. Na rotoru je postavljen i ventilator čija je uloga da odvodi toplotu nastalu radom motora.



Slika 6.23 - Delovi elektromotora iz kataloga proizvođača (preuzeto iz [140])

Prednosti ovog tipa elektromotora su pouzdanost u radu, jednostavnost konstrukcije, lako održavanje kao i pristupačna cena. Glavni razlog odabira navedenog pužnog reduktora je veliki prenosni odnos pomoću čega se ostvaruje odgovarajuća brzina i odgovarajući obrtni moment na izlaznom vratilu.

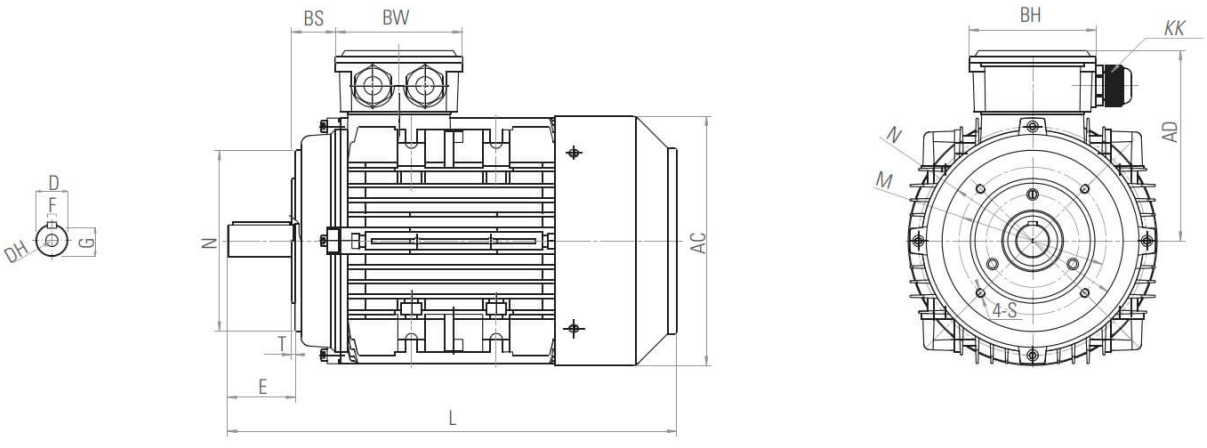
Kada se elektromotor priključi na trofaznu naizmeničnu struju putem priključne kutije na kućištu, struja protiče kroz statorske namotaje i stvara rotirajuće elektromagnetno polje. Rotor nema zasebno napajanje – u njemu se beskontaktno indukuje struja, usled dejstva elektromagnetnog polja statora. Indukovana struja dalje generiše sopstveno elektromagnetno polje, koje u sadejstvu sa rotirajućim poljem statora dovodi do okretanja vratila elektromotora.

Brzina rotacije rotora se ne poklapa sa brzinom rotirajućeg elektromagnetnog polja statora, što je razlog za naziv „asinhroni elektromotor“. Drugim rečima, rotor ne može dostići brzinu obrtnog polja statora, jer bi u tom slučaju prestala indukcija elektromagnetne sile u rotoru i ne bi se stvarao obrtni moment. Zaostajanje rotora u odnosu na polje statora naziva se klizanje rotora i tipično iznosi 2–10%. Kada elektromotor nije priključen na radnu mašinu, rotor prevazilazi samo otpore usled trenja u ležajevima i otpor sredine, te je njegova brzina vrlo bliska brzini obrtnog polja statora. Karakteristike ovog elektromotora prikazane su u tabeli 6.3.

Tabela 6.3 - Tehničke karakteristike elektromotora ELVEM 6T1 90L4 B14

Karakteristika	**Vrednost**
Snaga	1.5 kW
Broj obrtaja	1410 o/min
Frekvencija	50 Hz
Broj pari polova	2
Stepen iskorišćenosti	81 %
$\cos\phi$	0.78
Obrtni moment	10.16 Nm
Masa	15.1 kg

Na slici 6.24 su prikazane a u tabeli 6.4 su date vrednosti geometrijskih karakteristika ovog elektromotora u milimetrima.



Slika 6.24 - Geometrijske karakteristike elektromotora (preuzeto iz [140])

Tabela 6.4 - Vrednosti geometrijskih karakteristika elektromotora ELVEM 6T1 90L4 B14

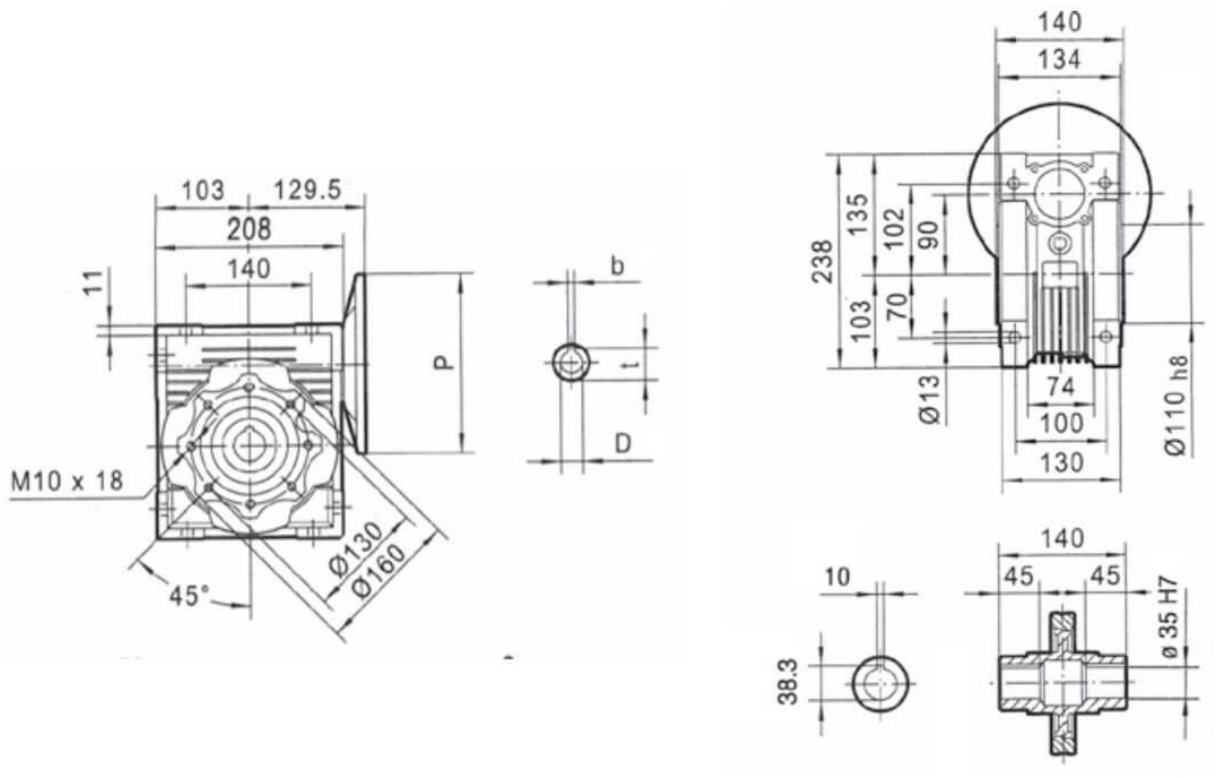
P	M	N	S	T	AC	AD	D	DH	E	F	G	H	KK	L	BS	BW	BH
140	115	95	M8	3	185	140	24	M8x20	50	50	20	90	M20x1.5	337	26	105	150

U tabeli 6.5 su date najznačajnije tehničke karakteristike izabranog pužnog reduktora.

Tabela 6.5 - Tehničke karakteristike reduktora CHM 90 50 B14

Karakteristika	**Vrednost**
Prenosni odnos	50
Snaga na ulazu	1.5 kW
Izlazni broj obrtaja	28 o/min
Izlazni obrtni moment	367 Nm
Masa	13 kg

Na slici 6.25 i u tabeli 6.6 su prikazane vrednosti geometrijskih karakteristika ovog reduktora u milimetrima.



Slika 6.25 - Geometrijske karakteristike reduktora (preuzeto iz [139])

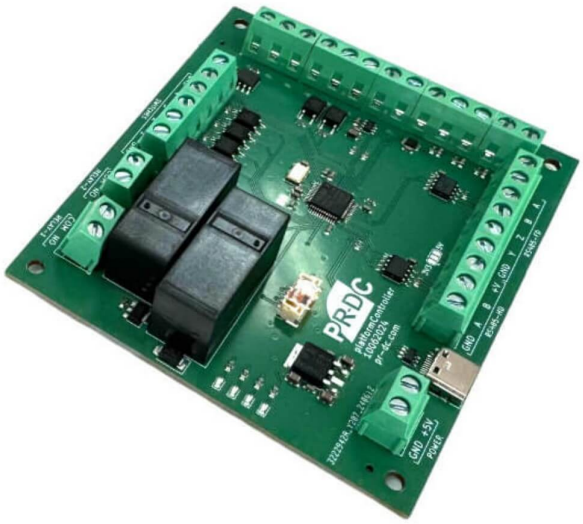
Tabela 6.6 - Vrednosti geometrijskih karakteristika reduktora CHM 90 50 B14

P	D	b	t
140	24	8	27.3

Proizvođači elektromotora i reduktora najčešće obezbeđuju i 3D modele koji se mogu koristiti prilikom projektovanja kako bi se obezbedilo odgovarajuće vezivanje sa ostatkom sistema i ostavilo dovoljno prostora. Prirubnice `B14` su izabrane baš u cilju smanjivanja prostora koji je potreban za ugradnju aktuatora.

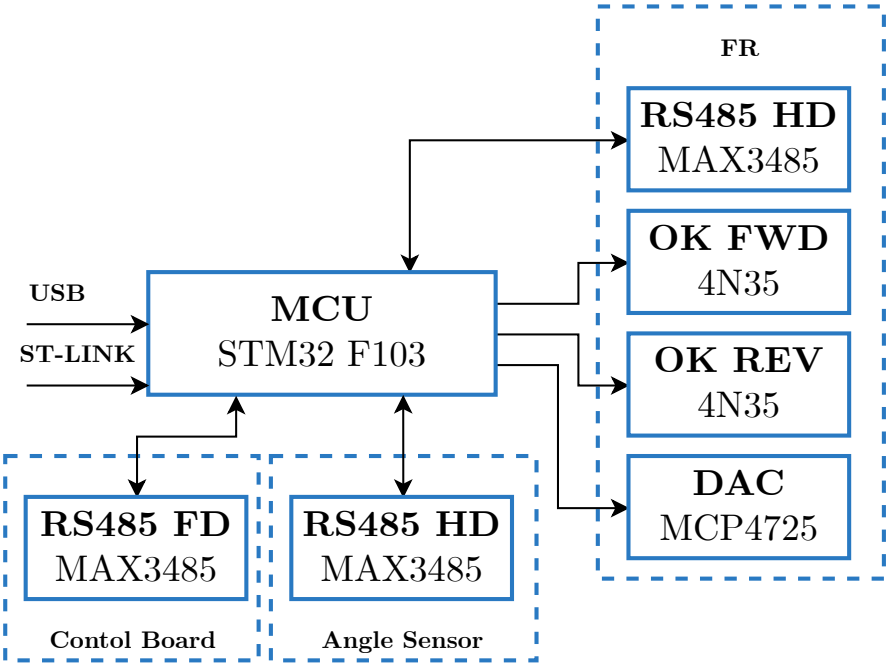
6.1.2.1.1.3 Elektronika Servo Board

U odnosu na prethodne dve verzije, finalna verzija elektronike `Servo Board` (prikazana na slici 6.26) je napravljena za nešto opštiji slučaj i kako bi potencijalno mogle da se iskoriste sve mogućnosti mikrokontrolera. Takođe, ova verzija je predviđena da se montira direktno na frekventni regulator koji koristi za upravljanje elektromotorom. Montažom direktno na frekventni regulator se skraćuju žice što je značajno u slučaju analognog signala koji se kao izlaz koristi za zadavanje ugaone brzine aktuatora.



Slika 6.26 - Elektronika `Servo Board V2`

Na slici 6.27 je prikazan blok dijagram `Servo Board V2` elektronike. Najveća razlika u odnosu na prethodne verzije je u tome što se sada za komunikaciju sa senzorom ugla koristi `Polu-dupleks RS485` i što se za komunikaciju sa elektronikom `Control Board` koristi `Pun-dupleks RS485`. Upotrebom protokola `RS485` je sistem postao značajno otporniji na smetnje.



Slika 6.27 - Blok dijagram `Servo Board V2` elektronike

Ova elektronika je i dalje zadužena za potpuno nezavisnu kontrolu jednog elektromotora preko frekventnog regulatora a na osnovu senzora ugla `Angle Sensor` i ulazne informacije o željenom uglu otklona donje poluge. Na ovaj način je, sa šest zasebnih elektronika, omogućeno paralelno izvršavanje programa upravljanja aktuatorom za svaku nogu mehanizma.

6.1.2.1.2 Frekventni regulator sa kočionim otpornikom i enkoderom

Frekventni regulator (FR) je elektronski uređaj koji omogućava upravljanje brzinom trofaznih asinhronih motora tako što vrši pretvaranje naizmenične struje jedne frekvencije u naizmeničnu struju druge frekvencije. Ovaj uređaj je izuzetno zastupljen u industriji. Sa razvojem i modernizacijom industrije razvijali su se i usavršavali i frekventni regulatori. Tokom istorije, FR je prešao dugačak put od velikog i teškog uređaja do modernog, kompaktnog, finansijski pristupačnijeg električnog uređaja bez čijeg postojanja bi današnja industrijska postrojenja znatno drugačije izgledala.

Za prototip u stvarnoj veličini je izabran frekventni regulator proizvođača `ENC` serije `EN600` i oznake `2S0015`. Ovo je frekventni regulator za elektromotor snage 1.5 kW sa monofaznim ulazom (prikazan na slici 6.28)[141].



Slika 6.28 - Frekventni regulator ENC EN600

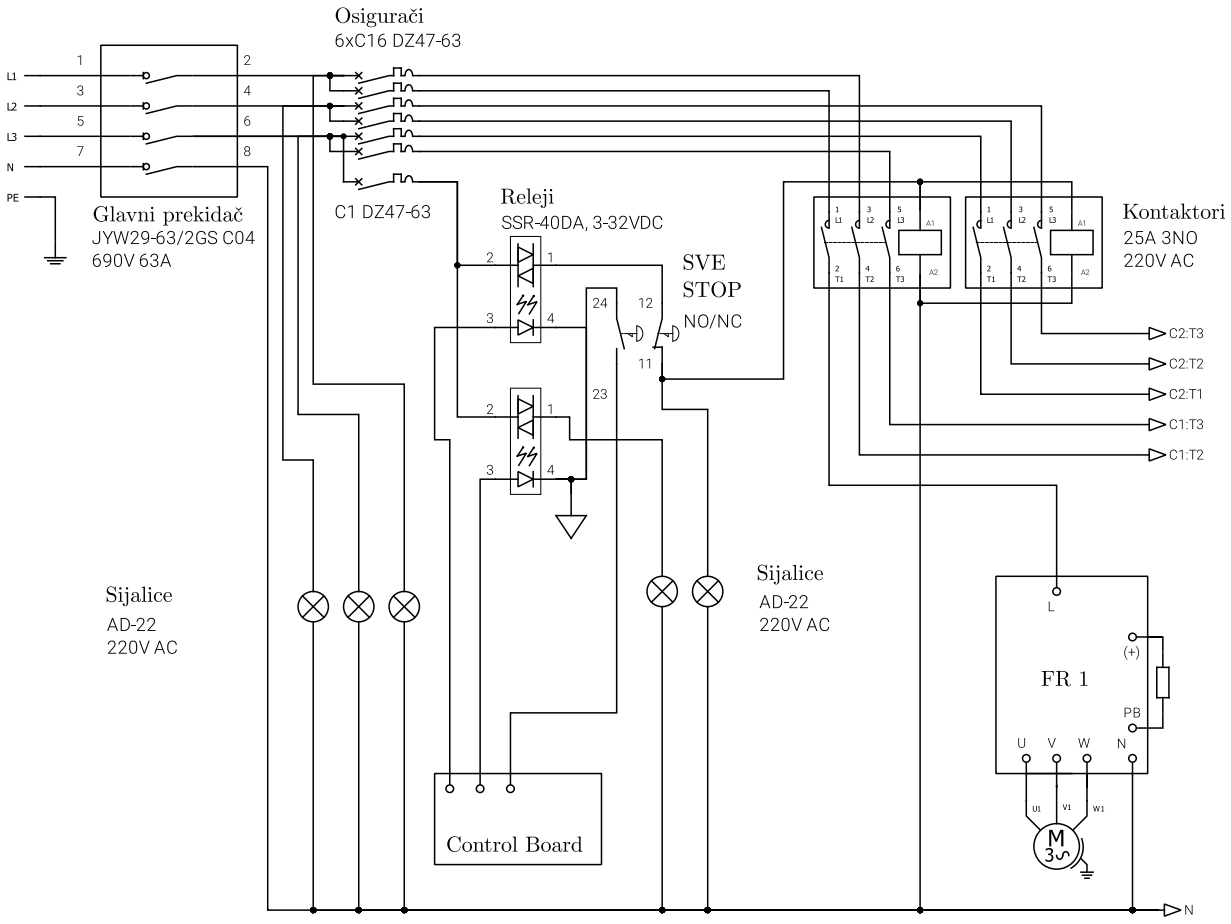
Izabran je frekventni regulator sa monofaznim ulazom kako bi se šest ovakvih uređaja podelilo na tri faze tako što su na svakoj od faza po dva elektromotora. Proizvođač frekventnog regulatora u uputstvu za korišćenje propisuje koja površina poprečnog preseka žica treba da se koristi i koje su deklarisane jačine struja za automatske osigurače i za kontaktore.

U tabeli 6.7 su date najznačajnije tehničke karakteristike izabranog frekventnog regulatora.

Tabela 6.7 - Tehničke karakteristike frekventnog regulatora ENC EN600-2S0015

Karakteristika	Vrednost
Snaga	1.5 kW
Izlazna jačina struje	7 A
Ulazni napon	220 V
Izlazni napon	3 x 220 V
Izlazna frekvencija	od 0 do 600 Hz
Dimenzije	200 mm x 115 mm x 151 mm

Na slici 6.29 je prikazana električna šema prototipa sa napajanjem prvog frekventnog regulatora preko glavnog prekidača, automatskih osigurača i kontaktora, na isti način se napajaju i preostalih pet.



Slika 6.29 - Električna šema napajanja prvog frekventnog regulatora

Kada frekventni regulator zaustavlja ili samo usporava motor, motor prelazi u generatorski režim rada, što znači da počinje da proizvodi električnu energiju umesto da je troši. Ova generisana energija mora negde da se apsorbuje da ne bi došlo do naglog porasta napona u jednosmernom kolu unutar frekventnog regulatora. Za tu svrhu je izabran aluminijumski kočioni otpornik (prikazan levo na slici 6.30), otpornosti 80 Ω i snage 400 W. Kada se unesu odgovarajuća podešavanja i kada se poveže otpornik, frekventni regulator preusmerava generisanu struju u njega kako bi on električnu energiju pretvorio u toplotu.

Kočioni otpornik sprečava preopterećenje frekventnog regulatora (i mogućnost kvara), omogućava brže i kontrolisano zaustavljanje ili usporavanje motora i povećava stabilnost sistema i produžava njegov radni vek.

Pored kočionog otpornika, performanse upravljanja se mogu značajno poboljšati korišćenjem enkodera za merenje ugaone brzine elektromotora. Na frekventni regulator se dodaje enkoderska kartica `EN-PG03` preko koje se inkrementalni enkoder povezuje sa izabranim frekventnim regulatorom. Izabran je optički inkrementalni enkoder sa `1024` pulseva po obrtaju (prikazan desno na slici 6.30).



Slika 6.30 - Kočioni otpornici (levo) i enkoder (desno)

Optički inkrementalni enkoderi šalju svetlost sa LED kroz precizno napravljene otvore na disku koji je kruto vezan sa vratilom motora. Prijemnici (fototranzistori) detektuju svetlost kao signal i pretvaraju ga u odgovarajući električni impuls koji se potom dalje kondicionira.

Enkoderi generišu osnovne pravougaone impulse koji se koriste za upravljanje motorima. Brojanjem ovih impulsa moguće je precizno odrediti položaj motora ili izračunati njegovu brzinu. Pošto su signali sa kanala `A` i `B` pomereni u fazi, poređenjem tih signala moguće je ustanoviti pravac okretanja rotora. Kanali `A` i `B` su fazno pomereni, pa se njihovim upoređivanjem može ustanoviti smer obrtanja rotora. Na osnovu ovako određene brzine je frekventni regulator u mogućnosti da prilagodi algoritam upravljanja skoro trenutno. Ovakva zatvorena petlja upravljanja je naročito značajna pri malim ugaonim brzinama elektromotora, što je neophodno za upravljanje mehanizmom prototipa u stvarnoj veličini.

Da bi sve funkcionisalo kao što je projektovano, preostaje da se frekventni regulator podesi (i mehanički i softverski) i da se poveže žicama na odgovarajući način sa elektronikom `Servo Board`, sa kočionim otpornikom i enkoderom.

Potrebna su sledeća mehanička podešavanja izabranog frekventnog regulatora:

- Kratkospojnikom se povezuju `PW` i `COM`.
- `SW1 (AI1)` podesiti na `V`.
- `SW2 (AI2)` podesiti na `V`.
- `SW3 (AO1)` podesiti na `V`.

- SW4 (AO2) podesiti na V.

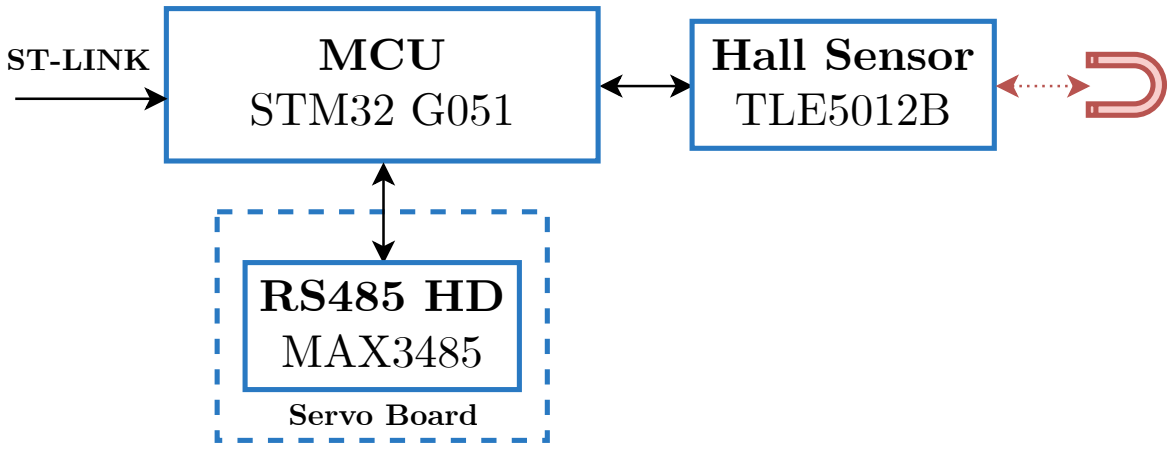
Potrebno je međusobno povezivanje sledećih terminala na Servo Board i na frekventnom regulatoru:

- A_{OUT} sa AI1.
- COM sa +24V.
- REV sa X1.
- FWD sa X2.
- GND sa GND.
- A sa 485+.
- B sa 485-.

6.1.2.1.2.1 Elektronika Angle Sensor

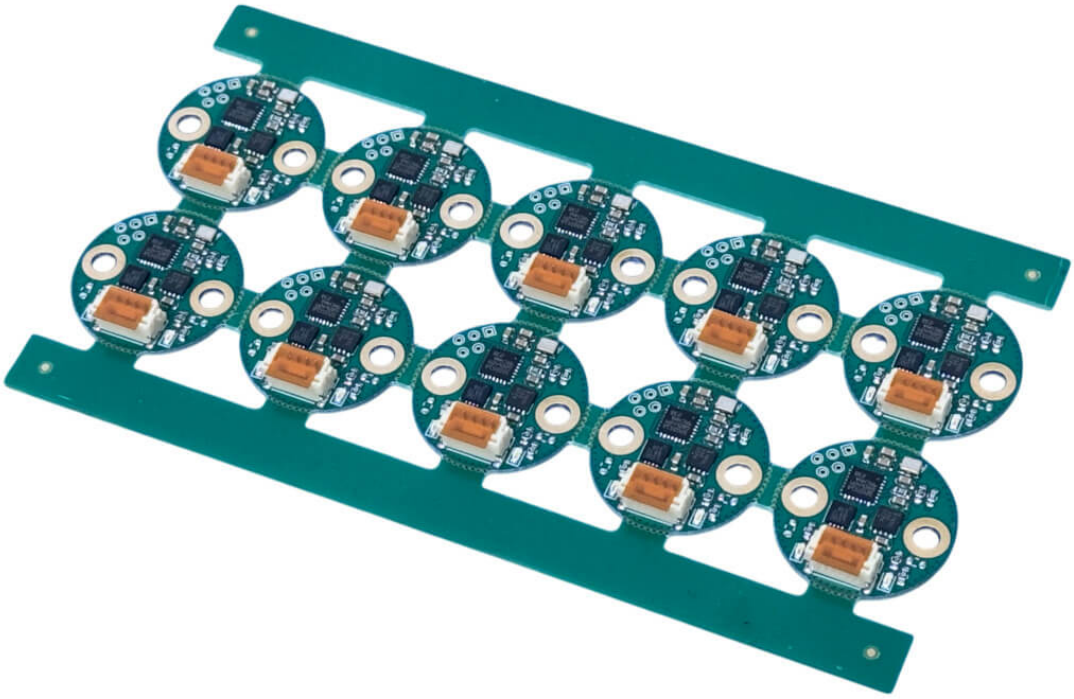
Ugao donjih poluga u okviru povratne sprege sistema upravljanja se meri pomoću magnetnih beskontaktnih senzora ugla zasnovanih na *Holovom efektu*. Ovaj tip senzora je izabran zbog preciznosti, odsustva habanja i odsustva potrebe za održavanjem.

Nakon testiranja dva komercijalno dostupna senzora je odlučeno da se ipak napravi potpuno novi, zbog zahteva ovog konkretnog projekta. Izabran je senzor kompanije *Infineon* oznake TLE5012B sa rezolucijom od 0.01°. Kao što je prikazano i na slici 6.31, senzor se precizno lemi na štampanu ploču koja se montira iznad magneta koji je čvrsto vezan za vratilo čiji ugao se meri. Za akviziciju ugla sa senzora se koristi mikrokontroler iz serije STM32 G051. Za komunikaciju sa elektronikom Servo Board se koristi Polu-dupleks RS485, što omogućava brzu i pouzdanu komunikaciju čak i uslovima velikih elektromagnetnih smetnji. Pored toga, izabrana komunikacija omogućava da se senzor nalazi i nekoliko metara dalje od elektronike sa kojom komunicira.



Slika 6.31 - Blok dijagram elektronike Angle Sensor

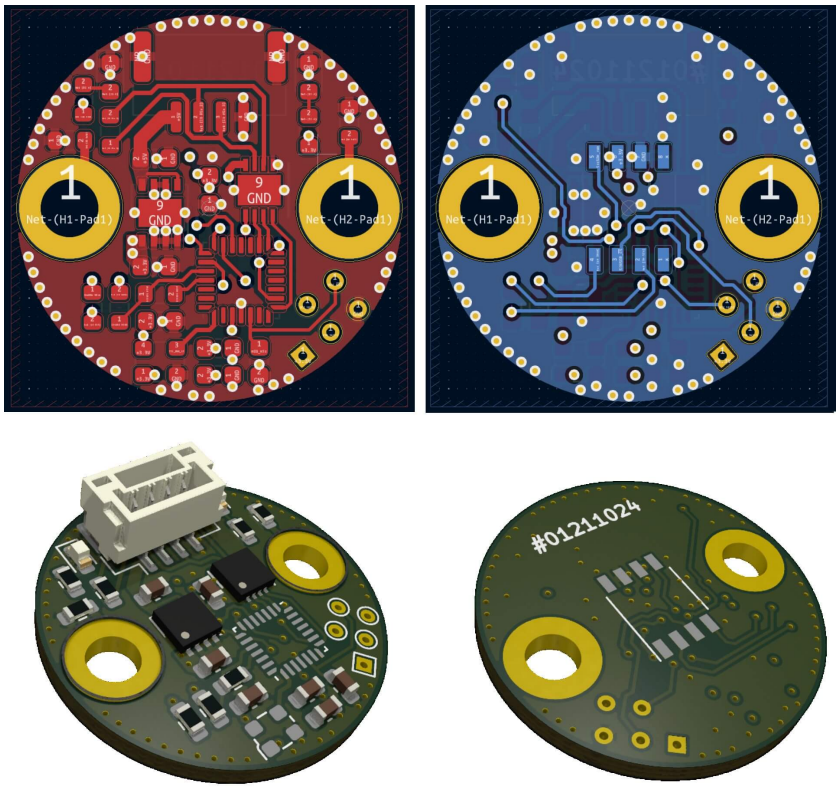
Još jedan od problema kod komercijalno dostupnih senzora je često i nepraktično povezivanje. Na slici 6.32 je prikazan panel sa 10 elektronika Angle Sensor nakon proizvodnje. Na ovoj slici se može videti i da su izabrani konektori kompanije *Molex* serije *CLIK-Mate* koji su veoma praktični zbog svoje veličine i pritom su otporni na vibracije.



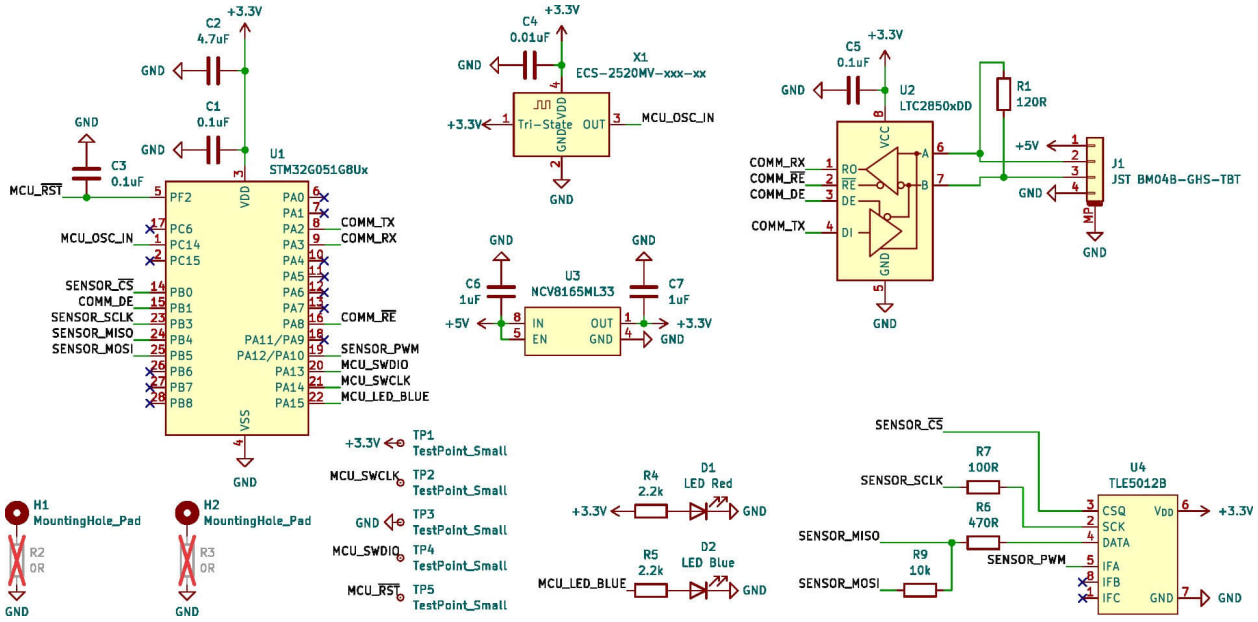
Slika 6.32 - Panel sa 10 elektronika `Angle Sensor`

Zbog potrebe za minimalnom veličinom senzora za programiranje se ne koristi standardni konektor već se za programiranje koriste pinovi sa oprugama (*pogo pin*) koji se prislone na označena mesta na štampanoj ploči.

Na slikama 6.33 i 6.34 su prikazane štampana ploča i električna šema elektronike `Angle Sensor`.



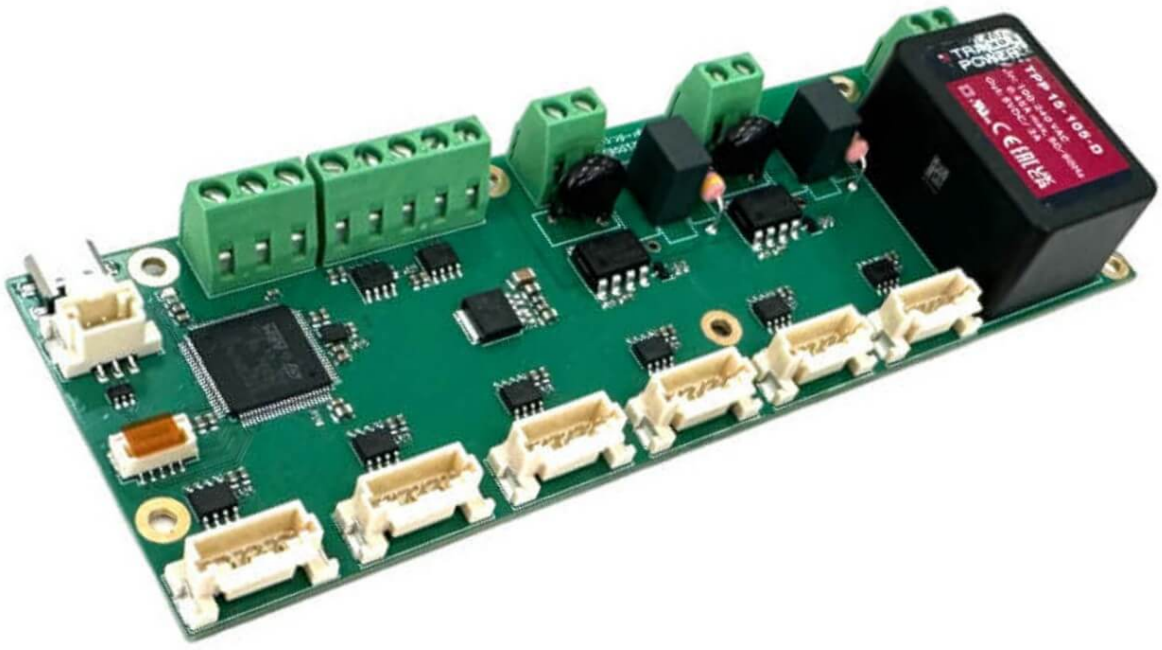
Slika 6.33 - Štampana ploča elektronike `Angle Sensor`



Slika 6.34 - Električna šema elektronike `Angle Sensor`

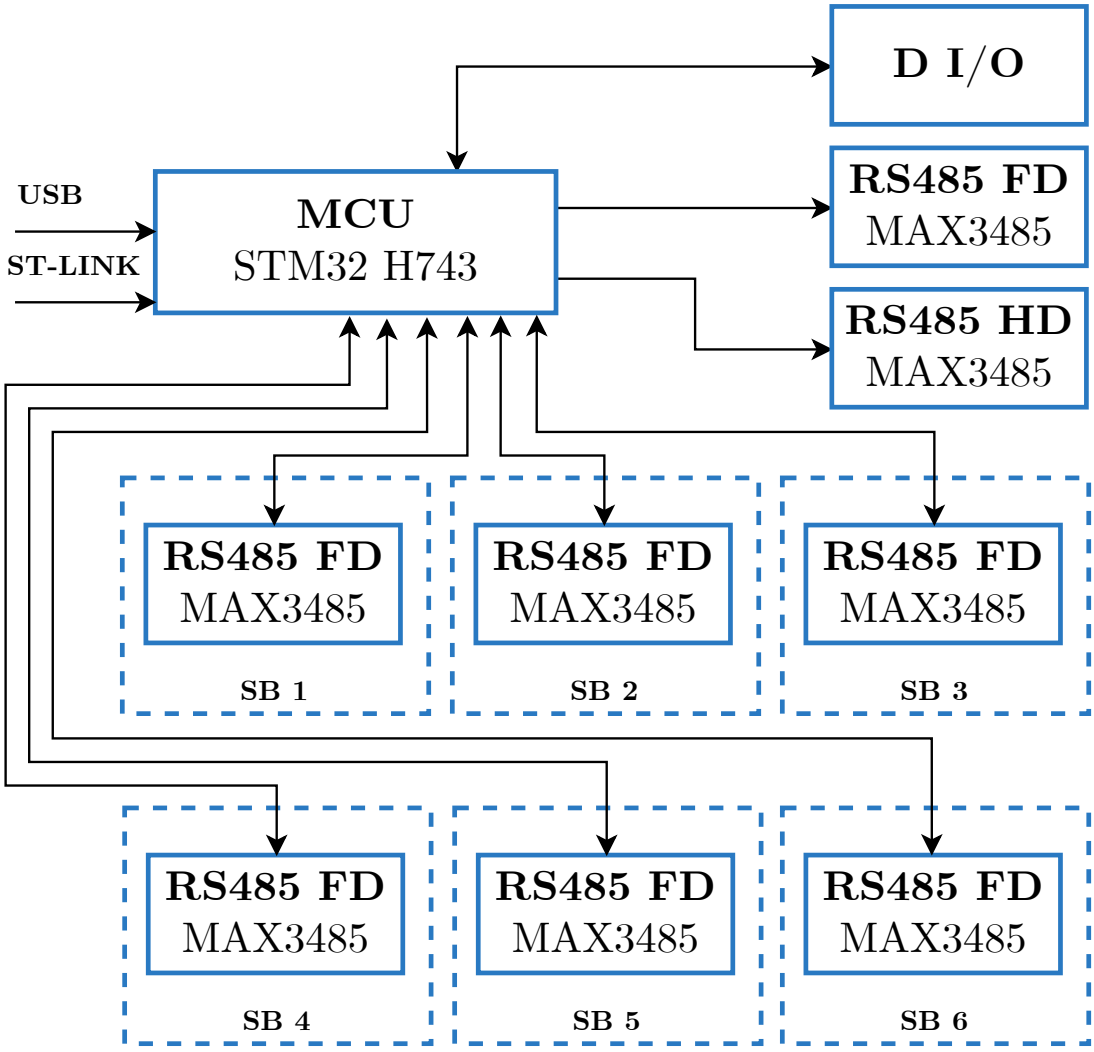
6.1.2.1.3 Elektronika Control Board

Elektronika `Control Board` (prikazana na slici 6.35) je veoma slična nultoj verziji. Izvedeno je na terminale i na konektore svih osam dostupnih `UART`-a mikrokontrolera serije `STM32 H743` od kojih se sedam koristi kao `Pun-dupleks RS485` i jedan kao `Polu-dupleks RS485`.



Slika 6.35 - Elektronika `Control Board V1`

Na slici 6.36 je prikazan blok dijagram elektronike. Ona preko jedne od komunikacija (`USB` ili `RS485`) dobija podatak o željenom kretanju mehanizma, proračunava potrebno kretanje poluga i zatim željeni ugao šalje elektronikama `Servo Board`. Osim toga, ova elektronika je zadužena i za sve sigurnosne mere, indikaciju, kao i za kontrolu napajanja frekventnih regulatora preko kontaktora.



Slika 6.36 - Blok dijagram elektronike `Control Board V1`

6.1.2.2 3D model kinematike i novi FreeCAD radni prostor ExternalAPI

Nakon definisanja geometrije sa 12 geometrijskih parametara i nakon čitavog procesa proračuna optimalnih parametara koji je prikazan u ovoj disertaciji, neophodno je odgovarajuće 3D modelirati mehanizam tako da mehanizam zapravo na kraju stvarno ima željene vrednosti geometrijskih parametara. Naročito zbog parametara koji definišu određene uglove, ovaj postupak može biti veoma zahtevan i zbog toga je mogućnost za grešku velika.

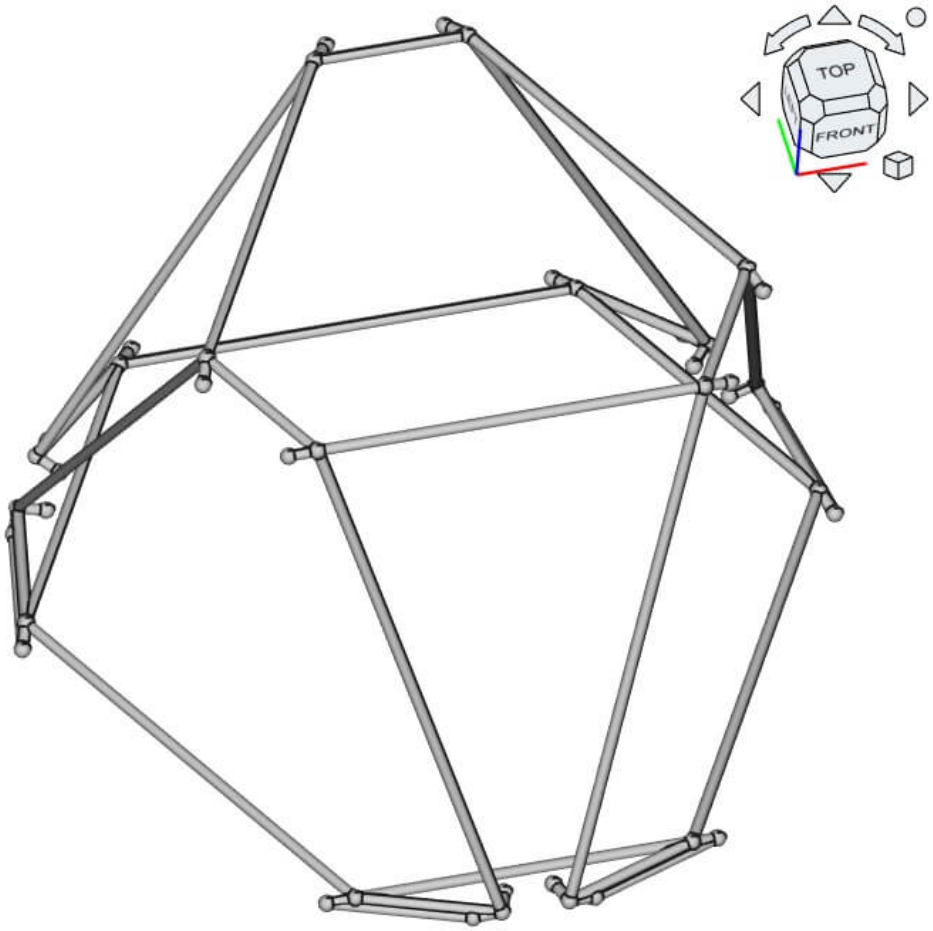
Kako bi se mogućnost za grešku prilikom 3D modeliranja mehanizma svela na minimum napravljen je novi radni prostor `ExternalAPI` (ikonica radnog prostora je prikazana na slici 6.37) za *FreeCAD* programski paket za trodimenzionalno modeliranje, koji omogućava komunikaciju sa *FreeCAD* aplikacijom preko `TCP` servera. Kompletna implementacija ovog radnog prostora je data u prilogu V.



Slika 6.37 - Ikonica FreeCAD ExternalAPI radnog prostora

Ovaj radni prostor je implementiran kroz više `Python` skripti koje zajedno omogućavaju interakciju sa *FreeCAD*-om spolja, omogućavajući automatsko upravljanje dokumentima, izvršavanje skripti i povrat informacija. Zahvaljujući ovom radnom prostoru i funkcijama koje su integrisane u `JSLAB` moguće je proceduralno generisati CAD modele iz `JSLAB` skripti.

U prilogu W je data `JSLAB` skripta `createModelFreeCAD.jsl` kojom se proceduralno direktno generiše kinematski model mehanizma na osnovu usvojene konfiguracije (vrednosti geometrijskih parametara). Ovako dobijen model za mehanizam prototipa u stvarnoj veličini `MP-6DOF-XXX` je prikazan na slici 6.38.



Slika 6.38 - 3D model kinematike mehanizma

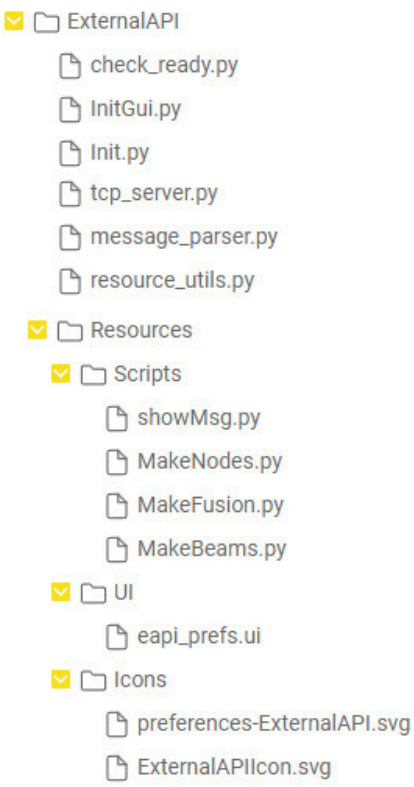
Ovakav model se potom koristi tokom modeliranja kako bi se osiguralo da su zadovoljeni svi geometrijski parametri. Pored toga, ovaj model se može koristi za verifikaciju konfiguracije jer je moguće precizno merenje i analiza, ali i za integraciju mehanizma sa ostalim sistemima u fazi konceptualnog dizajna.

6.1.2.2.1 Instalacija i upotreba radnog prostora ExternalAPI

Da bi se instalirao `ExternalAPI` radni prostor u *FreeCAD*-u, potrebno je:

1. Kopirati direktorijum radnog prostora (struktura direktorijuma je prikazana na slici 6.39) u `Mod` folder *FreeCAD*-a. Na primer, na *Windows*-u je ova putanja obično:

   ```
   %APPDATA%\FreeCAD\Mod\
   ```



Slika 6.39 - Struktura direktorijuma FreeCAD ExternalAPI radnog prostora

2. Ponovo pokrenuti *FreeCAD* kako bi prepoznao novi radni prostor.
3. U *FreeCAD*-u otvoriti `Tools` meni, zatim odabrati `Customize` i proveriti da li je `ExternalAPI` radni prostor dostupan.

Radni prostor `ExternalAPI` sastoji se od sledećih glavnih komponenti:

- `Init.py`: Glavna inicijalizaciona skripta koja se izvršava prilikom pokretanja *FreeCAD*-a.
- `InitGui.py`: Skripta koja se izvršava kada se *FreeCAD* pokreće u `GUI` modu. Ona dodaje stranicu za podešavanje i pokreće `TCP` server.
- `check_ready.py`: Proverava da li je *FreeCAD* spreman za pokretanje `TCP` servera.
- `tcp_server.py`: Implementira `TCP` server koji prihvata dolazne veze i obrađuje poruke.
- `message_parser.py`: Čita dolazne poruke i izvršava odgovarajuće akcije.
- `resource_utils.py`: Pruža pomoćne funkcije za upravljanje resursima kao što su ikonice, `UI` fajlovi i skripte.
- *Skripte*: Dodatne skripte koje se mogu izvršavati preko `TCP` servera, kao što su `MakeBeams.py`, `MakeFusion.py`, `MakeNodes.py` i `showMsg.py`.

Radni prostor omogućava izvršavanje prilagođenih skripti preko `TCP` servera. Neke od dostupnih skripti su:

- `MakeBeams.py`

  Ova skripta kreira cilindrične grede na osnovu zadatih koordinata i prečnika. Podaci se učitavaju iz `JSON` datoteke koja sadrži liste koordinata i prečnika.

- `MakeFusion.py`

  Skripta koja spaja sve vidljive objekte u jedan koristeći funkciju `Fusion`. Ovo je korisno za kombinovanje više objekata u jedan za potrebe daljeg modeliranja ili izvoza.

- `MakeNodes.py`

  Kreira sferične čvorove na osnovu zadatih koordinata i prečnika. Kao i `MakeBeams.py`, ova skripta koristi `JSON` datoteku za učitavanje podataka.

- `showMsg.py`

  Prikazuje poruku u dijalogu unutar *FreeCAD*-a. Ova skripta može biti korisna za prikazivanje informacija ili upozorenja korisniku.

Nakon instalacije radnog prostora, on se može koristiti na sledeći način:

1. Pokrenuti *FreeCAD*. `TCP` server će automatski biti pokrenut ako je omogućeno u podešavanjima.
2. Sa klijentske strane, uspostaviti vezu sa *FreeCAD*-om preko `TCP/IP` protokola koristeći odgovarajući `port`.
3. Poslati komande u formatu `COMMAND|PARAM1|PARAM2|...`. Na primer, `CMD|OPEN|path/to/file.fcstd` za otvaranje dokumenta.
4. Primati odgovore od servera koji ukazuju na uspešnost izvršene komande ili eventualne greške.

U programu `JSLAB` je napisana klasa `PRDC_JSLAB_FREECAD_LINK` koja je integrisana u sam program. Ova klasa implementira sve neophodne metode za pokretanje *FreeCAD* programa i komunikaciju sa radnim prostrom `ExternalAPI` u cilju formiranja 3D modela ili izvršavanja neke druge operacije u *FreeCAD*-u.

6.1.2.3 3D model konstrukcije i tehnička dokumentacija

Kada je mehanizam geometrijski nedvosmisleno definisan, može se preći na njegovu materijalizaciju, u smislu transformacije kinematičkog modela u trodimenzionalni model sa svim pratećim informacijama na osnovu kojih se može izraditi.

Simulator i pilotska kabina se dimenzionišu tako da mogu što bolje da ispunjavaju svoje uloge u okviru sistema. Što je detaljniji model to su i procene i analize koje se urade pomoću ovog modela približnije realnosti. Pošto se modelu dodeljuju i odgovarajući materijali od kojih se izrađuju delovi, moguće je odrediti masu i momente inercije, numerički uraditi strukturalnu analizu delova u odgovarajućim programskim paketima i analizirati približnije dinamiku sistema.

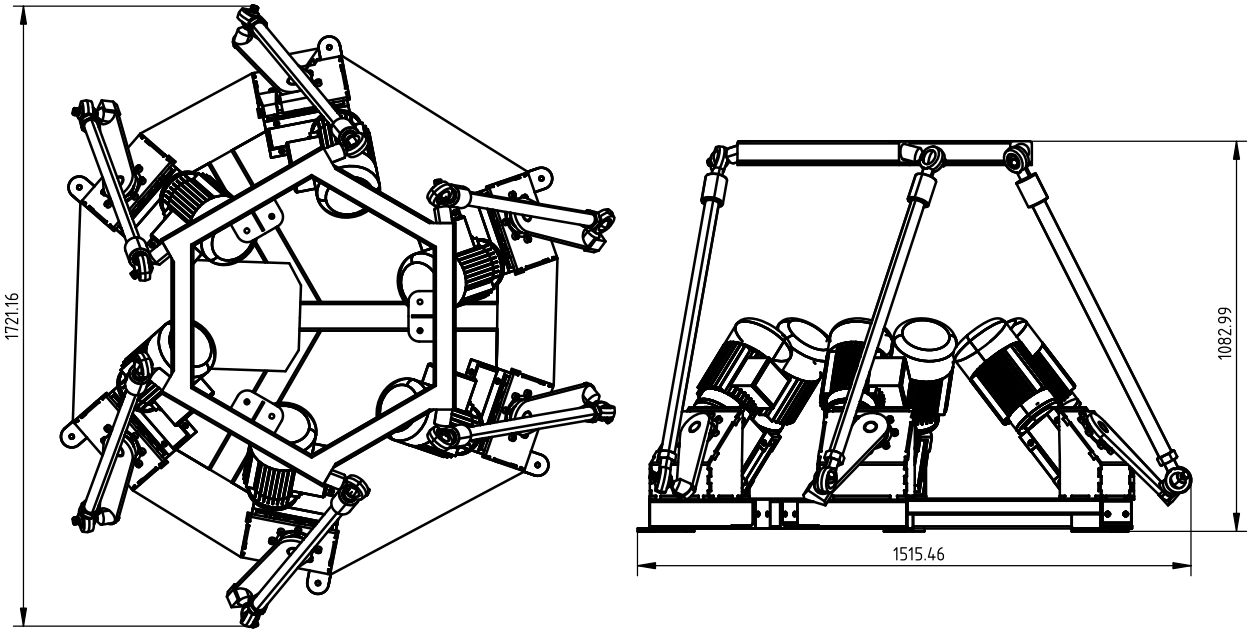
Kao i prilikom projektovanja umanjenog prototipa, projektovanje je započeto analizom raspoloživih tehnologija za proizvodnju. Odlučeno je da se maksimalno iskoristi potencijal rezanja metala laserom i zatim zavarivanje elemenata kako bi se dobila kompleksna prostorna struktura.

Nabavljeni su standardni sferni zglobovi (zglobni ležaji prikazan na slici 6.40), elektromotori i reduktori i formirani su njihovi 3D modeli. Nakon toga se prešlo na formiranje 3D modela mehanizma u skladu sa usvojenim geometrijskim parametrima.



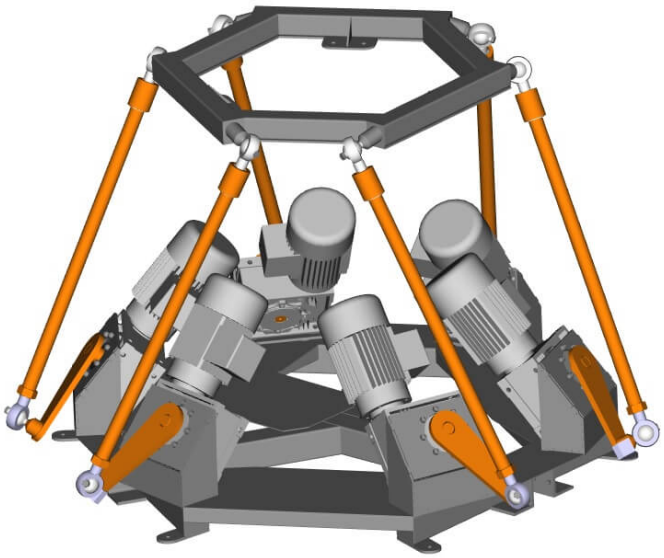
Slika 6.40 - Zglobni ležaj koji se koristi kao sferni zglob gornje poluge mehanizma

U odnosu na umanjeni model, prototip u stvarnoj veličini ima značajno kompleksniju geometriju i zato je prvo formiran kinematski 3D model koji je korišćen kao referentna baza prilikom modeliranja elemenata. Na slici 6.41 su prikazane gabaritne mere konačnog 3D modela.

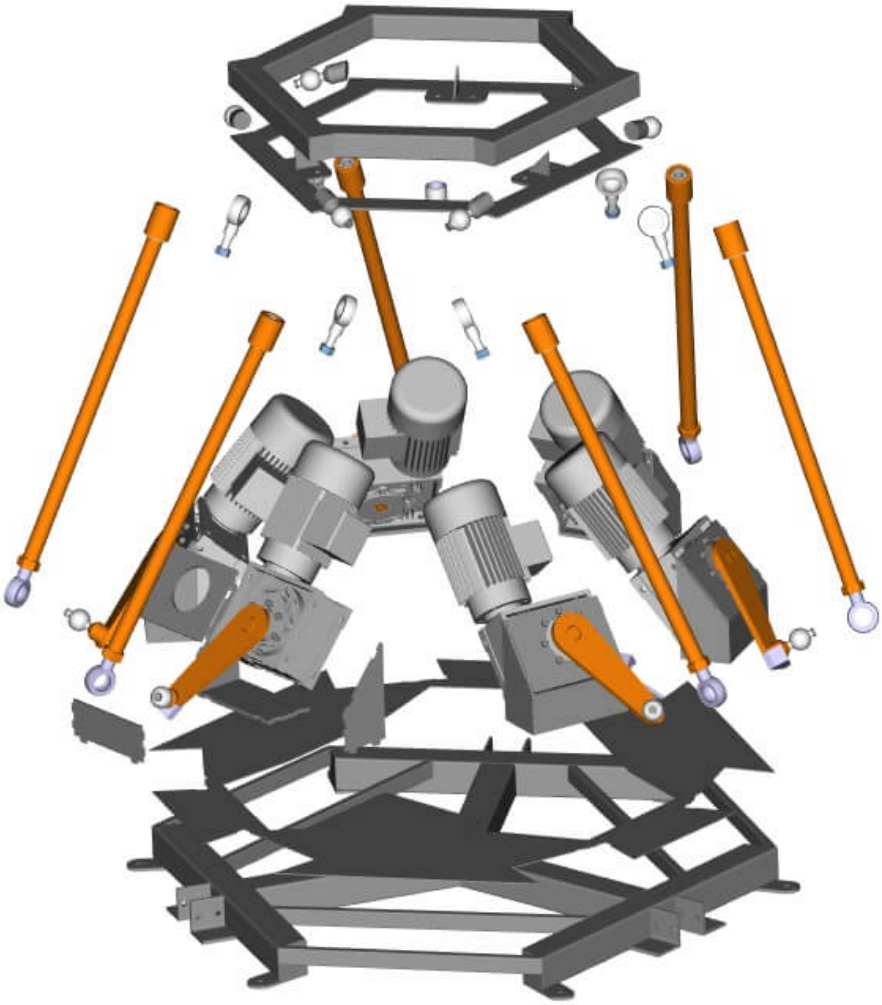


Slika 6.41 - Gabaritne mere prototipa u stvarnoj veličini

Na slici 6.42 je prikazan 3D model prototipa `MP-6DOF-XXX` u programu *FreeCAD*. Da bi se bolje prikazao ovaj sklop, na slici 6.43 je prikazan model sklopa sa razdvojenim delovima.



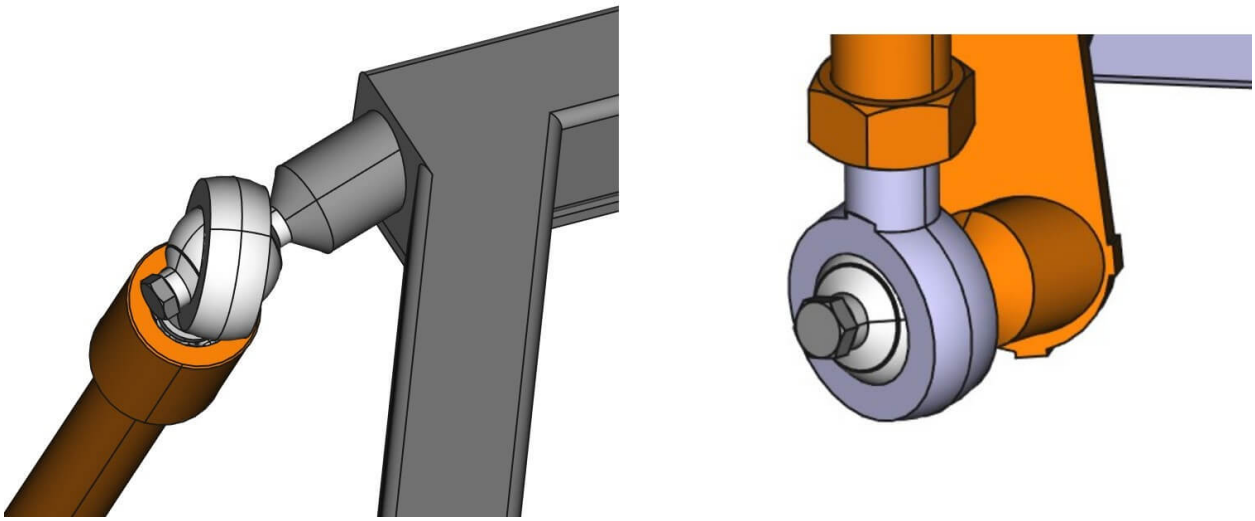
Slika 6.42 - 3D model sklopa MP-6DOF-XXX



Slika 6.43 - Razdvojeni delovi sklopa MP-6DOF-XXX

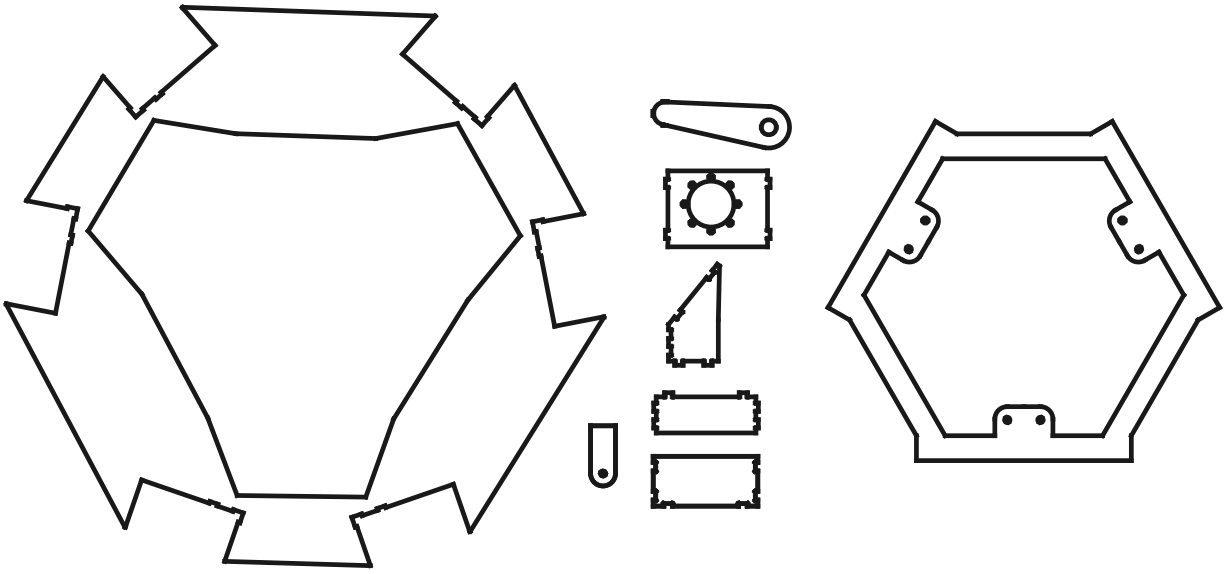
Reduktor se povezuje sa konstrukcijom na isti način kao kada se povezuje preko momentne ruke, bočno preko 8 vijaka oko otvora za vratilo, i time se osiguravaju povoljni radni uslovi za strukturu reduktora.

Specifični uglovi ugradnji sfernih zglobova se dobijaju rezanjem šipki kružnog poprečnog preseka pod određenim uglovima. U središte šipke se urezuje navoj preko kojeg se vijkom spajaju gornja poluga i platforma. Na slici 6.44 su prikazani ovi nosači zglobova. Za zavarivanje pod ovim uglovima je predviđeno da se koriste alati dobijeni 3D štampanjem.



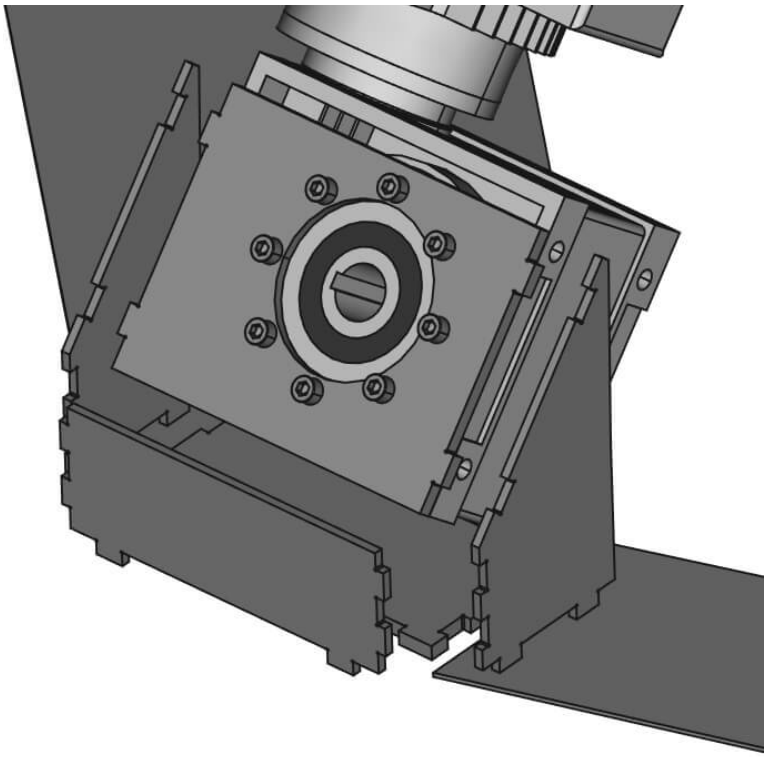
Slika 6.44 - Nosači sfernih zglobova

Na slici 6.45 su prikazani crteži najznačajnijih delova koji se dobijaju rezanjem laserom iz ploča. Na crtežima se mogu videti žlebovi koji se koriste tokom proizvodnje kako bi se značajno olakšalo međusobno pozicioniranje delova za zavarivanje. Prikazani najveći element ima zapravo funkciju da odgovarajuće pozicionira nosače aktuatora na bazi.



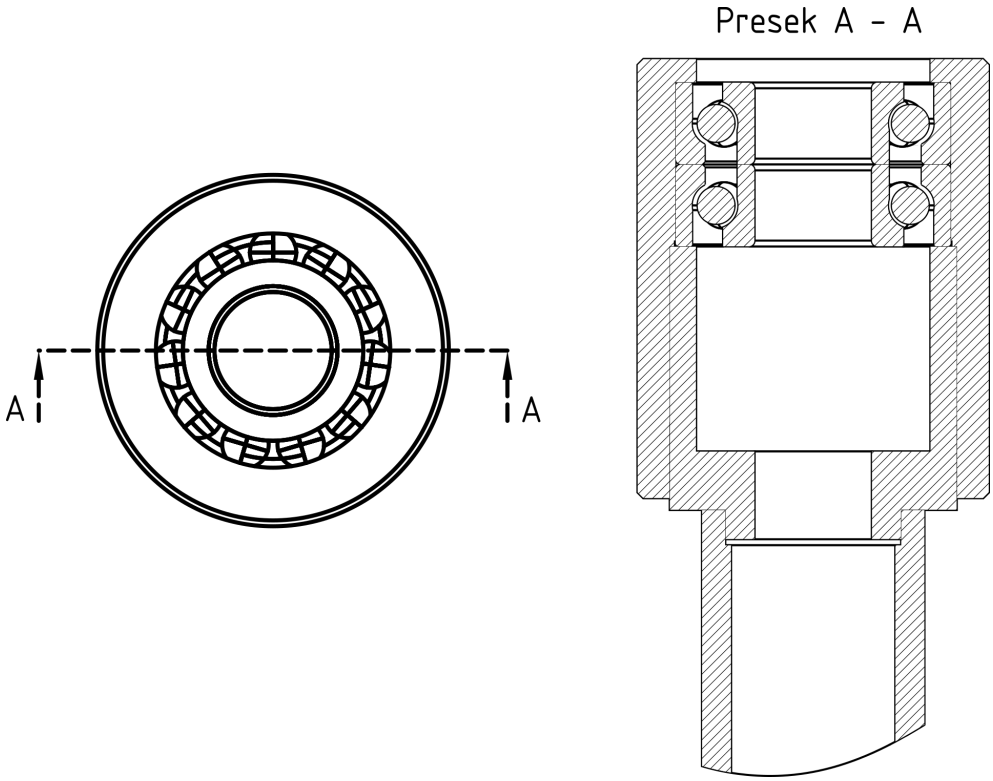
Slika 6.45 - Crteži najznačajnijih delova MP-6DOF-XXX za rezanje laserom

Nosač aktuatora, koji je rasklopljen prikazan na slici 6.46, se sastoji od pet pločastih elemenata koji se izrađuju na laseru i projektovan je tako da se jednostavno mogu prilagoditi usvojenom uglu pod kojim aktuator treba da bude orijentisan u odnosu na horizontalnu ravan.



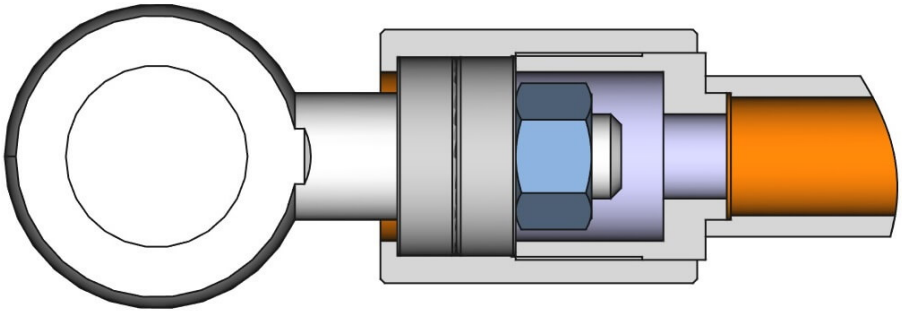
Slika 6.46 - Nosač aktuatora

Za ostale elemente se koriste standardne cevi i nosači. Gornje poluge mehanizma su cevi kružnog poprečnog preseka. Na slici 6.47 je prikazan crtež preseka dela gornje poluge koji omogućava slobodnu rotaciju jednog kraja u odnosu na drugi. Na kraj cevi se zavaruje element sa spoljnim navojem na koji se naslanjaju dva obrnuto postavljena radijalno-aksijalna ležaja. Ležaji se fiksiraju preko spoljnog elementa sa unutrašnjim navojem.



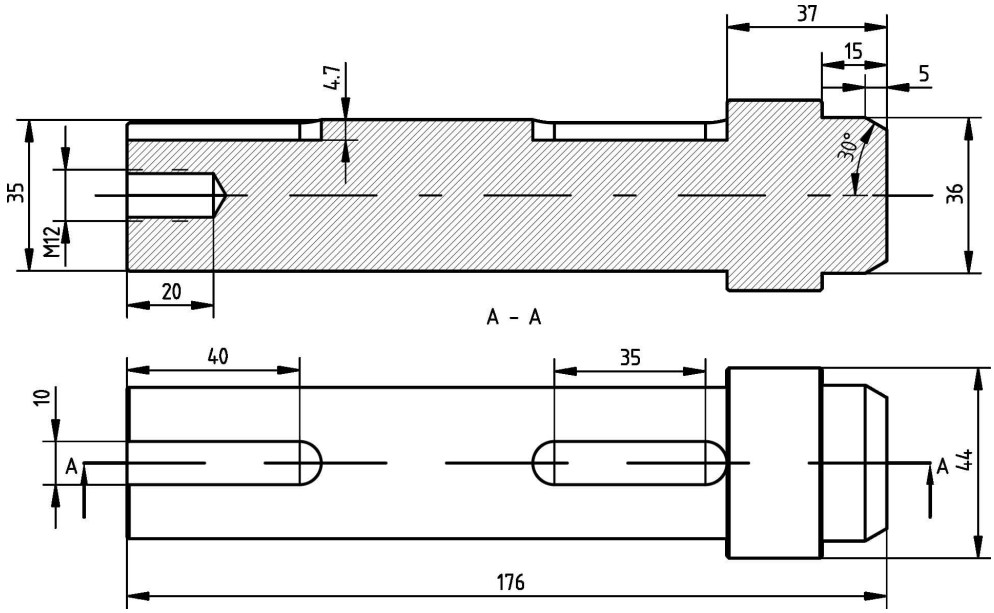
Slika 6.47 - Crteži preseka dela gornje poluge koji omogućava slobodnu rotaciju jednog kraja u odnosu na drugi

Na slici 6.48 je prikazan kompletan sklop obrtne ose sa sfernim zglobom koji se navrtkom fiksira kroz ležaje.

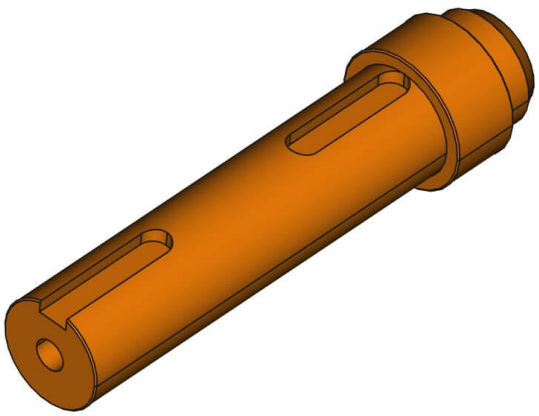


Slika 6.48 - 3D model obrtne ose sa sfernim zglobom

Tehnički crtež i 3D model vratila donjih poluga su prikazani na slikama 6.49 i 6.50. Vratilo se gotovo potpuno izrađuje na strugu od šipke kružnog poprečnog preseka, samo se na kraju na glodalici prave žlebovi za klinove. Na vratilo se sa bočne strane zavaruje poluga koja je izrađena iz pločastog materijala rezanjem laserom.

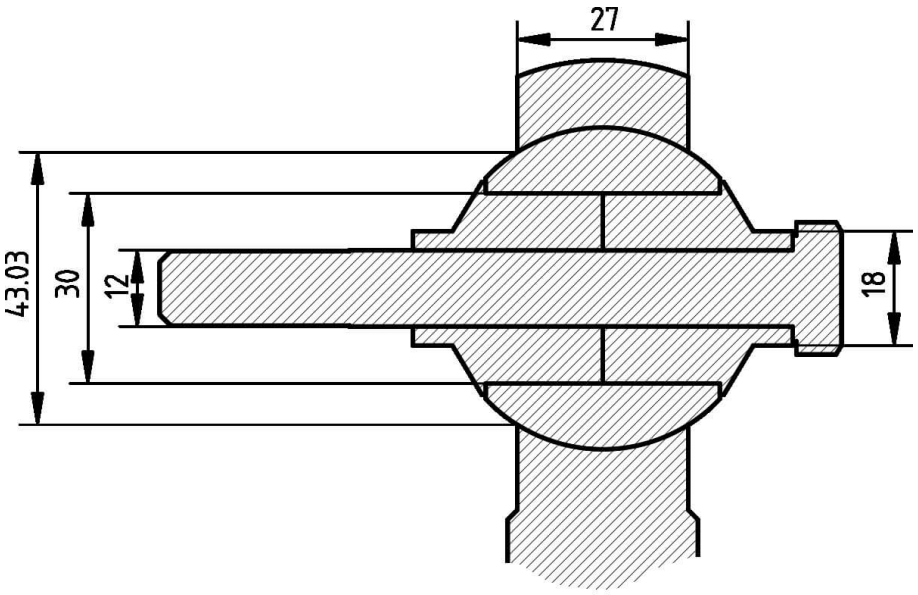


Slika 6.49 - Tehnički crtež vratila donjih poluga



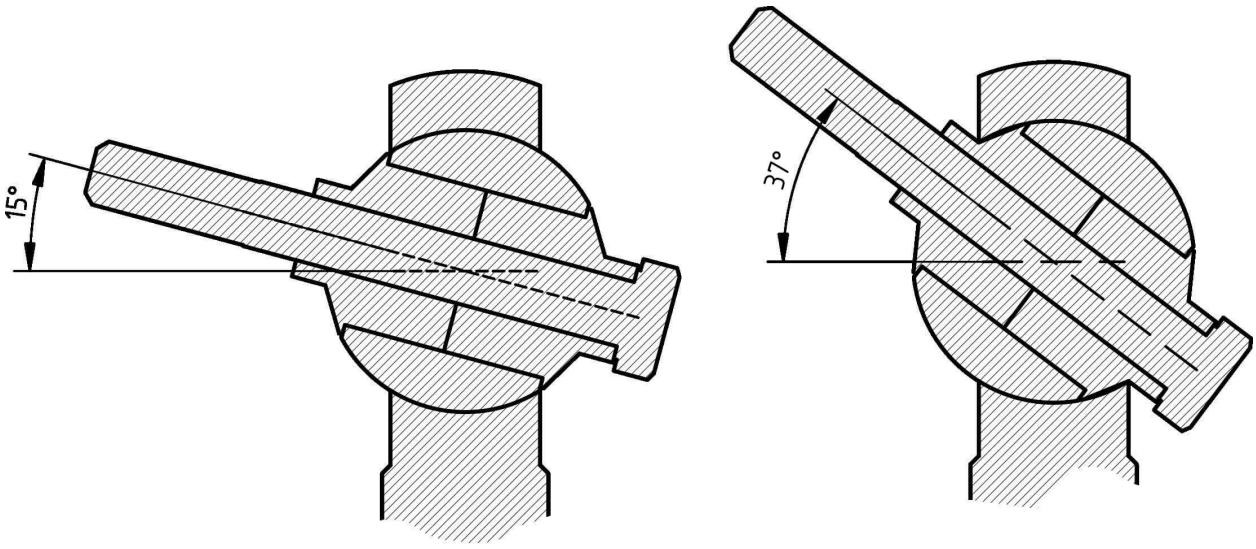
Slika 6.50 - 3D model vratila donjih poluga

Za sferni zglob je izabran zglobni ležaj SAKAC 30 (prikazan na slici 6.51) sa otvorom prečnika 30 mm koji se sa dve čaure adaptira za vijak M12 kroz koji se vrši spajanje sa ostatkom mehanizma.



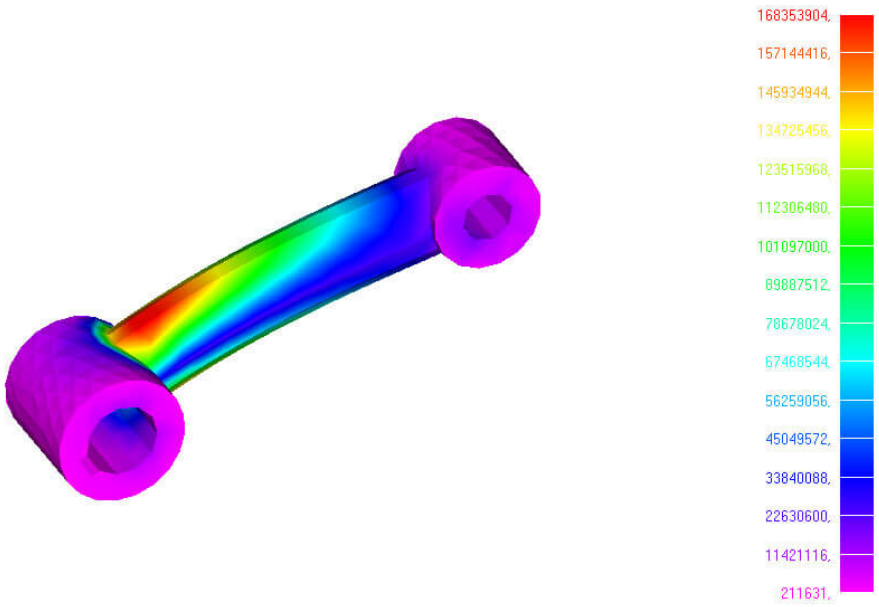
Slika 6.51 - Tehnički crtež sklopa sfernog zgloba

Zahvaljujući ovim čaurama je moguće ugaono pomeranje u zglobovima povećano sa ±15° na ±37° (vidi se na slici 6.52) što značajno utiče na veličinu radnog prostora mehanizma.



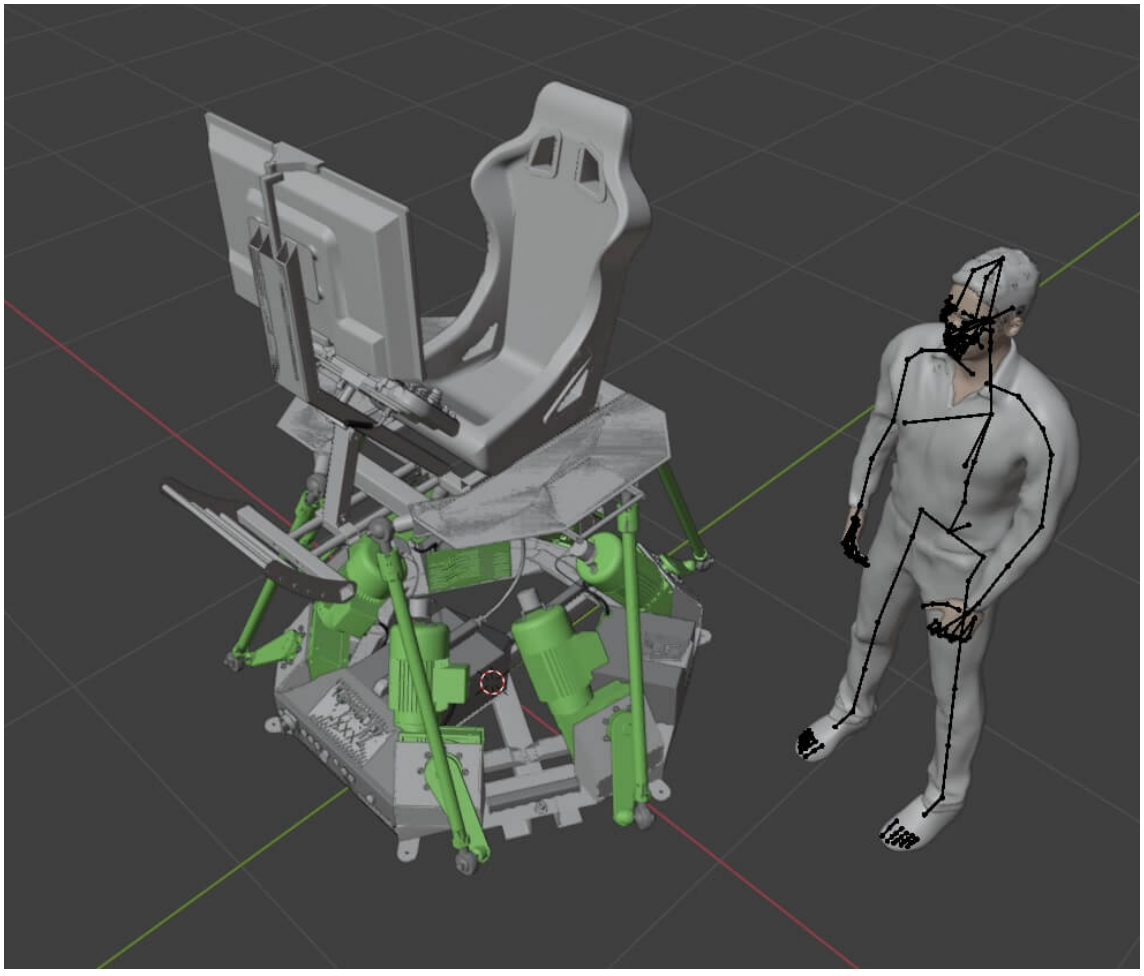
Slika 6.52 - Moguće ugaono pomeranje zgloba u zavisnosti od prečnika osovine

Jedan od nedostataka ovog tipa je što se zbog uslova rada može doći do zamora materijala donje poluge pa se prilikom projektovanja vrši i procena zamornog veka za donje poluge. Mogu se primeniti iste eksperimentalne i numeričke metode koje se koriste za procenu zamornog veka nosača krila [142]. Korišćenjem proširene metode konačnih elemenata (XFEM) moguće je numerički dobiti putanje prslina i procenu veka trajanja zamora [143]. Na slici 6.53 je prikazan primer provere čvrstoće donje poluge metodom konačnih elemenata.



Slika 6.53 - Primer provere čvrstoće donje poluge metodom konačnih elemenata

Realistični prikazi prototipa mehanizma simulatora leta sa operaterom, koji se koriste u ovoj disertaciji, su napravljeni na osnovu CAD modela pomoću programa otvorenog koda *Blender* i *MakeHuman*. Na slici 6.54 je u *Blender*-u prikazan model prototipa simulatora leta u stvarnoj veličini pored 3D modela operatera. Pored izrade realističnih prikaza ovakvi modeli se mogu koristiti i za ergonomsku analizu pilotske kabine na platformi simulatora leta.



Slika 6.54 - Model prototipa simulatora leta u stvarnoj veličini pored 3D modela operatera

6.1.2.4 Proizvodnja

Proizvodnja konstrukcije prototipa u stvarnoj veličini započeta je u novembru 2019. godine. Tokom izrade su pored klasičnih metala korišćeni najsavremeniji kompozitni materijali i korišćene su najsavremenije metode izrade na numerički upravljanim mašinama alatkama među kojima su i 3D štampači. 3D štampačima su izrađivani elementi za montažu i pozicioniranje kako bi konačna geometrija upotpunosti odgovarala konfiguraciji koja je dobijena proračunom.

Elementi dobijeni rezanjem CNC laserom su prikazani na slici 6.55, ovi elementi su potom obrušeni i očišćeni u okviru prirema za njihovo zavarivanje.



Slika 6.55 - Elementi konstrukcije prototipa u stvarnoj veličini nakon rezanja laserom

Nosači aktuatora su zapravo šest zasebnih sklopova koji se dobijaju zavarivanjem (kao što je prikazano na slici 6.56). Nakon zavarivanja se iz estetskih razloga, ali i da bi se omogućilo uklapanje sa ostalim elementima, zavareni spojevi obrađuju brušenjem.



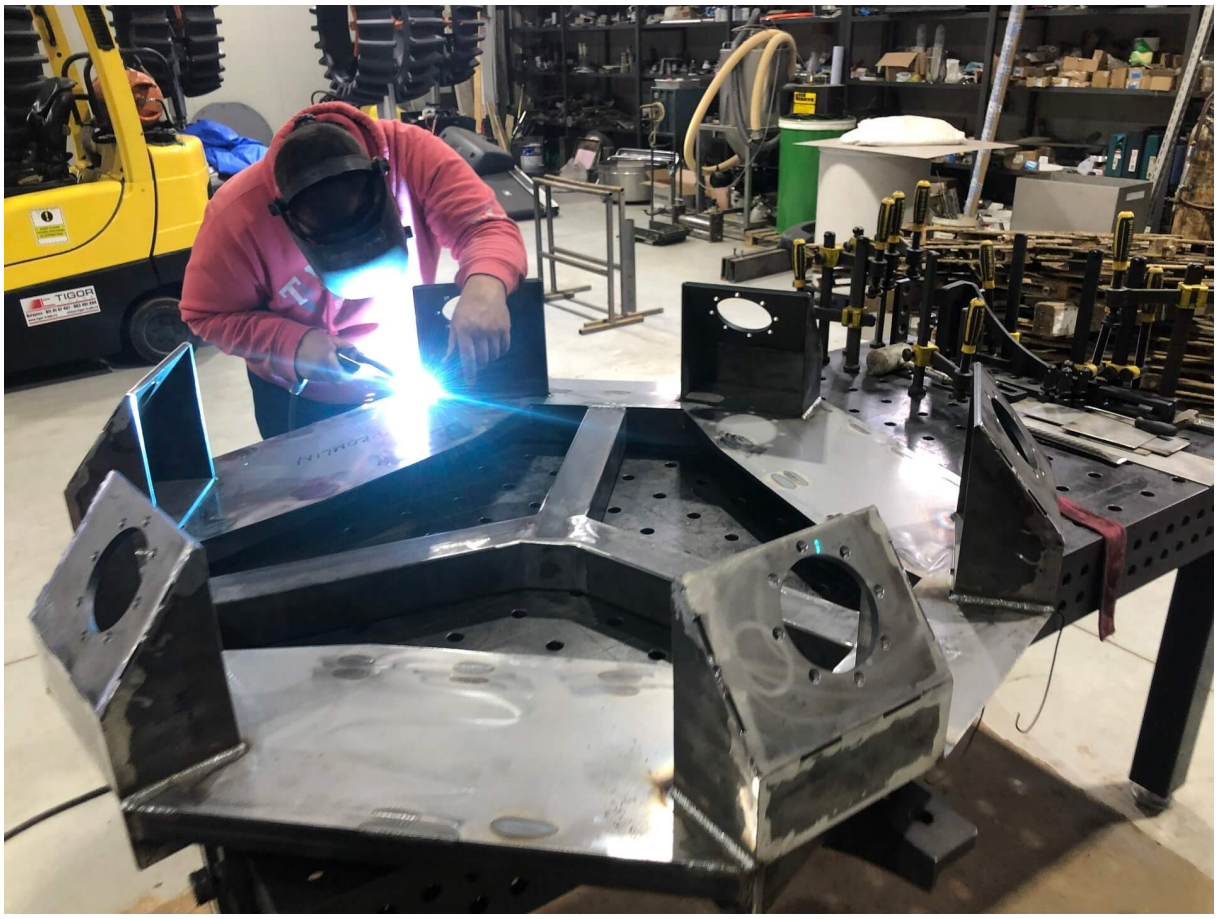
Slika 6.56 - Zavareni nosači aktuatora (levo) i obrađeni zavareni nosač aktuatora (desno)

Za zavarivanje se koriste visoko precizni zavarivački stolovi, mnoge stege i pomoćni alati za pozicioniranje kako bi se osigurala preciznost elemenata nakon zavarivanja. I platforma i baza se osim elemenata dobijenih laserom izrađuju i od različitih profila koji se prvo seku na određene dužine. Na slici 6.57 su prikazane pripremljene cevi i njihovo zavarivanje sa laserski isečenim elementom za formiranje platforme mehanizma.



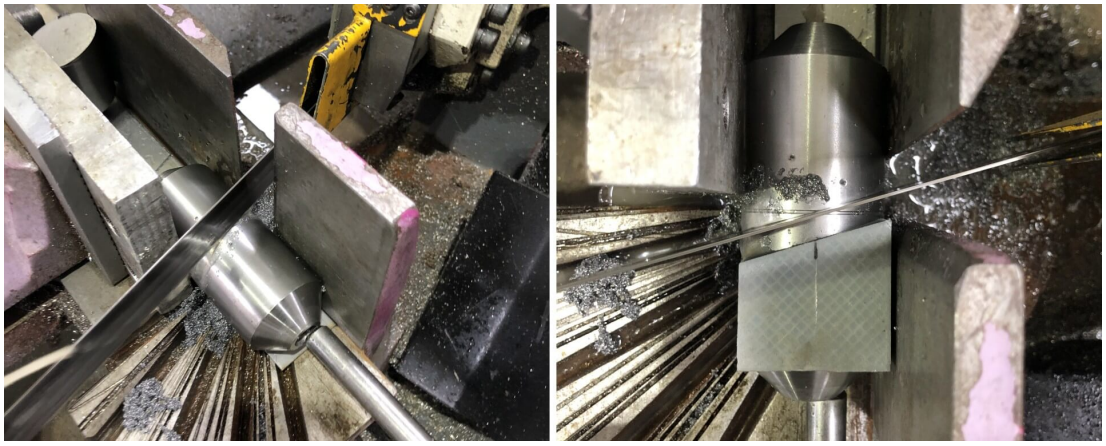
Slika 6.57 - Pripremljeni elementi od cevi (levo) i zavarivanje elemenata platforme mehanizma (desno)

Na slici 6.58 je prikazano zavarivanje sklopova nosača aktuatora i ostalih elemenata baze od kojih su nekih dobijeni rezanjem na laseru a drugi rezanjem cevi.



Slika 6.58 - Zavarivanje elemenata baze

Spajanje gornjih poluga sa ostatkom mehanizma, preko sfernih zglobova, se vrši sa M12 vijcima. Elementi koji obezbeđuju odgovarajuću orijentaciju ove veze se dobijaju rezanjem iz šipki. Na slikama 6.59 i 6.60 je prikazano rezanje elemenata za vezu gornjih i donjih sfernih zglobova uz pomoć 3D štampanih alata.

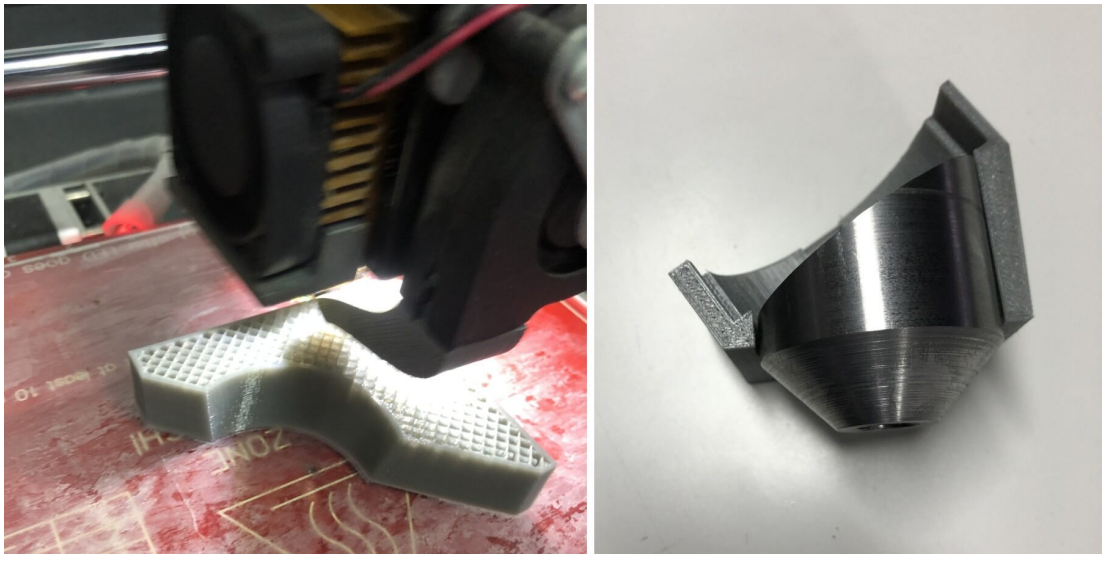


Slika 6.59 - Rezanje elementa za vezu gornjeg sfernog zgloba



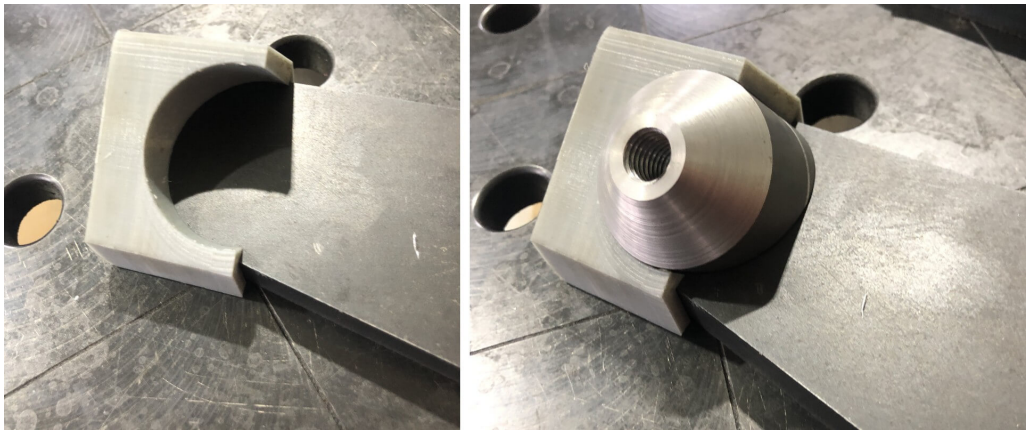
Slika 6.60 - Rezanje elementa za vezu donjeg sfernog zgloba

Alati za pozicioniranje prilikom zavarivanja prethodno prikazanih elemenata se dobijaju 3D štampanjem od plastike (i pored visoke radne temperature) kao što je prikazano na slici 6.61.



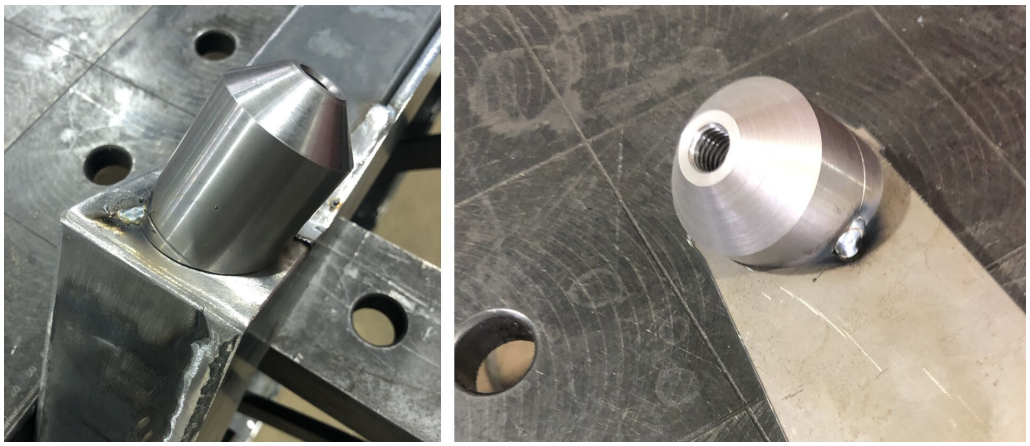
Slika 6.61 - 3D štampanje alata za pozicioniranje elemenata

Na slici 6.62 je prikazano kako se ovako dobijeni alati koriste za pozicioniranje elemenata na donjoj poluzi.



Slika 6.62 - Upotreba 3D štampanog alata za pozicioniranje elemenata na donjoj poluzi

Kada su svi elementi odgovarajuće pozicionirani, omogućeno je njihovo zavarivanje za platformu i za donje poluge, kao što je prikazano na slici 6.63.



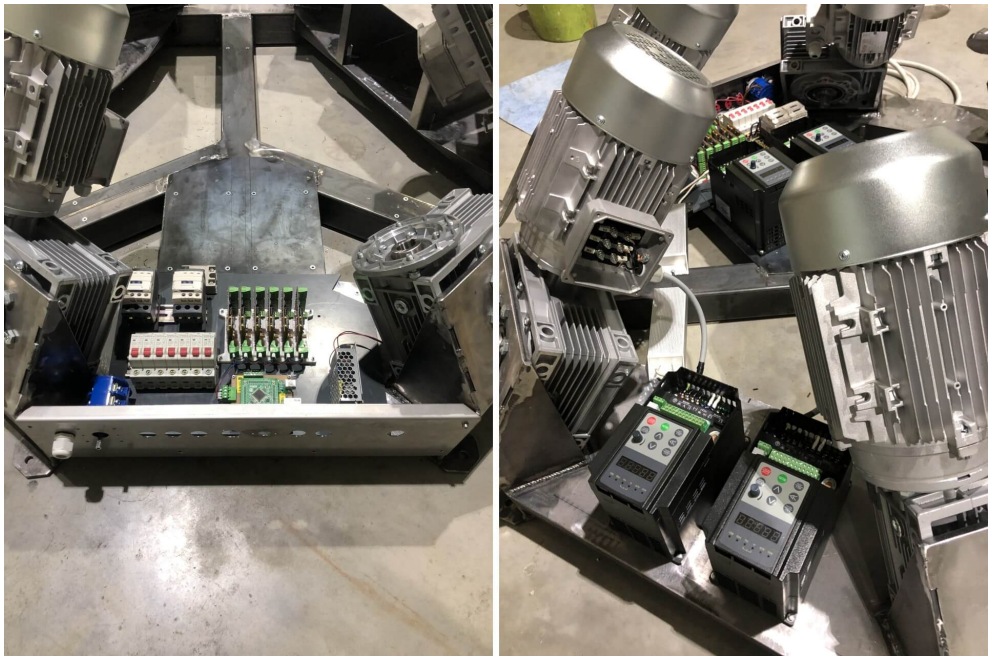
Slika 6.63 - Zavarivanje elemenata za vezu gornjih i donjih sfernih zglobova

Zavarivanje vratila za donju polugu je prikazano na slici 6.64 levo, dok su desno prikazani gotovi elementi gornjih poluga koji su svi obrađeni na strugu i neki zavareni.



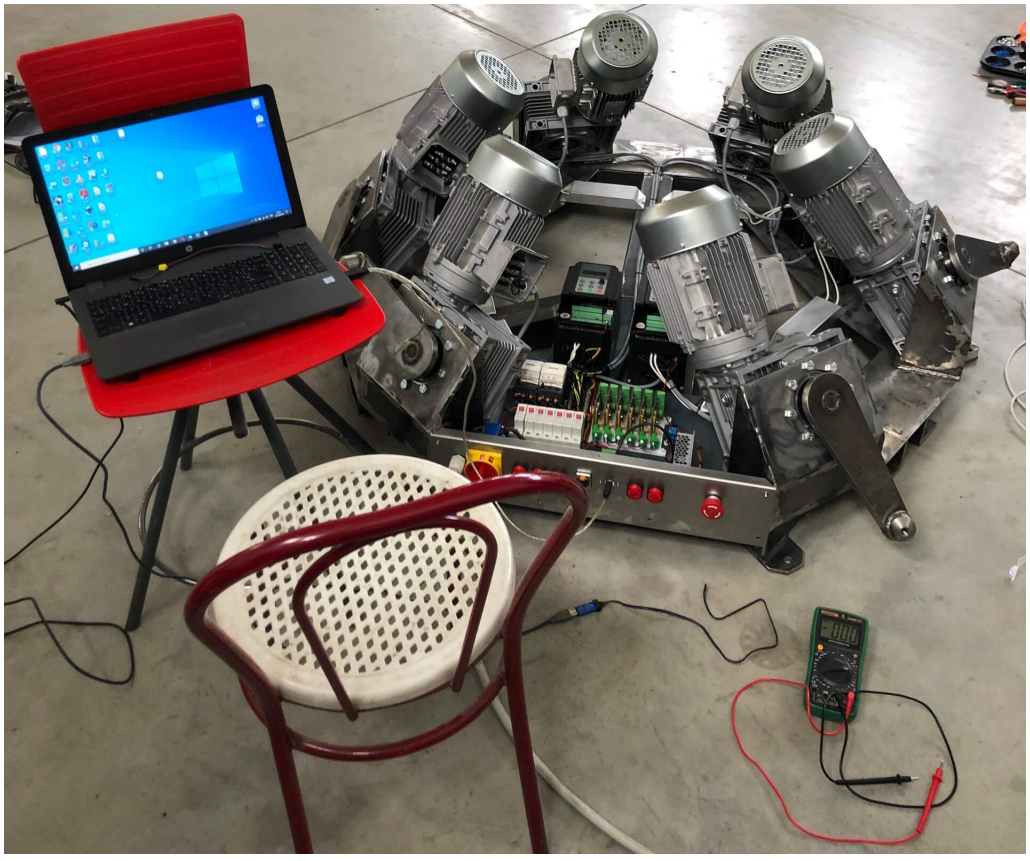
Slika 6.64 - Zavarivanje vratila i donje poluge (levo) i elementi gornjih poluga (desno)

Nakon što su izrađeni svi elementi konstrukcije, prešlo se na sklapanje i zatim ožičavanje električnih komponenti. Na slici 6.65 su prikazani delovi nulte verzije električnog sistema prototipa.



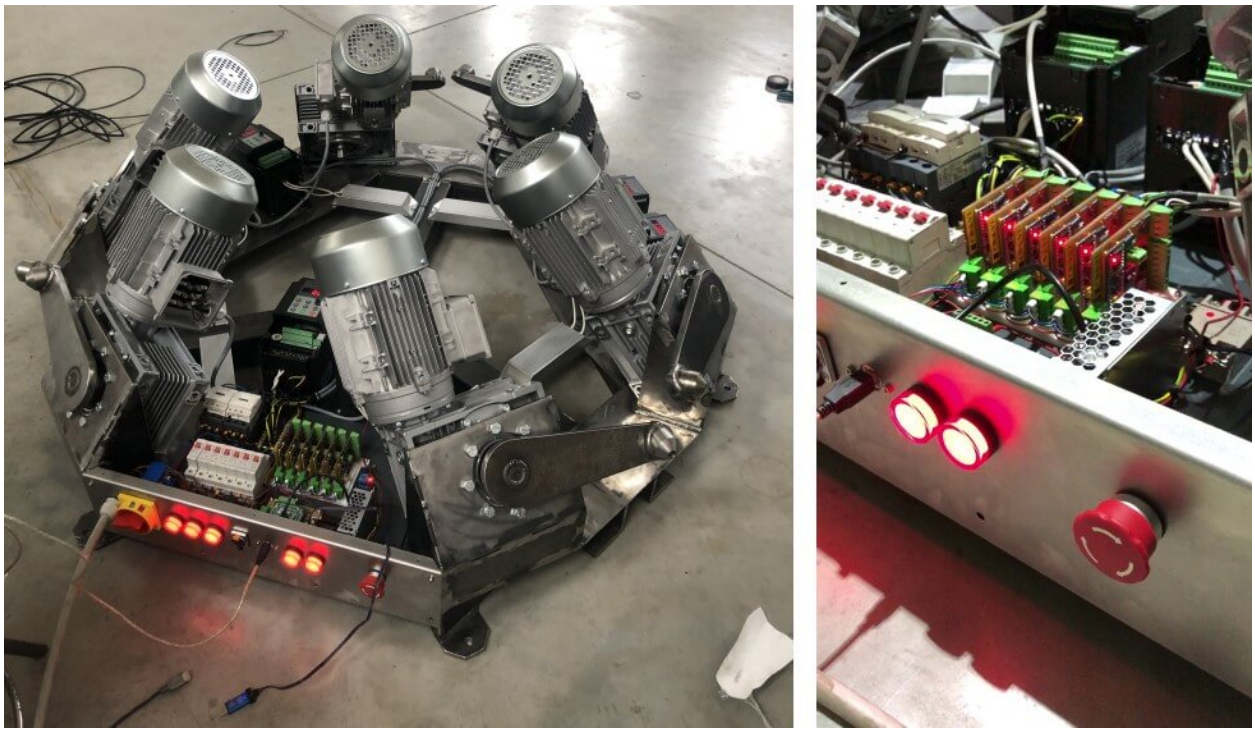
Slika 6.65 - Prvo ožičavanje električnog sistema

Na sve mikrokontrolere električnog sistema su zatim umemorisani njima odgovarajući programi (proces prikazan na slici 6.66) i frekventni regulatori su podešeni odgovarajućim vrednostima za parametre.



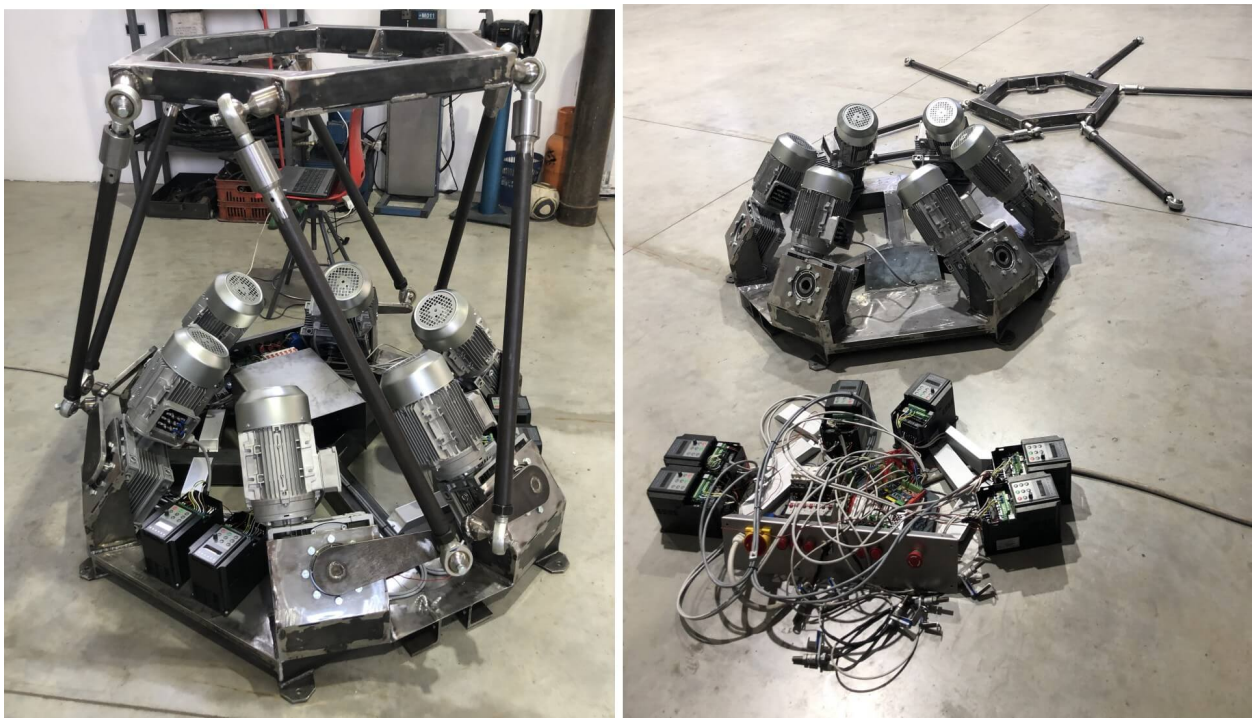
Slika 6.66 - Prvo programiranje mikrokontrolera

Na slici 6.67 je prikazano prvo uspešno uključivanje nulte verzije električnog sistema sa svim frekventnim regulatorima, `Control Board` i `Servo Board` elektronikama.



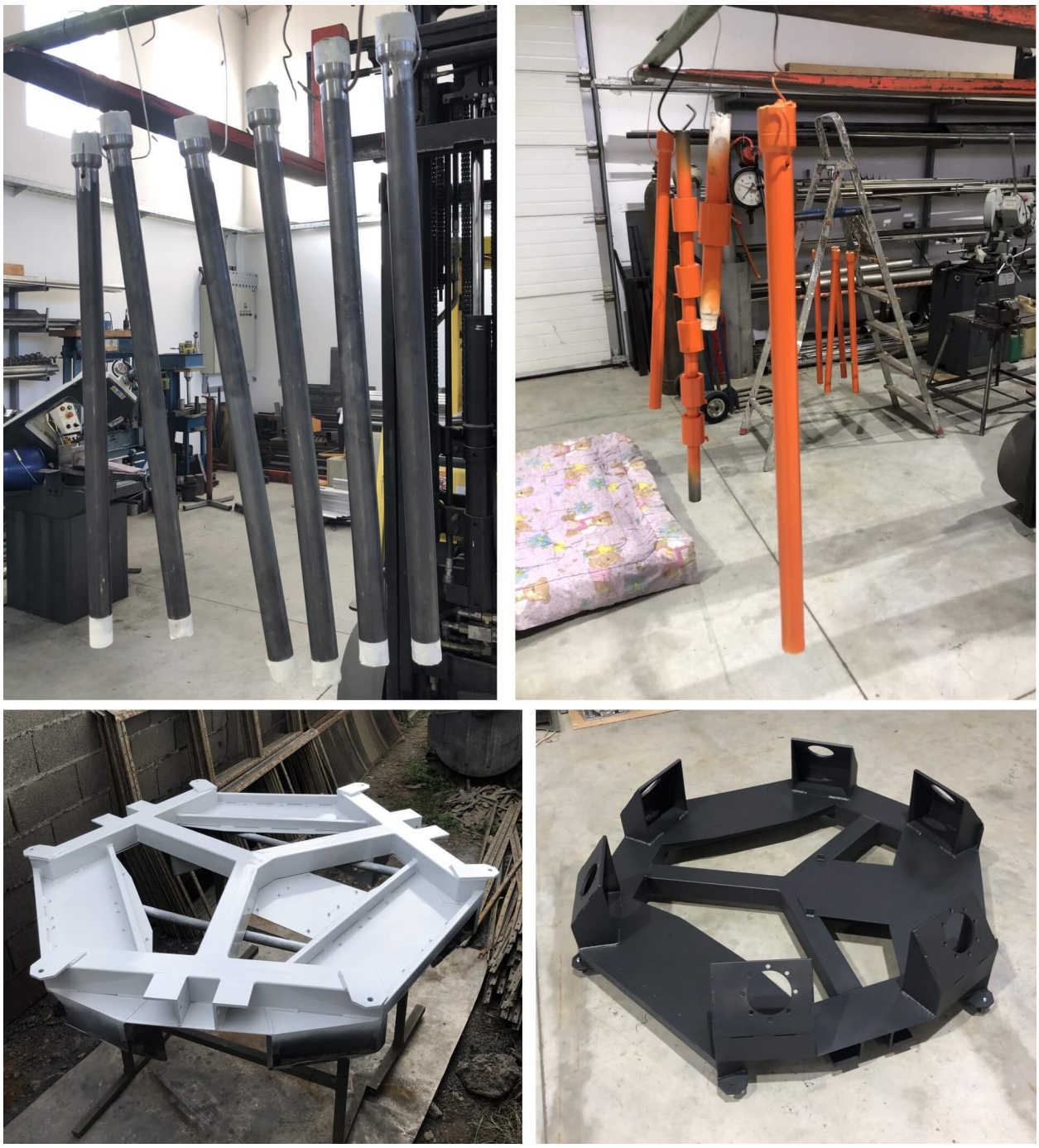
Slika 6.67 - Prvo uključivanje nulte verzije električnog sistema

Prvo su testirana pojedinačna kretanja aktuatora, zatim istovremeno kretanje aktuatora i tek onda su montirane gornje poluge i platforma, kao što je prikazano na slici 6.68. Kada je uspešno testiran potpuno sklopljen mehanizam prototipa u stvarnoj veličini, u ovoj fazi čistog metala, prelazi se na rasklapanje zbog farbanja.



Slika 6.68 - Potpuno sklopljen i zatim rasklopljen mehanizam prototipa u stvarnoj veličini

Zavareni spojevi i oštre ivice se obrađuju brušenjem i zatim se svi elementi čiste kako bi se prešlo na farbanje u cilju površinske zaštite metala. Prvo se nanosi osnovna farba (bele boje kao što je prikazano na slici 6.69) a tek onda završna farba u odabranoj boji. Za poluge je izabrana narandžasta boja koja je jako upadljiva dok je za bazu i platformu izabrana tamno siva boja. Poluge zbog bezbednosti i treba da budu jako uočljive jer se mogu brzo kretati i postoji više mesta na kojima može doći do povreda.



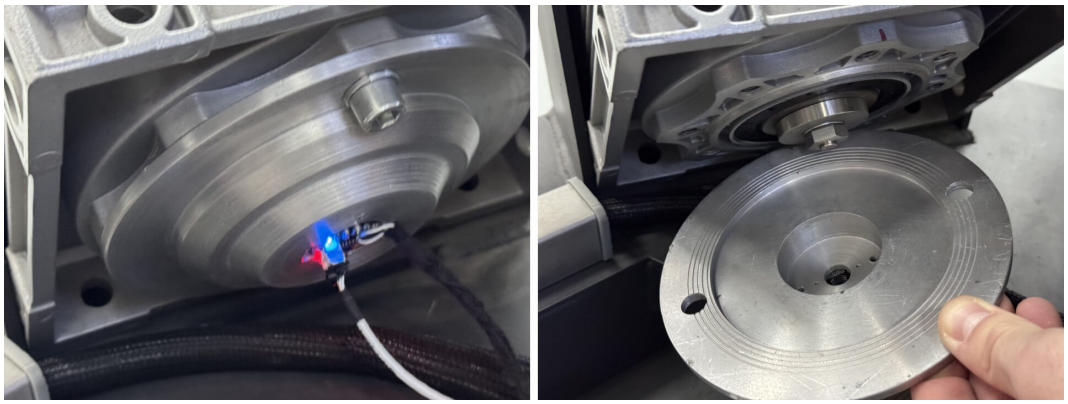
Slika 6.69 - Farbanje delova mehanizma

Na slici 6.70 je prikazan po prvi put potpuno sklopljen prototip simulatora leta, decembra 2019. godine. Na platformi se nalazilo sedište, konstrukcija na koju se oslanjaju stopala i monitor za prikaz podataka.



Slika 6.70 - Prvo potpuno sklapanje prototipa simulatora leta

Nakon prikazanog postupka proizvodnje konstrukcije ona više nije menjana, dok se električni sistem unapređivao postepeno. Najveće unapređenje predstavljao je prelazak na samostalno projektovani i proizveden senzor ugla `Angle Sensor` koji je prikazan na slici 6.71.



Slika 6.71 - Novi senzor ugla

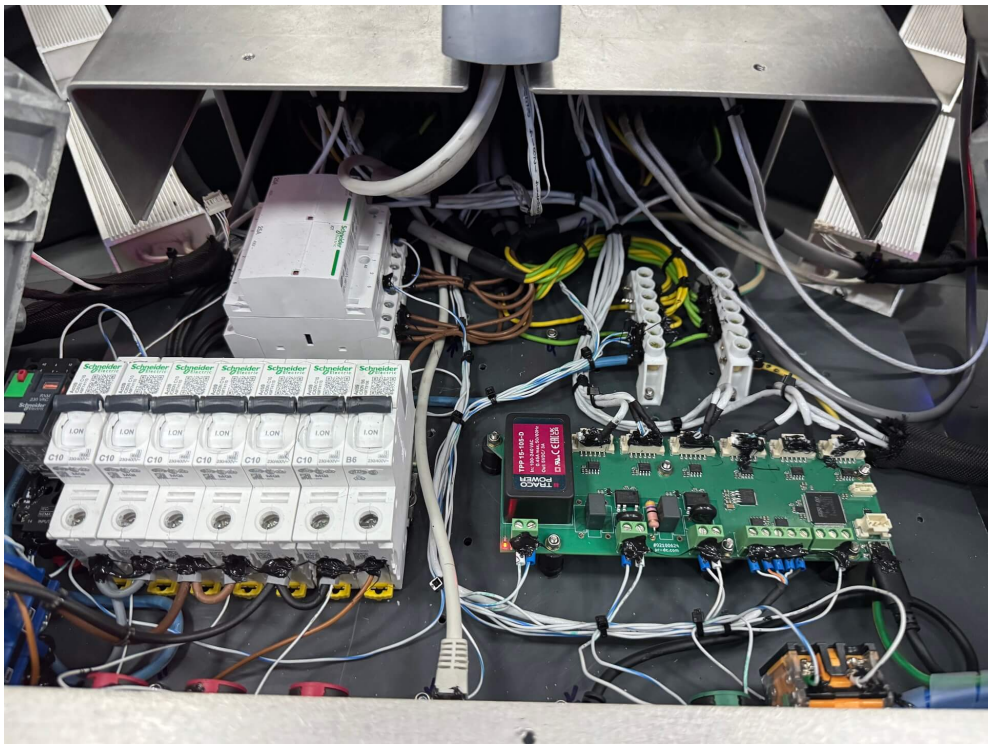
Na prethodnoj slici se vidi štampana ploča elektronike sa obe strane. Sa donje strane se nalazi senzor koji beskontaktno određuje ugao vratila na kome se nalazi stalni (neodijumski) magnet.

Za ožičenje su korišćene žice koje su u skladu sa vazduhoplovnim i vojnim standardima jer one obezbeđuju potrebne karakteristike u pogledu elektromagnetne imunosti i kompatibilnosti. Osim toga, svuda gde je to bilo moguće su korišćeni konektori otporni na vibracije koje se mogu javiti kao posledica rada sistema.

Na slikama 6.72 i 6.73 su prikazane ugrađene i ožičene poslednje verzije elektronika `Sensor Board` i `Control Board`.

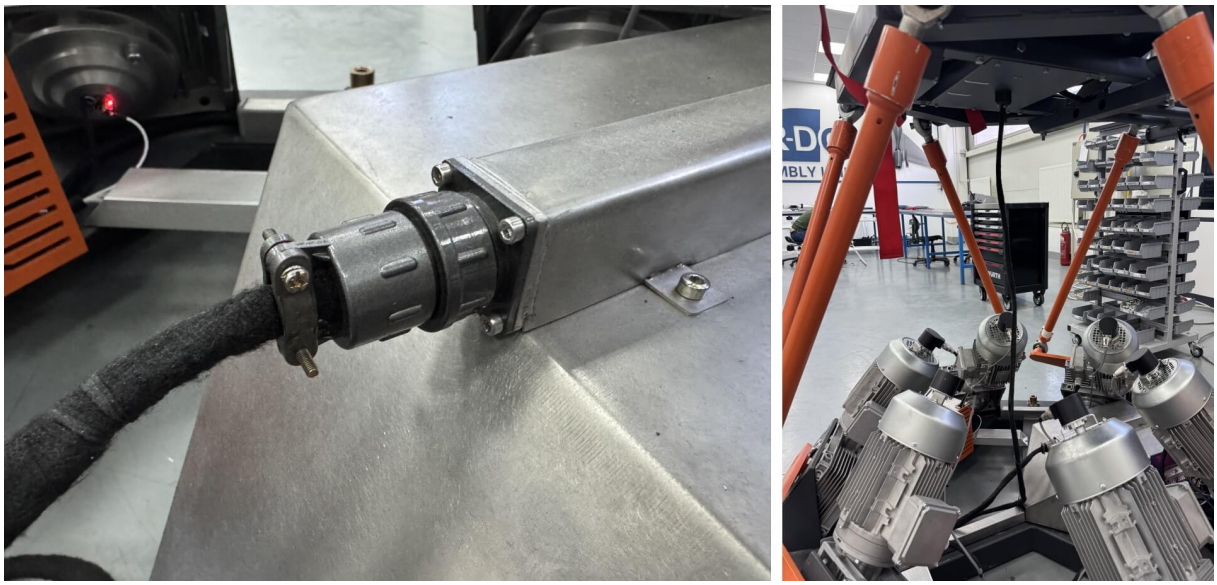


Slika 6.72 - Ugrađene poslednje verzije elektronike `Servo Board`



Slika 6.73 - Ugrađena poslednja verzija elektronike `Control Board`

Na slici 6.74 je prikazan konektor koji električno povezuje platformu i bazu simulatora leta. Ovaj konektor omogućava potpuno električno odvajanje platforme od baze što uz mogućnost mehaničkog odvajanja omogućava fleksibilniji transport i pakovanje.



Slika 6.74 - Konektor za električnu vezu platforme i baze

Enkoder za merenje ugaone brzine elektromotora je montiran na poklopac ventilatora i vezan je sa vratilom elektromotora preko elastične spojnice, što se vidi i na slici 6.75.



Slika 6.75 - Veza enkodera i elektromotora

Na slici 6.76 je prikazan električni panel koji se nalazi sa prednje strane baze. Na električnom panelu se nalaze (sa leve na desnu stranu) sijalica za indikaciju stanja, glavni prekidač za napajanje, tri sijalice za indikaciju sve tri ulazne faze, `RJ45` konektor, `USB C` konektor, sijalica za indikaciju mikrokontrolera, taster za hitno zaustavljanje i standardna `16A` industrijska utičnica za trofazno napajanje sa uzemljenjem.



Slika 6.76 - Električni panel prototipa

Za penjanje na platformu simulatora leta se koriste stepenice, kao što se vidi na slici 6.77, koje se oslanjaju na deo konstrukcije platforme. Penjanje je omogućeno sa obe strane sedišta.



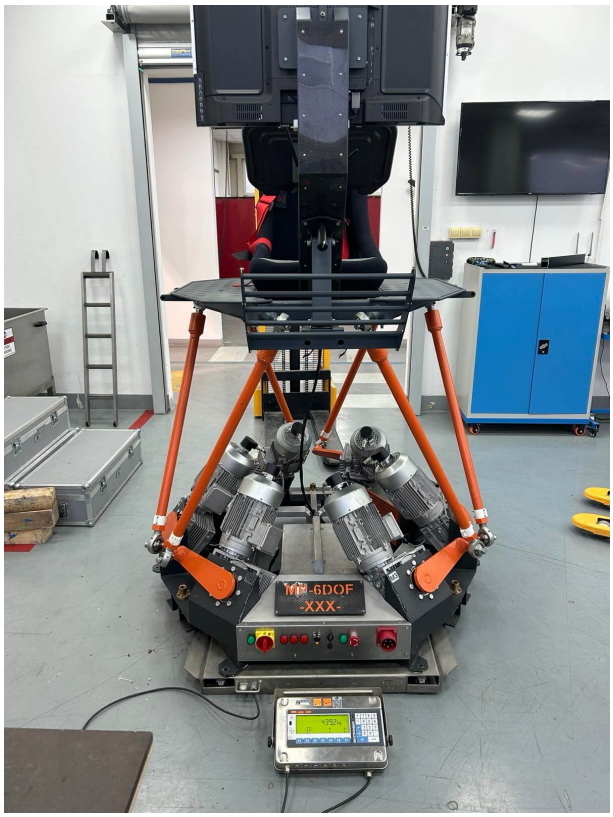
Slika 6.77 - Stepenice za penjanje na platformu prototipa simulatora leta

Finalna pilotska kabina, prikazana na slici 6.78, ima sedište sa pojasom za pilota, gazište sa obe strane sedišta (kako bi se olakšalo sedenje), daljinski upravljač sa podešavajućim uglom i rastojanjem od pilota, monitor za prikaz podataka i podesivu konstrukciju za oslanjanje stopala.



Slika 6.78 - Finalni izgled pilotske kabine prototipa simulatora leta

Na kraju, ukupna masa prototipa simulatora leta je oko 440 kilograma. Merenje mase je prikazano na slici 6.79.



Slika 6.79 - Merenje mase prototipa simulatora leta

Prvo javno predstavljanje prototipa simulatora leta je bilo na međunarodnom sajmu naoružanja i vojne opreme u Beogradu, *Partner 2023*, održanom krajem septembra 2023. godine (prikazano na slici 6.80).



Slika 6.80 - Prvo javno predstavljanje prototipa simulatora leta

6.1.2.5 Program za upravljanje

Sve elektronike koje se koriste u okviru električnog sistema prototipa u stvarnoj veličini su bazirane na nekoj od serija STM32 mikrokontrolera, koji su bazirani na ARM arhitekturi.

I pored toga što se projekti za ove mikrokontrolere najčešće rade u razvojnim okruženjima kao što su STM32CubeIDE (zvanično razvojno okruženje za STM32 od *STMicroelectronics*-a) ili Keil MDK (kompanije *Arm*) odlučeno je da se nastavi sa korišćenjem Arduino razvojnog okruženja zajedno sa Arduino_Core_STM32 softverskom podrškom za STM32 mikrokontrolere koju obezbeđuje kompanija *STMicroelectronics*. Na taj način je zadržano korišćenje svih prednosti Arduino zajednice i ostvaren je kontinuitet u razvoju ovog konkretnog projekta. Za upisivanje programa u memoriju mikrokontrolera se koristi JSLAB funkcija uploadArduino().

Svi programi koji se koriste za upravljanje prototipa MP-6DOF-XXX su dati u prilogu X. U odnosu na program za umanjeni prototip, ovo je značajno kompleksniji projekat. Ukupno se programira **13 mikrokontrolera sa tri različita programa**. Zbog potrebe da se osigura pouzdana, brza i efikasna komunikacija između svih mikrokontrolera i drugih računara, za sve komunikacije se koristi binarni komunikacioni protokol PRDC_SCBP koji je detaljno opisan u poglavlju 6.1.1.5 i čija je implementacija data u prilogu T. U sekciji 6.1.7 ove disertacije su data značenja kodova statusa, a u sekciji 6.1.8 definicije paketa i poruka koje se koriste za komunikaciju.

Za razliku od umanjenog prototipa, na glavnom mikrokontroleru (elektronike Control Board) prototipa u stvarnoj veličini se izvršava čitav proračun inverzne i direktne kinematike koji je implementiran korišćenjem specifičnih instrukcija mikrokontrolera. Biblioteka za proračunavanje CMSIS-DSP iz CMSIS (Common Microcontroller Software Interface Standard) kao zajedničkog standarda za programiranje mikrokontrolera kompanije *Arm* se koristi za matrični proračun (najznačajnije u slučaju određivanja inverzne matrice). Ovu biblioteku Arduino_Core_STM32 obezbeđuje automatski.

Programi su, tamo gde je to bilo logično i korisno, podeljeni na module i iskorišćene su mogućnosti programskog jezika C++ za objektno-orijentisano programiranje, pa je tako na primer definisano šest objekata klase ServoBoard za komunikaciju sa šest aktuatora. Za Polu-dupleks RS485 komunikacije se koristi biblioteka PRDC_RS485HD_STM32[144].

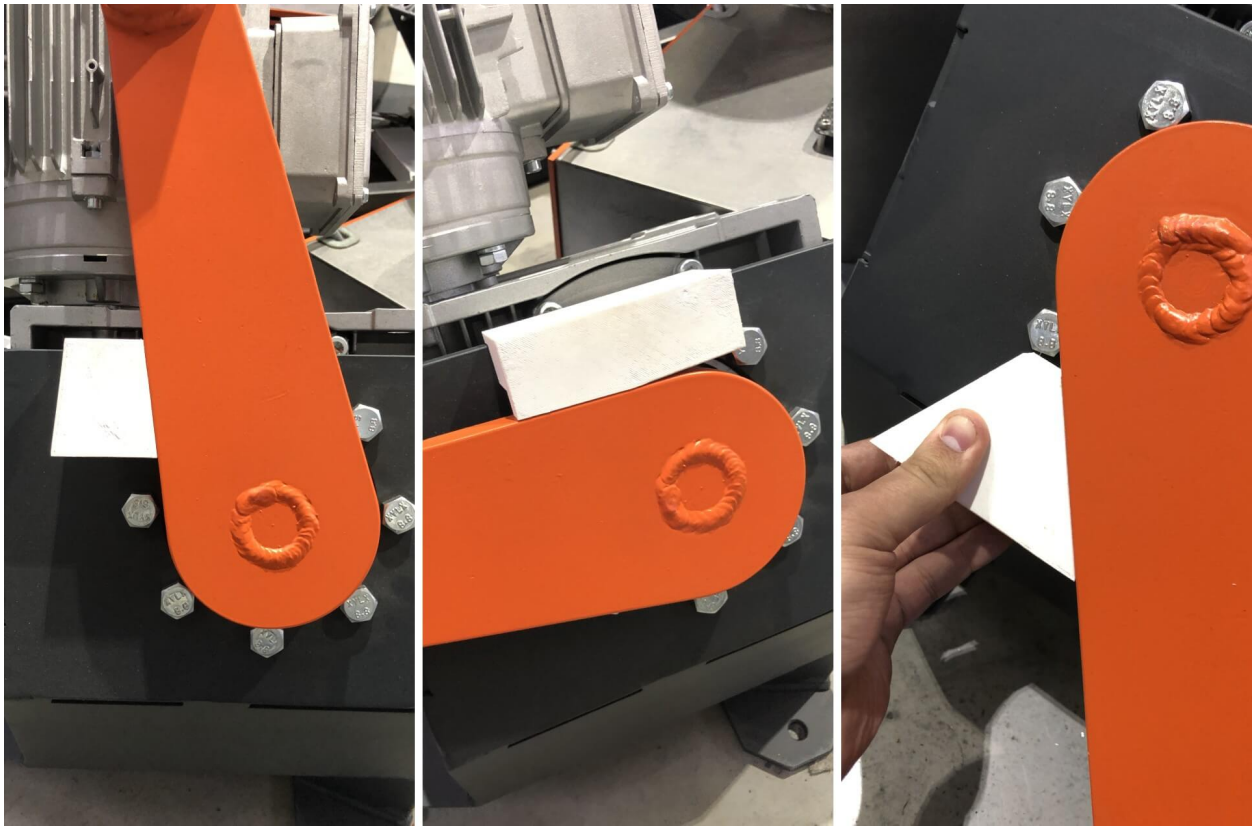
Program MP-6DOF-XXX_CB.ino za elektroniku Control Board se sastoji iz sledećih modula:

- config.h - za konfigurisanje rada samog programa, uključujući brzine komunikacija, unos vrednosti geometrijskih parametara mehanizma itd.
- BASIC_IO - za kontrolu osnovnih ulaza i izlaza kao što su lampe za indikaciju i kontrola kontaktora za napajanje elektromotora.
- MECHANISM_6RSS - za rešavanje inverznog i direktnog kinematičkog problema.
- MISSION_CONTROLLER - za dobijanje instrukcija za kretanje i realizaciju dobijene misije.
- MOTION_PLANNER - za planiranje kretanja mehanizma.
- SERVO_BOARD - za dvosmernu komunikaciju sa elektronikama Servo Board, šalju se trenutni željeni ugao donje poluge i koeficijenti za formiranje upravljanja, dobijaju se podaci o statusu aktuatora i to njegovo trenutno stanje, izmereni ugao, temperatura i trenutna frekvencija frekventnog regulatora. Ovaj modul vodi računa o gubitku komunikacije sa Servo Board elektronikom.
- math_helper - za matematičke operacije kao što je množenje matrica uz pomoć CMSIS-DSP biblioteke.

- `print_helper` - za slanje matrica, vektora i drugih podataka u tekstualnom obliku preko USB-a.
- `debug.h` - za kontinualni prikaz vrednosti seta promenljivih preko USB-a u cilju pronalaženje grešaka.
- `regression_test.h` - za pokretanje regresijskog testiranja i dobijanje rezultata koji se upoređuju sa drugim metodama za isti proračun (kao što je onaj u `JSLAB`-u).

Zadatak ovog programa je da na osnovu instrukcija, koje dobija preko neke od serijskih komunikacija, istovremeno upravlja svim servo motorima.

U okviru datoteke `config.h` se unosi i ugao koji odgovara nultom uglu otklona, do kojeg je moguće doći pomoću kalibracionih alata prikazanih na slici 6.81. 3D štampanjem su napravljeni alati za kalibraciju ugla otklona za uglove `0°`, `-70°` i `+70°`.



Slika 6.81 - Kalibracija ugla otklona aktuatora

Nominalni tok programa `MP-6DOF-XXX_CB.ino` je:

- Definisanje svih početnih promenljivih i objekata već definisanih klasa (modula).
- Uspostavljanje komunikacije sa glavnim računarom.
- Pokretanje testa za regresije, ukoliko je potrebno.
- Pokretanje svih modula.
- Omogućavanje napajanja frekventnih regulatora.
- Čekanje inicijalizacije svih `Servo Board` elektronika.

- Provera statusa sistema pre ulaska u glavnu petlju programa.
- U okviru glavne petlje programa se u svakom ciklusu izvršava sledeće:
  - Upravljanje digitalnim pinovima.
  - Čitanje podataka od glavnog računara.
  - Čitanje podataka od svih `Servo Board`-a.
  - Promena stanja na osnovu svih ulaza (state machine).
  - Slanje podataka za svaki Servo Board.
  - Slanje statusa glavnom računaru.

Program `MP-6DOF-XXX_SB.ino` za elektroniku `Servo Board` se sastoji iz sledećih modula:

- `config.h` - za konfigurisanje rada samog programa, uključujući brzine komunikacija, unos vrednosti za vremenska ograničenja itd.
- `BASIC_IO` - za kontrolu osnovnih ulaza i izlaza kao što su LED za indikaciju.
- `ANGLE_CONTROLLER` - za generisanje odgovarajućeg upravljanja na osnovu izmerenog i željenog ugla otklona poluga pomoću PID kontrolera.
- `ANGLE_SENSOR` - za komunikaciju sa senzorom ugla i za dobijanje izmerenih vrednosti ugla otklona.
- `CONTROL_BOARD` - za dvosmernu komunikaciju sa elektronikama `Control Board`, šalju se podaci o trenutnom stanju aktuatora a dobijaju se komande za kretanje aktuatora.
- `debug.h` - za kontinualni prikaz vrednosti seta promenljivih preko USB-a u cilju pronalaženje grešaka.

U okviru glavnog dela programa se koristi i biblioteka `PRDC_VFD_ENC_EN600` za komunikaciju sa frekventnim regulatorom, ova `Arduino` biblioteka je takođe data u prilogu X. Biblioteka `PRDC_VFD_ENC_EN600` koristi biblioteku `PRDC_FreePort` za protokol frekventnog regulatora[145].

Nominalni tok programa `MP-6DOF-XXX_SB.ino` je:

- Definisanje svih početnih promenljivih i objekata već definisanih klasa (modula).
- Pokretanje svih modula.
- Čekanje inicijalizacije `Control Board`-a.
- Čekanje inicijalizacije senzora ugla.
- Čekanje inicijalizacije frekventnog regulatora.
- Provera statusa sistema pre ulaska u glavnu petlju programa.
- U okviru glavne petlje programa se u svakom ciklusu izvršava sledeće:
  - Upravljanje digitalnim pinovima.
  - Čitanje podataka od `Control Board`-a.

- Čitanje podataka od senzora ugla.
- Čitanje podataka od frekventnog regulatora.
- Promena stanja na osnovu svih ulaza (state machine).
- Slanje statusa `Control Board`-u.

Za elektroniku `Angle Sensor` se koristi program `PRDC_AngleSensor.ino`. Program je baziran na `Arduino` biblioteci `PRDC_TLE5012B` za komunikaciju sa senzorom ugla, koja je takođe data u prilogu X. Kada se pročita ugao on se odmah šalje preko `Polu-dupleks RS485` sa `PRDC_SCBP` komunikacionim protokolom.

6.1.2.5.1 Programiranje frekventnog regulatora

Frekventni regulator proizvođača `ENC` i model `EN600` se programira unosom parametara direktno preko panela koji se nalazi na gornjoj strani frekventnog regulatora.

Parametri koji treba da se podese sa odgovarajućim vrednostima za frekventni regulator su:

- `F00.14` - 010 (reset svih parametara)
- `F00.15` - 0000 (REV na [REV/JOG] - za testiranje)
- `F00.19` - 3 (inkrementalni enkoder)
- `F00.24` - 2 (kontrola sa senzorom brzine)
- `F15.01` - 1.5 (snaga)
- `F15.02` - 220 (napon)
- `F15.03` - 6.0 (jačina struje)
- `F15.04` - 50 (učestanost)
- `F15.05` - 1410 (rpm)
- `F15.06` - 2 (broj pari polova)
- `F15.19` - 2 (autotune - potrebno je ići na [RUN] posle ovoga)
- `F01.17` - 50 (vreme ubrzavanja 0.5 sekundi)
- `F01.18` - 50 (vreme usporavanja 0.5 sekundi)
- `F01.19` - 0 (jedinica vremena 0.01s)
- `F02.11` - 2 (usporavanje sa kočenjem jednosmernom strujom)
- `F05.00` - 5 (serijska komunikacija, Free protocol 1)
- `F05.01` - 008 (brzina komunikacije 57600 bps)
- `F05.02` - 0004 (1-7-1 format, even parity, ASCII, sa odgovorom)
- `F05.03` - 1 (lokalna adresa)

- F07.00 - 0.000 (filter analognog signala 0s)
- F07.01 - 2.000 (pojačanje analognog signala)
- F07.02 - 0.000 (ofset signala)
- F08.18 - 2 (REV terminal)
- F08.19 - 1 (FWD terminal)
- F08.26 - 1 (kontrola FWD/REV sa dva prekidača)
- F09.04 - 22 (relejni izlaz za signal greške)
- F01.00 - 1 (ulazni naponski analogni signal za kontrolu frekvencije AI1) ili 4 (serijska komunikacija)
- F01.15 - 1 (terminali za komandu) ili 2 (serijska komunikacija)

Auto tuning za parametere F15.07 - F15.11 se vrši tako što se prvo parametar F15.19 postavi na 2 zatim se pritisne [ENTER] pa [RUN]. Nakon toga se F15.19 postavi na 1 i zatim se pritisne [ENTER] pa [RUN].

Tek nakon što je ovako isprogramiran frekventni regulator on može da se koristi zajedno sa elektronikom Servo Board za kontrolu ugla otklona donjih poluga.

6.1.2.6 Transportovanje simulatora

Za potrebe omogućavanja transportovanja simulatora je pored mogućnosti potpunog električnog i mehaničkog razdvanja platforme i baze napravljen i metod za jednostavno podizanje baze. Uz simulator se isporučuju tri navojna vretena, ključ i specijalne stope kojima se vrši podizanje baze. Na bazi su već ugrađene navrtke kroz koje se navojnim vretenima vrši podizanje. Nakon što se simulator podigne dovoljno na sve tri tačke, moguće je podvući paletar, paletarom dodatno podići simulator i zatim ga dalje transportovati. Na slici 6.82 se vidi navojno vreteno sa stopom za podizanje kao i ceo simulator podignut na paletar.



Slika 6.82 - Podizanje simulatora leta za transport paletarom

6.1.2.7 Analiza pristupačnosti tehničkog rešenja simulatora leta

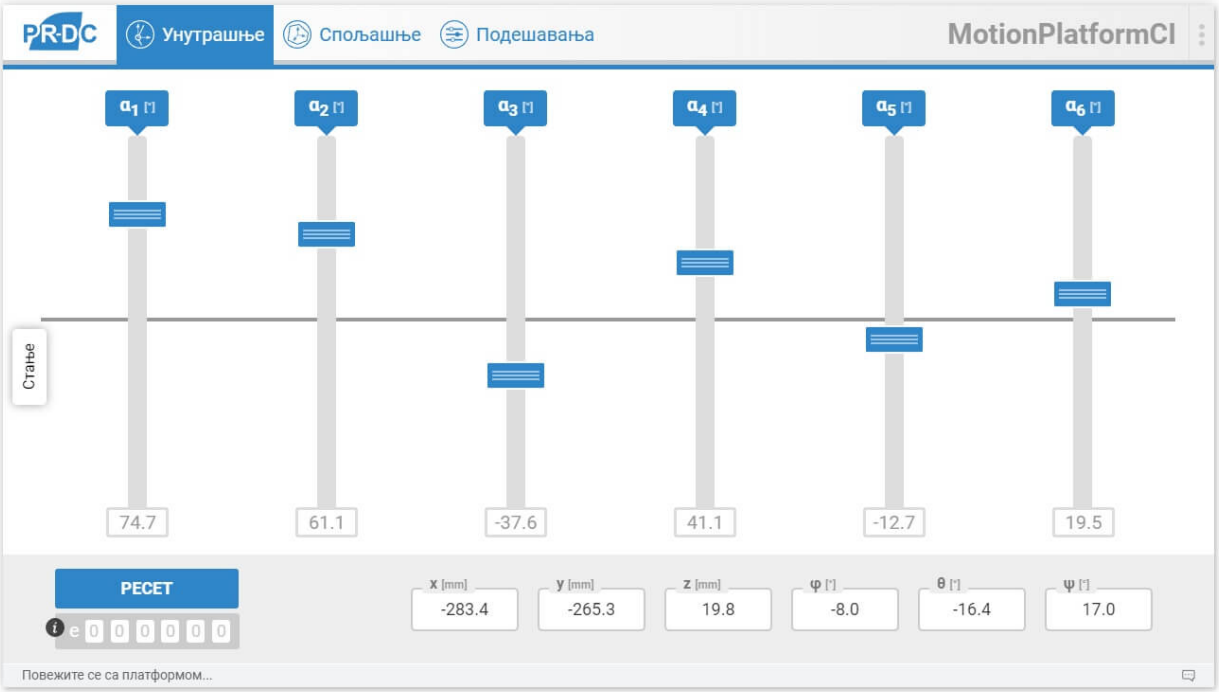
Do sada je u ovom poglavlju detaljno opisano tehničko rešenje simulatora leta. Prethodno je zaključeno da simulator leta za potrebe sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi mora biti ekonomski pristupačaniji od trenutno postojećih kako bi na kraju i bio upotrebljiv. Osim toga, za razliku od postojećih veoma kompleksnih i velikih sistema, simulator leta za ovakav sistem treba da bude relativno malih dimenzija i mase kako bi se omogućio njegov jednostavan transport i ugradnja. Za projektovani simulator leta se mogu konstatovati sledeće činjenice:

- Za kretanje mehanizma se koristi šest obrtnih električnih aktuatora i to standardnih asinhronih elektromotora sa pužnim reduktorima. Ovi elektromotori i reduktori se koriste za najrazličitije namene u industriji.
- Za upravljanje elektromotorom se koriste standardni industrijski frekventni regulatori i inkrementalni enkoderi.
- Mehatronički sistem simulatora je jednostavan jer pored zaštite na ulazu ima samo upravljačku elektroniku i šest zasebnih aktuatora.
- Elektronike *Servo Board*, *Angle Sensor* i *Control Board* se sastoje iz široko dostupnih električnih komponenata (kao što su mikrokontroleri, komunikacioni transiveri, otpornici, kondenzatori i slično). Ne koriste se komponente koje nemaju lako dostupnu alternativu ili su na listama za kontrolu trgovine.
- Za zglobove se koriste standardni zglobni ležaji koji se modifikuju.
- Većina elemenata strukture se izrađuje rezanjem na CNC laserima iz pločastih materijala. Ovi elementi se zatim zavaruju kako bi se dobila kompleksna prostorna struktura. Ovako napravljena tehnička dokumentacija omogućava veoma jednostavnu i brzu proizvodnju.
- Izabrani aktuatori, način snabdevanja energijom i podmazivanje zglobova omogućavaju tih rad.
- Mehanizam simulatora leta ima relativno male dimenzije i masu koje omogućavaju jednostavan transport kamionom bez značajnog rasklapanja.
- Omogućeno je jednostavno rastavljanje mehanizma na platformu, gornje poluge i bazu sa donjim polugama.
- Omogućeno je jednostavno podizanje mehanizma i transport paletarom.
- Simulator leta prima komande putem `USB`-a ili `RS485` protokola što omogućava veoma jednostavnu integraciju.

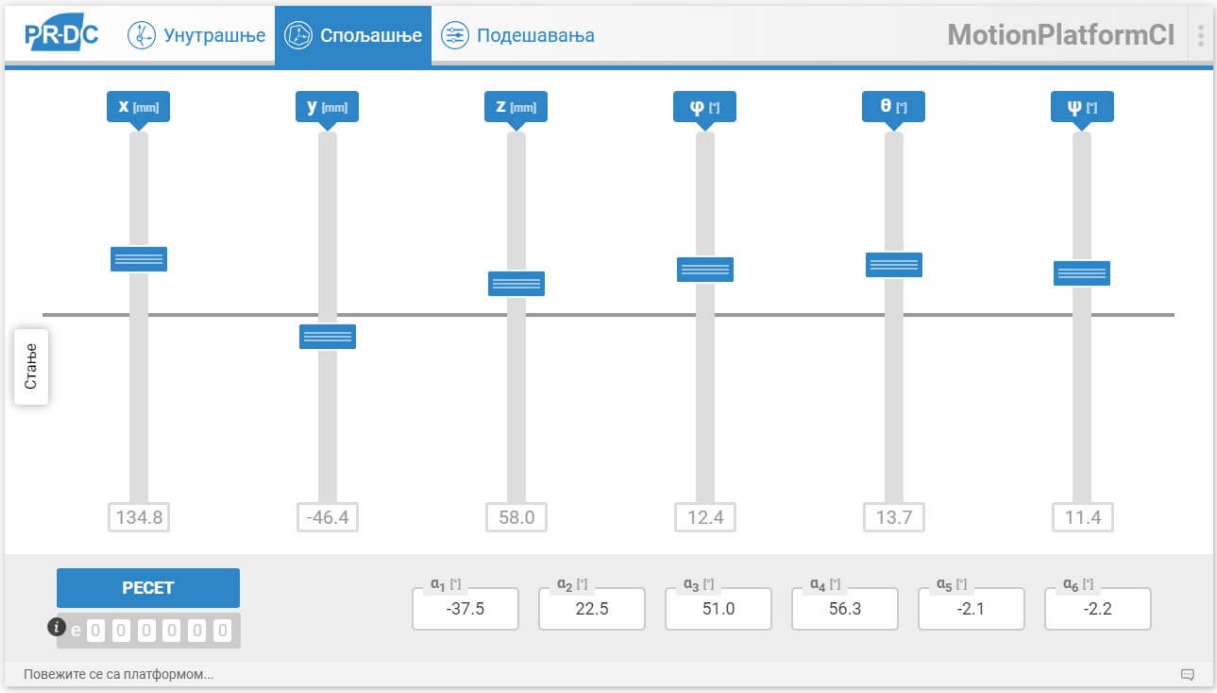
Na osnovu svega prethodnog se može zaključiti da je projektovan i proizveden pristupačan simulator leta, relativno male mase koji se jednostavno može pakovati i transportovati, dok je sa druge strane zadovoljen proračun kojim su dobijene optimalne performanse za simuliranje leta.

6.1.3 Program MotionPlatformCI

Za kontrolu mehanizma simulatora leta programirana je i posebna aplikacija pod nazivom `MotionPlatformCI`, koja ima napredan grafički korisnički interfejs za kontrolu kretanja mehanizma i praćenje njegovog stanja. Na slikama 6.83 i 6.84 su prikazane kartice za kontrolu mehanizma na osnovu unutrašnjih i spoljašnjih koordinata.



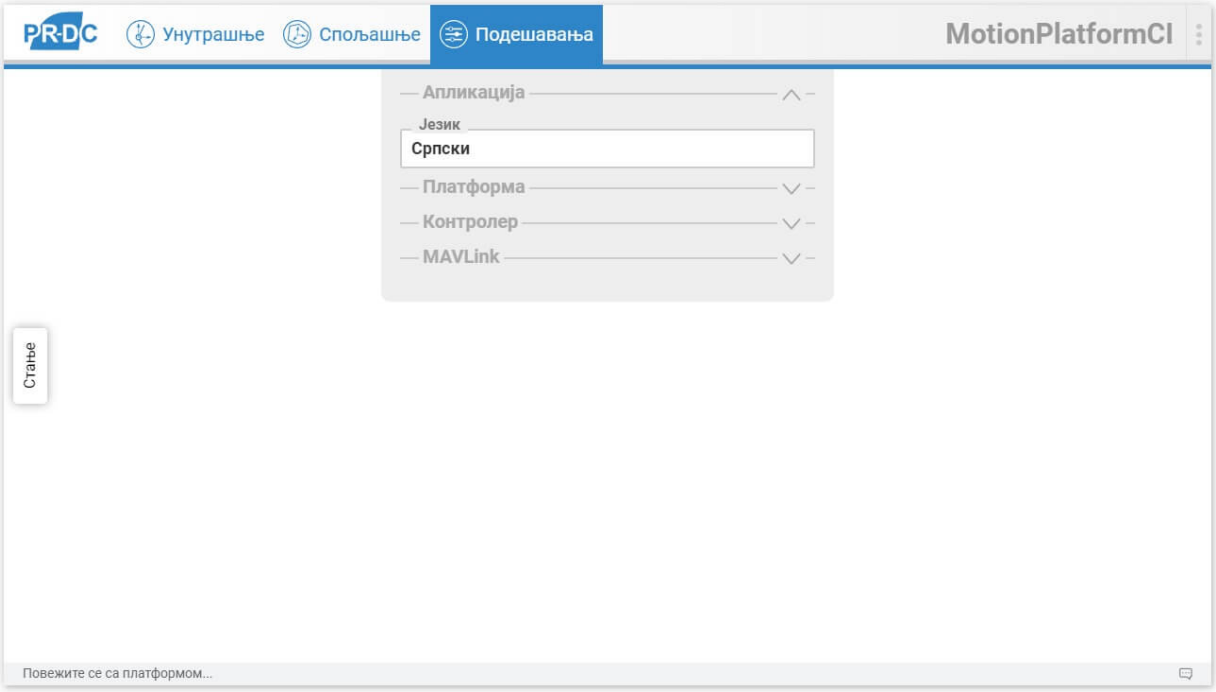
Slika 6.83 - Kartica aplikacije `MotionPlatformCI` za kontrolu mehanizma na osnovu unutrašnjih koordinata



Slika 6.84 - Kartica aplikacije `MotionPlatformCI` za kontrolu mehanizma na osnovu spoljašnjih koordinata

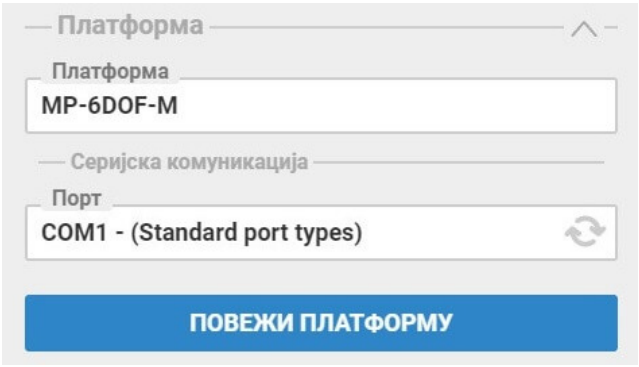
Aplikacija se bazira na programskom jeziku `JavaScript`, `Electron`-u i `Node.js` modulu pisanom u programskom jeziku `C++`. Zahvaljujući korišćenju `HTML`-a i `CSS`-a je jako jednostavno napraviti grafički korisnički interfejs.

Za kontrolu pojedinačnih unutrašnjih ili spoljašnjih koordinata koriste se klizači koji se mogu povlačiti ili polja za tekstualni unos vrednosti. Za resetovanje na nulu svih parametara se koristi dugme `[REST]` koje je dostupno u obe kartice. U kartici za kontrolu unutrašnjih koordinata se prikazuju rezultati za spoljašnje koordinate i obrnuto.



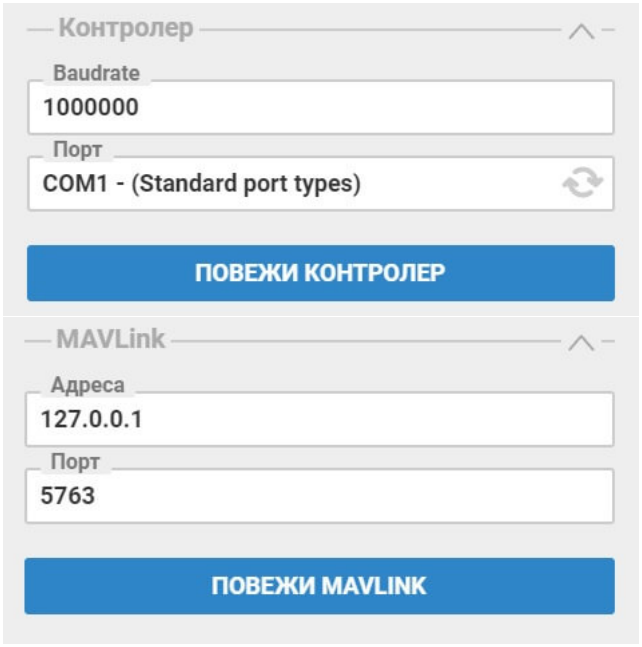
Slika 6.85 - Kartica za podešavanje aplikacije `MotionPlatformCI`

Treća i poslednja kartica ove aplikacije je prikazana na slici 6.85. U okviru ove kartice se vrše sva podešavanja same aplikacije, ali i pojedinačnih internih modula. Ovde je moguće izabrati jezik aplikacije. Implementirani su srpski jezik ćirilica, srpski jezik latinica i engleski jezik. Na slici 6.86 su prikazana podešavanja za platformu, te je tako moguće izabrati tip platforme (kao što je `MP-6DOF-XXX` ili `MP-6DOF-M`), port serijske komunikacije i pritiskom na dugme se povezati sa platformom.



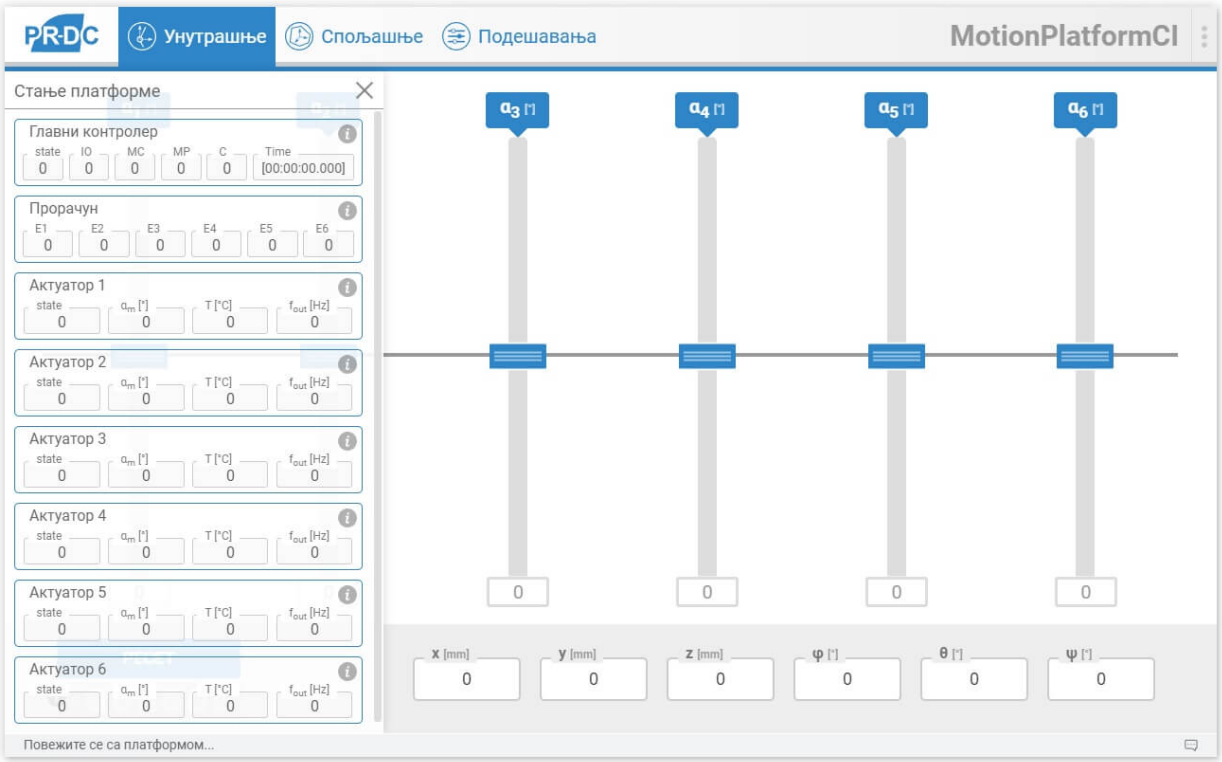
Slika 6.86 - Dostupna podešavanja za samu aplikaciju `MotionPlatformCI`

Na slici 6.87 su prikazana podešavanja za eksterne kontrolere i direktno povezivanje sa letelicom putem `MAVLink`-a.



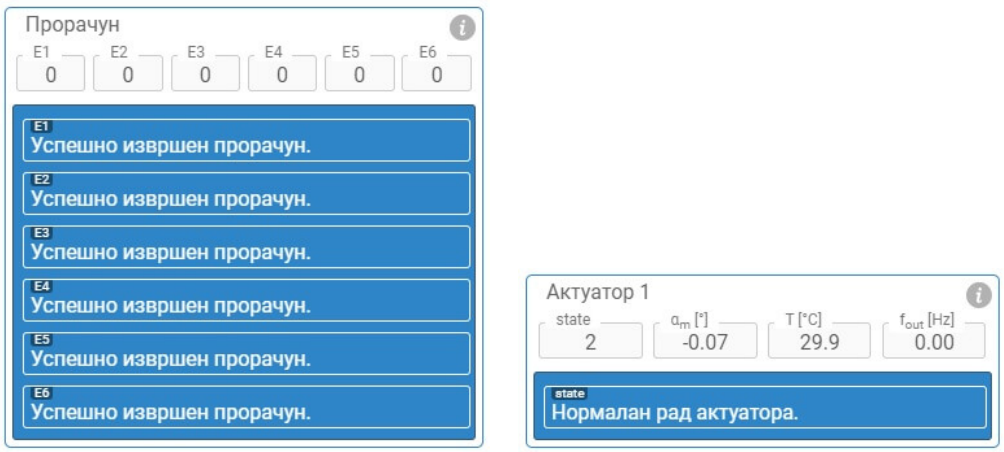
Slika 6.87 - Podešavanja u `MotionPlatformCI` za eksterne kontrolere i povezivanje sa letelicom putem `MAVLink`-a

Panel sa stanjem platforme je prikazan na slici 6.88.



Slika 6.88 - Prikaz panela sa stanjem platforme

Stanje platforme uključuje stanje glavnog kontrolera, stanje proračuna za svaku unutrašnju koordinatu i stanje za svaki aktuator. U sekciji 6.1.7 ove disertacije su data značenja kodova statusa, a u sekciji 6.1.8 definicije paketa i poruka koje se koriste za komunikaciju. U aplikaciji su dostupna i ova objašnjenja ali nakon otvaranja prozora za dodatne informacije, kao što se vidi na slici 6.89.

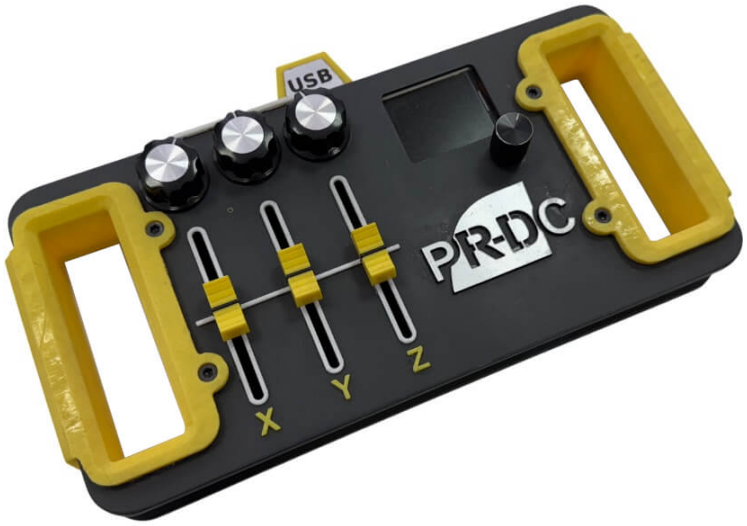


Slika 6.89 - Prikaz detaljnijih informacija za parametre stanja

Aplikacija je značajna jer pored implementiranih postupaka za rešavanje inverzne i direktne kinematike ima definisane sve potrebne pakete i poruke koje je potrebno poslati mehanizmu da bi se on odgovarajuće kretao.

6.1.4 Kontroler MP-6DOF-CTRLR

Kontroler pod nazivom `MP-6DOF-CTRLR`, koji je prikazan na slici 6.90, je kontroler inverzne kinematike i zapravo predstavlja uređaj koji na osnovu odgovarajućih pokreta korisnika preko `USB`-a, šalje komande za kontrolu mehanizma. Kontroler je baziran na mikrokontroleru serije `STM32 F103`.



Slika 6.90 - Kontroler inverzne kinematike MP-6DOF-CTRLR

Ovaj kontroler ima tri klizača za kontrolu translatornog kretanja platforme u odnosu na bazu i tri potenciometra za kontrolu orijentacije platforme u odnosu na bazu. Poslate vrednosti se korisniku prikazuju na integrisanom ekranu. Osim toga, kroz ekran je moguće vršiti određena podešavanja pomoću dugmeta za kontrolu. Ugradnjom IMU senzora i kompasa (kao u [146]) se omogućava i direktno upravljanje orijentacijom platforme na osnovu orijentacije kontrolera.

6.1.5 Kontroler MP-6DOF-M-CTRLR

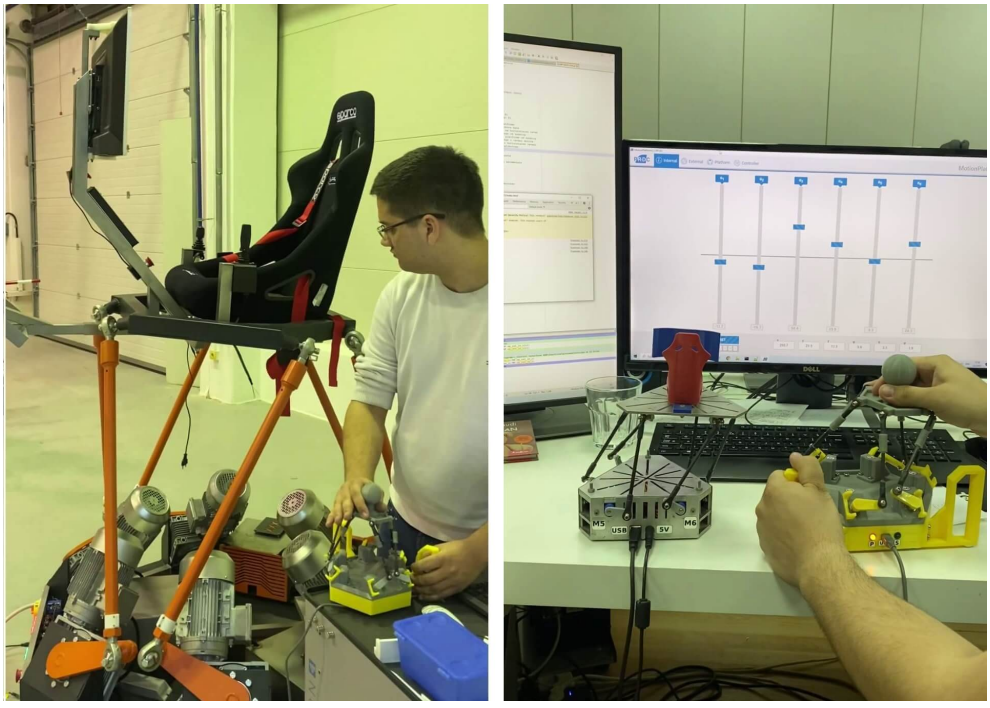
Kontroler pod nazivom `MP-6DOF-M-CTRLR`, koji je prikazan na slici 6.91, je kontroler direktne kinematike i zapravo predstavlja uređaj koji na osnovu odgovarajućih pokreta korisnika preko `USB`-a, šalje komande za kontrolu mehanizma.

Kontroler je baziran na `Arduino Nano` razvojnoj ploči koja se koristi za određivanje ugla otklona donjih poluga pomoću šest potenciometara. Dobijene vrednosti se pretvaraju i šalju u okviru odgovarajuće poruke za kontrolu direktne kinematike.



Slika 6.91 - Kontroler direktne kinematike `MP-6DOF-M-CTRLR`

Na slici 6.92 je prikazana kontrola oba prototipa pomoću ovog kontrolera. Ovaj kontroler je veoma interesantan, jer kontrolisani mehanizam sve vreme kopira položaj kontrolera.



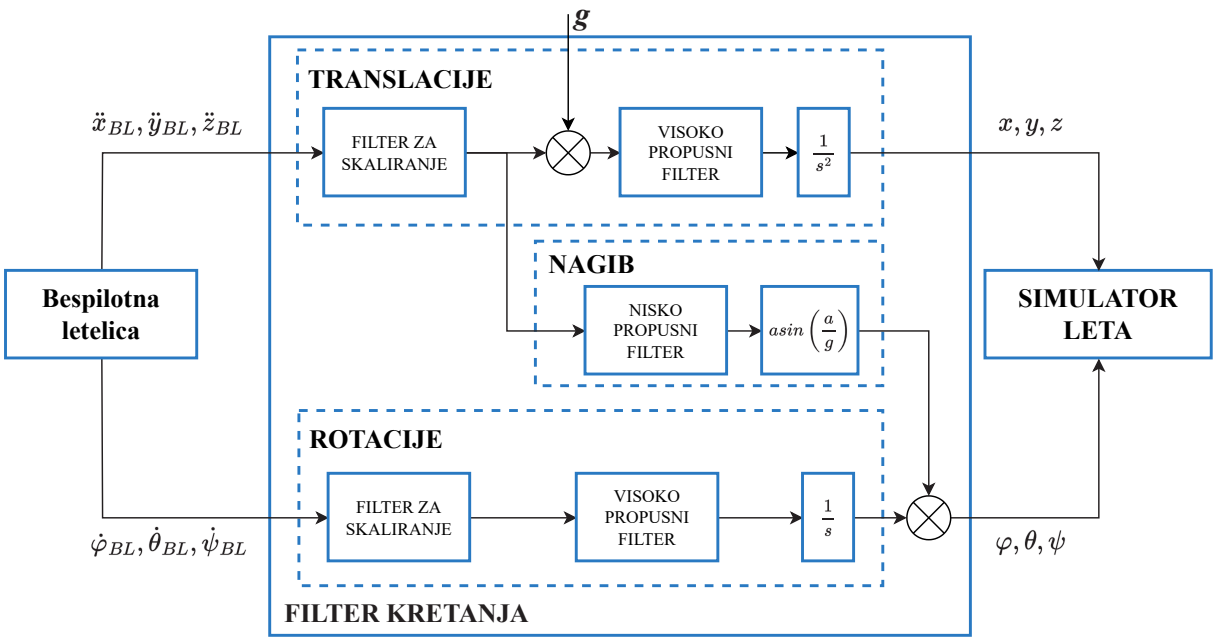
Slika 6.92 - Kontrola oba prototipa mehanizma pomoću `MP-6DOF-M-CTRLR` kontrolera

6.1.6 Filteri kretanja

Svrha filtera kretanja je transformacija trajektorije kretanja bespilotne letelice, generisane dinamičkim modelom bespilotne letelice ili u slučaju sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi merenjem na letelici. Kretanje letelice je takvo da nije ograničeno po maksimalnom translatornom pomeranju i po uglovima rotacije, kao što je to slučaj sa platformom simulatora. Zato je potrebno transformisati trajektoriju u upravljačke veličine simulatora leta koje pilotu omogućavaju realističan osećaj o kretanju i to u granicama radnog prostora mehanizma simulatora leta.

Sve dok vestibularni, vizuelni i ostali sistemi centralnog nervnog sistema pružaju pilotu istu percepciju kao da se nalazi u letelici, proces donošenja odluka i generisanje korektivnog upravljanja (pilotiranje) biće isto [147].

Projektovanje efikasnog filtera kretanja je veoma kompleksan problem. Potrebno je osigurati stabilnost i robusnost kako bi se sprečila mehanička oštećenja konstrukcije simulatora. Realističan osećaj o kretanju je jedan od problema za koje ne postoji opšte prihvaćena definicija na osnovu koje bi bilo moguće definisati koliko je efikasan filter kretanja. Na sledećem dijagramu (slika 6.93) je dat primer klasičnog filtera kretanja.



Slika 6.93 - Blok dijagram klasičnog filtera kretanja

Možda najjednostavniji filter i za razumevanje i za implementaciju bi bio kada bi na dijagramu 6.93 između bespilotne letelice i simulatora leta ostao samo blok za skaliranje ubrzanja i skaliranje uglova rotacije.

Osim ovakvog klasičnog filtera u literaturi se mogu naći i adaptivni algoritmi, filteri optimalne kontrole, hibridni klasični-adaptivni filteri, kao i filteri sa podešavajućim skaliranjem, filteri ugla skretanja i drugi.

6.1.7 Značenje kodova statusa

Da bi se informacija o stanju jednog podsistema prosledila drugom, umesto slanja dugih tekstualnih poruka (na osnovu kojih je komplikovanije programirati reakcije), koriste se enumeracije (kao što je `enum` u programskom jeziku `C`). Na taj način definišu se konstante koje

imaju brojčanu vrednost, što olakšava njihovo slanje i analizu. U okviru svih programa, prikazanih u ovoj disertaciji, implementirane su tri enumeracije i to za status platforme, proračuna i aktuatora.

6.1.7.1 Status platforme

Prilikom rada simulatora leta vrši se određeni broj provera podsistema. Status platforme može imati sledeće vrednosti:

- `0` - nije inicijalizovana platforma.
- `1` - započeto čekanje komunikacije sa SB.
- `2` - čekanje komunikacije sa SB.
- `3` - započeto vraćanje u početni položaj.
- `4` - vraćanje u početni položaj.
- `5` - normalan rad.
- `6` - pritisnut taster za hitno zaustavljanje.
- `7` - izmerena pozicija nije validna.
- `od 111 do 116` - isteklo vreme čekanja za SB.
- `od 121 do 126` - isteklo vreme za nove podatke od SB.
- `od 131 do 136` - greška na SB.

6.1.7.2 Status proračuna

Prilikom proračuna kinematike mehanizma vrši se određeni broj provera za svaku nogu mehanizma. Status (`MP.e`) može imati sledeće vrednosti:

- `0` - uspešno izvršen proračun.
- `1` - greška prilikom izračunavanja ugla otklona donje poluge (α) prilikom proračuna inverzne kinematike.
- `2` - greška usled toga što je ugao u gornjem sfernom zglobu (λ_{uj}) veći od dozvoljenog.
- `3` - greška usled toga što je ugao u donjem sfernom zglobu (λ_{lj}) veći od dozvoljenog.
- `4` - greška usled toga što može doći do sudaranja između donjih poluga.
- `5` - greška usled toga što je ugao ugradnje fiksnih sfernih zglobova (μ_{ul}) van potrebnog opsega.
- `6` - greška prilikom izračunavanja ugla otklona donje poluge (α) prilikom proračuna direktne kinematike.
- `7` - greška usled singularnosti Jakobijeve matrice (J).
- `8` - greška usled dostizanja maksimalnog broja iteracija u okviru proračuna direktne kinematike.

6.1.7.3 Status aktuatora

Prilikom rada svakog od aktuatora se vrši određeni broj provera rada motora, senzora ugla i frekventnog regulatora. Status rada aktuatora (`SB_STATE`) može imati sledeće vrednosti:

- `0` - aktuator nije inicijalizovan.
- `1` - sprečen rad aktuatora.
- `2` - normalan rad aktuatora.
- `3` - komunikacija sa senzorom ugla nije uspostavljena.
- `4` - komunikacija sa frekventnim regulatorom nije uspostavljena.
- `5` - komunikacija sa frekventnim regulatorom je prekinuta.
- `6` - komunikacija sa frekventnim regulatorom nije validna.
- `7` - komunikacija sa senzorom ugla je prekinuta.
- `8` - motor ili senzor ugla nisu u funkciji.
- `od 101 do 134` - greška u radu frekventnog regulatora (kod greške koji odgovara onom u uputstvu frekventnog regulatora je $S - 100$, odnosno od 1 do 34).

6.1.8 Definicija paketa i programa za komunikaciju

Da bi se primenio specijalni binarni komunikacioni protokol `PRDC_SCBP` koji je detaljno opisan i čija implementacija je data u prilogu T, neophodno je formirati binarne pakete, odnosno proceduru za transformaciju različitih tipova podataka u binaran niz podataka i nazad.

Transformacija se u slučaju programskih jezika `C` i `C++` vrši prostim korišćenjem `unije`, kao što je ona data u listingu 6.1, između niza bajtova i strukture kojom je definisan paket (paket i ovaj niz dele isti prostor u memoriji pa je moguće ili koristiti strukturu ili direktno niz bajtova, što je prigodno za slanje i primanje podataka).

```
1 union data_t{
2   packet_t p;
3   uint8_t bytes[sizeof(packet_t)];
4 };
```

Listing 6.1 - Primer unije za transformaciju

U nastavku se za kontroler misije smatra računarski program koji generiše poruke ili različiti fizički kontroleri koji generišu iste poruke.

Sve strukture se definišu sa atributom `packed` koji obezbeđuje da se svi elementi strukture raspoređuju tačno po redosledu definisanja, bez uobičajenog poravnavanja u memoriji. Time se sprečava nepotrebno povećavanje dužine paketa za slanje podataka.

6.1.8.1 MP-6DOF-XXX Control Board sa Servo Board i kontrolerom misije

U listingu 6.2 je data struktura `packetOutSB_t` koja se koristi za poruku od `Control Board` elektronike za `Servo Board` elektronike. U tabeli 6.8 je objašnjena i jedina poruka koja koristi ovu strukturu.

```
struct __attribute__((packed)) packetOutSB_t {
  uint8_t enable;
  float alpha_w;
  float f_max;
  float Kp;
  float Ki;
  float Kd;
  uint32_t N;
};
```

Listing 6.2 - Struktura `packetOutSB`

Ovo je struktura podataka paketa u programskim jezicima `C` i `C++`, za JavaScript se koristi biblioteka `PRDC_PACKET` koja je data u prilogu U.

Tabela 6.8 - Poruka strukture `packetOutSB`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
	E	α_w	f_{max}	k_P	k_I	k_D	N
Objašnjenje	• E - kontrola statusa rada aktuatora. $E = \begin{cases} 0, & \text{Sprečiti rad aktuatora} \\ 1, & \text{Omogućiti rad aktuatora} \end{cases}$ • α_w - vrednost željenog ugla otklona donje poluge. • f_{max} - maksimalna učestanost naizmenične struje aktuatora. • k_P - koeficijent proporcionalnog dejstva regulatora. • k_I - koeficijent integralnog dejstva regulatora. • k_D - koeficijent diferencijalnog dejstva regulatora. • N - redni broj paketa.						

U listingu 6.3 je data struktura `packetInSB_t` koja se koristi za poruku od `Servo Board` elektronike za `Control Board` elektroniku. U tabeli 6.9 je objašnjena i jedina poruka koja koristi ovu strukturu.

```
struct __attribute__((packed)) packetInSB_t {
  uint8_t state;
  float alpha_m;
  float temp;
  float f_out;
  uint8_t switches;
  uint32_t N;
};
```

Listing 6.3 - Struktura `packetInSB`

Tabela 6.9 - Poruka strukture `packetInSB`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
	S	α_m	T	f_{OUT}	*switches*	N
Objašnjenje	• S - status rada aktuatora. • α_m - trenutna izmerena vrednost ugla otklona donje poluge. • T - temperatura senzora ugla na reduktoru. • f_{OUT} - trenutna učestanost naizmenične struje u Hz. • *switches* - stanja prekidača (bit maske). • N - redni broj paketa.					

U listingu 6.4 je data struktura `packetInMC_t` koja se koristi za poruke od kontrolera misije za `Control Board` elektroniku. Kao što se vidi iz listinga strukture, prvi element je identifikacioni broj poruke, pa su tako u tabelama 6.10, 6.11, 6.12 i 6.13 data objašnjenja za svaku pojedinačnu poruku koja koristi ovu strukturu. U svim porukama je N redni broj paketa.

```
1 struct __attribute__((packed)) packetInMC_t {
2   uint16_t ID;
3   uint16_t P[6];
4   uint32_t N;
5 };
```

Listing 6.4 - Struktura `packetInMC`

Tabela 6.10 - Poruka ID = 11 strukture `packetInMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 11$	α_1	α_2	α_3	α_4	α_5	α_6	N
Objašnjenje	Podaci od α_1 do α_6 u paketu predstavljaju unutrašnje koordinate (uglove otklona donjih poluga mehanizma) i definišu se na sledeći način: $$\alpha_i = \left([-180° : 180°] + 180\right) \cdot \frac{65534}{360}$$							

Tabela 6.11 - Poruka ID = 12 strukture `packetInMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 12$	x	y	z	$rx\ (\varphi)$	$ry\ (\theta)$	$rz\ (\varphi)$	N
Objašnjenje	Podaci od P2 do P7 u paketu predstavljaju relativne vrednosti spoljašnjih koordinata platforme mehanizma i definišu se na sledeći način: $$\text{Pi} = \left([-100\% : 100\%] + 100\right) \cdot \frac{65534}{200}$$							

Tabela 6.12 - Poruka ID = 13 strukture `packetInMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 13$	x	y	z	$rx\ (\varphi)$	$ry\ (\theta)$	$rz\ (\psi)$	N
Objašnjenje	Podaci od P2 do P7 u paketu predstavljaju apsolutne vrednosti spoljašnjih koordinata platforme mehanizma i definišu se na sledeći način: $$x = \big([-1000\text{ mm} : 1000\text{ mm}] + 1000\big) \cdot \frac{65534}{2000}$$ $$y = \big([-1000\text{ mm} : 1000\text{ mm}] + 1000\big) \cdot \frac{65534}{2000}$$ $$z = \big([-500\text{ mm} : 500\text{ mm}] + 500\big) \cdot \frac{65534}{1000}$$ $$rx = \varphi = \big([-45^\circ : 45^\circ] + 45\big) \cdot \frac{65534}{90}$$ $$ry = \theta = \big([-45^\circ : 45^\circ] + 45\big) \cdot \frac{65534}{90}$$ $$rz = \psi = \big([-45^\circ : 45^\circ] + 45\big) \cdot \frac{65534}{90}$$							

Tabela 6.13 - Poruka ID = 201 strukture `packetInMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 201$	f_{max}	k_P	k_I	k_D	/	/	N
Objašnjenje	Podaci u paketu predstavljaju konfiguracione koeficijente i svaki je pomnožen sa 1000 pre slanja							

U listingu 6.5 je data struktura `packetOutMC_t` koja se koristi za poruke od `Control Board` elektronike za kontrolere misije. Kao što se vidi iz listinga strukture, prvi element je identifikacioni broj poruke, pa su tako u tabelama 6.14, 6.15, 6.16, 6.17, 6.18, 6.19, 6.20 i 6.21 data objašnjenja za svaku pojedinačnu poruku koja koristi ovu strukturu. U svim porukama je N redni broj paketa.

```
1 struct __attribute__((packed)) packetOutMC_t {
2   uint8_t ID;
3   float P[6];
4   uint32_t N;
5 };
```

Listing 6.5 - Struktura `packetOutMC`

Tabela 6.14 - Poruka ID = 0 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 0$	/	/	/	/	/	/	N
Objašnjenje	Heartbeat signal za detekciju i održavanje veze.							

Tabela 6.15 - Poruka ID = 101 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 101$	S	IO	MC	MP	M6RSS	/	N
Objašnjenje	Podaci u paketu predstavljaju stanja odgovarajućih podsistema.							

Tabela 6.16 - Poruka ID = 102 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 102$	e_1	e_2	e_3	e_4	e_5	e_6	N
Objašnjenje	Podaci u paketu predstavljaju vrednosti vektora `M6RSS.e`, odnosno status proračuna kinematike mehanizma.							

Tabela 6.17 - Poruka ID = 103 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 103$	s_1	s_2	s_3	s_4	s_5	s_6	N
Objašnjenje	Podaci u paketu predstavljaju vrednosti vektora `SB_state`, odnosno status aktuatora.							

Tabela 6.18 - Poruka ID = 104 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 104$	$\alpha_{m,1}$	$\alpha_{m,2}$	$\alpha_{m,3}$	$\alpha_{m,4}$	$\alpha_{m,5}$	$\alpha_{m,6}$	N
Objašnjenje	Podaci u paketu predstavljaju vrednosti vektora α_m, odnosno izmerene uglove donjih poluga.							

Tabela 6.19 - Poruka ID = 105 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 105$	T_1	T_2	T_3	T_4	T_5	T_6	N
Objašnjenje	Podaci u paketu predstavljaju vrednosti vektora `temp`, odnosno temperature senzora ugla na reduktorima.							

Tabela 6.20 - Poruka ID = 106 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 106$	f_1	f_2	f_3	f_4	f_5	f_6	N
Objašnjenje	Podaci u paketu predstavljaju vrednosti vektora f_{out}, odnosno izlazne frekvencije za upravljanje brzinom motora.							

Tabela 6.21 - Poruka ID = 107 strukture `packetOutMC`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
	$ID = 107$	f_{max}	k_P	k_I	k_D	/	/	N
Objašnjenje	Podaci u paketu predstavljaju konfiguracione koeficijente i svaki je pomnožen sa 1000 pre slanja.							

6.1.8.2 MP-6DOF-XXX Servo Board sa Angle Sensor

U listingu 6.6 je data struktura `packetInAS_t` koja se koristi za poruku od `Angle Sensor` elektronike za `Servo Board` elektronike. U tabeli 6.22 je objašnjena i jedina poruka koja koristi ovu strukturu.

```
1 struct __attribute__((packed)) packetInAS_t {
2   float angle;
3   float temp;
4   uint32_t N;
5 };
```

Listing 6.6 - Struktura `packetInAS`

Tabela 6.22 - Poruka strukture `packetInAS`

Struktura	**P1**	**P2**	**P3**
	α_m	T	N
Objašnjenje	• α_m - trenutna izmerena vrednost ugla. • T - temperatura u okolini senzora. • N - redni broj paketa.		

6.1.8.3 MP-6DOF-M sa kontrolerom misije

U listingu 6.7 je data struktura `packetInOut_t` koja se koristi za poruke od `MP-6DOF-M` elektronike za kontrolere misije i obrnuto. Kao što se vidi iz listinga strukture, prvi element je identifikacioni broj poruke, pa su tako u tabelama 6.23, 6.24, 6.25 i 6.26 data objašnjenja za svaku pojedinačnu poruku koja koristi ovu strukturu. Ove poruke se koriste za slanje podataka od `MP-6DOF-M` elektronike do kontrolera misije. Sa druge strane, poruke od kontrolera misije za `MP-6DOF-M` elektroniku su gotovo iste kao one u tabelama 6.10, 6.11 i 6.12 ali bez parametra N kao rednog broja paketa.

```
struct __attribute__((packed)) packetInOut_t {
  uint16_t ID;
  uint16_t P[6];
};
```

Listing 6.7 - Struktura `packetInOut`

Tabela 6.23 - Poruka ID = 1 strukture `packetInOut`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
	$ID = 1$	S_1	S_2	S_3	S_4	S_5	S_6
Objašnjenje	Podaci od S_1 do S_6 u paketu predstavljaju kontrole stanja aktuatora i definišu se na sledeći način: $$S_i = \begin{cases} 0, & \text{Sprečiti rad } i\text{-tog aktuatora} \\ 1, & \text{Omogućiti rad } i\text{-tog aktuatora} \end{cases}$$						

Tabela 6.24 - Poruka ID = 7 strukture `packetInOut`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
	$ID = 7$	W_{h1}	W_{h2}	W_{h3}	W_{h4}	W_{h5}	W_{h6}
Objašnjenje	Podaci od W_{h1} do W_{h6} u paketu predstavljaju širine pulsa koje odgovaraju horizontalnom položaju donjih poluga mehanizma i definišu se na sledeći način: $$W_{hi} = [0:3000] \cdot \frac{65534}{3000}$$						

Tabela 6.25 - Poruka ID = 8 strukture `packetInOut`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
	$ID = 8$	W_{l1}	W_{l2}	W_{l3}	W_{l4}	W_{l5}	W_{l6}
Objašnjenje	Podaci od W_{l1} do W_{l6} u paketu predstavljaju širine pulsa koje odgovaraju položaju donjih poluga prilikom $\alpha_i = -90°$ mehanizma i definišu se na sledeći način: $$W_{li} = [0:3000] \cdot \frac{65534}{3000}$$						

Tabela 6.26 - Poruka ID = 9 strukture `packetInOut`

Struktura	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
	$ID = 9$	W_{u1}	W_{u2}	W_{u3}	W_{u4}	W_{u5}	W_{u6}
Objašnjenje	Podaci od W_{u1} do W_{u6} u paketu predstavljaju širine pulsa koje odgovaraju položaju donjih poluga prilikom $\alpha_i = +90°$ mehanizma i definišu se na sledeći način: $$W_{ui} = [0:3000] \cdot \frac{65534}{3000}$$						

6.2 Letelice

Letelice korišćene u realizaciji ove disertacije bile su ili nabavljene ili barem delimično razvijene baš za ovu namenu. U nastavku su detaljno opisane letelice i način na koji su one korišćene.

6.2.1 Multikopter SYMA X6

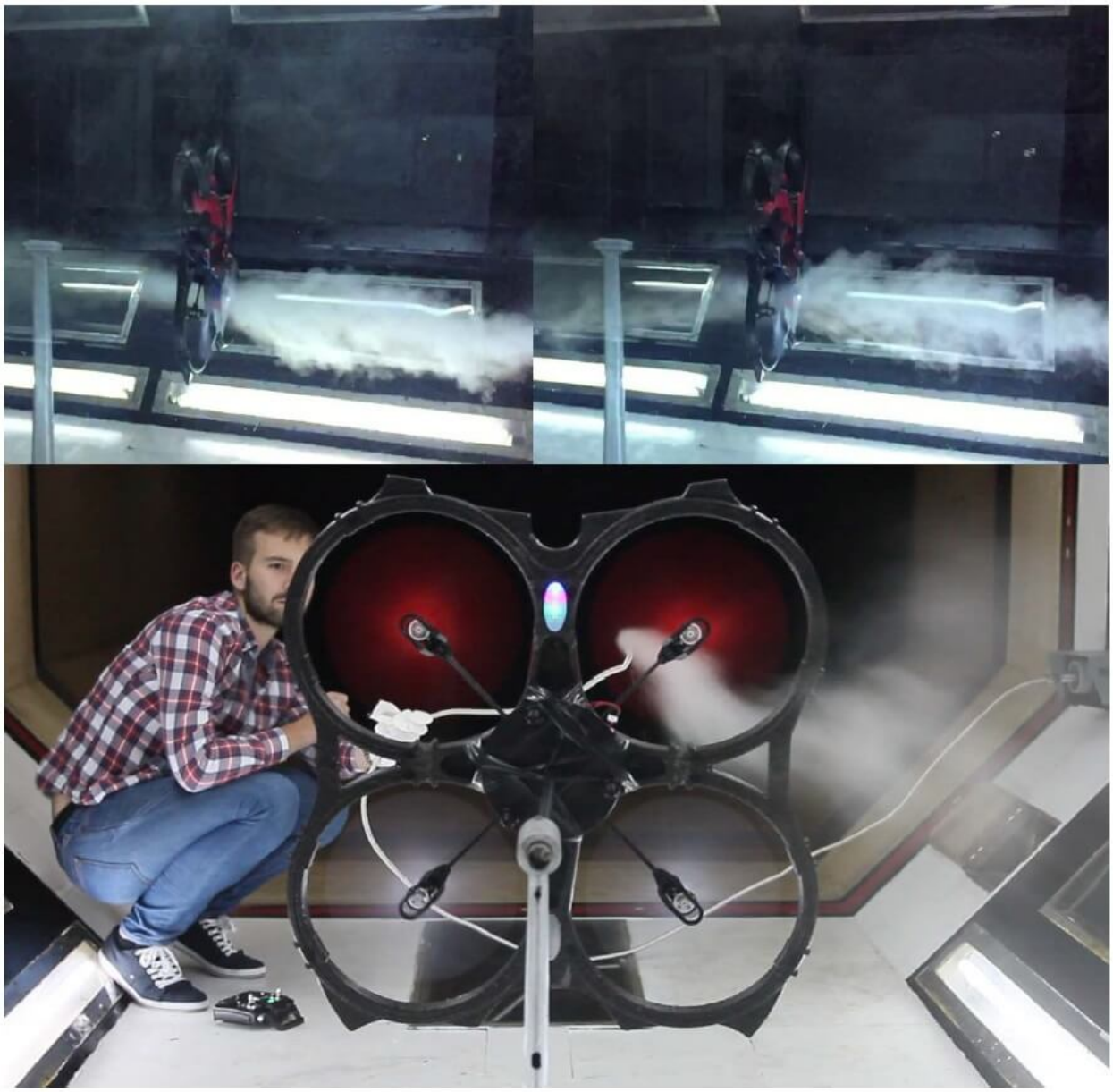
Prva letelica, koja je nabavljena 2015. godine, je bila električni multikopter kompanije *Syma* oznake *X6*. Na slici 6.94 je prikazana kompletna konfiguracija, uz multikopter se isporučuje i električna baterija, punjač baterije sa balansiranjem, jedan set rezervnih elisa i daljinski upravljač.



Slika 6.94 - Konfiguracija multikoptera SYMA X6

Osnovna frekvencija radio veze je 2.4 GHz i domet je oko 50 metara. Letelica koristi žiroskop i akcelerometar za stabilizaciju. Ova letelica je posebno interesantna zbog svojih dimenzija, širine i dužine 580 mm i visine 100 mm. Baterija je Li-Ion napona 7.4V i kapaciteta 850mAh, što omogućava let u trajanju od oko 7 minuta.

Osnovna svrha ove letelice je bila upoznavanje sa stanjem tehnike u tom trenutku, početnim ovladavanjem veštine daljinskog pilotiranja kao i za neka početna aerodinamička ispitivanja. U aerotunelu Mašinskog fakulteta Univerziteta u Beogradu urađena je vizualizacija strujne cevi rotora, što je prikazano na slici 6.95.



Slika 6.95 - Aerotunelsko ispitivanje multikoptera Syma X6

6.2.2 Kvadkopter QFLY-250

Druga letelica je samostalno izrađen multikopter sa delovima iz uvoza, koji je nazvan *QFLY-250*. Zapravo se radi o kvadkopteru (letelici sa četiri rotora) sa prečnikom elisa od 5 inča (prikazan na slici 6.96). Letelica je izrađena 2016. godine i sve komponente su uvezene iz inostranstva jer u tom trenutku na domaćem tržištu nije postojao nijedan distributer delova za ovu vrstu proizvoda.

Ova letelica je postala osnovna platforma za istraživanje sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi zajedno sa umanjenim prototipom simulatora leta.



Slika 6.96 - Kvadkopter QFLY-250 klase 250

Na slici 6.97 je prikazano poletanje letelice.



Slika 6.97 - Poletanje kvadkoptera QFLY-250

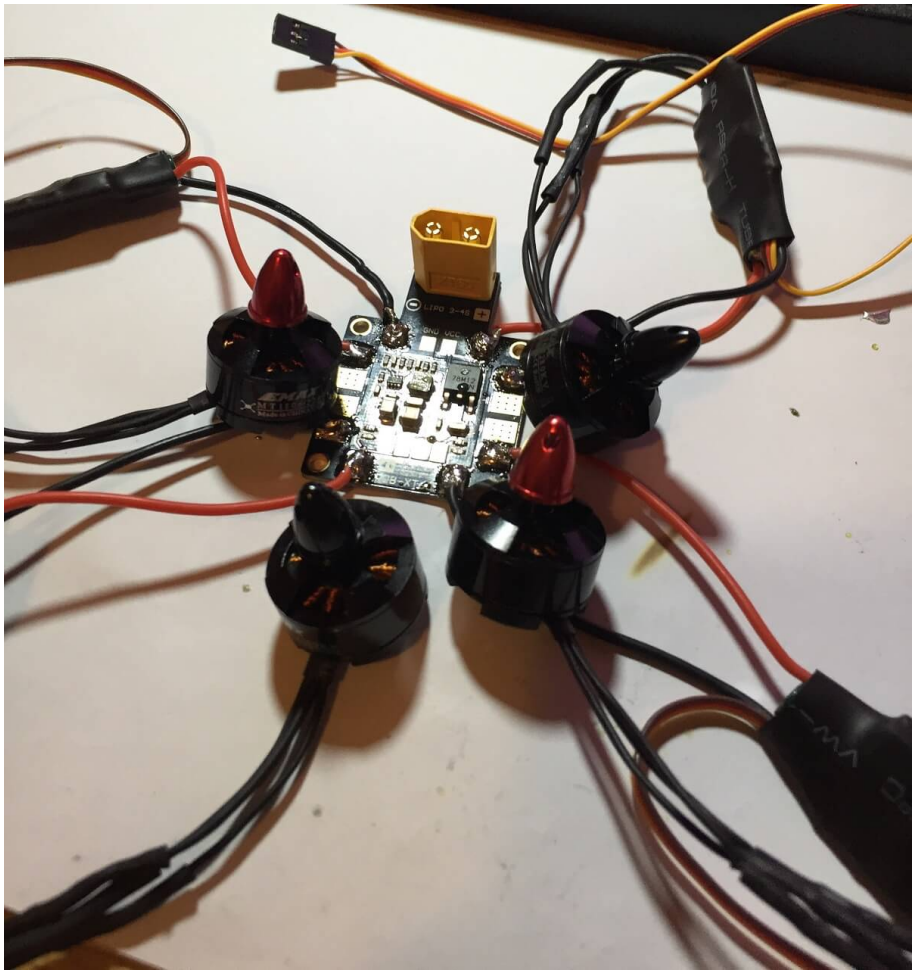
Dijagonalno rastojanje između osa obrtanja rotora je 250 mm, što ovu letelicu svrstava u klasu 250. Konstrukcija je izrađena od elemenata dobijenih iz ploča od kompozitnog materijala sa karbonskim vlaknima. U tabeli 6.27 je data specifikacija delova ovog kvadkoptera.

Tabela 6.27 - Specifikacija delova letelice QFLY-250

Komponenta	**Proizvođač i model**
Aktuatori	EMAX MT1806 4 kom
ESC	Emax 12A BLHELI
Kontroler	OpenPilot CC3D
PDB	Matek PDB-XT60
Elise	Gemfan 5030
Kamera	Eachine 700TVL 1/3" CMOS
Baterija	Turnigy LiPo 3S 2200 mAh

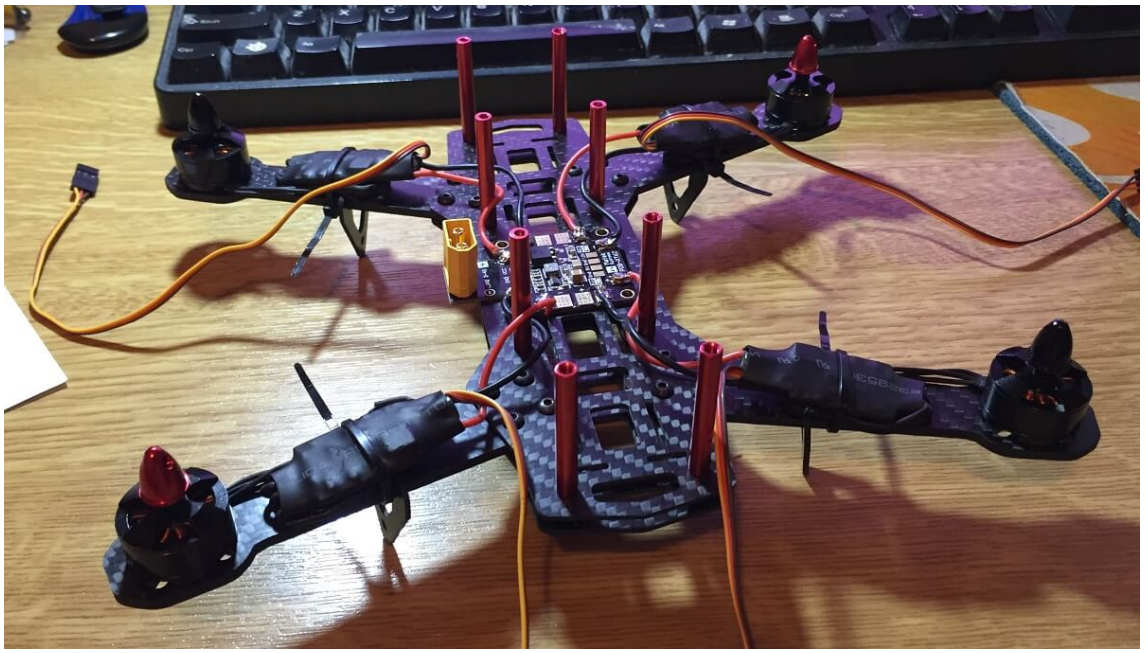
Ovakve letelice su najčešće namenjene za upravljanje iz prvog lica pa se tako operateri takmiče u trkanju na specijalno napravljenim poligonima. Osim toga, ovaj tip letelica sve više pronalazi primenu i u vojne svrhe.

Za sklapanje ovakve letelice je potreban samo osnovni alat i poznavanje rada sa lemilicom kako bi se sklopio električni sistem. Na slici 6.98 je prikazano lemljenje žica aktuatora na specijalnu štampanu ploču za distribuciju električne energije. Ova ploča je veoma korisna jer na sebi ima i konektor za bateriju što olakšava uključivanje same letelice.



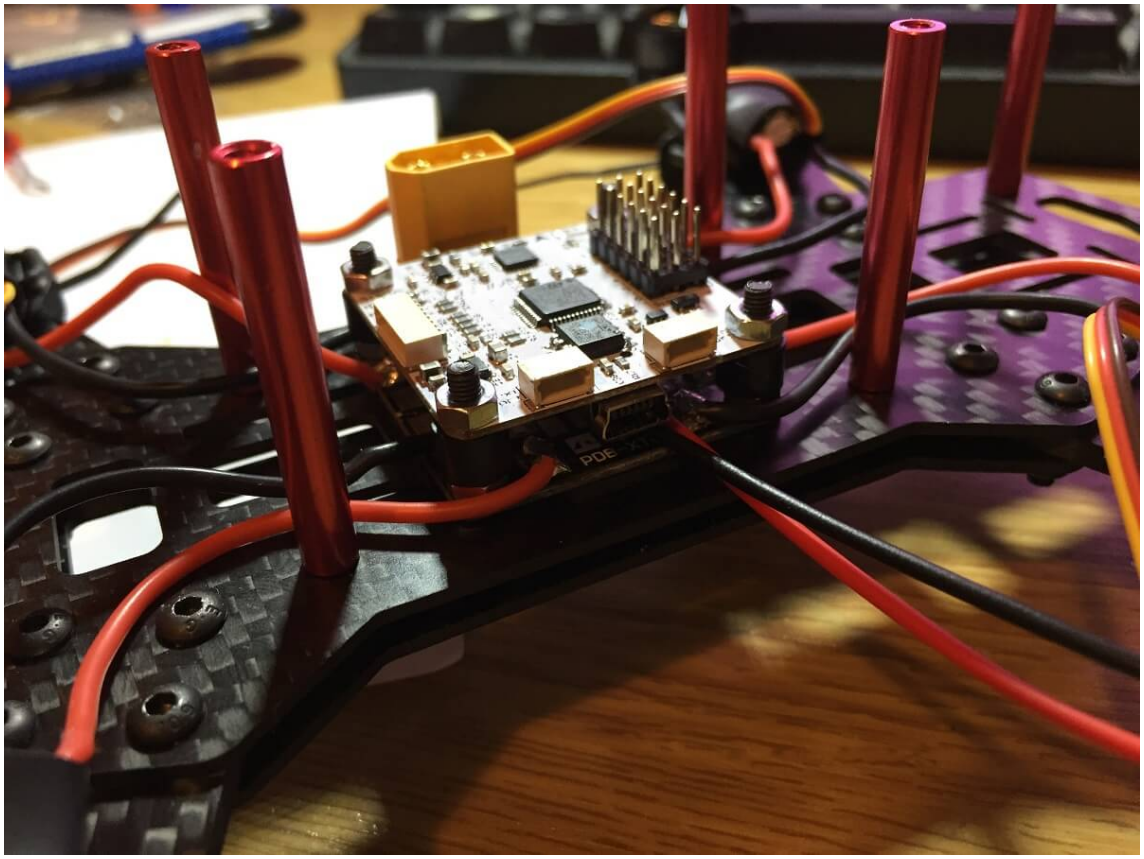
Slika 6.98 - Spajanje aktuatora sa pločom za distribuciju električne energije

Nakon lemljenja je moguće montiranje motora i elektronike na konstrukciju kvadkoptera. Na slici 6.99 je prikazano montiranje.



Slika 6.99 - Montaža motora i elektronike na konstrukciju kvadkoptera

Štampana ploča za distribuciju električne energije i kontroler leta su projektovani tako da se vertikalno montiraju jedna iznad druge kako bi zauzimali minimalno prostora. Na slici 6.100 je prikazana montaža kontrolera leta.



Slika 6.100 - Montaža kontrolera leta na konstrukciju kvadkoptera

Nakon toga preostaje još samo da se montiraju kamera, analogni video-predajnik i radio-prijemnik kako bi letelica bila kompletna. Za daljinsko upravljanje se koristi daljinski upravljač *FLYSKY FS-i6* koji je prikazan na slici 6.101.



Slika 6.101 - Daljinski upravljač FLYSKY FS-i6

Projekat *LibrePilot* je korišćen za programiranje kontrolera leta. U okviru konfiguracije se podešava broj i prostorni raspored aktuatora, kalibrišu se integrisani senzori za stabilizaciju leta, ulazni radio kanali za kontrolu kao i izlazni signali za aktuatore, nakon čega je letelica spremna za let. Na slici 6.102 je prikazana konfiguracija za let, sa daljinskim upravljačem koji na sebi ima i analogni ekran za prikaz video snimka sa letelice u realnom vremenu. Osim ovakvog ekrana mogu se koristiti i naočare sa sličnim ekranima.



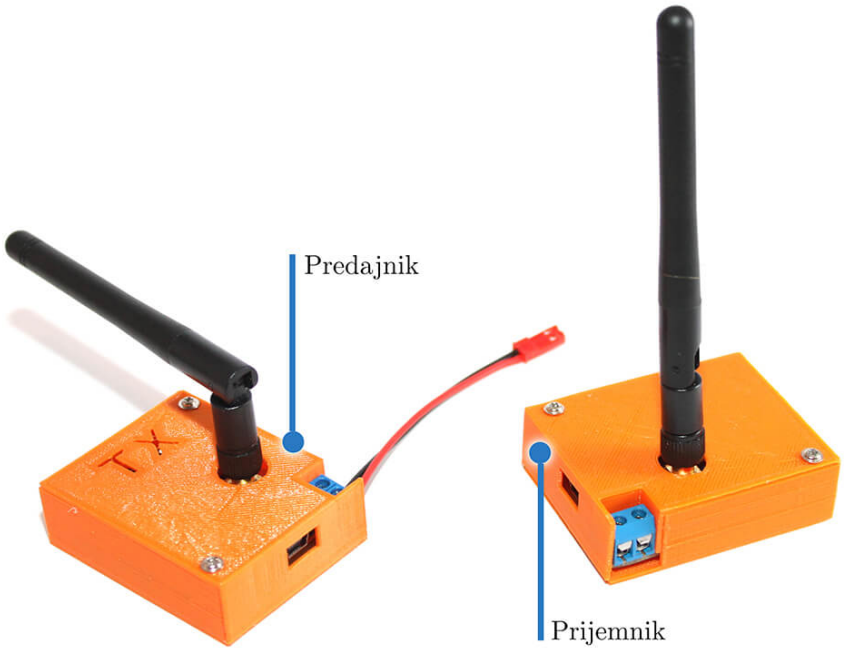
Slika 6.102 - Daljinski upravljač za kvadkopter sa analognim ekranom

Kvadkopter *QFLY-250* je prvi put javno prikazan na 61. međunarodnom sajmu tehnike i tehničkih dostignuća u Beogradu, maja 2017. godine, na štandu Mašinskog fakulteta Univerziteta u Beogradu (prikazano na slici 6.103).



Slika 6.103 - Kvadkopter QFLY-250 na sajmu tehnike 2017. godine u Beogradu

Da bi se testirao sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi bilo je potrebno omogućiti digitalnu bežičnu komunikaciju između kvadkoptera i umanjenog prototipa simulatora leta. Napravljeni su radio-prijemnik i radio-predajnik (prikazani na slici 6.104) zasnovani na integralnom kolu *nRF24L01+* koje je povezano sa *Arduino Nano* razvojnom pločom zasnovanom na *ATmega328P* mikrokontroleru.



Slika 6.104 - Radio-prijemnik i predajnik za bežičnu komunikaciju

Ovaj modul za bežičnu komunikaciju koristi osnovnu frekvenciju u opsegu od 2.4 do 2.4835 GHz i povezuje sa mikrokontrolerom putem SPI komunikacije[148].

U okviru modula predajnika (prikazan na slici 6.105) koji se montira na konstrukciju kvadkoptera se nalazi i inercijalni merni sistem *MPU-9250*. Ovo je *mikro-elektro-mehanički sistem* (*MEMS*) koji se sastoji od dva čipa *MPU-6500* i *AK8963* digitalnog kompasa. *MPU-6500* se sastoji iz žiroskopskog senzora, senzora ubrzanja i procesora za obradu digitalnog signala kojim se radi filtriranje signala i određuje orijentacija u prostoru. Povezivanje ovog modula sa mikrokontrolerom je putem I2C komunikacije. Predajnik se napaja sa baterije kvadkoptera za šta su obezbeđeni odgovarajući terminali. Konstrukcija kućišta je izrađena 3D štampanjem od plastike.

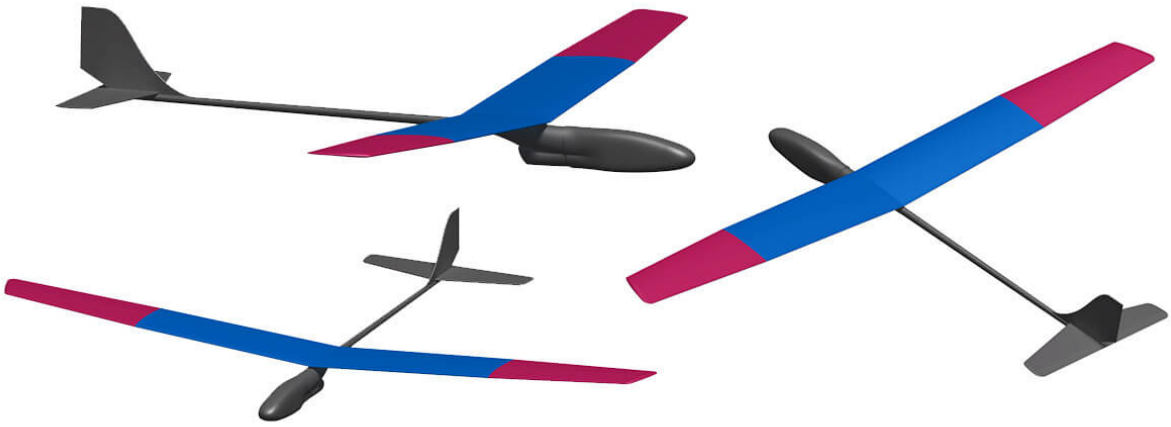


Slika 6.105 - Predajnik sa senzorom ubrzanja i orijentacije

Osim napajanja sa kvadkoptera, predajnik je potpuno nezavistan u pogledu senzora, procesora i radio-veze, dovoljno je malih dimenzija i male mase i kao takav se može montirati na praktično bilo koju letelicu. Napisani su svi potrebni programi za predajnik, prijemnik i za računar koji kontroliše simulator leta. Konačno ispitivanje je prikazano u poglavlju 7.3.

6.2.3 Raketoplan LABFLY-RM01

Prva bespilotna letelica sa fiksnim krilom koja je korišćena je letelica pod nazivom *LABFLY-RM01*. Ova letelica nosi oznaku RM jer je model modifikovanog radio-upravljanog raketoplana. Radio-upravljani raketoplan je takmičarska kategorija u raketnom modelarstvu, zvanična oznaka ove kategorije je S-8E/P. U ovoj kategoriji koristi se raketni motor na čvrsto gorivo prečnika 18 mm i dužine 70 mm. Oznaka E se odnosi na srednji potisak koji je potrebno da bude u opsegu od 20.01 do 40 Ns. Oznaka P se odnosi na precizno sletanje. Trodimenzionalni model letelice je prikazan na slici 6.106.



Slika 6.106 - 3D model letelice LABFLY-RM01

Nakon lansiranja, zadatak je održati raketoplan komandama tačno 6 minuta u vazduhu i na samom sletanju pogoditi definisani cilj na zemlji. Pravilnikom takmičenja je definisano bodovanje i broj bodova zavisi od preciznosti, kako vremenske tako i preciznosti prilikom pogađanja cilja tokom sletanja. Takođe, pravilnikom je definisana geometrija samog radio-komandovanog raketoplana kao i sistemi koji smeju biti korišćeni prilikom njegovog upravljanja. Od geometrijskih karakteristika ove letelice, razmah krila je 1200 mm, površina krila 0.124 m^2, tetiva u korenu krila 120 mm i dužina trupa je 780 mm. U tabeli 6.28 je data specifikacija delova modifikovane verzije raketoplana.

Tabela 6.28 - Specifikacija delova letelice LABFLY-RM01

Komponenta	**Proizvođač i model**
Aktuatori	EMAX ES9051 4 kom
Kontroler	Arduino Nano 2 kom
Radio modul	nRF24L01+
IMU	MPU9250
Barometar	BMP280
GNSS modul	NEO-M8N

U cilju postizanja što je moguće manje mase letelice, a da se sa druge strane sačuva čvrstoća, prilikom izrade je korišćeno više različitih materijala i korišćene su specifične tehnologije izrade. Od materijala se najčešće koriste različiti kompozitni materijali i drvo balza.

Zid ramenjače krila raketoplana izrađen je od balze, dok su pojasevi izrađeni od kompozitnog materijala sa karbonskim vlaknima. Veza između ove dve vrste materijala ostvarena je epoksidnim dvokomponentnim lepkom. Rebra su izrađena od balze i ojačana su karbonskim trakama.

Oplata krila izrađena je od *Icarex* platna. Karakteristika ovog poliesterskog platna je da prilikom zagrevanja dolazi do skupljanja u svim pravcima što je pogodno jer može da se ostvari zadovoljavajuća zategnutost oplate.

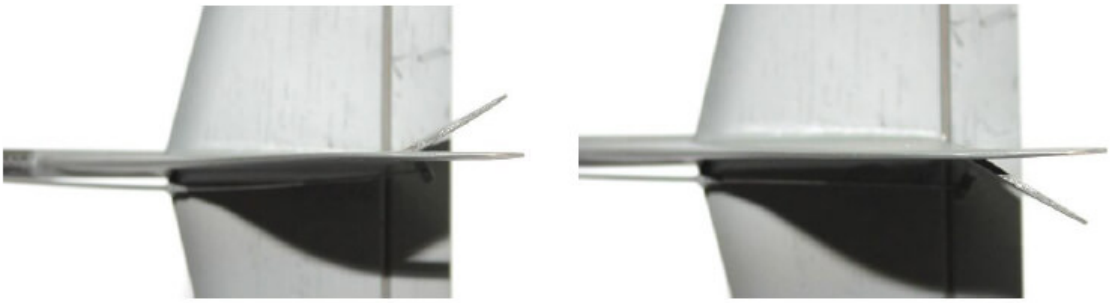
Komandne površine na krilima izrađene su od balze i presvučena su *Icarex* platnom. Deo krila koji se nalazi između napadne ivice i ramenjače izrađen je od kombinacije kevlara, karbona i epoksidne smole.

Repne površine raketoplana koje su prikazane na slici 6.107 su izrađene od balze. Balza je premazana epoksidnom smolom kako bi se postigla veća čvrstoća i otpornost na vlagu.



Slika 6.107 - Horizontalni i vertikalni rep sa komandnim površinama

Kod ovog tipa letelice se zarad uštede na masi i prostoru, i zbog jednostavnosti izrade, koriste mehanizmi sa elastičnim polugama za upravljanje komandnim površinama. Ugao otklona sve tri komandne površine (krmilo pravca, visine i krilca) se upravlja servo motorima preko poluga, zglobova i neistegljivih užadi (otkloni su prikazani na slikama 6.108, 6.109 i 6.110). Eksperimentalno su utvrđene dužine samih poluga i obrtni moment servo motora koji su potrebni kako bi se omogućilo efikasno upravljanje raketoplanom u letu.



Slika 6.108 - Otklon krmila pravca



Slika 6.109 - Otklon krmila visine



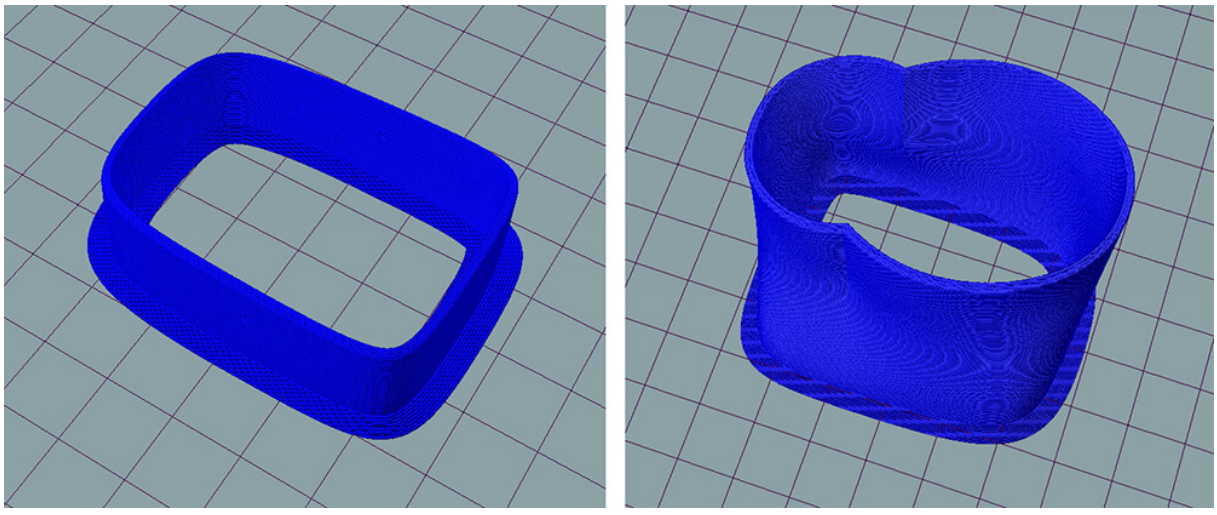
Slika 6.110 - Otklon krilaca

Zadnji deo trupa je u obliku konusne cevi i izrađen je od kombinacije karbonskih vlakana i epoksidne smole. Prednji deo trupa (nos letelice, prikazana na slici 6.111) je modifikovan za smeštaj kontrolera leta i baterija. Ovaj deo trupa je kod standardnog raketoplana izrađen od kombinacije staklenih vlakana i epoksidne smole. Zbog potrebe za brzim modifikacijama je za ovu letelicu izrađen 3D štampanjem od plastike.



Slika 6.111 - Prednji deo trupa letelice

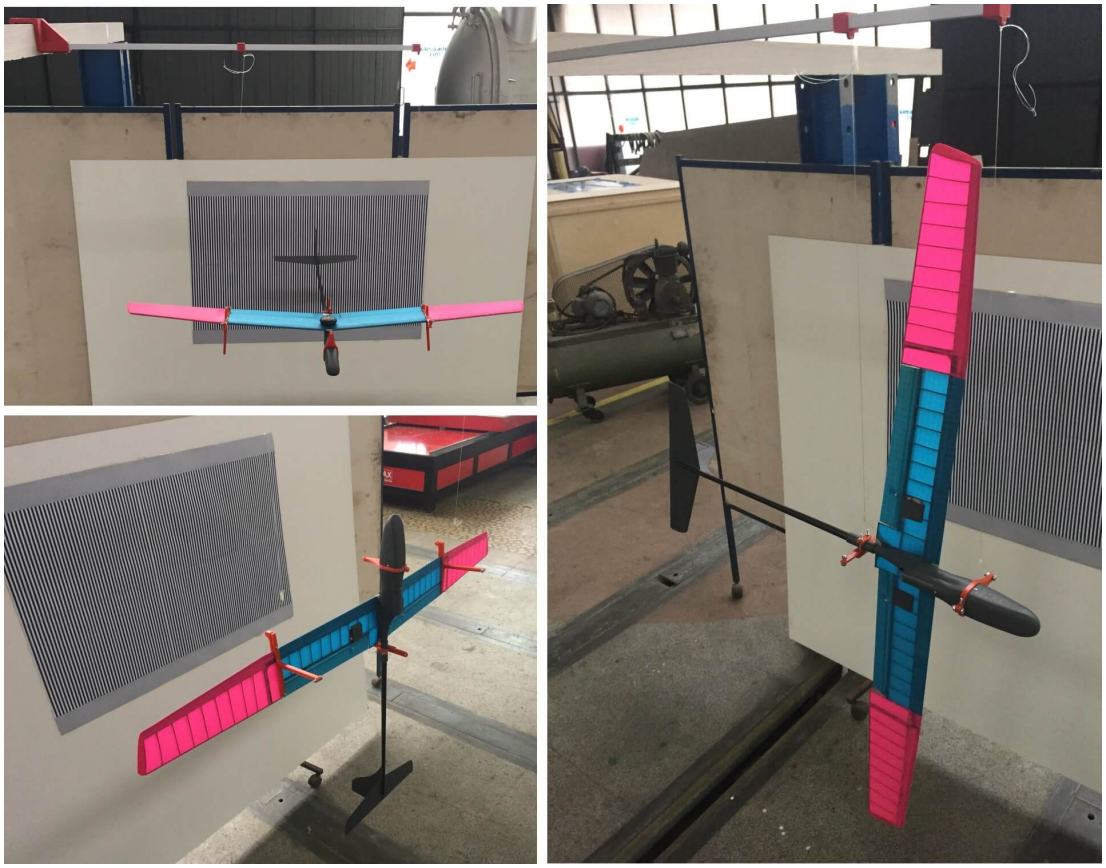
Na slici 6.112 je prikazana priprema modela nosa letelice za štampu na 3D štampaču.



Slika 6.112 - Priprema 3D modela nosa letelice za 3D štampanje

Ova letelica je upotrebljena za nekoliko analiza na osnovu kojih su stečena mnoga znanja u vezi sa bespilotnim letelicama sa fiksnim krilima. Najpre su analizirani otkloni krmila, njihovi mehanizmi i drugi uticaji.

Na ovoj letelici je vršeno eksperimentalno određivanje momenata inercije oko sve tri ose pomoću *bifilarnog klatna*. Na slici 6.113 je prikazana postavka za sva tri eksperimenta.



Slika 6.113 - Određivanje momenata inercije letelice LABFLY-RM01

Telo, čiji moment inercije se određuje, visi na dve paralelne niti poznate dužine i na poznatoj udaljenosti jedna od druge. Telo se zatim blago zarotira oko vertikalne ose i zatim pusti da slobodno osciluje. Ideja je da se iz perioda tih malih oscilacija izračuna moment inercije tela oko vertikalne ose, koja se može podesiti tako da prolazi kroz težište tela. Metoda za eksperimentalno određivanje momenta inercije celog putničkog aviona je prikazana u [149], dok je eksperimentalno određivanje momenta inercije bespilotne letelice pomoću složenog klatna prikazano u [150].

Određivanje mase, položaja težišta i momenta inercije predstavlja deo procesa identifikacije sistema kako bi se formirao matematički model dinamičkog sistema. Na osnovu ovakvog modela je za neke određene vrednosti ulaznih parametara matematički moguće odrediti vrednosti izlaznih parametara.

6.2.4 Letelica LABFLY-EM01

Prethodno prikazana letelica *LABFLY-RM01* je dodatno modifikovana, dodavanjem elektromotora i dobijena je letelica pod nazivom *LABFLY-EM01*. Letelica sa elektromotorom omogućava jednostavnija letna ispitivanja u odnosu na ona sa raketnim motorom, koji samo deo vremena obezbeđuje propulzivnu silu. Trodimenzionalni model ove letelice prikazan je na slici 6.114.



Slika 6.114 - 3D model letelice LABFLY-EM01

U letelicu se ugrađuje električni motor jednosmerne struje bez četkica *EMAX MT1806* sa elisom *Gemfan 5030* (prečnik elise je 5 inča) i *Emax 12A BLHELI* ESC-om. Modifikacija se fizički izvodi montiranjem stajnog trapa i zamenom prednjeg dela trupa koji je modularan. Potrebno je samo izvršiti zamenu modula i ponovo povezati žice.

Geometrija uzgonskih površina letelice, senzori, aktuatori (osim dodatog elektromotora) ostaju nepromenjeni u odnosu na *LABFLY-RM01*. Na slici 6.115 je prikazana letelica *LABFLY-EM01* u letu.



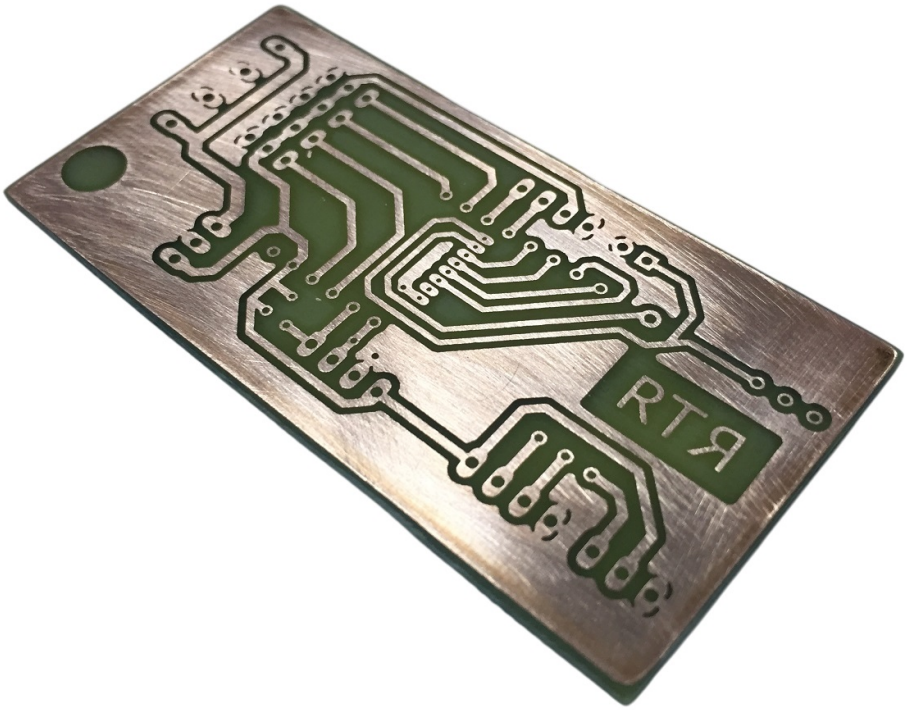
Slika 6.115 - Letna ispitivanja letelice LABFLY-EM01

Na slici 6.116 prikazan je konačan izgled letelice *LABFLY-EM01*. Ova letelica je upotrebljena za nastavak započetog sa letelicom *LABFLY-RM01*, a to je analiza bespilotnih letelica sa fiksnim krilima.



Slika 6.116 - Konačan izgled letelice LABFLY-EM01

Da bi se nastavio proces identifikacije sistema razvijene su posebne elektronike. Tokom ispitivanja bilo je potrebno napraviti nekoliko verzija različitih štampanih ploča za elektroniku. Na slici 6.117 je prikazan primer jednoslojne štampane ploče za elektroniku koja je izrađena foto postupkom. Ovaj postupak nekada može biti suviše kompleksan i spor pa je ovom prilikom korišćen i postupak izrade štampanih ploča glodanjem (kao što je prikazano na slici 6.118). Ovo je veoma jednostavan postupak kojim se pomoću glodala na opruzi na CNC glodalici glođe bakarni sloj. Ograničenje ovog načina izrade je širina bakarnih vodova koje je precizno moguće izraditi pa se ovaj postupak koristi praktično samo za velike komponente i čitave module koji se međusobno spajaju.



Slika 6.117 - Primer štampane ploče za elektroniku letelice izrađene foto postupkom



Slika 6.118 - Izrada štampane ploče za elektroniku glodanjem

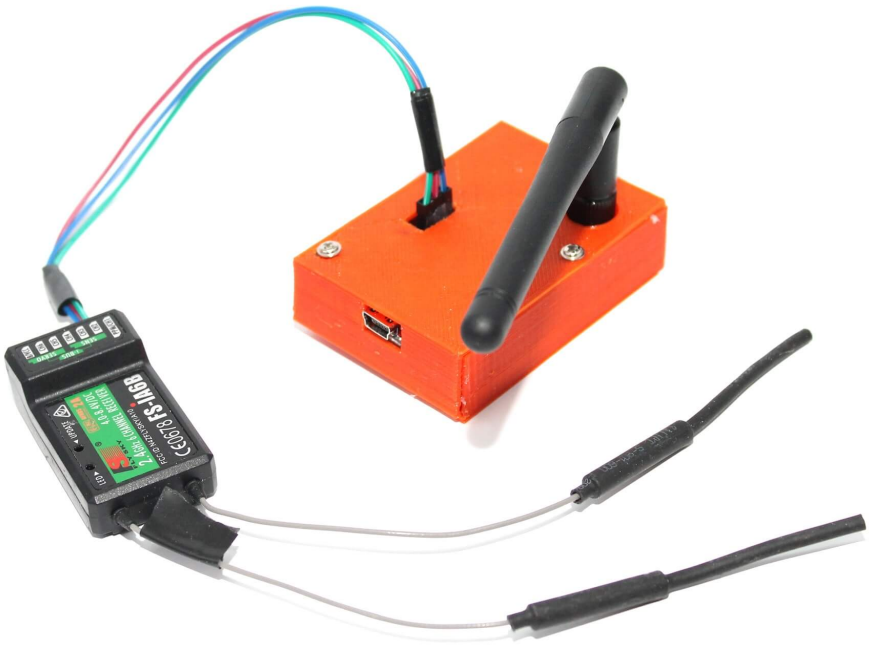
Elektronika za ovu letelicu koristi modul pod nazivom *GY91* na kome se, pored već prikazanog inercijalnog mernog sistema *MPU-9250*, nalaze i barometar i termometar u okviru senzora *BMP280*. Pomoću ovog senzora, a na osnovu promene statičkog pritiska se može izračunati visina leta kao i brzina penjanja ili poniranja.

Osim ovih senzora na letelici se nalazi i *GNSS* modul proizvođača *UBLOX* i to model *NEO-M8N* kojim se sa frekvencijom uzorkovanja od 10 Hz određuje položaj letelice i brzina kretanja letelice, modul se nalazi na gornjaci krila u osi simetrije letelice.

Da bi se upravljalo letelicom potrebno je uspostaviti vezu između pilota i letelice, kojom će pilot poslati letelici informaciju o tome u kom položaju želi da se kontrolne površine letelice nalaze. Kao i za letelicu *QFLY-250*, koristi se daljinski upravljač *FLYSKY FS-i6*.

Otklonom dve kontrolne palice pilot može da kontroliše četiri radio kanala. Obično se u letelici nalazi radio prijemnik i direktno se vrši upravljanje servo motorima. Za potrebe ovog sistema to nije pogodno, pošto je potrebno sačuvati promene otklona kontrolnih palica tokom ispitivanja zajedno sa svim ostalim podacima sa senzora letelice. Za potrebe dvosmerne komunikacije letelice sa stanicom na zemlji koristi se kao i u slučaju kvadkoptera radio modul *nRF24L01+*.

Na slici 6.119 prikazan je primopredajnik za slanje komandi za upravljanje letelicom i za dobijanje podataka sa senzora letelice. Ovaj modul sve podatke, uključujući i one od daljinskog upravljača u realnom vremenu prosleđuje na računar koji sve podatke zapisuje u dnevnik leta. Daljinski upravljač u ovom slučaju bežično šalje podatke samo do primopredajnika koji ih prosleđuje letelici i računaru.



Slika 6.119 - Predajnik za slanje komandi za upravljanje letelicom

Nakon letnih ispitivanja na glavnom računaru ostaju zapisani dnevnici leta koji se potom mogu analizirati u cilju identifikacije sistema.

Za potrebe prikazivanja elektronike letelice prilikom održavanja predavanja ili prikaza na sajmovima je napravljena i modifikacija prikazana na slici 6.120.



Slika 6.120 - Model letelice *LABFLY-EM01* za prikaz njene elektronike u trupu

6.2.5 Letelica FW-01

Nakon letelice *LABFLY-EM01* je projektovana i proizvedena potpuno nova letelica sa fiksnim krilom i sa većom masom korisnog tereta, ova letelica je dobila naziv *FW-01*. Letelica ima dva pogonska motora na polukrilima i konstrukcija joj je izrađena u potpunosti od kompozitnih materijala (epoksidna smola kao matrica sa staklenim ili karbonskim vlaknima). Na slici 6.121 prikazan je trodimenzionalni model ove letelice, dok je na slici 6.122 prikazan i njen konačan izgled.



Slika 6.121 - Trodimenzionalni model letelice FW-01



Slika 6.122 - Konačan izgled letelice FW-01

Letelica ima dva pogonska BLDC motora proizvođača *EMAX* i tipa `MT 1806 2280KV` na polukrilima koji se kontrolišu pomoću dva ESC-a proizvođača *EMAX* serije *SimonK* i tipa `Simon-12A`. Komandne površine se upravljaju sa ukupno četiri servo motora proizvođača *EMAX* i tipa `ES09MD`.

Svrha letelice je bila nastavak letnih ispitivanja započetih sa letelicom *LABFLY-EM01*, usavršavanje električnog sistema letelice, istraživanje različitih senzora, radio modula i modulacija. Elektronika letelice je projektovana tako da je omogućavala brzu zamenu radio modula.

Ideja je bila da baš letelica *FW-01* bude glavna letelica sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi. Na slici 6.123 se vidi realističan prikaz kompletnog sistema sa svim neophodnim delovima kao što su simulator leta i letelica.



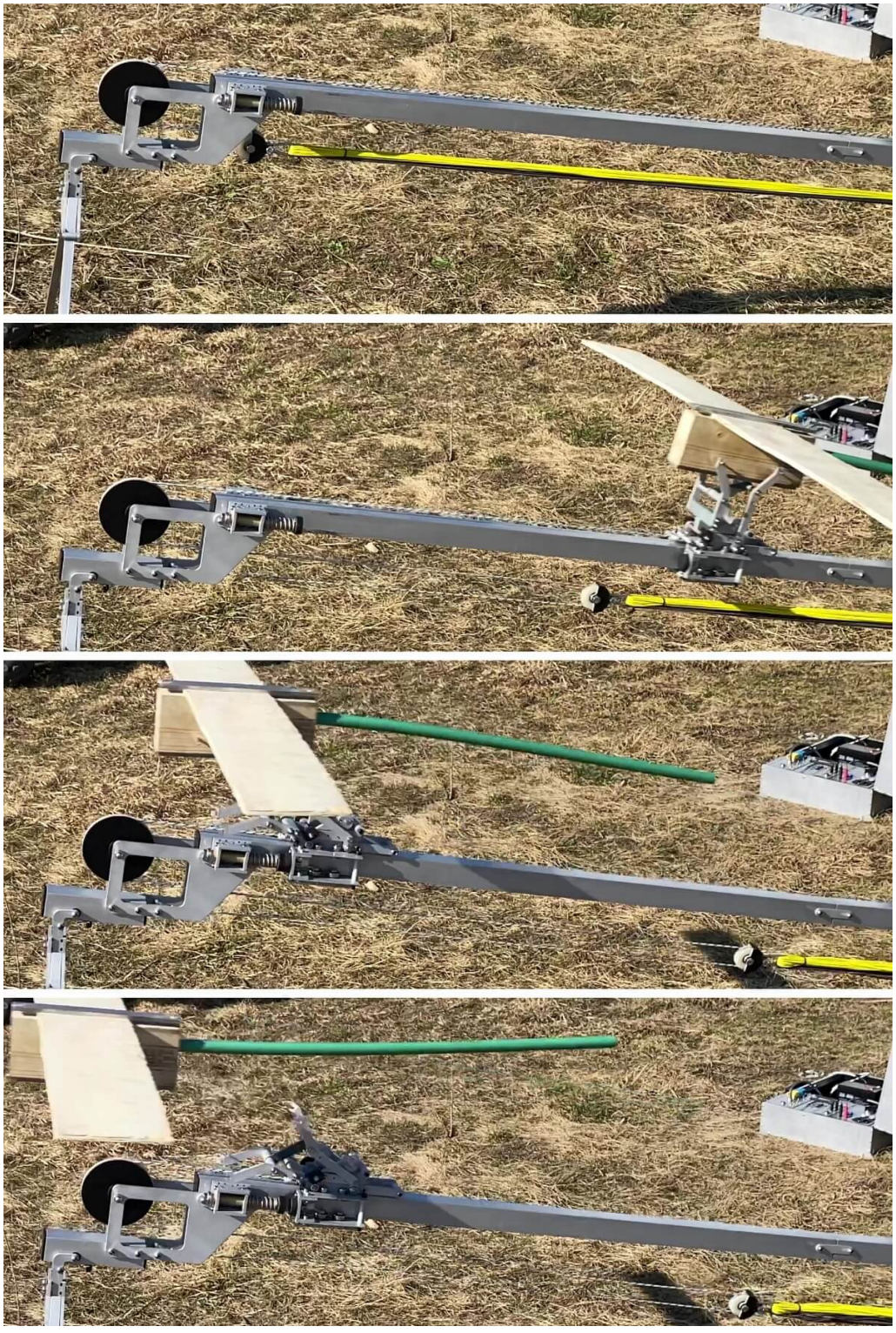
Slika 6.123 - Realističan prikaz kompletnog sistema sa letelicom FW-01

U cilju eliminacije potrebe za uređenom pistom za horizontalno poletanje se koristi katapult (prikazan na slici 6.124) kojim se letelica lansira tako da na izlazu sa katapulta već ima dovoljnu brzinu za uspostavljanje stabilnog leta.



Slika 6.124 - Katapult za letelicu

Konstrukcija katapulta je rasklopiva, što omogućava jednostavan transport. Pored toga, katapult je zamišljen tako da ima sve potrebne električne aktuatore da bi bilo moguće postaviti letelicu na katapult i zatim se popeti na simulator leta sa kojeg se potom šalje bežičnom komunikacijom komanda za lansiranje. Sila neophodna za lansiranje se dobija korišćenjem elastičnog gumenog užeta koje se zateže i zatim otpušta prilikom lansiranja. Podsistem se napaja preko punjivih baterija. Ovim katapultom je omogućeno poletanje za veoma kratko vreme i sa veoma malog i neuređenog prostora. Na slici 6.125 prikazano je ispitivanje kretanja kolica katapulta.



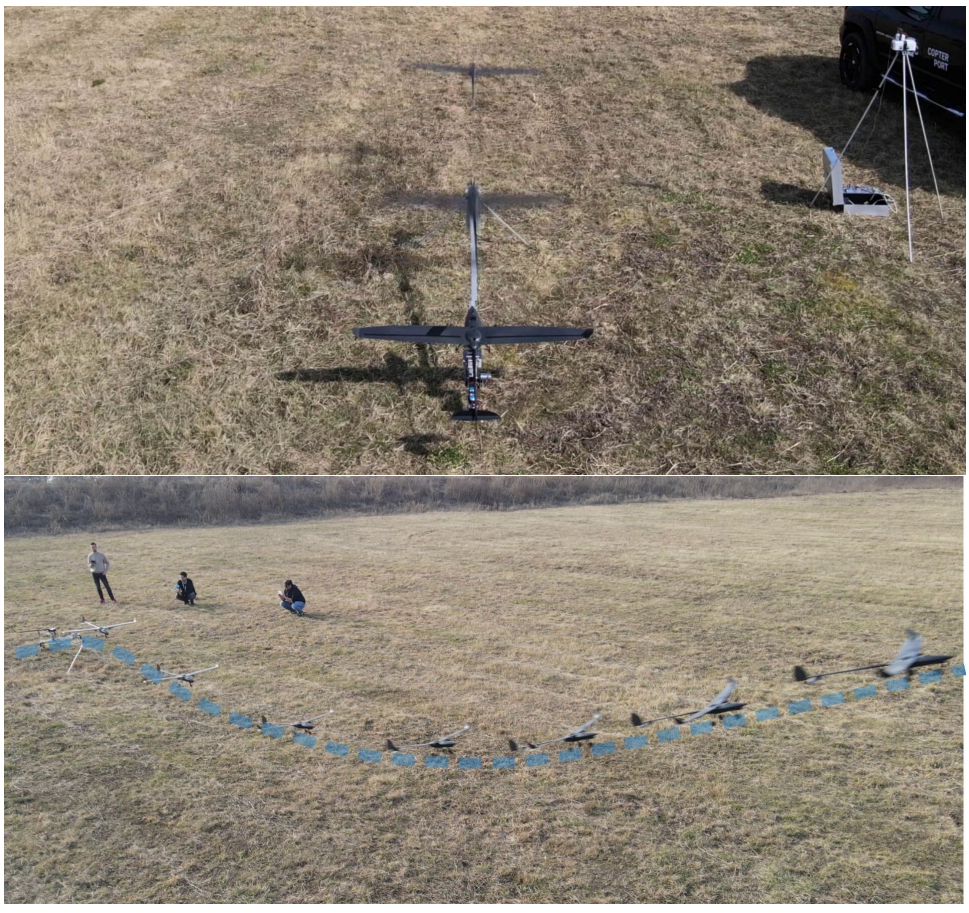
Slika 6.125 - Ispitivanje kretanja kolica katapulta

Kolica katapulta se kreću translatorno duž šine i osnovni zadatak im je da odgovarajuće drže letelicu tokom njenog ubrzanja i zatim na kraju šine katapulta da je otpuste i to bez kočenja letelice. Katapult drži letelicu preko dve osovine i ukupno četiri tačke. Kada kolica katapulta udare u opruge na kraju šine dolazi do obrtnog kretanja koje je posledica inercijalnih sila. Prilikom ove rotacije dela kolica dolazi do odbravljivanja letelice i njenog napuštanja katapulta.

Sa ovim sistemom je urađeno mnogo uspešnih letnih ispitivanja. Na slikama 6.126 i 6.127 je prvo prikazana letelica na katapultu, a zatim i njeno katapultiranje.



Slika 6.126 - Letelica FW-01 na katapultu



Slika 6.127 - Katapultiranje letelice FW-01

Na slikama 6.128 i 6.129 se vidi pogled sa kamere na letelici za let iz prvog lica, tokom katapultiranja i tokom leta. Vidi se očigledna razlika u kvalitetu slike što je posledica unapređivanja video-sistema letelice tokom vremena.



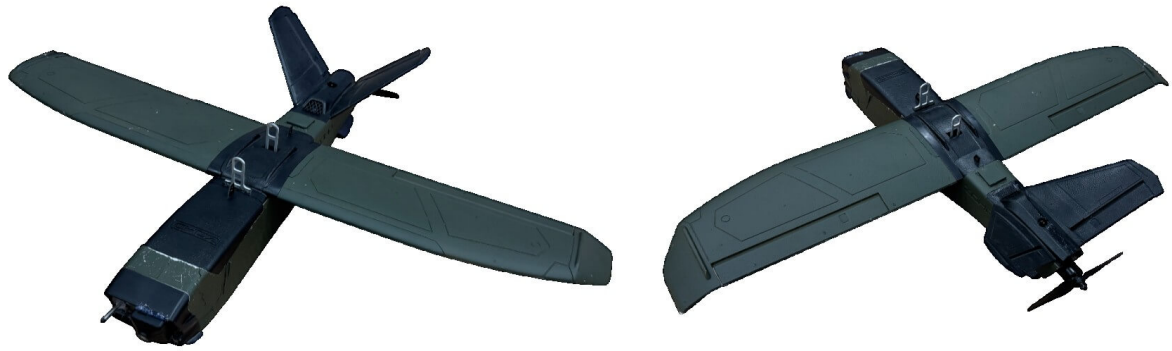
Slika 6.128 - Pogled sa kamere na letelici FW-01



Slika 6.129 - Pogled sa kamere na letelici FW-01 sa unapređenim video-sistemom

6.2.6 Letelica REMILP-01

Na kraju se pokazalo da je za sistem, koji je predmet istraživanja, moguće koristiti komercijalno dostupne konstrukcije, kao što je `Talon GT "Rebel"` proizvođača *ZOHD*. Ovakva letelica ima dovoljno prostora i dovoljnu nosivost za sve što je potrebno za uspešnu realizaciju sistema. Modifikovana letelica je dobila naziv *REMILP-01* i prikazana je na slici 6.130. Ova letelica se vrlo lako može lansirati iz ruke.



Slika 6.130 - Izgled letelice REMILP-01

Konstrukcija letelice je izrađena od biorazgradivog ekspandiranog polipropilena. Krilo i trup su ojačani šipkama od kompozitnog materijala sa karbonskim vlaknima kako bi se obezbedila potrebna čvrstoća. Razmah krila je `1000` mm, dok je dužina trupa 660 mm. Letelica koristi jedan pogonski motor potisnog tipa. Za kontrolne površine se koristi ukupno četiri servo motora. Jednostavna demontaža krila i repnih površina (omogućena integrisanim konektorima za žice servo motora) omogućava daleko jednostavniji transport letelice. U tabeli 6.29 data je specifikacija delova letelice *REMILP-01*.

Tabela 6.29 - Specifikacija delova letelice REMILP-01

Komponenta	**Proizvođač i model**
Elektromotor	ZOHD 2216-1300KV
Elisa	ZOHD 8x5
ESC	ZOHD 40A sa 5V 3A BEC
Servo motor	ZOHD 9g metalni 4 kom
Radio modul	Digitalni 2.4 GHz
GNSS modul	U-blox NEO-M8N
Senzor brzine vazduha	Matek ASPD-4525
Autopilot	Holybro Pixhawk 4

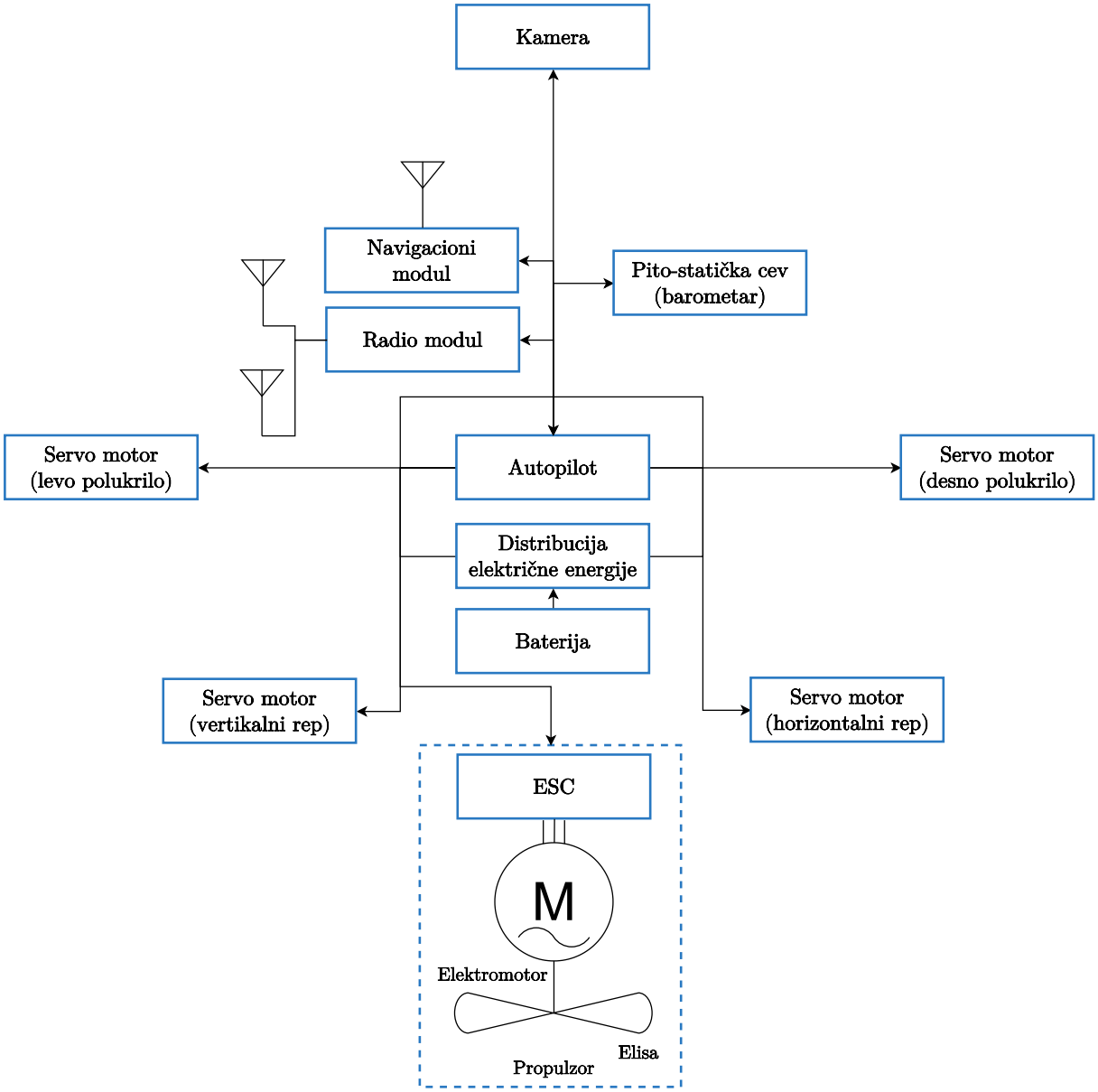
Nakon svih urađenih analiza, korišćenjem prethodno prikazanih letelica, odlučeno je da se za sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi koriste radio moduli sa implementiranim `IP`. Na radio modul se direktno preko `LAN`-a povezuje kamera. Kamera je opremljena odgovarajućom elektronikom i implementira `RTSP` server za prenos video signala preko mreže.

Za autopilot je izabrana elektronika `Pixhawk 4` kompanije *Holybro* dok je za program izabran `ArduPilot` projekat, koji je detaljno opisan u poglavlju 6.2.7. Praktično je moguće koristiti bilo koji hardver koji je podržan od strane `ArduPilot` projekta, pored ovde izabranog, često se koristi i `Cube Orange` kompanije *HEX*. Autopiloti `Cube` su interesantni zbog modularnog dizajna. Proizvođač obezbeđuje tehničku dokumentaciju za konektor

modula koji se potom povezuje sa nosećom pločom. Projektanti bespilotnih letelica mogu koristiti gotov modul autopilota sa posebno razvijenom nosećom pločom koja nije univerzalna već specifično odgovara ostatku sistema za koji se koristi. Zajedničko za trenutne autopilote koji su u upotrebi jeste da se koriste nekoliko puta redundantni IMU senzori, barometri i magnetometri.

Osim integrisanih senzora u autopilotu, za *ArduPilot* letelice sa fiksnim krilima je poželjno koristiti i `GNSS` navigacioni modul i pitot-statičku cev sa barometrom za merenje brzine vazduha.

Na slici 6.131 je prikazan konačan električni sistem letelice *REMILP-01*.



Slika 6.131 - Električni sistem letelice REMILP-01

ArduPilot projekat omogućava jednostavnu implementaciju novih funkcionalnosti za letelicu ali je za integraciju sa ostatkom sistema zapravo najbitnije što je moguće koristiti `MAVLink` protokol za razmenu poruka između letelice i ostatka sistema. Putem `MAVLink` protokola je moguće u potpunosti upravljati i nadgledati letelicu sa *ArduPilot* programom. Danas većina zemaljskih stanica i daljinskih upravljača koristi `MAVLink` protokol.

Digitalni radio modul sa `LAN` kamerom i *ArduPilot* sa `MAVLink` protokolom omogućavaju potpunu realizaciju sistema koji je ovde predmet istraživanja.

Na slici 6.132 je prikazan let letelice *REMILP-01*.



Slika 6.132 - Let letelice REMILP-01

Električni sistem je napravljen tako da se može lako integrisati u različite konstrukcije letelica. Na slici 6.133 su pored prototipa simulatora leta prikazane i neke konstrukcije letelica (sa različitim aerodinamičkim rešenjima) koje se mogu koristiti. Mogućnost korišćenja različitih konstruktivnih tipova letelica je značajna karakteristika samog sistema.



Slika 6.133 - Razne letelice zajedno sa prototipom simulatora leta

6.2.7 Autopilot program ArduPilot

ArduPilot je najnapredniji, najpotpuniji i najpouzdaniji softver za autopilot otvorenog koda koji je javno dostupan. Od 2010. godine ga razvija raznovrstan tim profesionalnih inženjera, naučnika i saradnika zajednice[151]. Glavni deo koda je napisan u programskom jeziku `C++`. Prefiks *Ardu* je zapravo od *Arduino*, jer se on kao razvojno okruženje koristio na početku razvoja projekta od 2007. godine (kao što je prikazano u [152]). Projekat se bavi zadacima kao što su očitavanje senzora, stabilizacija letelice, izvršavanje petlji upravljanja, komunikacija sa zemaljskim stanicama i daljinskim upravljačima i izvršavanje komandi misije.

Pored *ArduPilot* projekta, popularni su još i drugi autopiloti otvorenog koda kao što su *PX4* i *INAV* projekti. Sa druge strane, postoje i specijalizovane kompanije za razvoj autopilota za bespilotne letelice kao što su *MicroPilot*, *UAV Navigation*, *Embention*, *UAVOS Inc* i druge.

Sam projekat *ArduPilot* je podeljen na osnovu tipa vozila na:

- *ArduCopter* – za multikoptere i helikoptere.
- *ArduPlane* – za avione (letelice sa fiksnim krilima).
- *ArduSub* – za plovna i podvodna vozila (podmornice).
- *Blimp* – za letelice lakše od vazduha (balone i cepeline).
- *Rover* – za kopnena vozila (vozila na točkovima ili gusenicama).
- *AntennaTracker* – za kontinuirano usmeravanje antene ka vozilu.

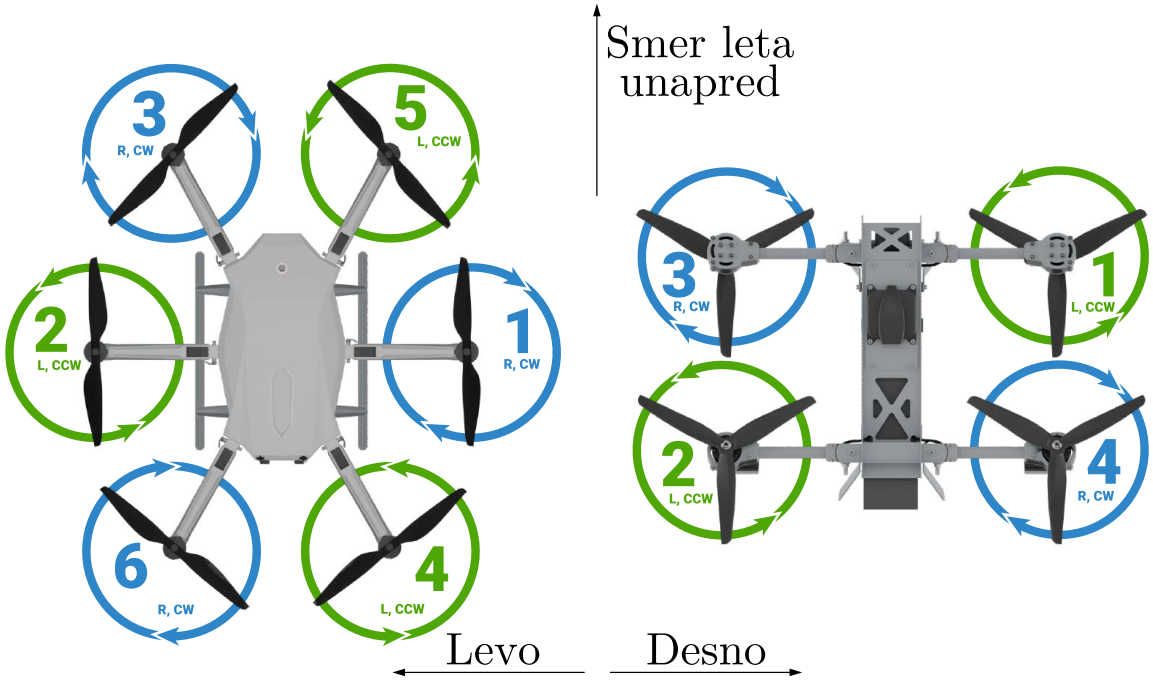
Sva vozila imaju svoj folder u korenu samog projekta. Pored vozila ovde se nalaze i folderi kao što su `Tools` (sa alatima među kojima su najznačajniji `AP_Bootloader` i `AP_Periph`), `libraries` (sa zajedničkim bibliotekama kao što je `AP_HAL`) i `modules` (sa eksternim modulima kao što su operativni sistem `ChibiOS`, komunikacioni protokol `MAVLink` ili sistem za kompajliranje `waf`).

Neke od najznačajnijih biblioteka su:

- `AP_Common` – Osnovna biblioteka i zaglavlja koja su potrebna svim ostalim delovima koda i bibliotekama.
- `AP_Math` – Razne matematičke funkcije, naročito korisne za vektorsku manipulaciju.
- `AP_HAL` – Sloj za apstrakciju hardvera (eng. *Hardware Abstraction Layer*). Omogućava da se isti program koristi na različitim tipovima hardvera, bez potrebe za prevelikim izmenama koda.
- `AC_PID` – `PID` kontroler biblioteka. Koristi se za regulaciju i održavanje stabilnosti različitih sistema (npr. kontrola visine, položaja, brzine).
- `AP_AHRS` – Zadužen za procenu položaja letelice (eng. *Attitude and Heading Reference System*) korišćenjem `DCM` (eng. *Direction Cosine Matrix*) ili `EKF`.
- `AP_NavEKF3` – Druga generacija proširenog Kalmanovog filtera (`EKF`, eng. *Extended Kalman Filter*). Kombinuje podatke iz `IMU` senzora, `GNSS` i kompasa radi procene položaja, brzine i orijentacije uz visoku preciznost.
- `AP_InertialNav` – Inercijalna navigaciona biblioteka, kombinuje podatke akcelerometra (`IMU`) sa `GNSS` i barometrom radi preciznijeg izračunavanja pozicije i brzine.

- `AC_AttitudeControl` – Biblioteka za kontrolu orijentacije i pozicije, zasnovane na PID regulaciji.
- `AC_WPNav` – Biblioteka za navigaciju po definisanim tačkama (eng. *waypoints*). Omogućava automatski let na osnovu unapred postavljenih koordinata.
- `AP_InertialSensor` – Očitava podatke sa žiroskopa i akcelerometra, vrši njihovu kalibraciju i obezbeđuje podatke u standardnim jedinicama za glavni kod i druge biblioteke.
- `AP_Baro` – Interfejs za barometre, omogućava precizno očitavanje visine na osnovu pritiska.
- `AP_GPS` – Biblioteka za rad sa `GNSS` modulima, omogućava očitavanje globalnog položaja i brzine.
- `AP_Compass` – Biblioteka za troosne magnetometre (kompase), omogućava čitanje orijentacije prema magnetnom polju Zemlje.
- `AP_OpticalFlow` – Interfejs za senzore koji se koriste za merenje kretanja u odnosu na površinu ispod letelice.

Za podešavanje programa u cilju prilagođavanja određenoj letelici se koriste parametri. Jedni od stotine parametara su i parametri `FRAME_CLASS` i `FRAME_TYPE` koji treba da budu podešeni tako da odgovaraju konstruktivnom tipu letelice koji se koristi. Neophodno je voditi računa o položaju rotora i o njihovom smeru obrtanja[153]. Na sledećoj slici (slika 6.134) su prikazani primeri dva konstruktivna tipa od mnoštva dostupnih. Pored već dostupnih tipova omogućeno je i jednostavno definisanje novih. Pravilan smer obrtanja je ključan za uspostavljanje leta pa se zbog toga najčešće smer obrtanja motora proverava pre svakog poletanja. Motorima se pošalje komanda za minimalnu brzinu obrtanja i zatim se proveri da li su smerovi odgovarajući i da li su montirane odgovarajuće elise za taj smer obrtanja.



Slika 6.134 - Primer konstruktivnih tipova multikoptera, HEXA X (levo) i QUAD X (desno)

6.2.7.1 Kompajliranje

U dokumentaciji projekta je dostupno i zvanično uputstvo za kompajliranje *ArduPilot*-a na operativnim sistemima kao što su `Windows` ili `Ubuntu`. Za kompajliranje na operativnom sistemu `Windows` se zbog jednostavnosti preporučuje korišćenje projekta *Cygwin*[154].

Nakon preuzimanja i instaliranja programa *Cygwin*, potrebno je podesiti ovo okruženje kao što je prikazano u listingu 6.8.

```
setup-x86_64.exe --root=C:\cygwin64 --no-startmenu --site=http://cygwin.mirror.
    constant.com --packages autoconf,automake,ccache,cygwin32-gcc-g++,gcc-g
    ++=7.4.0-1,libgcc1=7.4.0.1,gcc-core=7.4.0-1,git,libtool,make,gawk,libexpat-
    devel,libxml2-devel,python37,python37-future,python37-lxml,python37-pip,
    libxslt-devel,python37-devel,procps-ng,zip,gdb,ddd,xterm
ln -sf /usr/bin/python3.7 /usr/bin/python3
ln -sf /usr/bin/pip3.7 /usr/bin/pip
python3.7 -m pip install empy==3.3.4 pyserial pymavlink intelhex dronecan
    pexpect
```

Listing 6.8 - Podešavanje Cygwin okruženja

Zatim se na osnovu komandi iz listinga 6.9 preuzima *ArduPilot* kod.

```
git clone --recurse-submodules https://github.com/ArduPilot/ardupilot
cd ardupilot
git fetch origin Copter-4.5
git switch Copter-4.5
```

Listing 6.9 - Preuzimanje ArduPilot koda

Na kraju se `ArduCopter` kod kompajlira za ploču `Pixhawk4` komandama iz listinga 6.10.

```
./waf list_boards
./waf configure --board Pixhawk4
./waf copter
```

Listing 6.10 - Kompajliranje ArduCopter koda za ploču Pixhawk4

Da bi se *ArduPilot* izmenio ili proširio, svakako je dobro prvo savladati kompajliranje. Umesto kompajliranja programa za vozila (kao što je npr. `ArduCopter`) moguće je kompajliranje novog jednostavnog programa kao što je `Blink` (za treperenje LED na ploči) za koji je potrebno napraviti folder u nekoj od biblioteka (npr. u `libraries/AP_HAL/examples`) i zatim napraviti fajl `Blink.cpp` sa sadržajem iz listinga 6.11 i fajl `wscript` sa sadržajem iz listinga 6.12.

```
#include <AP_HAL/AP_HAL.h>

void setup();
void loop();
const AP_HAL::HAL& hal = AP_HAL::get_HAL();

void setup(void) {
  hal.gpio->pinMode(LED_BUILTIN_PIN, HAL_GPIO_OUTPUT);
}
void loop(void) {
  hal.gpio->write(LED_BUILTIN_PIN, 1);
  hal.scheduler->delay(1000);
  hal.gpio->write(LED_BUILTIN_PIN, 0);
  hal.scheduler->delay(1000);
}
AP_HAL_MAIN();
```

Listing 6.11 - Blink.cpp

```
#!/usr/bin/env python
# encoding: utf-8

def build(bld):
    bld.ap_example(
        use='ap',
    )
```

Listing 6.12 - wscript za Blink.cpp primer

Nakon toga je moguće kompajlirati ovaj primer korišćenjem komandi iz listinga 6.13.

```
./waf clean
Tools/scripts/build_bootloaders.py Pixhawk4 # build bootloader
./waf configure --board Pixhawk4
./waf build --target examples/Blink --upload
```

Listing 6.13 - Komande za kompajliranje Blink.cpp primera

Ili isto prethodno samo u jednoj liniji kao u listingu 6.14.

```
./waf clean && Tools/scripts/build_bootloaders.py Pixhawk4 && ./waf configure --
    board=Pixhawk4 && ./waf build --target examples/Blink --upload
```

Listing 6.14 - Jedna komanda za kompajliranje Blink.cpp primera

Na ovaj način omogućeno je programiranje korišćenjem lanca alata projekta *ArduPilot*.

6.2.7.2 Nova ploča

Osim toga što se u *ArduPilot*-u mogu koristiti raznovrsne ploče (*hardveri*), vrlo je jednostavno dodavati nove. Postupak za dodavanje nove ploče je sledeći:

- Prvo je u okviru foldera `libraries/AP_HAL_ChibiOS/hwdef` potrebno napraviti novi folder za novu ploču. Naziv ovog foldera (npr. `BOARD_XXX`) predstavlja naziv same ploče koji se koristi dalje za izbor ove ploče tokom programiranja.
- U okviru ovog foldera potrebno je napraviti fajlove `hwdef.dat` i `hwdef-bl.dat` kojima se definiše hardver ploče za glavni program i za `bootloader`. Preporučuje se da se ovi fajlovi definišu po ugledu na postojeće sa istim procesorima (sve ih je moguće naći u `hwdef` folderu).
- Sada je moguće izvršiti kompajliranje `bootloader` programa za ovu ploču pomoću sledećih komandi:

  ```
  ./waf configure --board BOARD_XXX --bootloader # configure for bootloader
  ./waf bootloader # compile bootloader
  ```

- Zatim je moguće ovaj program učitati u memoriju procesora. Za učitavanje programa u memoriju mikrokontrolera familije `STM32` neophodno je instalirati program `STM32CubeProgrammer`. Nakon pokretanja ovog programa, sa leve strane iz menija izabrati `Memory & File edition` zatim izabrati tab `[+]` i iz padajućeg menija izabrati `Open file` nakon čega je potrebno izabrati program sa diska (sa ekstenzijom `.hex` ili `.bin`). Kada je učitan program potrebno ga je otpremiti na mikroprocesor pritiskom na dugme `[Download]`. Da bi se program učitao u memoriju računar mora da bude povezan sa njim pomoću `ST-LINK` programatora.
- Nakon toga je moguće kompajlirati i *ArduPilot* i izvršiti njegovo učitavanje u memoriju procesora kada je ploča povezana sa računarom preko `USB`-a.

```
$ ./waf configure --board BOARD_XXX # configure board
$ ./waf copter --upload # build and upload firmware
```

- U fajlove `hwdef.dat` i `hwdef-bl.dat` dodati sve pinove koji se koriste za periferije, zatim izabrati senzore i definisati dodatnu eksternu memoriju ukoliko postoji.

Na ovaj način omogućeno je korišćenje potpuno novih ploča u okviru projekta *ArduPilot*.

6.2.7.3 Proširenje programa

Tokom razvoja novih bespilotnih letelica može doći do potrebe za implementacijom neke nove funkcionalnost ili potrebe za izmenom postojeće. Izmena postojeće funkcionalnosti zahteva izmenu koda i zatim kompajliranje i učitavanje programa u memoriju procesora. Što se tiče proširenja programa, u programu `ArduCopter` je već predviđeno na koji način se dodaju specijalne funkcionalnosti od strane korisnika. Predviđena procedura za proširenje programa od strane korisnika je sledeća:

- U fajlu `APM_Config.h` se bira šta se koristi.
- Već dostupni šabloni funkcija (bez bloka komandi, koje korisnik sam definiše) za proširivanje funkcionalnosti samog programa su dostupne u okviru fajla `UserCode.cpp`.
- Za izvršavanje neke funkcionalnosti na početku programa je dostupna funkcija `userhook_init()` koju je moguće aktivirati otkomentarisanjem sledeće linije u fajlu `APM_Config.h`.

```
#define USERHOOK_INIT userhook_init();
```

- Da bi se koristile funkcije za pomoćne prekidače na radio stanici
  - `userhook_auxSwitch1()`,
  - `userhook_auxSwitch2()` i
  - `userhook_auxSwitch3()`

  neophodno je otkomentarisati sledeću liniju.

```
#define USERHOOK_AUXSWITCH ENABLED
```

  Zatim za određeni `RC` kanal promeniti parametar `RCx_OPTION` tako da uzima vrednost od `47` do `49`.

- U zavisnosti od potrebne učestanosti pozivanja dostupne su funkcije:
  - `userhook_FastLoop()` (100 Hz),
  - `userhook_50Hz()` (50 Hz),
  - `userhook_MediumLoop()` (10 Hz),
  - `userhook_SlowLoop()` (3.3 Hz),
  - `userhook_SuperSlowLoop()` (1 Hz),

  u skladu sa tim koje se funkcije koriste otkomentarisati sledeće linije.

```
#define USERHOOK_FASTLOOP userhook_FastLoop();
#define USERHOOK_50HZLOOP userhook_50Hz();
#define USERHOOK_MEDIUMLOOP userhook_MediumLoop();
#define USERHOOK_SLOWLOOP userhook_SlowLoop();
#define USERHOOK_SUPERSLOWLOOP userhook_SuperSlowLoop();
```

- Nakon izmene programa, izvršiti kompajliranje i otpremu preko USB-a, korišćenjem sledećih komandi.

```
./waf clean
./waf configure --board CubeOrange
./waf copter --upload
```

6.2.7.4 SITL

SITL (softver u petlji, eng. *Software in the Loop*) simulator omogućava izvršavanje *ArduPilot* programa bez hardvera autopilota. Kod autopilota se kompajlira korišćenjem običnog C++ kompajlera, koji daje izvorni izvršni fajl koji omogućava testiranje ponašanja programa bez dodatnog hardvera. Kada se program testira na hardveru autopilota to postaje HITL (hardver u petlji, eng. *Hardware in the Loop*).

Sledećim komandama se preuzima kod određene verzije kao što je na primer Copter-4.5.

```
git clone --recurse-submodules https://github.com/ArduPilot/ardupilot
cd ardupilot
git fetch origin Copter-4.5
git switch Copter-4.5
```

Ovaj kod se kompajlira kao SITL za operativni sistem *Windows* (korišćenjem projekta *Cygwin*) pomoću sledećih komandi.

```
./waf configure --board=sitl --toolchain x86_64-pc-cygwin
./waf copter
```

Nakon kompajliranja izvorni izvršni fajl se nalazi u folderu:

```
.\build\sitl\bin
```

Da bi se pokrenuo ovaj fajl potrebno je da procesu budu poznate putanje do pojedinih *Cygwin* biblioteka.

SITL je praktično nezaobilazan resurs tokom razvoja programa autopilota i aplikacija za daljinsko upravljanje jer omogućava testiranja bez potrebe za hardverom autopilota ili kompletnih letelica. Ovaj program omogućava jednostavno testiranje izmena i proširenja programa autopilota kao što je, na primer, prikazano u [155].

6.2.7.5 AP_Periph

Za programiranje dodatnih CAN uređaja (periferija) predviđen je potprogram AP_Periph. Ovaj potprogram se najčešće koristi za dodatne uređaje koji se koriste u električnom sistemu letelice a ne nalaze se u samoj blizini autopilota pa se sa njima komunicira putem CAN-a. U nastavku je prikazana procedura razvoja novog AP_Periph uređaja:

- Prvo je potrebno preuzeti program i to određenu stabilnu verziju.

```
rm -r -f ardupilot_AP_Periph1-7 # delete previous data
git clone https://github.com/ArduPilot/ardupilot.git ardupilot_AP_Periph1-7
cd ardupilot_AP_Periph1-7
```

```
4 git branch -r # see all available branches, press q to exit
5 git fetch origin AP_Periph-1.7
6 git switch AP_Periph-1.7
```

- Zatim je potrebno definisati novu ploču za periferiju slično kao za autopilot, samo što se u ovom slučaju za ugled prilikom formiranja `hwdef.dat` i `hwdef-bl.dat` fajlova koriste već implementirane periferije `f103-periph` i `f303-periph` koje se takođe nalaze u folderu `libraries/AP_HAL_ChibiOS/hwdef` zajedno sa ostalim pločama.

- Kompajliranje programa za ploču `BOARD_XXX` i učitavanje programa na mikrokontroler se vrši sa:

```
1 ./waf clean
2 Tools/scripts/build_bootloaders.py BOARD_XXX # build bootloader
3 ./waf configure --board BOARD_XXX
4 ./waf AP_Periph
```

  Ili isto prethodno samo u jednoj liniji:

```
1 $ ./waf clean && Tools/scripts/build_bootloaders.py BOARD_XXX && ./waf
    configure --board=BOARD_XXX && ./waf AP_Periph
```

- Za izmenu samog programa je bitno da je glavni deo programa definisan u fajlovima:

  `Tools/AP_Periph/AP_Periph.h`

  `Tools/AP_Periph/AP_Periph.cpp`

- Na početku izvršavanje programa se poziva funkcija `AP_Periph_FW::init()` a zatim se ulazi u beskonačnu petlju u kojoj se poziva `AP_Periph_FW::update()` (unutar koje se poziva `can_update()` funkcija).

- U fajlu `can.cpp` je definisana `CAN` komunikacija.

- Kada paket stigne preko `CAN` komunikacije prvo se poziva funkcija `shouldAcceptTransfer()` na osnovu koje se odlučuje da li se paket čita ili odbacuje.

- Ukoliko se paket prihvata poziva se funkcija `onTransferReceived()`.

- Paketi se šalju pomoću funkcije `canard_broadcast()`.

- Za `UART debug` je potrebno u okviru `hwdef.dat` fajla definisati sledeće linije.

```
1 # order of UARTs
2 SERIAL_ORDER EMPTY EMPTY USART2 USART1
3
4 # USART2 for debug (disabled)
5 PA2 USART2_TX USART2 SPEED_HIGH NODMA
6 PA3 USART2_RX USART2 SPEED_HIGH NODMA
```

  Tada je moguće koristiti `UART` za slanje poruka, kodom u sledećem obliku.

```
1 AP_HAL::UARTDriver *debug_uart;
2
3 void setup(void)
4 {
5     ...
6     debug_uart = hal.serial(2);
```

```
7       debug_uart->begin(115200, 256, 256); // debug (baud, rxSpace, txSpace)
8       ...
9   }
10
11  void loop(void)
12  {
13      ...
14      debug_uart->printf("Custom debug text\n");
15      ...
16  }
```

- Kompajlirane definicije za `CAN` poruke nalaze se na sledećoj adresi.

  `.\build\BOARD_XXX\modules\libcanard\dsdlc_generated\uavcan`

- Za `PWM` izlaze je potrebno u okviru `hwdef.dat` fajla definisati pinove slično sledećem.

```
1  PA0 TIM2_CH1 TIM2 PWM(1) GPIO(101)
```

  Tada je moguće koristiti `PWM` pinove kodom u sledećem obliku.

```
1   void setup(void)
2   {
3       ...
4       hal.rcout->init();
5       hal.rcout->enable_ch(channel); // channel index
6       ...
7   }
8
9   void loop(void)
10  {
11      ...
12      hal.rcout->write(channel, value); // value is pulse width in us
13      ...
14  }
```

Prikazani postupak je veoma značajan jer omogućava razvoj programa za nove električne komponente sistema sa mogućnošću korišćenja svih već implementiranih funkcionalnosti u projektu *ArduPilot* (kao što je na primer slanje i primanje `CAN` poruka).

6.2.7.6 Parametri

ArduPilot parametri predstavljaju skup konfiguracionih vrednosti koji omogućavaju detaljno prilagođavanje ponašanja letelice ili vozila. Program `ArduCopter` ima više od 1000 parametara. Za pravilno konfigurisanje letelice je potrebno podesiti i nekoliko desetina parametara. Ovi parametri pokrivaju ključne oblasti kao što su `PID` regulacija, kalibracija senzora i drugo. Parametri se mogu pregledati i menjati i daljinski pomoću `MAVLink` komunikacionog protokola (iz aplikacija na daljinskom upravljaču ili zemaljskoj stanici), čime se omogućava lakše praćenje i precizno podešavanje svakog pojedinačnog parametra u skladu sa potrebama korisnika.

Kao primer, eksperimentalnim ispitivanjima je moguće tačno odrediti vrednosti za parametre `RCx_MIN`, `RCx_TRIM`, `RCx_MAX`, `RCx_DZ`, `SERVOx_MIN`, `SERVOx_TRIM` i `SERVOx_MAX`. Da bi se koristili ovi parametri za vrednost parametra `SERVOn_FUNCTION` treba izabrati `RCINnScaled`.

6.2.7.7 Čitanje dnevnika leta

Dnevnik leta se u projektu *ArduPilot* naziva `DataFlash log` i predstavlja detaljan zapis rada i performansi letelice tokom leta. U ovim dnevnicima beleži se širok spektar podataka, uključujući `GNSS` koordinate, očitavanja sa senzora, režimi leta i drugi ključni parametri neophodni za dijagnostiku i analizu. Zahvaljujući tim podacima, korisnici mogu da prate rad sistema, brzo otkriju eventualne probleme i optimizuju performanse i stabilnost letelice.

Ovo su binarni zapisi koji omogućavaju efikasno skladištenje i obradu velikog broja podataka. Zapisi se sastoje iz pojedinačnih poruka u binarnom formatu od kojih svaka ima svoj `ID`, dužinu i podatke. Za čitanje dnevnika leta nije potrebno ništa osim samog dnevnika jer je format svake poruke definisan porukom `FMT` koja definiše format drugih poruka. Prilikom čitanja dnevnika leta se prvo pronalaze `FMT` poruke i zatim se na osnovu njih čitaju sve ostale poruke.

U projektu *ArduPilot* se `DataFlash log`-ovi programskim jezikom `JavaScript` čitaju pomoću modula `JsDataflashParser`[156]. Primer čitanja dnevnika leta u programu `JSLAB` je dat u listingu 6.15.

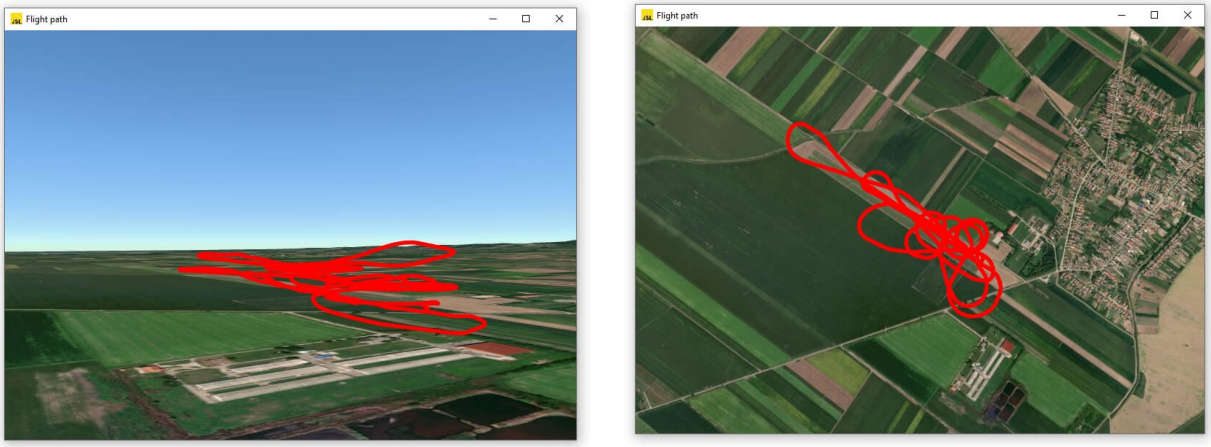
```
/**
 * @file Read ArduPilot Log
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');

var log_path = pwd + '2024-07-31 18-26-19.bin';

// Read log
var { DataflashParser } = require('./ap_log_parser.js');
var data_bin = readFile(log_path);
var parser = new DataflashParser(false);
parser.processData(data_bin.buffer, ['ATT', 'POS', 'RATE', 'RCOU', 'XKF1']);
disp(' --------------------');
```

Listing 6.15 - `readArdupilotLog.jsl`

U listingu 6.16 je data `JSLAB` skripta koja čita dnevnik leta i zatim prikazuje putanju leta na osnovu podataka koji su isčitani kao što je prikazano na slici 6.135.



Slika 6.135 - Prikaz putanje leta na osnovu podataka iz dnevnika leta

```
/**
 * @file Show flight path from ArduPilot Log
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
var token = '';

// Inputs
var log_path = pwd + '2024-07-31 18-26-19.bin';
var log_msgs = ['POS'];

// Read log
var { DataflashParser } = require('./ap_log_parser.js');
var data_bin = readFile(log_path);
var parser = new DataflashParser(false);
parser.processData(data_bin.buffer, log_msgs);
var lat = scale(array(parser.messages.POS.Lat), 1E-7);
var lon = scale(array(parser.messages.POS.Lng), 1E-7);
var alt = array(parser.messages.POS.Alt);
var flight_path = latLonAlt2cartesian(lat, lon, alt);

var map_3d = await geoglobe(token);
map_3d.win.setTitle('Flight path');
var entity = map_3d.addEntity({
  polyline: {
    positions: flight_path, width: 5.0, material: map_3d.Cesium.Color.RED
  }
});
entity.flyTo();
disp(' --------------------');
```

Listing 6.16 - `showFlightPath.jsl`

Prikaz na 2D mapi je baziran na biblioteci `Leaflet`, dok je prikaz na 3D mapi baziran na biblioteci `CesiumJS`.

6.2.7.8 MAVLink komunikacioni protokol

`MAVLink` (eng. *Micro Air Vehicle Link*) je veoma jednostavan protokol za razmenu poruka namenjen za komunikaciju sa dronovima (i između ugrađenih komponenti dronova)[157]. Protokol obezbeđuje efikasnu i pouzdanu razmenu podataka, jer se radi o binarnom protokolu promenljive dužine paketa sa proverom integriteta paketa (pomoću `CRC-16/MCRF4XX`). Poruke su definisane u `XML` fajlovima i posebnim generatorima se generišu sve neophodne funkcije i promenljive za određene programske jezike.

Ovaj protokol je veoma značajan zbog svoje široko rasprostranjene primene. Mnoge kompanije su usvojile ovaj protokol i koriste ga u svojim proizvodima već dugi niz godina, što ga čini veoma pouzdanim. Za bespilotne letelice ovaj protokol ima definisane praktično sve potrebne poruke za uobičajenu upotrebu. Ovakav protokol omogućava poseban nivo apstrakcije za autopilote. U radu [158] je analizirano kašnjenje poruka usled korišćenja ovog protokola kao i procenat gubitka paketa. U radovima [159], [160] su prikazane ranjivosti ovog protokola i kako se one mogu iskoristiti, dok je u radu [161] prikazano kako se može implementirati enkripcija poruka.

U listingu 6.17 je prikazana struktura `MAVLink V2` paketa.

```
struct __attribute__((packed)) packet_t {
  uint8_t magic; // packet start marker 0xFD
  uint8_t len;
  uint8_t incompat_flags;
  uint8_t compat_flags;
  uint8_t seq;
  uint8_t sysid;
  uint8_t compid;
  uint8_t msgid1; // first 8 bits of the ID of the message
  uint8_t msgid2; // middle 8 bits of the ID of the message
  uint8_t msgid3; // last 8 bits of the ID of the message
  uint8_t payload[len];
  uint16_t checksum;
  uint8_t signature[13]; // optional
};
```

Listing 6.17 - Struktura MAVLink V2 paketa

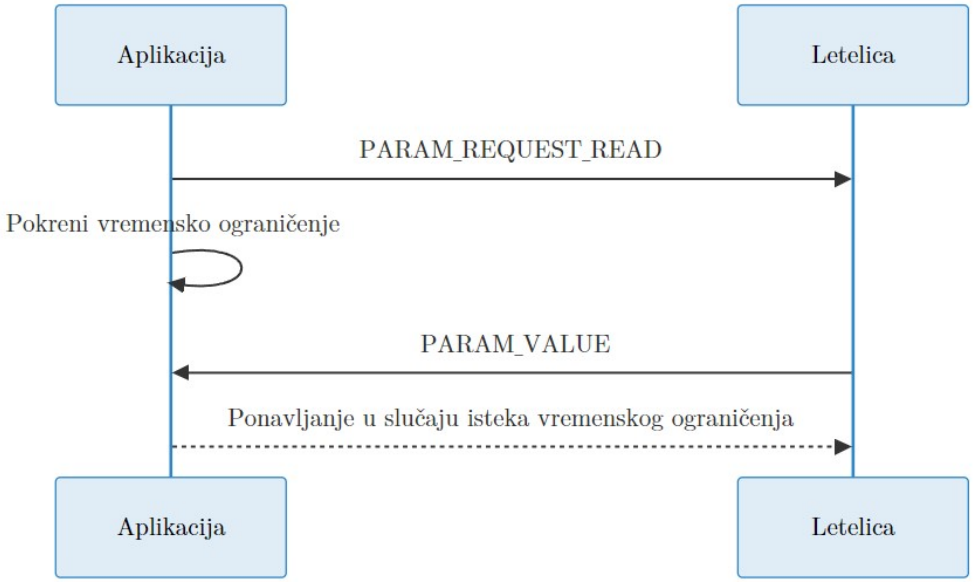
Na osnovu `MAVLink` poruka se formiraju i protokoli višeg nivoa kao što su:

- *Protokol za povezivanje* - Ovaj protokol omogućava svim uređajima unutar mreže da prepoznaju prisustvo jedni drugih i razmene osnovne informacije o stanju sistema. `HEARTBEAT` poruke se periodično šalju radi validacije konekcije i detekcije eventualnih prekida konekcije. Na taj način, protokol za povezivanje osigurava da je svaki uređaj svestan tipa uređaja, trenutnog moda rada, kao i drugih kritičnih parametara koji utiču na koordinaciju i bezbednost u toku rada.

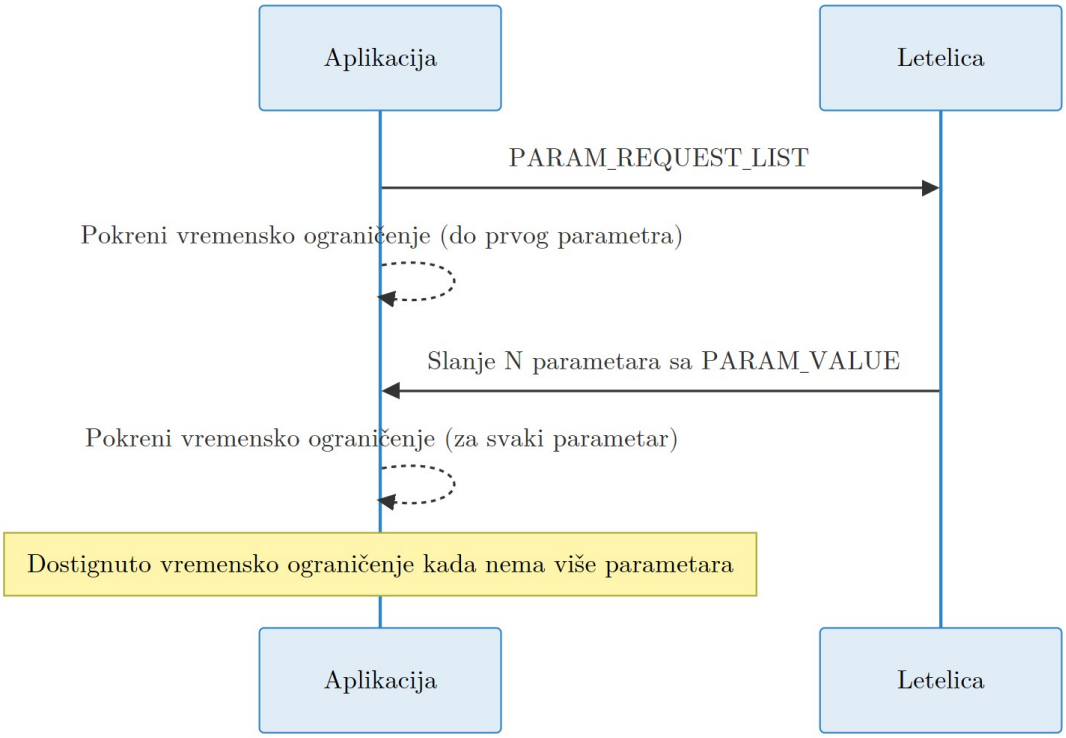
  Najčešće se ove poruke šalju jednom u sekundi (1 Hz) i ukoliko se ova poruka ne primi pet sekundi (pet propuštenih poruka) može se smatrati da je došlo do prekida u komunikaciji.

- *Protokol za parametre* - Protokol za parametre definiše mehanizme čitanja, menjanja i skladištenja parametara. Nakon menjanja nekog parametra, sistem automatski odgovara sa vrednošću koja je podešena što se može iskoristiti za proveru uspešnosti ove aktivnosti.

  Na slici 6.136 je prikazan sekvencijalni dijagram za preuzimanje jednog parametra putem `MAVLink` protokola, dok je na slici 6.137 prikazan dijagram za preuzimanje svih parametara.

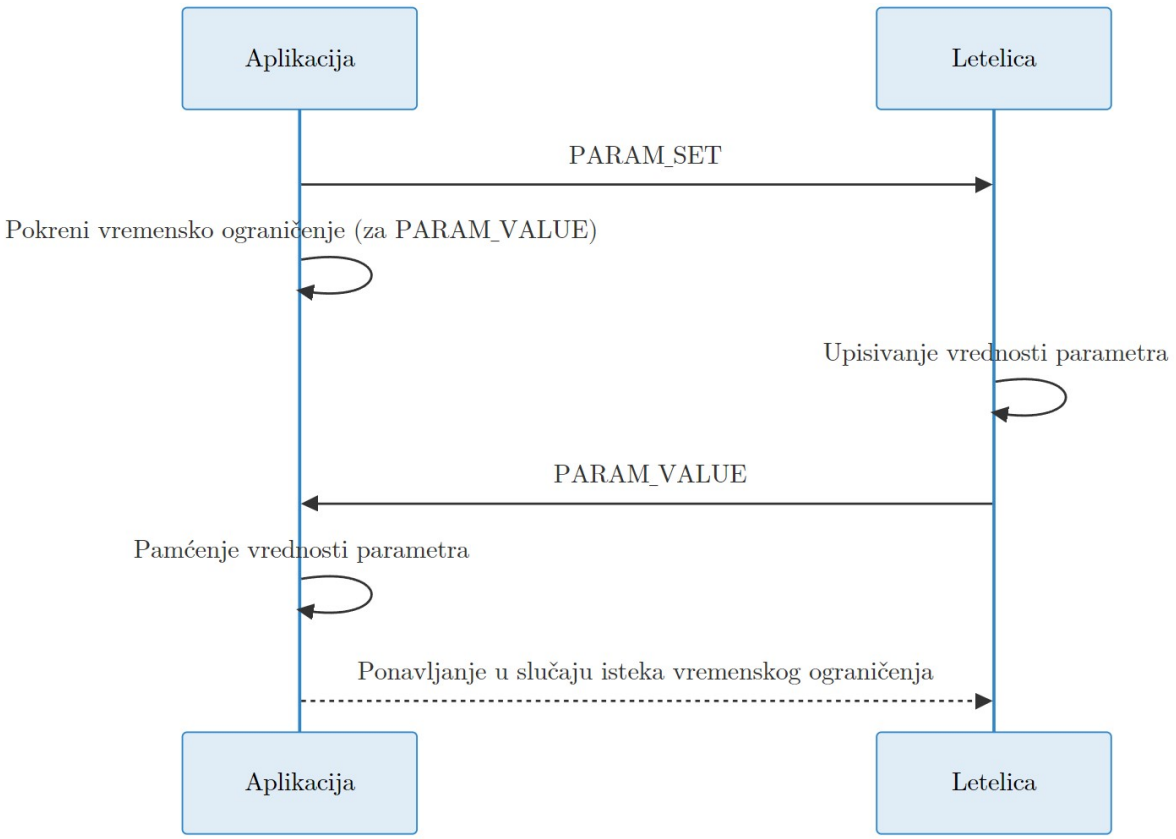


Slika 6.136 - Preuzimanje parametra putem MAVLink-a



Slika 6.137 - Preuzimanje svih parametra putem MAVLink-a

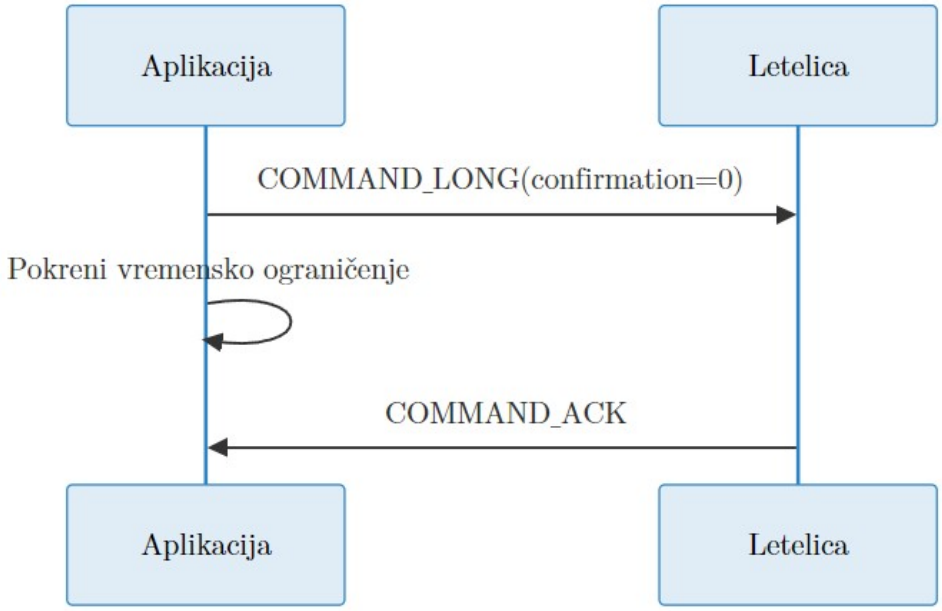
Na slici 6.138 je prikazan sekvencijalni dijagram za podešavanje jednog parametra putem `MAVLink` protokola. Značajno je proveriti nakon što se vrati vrednost da li je željena vrednost zapravo i prihvaćena i sačuvana na letelici.



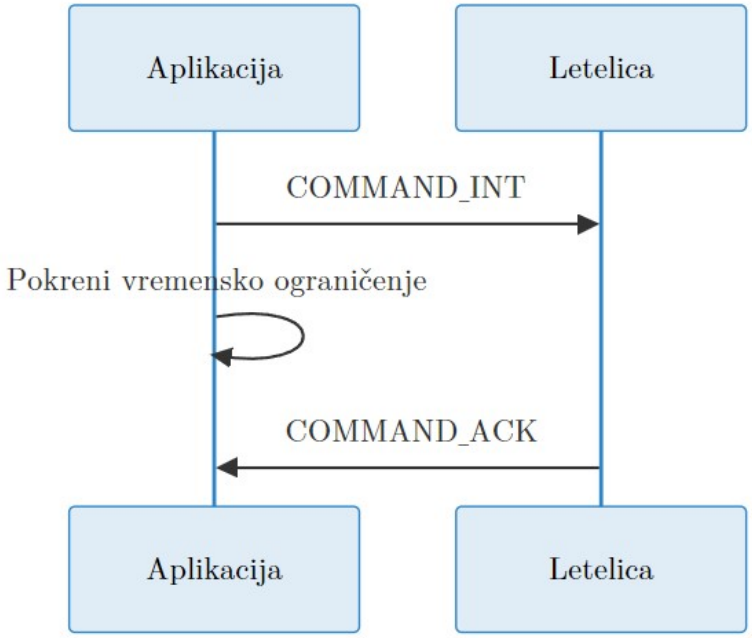
Slika 6.138 - Podešavanje parametara putem MAVLink-a

- *Protokol za komande* - Protokol za komande koristi poseban skup poruka za slanje komandi. Ovaj protokol obuhvata i dvosmernu potvrdu izvršenih komandi, čime se osigurava pouzdanost u uslovima gde može doći do gubitka poruka.

  Na slikama 6.139 i 6.140 su prikazani sekvencijalni dijagrami za slanje komandi letelici putem `MAVLink` protokola. Razlika između komande `COMMAND_LONG` i `COMMAND_INT` je u tipu podataka parametara unutar paketa (*realni* ili *celobrojni*). Kada se šalju geografska širina i dužina treba koristiti komandu `COMMAND_INT` jer ona omogućava veću preciznost sa celobrojnim parametrima `5` i `6` jer se geografska širina i dužina inače skaliraju. Sa druge strane, komandu `COMMAND_LONG` je neophodno koristiti kada su parametri realni brojevi.



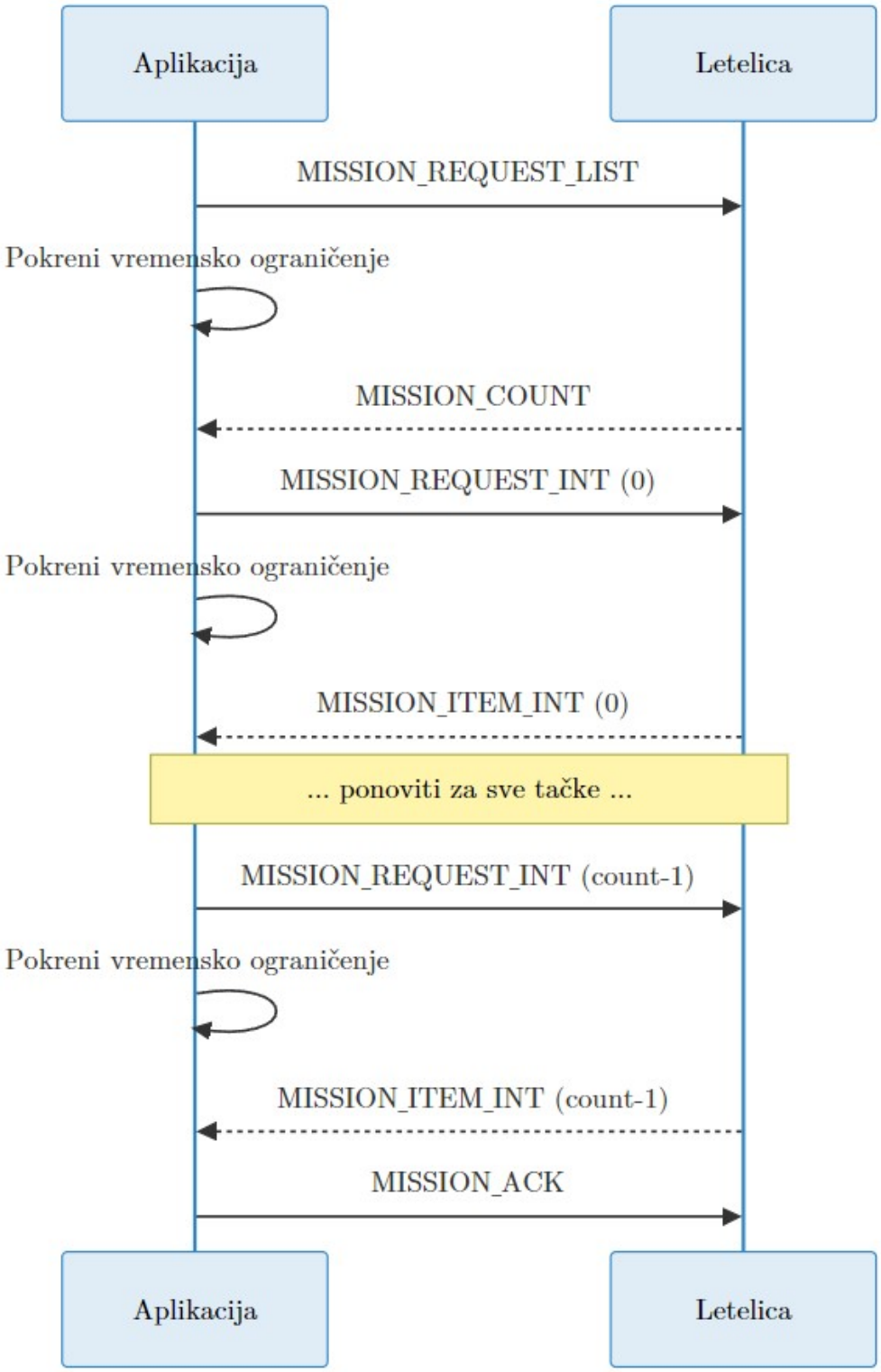
Slika 6.139 - Slanje komande MAVLink porukom COMMAND_LONG



Slika 6.140 - Slanje komande MAVLink porukom COMMAND_INT

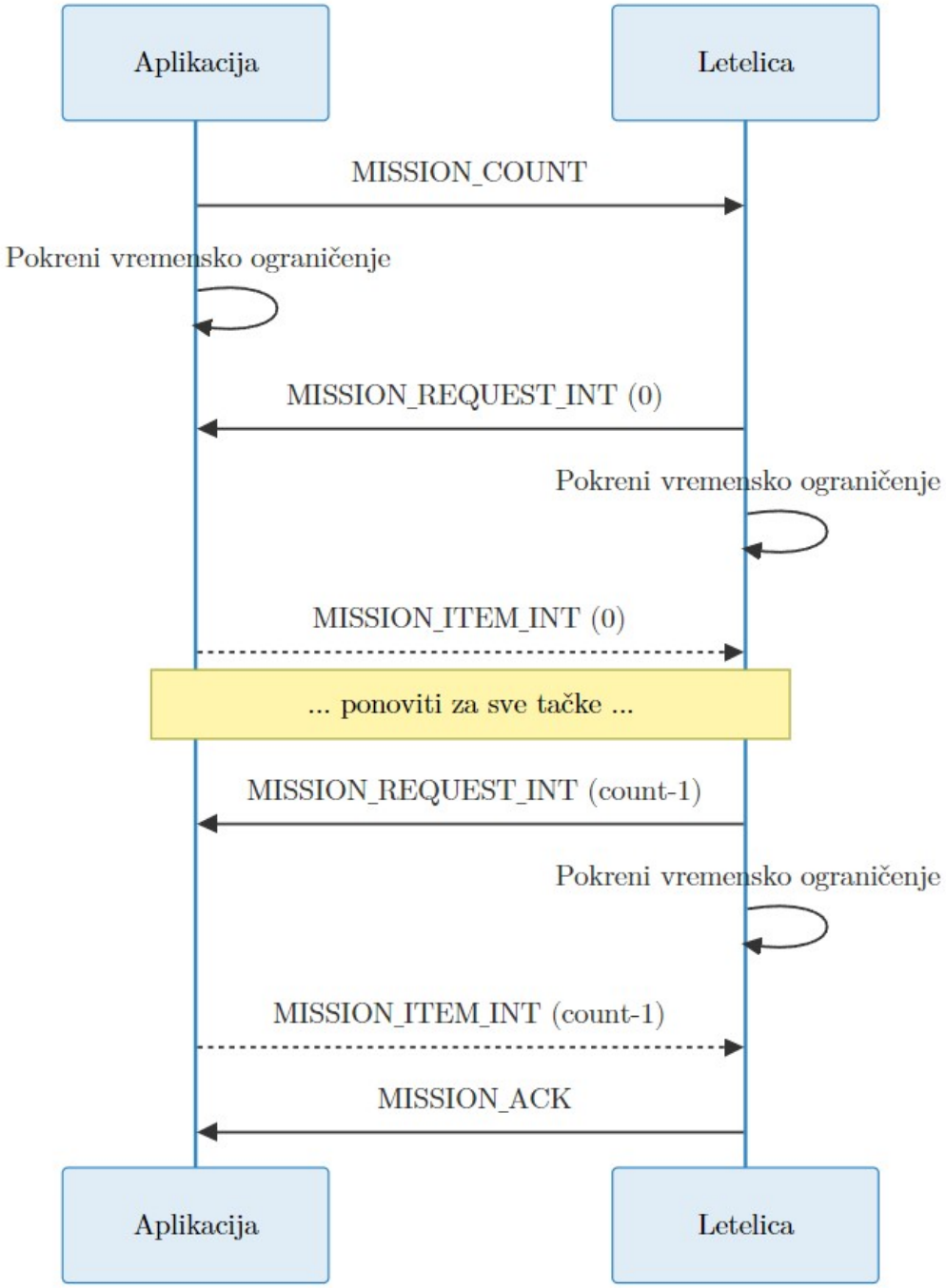
- *Protokol za misije* - Protokol za misije služi za kreiranje, čuvanje i izvršavanje misija. Putem ovog protokola razmenjuju se informacije o putnim tačkama, visini leta, brzinama i drugim relevantnim parametrima, dok letelica istovremeno može da šalje potvrdu o uspešnom izvršenju i status misije.

  Na slici 6.141 je prikazan sekvencijalni dijagram za preuzimanje misije putem `MAVLink` protokola. Svaka tačka misije (kao red u tabeli) se dobija zasebnom porukom.



Slika 6.141 - Preuzimanje misije putem MAVLink-a

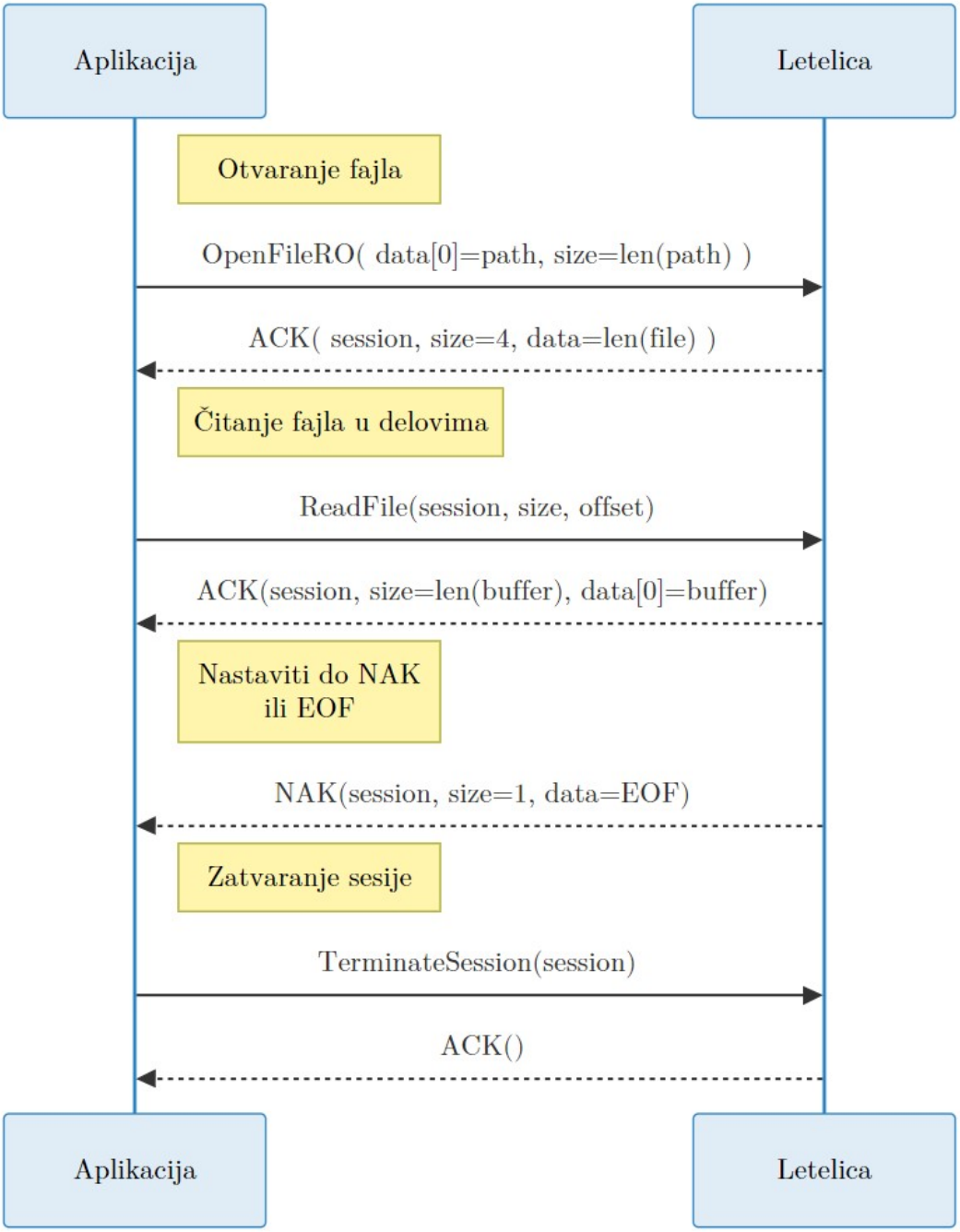
Na slici 6.142 je prikazan sekvencijalni dijagram za otpremanje misije putem `MAVLink` protokola. Svaka tačka misije (kao red u tabeli) se šalje zasebnom porukom, slično kao i prilikom preuzimanja.



Slika 6.142 - Otpremanje misije putem MAVLink-a

- *Protokol za transfer fajlova* (`MAVFTP`) - Ovaj protokol omogućava pouzdanu i sigurnu razmenu fajlova između aplikacije i letelice. Tipičan primer primene uključuje preuzimanje dnevnika leta radi naknadne analize. Protokol podržava segmentiranu razmenu podataka, detekciju grešaka i ponovni prenos oštećenih ili izgubljenih paketa, čime se obezbeđuje integritet sadržaja.

  Na slici 6.143 je prikazan sekvencijalni dijagram za čitanje fajla putem `MAVFTP`-a preko `MAVLink` protokola. Fajl se prenosi u delovima nizom poruka.



Slika 6.143 - Čitanje fajla putem MAVFTP-a

6.2.7.8.1 Implementacija nove poruke

Za sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi je potrebno da letelica šalje sve tri komponente translatornog ubrzanja, ugao propinjanja, ugao valjanja i ugaonu brzinu skretanja.

U listingu 6.18 je data definicija nove `BASIC_REMILP` poruke. Ovu definiciju je potrebno dodati u fajl `ardupilotmega.xml` koji se nalazi na adresi:

`.\modules\mavlink\message_definitions\v1.0\ardupilotmega.xml`

```
<message id="11050" name="BASIC_REMILP">
  <description>Basic Remilp message providing minimal attitude and acceleration
     data</description>
  <field type="uint32_t" name="time_boot_ms" units="ms">Timestamp (time since
     system boot)</field>
  <field type="float" name="ax">Translational acceleration along X axis (m/s^2)<
     /field>
  <field type="float" name="ay">Translational acceleration along Y axis (m/s^2)<
     /field>
  <field type="float" name="az">Translational acceleration along Z axis (m/s^2)<
     /field>
  <field type="float" name="pitch">Pitch angle in radians</field>
  <field type="float" name="roll">Roll angle in radians</field>
  <field type="float" name="yaw_rate">Yaw rate in radians/second</field>
</message>
```

Listing 6.18 - Definicija nove BASIC_REMILP poruke

Za slanje BASIC_REMILP je potrebno 40 bajtova (12+28). Kako bi se dobile iste ove informacije standardnim porukama je potrebno primiti poruke ATTITUDE (12+24) i SCALED_IMU (12+24) za koje je potrebno ukupno 72 bajta, što znači da je moguće postići isto sa samo 55% protoka koji je standardno potreban.

Za slanje poruke BASIC_REMILP je prvo potrebno u fajlu APM_Config.h otkomentarisati sledeću liniju.

```
#define USERHOOK_50HZLOOP userhook_50Hz();
```

Zatim, u okviru fajla UserCode.cpp uključiti potrebne biblioteke i u funkciji userhook_50Hz() poslati poruku BASIC_REMILP kao što je to urađeno u listingu 6.19.

```
#include <AP_AHRS/AP_AHRS.h>
#include <AP_InertialSensor/AP_InertialSensor.h>
#include <AP_Math/AP_Math.h>

...

#ifdef USERHOOK_50HZLOOP
void Copter::userhook_50Hz()
{
  const mavlink_basic_remilp_t packet {
      time_boot_ms : AP_HAL::millis(),
      ax : ahrs.get_accel_ef().x, // m/s^2
      ay : ahrs.get_accel_ef().y, // m/s^2
      az: ahrs.get_accel_ef().z + GRAVITY_MSS, // m/s^2
      pitch: ahrs.get_pitch() * RAD_TO_DEG, // deg
      roll: ahrs.get_roll() * RAD_TO_DEG, // deg
      yaw_rate: ins.get_gyro().z * RAD_TO_DEG // deg/s
  };

  gcs().send_to_active_channels(MAVLINK_MSG_ID_BASIC_REMILP, (const char *)&
     packet);
}
#endif

...
```

Listing 6.19 - Slanje BASIC_REMILP poruke iz UserCode.cpp

Za testiranje moguće je kompajlirati SITL pomoću sledećih komandi:

```
./waf configure --board=sitl --toolchain x86_64-pc-cygwin
```

```
./waf copter
.\build\sitl\libraries\GCS_MAVLink\include\mavlink\v2.0\ardupilotmega
```

Nakon kompajliranja se automatski generiše fajl `mavlink_msg_basic_remilp.h` sa svim podacima o poruci. Ovaj fajl se nalazi u folderu:

`.\build\sitl\libraries\GCS_MAVLink\include\mavlink\v2.0\ardupilotmega`

Na osnovu ovog fajla se u `JSLAB`-u formira nova poruka (`BasicRemilp`) kao što je prikazano u listingu Z.1.

6.2.8 Virtuelne letelice

Virtuelne bespilotne letelice predstavljaju savremeni pristup obuci i usavršavanju pilota dronova, kombinujući napredne simulacione tehnologije sa sofisticiranim prikazima realnih okruženja. Njihova ključna vrednost ogleda se u mogućnosti kreiranja bezbednog i kontrolisanog okruženja za sticanje početnih znanja o letelicama, kao i za kontinuirano uvežbavanje upravljačkih veština nakon završetka obuke. Na taj način, polaznici mogu da se upoznaju sa funkcionalnostima i ograničenjima dronova bez stvarnog rizika od ljudskih ili materijalnih gubitaka, što obuku čini efikasnijom i pristupačnijom u poređenju sa klasičnim obukama sa pravim letelicama.

Osim toga, obuka u virtuelnom okruženju može da se organizuje u bilo koje vreme, nezavisno od dnevnog svetla ili meteoroloških uslova, što doprinosi fleksibilnosti i kontinuitetu obuke. Nivo realizma kod simulacija omogućava pilotima dronova da se susretnu sa raznolikim scenarijima, poput iznenadnih vremenskih nepogoda, gubitka signala ili kvara određenih komandi na letelici. Upravo ovakvi uslovi, koji se relativno retko sreću tokom klasične obuke, mogu biti od ključnog značaja za reagovanje u realnim situacijama.

Ovakav pristup obuci takođe nudi kontinuirano praćenje napretka i objektivno ocenjivanje polaznika obuke. Analiza učinka može da uključuje preciznost manevrisanja, vreme reakcije i kvalitet donošenja odluka u složenim situacijama. Na taj način, polaznici obuke dobijaju detaljan uvid u sopstvene performanse, kao i konkretne smernice za unapređenje. Kontinuirana evaluacija predstavlja ključ za razvoj visokokvalifikovanih stručnjaka, čiji rad u realnim uslovima može znatno doprineti povećanju bezbednosti i efikasnosti misija u kojima se koriste bespilotne letelice.

Od sistema za generisanje virtuelne letelice se očekuje da implementira fiziku koja je što je moguće približnija stvarnoj, da su letelice modelirane tako da reaguju na komande i sredinu gotovo na isti način kao prave ali i da pružaju trodimenzionalni prikaz okoline oko letelice.

U radu [162] je prikazana implementacija realističnog simulatora za multikoptere baziranog na `Unity` biblioteci, dok je u [163] takođe napravljen matematički model multikoptera (virtuelna letelica) sa ciljem analize njegovog upravljanja.

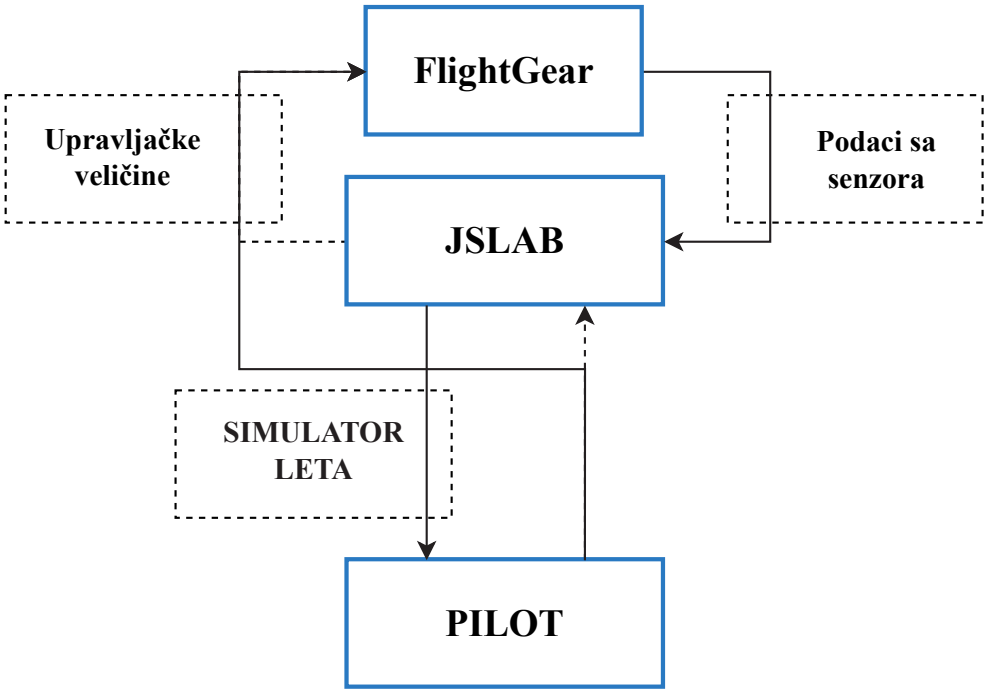
6.2.8.1 FlightGear

Od mnogih programskih paketa za simuliranje leta od kojih su možda najpoznatiji *X-Plane* i *Microsoft Flight Simulator* je izabran programski paket *FlightGear*, koji predstavlja simulator leta otvorenog koda baziran na `JSBSim` modelu dinamike leta (`FDM`, eng. *Flight Dynamics Model*). Model dinamike leta je u suštini fizički i matematički model koji definiše kretanje aviona, rakete, itd. u zavisnosti od sila i momenata koji na njega deluju. U radu [164] je prikazano korišćenje *X-Plane* simulatora u sprezi sa pravim autopilotom.

Cilj projekta *FlightGear* je da se napravi sofisticirani i otvoreni simulator leta za upotrebu u istraživanju ili akademskom okruženju, za obuku pilota, kao alat za inženjere u vazduhoplovnoj

industriji, i na kraju za zabavu kao realni izazov za svakoga ko želi da se oproba u ulozi pilota[165].

Na slici 6.144 prikazan je blok dijagram sistema koji čine pilot, simulator leta, *FlightGear* i program `JSLAB`.



Slika 6.144 - Tok podataka između JSLAB-a i FlightGear-a

Skripta `flightGearSimulationUDP.jsl` koja implementira prethodno definisan sistem data je u okviru priloga Y. Na slici 6.145 prikazana je upotreba ovog sistema.



Slika 6.145 - Povezivanje FlightGear programa sa umanjenim prototipom simulatora leta

Ova skripta se povezuje sa umanjenim prototipom simulatora leta (koji se na početku odgovarujuće konfiguriše), zatim pokreće *FlightGear* sa odgovarajućim parametrima kako bi se pokrenula željena simulacija i kako bi bilo moguće dobijanje sve tri komponente translatornog ubrzanja, zatim dobijanje ugla propinjanja, ugla valjanja i ugaone brzine skretanja preko `UDP`-a. Program *FlightGear* obično automatski detektuje povezanu kontrolnu palicu za upravljanje letelicom. Kada se parametri kretanja dobiju od *FlightGear* programa oni se prosleđuju odmah simulatoru leta koji se na osnovu njih kreće.

6.2.8.2 AirSim

Kompanija *Microsoft* je razvila simulator otvorenog koda pod nazivom `AirSim`, baziran na `Unreal Engine` 3D platformi za razvoj video-igara, za autonomna vozila i letelice[166].

Prednost ovog sistema u odnosu na *FlightGear* je u tome što koristi izuzetno napredno trodimenzionalno okruženje što pruža veoma realističan prikaz okoline. Projekat `AirSim` nadograđuje ovo okruženje sa implementacijom dinamike leta letelice. Cilj projekta bio je da posluži u procesu testiranja autonomnih vozila i letelica, kao i u procesu mašinskog učenja. Implementiraju se različiti napredni senzori kao što su optičke kamere, senzori daljine, termalne kamere, kamere dubine, lidari, radari i drugi. Sa svim ovim dostupnim senzorima se testiraju algoritmi za autonomno kretanje. Na slici 6.146 je prikazan model letelice, animacija ispuštanja bombe i eksplozija pomoću `Unreal Engine`-a.

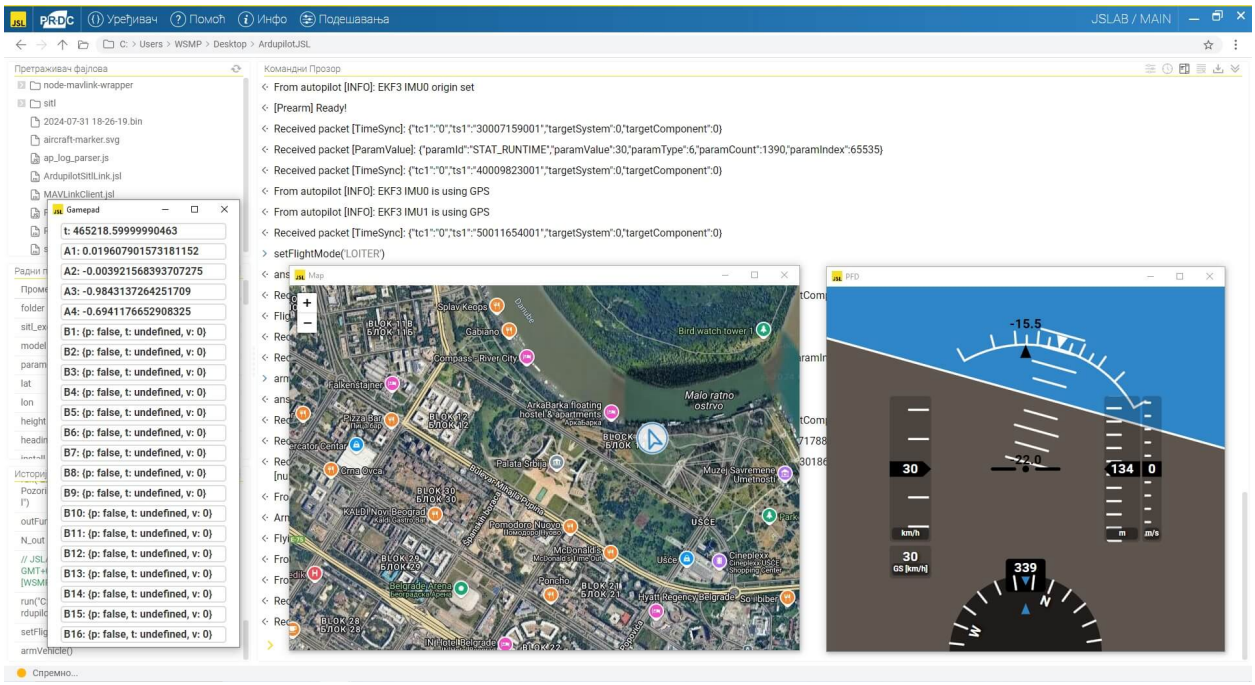


Slika 6.146 - Prikaz letelice, animacija ispuštanja bombe i eksplozija pomoću Unreal Engine-a

Kompanija *Microsoft* je odustala od daljeg razvoja ovog projekta, ali je projekat otvorenog koda[167] pa se nastavlja njegov razvoj od strane drugih lica i kompanija, ali sa neizvesnom budućnošću. U [168] je prikazano povezivanje `AirSim`-a sa `ROS` i `PX4` autopilotom.

6.2.8.3 ArduPilot SITL

Virtuelnom letelicom se može smatrati i *ArduPilot* `SITL`[169] čak i ako ne obezbeđuje prikaz okoline oko letelice već samo model dinamike leta. Ono što je velika prednost `SITL`-a u odnosu na prethodno opisane i spomenute sisteme jeste što je moguće korišćenje `MAVLink` protokola na potpuno identičan način kao i sa stvarnom letelicom.



Slika 6.147 - Prikaz instrumenata u JSLAB-u nakon povezivanja sa ArduPilot SITL

U prilogu Z je data skripta `ArdupilotSitlLink.jsl` koja pokreće *ArduPilot* `SITL` i povezuje sa njim putem `TCP`-a sa `MAVLink` protokolom. Za `MAVLink` se koristi modul `node-mavlink` koji je zvanični `MAVLink` modul za *Node.js*. Na slici 6.147 su prikazani instrumenti ove virtuelne letelice. Skripta takođe detektuje da li je povezan `USB` kontroler i ako jeste prosleđuje kontrole letelici putem `MAVLink`-a. Pri pokretanju se za `SITL` definiše model letelice, njeni parametri kao i početna lokacija, visina i ugao skretanja.

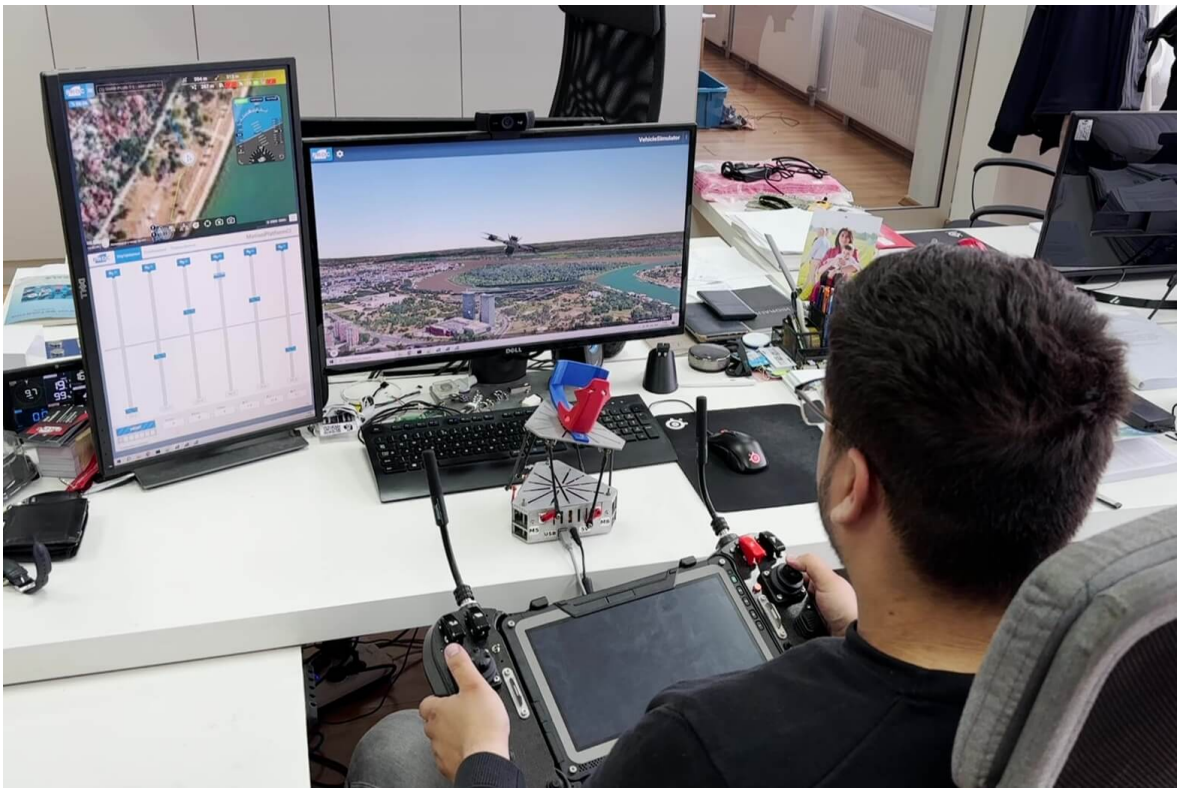
Na slici 6.147 se dole desno vidi primarni ekran leta (`PFD`, eng. *Primary Flight Display*), kao najbitniji instrument sa indikacijom veštačkog horizonta (uglovi propinjanja i valjanja), ugla skretanja, brzine kretanja u odnosu na zemlju, brzine kretanja u odnosu na vazduh, nadmorske visine leta i brzine penjanja. Pored njega levo se nalazi prikaz lokacije i ugla skretanja letelice na 2D mapi sveta. Implementacija primarnog ekrana leta je u celosti data zajedno sa skriptom u prilogu Z.

6.2.8.4 VehicleSimulator

Program `VehicleSimulator` takođe omogućava upravljanje virtuelnom letelicom putem `MAVLink` komunikacionog protokola. Upravljanje letelicom je prikazano na slici 6.148. U ovom slučaju je korišćena aplikacija `MotionControlCI` za povezivanje virtuelne letelice preko `MAVLink`-a sa umanjenim prototipom simulatora leta.

Aplikacija omogućava vizualizaciju okruženja letelice i bazirana je na eksperimentalno dobijenom matematičkom modelu koji odgovara stvarnim letelicama proizvođača programa. Program omogućava simulaciju pojedinačnih letelica ili roja letelica. U odnosu na sve prethodno prikazane letelice ovaj simulator omogućava i `RTSP` strimovanje kamere sa pogledom sa letelice u realnom vremenu kao i `REST API` za telemetriju i kontrolu.

IKA-TRAINER je trenažer za obuku operatera koji se bazira na aplikaciji `VehicleSimulator`[170]. Trenažer ima radio modul koji odgovara onom na letelici pa se može zaključiti, a na osnovu toga što je omogućeno bežično povezivanje putem `MAVLink` komunikacionog protokola i `RTSP` strimovanje kamere, da ova virtuelna letelica može u potpunosti simulirati letelicu kao što je *REMILP-01* tako da za korisnika ne postoji nikakva razlika da li se povezuje sa pravom letelicom ili virtuelnom.



Slika 6.148 - Upravljanje virtuelnom letelicom programa VehicleSimulator

Kada se daljinski upravljač poveže sa trenažerom sve na ekranu je potpuno isto kao u pravoj misiji. Trenažer se može koristiti za osnovnu letačku obuku, ali i za simulaciju složenih izviđačkih i borbenih zadataka. Trodimenzionalno okruženje omogućava korišćenje stvarnih lokacija ili formiranje novih modela okoline. Na slici 6.149 je prikazana kompletna virtuelna soba za *IKA-TRAINER*. Slično je relativno lako moguće uraditi i za simulator leta.



Slika 6.149 - Virtuelna soba za IKA-TRAINER

6.2.9 RTSP server

Prethodno je prikazano da kamere na letelicama, ali i virtuelne letelice, implementiraju RTSP servere kako bi se prenosila slika sa kamere do operatera daljinski upravljane letelice. Ovo je jedan od ključnih elemenata sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

U prilogu AA je data skripta ServerRTSP.jsl kojom se implementira RTSP server u programu JSLAB. Ova skripta je i eksperimentalno ispitana kao što je prikazano u poglavlju 7.5. Za samu implementaciju RTSP servera zaslužan je modul server-rtsp-live555 napisan u programskom jeziku C++ koji koristi biblioteku LIVE555 za server i biblioteku FFmpeg za enkodiranje video snimka. Dodatno se može koristiti i OpenCV biblioteka za obradu i analizu slika. Skripta uzima sliku ekrana ili sliku sa kamere, u obliku niza brojeva koji odgovaraju bojama po pikselima, ovu sliku prosleđuje modulu server-rtsp-live555 koji je enkodira i, ukoliko postoje RTSP klijenti, prosleđuje RTP pakete klijentima servera.

Najviše dostupnih podešavanja se zapravo tiče načina enkodiranja video snimka dok su ostali delovi ovog procesa gotovo fiksni. Tokom analize sistema kao celine, treba uzeti u obzir efikasnost akvizicije slike kao i dostupan protok i stabilnost komunikacije. Danas su sve više u upotrebi FPGA čipovi i procesori sa integrisanim blokovima za hardversko enkodiranje i dekodiranje video snimka kako bi se maksimalno smanjilo kašnjenje i opterećenje glavnog procesora. Ovakvi čipovi su sposobni za efikasnu i paralelnu obradu velikog broja piksel-podataka u realnom vremenu.

6.2.10 Matematički modeli letelice i pilota za predupravljanje simulatora

Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi osim simulatora leta čine još i pilot i bespilotna letelica. Ukoliko je moguće pretpostaviti kako će pilot reagovati na neki poremećaj ili kako će letelica reagovati na komandu pilota, tada je moguće izvršiti predupravljanje simulatorom. To praktično znači da se platforma upravlja na osnovu pretpostavke kako će se letelica kretati što omogućava bolji odziv simulatora leta nego u slučaju da se koristi samo informacija dobijena merenjima na letelici.

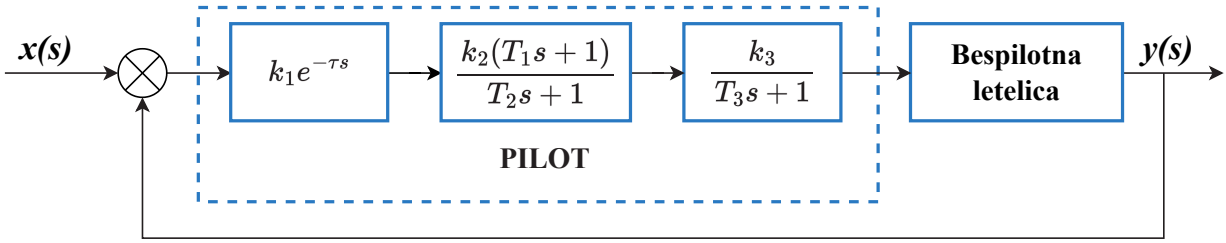
U sistemu ručnog upravljanja letelicom, gde se sama letelica posmatra kao objekat upravljanja, pilot se pojavljuje kao dinamički element sa povratnom spregom. Na osnovu informacija koje prikuplja, bilo pre leta ili tokom njega, pilot određuje parametre kretanja letelice, nastojeći da što efikasnije vrši korekciju eventualne razlike između ostvarenih i željenih parametara.

Istraživanja i eksperimentalna ispitivanja, kao što je prikazano u radovima [171]–[174], sugerišu da se čovek pilot može matematički modelirati kao servouređaj sa određenim dinamičkim karakteristikama. Istovremeno, [175] nudi modele za pojedinačne čovekove sisteme. Eksperiment za određivanje prenosne funkcije pilota se zasniva na tome što pilot prima ulazne signale, kao što su otkloni kazaljki instrumenata od nultog položaja, vizuelne informacije sa ekrana ili signali percipirani sluhom i na osnovu tih podataka preduzima korektivne radnje kako bi vratio instrumente u željeno stanje ili neutralizovao primljene signale.

Statistička analiza dobijenih rezultata eksperimenta za određivanje prenosne funkcije na velikom broju pilota ukazuje na to da prenosna funkcija zavisi kako od karakteristika ulaznih veličina, tako i od dinamičkih osobina same letelice. Iako je priroda prenosne funkcije zapravo nelinearna, eksperimentalna opažanja pokazuju da se dinamička svojstva pilota mogu aproksimirati kao kvazilinearna u opsegu frekvencija od 1.2 Hz do 1.5 Hz u određenom

vremenskom, ne tako dugom periodu [176]. Na osnovu tih rezultata, matematički model pilota je moguće prikazati blok dijagramom na slici 6.150 i predstaviti prenosnom funkcijom sledećeg oblika:

$$W(s) = \frac{ke^{-\tau s}(T_1 s + 1)}{(T_2 s + 1)(T_2 s + 1)} \tag{6.1}$$



Slika 6.150 - Blok dijagram sa prenosnom funkcijom pilota

Na prikazanom blok dijagramu, prvi blok predstavlja rad *nervnog sistema* koji inicira reakciju, dok drugi blok, delujući kao modulator, pretvara ulazne signale u odgovarajuće komandne veličine unutar *neuromuskularnog sistema*. Treći blok, koji simbolizuje sam *neuromuskularni sistem*, predstavlja aktuator komandnog organa.

Važno je istaći da pilot kao čovek operater ne funkcioniše isključivo kao linearni servouređaj. Pored osnovnog upravljačkog signala, on unosi i slučajne pomeraje (kao šum) koji nisu obuhvaćeni ovim modelom, čime se uvodi dodatna nelinearnost. Takođe, model ne obuhvata tendenciju pilota da, nakon prestanka ulaznog signala, nastavi da reaguje na osnovu prethodne vrednosti tog signala. Upravljanje letelicom se, uz to, vrši diskretnim pomeranjem komandnih organa tokom vremena.

Na kraju, prenosna funkcija pilota može se dodatno uprostiti tako što se može zameniti jednom od dve predložene funkcije, pri čemu se uzimaju u obzir ili kombinovano prvi i treći blok, ili samo prvi blok.

$$W(s) = \frac{ke^{-\tau s}}{Ts + 1} \quad ili \quad W(s) = ke^{-(\tau + T)s} \tag{6.2}$$

Kao što je pomenuto, dodatna mogućnost za formiranje predupravljanja je matematičko modeliranje odziva bespilotne letelice. Moguće je formirati proceduru i definisati opremu na osnovu koje je moguće eksperimentalno odrediti matricu sistema i matricu ulaza iz linearizovane jednačine stanja ili određenu prenosnu funkciju sistema bespilotne letelice. U nastavku su date linearizovane jednačine stabilnosti letelice sa fiksnim krilom i klasične aerodinamičke konstrukcije.

Jednačine stabilnosti letelice se mogu podeliti u dve grupe jednačina i to, jednačine longitudinalne (uzdužne ili simetrične) i jednačine lateralne (poprečne ili asimetrične) dinamike [177]. Linearizovane jednačine stabilnosti se mogu napisati u obliku jednačine stanja i jednačine izlaza što omogućava matrični račun koji je jednostavniji za implementaciju u okviru računarskih programa.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{B}\,\boldsymbol{u} \tag{6.3}$$

$$\boldsymbol{y} = \boldsymbol{C}\,\boldsymbol{x} + \boldsymbol{D}\,\boldsymbol{u} \tag{6.4}$$

Za *longitudinalnu dinamiku* su vektori veličina stanja i ulaznih veličina:

$$\boldsymbol{x} = \left[\begin{array}{cccc} u, & w, & q, & \theta \end{array} \right]^T \tag{6.5}$$

$$\boldsymbol{u} = \left[\begin{array}{cc} \delta_e, & \delta_T \end{array} \right]^T \tag{6.6}$$

Matrica sistema A_{long} i matrica ulaza B_{long} *longitudinalne dinamike* u stabilnosnom koordinatnom sistemu su pomoću derivativa stabilnosti definisane na sledeći način:

$$\boldsymbol{A_{long}} = \left[\begin{array}{cccc} X_u & X_w & 0 & -\mathrm{g} \\ Z_u & Z_w & u_0 & 0 \\ M_u + M_{\dot{w}} Z_u & M_w + M_{\dot{w}} Z_w & M_q + M_{\dot{w}} u_0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right] \tag{6.7}$$

$$\boldsymbol{B_{long}} = \left[\begin{array}{cc} X_{\delta_e} & X_{\delta_T} \\ Z_{\delta_e} & Z_{\delta_T} \\ M_{\delta_e} + M_{\dot{w}} Z_{\delta_e} & M_{\delta_T} + M_{\dot{w}} Z_{\delta_T} \\ 0 & 0 \end{array} \right] \tag{6.8}$$

Vektori veličina stanja i ulazne veličine *lateralne dinamike* su:

$$\boldsymbol{x} = \left[\begin{array}{cccc} \beta, & p, & r, & \phi \end{array} \right]^T \tag{6.9}$$

$$\boldsymbol{u} = \left[\begin{array}{cc} \delta_a, & \delta_r \end{array} \right]^T \tag{6.10}$$

Za *lateralnu dinamiku* su matrica sistema A_{lat} i matrica ulaza B_{lat} pomoću derivativa stabilnosti definisane na sledeći način:

$$\boldsymbol{A_{lat}} = \left[\begin{array}{cccc} \frac{Y_\beta}{u_0} & \frac{Y_p}{u_0} & \frac{Y_r}{u_0} - 1 & \frac{g \cos\theta_0}{u_0} \\ L_\beta & L_p & L_r & 0 \\ N_\beta & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{array} \right] \tag{6.11}$$

$$\boldsymbol{B_{lat}} = \left[\begin{array}{cc} 0 & \frac{Y_{\delta_r}}{u_0} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \end{array} \right] \tag{6.12}$$

Na osnovu prethodnih jednačina (izvedenih u [178]) je do dinamičkog modela letelice moguće doći ukoliko su poznate geometrijske veličine letelice, inercijalne karakteristike, ako je odgovarajućim veličinama u potpunosti definisan slučaj leta i ako su poznati svi potrebni aerodinamički derivativi.

Do dinamičkog modela letelice je moguće doći i na inverzni način eksperimentom, ukoliko je poznat odziv sistema, poremećaji koji u toku eksperimenta deluju na sistem i promene ulaznih veličina sistema u vremenu moguće je izvršiti identifikaciju sistema. Za potrebe merenja tokom eksperimenta treba izabrati odgovarajuće senzore i njih ugraditi u letelicu. Tokom eksperimenta je potrebno skladištiti rezultate merenja, kako bi se oni nakon eksperimenta obradili na odgovarajući način i na osnovu njih matematičkim modelima uradila identifikacija sistema.

6.3 Pilotska kabina

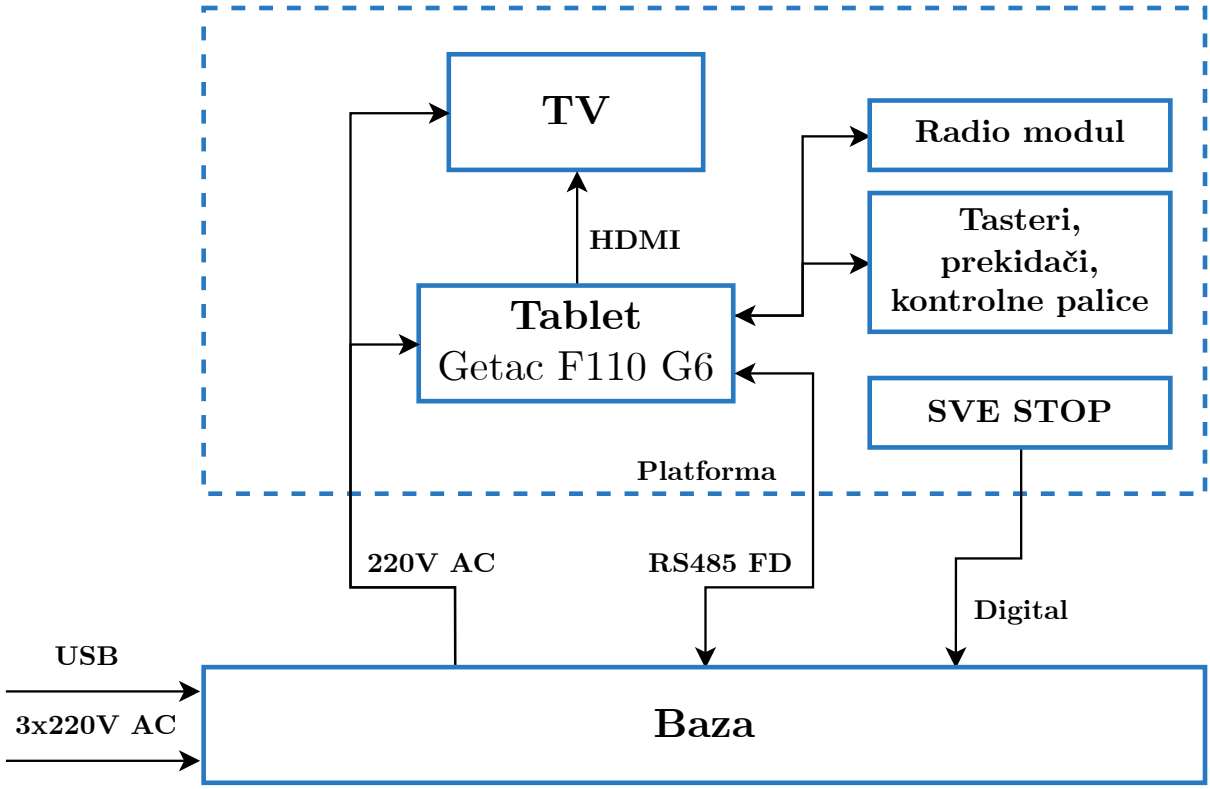
Da bi se realizovao sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi na platformi mehanizma simulatora leta je potrebno formirati pilotsku kabinu koja ima sve ono potrebno pilotu za daljinsko upravljanje letelicom. Na slici 6.151 je prikazan model pilotske kabine za prototip simulatora leta u pravoj veličini.



Slika 6.151 - Pilotska kabina na platformi simulatora leta

Na slici 6.152 je prikazan blok dijagram pilotske kabine prototipa simulatora leta u stvarnoj veličini `MP-6DOF-XXX`. Pilotu se na velikom ekranu u visini njegove glave prikazuje snimak sa kamere na letelici. Osnovni instrumenti se prikazuju na tablet računaru daljinskog upravljača. Ovaj daljinski upravljač je montiran na nosač podesiv po visini i po orijentaciji. Daljinski upravljač ima integrisan radio modul, tastere, prekidače i kontrolne palice. Za hitno zaustavljenje je integrisan taster `SVE STOP` kojim se direktno isključuje napajanje aktuatorima. Mehanička veza između baze i platforme mehanizma je ukupno 12 poluga dok je

električna veza nekoliko žica i to monofazna struja (za napajanje tableta, ekrana i daljinskog upravljača), `Pun-dupleks RS485` za komunikaciju i signal tastera za hitno zaustavljanje.



Slika 6.152 - Blok dijagram pilotske kabine simulatora leta `MP-6DOF-XXX`

Tablet računar daljinskog upravljača komunicira istovremeno i sa letelicom i sa simulatorom leta, što znači da on generiše upravljanje za simulator leta na osnovu podataka koje je dobio od letelice. U okviru ovog programa koji generiše upravljanje se implementiraju i filteri kretanja. Električna integracija simulatora sa ostatkom sistema je zapravo veoma jednostavna, razmenjuju se unapred definisane poruke putem `USB`-a ili `RS485` komunikacije. Istom komunikacijom kojom tablet računar šalje komande za simulator leta, on dobija nazad i podatke o statusu celog sistema i pojedinačnih podsistema simulatora leta.

U sedištu na platformi sedi pilot. Ovo sedište zajedno sa specijalnim pojasevima mora da zadrži pilota na samom simulatoru dok na njega dejstvuju inercijalne sile kao posledica kretanja. Pilot drži stopala oslonjena na podesivom nosaču. Za penjanje na platformu i korišćenje sedišta su obezbeđene stepenice i gazište na platformi.

6.3.1 Interakcija pilota i sistema

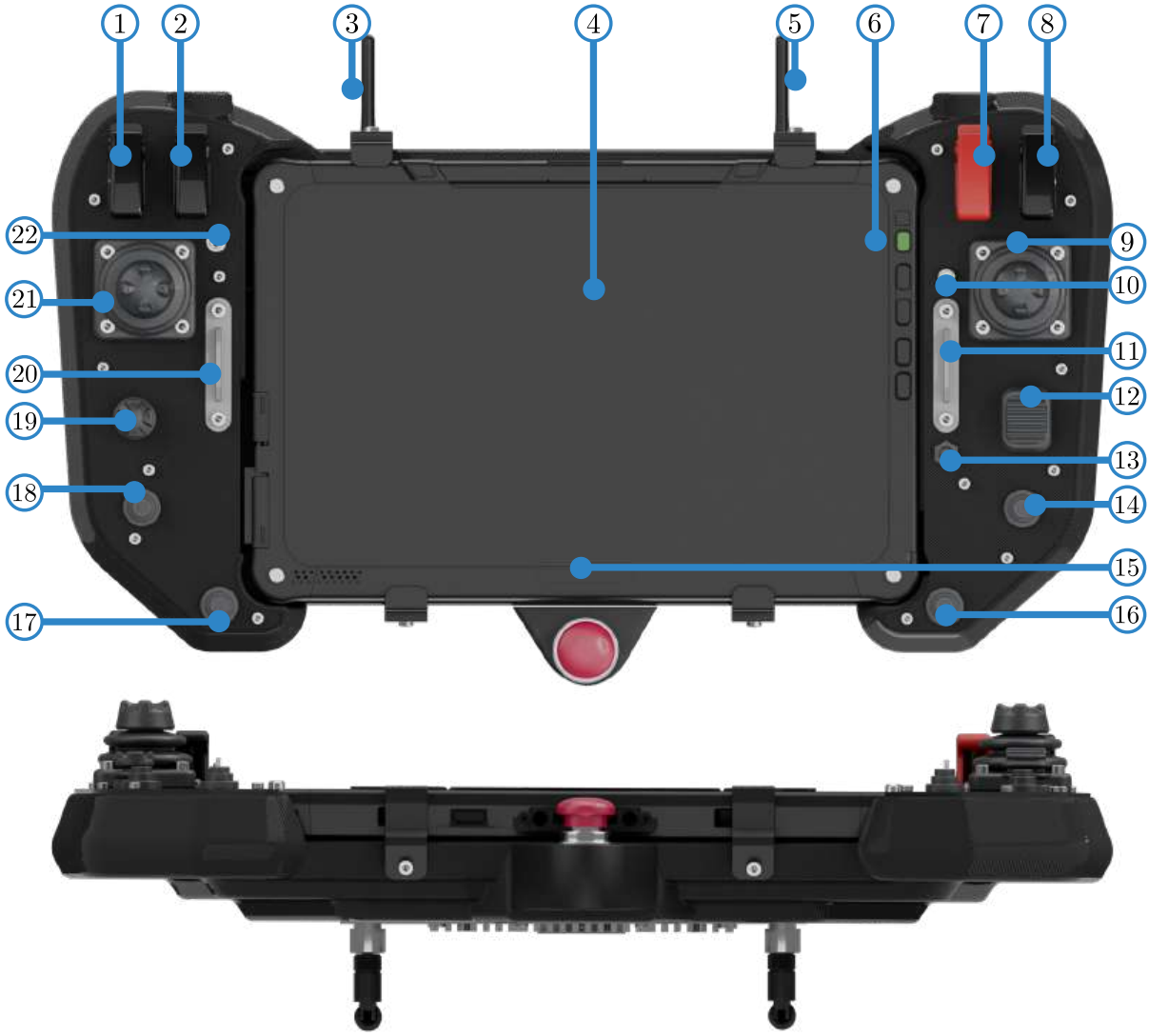
Najbitniji deo pilotske kabine je zapravo modifikovana verzija daljinskog upravljača `IKA-CTRL`. Ovaj daljinski upravljač omogućava interakciju između pilota i ostatka sistema. Na slici 6.153 je prikazana pilotska kabina prototipa simulatora leta u stvarnoj veličini `MP-6DOF-XXX` iz ugla pilota. Taster za hitno zaustavljanje je na lako pristupačnom položaju ispod tableta daljinskog upravljača. Samim tim što se koristi daljinski upravljač koji je već ispitan i praktično dokazan se otklanja sumnja u to da li su pilotu dostupne sve neophodne komande. Jedine modifikacije ovog daljinskog upravljača su mehaničke. Antene se montiraju sa zadnje strane i promenjen je način veze sa nosačem (zbog kretanja je značajno ojačana veza u odnosu na onu standardnu za stalak daljinskog upravljača).



Slika 6.153 - Pilotska kabina iz ugla pilota

Značajno je da se upravljanje letelicom može nastaviti u slučaju otkaza tableta ili aplikacije za upravljanje jer je let moguć samo na osnovu kontrolnih palica, tastera i prekidača daljinskog upravljača. Sa druge strane u slučaju otkaza neke kontrolne palice, tastera ili prekidača, u aplikaciji za upravljanje se implementiraju sekundarne komande koje je moguće koristiti direktno preko ekrana osetljivog na dodir (tablet računara).

Na slici 6.154 su označeni svi delovi daljinskog upravljača IKA-CTRL.



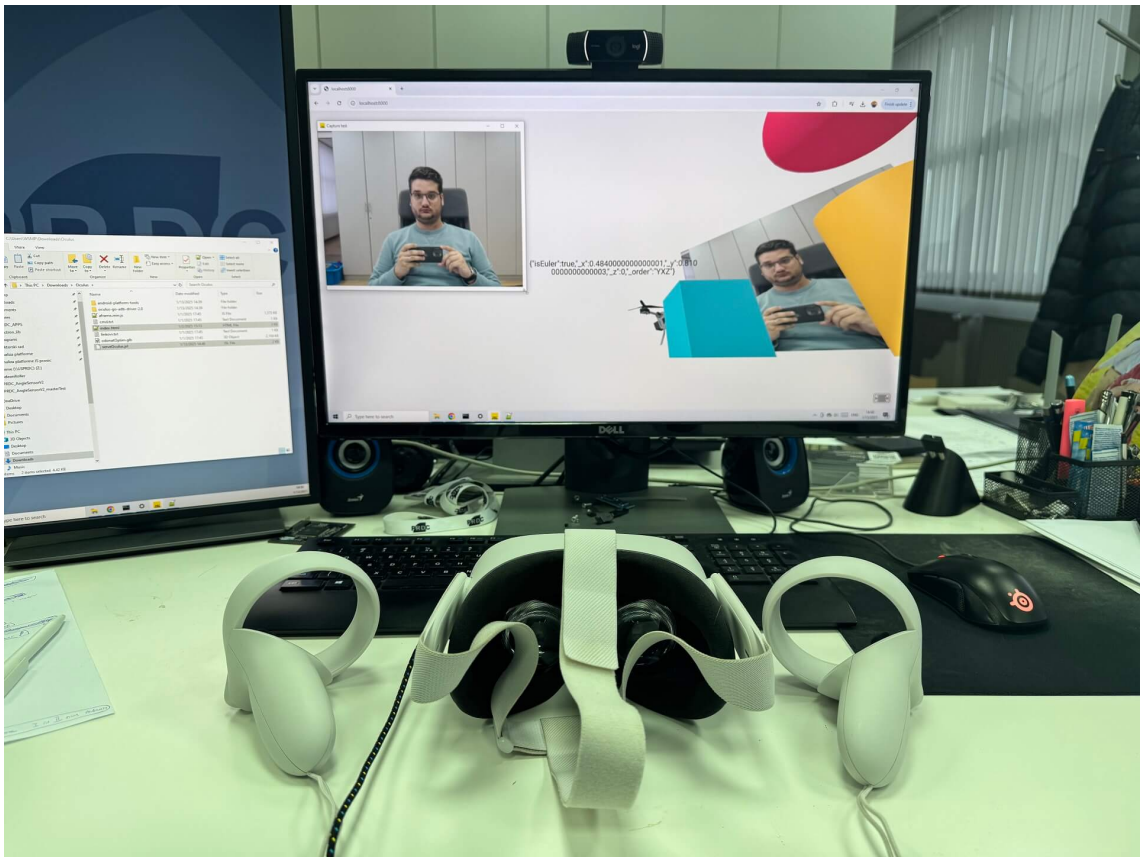
Slika 6.154 - Označeni delovi daljinskog upravljača IKA-CTRL

Delovi daljinskog upravljača i njihove oznake sa prethodne slike su:

1. Prekidač LT1
2. Prekidač LT2
3. Leva antena
4. Računar daljinskog upravljača
5. Desna antena
6. Taster za uključivanje i isključivanje računara daljinskog upravljača
7. Prekidač RT2
8. Prekidač RT1
9. Kontrolna palica RJ1
10. Indikator stanja daljinskog upravljača
11. Desna zakačka
12. Kontrolna palica RJ2
13. Taster za uključivanje i isključivanje daljinskog upravljača
14. Taster RS1
15. Indikatori stanja i baterije računara daljinskog upravljača
16. Taster RS2
17. Taster LS2
18. Taster LS1
19. Kontrolna palica LJ2
20. Leva zakačka
21. Kontrolna palica LJ1
22. Prekidač LT3

Često se postavlja pitanje zašto koristiti veliki ekran i tablet sa svim ostalim delovima daljinskog upravljača kada je moguće koristiti sisteme za virtuelnu realnost kao što je `Oculus Quest 2`. Odgovor na ovo pitanje je da je moguće koristiti i jedno i drugo, u zavisnosti od toga šta je potrebno pilotu i eventualno na šta je pilot navikao do tog momenta.

Sistem kao što je `Oculus Quest 2` je moguće povezati sa računarom daljinskog upravljača preko `USB`-a i pokrenuti skriptu kao što je `serveOculus.jsl` u `JSLAB`-u (ova skripta je data u prilogu AC). Na slici 6.155 je prikazan primer sa ovim sistemom za virtuelnu realnost, u trodimenzionalnom prostoru se prikazuje slika sa kamere u realnom vremenu što demonstrira da je na isti način moguće prikazivati i sliku sa kamere letelice.



Slika 6.155 - Primer prikaza za Oculus

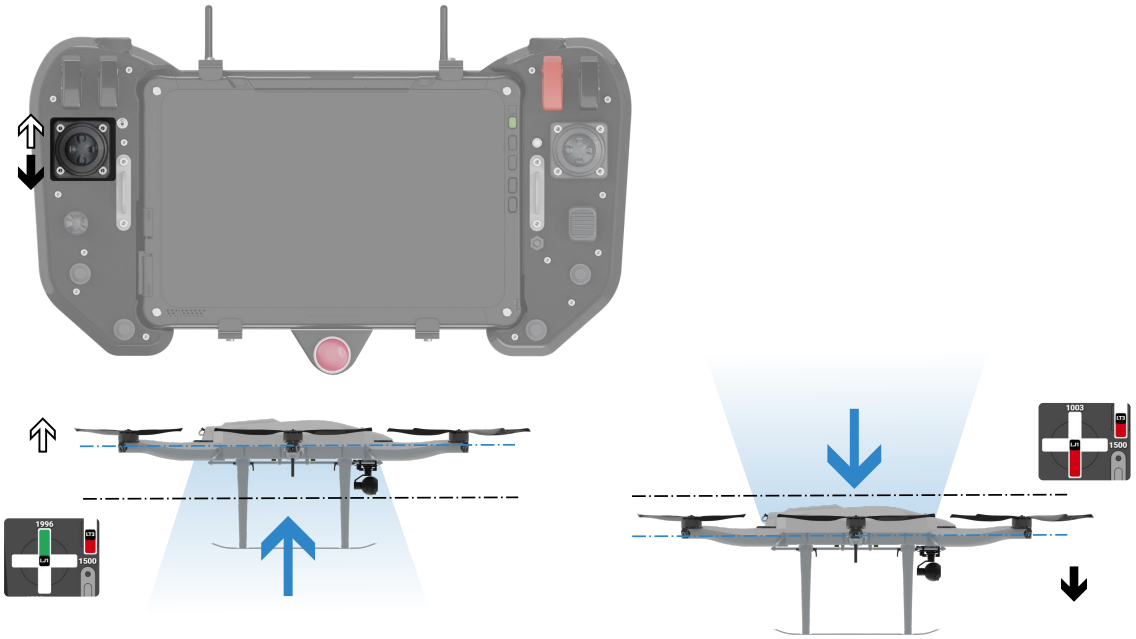
Slika sa kamere je prikazana u trodimenzionalnom prostoru zajedno sa drugim osnovnim elementima kao što su kocke, sfere i cilindri. Osim toga, kao test, učitan je i model multikoptera koji ima nešto komplikovaniju geometriju i materijale (model je u `GLB` formatu). Na sredini prozora se još vidi i da je očitana orijentacija naočara za virtuelnu realnost. Da bi se koristile ove naočare na platformi simulatora leta neophodno je da se prikaz vrši relativno u odnosu na platformu a ne u odnosu na apsolutnu orijentaciju (zato su uglovi orijentacije naočara, kao i orijentacije platforme neophodni).

Za ovaj primer je korišćena biblioteka `A-Frame` koja je bazirana na `JavaScript` biblioteci `Three.js` koja se već koristi i za druge svrhe u ovoj disertaciji. Biblioteka `A-Frame` omogućava jednostavno formiranje trodimenzionalne scene i dobijanje podataka o orijentaciji naočara, kao što se i vidi iz primera (u `index.html` fajlu). Za formiranje `HTTP` servera se koristi modul `http` koji je integrisan u `Node.js`, dok se za prenos slike koristi `Web API` pod nazivom Server-sent events. Da bi same naočare iz pregledača interneta pristupile ovom serveru na adresi `localhost:8000` potrebno ih je povezati sa računarom preko `USB`-a i izvršiti sledeću komandu:

```
1 adb reverse tcp:8000 tcp:8000
```

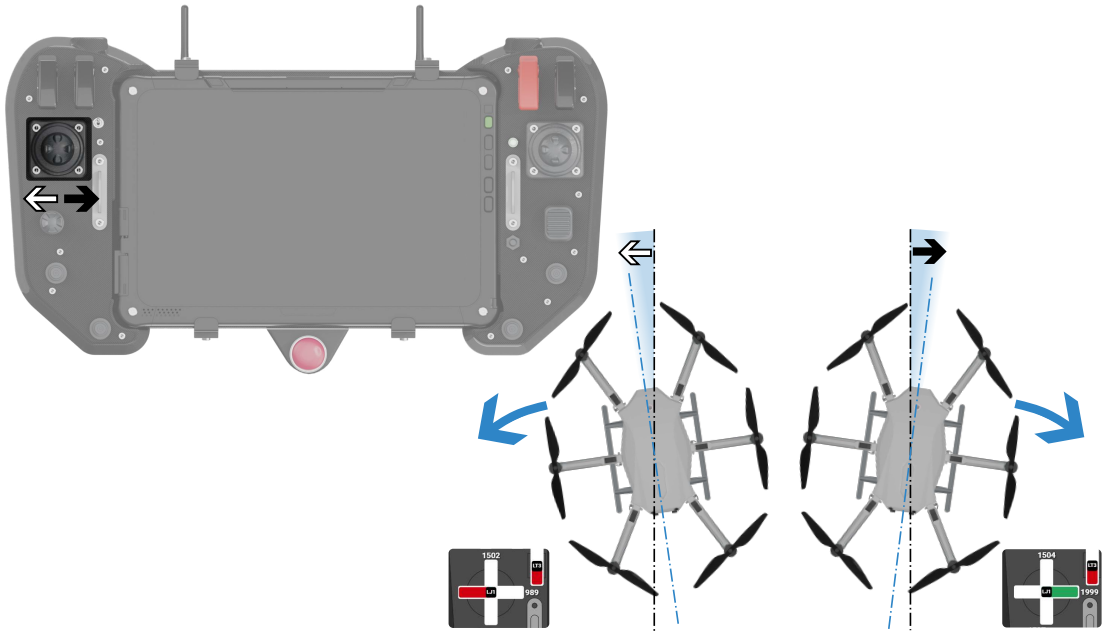
U nastavku je prikazano kako na primer letelica *IKA-BOMBARDER* reaguje na različite komande daljinskog upravljača `IKA-CTRL`.

- Na slici 6.156 je prikazano vertikalno pomeranje kontrolne palice `LJ1` - kanal za promenu visine (gas) - pomeranje letelice gore/dole (`kanal 1`, kontrolna palica označena na slici 6.154 brojem `21`).



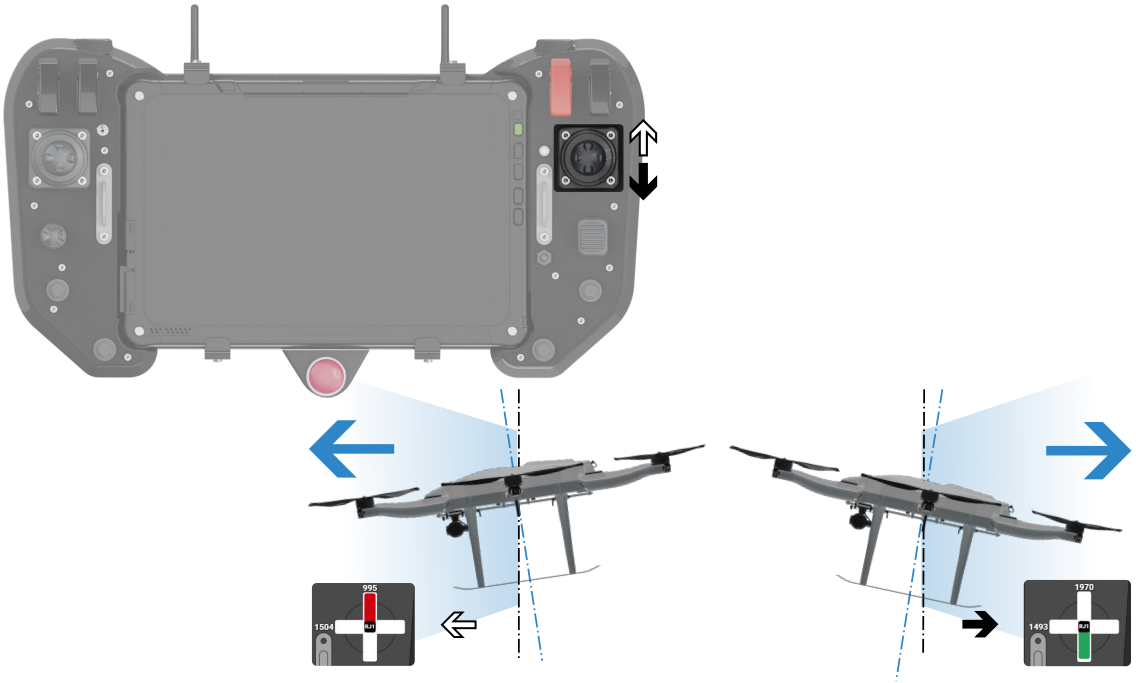
Slika 6.156 - Uticaj vertikalnog pomeranja kontrolne palice LJ1

- Na slici 6.157 je prikazano horizontalno pomeranje kontrolne palice `LJ1` - kanal za skretanje - promenu pravca letelice (`kanal 2`, kontrolna palica označena na slici 6.154 brojem `21`).



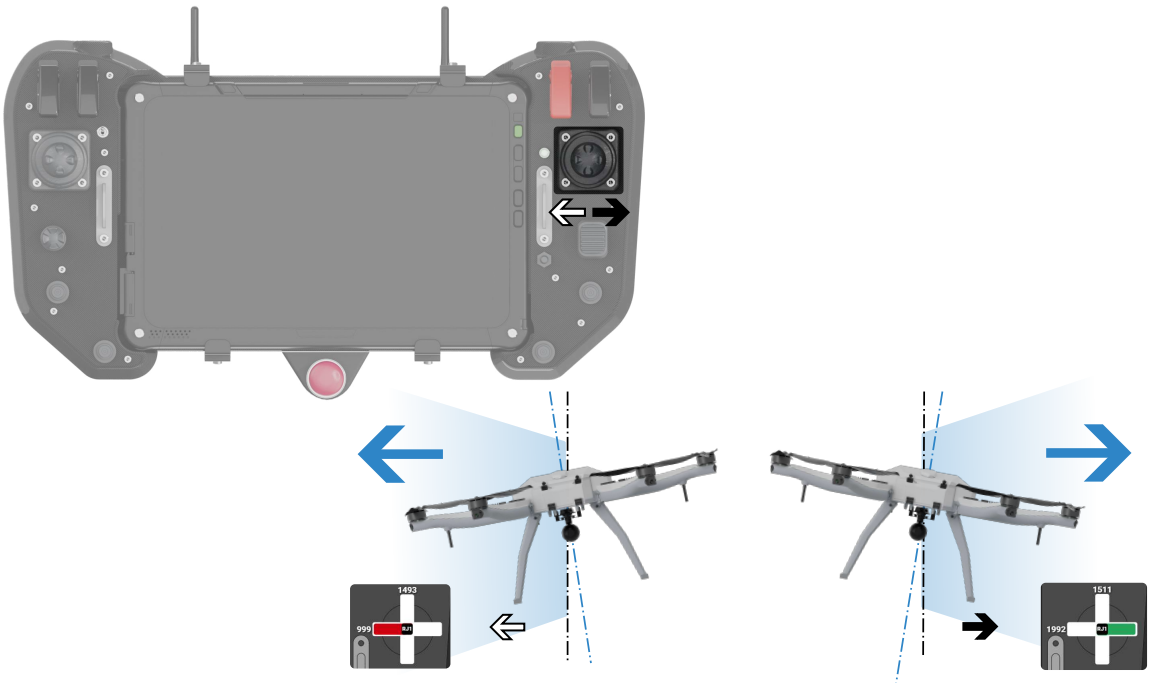
Slika 6.157 - Uticaj horizontalnog pomeranja kontrolne palice LJ1

- Na slici 6.158 je prikazano vertikalno pomeranje kontrolne palice `RJ1` - kanal za propinjanje - pomeranje letelice napred/nazad (`kanal 3`, kontrolna palica označena na slici 6.154 brojem `9`).



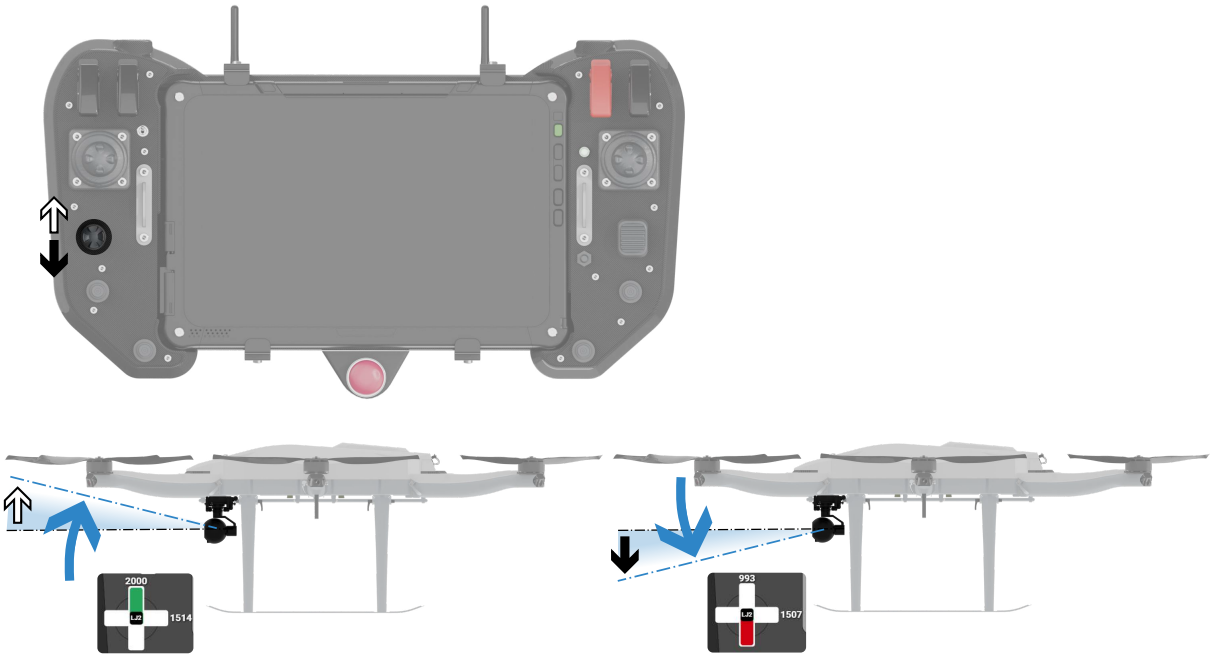
Slika 6.158 - Uticaj vertikalnog pomeranja kontrolne palice RJ1

- Na slici 6.159 je prikazano horizontalno pomeranje kontrolne palice `RJ1` - kanal za valjanje - pomeranje letelice levo/desno (`kanal 4`, kontrolna palica označena na slici 6.154 brojem `9`).



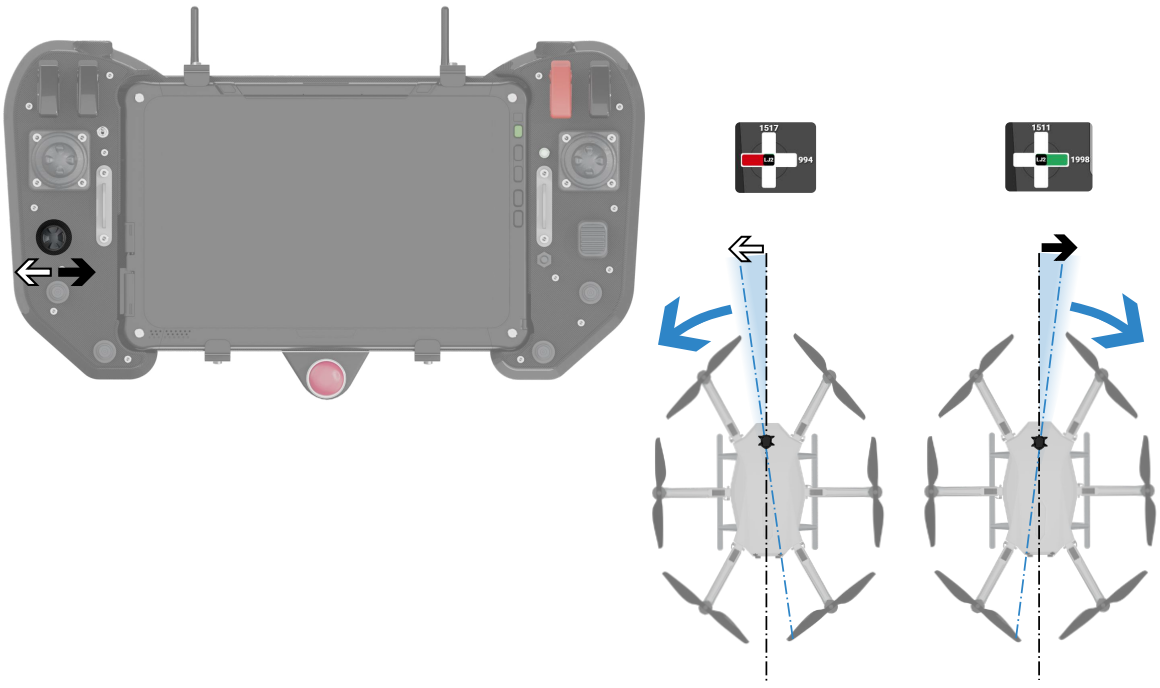
Slika 6.159 - Uticaj horizontalnog pomeranja kontrolne palice RJ1

- Na slici 6.160 je prikazano vertikalno pomeranje kontrolne palice `LJ2` - kanal za pomeranje kamere gore/dole (`kanal 5`, kontrolna palica označena na slici 6.154 brojem `19`).



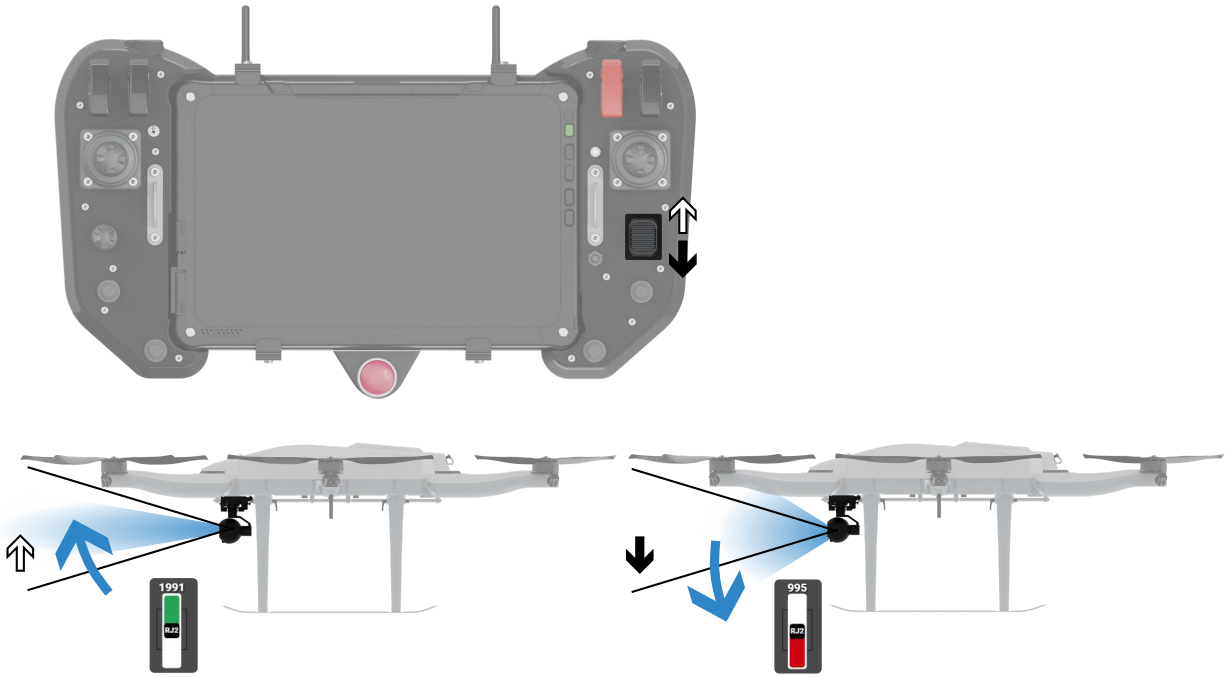
Slika 6.160 - Uticaj vertikalnog pomeranja kontrolne palice LJ2

- Na slici 6.161 je prikazano horizontalno pomeranje kontrolne palice `LJ2` - kanal za pomeranje kamere levo/desno (`kanal 6`, kontrolna palica označena na slici 6.154 brojem `19`).



Slika 6.161 - Uticaj horizontalnog pomeranja kontrolne palice LJ2

- Na slici 6.162 je prikazano vertikalno pomeranje kontrolne palice `RJ2` - kanal za promenu uvećanja kamere (zum, `kanal 7`, kontrolna palica označena na slici 6.154 brojem `12`).



Slika 6.162 - Uticaj vertikalnog pomeranja kontrolne palice RJ2

- Na slici 6.163 je prikazan prekidač `LT1` za promenu moda leta između pozicionog (prekidač u donjem položaju) i manuelnog (prekidač u gornjem položaju) (`kanal 8`, prekidač označen na slici 6.154 brojem `1`).



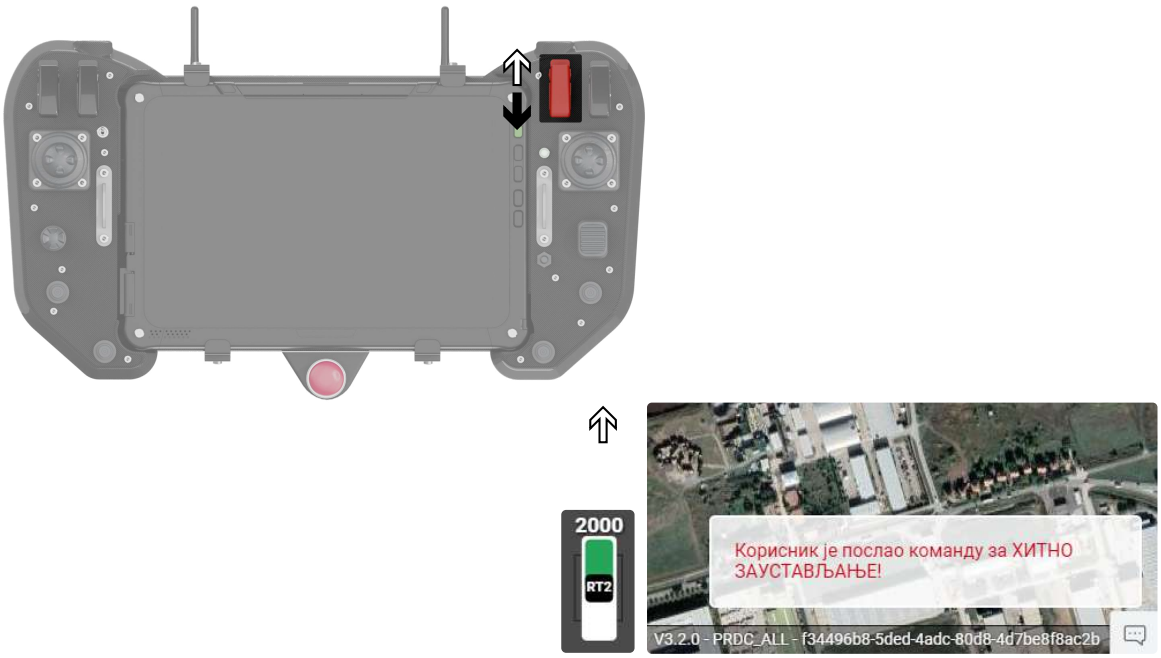
Slika 6.163 - Uticaj promene položaja prekidača LT1

- Na slici 6.164 je prikazan prekidač `LT2` za aktivaciju povratka na mesto poletanja (`kanal 9`, prekidač označen na slici 6.154 brojem `2`).



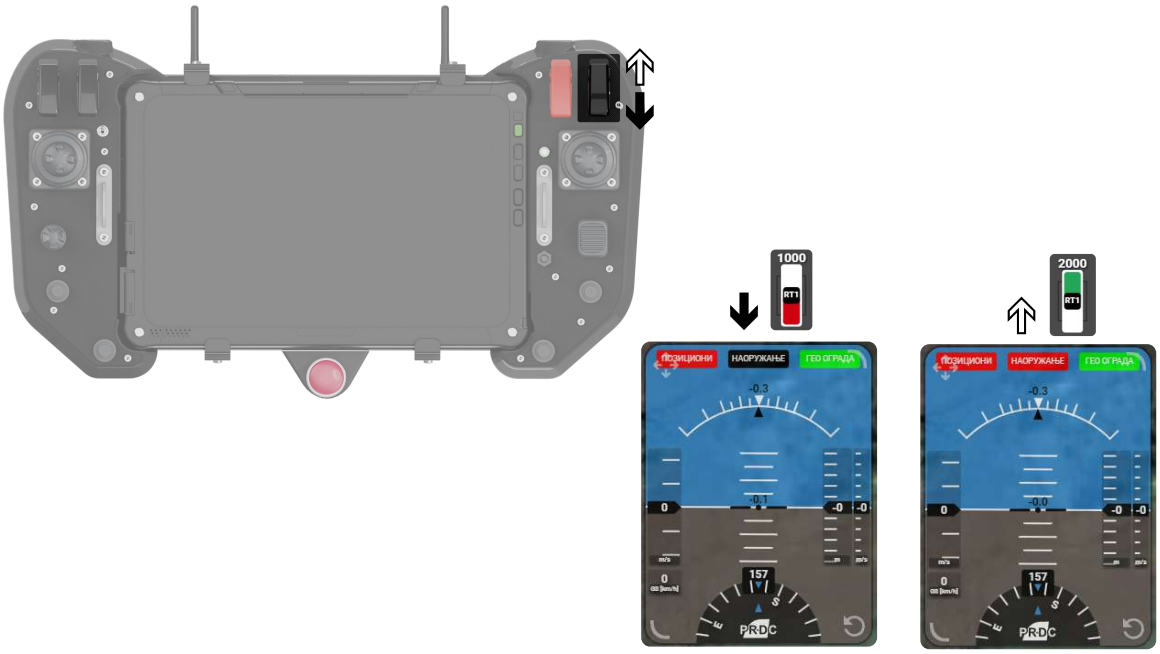
Slika 6.164 - Uticaj promene položaja prekidača LT2

- Na slici 6.165 je prikazan prekidač `RT2` za **hitno zaustavljanje motora** - gase se svi motori bez obzira na to da li je letelica u vazduhu ili ne (`kanal 10`, prekidač označen na slici 6.154 brojem 7).



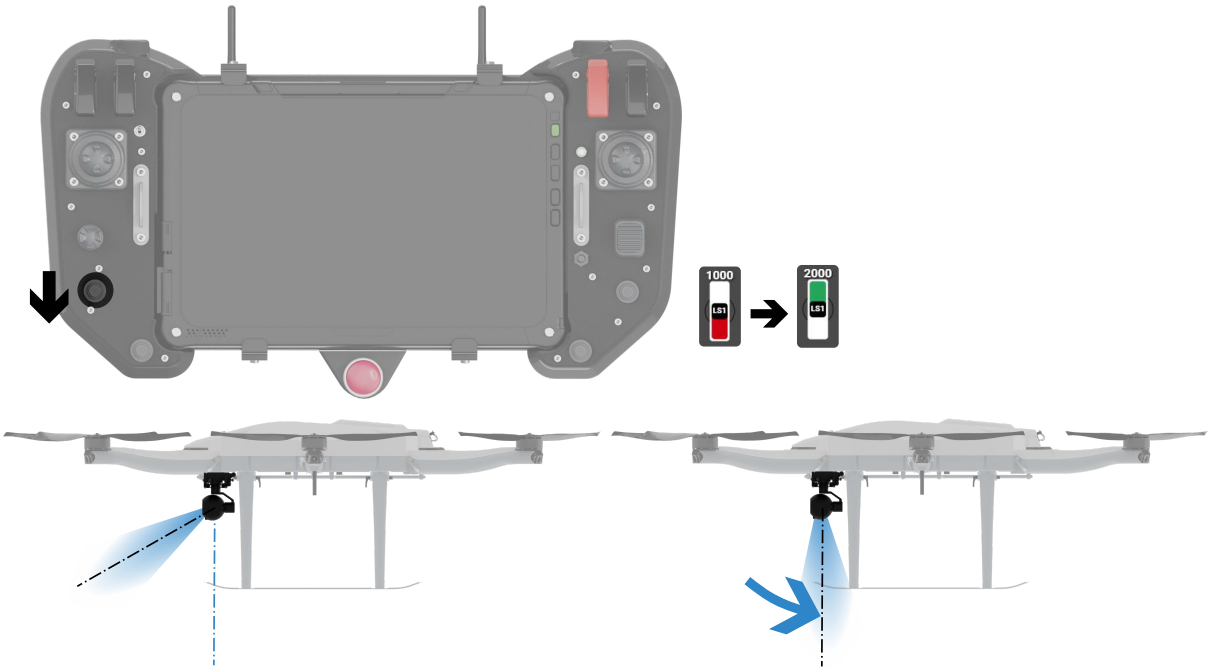
Slika 6.165 - Uticaj promene položaja prekidača RT2

- Na slici 6.166 je prikazan prekidač `RT1` za aktivaciju mehanizama za ispuštanje avio-bombi (`kanal 11`, prekidač označen na slici 6.154 brojem `8`).



Slika 6.166 - Uticaj promene položaja prekidača RT1

- Na slici 6.167 je prikazan taster `LS1` za prebacivanje kamere u mod za nišanjenje - pogled nadole (`kanal 12`, taster označen na slici 6.154 brojem `18`).



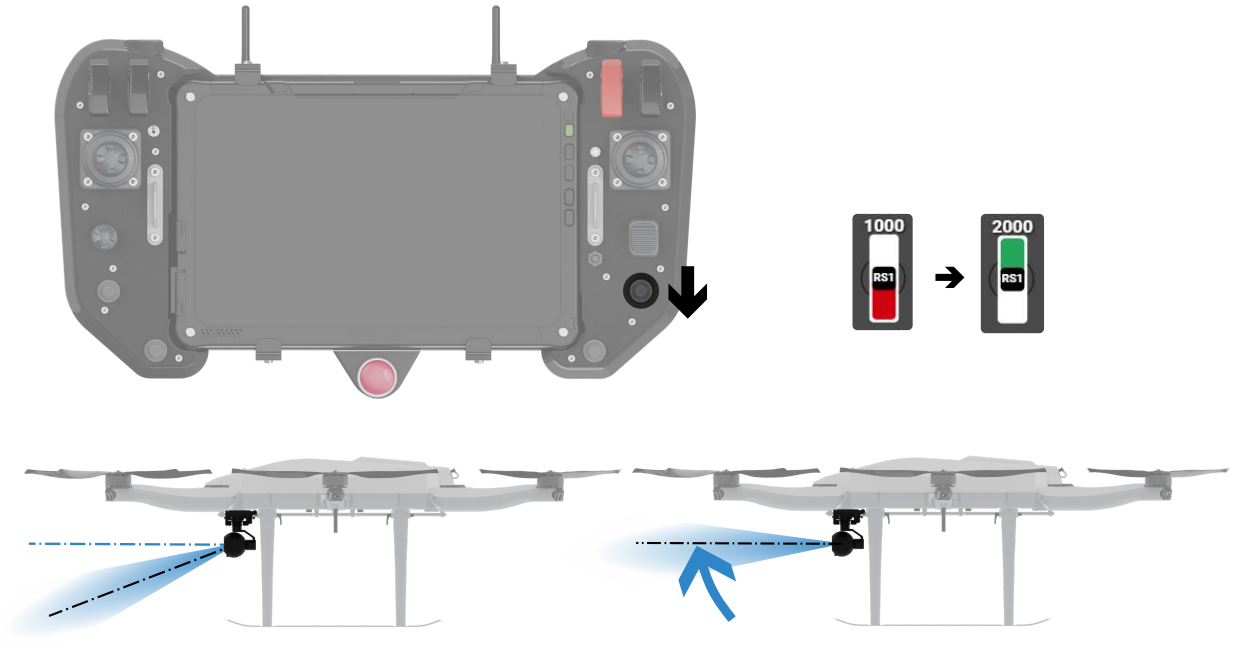
Slika 6.167 - Uticaj tastera LS1

- Na slici 6.168 je prikazan taster `LS2` za izbacivanje podešavanjima definisanog broja avio-bombi (`kanal 13`, taster označen na slici 6.154 brojem `17`).



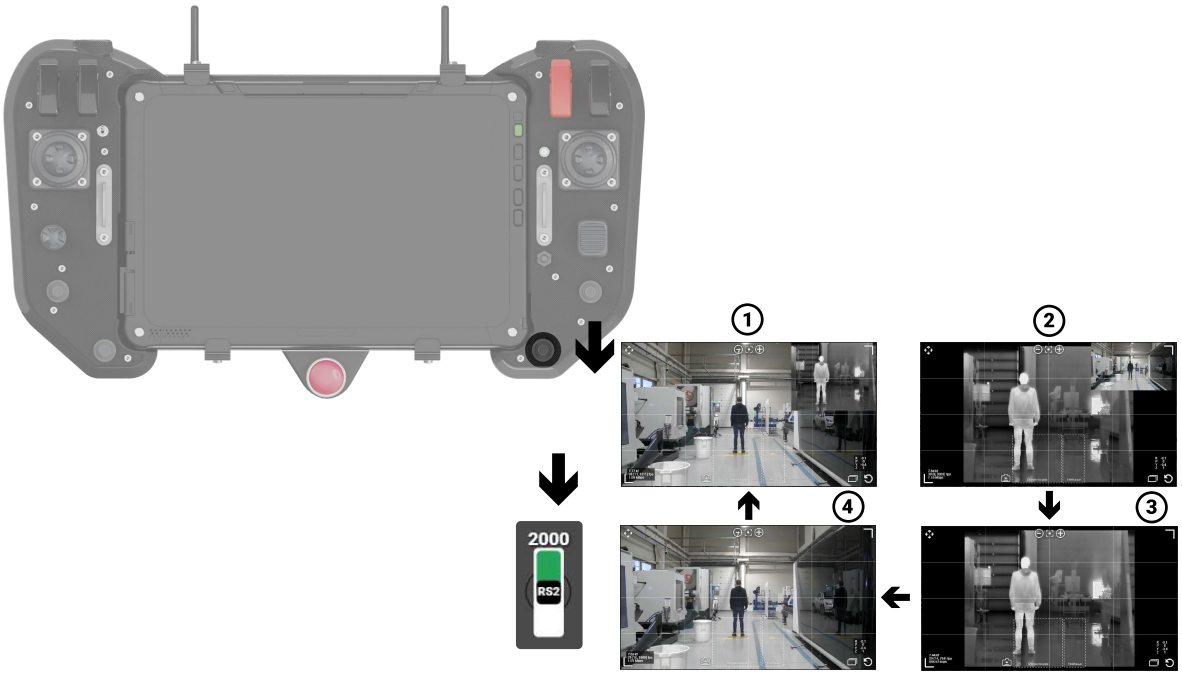
Slika 6.168 - Uticaj tastera LS2

- Na slici 6.169 je prikazan taster `RS1` za reset orijentacije kamere (`kanal 14`, taster označen na slici 6.154 brojem `14`).



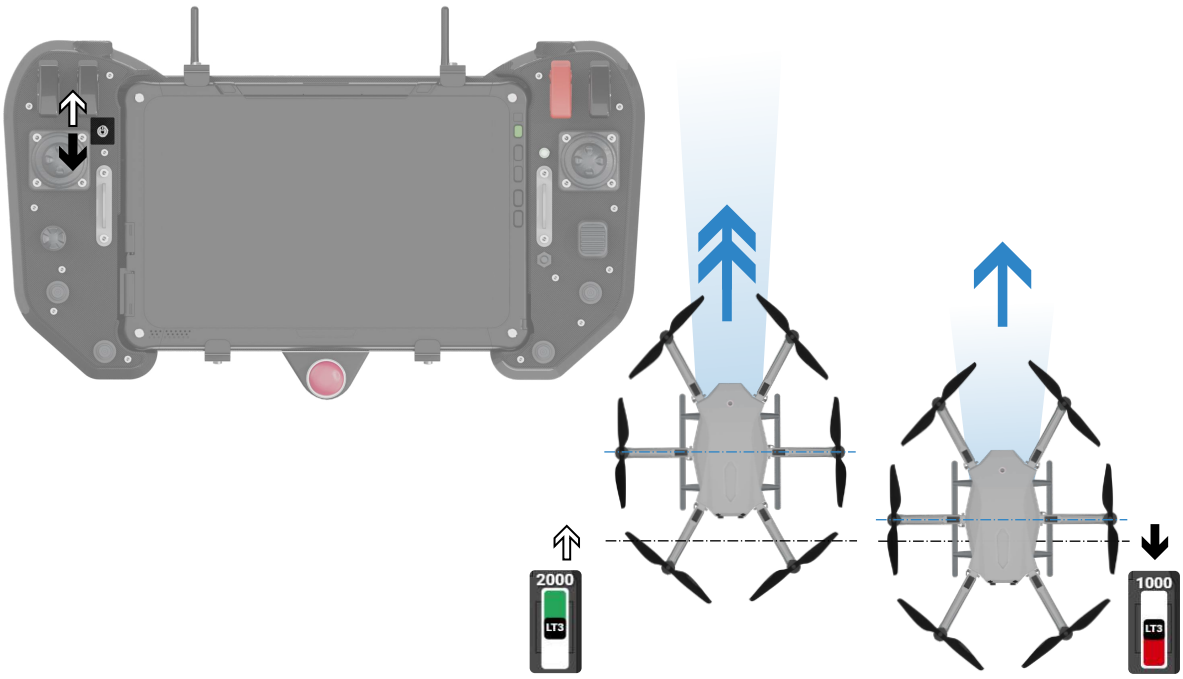
Slika 6.169 - Uticaj tastera RS1

- Na slici 6.170 je prikazan taster `RS2` za promenu prikaza kamera (`kanal 15`, taster označen na slici 6.154 brojem `16`).



Slika 6.170 - Uticaj promene položaja taster RS2

- Na slici 6.171 je prikazan prekidač `LT3` za promenu profila leta između sporog (prekidač u donjem položaju) i normalnog (prekidač u gornjem položaju) (`kanal 16`, prekidač označen na slici 6.154 brojem `22`).



Slika 6.171 - Uticaj promene položaja prekidača LT3

Pored prikazanih komandi, pilot komanduje još i preko tableta i njegovog ekrana osetljivog na dodir. Sa druge strane svi podaci koje pilot dobija mu se prezentuju preko ekrana tableta i velikog ekrana iznad tableta.

6.3.2 Prikaz telemetrijskih podataka

U sistemu bespilotne letelice kao što je `REMILP-01` se za dobijanje telemetrijskih podataka koristi komunikacioni protokol `MAVLink` kojim se zapravo dobijaju podaci iz autopilota sa programom *ArduPilot*. Na slici 6.172 se vidi prikaz telemetrijskih podataka sa letelice `REMILP-01` u programu `JSLAB`. Za ovaj primer se koristi skripta `ArdupilotLink.jsl` koja je data u prilogu Z.



Slika 6.172 - Prikaz telemetrijskih podataka sa letelice u `JSLAB`-u

U ovom primeru se koristi modul `node-mavlink` za `MAVLink`, koji je zvanični `MAVLink` modul za *Node.js*, formira se primarni ekran leta i prikazuju se lokacije i ugao skretanja letelice na 2D mapi sveta.

Neki od krucijalnih podataka koji se obično dobijaju preko telemetrije i prikazuju pilotu su:

- trenutni mod leta,
- aktivnost naoružanja (ako ga letelica ima),
- aktivnost geo-ograda,
- uglovi propinjanja i valjanja (veštački horizont),
- ugao skretanja (kompas),
- brzina kretanja u odnosu na zemlju,
- brzina kretanja u odnosu na vazduh,
- nadmorska visina leta,

- brzina penjanja,
- procenat napunjenosti baterija,
- temperatura baterija,
- broj dostupnih GNSS satelita,
- brzina i pravac vetra,
- pređeni put letelice,
- vreme trajanja leta,
- udaljenost od tačke poletanja,
- udaljenost od operatera,
- relativna visina leta (u odnosu na mesto poletanja),
- varijanse grešaka senzora,
- nivo vibracija na letelici i
- indikator jačine radio signala.

6.3.3 RTSP klijent

Prethodno je prikazano da kamere na letelicama, ali i virtuelne letelice implementiraju `RTSP servere` kako bi se strimovala slika, što znači da je sa druge strane na daljinskom upravljaču ili zemaljskoj stanici potrebno imati `RTSP klijent` kako bi se primao strim sa kamere. Ovo je jedan od krucijalnih elemenata sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

U prilogu AB je data skripta `ClientRTSP.jsl` kojom se implementira `RTSP klijent` u programu `JSLAB`. Ova skripta je i eksperimentalno ispitana kao što je prikazano u poglavlju 7.5. U ulaznim parametrima ove skripte se bira koji modul će da se koristi za prijem `RTP paketa`, koji modul će da se koristi za dekodiranje i na koji način će da se vrši prikaz.

Promenljiva `demuxer` može uzeti jednu od sledećih vrednosti:

- `js` - koristi modul `RTSP_DEMUXER.js` za uspostavljanje veze sa `RTSP` serverom (definisan u `RFC 7826`[179]) i dobijanje `RTP` paketa (definisanih u `RFC 3550`[180]). Ovo je modul implementiran od nule u programskom jeziku `JavaScript` sa ciljem da se paketi dekodiraju čim stignu bez ikakvog odlaganja kako bi se dobilo najmanje moguće kašnjenje. Drugi klijenti obično implementiraju određena kašnjenja kako bi se video prikazivao tečno. U ovom modulu je implementirana mogućnost korišćenja i `TCP`-a i `UDP`-a za `RTP` pakete. Iz `RTP` paketa izdvaja enkodirane `H264` i `H265` pakete koje prosleđuje na dekodiranje.
- `ffmpeg` - koristi modul `stream-decoder-ffmpeg` napisan u programskom jeziku `C++` koji koristi biblioteku `FFmpeg` za primanje i dekodiranje strima. Modul za svaki frejm vraća niz boja svakog piksela.
- `gst` - koristi modul `stream-decoder-gst` napisan u programskom jeziku `C++` koji koristi biblioteku `GStreamer` za primanje i dekodiranje strima. Modul za svaki frejm vraća niz boja svakog piksela.

Promenljiva `decoder` može uzeti jednu od sledećih vrednosti:

- `js` - koristi modul `DECODER_JS.js` za dekodiranje paketa pomoću `Web API`-a `VideoDecoder` (deo `WebCodecs`). Modul za svaki frejm vraća niz boja svakog piksela.
- `ffmpeg` - koristi modul `decoder-ffmpeg` napisan u programskom jeziku `C++` koji koristi biblioteku `FFmpeg` za dekodiranje paketa. Modul za svaki frejm vraća niz boja svakog piksela.

Promenljiva `container` može uzeti jednu od sledećih vrednosti:

- `canvas` - koristi metode `putImageData` i `drawImage` za prikaz slike, na osnovu boja svakog piksela, u okviru `HTML` elementa `canvas`.
- `video` - koristi `Web API` pod nazivom `VideoFrame` (deo `WebCodecs`) za generisanje frejma, na osnovu boja svakog piksela, koji se prikazuje u okviru `HTML` elementa `video` pomoću `MediaStream` i `MediaStreamTrackGenerator`.

Danas su sve više u upotrebi `FPGA` čipovi i procesori sa integrisanim blokovima za hardversko enkodiranje i dekodiranje videa kako bi se maksimalno smanjilo kašnjenje i opterećenje glavnog procesora. Ovakvi čipovi su sposobni za efikasnu i paralelizovanu obradu velikog broja piksel-podataka u realnom vremenu. Za pilotsku kabinu osim prijema slike sa kamere moguće je direktno, preko slike, dodavati određenu simboliku za pilota.

Od simbolike se najčešće dodaju podaci slični onim na primarnom ekranu leta, indikacija veštačkog horizonta (uglovi propinjanja i valjanja), ugao skretanja, brzina kretanja u odnosu na zemlju, brzina kretanja u odnosu na vazduh, nadmorska visina leta i brzina penjanja.

U `JSLAB`-u je veoma jednostavno dodavati simboliku preko slike sa kamere tako da se uopšte ne radi modifikacija slike već se element u kome se prikazuje slika preklapa sa elementom za simboliku kao što je `HTML` element `canvas` koji omogućava 2D i 3D crtanje.

U slučaju kada iz nekog razloga nije dostupan video strim a dostupni su telemetrijski podaci može se koristiti virtuelna letelica, kao što je prikazano u poglavlju 6.2.8, koja ima mogućnost generisanja video strima kako bi pilot ipak imao simuliranu okolinu kao referencu tokom upravljanja.

6.3.4 Aplikacija FlightControl

Veoma značajan deo sistema je aplikacija za upravljanje bespilotnom letelicom kao što je aplikacija `FlightControl`. Ovo je izuzetno moderna aplikacija iz perspektive korišćenih tehnologija i same arhitekture programa koja je modularna i omogućava brzu integraciju i implementaciju novih funkcionalnosti.

Jedna od najznačajnijih komparativnih prednosti ove aplikacije u odnosu na druge kao što su *Mission Planner* ili *QGroundControl* jeste savremeni grafički interfejs optimizovan za upravljanje letelicama kao što su one prikazane u ovoj disertaciji i činjenica da za korišćenje aplikacije nije potreban internet. Takođe, moguće je izabrati srpski jezik (latinicu ili ćirilicu) ili engleski jezik za grafički korisnički interfejs aplikacije.

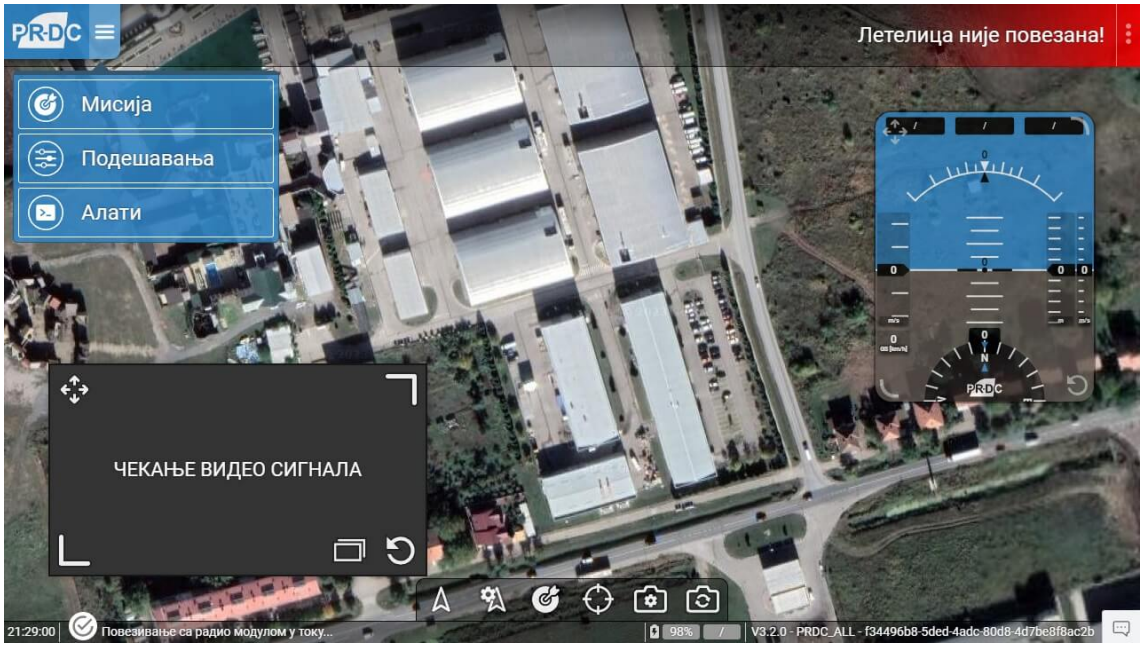
Neke od najznačajnijih funkcionalnosti aplikacije `FlightControl` su digitalni video-prenos u realnom vremenu, video-snimanje, telemetrija (podržava i `MAVLink` komunikacioni protokol), sekundarne komande, prikaz poruka upozorenja, izdavanje komande za automatski odlazak na cilj, programiranje misije za navigaciju od tačke do tačke, izdavanje komande za automatski povratak kući i automatsko sletanje, kao i slanje komande za kontrolu pozicionih svetala.

Nakon prvog pokretanja programa `FlightControl` otvara se tab za izbor letelice, kao što je prikazano na slici 6.173. Ovde je moguće dati određene nadimke svakoj letelici i identifikovati pojedinačne letelice, a izbor letelice se vrši prostim pritiskom na karticu te letelice. Nakon što je izabrana neka letelica, pri sledećem pokretanju programa se automatski bira ova letelica i direktno otvara tab za let. Statusna linija u donjem delu ekrana se prikazuje uvek i u njoj se vidi (sa leva na desno) trenutno vreme, status sistema, status punjača, procenat napunjenosti baterije računara i procenat napunjenosti baterije daljinskog upravljača, verzija programa, identifikacioni broj i dugme za otvaranje panela sa notifikacijama.



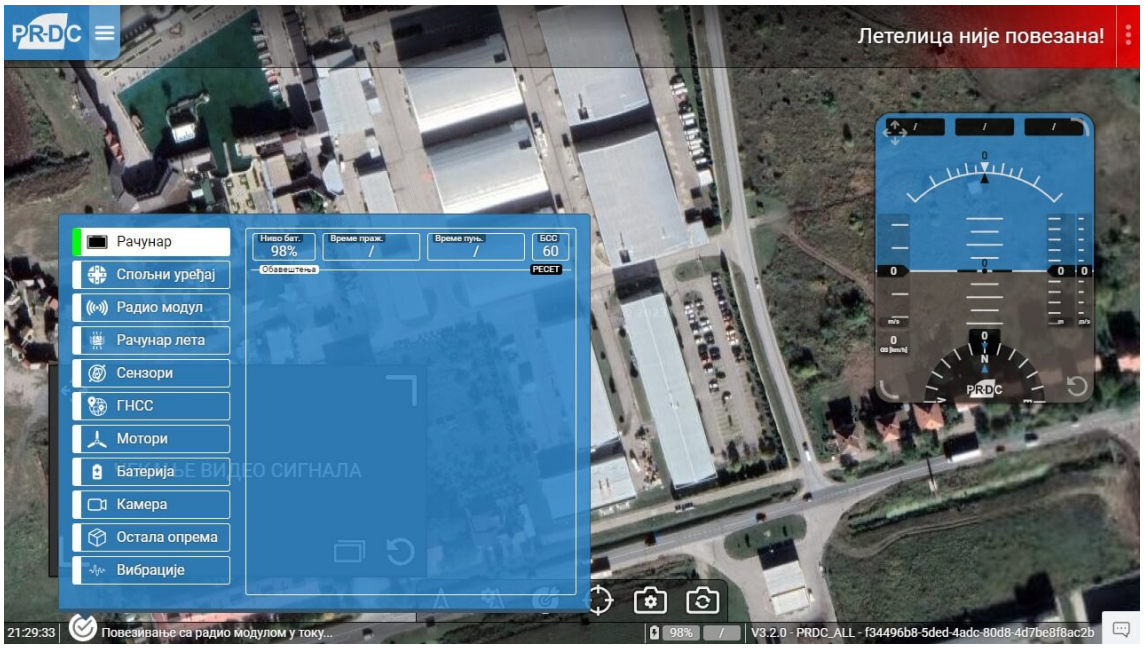
Slika 6.173 - Početni prozor aplikacije FlightControl za izbor letelice

Nakon izbora letelice se otvara tab za let (prikazan na slici 6.174). Pored ovog, postoje još i tab za misiju, podešavanja i alate.



Slika 6.174 - Aplikacija FlightControl u tabu za let

Pored različitih tabova, najznačajniji deo programa je prikaz statusa podsistema kao što je prikazano na slici 6.175. Aplikacija prikazuje sve relevantne podatke, ali i pojedinačne statuse (koji mogu biti da podsistem nije povezan, da se ne koristi, da ima upozorenje, da ima grešku ili da je normalan režim rada). Na osnovu pojedinačnih statusa se određuje ukupni status sistema i ukoliko je status zelene boje, to znači da je u normalnom režimu rada. Podsistemi su `Računar`, `Spoljni uređaj`, `Radio modul`, `Računar leta`, `Senzori`, `GNSS`, `Motori`, `Baterija`, `Kamera`, `Ostala oprema` i `Vibracije`. Poruke upozorenja, kao i sve druge poruke, se prikazuju u tabovima svakog pojedinačnog podsistema.



Slika 6.175 - Prikaz statusa podsistema u FlightControl aplikaciji

U okviru taba Let (kao što se vidi na slici 6.176) se prikazuje primarni ekran leta putem kojeg se korisnik, pored podataka o orijentaciji letelice u prostoru, brzini i visini leta, obaveštava i o trenutno aktivnom modu leta, aktivnosti naoružanja i geo-ograde.



Slika 6.176 - Prikaz slike sa kamere i instrumenata nakon povezivanja aplikacije FlightControl sa letelicom

Takođe, u ovom tabu se prikazuje slika sa kamere na letelici, sekundarne komande ukoliko se aktiviraju (komanduje se preko ekrana osetljivog na dodir), kao i položaj letelice i drugih značajnih tačaka na mapi sveta.

Na slici 6.177 je prikazan tab `Alati/Spoljni ulazi` u okviru koga je moguće videti vrednosti koje se šalju letelici putem određenih kanala daljinskog upravljača.



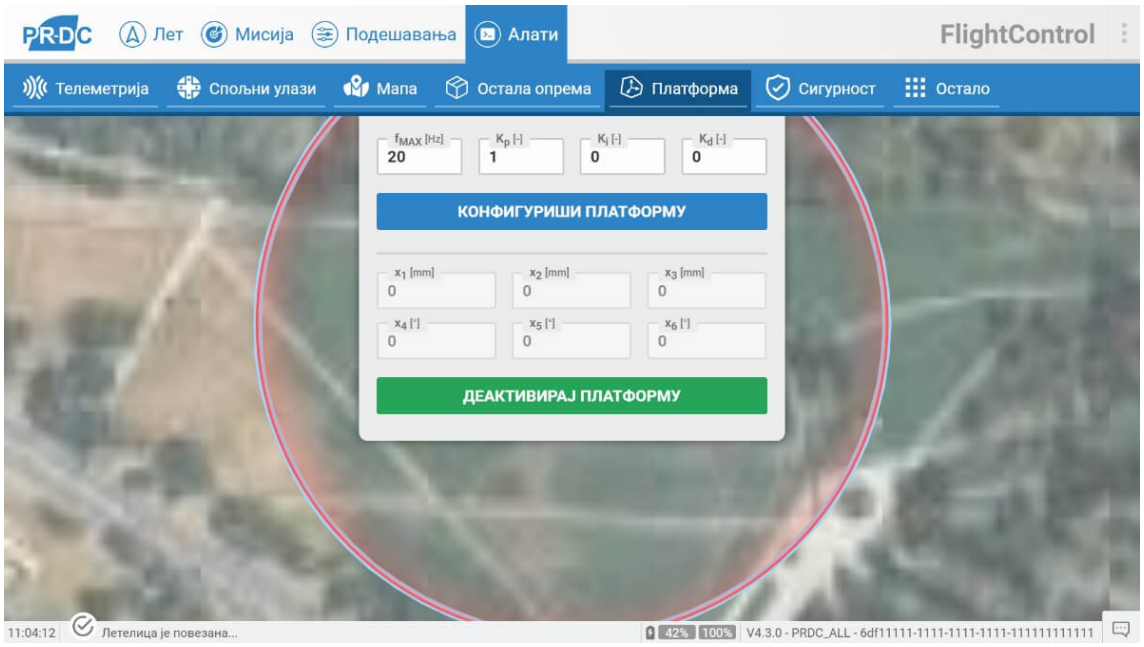
Slika 6.177 - Prikaz kanala daljinskog upravljača u aplikaciji FlightControl

Od ostalih alata je u aplikaciji, na primer, dostupno i slanje `RTCM 3` poruka za `RTK` korekcije za `GNSS` module, crtanje i merenje na mapi sveta, kontrolisanje učestanosti telemetrije ali i kontrolisanje opreme letelice i naoružanja. Značajan je i rad sa trenažerom na kom se pokreće aplikacija `VehicleSimulator`, iz taba `Alati/Ostalo` je moguće potpuno konfigurisati trenažer, resetovati misiju ili čak kompletnu aplikaciju.

Neke od implementiranih sigurnosnih opcija u aplikaciji i na daljinskom upravljaču su i to da nije moguće isključiti daljinski upravljač sve dok je aplikacija uključena (što eliminiše opasnost od slučajnog pritiska prekidača), opcije za isključivanje programa ili promenu veličine prozora su izmeštene i potrebno je prvo otvoriti padajući meni u kome je onda moguće izabrati željenu komandu, program sve vreme vrši provere veze sa radio modulom i sa spoljašnjim uređajem (procesorom daljinskog upravljača), u slučaju gubitka veze sa nekim podsistemom (npr. sa kamerom) aplikacija automatski pokušava da ponovo uspostavi vezu bez potrebe za intervenisanjem od strane korisnika.

Ova aplikacija takođe ima podsistem za kontrolu simulatora leta (u prikazu statusa podsistema se ovaj podsistem zove `Platforma`) što konačno kompletira sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

Na slici 6.178 je prikazan tab `Alati/Platforma` u okviru koga se vrše podešavanja platforme i u okviru koga je moguće aktivirati ili deaktivirati samu platformu i videti koji su to spoljašnji parametri koji se trenutno šalju platformi. Tek kada je platforma aktivirana se šalju podaci platformi na osnovu kretanja letelice. Od podešavanja se mogu uneti parametri za aktuatore pa se tako ovde podešavaju maksimalna ugaona brzina elektromotora preko maksimalne učestanosti frekvencije (od `0 Hz` do `50 Hz`) i koeficijenti `PID` regulatora. Ovaj tab se koristi samo na početku rada, dok se za praćenje podsistema koristi isti prikaz statusa kao i za sve ostale podsisteme u ovoj aplikaciji.



Slika 6.178 - Podešavanje podsistema platforme u FlightControl aplikaciji

Na slici 6.179 se vidi prikaz statusa podsistema `Platforma`. Ovde se može videti trenutni status kompletnog mehanizma ali i svakog pojedinačnog aktuatora.



Slika 6.179 - Prikaz statusa podsistema platforme u FlightControl aplikaciji

6.3.5 Električna promena antena

Tokom razvoja ovog sistema analizirano je kako omogućiti što veći domet bežične komunikacije i jedan od zaključaka jeste da je izbor antene izuzetno bitan i da zapravo antene treba birati u skladu sa planom leta odnosno misijom. Zbog toga je napravljen uređaj koji je nazvan `Switch Box` (prikazan na slici 6.180) koji ima zadatak da izvrši električnu promenu antena koje se koriste.

Uređaj je baziran na elektronici sa dva radio-frekvencijska prekidača, odnosno dva jednopolna dvostruka prekidača za radio-frekvencijske signale određene učestanosti i snage.

Inicijalno je jedan prekidač korišćen za dvosmernu bežičnu komunikaciju sa letelicom (telemetriju) dok je drugi prekidač korišćen za analogni audio i video signal. Prelaskom na digitalni radio modul, svaki prekidač se koristi za promenu po jedne antene `MIMO` sistema. Osnovni cilj ovog podsistema je omogućiti promenu između direkcionih i omnidirekcionih antena u zavisnosti od potrebe.



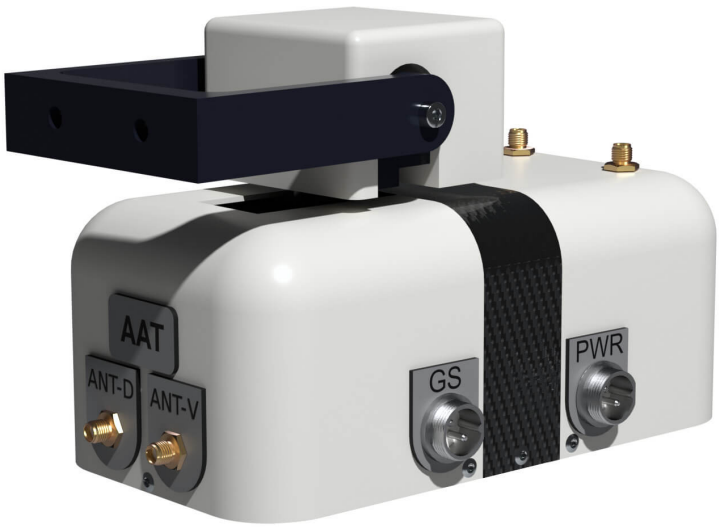
Slika 6.180 - Uređaj Switch Box

Na kućištu uređaja su konektori za antenske kablove kao i za napajanje i komunikaciju sa ostatkom sistema preko `Pun-dupleks RS485`.

6.3.6 Sistem za automatsko usmeravanje antena

Kao i u slučaju uređaja za električnu promenu antena, jedan od zaključaka analize kako omogućiti što veći domet bežične komunikacije jeste da je moguće koristiti direkcione antene usmerene prema letelici. Zbog toga je napravljen uređaj koji je nazvan `Automatic Antenna Tracker` (`AAT`, prikazan na slici 6.181) koji ima zadatak da na odgovarajući način orijentiše antene.

Konstrukcijski ovaj podsistem predstavlja robotski mehanizam sa dva stepena slobode. Pomoću ova dva stepena slobode je moguće uperiti direkcionu antenu prema letelici gde god da se ona nalazila. Na ovaj način se značajno povećava domet bežične komunikacije i video-prenosa i ujedno se smanjuje mogućnost interferencije i ometanja.



Slika 6.181 - Uređaj Automatic Antenna Tracker

Elektronika uređaja je zadužena za upravljanje oba aktuatora na osnovu informacija o uglovima koji se mere beskontaktno senzorima kao što su prikazani senzori simulatora leta (*Angle Sensor*). `AAT` takođe koristi `Pun-dupleks RS485` za komunikaciju sa ostatkom sistema. Na kućištu uređaja su konektori za antenske kablove kao i za napajanje i komunikaciju sa ostatkom sistema. Na drugom segmentu robota se nalazi univerzalni nosač antena koji omogućava jednostavnu promenu antena.

Elektronika ovog podsistema uključuje i `GNSS` modul (u modu stacionarne baze) na osnovu kojeg se određuje pozicija ali i omogućava preciznije pozicioniranje letelice na osnovu `GNSS RTK` sistema. U slučaju gubljenja komunikacije i kada je nije moguće povratiti tada nije poznata pozicija letelice, eventualno se može preći na omnidirekcione antene korišćenjem uređaja `Switch Box`.

Za fiksiranje uređaja `Automatic Antenna Tracker` i `Switch Box` se koristi modularni stativ. Ovaj stativ se jednostavno sklapa iz elemenata i na njega se montiraju uređaji kao što se vidi na slici 6.182.



Slika 6.182 - Montiranje stativa i AAT uređaja

7

Numerička i eksperimentalna ispitivanja elemenata i celog sistema

*"Nismo ograničeni samo tehnologijom, već i našim idejama i našim eksperimentisanjem i time koliko brzo možemo da **isprobamo stvari**."*

– Tim Sweeney[1]

Da bi se razvio i kasnije verifikovao sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi postepeno su tokom višegodišnjeg istraživanja vršena različita numerička i eksperimentalna ispitivanja. Ispitivani su najpre elementi sistema, a kasnije i kompletan sistem. Kao što je prikazano u radu, prvo su rađena ispitivanja na umanjenom prototipu sistema, a nakon toga i na prototipu sistema u stvarnoj veličini.

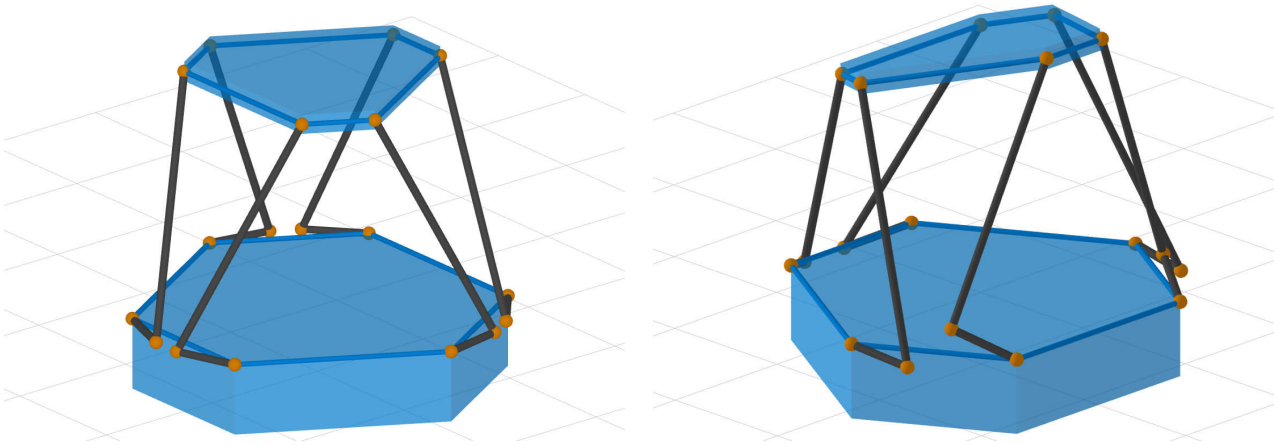
7.1 Numeričke simulacije mehanizma simulatora leta

7.1.1 Vizualizacija simulatora leta pomoću Three.js biblioteke u JSLAB-u

Pored fizičkih eksperimenata, za istraživanje i verifikaciju, veoma je korisno raditi i kompjuterske simulacije. U radu [44] se za vizualizaciju radnog prostora koristi softver za kompjutersko projektovanje (`CAD`), dok su u radovima [45], [46], [181] predstavljena simulaciona okruženja za ovu vrstu mehanizma. U podpoglavlju 4.9.4 je prikazano određivanje radnog prostora i njegov 3D prikaz pomoću `JSLAB` programa.

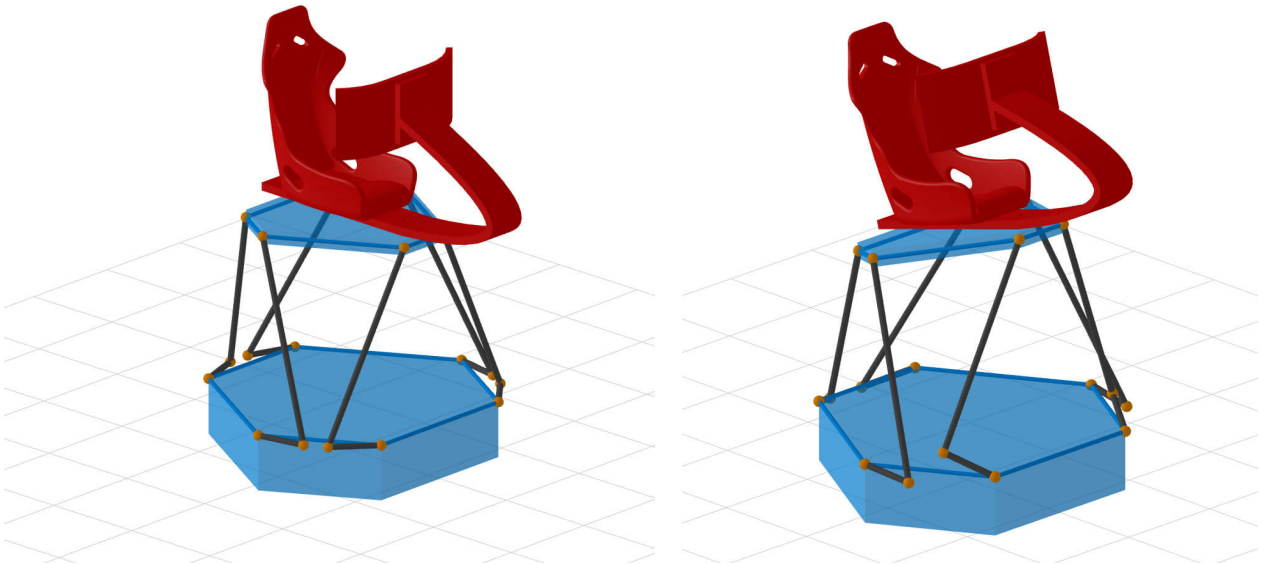
Skriptom `showModel3D.jsl`, koja je u celosti data u listingu AD.1 u prilogu AD, vrši se vizualizacija simulatora leta pomoću `Three.js` biblioteke, kao što je prikazano na slici 7.1.

[1] Tim Sweeney (21. decembar 1970) je američki programer video-igara i biznismen. Osnivač je i izvršni direktor *Epic Gamesa* i kreator *Unreal Engine*-a, platforme za razvoj igara.



Slika 7.1 - Vizualizacija simulatora leta pomoću `Three.js` biblioteke u `JSLAB`-u

U odnosu na klasičan prikaz trodimenzionalnog grafika, prikazana implementacija omogućava integraciju proceduralno generisanog 3D modela mehanizma simulatora leta sa drugim 3D modelima (kao što su oni u `GLTF` ili `GLB` formatu). Cilindri i sfere koje definišu geometriju mehanizma su generisane na osnovu ulaznih parametara skripte. Izmenom ovih parametara se menja i geometrija mehanizma na odgovarajući način. Biblioteka `Three.js` je 3D biblioteka opšte namene bazirana na `WebGL` i `WebGPU` tehnologijama što omogućava potpunu slobodu pri kreiranju naprednih vizualizacija [182]. Na slici 7.2 je prikazan primer integracije modela mehanizma sa modelom pilotske kabine koji je prethodno formiran u `FreeCAD`-u i `Blender`-u.



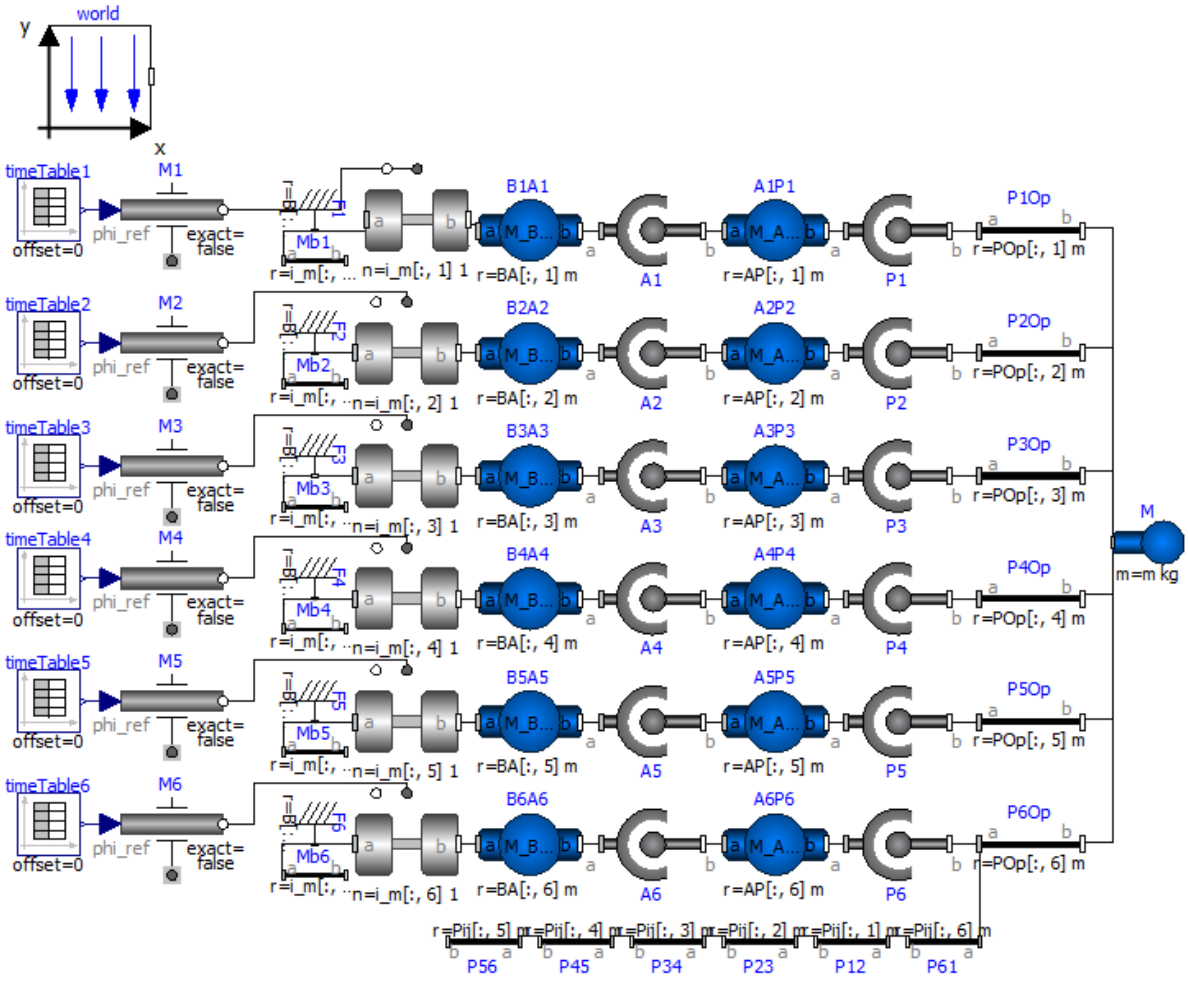
Slika 7.2 - Vizualizacija simulatora leta sa pilotskom kabinom pomoću Three.js biblioteke u JSLAB-u

Promenom spoljašnjih koordinata u skripti može se kreirati animacija kretanja i može se napraviti video koji može poslužiti prilikom analize kretanja mehanizma. U okviru skripte je pozvan i dodatak koji omogućava da se korisnik kreće kroz 3D scenu, rotira i uvećava ili umanjuje scenu.

7.1.2 Određivanje potrebnog obrtnog momenta pomoću programa OpenModelica

Osim svih prethodno prikazanih napisanih programa za analizu i urađenih eksperimenata na funkcionalnim prototipovima, urađena je i simulacija mehanizma simulatora leta u programskom paketu `OpenModelica`. Ovo je programski paket otvorenog koda koji se bazira na `Modelica` jeziku za precizno i fleksibilno modeliranje kompleksnih fizičkih sistema. Funkcioniše na osnovu povezivanja grafičkih blokova koji se mogu praviti, prilagođavati ili je moguće koristiti gotove blokove iz biblioteke.

Formiran je model `modelDinamikaParametarski.mo` za rešavanje inverzne statike mehanizma simulatora leta. Iskorišćena je biblioteka blokova `Modelica.Mechanics.MultiBody`[183] koja sadrži sve neophodne blokove za definisanje, analizu i vizualizaciju sistema tela [184]. Korišćenjem ove biblioteke, fizički sistemi se definišu kroz blokove poput fiksnih delova, obrtnih, sfernih i drugih zglobova, kao i pokretnih tela koja predstavljaju različite delove simulatora leta. Parametri blokova definišu geometriju, masu i moment inercije svih komponenti, osiguravajući tačnost proračuna i simulacija.



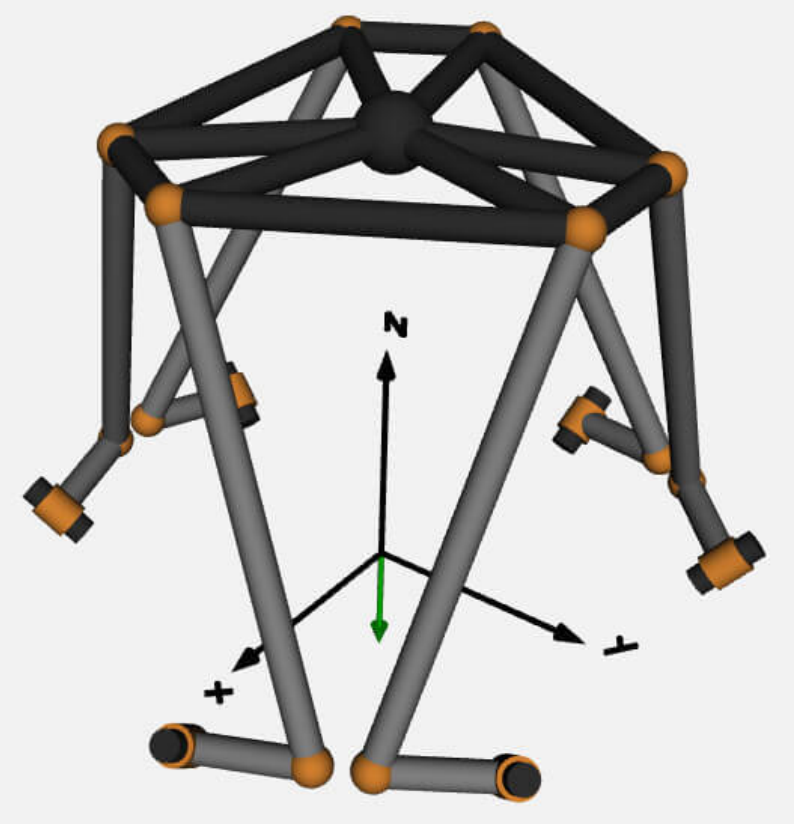
Slika 7.3 - OpenModelica model mehanizma

Na slici 7.3 je prikazan kompletan model mehanizma simulatora leta u programskom paketu `OpenModelica Connection Editor` (ovo je samo grafički prikaz, dok je model sačuvan u tekstualnom dokumentu sa ekstenzijom `.mo`). Ovo je takođe parametarski model, čime je omogućeno menjanje geometrije mehanizma koji se analizira. Mehanizam je uspešno modeliran u potpunosti u skladu sa svim prethodno prikazanim analizama.

Parametri modela uključuju uglove donjih poluga α, masu tereta i platforme (m), koordinate težišta tereta (cg), početni položaj platforme ($Op0$), kao i položaje tačaka A_i, B_i i P_i. Jedinični vektori osa aktuatora (i_m) definišu osu rotacije svakog od šest aktuatora, omogućavajući precizno upravljanje kretanjem platforme simulatora. Dimenzije i mase različitih delova mehanizma, kao što su dužine i prečnici cilindara, takođe su definisane kroz parametre modela.

Struktura modela uključuje globalni koordinatni sistem (`world`) sa definisanim gravitacionim ubrzanjem. Fiksni delovi na bazi simulatora su povezani sa obrtnim zglobovima u tačkama B_i (od `B1` do `B6`), koji omogućavaju rotaciju oko definisanih osa. Obrtni zglobovi su dalje povezani sa sfernim zglobovima u tačkama A_i i P_i (od `A1` do `A6` i od `P1` do `P6`), preko donjih (od `B1A1` do `B6A6`) i gornjih poluga (od `A1P1` do `A6P6`) definisanih kao cilindrična tela. Centralna platforma (`M`) je definisana kao telo sa definisanim težištem i masom, povezana sa svim gornjim polugama preko sfernih zglobova.

Na slici 7.4 je prikazana vizualizacija mehanizma u okviru istog programa. Program ima ugrađenu sposobnost i za detaljne trodimenzionalne animacije kretanja mehanizma i grafički prikaz promene parametara. U radu [185] je prikazana slična simulacija samo za *Strjuartovu platformu sa linearnim aktuatorima* i za *Delta robot*.



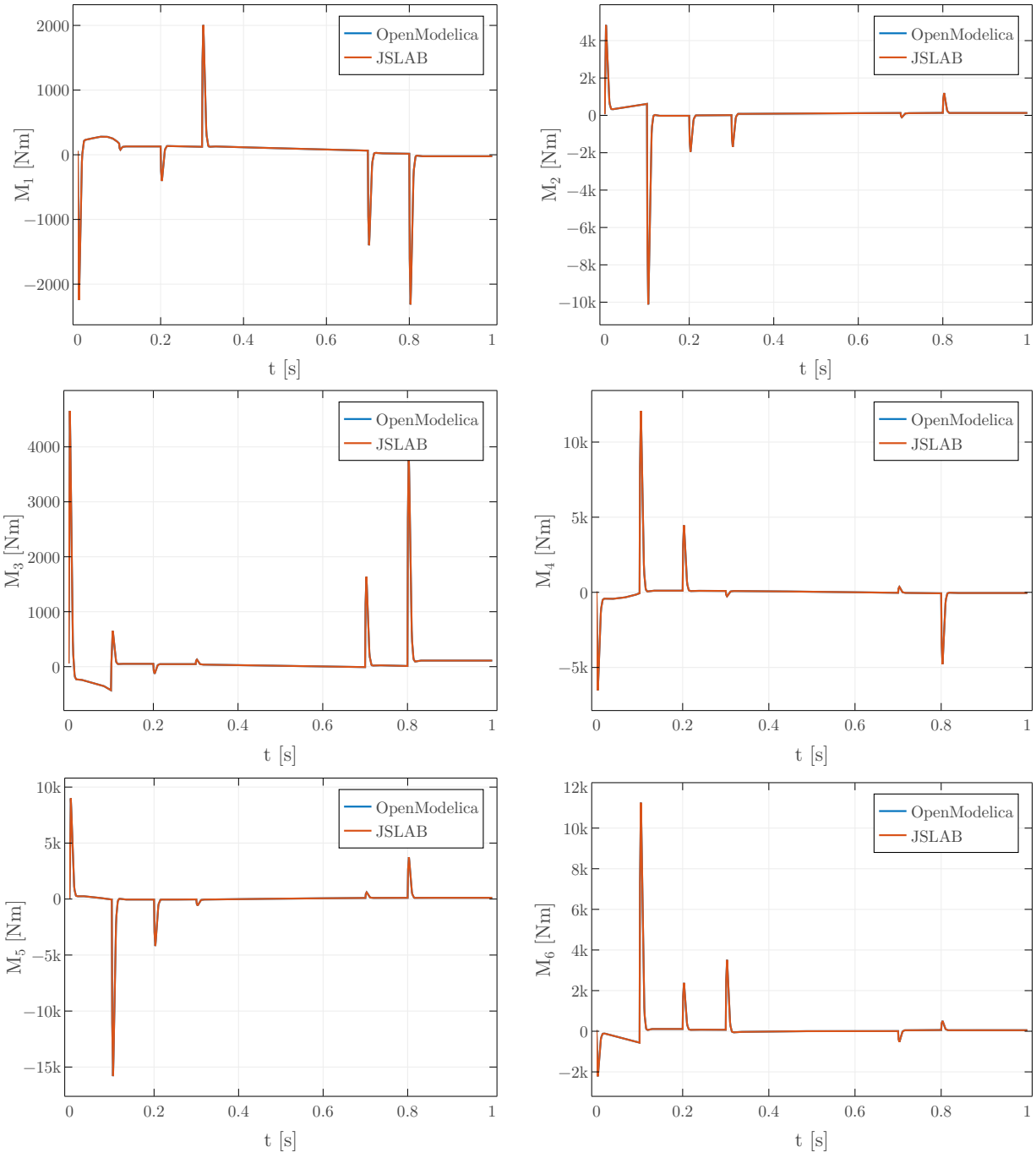
Slika 7.4 - Vizualizacija OpenModelica modela mehanizma

U prilogu AE je dat program `solveInverseDynamicsOpenModelica.jsl` u okviru koga se poziva klasa `PRDC_JSLAB_OPENMODELICA_LINK` koja je implementirana za potrebe ovog rada kako bi se omogućila komunikacija između programskih paketa `JSLAB` i `OpenModelica`. Kada je omogućena ova komunikacija omogućena je i jednostavna uporedna analiza rezultata.

Da bi se omogućilo menjanje parametara mehanizma pre kompajliranja modela, napravljen je šablon modela i iz programskog paketa `JSLAB` direktno se menjaju vrednosti parametara u tekstualnom dokumentu koji definiše model. Uglove otklona donjih poluga je moguće menjati i nakon kompajliranja tako što se prosleđuju kao parametri simulacije.

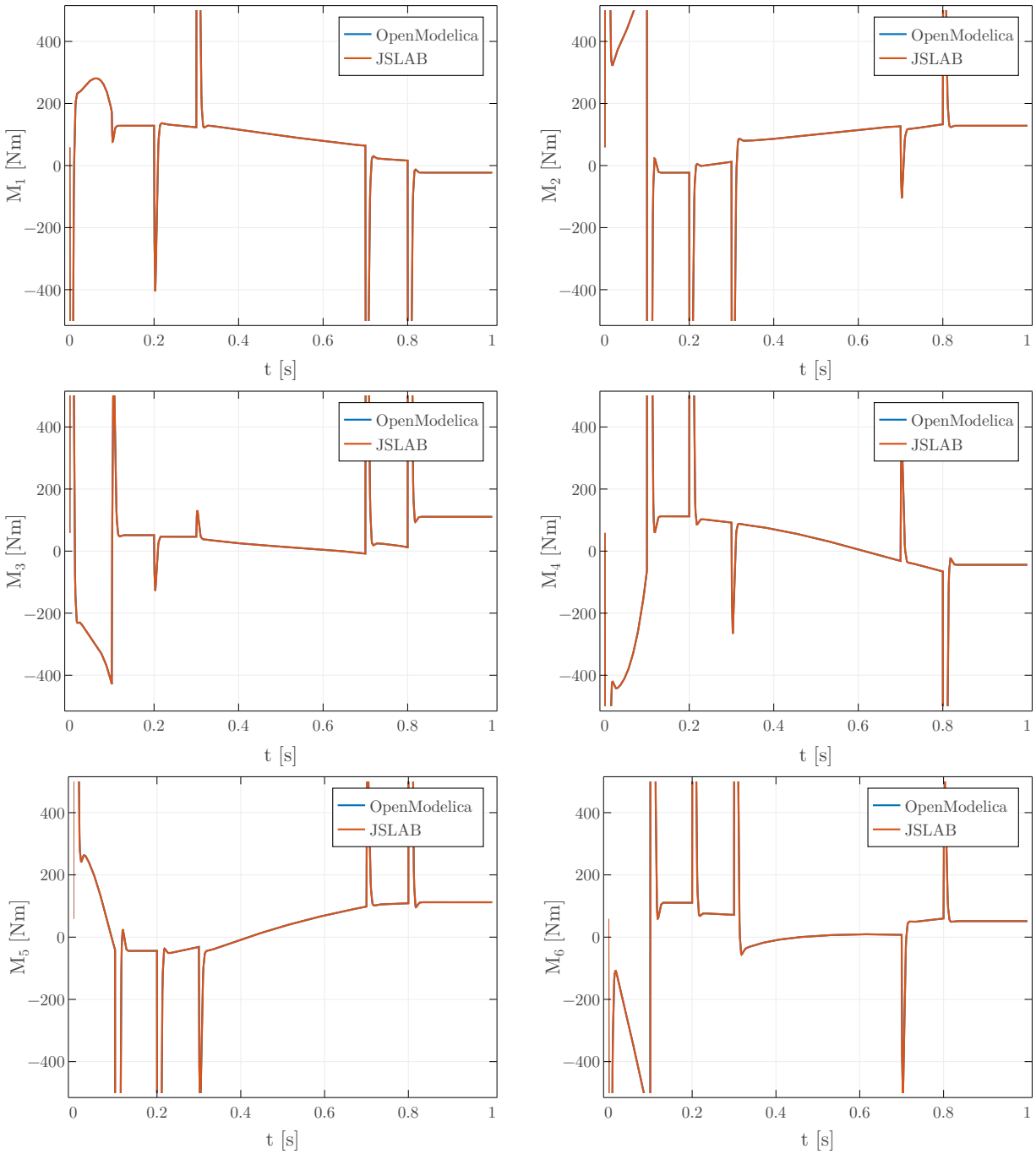
Inverzna kinematika je implementirana unutar `Modelica` modela, omogućavajući izračunavanje potrebnog obrtnog momenta za svaki aktuator za postizanje željene pozicije i orijentacije platforme. Model takođe omogućava izvođenje direktne kinematike i dinamičke analize, što je ključno za dimenzionisanje delova simulatora. Ulazni signali za uglove aktuatora se formiraju na osnovu `TimeTable` bloka. Ovaj blok generiše izlazni signal linearnom interpolacijom na osnovu tabele.

Na sledećim dijagramima (slika 7.5) se jasno vidi potpuno poklapanje rezultata analize potrebnih momenata dobijenih ovom simulacijom i onih dobijenim proračunom na osnovu rešenja inverznog dinamičkog problema. Značajno je da je u isto vreme potvrđeno uspešno rešenje i statičkog i dinamičkog inverznog problema, jer su izabrani takvi signali za aktuatore da dolazi i do zaustavljanja kretanja platforme tokom simulacije.



Slika 7.5 - Promena potrebnog obrtnog momenta

U nastavku su prikazani isti dijagrami (slika 7.6) samo sa ograničenim prikazom od -500 Nm do 500 Nm kako bi se jasnije videle značajne promene umesto samo skokova. Skokovi su posledica toga kako su definisani ulazni signali za kretanje aktuatora.



Slika 7.6 - Promena potrebnog obrtnog momenta sa ograničenim prikazom od -500 Nm do 500 Nm

Treba napomenuti da je ovakve tipove mehanizama (sa zatvorenim kinematičkim lancem) u programima ovog tipa često ili potpuno nemoguće analizirati ili su mogućnosti veoma ograničene.

7.2 Numerička i eksperimentalna verifikacija rešenja inverznog i direktnog kinematičkog problema

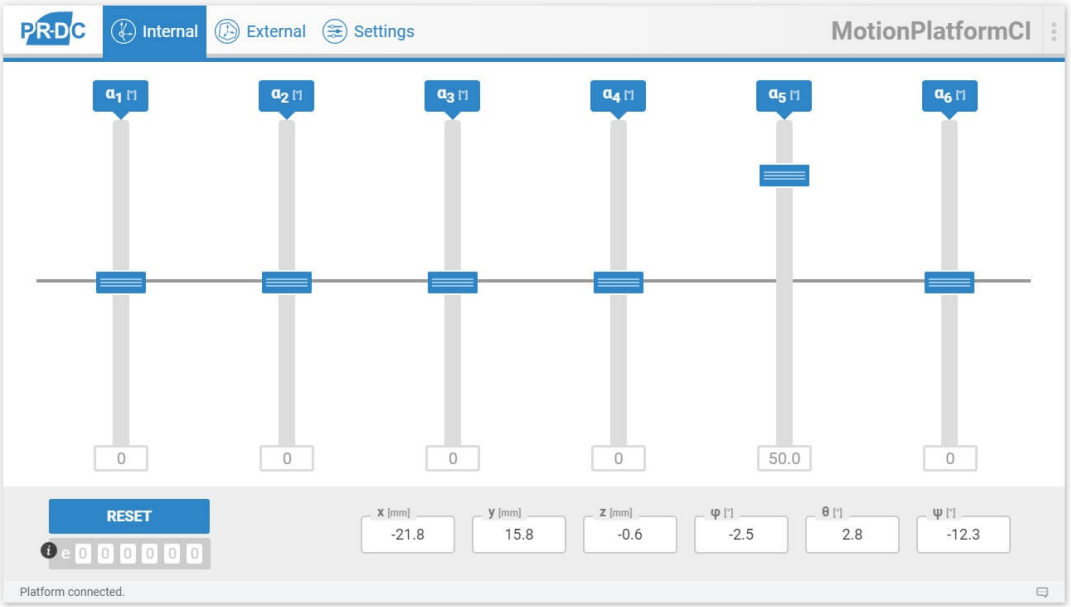
U poglavlju 4 ove disertacije su data rešenja inverznog i direktnog kinematičkog problema za izabrani mehanizam simulatora leta. Pored rešenja su date i skripte kojima su ova rešenja u potpunosti implementirana u programu `JSLAB`.

Rešenja, kao i samu implementaciju, moguće je jednostavno numerički proveriti. U poglavlju 4.2 je prikazano da su koordinate tačaka B_i fiksne, odnosno da se ne menjaju tokom rešavanja inverznog i direktnog kinematičkog problema. Sa druge strane koordinate tačaka P_i se u oba slučaja određuju korišćenjem samo spoljašnjih koordinata i matrice rotacije. Ukoliko se usvoji da su koordinate tačaka B_i i P_i ispravno izračunate, ostatak proračuna moguće je proveriti na osnovu toga što svih 12 poluga imaju konstante dužine i to je rastojanje između tačaka B_i i A_i jednako geometrijskom parametru a, dok je rastojanje između tačaka A_i i P_i jednako geometrijskom parametru s. U sledećem listingu (listing 7.1) je kod kojim je moguće izvršiti ovu proveru.

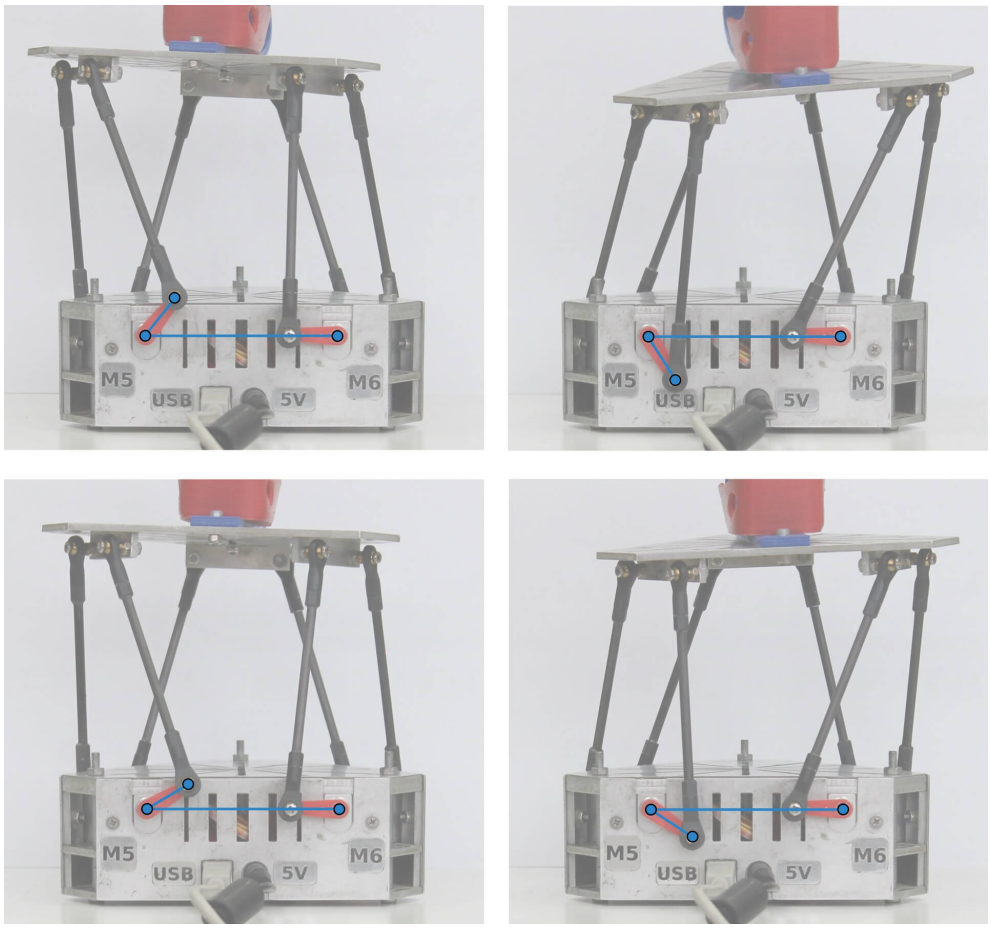
```
var BA = minus(MP.A, MP.B);
var APb = minus(MP.Pb, MP.A);
var a = [];
var s = [];
for(var j = 0; j < MP.m; j++) {
   a[j] = norm([BA[j * 3], BA[j * 3 + 1], BA[j * 3 + 2]]);
   s[j] = norm([APb[j * 3], APb[j * 3 + 1], APb[j * 3 + 2]]);
}
dispRowVector('a', a, MP.m);
dispRowVector('s', s, MP.m);
```

Listing 7.1 - `checkSolver.jsl`

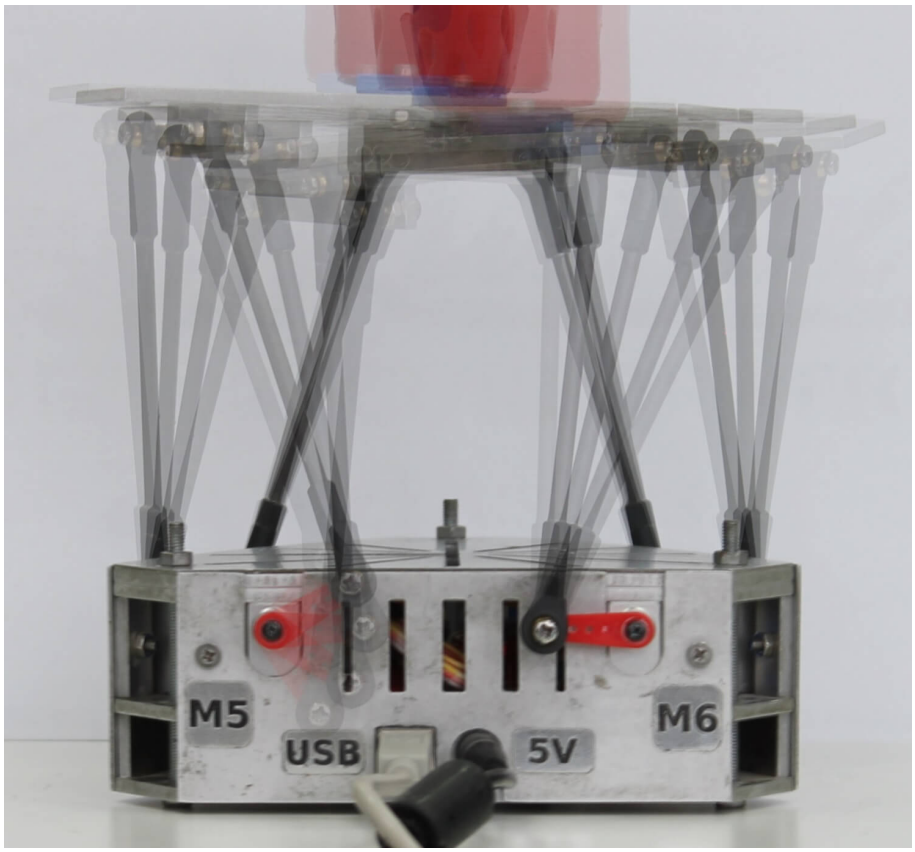
Ovom metodom numerički su validirana oba rešenja, i inverzne i direktne kinematike. Za eksperimentalnu validaciju se može koristiti već prikazani program `MotionPlatformCI` koji omogućava jednostavno upravljanje korišćenjem ili spoljašnjih ili unutrašnjih koordinata. Na slici 7.7 je prikazano podešavanje ugla motora `M5` i to postavljanjem na 50 °.



Slika 7.7 - Podešavanje ugla motora M5 korišćenjem programa `MotionPlatformCI`

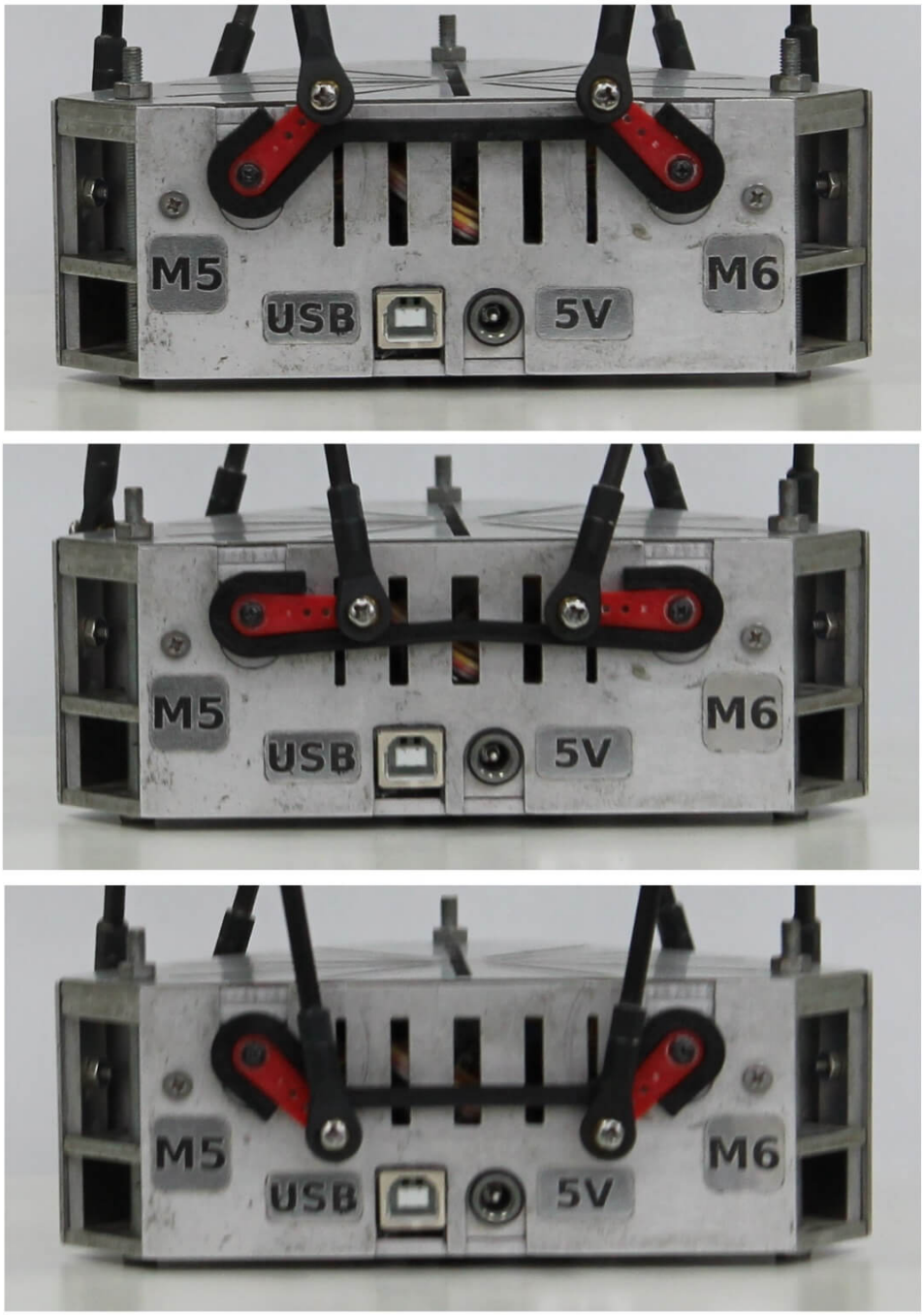


Slika 7.8 - Prikaz rotacije motora M5 za 50°, -50°, 30° i -30°



Slika 7.9 - Prikaz kretanja mehanizma simulatora leta sa upravljanjem samo motora M5

Na slici 7.8 prikazana je rotacija motora `M5` za 50°, -50°, 30° i -30°. Ovo predstavlja statički validacioni eksperiment, sličan onome prikazanom u [186]. Fotografije su načinjene sa nepomičnom kamerom dok se mehanizam kretao na osnovu poslatih željenih vrednosti unutrašnjih koordinata. Sa ovih fotografija moguće je oceniti grešku pozicioniranja odnosno odstupanje ugla otklona servo motora od željenog. Na slici 7.9 su preklopljene prethodne slike kako bi se vizualizovalo kretanje platforme u odnosu na bazu.

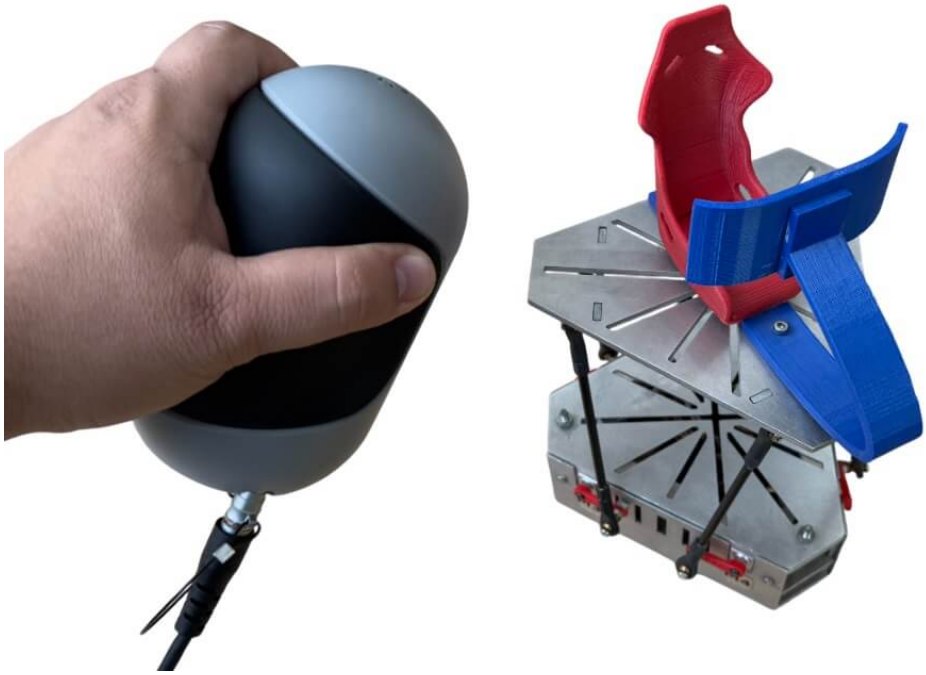


Slika 7.10 - Kalibracija ugla servo motora

Na slici 7.10 prikazana je kalibracija servo motora sa fiksnim kontrolnicima, kako bi poslati komandni signal zapravo odgovarao željenom uglu otklona. Umesto fotografija, video-snimci se mogu direktno analizirati programima kao što je `Kinovea`[187], za merenje i praćenje kretanja na video-snimcima. Na osnovu dobijenih rezultata i poznatih željenih vrednosti moguće je i oceniti grešku pozicioniranja.

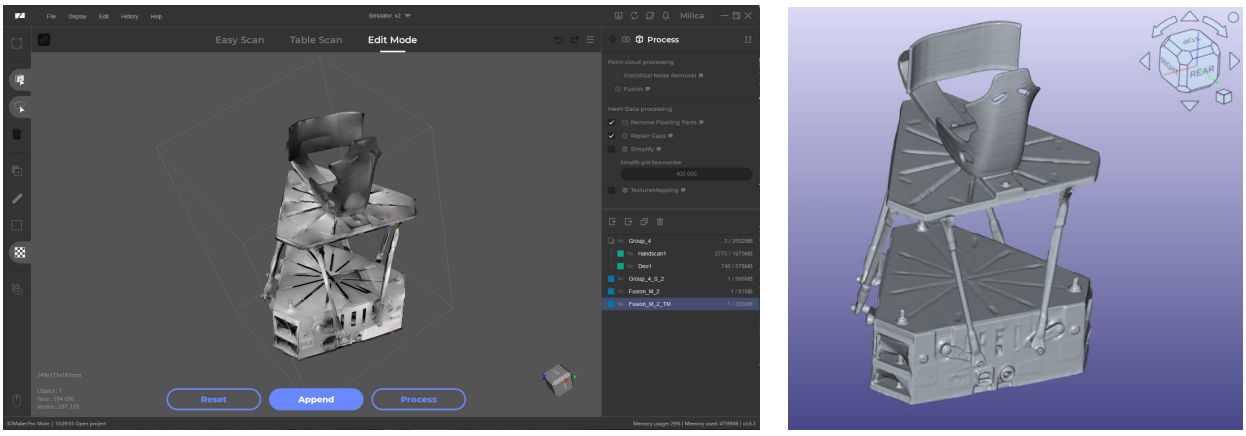
Na osnovu prethodno prikazanog može se zaključiti da su rešenja inverznog i direktnog kinematičkog problema tačna i da se mogu koristiti za upravljanje kretanjem platforme.

U radovima [188]–[190] prikazana je vizuelna, identifikacija, kalibracija i upravljanje na osnovu kamere i reflektujućih markera. Analizu fotografisanjem ili pravljenjem video-zapisa moguće je podići na viši nivo korišćenjem 3D skenera. Ovi uređaji zapravo mogu imati jako slične senzore onima koji se koriste za fotografisanje, ali pored toga imaju implementirane i odgovarajuće metode koje omogućavaju da se na osnovu snimljenih podataka generiše trodimenzionalni model. Na slici 7.11 prikazano je skeniranje simulatora leta pomoću uređaja *Mole* proizvođača *3DMakerpro*. Pokazuje se da je čak i sa uređajima najniže klase moguće postići solidne rezultate i dobiti 3D model koji je upotrebljiv u pogledu merenja dužina i određivanja uglova.



Slika 7.11 - Skeniranje simulatora leta sa uređajem *3DMakerpro Mole*

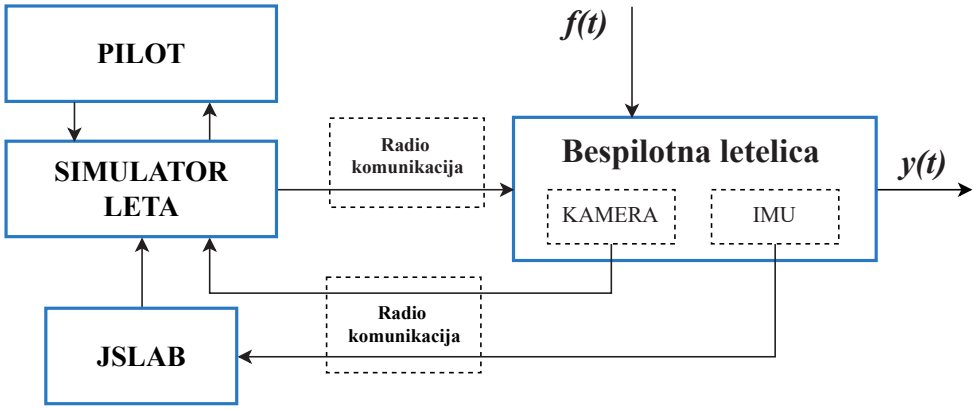
Na slici 7.12 je prikazan program *JMStudio* kojim se vrši skeniranje, obrada i skladištenje rezultata. Na istoj slici je prikazan i model koji je izvezen iz programa *JMStudio* i potom uvezen u standardan program za 3D modeliranje i to *FreeCAD*.



Slika 7.12 - Rezultat skeniranja simulatora leta u programima *JMStudio* (levo) i *FreeCAD* (desno)

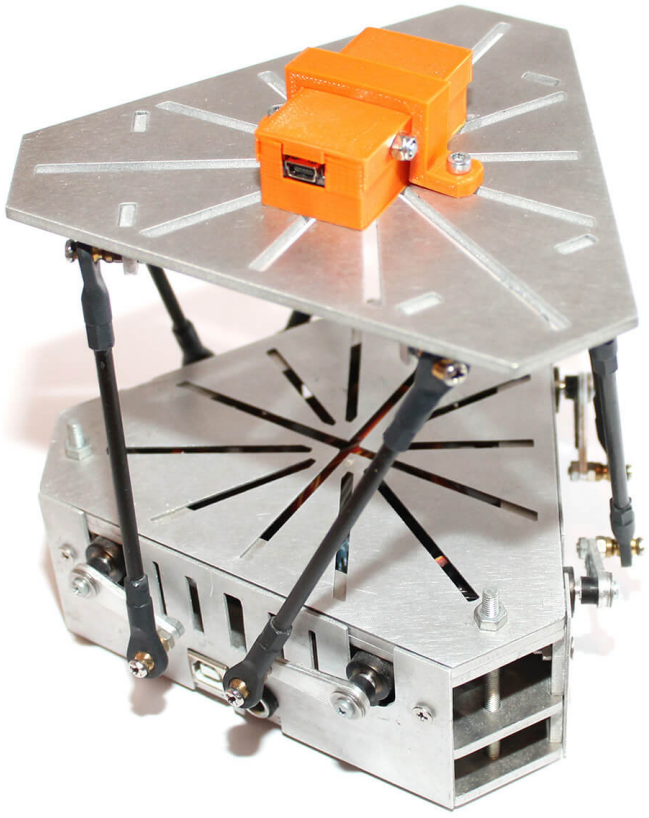
7.3 Osnovno ispitivanje koncepta sistema sa umanjenim prototipom simulatora leta

Na slici 7.13 prikazan je blok dijagram za osnovno ispitivanje koncepta sistema. Na dijagramu je prikazano da pilot šalje upravljanje na letelicu preko simulatora leta, odnosno pilotske kabine koja se nalazi na mehanizmu simulatora leta.



Slika 7.13 - Blok dijagram za osnovno ispitivanje koncepta sistema

Sličan inercijalni merni sistem koji se nalazi na letelici *QFLY-250* je postavljen i na pokretnu platformu umanjenog prototipa simulatora leta za potrebe ovog eksperimenta (kao što se vidi na slici 7.14). U okviru eksperimenta se meri kretanje simulatora leta, odnosno ono što bi pilot koji bi sedeo u pilotskoj kabini simulatora leta osećao.



Slika 7.14 - Modul sa senzorom ubrzanja i orijentacije na platformi umanjenog prototipa simulatora leta

Radio prijemnik i senzor na platformi su tokom eksperimenta serijskom komunikacijom povezani sa računarom na kome je pokrenuta skripta napisana u programu JSLAB. Nakon pokretanja programa, podaci o kretanju bespilotne letelice kroz prostor se prvo prenose sa predajnika na prijemnik, a zatim i do programa. Zatim se izračunavaju potrebni uglovi otklona poluga servo motora i prenose na simulator leta koji izračunate uglove pretvara u odgovarajuće signale za upravljanje servo motorima. Paralelno sa ovim se očitavaju podaci sa senzora koji se nalazi na platformi simulatora leta.

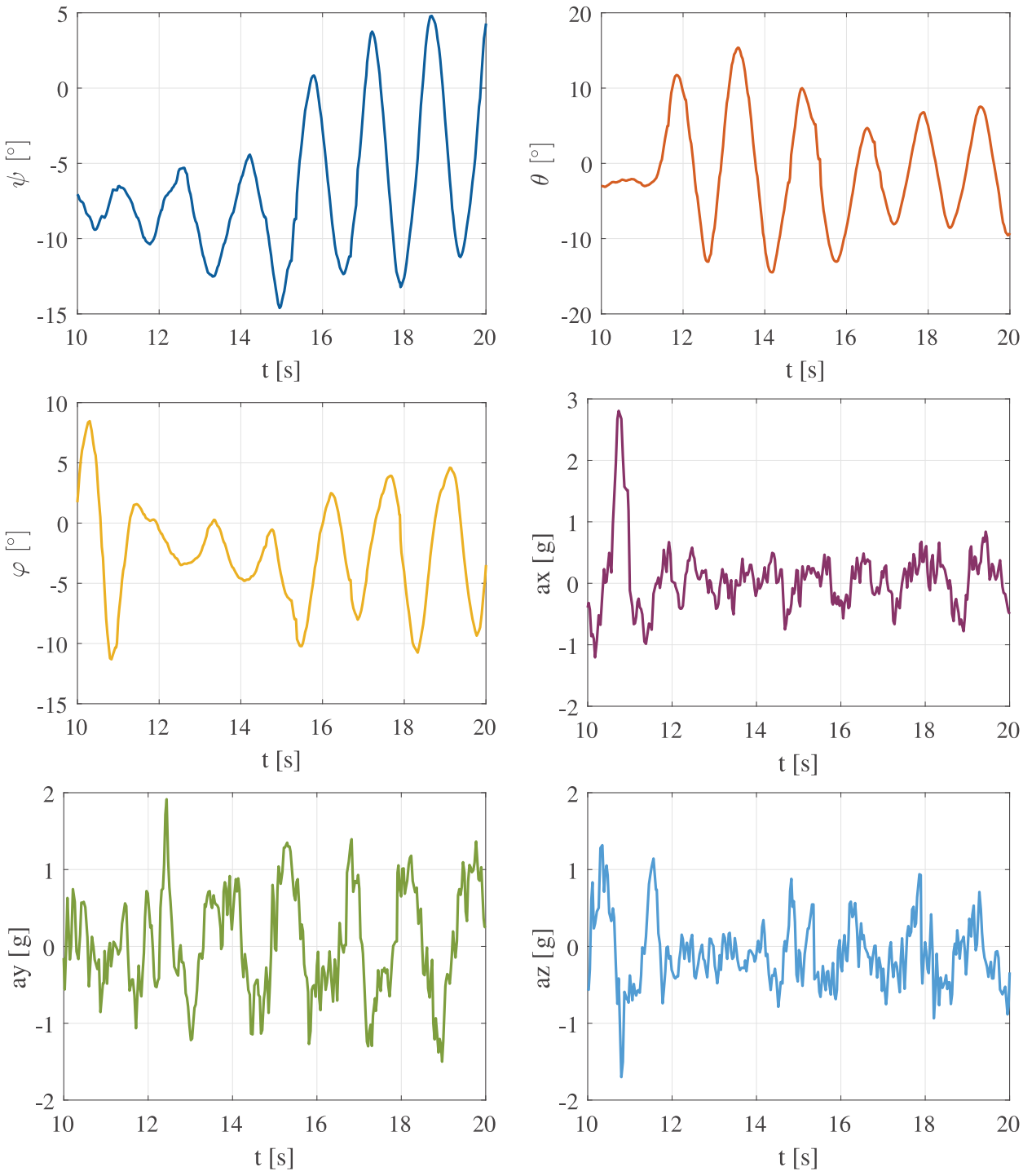
Na slici 7.15 prikazana je bespilotna letelica sa modulom kojim se meri ubrzanje i orijentacija letelice u prostoru.



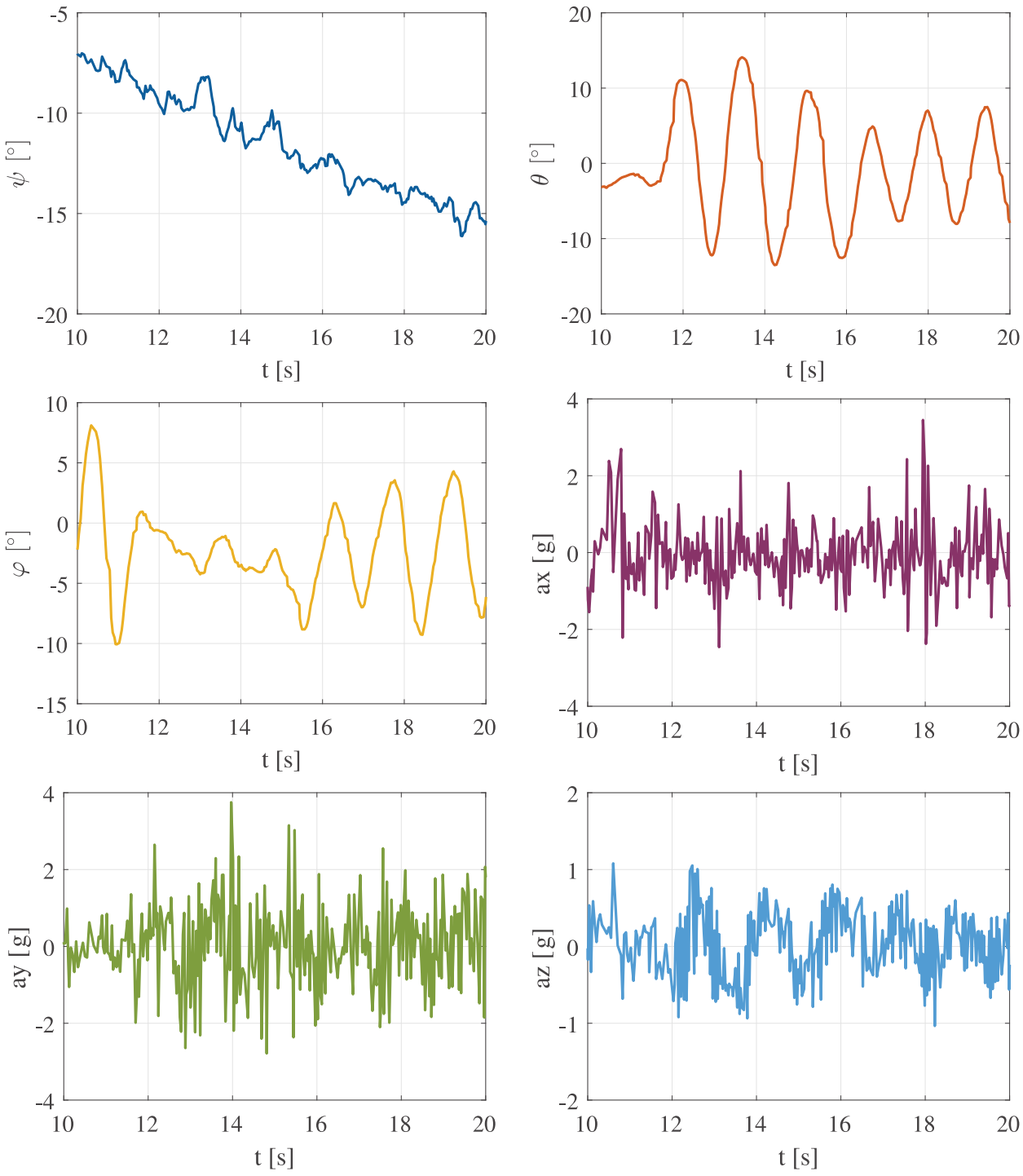
Slika 7.15 - Modul sa senzorom ubrzanja i orijentacije na kvadkopteru

Rezultati merenja su prikazani na sledećim dijagramima (slike 7.16 i 7.17). Prvo su prikazane promene ugla skretanja, propinjanja i valjanja, zatim i tri komponente ubrzanja bespilotne letelice. Nakon toga su prikazani isti dijagrami za platformu simulatora leta.

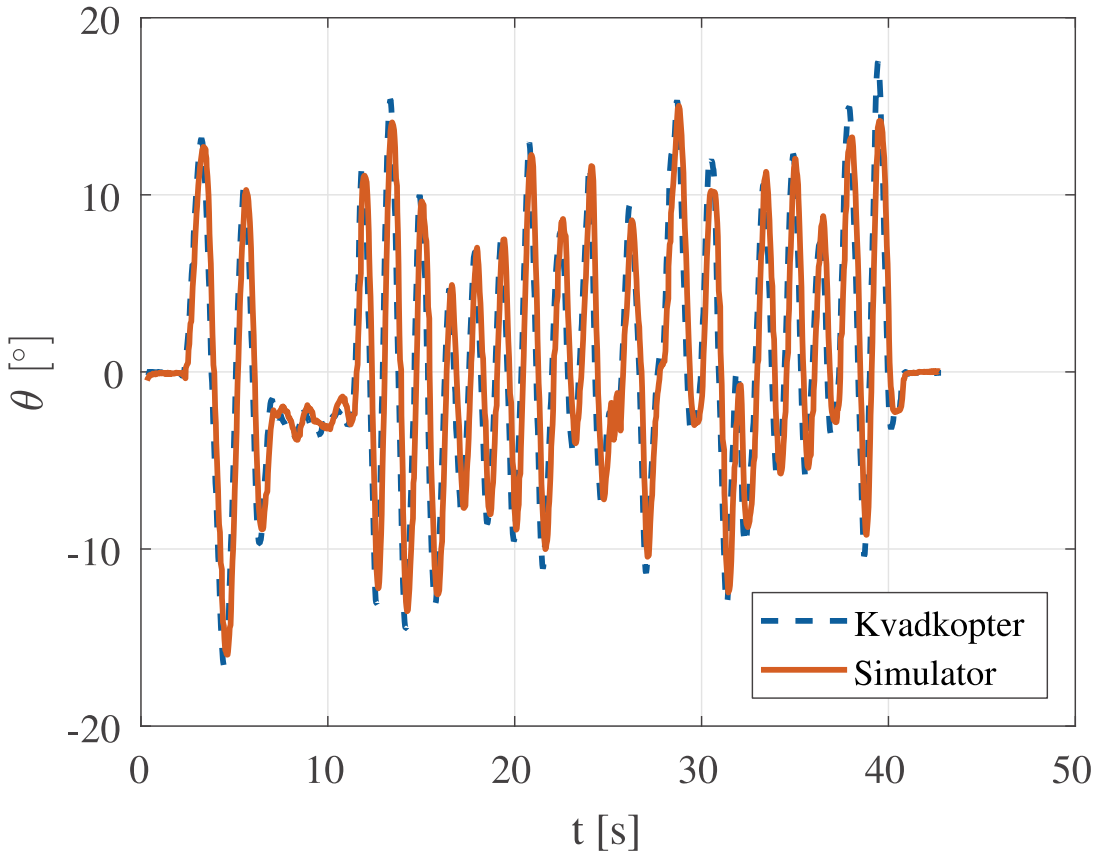
Na slikama 7.18 i 7.19 su prikazani dijagrami sa promenom ugla propinjanja i valjanja tokom eksperimenta za kvadkopter i simulator uporedno. Ova dva parametra su data kao dva primarna pokazatelja rada ovog sistema. Na dijagramima se može uočiti da je upravljanje asinhrono. Iako je moguće upravljati platformom tako da se nova instrukcija motorima šalje tek kada je prethodna zadata vrednost ostvarena, u ovom slučaju novo upravljanje generiše se na osnovu trenutnih vrednosti sa kvadkoptera.



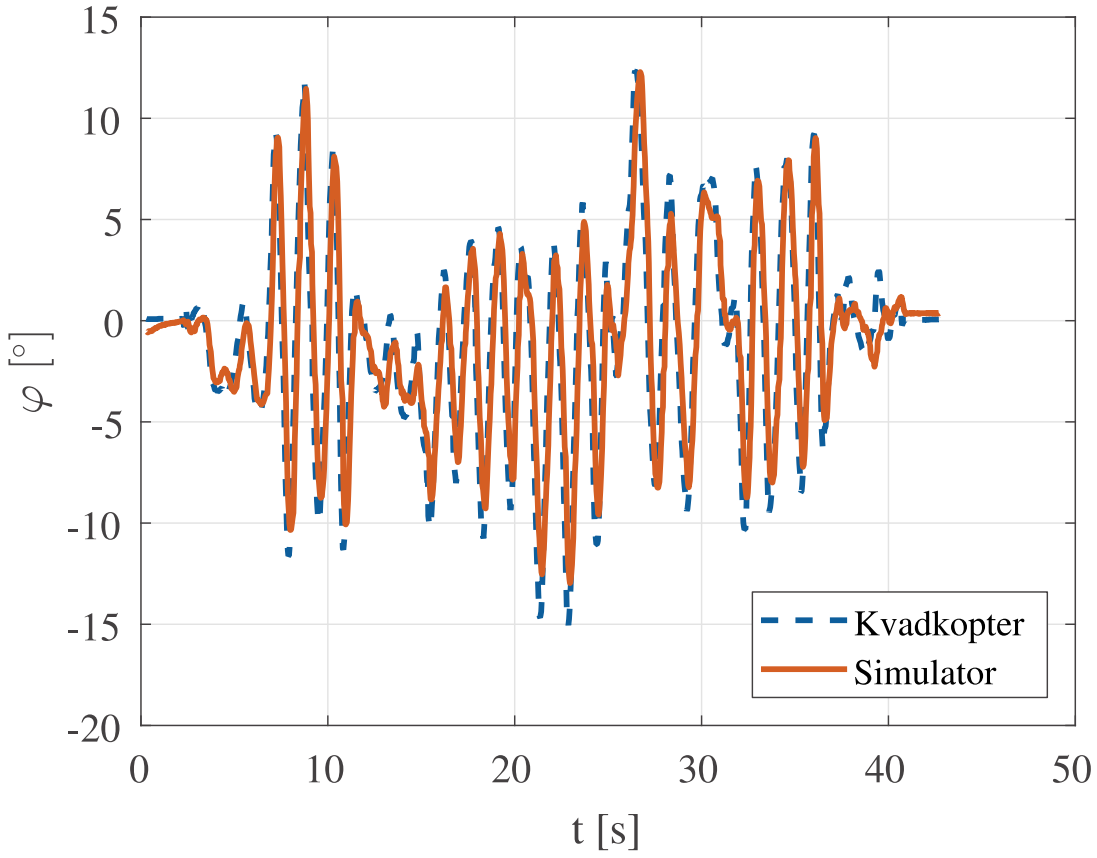
Slika 7.16 - Rezultati merenja na kvadkopteru



Slika 7.17 - Rezultati merenja na simulatoru leta

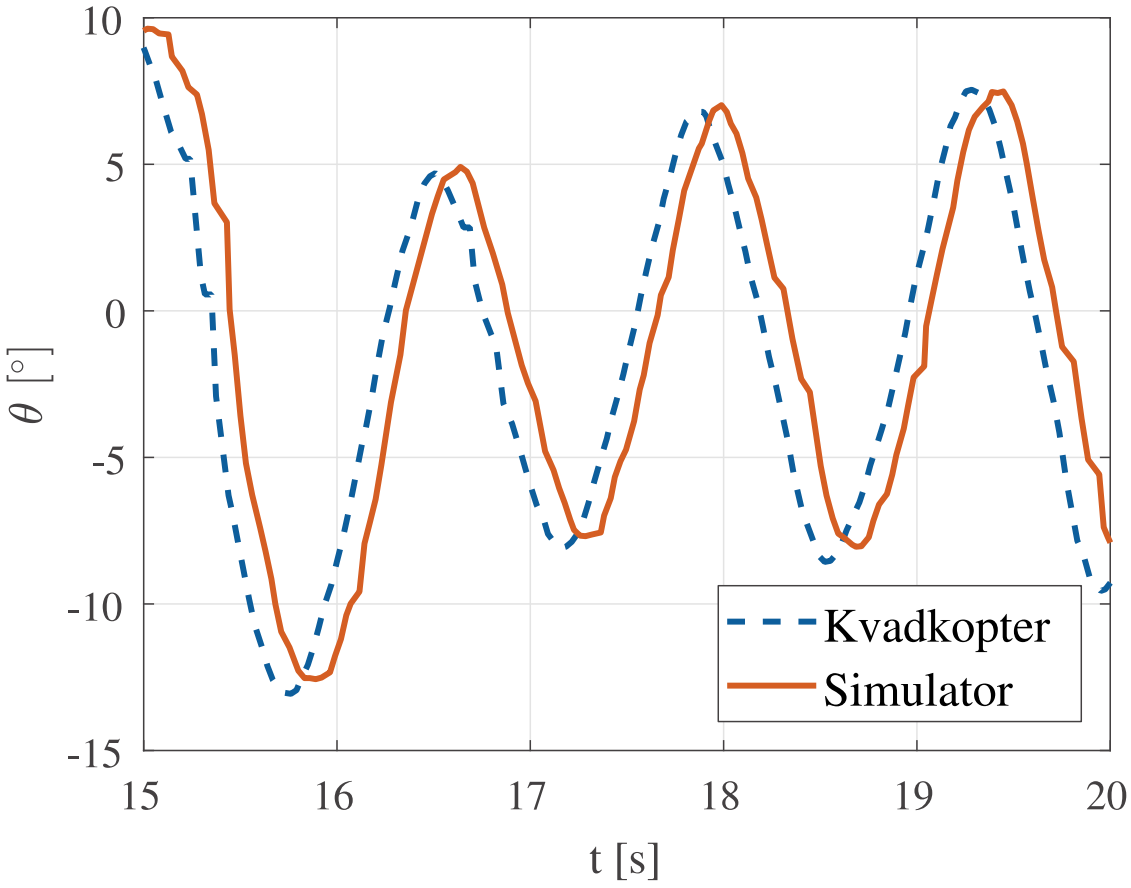


Slika 7.18 - Promena ugla propinjanja kvadkoptera i simulatora tokom eksperimenta



Slika 7.19 - Promena ugla valjanja kvadkoptera i simulatora tokom eksperimenta

Na dijagramu sa slike 7.20 se jasnije može videti odstupanje izmerenog ugla propinjanja na simulatoru i na kvadkopteru. Za preciznije dobijanje ovog odstupanja potrebno je precizno sinhronizovati merenja na oba modula. U radu [191] prikazano je eksperimentalno određivanje baš ovog kašnjenja. Bitno je uočiti da se karakter smera kretanja održava bez obzira na vremensko odstupanje. Drugim rečima, kada letelica nakon poniranja krene u propinjanje, pilot neće još neko kratko vreme ostati u položaju poniranja, već će skoro trenutno preći u propinjanje.



Slika 7.20 - Odstupanje ugla propinjanja simulatora u odnosu na kvadkopter

Vremenska odstupanja u kretanjima kvadkoptera i platforme simulatora leta su neizbežna i posledica su potrebnog vremena za merenja, prenos i obradu signala i samog odziva aktuatora. Unapređenjem svakog od ovih segmenata sistema se vremensko odstupanje može dovesti u dozvoljene granice koje pilot ne može da oseti tokom upravljanja kao što je slučaj i sa amplitudnim odstupanjima. Za slučajeve kada mehanizam ne može da se dovede u određeni položaj, zbog zasićenja nekog stepena slobode, koriste se već prikazani filteri kretanja.

7.4 Ispitivanje kašnjenja telemetrije

U sistemu daljinskog pilotiranja letelicom, jedan od ključnih zahteva za uspešno formiranje povratne sprege sa parametrima kretanja letelice jeste pouzdano i što manje kašnjenje telemetrije (latencija, eng. *latency*).

Telemetrija obuhvata prenos parametara kretanja letelice u realnom vremenu, a svako uočljivo kašnjenje može negativno da utiče na tačnost i efikasnost upravljanja. Kako je vreme

odziva operatera često ključno za stabilnost letelice i bezbednu misiju, neophodno je da ukupno kašnjenje, od trenutka merenja vrednosti na letelici do prikaza tih podataka operateru, bude ispod nivoa uočljivog operateru.

Ovde je prikazana procedura ispitivanja kašnjenja koja daje uvid u realnu vrednost prosečnog vremena kašnjenja telemetrije. Eksperimenti su sprovedeni na računaru `Getac F110 G6` (prikazan levo na slici 7.21), koji se i inače koristi u sklopu daljinskog sistema za upravljanje letelicom, čime su u merenje uključeni i svi dodatni faktori (performanse računara, prikaz na ekranu i sl.) koji bi mogli dodatno uticati na kašnjenje.

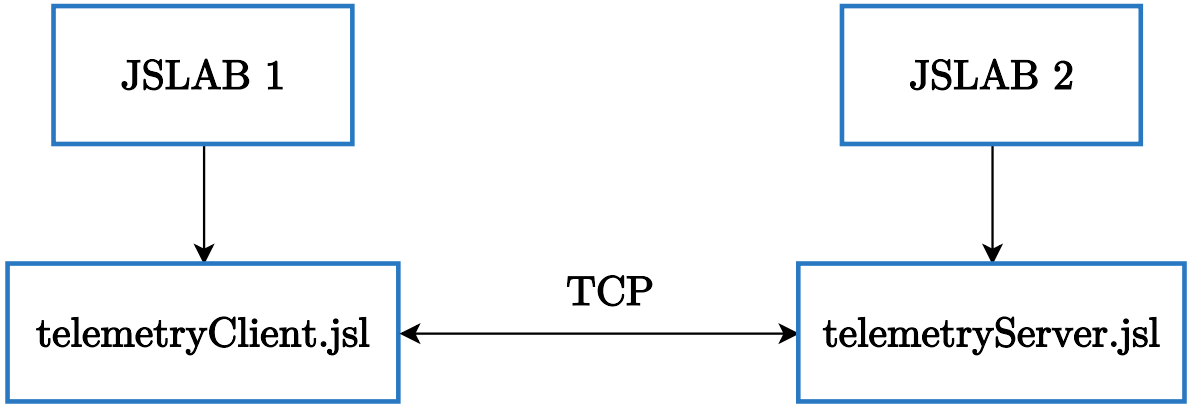


Slika 7.21 - Daljinski upravljač sa tablet računarom `Getac F110 G6` (levo) i laptop računar `Getac X600` sa digitalnim radio modulom (desno)

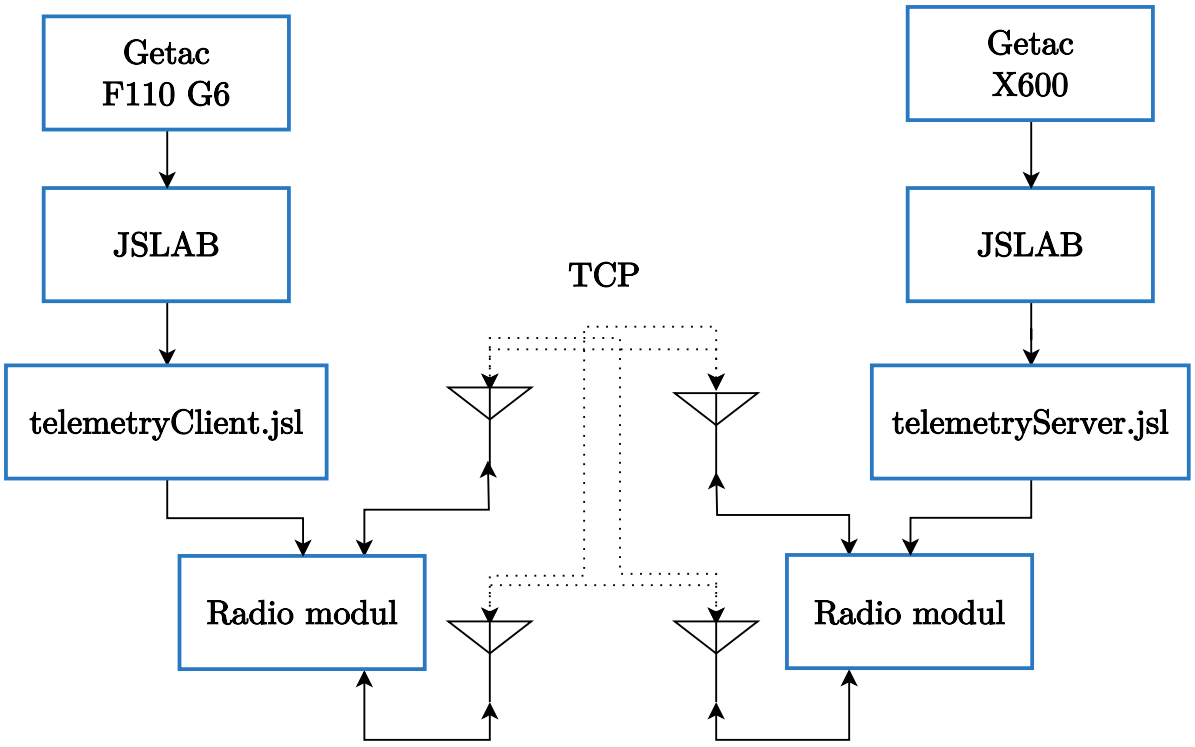
7.4.1 Metodologija ispitivanja

Kašnjenje telemetrije ispitivano je na računaru `Getac F110 G6` u dva različita slučaja:

1. Pri korišćenju `TCP`-a lokalno (slika 7.22).
2. Pri korišćenju `TCP`-a preko radio modula (sa računarom `Getac X600`, slika 7.23).



Slika 7.22 - Postavka ispitivanja kašnjenja podataka usled korišćenja TCP protkola za telemetriju



Slika 7.23 - Postavka ispitivanja kašnjenja podataka usled korišćenja TCP-a i bežične komunikacije za telemetriju

U oba slučaja, za komunikaciju između servera i klijenta se koriste odgovarajuće `JSLAB` skripte i to `telemetryServer.jsl` i `telemetryClient.jsl` čiji kod je dat u listinzima 7.2 i 7.3. Skripte osim što omogućavaju komunikaciju preko `TCP`-a otvaraju i nove prozore u kojima se prikazuju trenutne vremenske oznake.

Server se nalazi na strani letelice ili simuliranog sistema, dok se klijent izvršava na računaru operatera. Za potrebe eksperimenta u oba scenarija (sa i bez radio modula) prikazuju se vremenske oznake (u milisekundama) na ekranu, beležeći proteklo vreme od pokretanja programa.

```
/**
 * @file Telemetry latency experiment - Server
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var tcp_ip = '127.0.0.1';
var tcp_port = 5760;
// --------------------

var win = await openWindowBlank();
win.setSize(500, 90);
win.setTitle('TCP server');
win.document.body.innerHTML += '<div id="timestamp" style="padding: 10px; font-
    size: 36px; text-align: center;">0</div>';
var timestamp_cont = win.document.getElementById('timestamp');

// Create a TCP server
```

```
var com = tcpServer(tcp_ip, tcp_port, function(socket) {
  disp('New client connected!');
});

var t = tic;

function frame(timestamp) {
  var timestamp = stringify(round(tocms(t), 0));
  timestamp_cont.innerText = timestamp;
  for(sid in com.sockets) {
    com.write(com.sockets[sid], timestamp+'\0');
  }
  requestAnimationFrame(frame);
}
requestAnimationFrame(frame);
```

Listing 7.2 - `telemetryServer.jsl`

```
/**
 * @file Telemetry latency experiment - Client
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var tcp_ip = '127.0.0.1';
var tcp_port = 5760;
var tcp_timeout = 3000;
// --------------------

var win = await openWindowBlank();
win.setSize(500, 90);
win.setTitle('TCP client');
win.document.body.innerHTML += '<div id="timestamp" style="padding: 10px; font-
    size: 36px; text-align: center;">0</div>';
var timestamp_cont = win.document.getElementById('timestamp');

// Create a TCP client
var com = tcp(tcp_ip, tcp_port, function() {
  com.setTimeout(0);
  console.log('Connected to TCP telemetry stream');
});
com.setNoDelay(true);
com.setKeepAlive(true);
com.setTimeout(tcp_timeout);
com.setOnError(function(err) {
  disp('Error:', err);
});
com.setOnData(function(data) {
  var timestamp = data.toString().split('\0').reverse().find((s) => s !== '') ||
      null;
  timestamp_cont.innerText = timestamp;
});
```

Listing 7.3 - `telemetryClient.jsl`

Tokom eksperimenata, kamera snima ekran (ili ekrane) kako bi bilo moguće uporediti momenat slanja (merenja) i momenat prijema (prikaza) vrednosti. Snimanje se vrši tako da se

istovremeno vide obe vremenske oznake. Nakon toga, skripta `video2randFrames.jsl` (listing 7.4) generiše pojedinačne slike (eng. *frame*) iz video-snimka, a zatim nasumično bira 18 reprezentativnih slika, dok ostale uklanja. Time se analiza znatno ubrzava i omogućava precizniju procenu kašnjenja.

```
/**
 * @file Convert video to png frames
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc;
disp(' --- ' + jsl_file_name + ' ---');
var N = 18;
var args = ['', '-vf transpose=1', '-vf transpose=2'][2];

// Choose video files
var options = {
  title: 'Choose video file',
  buttonLabel: 'Choose video file',
  properties: ['multiSelections'],
  filters: [
    {name: 'All Files', extensions: ['*']}
  ]
};

var files = chooseFile(options);
if(!Array.isArray(files)) {
  files = [files];
}

tic;
for(var i = 0; i < files.length; i++) {
  var file = files[i];
  // Convert video
  disp(' Converting video ' + file + ' to png (' + (i+1) + '/' +
       files.length + ').');

  var folder = removeExtension(file);
  mkdir(folder);

  // Install FFmpeg https://www.ffmpeg.org/ add path to environment variables
     path
  system('ffmpeg -i "' + file + '" ' + args + ' "'+folder+'/frame_%04d.png"');
  var files = filesInFolder(folder);
  var I_frames = zeros(N).map(() => randInt(0, files.length-1));
  for(var i = 0; i < files.length; i++) {
    if(!I_frames.includes(i)) {
      deleteFile(files[i]);
    }
  }
}

// End of program
disp(' The program was executed successfully... ');
disp(' Execution time: ' + num2str(toc(), 2) + ' seconds');
disp(' -------------------- ');
```

Listing 7.4 - `video2randFrames.jsl`

7.4.2 Rezultati ispitivanja kašnjenja pri korišćenju TCP-a lokalno

Na slici 7.24 je prikazano 18 nasumično izabranih slika iz video-snimka ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja telemetrije preko TCP-a lokalno. U nastavku je data i tabela 7.1 u kojoj su obrađeni podaci sa slika i date vrednosti kašnjenja.



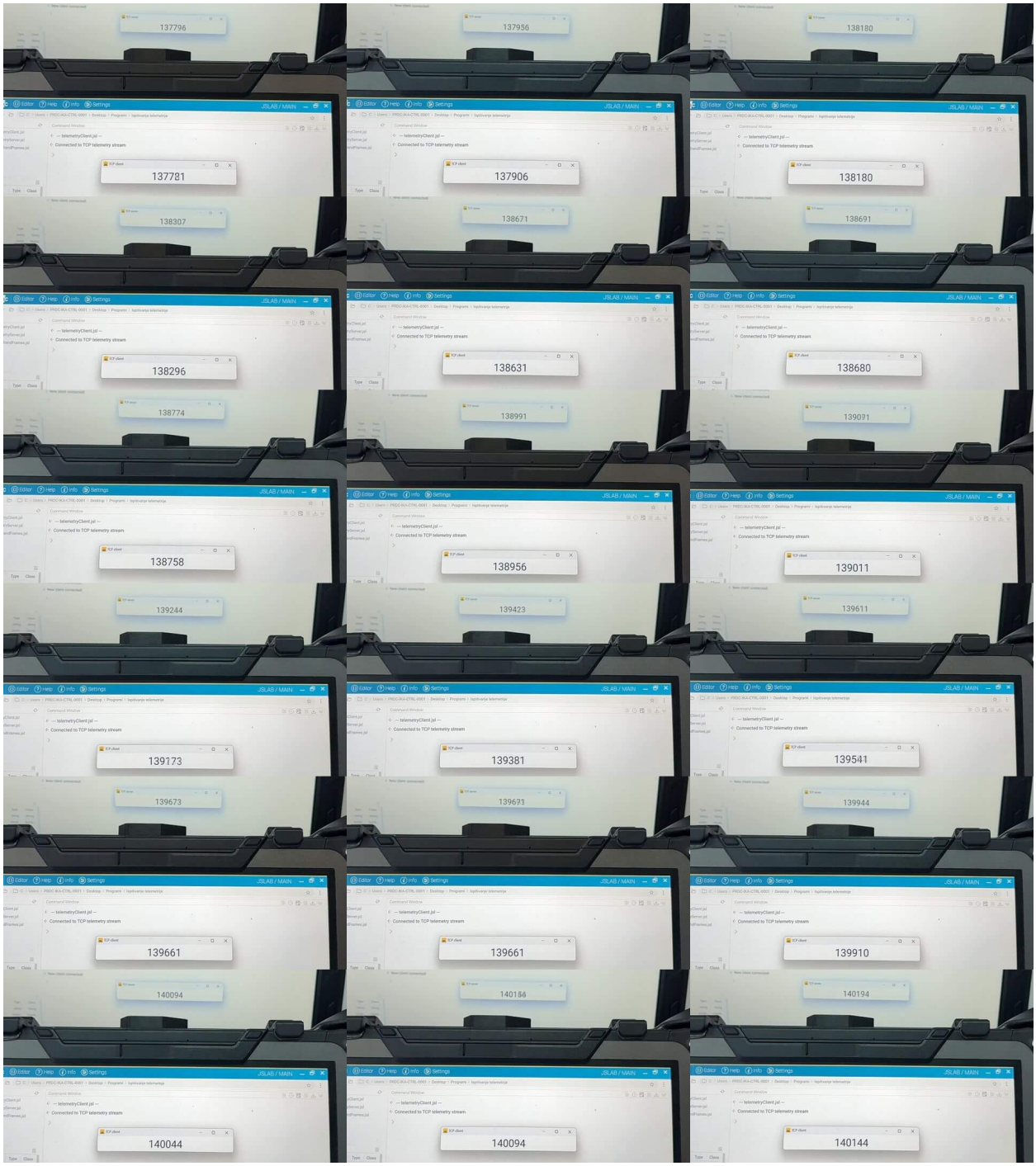
Slika 7.24 - Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja telemetrije preko TCP-a lokalno

Tabela 7.1 - Izračunata kašnjenja sa slika ekrana tokom ispitivanja kašnjenja telemetrije preko TCP-a lokalno

RB	Frame	Δt [ms]	RB	Frame	Δt [ms]	RB	Frame	Δt [ms]
1	11	0	2	94	8	3	164	0
4	234	0	5	337	0	6	402	0
7	490	0	8	514	7	9	536	6
10	556	0	11	565	0	12	575	20
13	590	9	14	614	0	15	659	20
16	747	20	17	754	0	18	817	5

7.4.3 Rezultati ispitivanja kašnjenja pri korišćenju TCP-a preko radio modula

U nastavku je ispitano kako se kašnjenje menja kada se koristi bežična komunikacija preko radio modula. Na slici 7.25 je prikazano 18 nasumično izabranih slika iz video-snimka ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja telemetrije preko TCP-a i radio modula. U nastavku je data i tabela 7.2 u kojoj su obrađeni podaci sa slika i date vrednosti kašnjenja.



Slika 7.25 - Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja telemetrije preko TCP-a i radio modula

Tabela 7.2 - Izračunata kašnjenja sa slika ekrana tokom ispitivanja kašnjenja telemetrije preko TCP-a i radio modula

RB	Frame	Δt [ms]	RB	Frame	Δt [ms]	RB	Frame	Δt [ms]
1	32	15	2	69	50	3	124	0
4	152	11	5	238	40	6	244	11
7	263	16	8	315	35	9	337	60
10	374	71	11	420	42	12	462	70
13	478	12	14	481	12	15	544	34
16	580	50	17	593	62	18	604	50

7.4.4 Analiza rezultata

Na osnovu prikazanih *rezultata ispitivanja kašnjenja pri korišćenju TCP-a lokalno*, obrađeni rezultati pokazuju sledeće:

- Minimalno kašnjenje: **0 ms**
- Prosečno kašnjenje: **5.3 ms**
- Maksimalno kašnjenje: **20 ms**

Može se zaključiti da:

- Minimalno kašnjenje od 0 ms (unutar rezolucije merenja) ukazuje na praktično istovremeni prikaz podataka.
- Prosečno kašnjenje od 5.3 ms je izuzetno nisko i ne predstavlja smetnju operateru.
- Maksimalno izmereno kašnjenje iznosi 20 ms, što je i dalje u granicama koje operater neće značajno osetiti (manje od vremena koje protekne između dve slike u videu na 30 fps). Ovakva kašnjenja mogu biti rezultat kratkotrajnog zagušenja mrežnog saobraćaja, opterećenja procesora, ekrana ili drugih, ređih faktora.

Za potrebe daljinskog pilotiranja, ovakav nivo kašnjenja usled `TCP` komunikacije je praktično zanemarljiv, jer operater gotovo da neće primetiti kašnjenje od nekoliko milisekundi.

Na osnovu prikazanih *rezultata ispitivanja kašnjenja pri korišćenju TCP-a preko radio modula*, obrađeni rezultati pokazuju sledeće:

- Minimalno kašnjenje: **0 ms**
- Prosečno kašnjenje: **35.6 ms**
- Maksimalno kašnjenje: **71 ms**

Može se zaključiti da:

- Rezultati pokazuju da je raspon kašnjenja veći nego kod `TCP`-a lokalno što nedvosmisleno ukazuje da je do povećanja došlo usled korišćenja bežične veze preko radio modula.
- Iako se minimalno kašnjenje ponekad meri kao 0 ms, bežična komunikacija u proseku ima dosta veće vrednosti (35.6 ms).

- Maksimalna zabeležena vrednost kašnjenja od 71 ms obično se javlja usled zagušenja mrežnog saobraćaja, trenutnog opterećenja uređaja za obradu podataka, načina formiranja paketa ili gubitaka paketa prilikom bežičnog prenosa.
- Prosečno kašnjenje od 35.6 ms i dalje je dovoljno niska vrednost za većinu tipičnih primena, približno je vremenu između dve slike u video-snimku na 30 fps.

Na osnovu *sprovedenih ispitivanja i analiziranih rezultata za oba slučaja ispitivanja*, može se zaključiti da:

- Lokalno `TCP` pruža izuzetno nisko prosečno kašnjenje.
- `TCP` komunikacija preko radio modula unosi veće, ali i dalje prihvatljivo prosečno kašnjenje (35.6 ms) uz povremene pikove.

Generalno gledano, za većinu operacija u kojima se telemetrija prikazuje čoveku, kašnjenje od nekoliko desetina milisekundi je zadovoljavajuće. U slučaju kada je potrebno dodatno smanjiti ovo vreme, mogu se koristiti i drugi protokoli (kao što je `UDP`) koji se fokusiraju na brzinu prenosa više nego na pouzdanost a zatim je moguće optimizovati i ostatak softvera i hardver koji se koristi. Ovim ispitivanjima je potvrđeno da, pri adekvatnom izboru komunikacionog protokola i odgovarajuće opreme, sistem daljinskog pilotiranja može pouzdano da funkcioniše sa prihvatljivim nivoom kašnjenja. Značajno je i to što je prikazana komunikacija potpuno digitalna i omogućava prenos podataka u bilo kom digitalnom obliku.

7.5 Ispitivanje kašnjenja video-snimka

Ovo potpoglavlje disertacije predstavlja logičan nastavak potpoglavlja 7.4 u kom je prikazano ispitivanje kašnjenja telemetrije. U sistemu daljinskog pilotiranja letelicom, koji je predmet ove disertacije, se osim telemetrije koristi i video-snimak sa letelice koji se prikazuje operateru u realnom vremenu. Vreme kašnjenja video-snimka, kao i u slučaju telemetrije, može negativno da utiče na tačnost i efikasnost upravljanja.

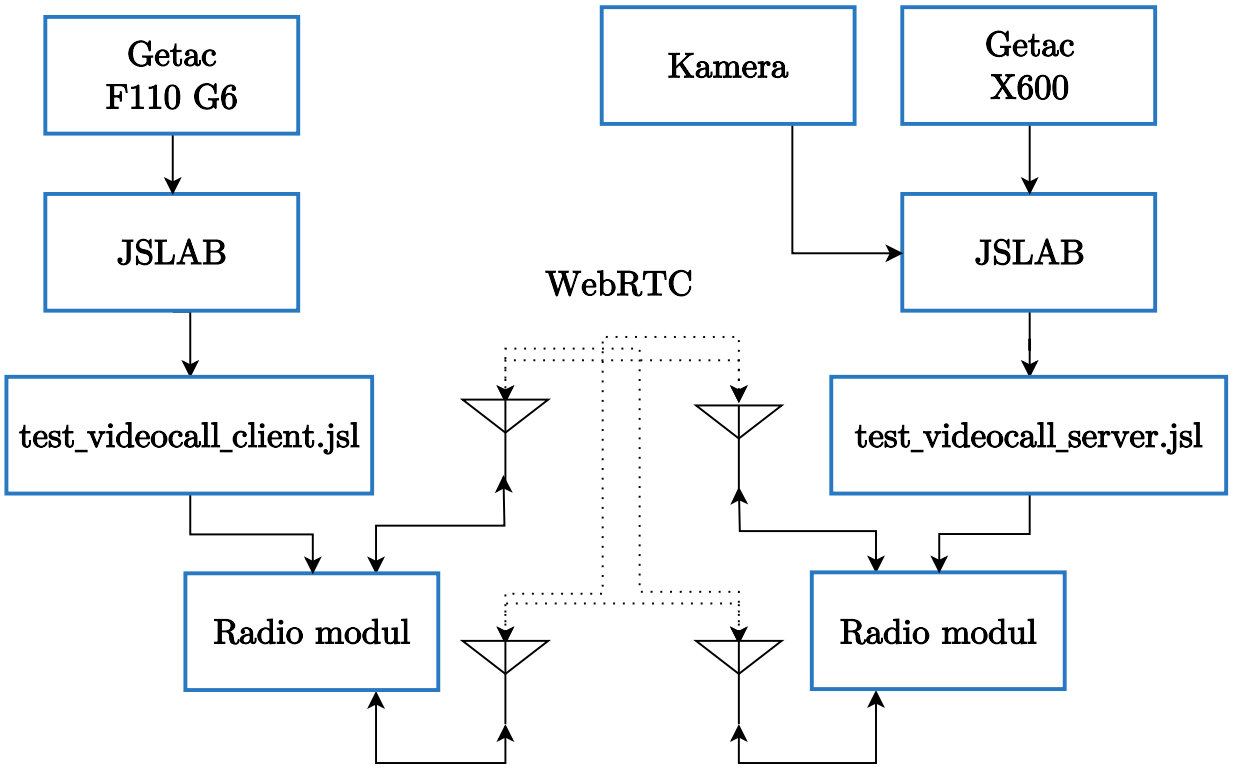
Ovde je prikazana procedura ispitivanja kašnjenja koja daje uvid u realnu vrednost prosečnog vremena kašnjenja video-snimka. Eksperimenti su sprovedeni na računaru `Getac F110 G6`, koji se i inače koristi u sklopu daljinskog sistema za upravljanje letelicom, čime su u merenje uključeni i svi dodatni faktori (performanse računara, prikaz na ekranu i sl.) koji bi mogli dodatno uticati na kašnjenje.

7.5.1 Metodologija ispitivanja

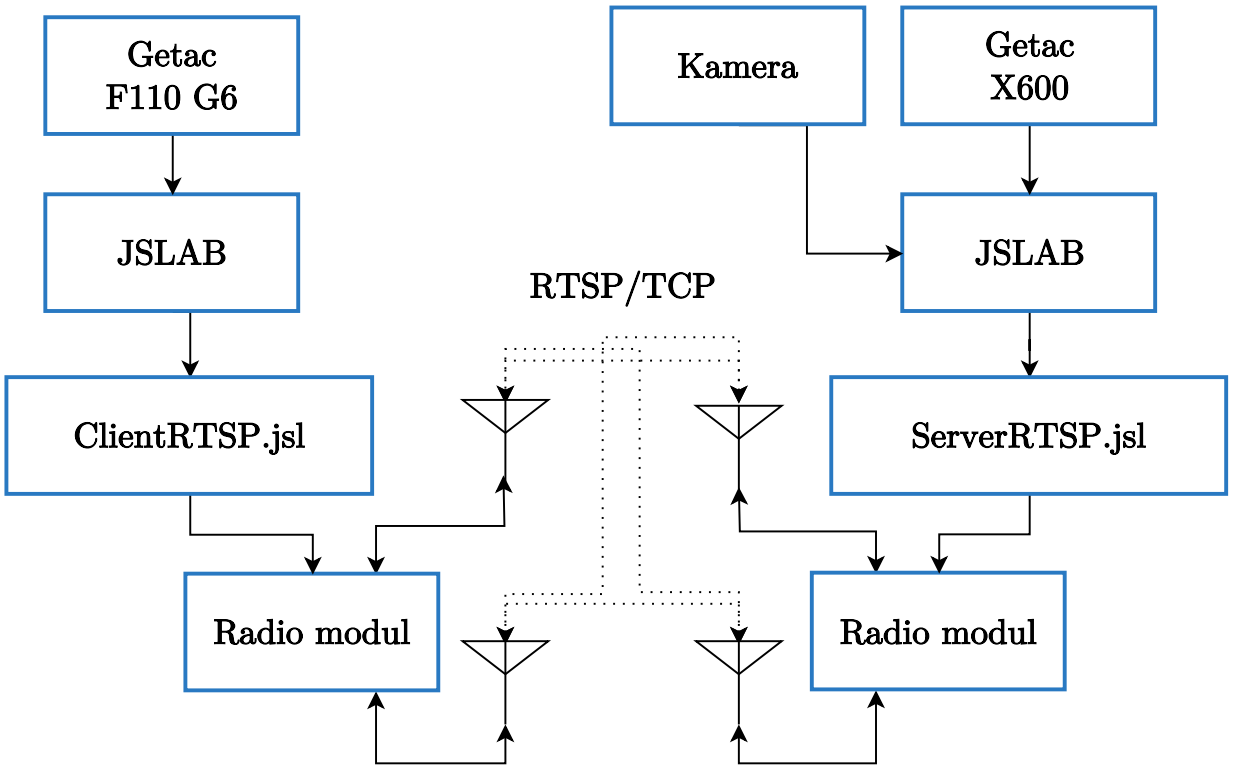
Metodologija ispitivanja je identična kao i prilikom ispitivanja kašnjenja telemetrije. Ponovo se snimaju ekrani sa vremenskim oznakama i video-snimci se potom obrađuju skriptom `video2randFrames.jsl`.

Kašnjenje video-snimka ispitivano je na računaru `Getac F110 G6` u tri različita slučaja:

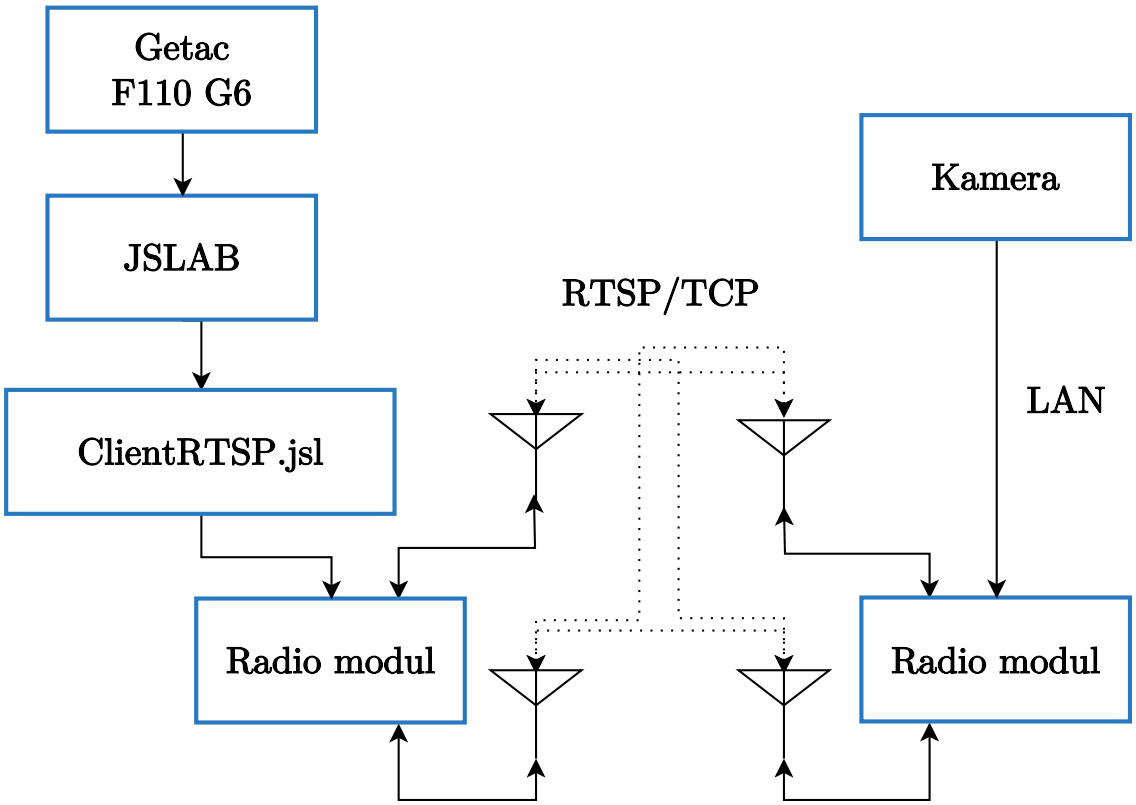
1. Pri korišćenju `WebRTC`-a sa integrisanom `USB` kamerom na `Getac X600` (slika 7.26).
2. Pri korišćenju `RTSP`-a sa integrisanom `USB` kamerom na `Getac X600` (slika 7.27).
3. Pri korišćenju `RTSP`-a sa `LAN` kamerom na letelici (slika 7.28).



Slika 7.26 - Postavka ispitivanja kašnjenja video-snimka korišćenjem WebRTC-a



Slika 7.27 - Postavka ispitivanja kašnjenja video-snimka korišćenjem RTSP-a



Slika 7.28 - Postavka ispitivanja kašnjenja video-snimka korišćenjem RTSP-a i LAN kamere

Na slici 7.29 je prikazano kako je izvršeno snimanje ekrana tableta `Getac F110 G6` pomoću kamere sa laptopa `Getac X600`.



Slika 7.29 - Snimanje ekrana tableta `Getac F110 G6` pomoću kamere sa laptopa `Getac X600`

Na slici 7.30 je prikazano kako je izvršeno snimanje ekrana tableta `Getac F110 G6` pomoću `LAN` kamere na letelici.



Slika 7.30 - Snimanje ekrana tableta `Getac F110 G6` pomoću `LAN` kamere na letelici

Za ispitivanje kašnjenja video-snimka korišćenjem `WebRTC`-a se koriste `JSLAB` skripte i to `test_videocall_server.jsl` i `test_videocall_client.jsl` čiji kod je dat u listinzima 7.6 i 7.5. Skripte osim što omogućavaju video-poziv otvaraju i nove prozore u kojima se prikazuju trenutne vremenske oznake.

```
/**
 * @file Video call client test
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
var capture_type = ['webcam', 'desktop'][0];
var video_id = "5ef5e0fd960336959474ed288e44c7b50c8c4c364f38ca6256c6f79b3ef9f29b
    "
var mic_id = "default";
var tcp_host = "192.168.15.100";
var tcp_port = 5000;

//getWebcams();
//getDesktopSources();
//getMicrophones();

videoCall('client', capture_type, video_id, mic_id, tcp_host, tcp_port);

var win = await openWindowBlank();
win.setSize(500, 90);
```

```
win.setTitle('Timestamp');
win.document.body.innerHTML += '<div id="timestamp" style="padding: 10px; font-size: 36px; text-align: center;">0</div>';
var timestamp_cont = win.document.getElementById('timestamp');

var t = tic;
function frame(timestamp) {
  var timestamp = stringify(round(tocms(t), 0));
  timestamp_cont.innerText = timestamp;
  requestAnimationFrame(frame);
}
requestAnimationFrame(frame);
```

Listing 7.5 - `test_videocall_client.jsl`

```
/**
 * @file Video call server test
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
var capture_type = ['webcam', 'desktop'][0];
var video_id = "5ef5e0fd960336959474ed288e44c7b50c8c4c364f38ca6256c6f79b3ef9f29b"
var mic_id = "default";
var tcp_host = "192.168.15.100";
var tcp_port = 5000;

//getWebcams();
//getDesktopSources();
//getMicrophones();

videoCall('server', capture_type, video_id, mic_id, tcp_host, tcp_port);

var win = await openWindowBlank();
win.setSize(500, 90);
win.setTitle('Timestamp');
win.document.body.innerHTML += '<div id="timestamp" style="padding: 10px; font-size: 36px; text-align: center;">0</div>';
var timestamp_cont = win.document.getElementById('timestamp');

var t = tic;
function frame(timestamp) {
  var timestamp = stringify(round(tocms(t), 0));
  timestamp_cont.innerText = timestamp;
  requestAnimationFrame(frame);
}
requestAnimationFrame(frame);
```

Listing 7.6 - `test_videocall_server.jsl`

Za ispitivanje kašnjenja video-snimka korišćenjem RTSP-a se koriste `JSLAB` skripte i to `ServerRTSP.jsl` i `ClientRTSP.jsl` koje su u potpunosti date u prilozima AA i AB. U sva tri slučaja se vrši prenos videa sa slikom rezolucije `1280` puta `720` piksela i sa `30` `slika u sekundi`.

7.5.2 Rezultati ispitivanja kašnjenja pri korišćenju WebRTC-a

Na slici 7.31 je prikazano 18 nasumično izabranih slika iz video-snimka ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka pri korišćenju `WebRTC`-a. U nastavku je data i tabela 7.3 u kojoj su obrađeni podaci sa slika i date vrednosti kašnjenja.



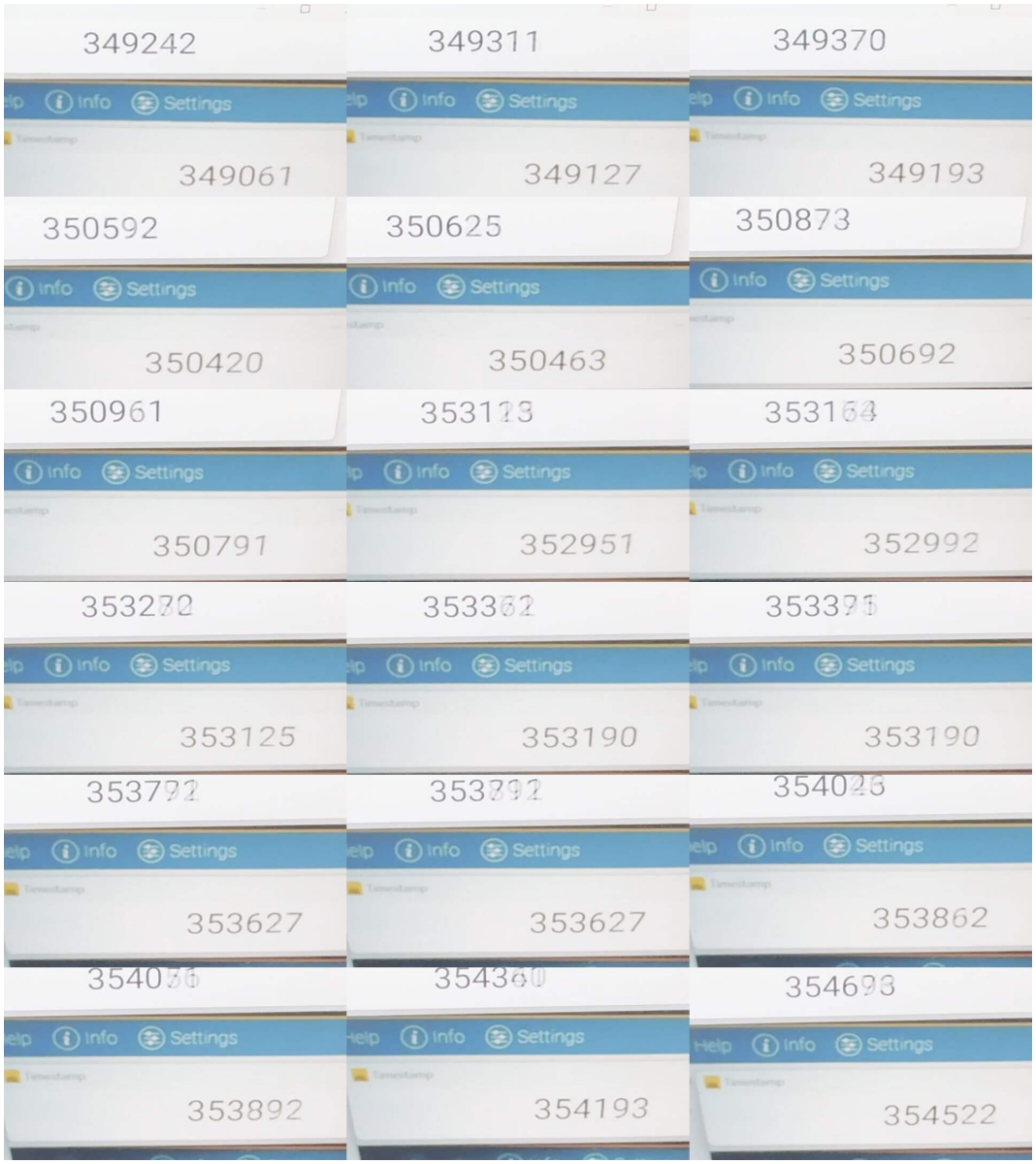
Slika 7.31 - Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka preko `WebRTC`-a

Tabela 7.3 - Izračunata kašnjenja tokom ispitivanja kašnjenja video-snimka preko `WebRTC`-a

RB	Frame	Δt [ms]	RB	Frame	Δt [ms]	RB	Frame	Δt [ms]
1	84	208	2	100	174	3	127	222
4	168	196	5	199	178	6	206	188
7	214	222	8	225	200	9	321	201
10	483	202	11	489	201	12	495	176
13	596	222	14	613	212	15	614	212
16	787	175	17	825	198	18	851	190

7.5.3 Rezultati ispitivanja kašnjenja pri korišćenju RTSP-a

Na slici 7.32 je prikazano 18 nasumično izabranih slika iz video-snimka ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka pri korišćenju RTSP-a. U nastavku je data i tabela 7.4 u kojoj su obrađeni podaci sa slika i date vrednosti kašnjenja.



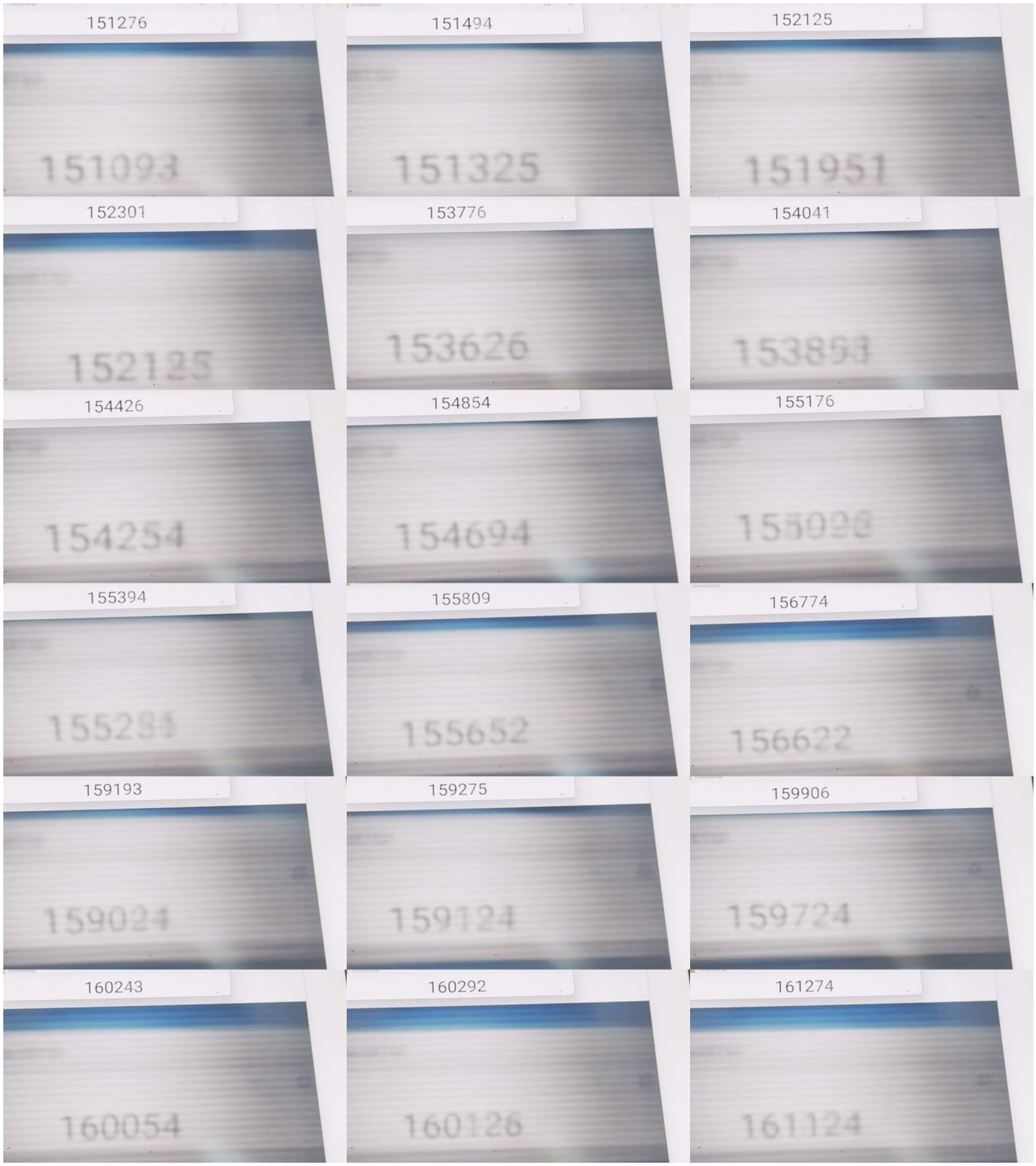
Slika 7.32 - Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka preko RTSP-a

Tabela 7.4 - Izračunata kašnjenja tokom ispitivanja kašnjenja video-snimka preko RTSP-a

RB	Frame	Δt [ms]	RB	Frame	Δt [ms]	RB	Frame	Δt [ms]
1	6	181	2	10	184	3	14	177
4	87	172	5	89	162	6	104	181
7	109	170	8	238	172	9	241	172
10	248	155	11	253	172	12	254	185
13	278	165	14	279	184	15	293	181
16	295	179	17	312	167	18	332	171

7.5.4 Rezultati ispitivanja kašnjenja pri korišćenju RTSP-a i LAN kamere

Na slici 7.33 je prikazano 18 nasumično izabranih slika iz video-snimka ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka pri korišćenju RTSP-a i LAN kamere na letelici. U nastavku je data i tabela 7.5 u kojoj su obrađeni podaci sa slika i date vrednosti kašnjenja.



Slika 7.33 - Slike ekrana sa vremenskim oznakama tokom ispitivanja kašnjenja video-snimka preko RTSP-a i LAN kamere na letelici

Tabela 7.5 - Izračunata kašnjenja tokom ispitivanja kašnjenja video-snimka preko RTSP-a i LAN kamere na letelici

RB	Frame	Δt [ms]	RB	Frame	Δt [ms]	RB	Frame	Δt [ms]
1	5	183	2	18	169	3	56	174
4	67	176	5	155	150	6	171	188
7	194	172	8	220	160	9	239	150
10	252	163	11	277	157	12	325	152
13	480	169	14	485	151	15	523	182
16	543	189	17	546	166	18	605	150

7.5.5 Analiza rezultata

Rezultati ispitivanja kašnjenja pri korišćenju različitih tehnologija za prenos video-snimka prikazani su tabelarno u prethodnim podsekcijama. Na osnovu dobijenih vrednosti kašnjenja, mogu se izvući sledeći zaključci:

1. Za *WebRTC (USB kamera)*:

   - Minimalno kašnjenje: 174 ms
   - Prosečno kašnjenje: 198.7 ms
   - Maksimalno kašnjenje: 222 ms

   Uočen je nešto veći opseg kašnjenja u poređenju sa RTSP. Razlog je složenija arhitektura WebRTC koja, pored enkodiranja i dekodiranja, može uključivati i dodatne mrežne mehanizme za adaptaciju bit rate-a i korekciju grešaka. Ova kašnjenje odgovara kašnjenju od 5 do 6 slika u video-snimku na 30 fps. Za daljinsko pilotiranje letelicom, ovakav nivo kašnjenja može biti prihvatljiv ako brzina letelice nije izuzetno velika i ako su manevri relativno spori.

2. Za *RTSP (USB kamera)*:

   - Minimalno kašnjenje: 155 ms
   - Prosečno kašnjenje: 173.9 ms
   - Maksimalno kašnjenje: 185 ms

   RTSP prenos obično ima manje kašnjenje od WebRTC-a jer se fokusira primarno na prenos audio/video strima, bez kompleksnih mehanizama interakcije (kao što je pregovaranje kanala i korekcija grešaka u realnom vremenu). Rezultati pokazuju da se tipično kašnjenje kreće oko 170 ms, što je osetno manje od 200 ms zabeleženih pri WebRTC-u. Za većinu primena daljinskog pilotiranja, takva latencija je umerena i omogućava operateru relativno pouzdano sagledavanje okoline i situacije u kojoj se nalazi.

3. Za *RTSP (LAN kamera na letelici)*:

   - Minimalno kašnjenje: 150 ms
   - Prosečno kašnjenje: 166.7 ms
   - Maksimalno kašnjenje: 189 ms

U ovom slučaju, kamera je na letelici spojena sa radio modulom direktno putem LAN-a. Generalno, rezultati su vrlo slični prethodnom RTSP slučaju sa USB kamerom. Prosečna vrednost kašnjenja je neznatno niža (166.7 ms naspram 173.9 ms). Ovakvi rezultati pokazuju da LAN kamera može dati nešto manje kašnjenje što je najverovatnije posledica toga što je kamera hardverski optimizovana za slanje video-snimka putem RTSP-a.

Tokom prenosa videa, kašnjenje može da nastane na više tačaka u lancu akvizicije, obrade i prenosa:

- Akvizicija slike na kameri (senzor, autoekspozicija, međumemorija),
- Digitalizacija i enkodiranje (u kamerama ili softverskim kodecima),
- Paketizacija i slanje (putem WebRTC ili RTSP protokola, preko mreže),
- Prenos mrežnim putem (TCP, UDP, lokalno ili bežično),
- Prijem, depaketizacija i dekodiranje na računaru (Getac F110 G6),
- Prikaz na ekranu (upravljanje frame bufferom, osvežavanje ekrana).

Cilj ovog ispitivanja bio je da se uzmu u obzir svi izvori kašnjenja i da se izmeri realna vrednost sa kojom se operater suočava.

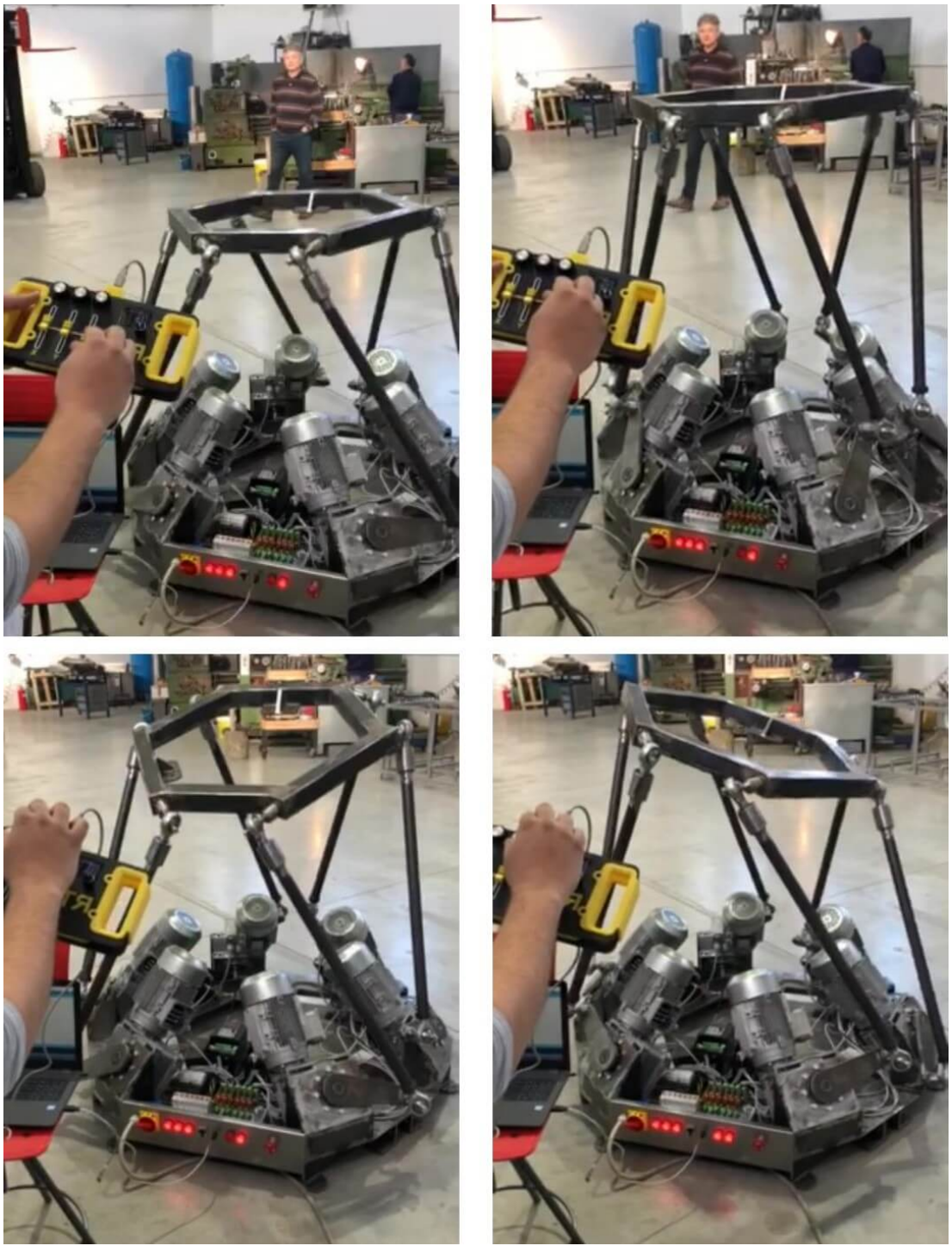
Nakon svih *urađenih ispitivanja i analiziranih rezultata*, mogu se doneti sledeći zaključci:

- Sva prikazana kašnjenja na nivou od oko 150 ms – 220 ms mogu se smatrati *umerenim* u kontekstu daljinskog pilotiranja letelicom (ovo odgovara kašnjenju od između 4 i 6 slika u video-snimku na 30 fps). Budući da se većina letelica koja su predmet istraživanja ne kreće pri ekstremno visokim brzinama, kašnjenje na nivou dve desetine sekunde i dalje omogućava operateru da reaguje blagovremeno.
- Ipak, ako je letelica vrlo brza ili se manevri obavljaju u uskim prostorima i kašnjenje od 150 ms ili 220 ms može biti preveliko. U takvim slučajevima se razmatraju alternativne tehnologije prenosa, brze kamere, manje rezolucije slike, kodeci malog kašnjenja, napredne bežične veze i slično. Za uspešno upravljanje u ovakvim slučajevima se očekuje kašnjenje ne veće od 100 ms.
- I ako nije namenjen za ovakvu primenu, prikazan je WebRTC pre svega zbog svoje jednostavnosti za implementaciju. Ovaj protokol može pružiti dvosmerni prenos i zvuka i video-snimka sa mogućnošću dinamičke adaptacije uslova prenosa. Razvoj ovakvog protokola je posledica potrebe za razgovorima putem interneta i zbog tako široke primene u svakodnevnom životu je predmet konstantnog unapređivanja.
- Korišćenjem LAN kamere na letelici se može smanjiti vreme kašnjenja jer su takve kamere opremljene specijalizovanim hardverom za prenos video-snimka putem RTSP-a.

Ovim ispitivanjima je pokazano da i WebRTC i RTSP uz adekvatno podešavanje mogu da pruže dovoljno dobre performanse video-prenosa za tipične operacije daljinskog pilotiranja u realnom vremenu. Svaka od tehnologija može se dodatno optimizovati (kvalitet enkodera, protok, rezolucija, framerate) da bi se postigli još bolji rezultati u pogledu kašnjenja, što zavisi od konkretne primene i zahteva misije.

7.6 Eksperimentalna validacija celog sistema u stvarnoj veličini

U novembru 2019. godine je prvi put pokrenut mehanizam prototipa simulatora leta u stvarnoj veličini sa oznakom `MP-6DOF-XXX`. Kretanje ovog mehanizma je prikazano na slici 7.34.



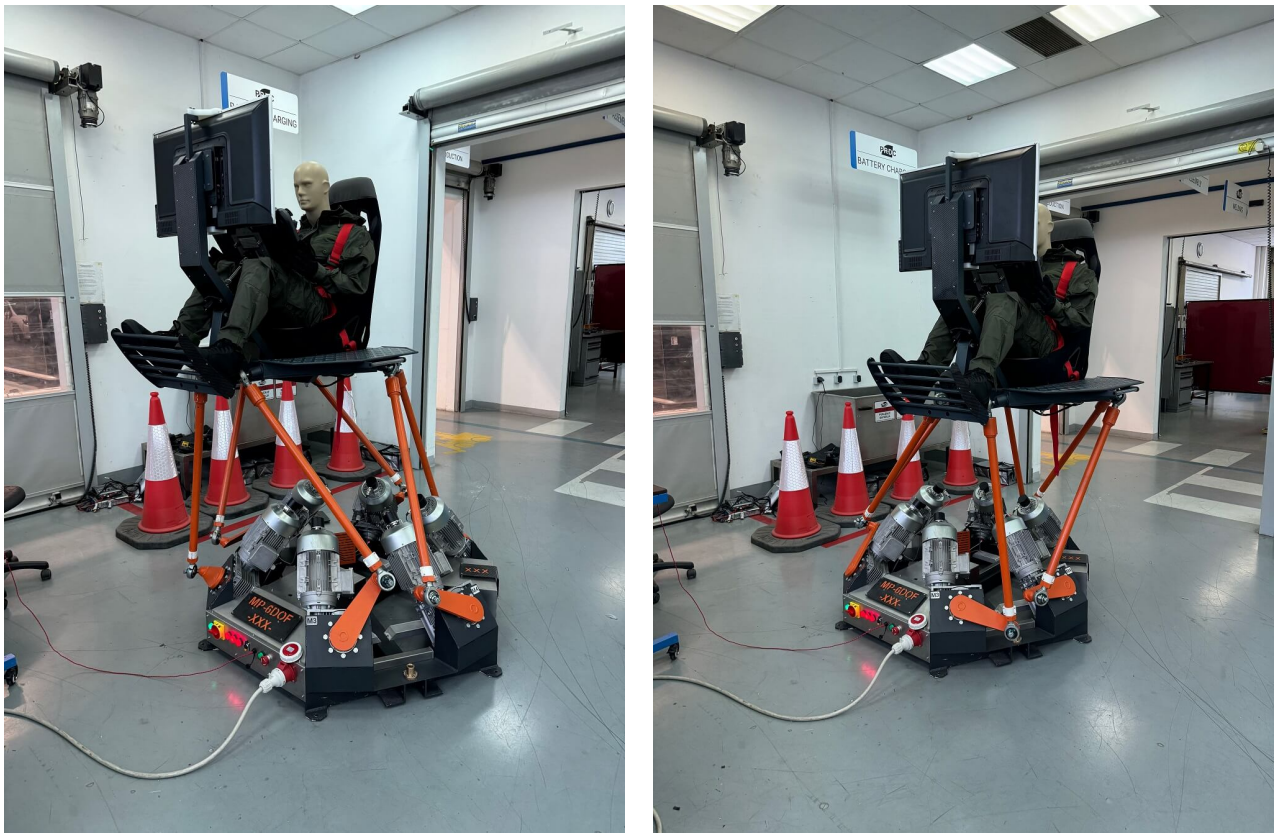
Slika 7.34 - Prvo ispitivanje mehanizma u stvarnoj veličini novembra 2019. godine

U decembru 2019. godine je napravljena i prva verzija pilotske kabine (slika 7.35) sa sedištem i urađena su prva ispitivanja sa teretom na platformi. Za oba prva ispitivanja je korišćen kontroler `MP-6DOF-CTRLR` za kontrolu na osnovu inverzne kinematike.



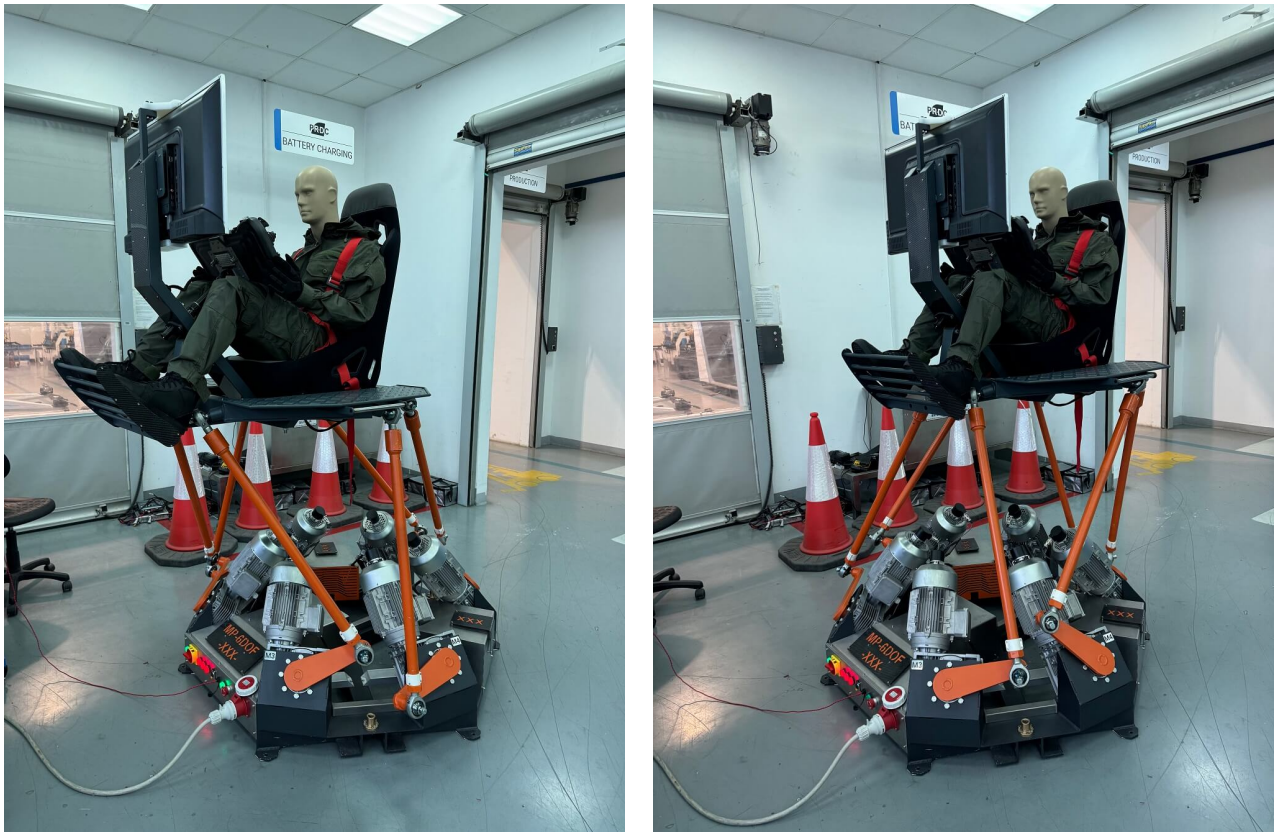
Slika 7.35 - Ispitivanje prototipa simulatora leta decembra 2019. godine

Nakon gotovo pet godina, juna 2024. godine je završena i finalna verzija pilotske kabine sa svim potrebnim elementima da bi se uspešno testirao kompletan sistem. Na slici 7.36 je prikazano translatorno kretanje platforme mehanizma duž x ose.

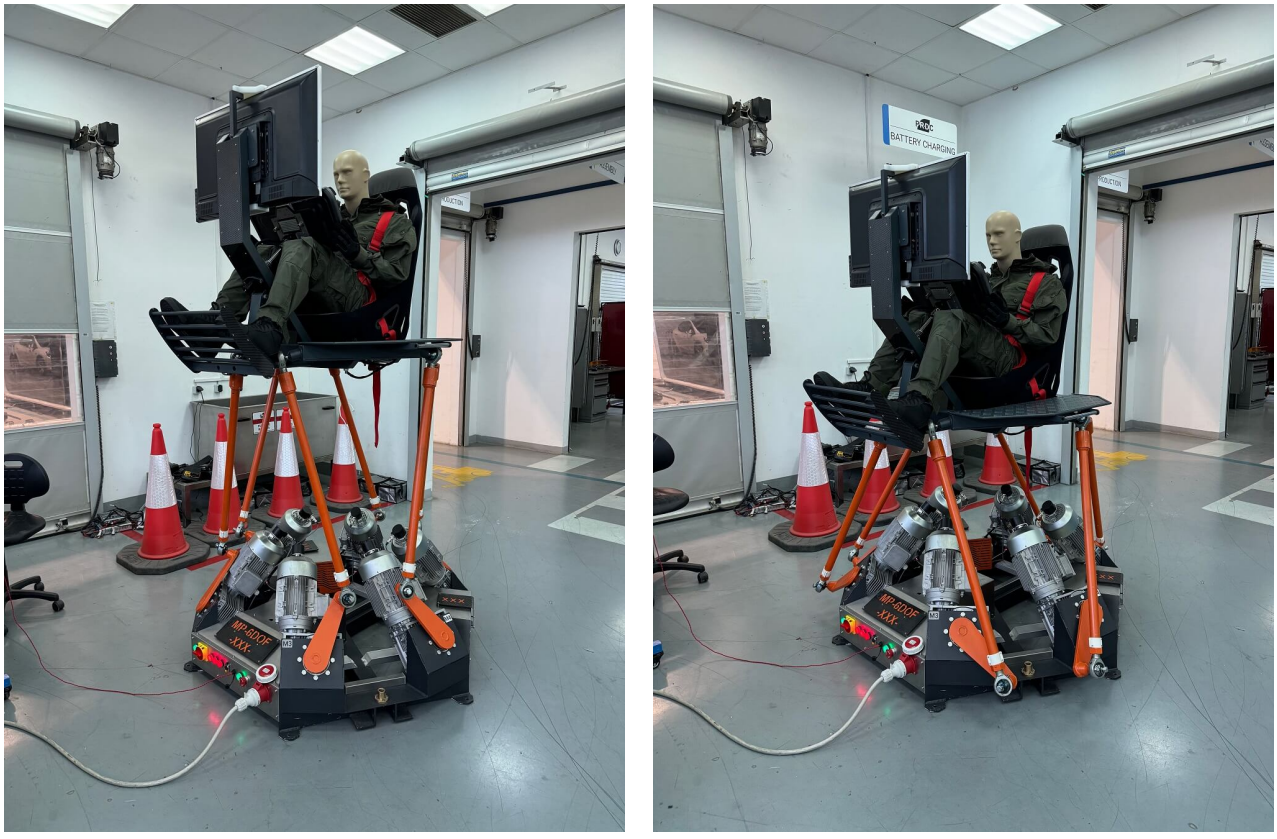


Slika 7.36 - Translacija platforme duž x ose

Na slikama 7.37 i 7.38 je prikazano translatorno kretanje platforme mehanizma duž y i z osa.

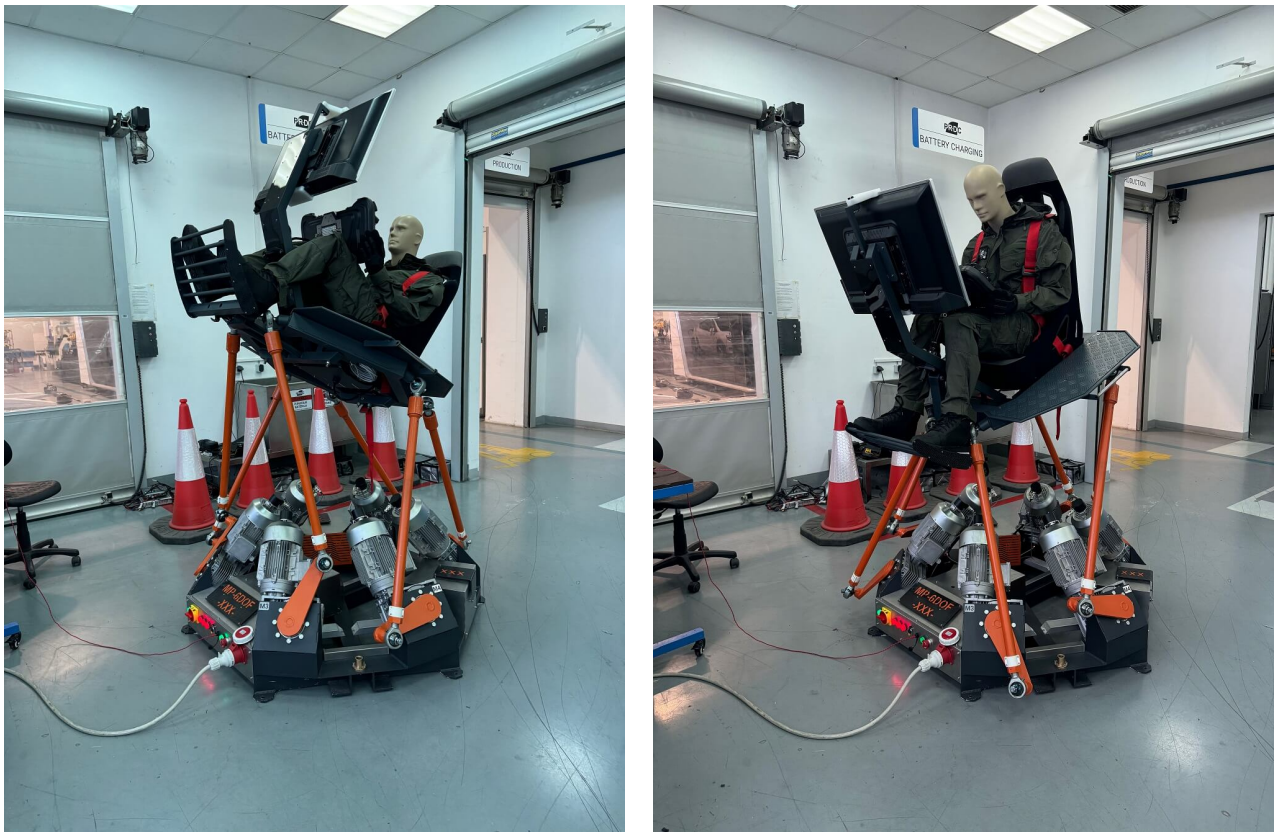


Slika 7.37 - Translacija platforme duž y ose

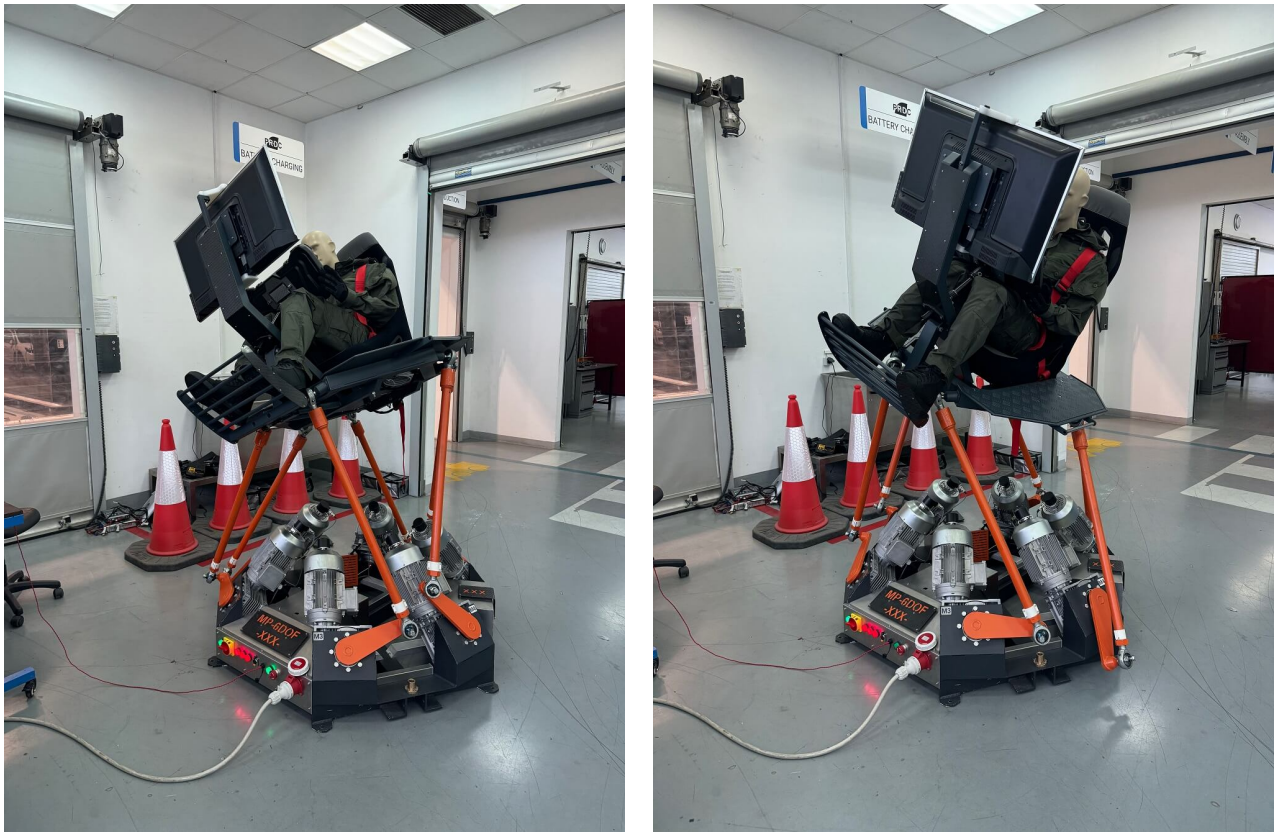


Slika 7.38 - Translacija platforme duž z ose

Na slikama 7.39 i 7.40 je prikazana rotacija platforme mehanizma oko x i y osa.

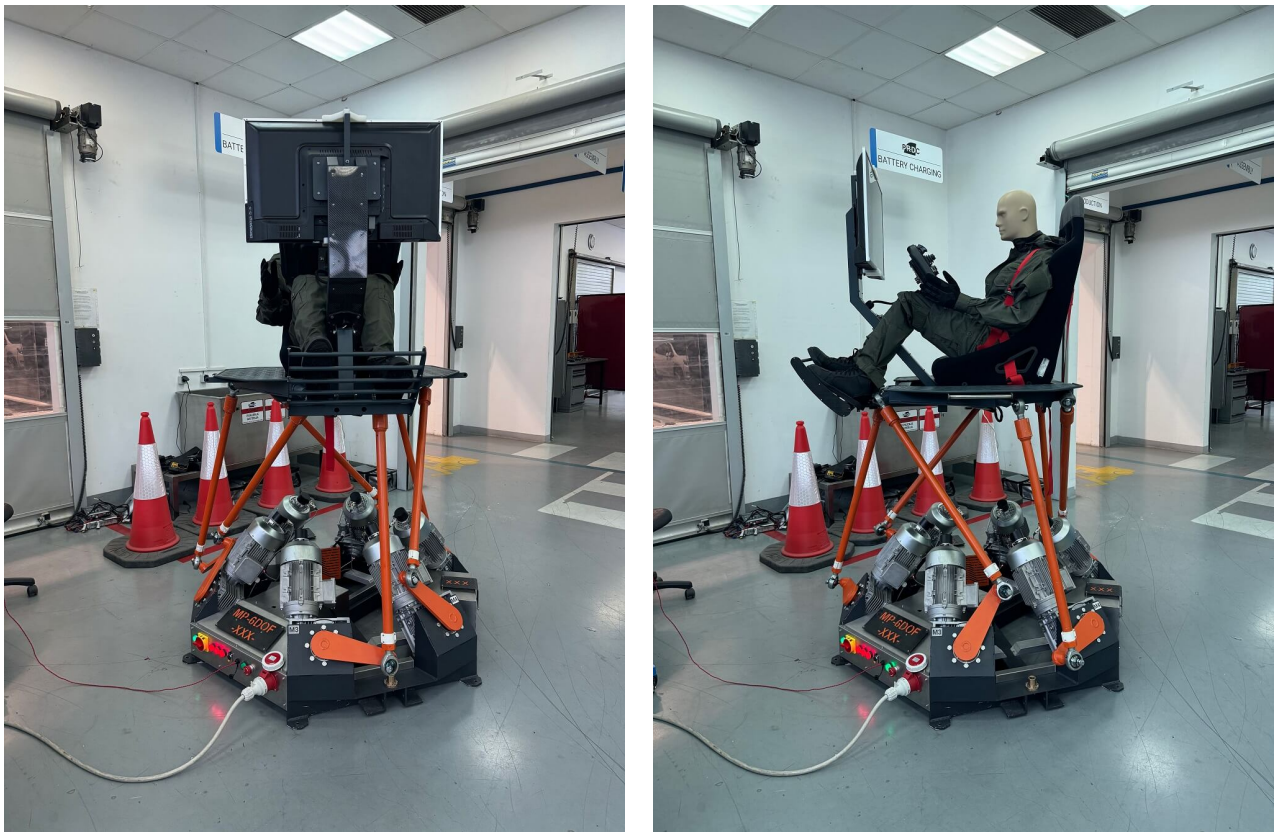


Slika 7.39 - Rotacija platforme oko x ose



Slika 7.40 - Rotacija platforme oko y ose

Na slici 7.41 je prikazana rotacija platforme mehanizma oko z ose.



Slika 7.41 - Rotacija platforme oko z ose

Osim provera kretanja mehanizma, urađena je i provera sa dodavanjem tereta od 100 kilograma na platformu, kao što je prikazano na slici 7.42.



Slika 7.42 - Ispitivanje prototipa simulatora leta sa 100 kilograma tereta

Na osnovu prethodno prikazanog rešenja inverznog kinematičkog problema, implementirane su sve jednačine u program mikrokontrolera upravljačkog sistema prototipa. Kao deo procesa verifikacije, platforma se sve vreme kretala u skladu sa željenim kretanjem, koje je bilo unapred programirano. Sva pretpostavljena fizička ograničenja mehanizma su se takođe pokazala kao korektno sprovedena, a izračunati radni prostor odgovara stvarno dobijenom.

Koristeći različite senzore, moguće je izmeriti preciznost pozicioniranja platforme. Sistem za praćenje kretanja za platformu Stjuart predstavljen je u radu [42] i može se koristiti za verifikaciju. Postoje i kontrolni sistemi zasnovani na viziji, kao što je opisano u [43], koji se takođe mogu koristiti za praćenje i verifikaciju kretanja. Za potrebe validacije je prvo uspešno testirano da li aktuatori postižu željeni ugao na osnovu poslate kontrolne vrednosti. Eksperimenti sa sklopljenim mehanizmom zasnivali su se na slanju kontrolnih vrednosti mehanizmu i merenju položaja i orijentacije platforme kako bi se ona uporedila sa izračunatim vrednostima. Ovaj metod se koristio za testiranje svake pozicije i orijentacije unutar radnog prostora.

Nakon što su svi programi sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi testirani na umanjenom prototipu simulatora leta i nakon što je proračunom dobijena optimalna geometrija, razvijen je i proizveden funkcionalni prototip mehanizma simulatora leta u punoj veličini i sa dovoljnom preciznošću da odgovara optimizacijom dobijenim vrednostima geometrijskih parametara. Ovaj prototip pristupačnog simulatora leta sa električnim rotacionim aktuatorima i optimizovanom geometrijom za simulaciju leta je kao deo kompletnog sistema, ispitan u julu 2024. godine kao što je prikazano na slici 7.43.



Slika 7.43 - Validacija kompletnog sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi

Tokom poslednjeg eksperimenta je korišćen kompletan sistem i pokazano je da je moguće uspešno realizovati sistem koji je bio predmet istraživanja u okviru ove disertacije. Pilot je sa simulatora leta, u realnom vremenu, uspešno upravljao daljinski upravljanom letelicom i pritom se mehanizam simulatora leta kretao na odgovarajući način. Osim upravljanja pravom letelicom, testirano je i upravljanje virtuelnom letelicom. Sistem se može iskoristiti i za ponovno simuliranje određenog leta nakon njegovog snimanja kako bi operater osetio šta se sa letelicom dešavalo tokom tog konkretnog leta.

8

Diskusija i predlog za nastavak istraživanja

8.1 Diskusija

Zbog nedostatka realističnog čulnog doživljaja i prostorne orijentacije tokom upravljanja daljinski upravljanom letelicom može doći do otežanog upravljanja i do povećanja rizika od pojave nepredviđenih situacija tokom leta koje je još teže rešiti daljinskim putem. Zbog toga se javila potreba za naprednim sistemima koji daljinsko pilotiranje čine što sličnijim onom neposrednom, iz kabine letelice. Osnovna motivacija razvoja sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi jeste potreba da se operateru bespilotne letelice pruža osećaj orijentacije i kretanja koji je uporediv sa onim koji bi imao kada bi se stvarno nalazio u letelici, što može značajno uticati na bezbednost, uspešnost i efikasnost izvršenja misija.

U ovoj disertaciji su nakon uvoda, naučne zamisli disertacije, pregleda stanja relevantne tehnike, prikaza dosadašnjih relevantnih istraživanja, definisane bespilotna letelica i simulator leta koji su uz pilotsku kabinu tri osnovna elementa sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi. Pritom je, nakon analize mogućih rešenja, za mehanizam koji omogućava kretanje pilota izabrana *Stjuartova platforma* sa obrtnim aktuatorima (*6RSS tip mehanizma*). Ovaj tip mehanizma je izabran zbog svoje jednostavnosti za simulaciju, proizvodnju i održavanje.

Ključni inženjerski izazov u okviru istraživanja jeste proračun optimalnih geometrijskih parametara ovog mehanizma, što je uspešno realizovano primenom genetskog algoritma. Implementiran je i primenjen *realno kodirani mešovito celobrojni genetski algoritam sa ograničenjima*. Ipak, u realnim uslovima može doći do situacija kada simulator zbog ograničenja opsega kretanja ne može da postigne zadati položaj. Tada se koristi jedan od filtera kretanja (algoritama ispiranja, *washout*), koji obezbeđuje osećaj da je ostvareno pravo kretanje, iako je ono delimično filtrirano. Gledano sa stanovišta pilotovog subjektivnog doživljaja, neophodno je pronaći dobar kompromis između realizma i filtriranja kretanja. Dalja istraživanja mogla bi obuhvatiti detaljno psihofizičko testiranje različitih varijacija filtera kretanja.

Za numeričko rešavanje direktnog kinematičkog problema je korišćena *Njutnova metoda* na osnovu izvedene *Jakobijeve matrice* za čitav mehanizam. Može se primeniti i neki napredniji metod za iterativno rešavanje sistema nelinearnih jednačina kako bi se do rešenja direktnog kinematičkog problema došlo brže sa određenom preciznošću.

Izrađeni su prototipovi, jedan umanjeni za inicijalna ispitivanja i jedan u stvarnoj veličini,

što je eksperimentalno potvrdilo da su proračuni i algoritmi osmišljeni u ovoj disertaciji validni i u stvarnim uslovima. Ovakav vid validacije jedan je od ključnih aduta disertacije, jer pokazuje da istraživanje nije ostalo na teorijskom nivou, već je sistem materijalizovan i testiran.

Sa softverske strane, disertacija obuhvata programe razvijene u programskim jezicima `JavaScript`, `GNU Octave`, `C`, `C++` i `Python`. Koristi se nekoliko velikih biblioteka kao što su `CGAL` za određivanje radnog prostora ili `FFmpeg` za kodiranje video snimka. Ovakva raznovrsna arhitektura nameće organizacione izazove u pogledu održavanja i saradnje sa drugim istraživačima. Dalji napredak mogao bi da se ogleda u pojednostavljenju ove arhitekture.

Tokom istraživanja je razvijeno nekoliko bespilotnih letelica od kojih je svaka imala određeni zadatak. Određeno je koje to elemente bespilotna letelica treba da ima kako bi bilo moguće da se koristi u okviru ovog sistema. Od parametara kretanja koji su izmereni direktno zavisi i kakve će biti performanse celog sistema, pa je tako potrebno imati što bolje senzore i algoritme za precizno i brzo određivanje parametara kretanja. Izvor grešaka u sistemu može biti pojava velikih vibracija na letelici zbog kojih senzori nisu u stanju da izmere parametre kretanja. Takođe, digitalna bežična komunikacija kojom se ovi parametri šalju simulatoru leta treba da omogućava stabilnu vezu i minimalno kašnjenje. Senzori (uključujući kamere) sa algoritmima za obradu podataka i moduli za bežičnu komunikaciju se konstantno unapređuju, pri čemu brzina unapređivanja direktno zavisi od potražnje za ovim proizvodima, koja je u ovom trenutku na istorijskom maksimumu.

Projektovana je i proizvedena nova pilotska kabina za daljinsko upravljanje bespilotnom letelicom sa simulatora leta. Ideja je bila da se zadrži daljinski upravljač sa svojim programom za upravljanje, na ovaj deo sistema su operateri već navikli, i da se dodavanjem još jednog ekrana samo prošire mogućnosti za prikaz podataka operateru. Analizirane su i mogućnosti upotrebe sistema za virtuelnu realnost u nastavku istraživanja, koji bi dodatno doprineli povećanju mogućnosti za prikaz podataka operateru. U disertaciji je prikazana integracija simulatora leta, letelice i pilotske kabine u funkcionalan zatvoren sistem.

Funkcionalnost sistema za daljinsko pilotiranje letelicom sa parametrima kretanja u povratnoj sprezi potvrđena je eksperimentima u realnom vremenu. Inercijalni podaci i slika sa kamere prenosili su se bežično do simulatora leta, omogućavajući pilotu da oseti približno realno ubrzanje i orijentaciju. Iako je rešenje koncipirano da bude za ovu specifičnu primenu, moguće je sa eventualno dodatnim usavršavanjem mehanike, elektronike i softvera proširiti spektar primene.

Na kraju treba istaći da je, uz sva teorijska i eksperimentalna rešenja, neophodna kontinuirana provera usklađenosti sa savremenim trendovima na tržištu bespilotnih letelica, koje se veoma brzo razvija. Kroz redovno osvežavanje baze dronova i unapređivanje algoritama, ovaj sistem kao novi proizvod može dugo ostati konkurentan i relevantan.

Na osnovu dosadašnjih prezentovanja sistema se može zaključiti da potencijalni klijenti razumeju naučni i praktičan doprinos ove disertacije i da postoji potražnja za ovakvim sistemom kao novim proizvodom na tržištu. U odnosu na postojeće, ovaj pristupačan simulator leta je proračunat tako da ima geometriju optimalnu za simuliranje leta, bespilotne letelice su prilagođene kako bi mogle da funkcionišu u okviru ovog sistema a pilotska kabina je potpuno nova.

8.2 Predlog za nastavak istraživanja

S obzirom na dobijene rezultate i razmatranja u disertaciji, nameću se sledeći pravci daljeg istraživanja i razvoja:

- Unapređenje programa - u pogledu mogućnosti koje korisniku pružaju.
- Smanjenje potrebe za upotrebom algoritama ispiranja - dodavanjem dodatnih stepeni slobode mogla bi značajno da se smanji potreba za upotrebom algoritama ispiranja.
- Unapređenje algoritma za proračun optimalne geometrije mehanizma simulatora leta - analizirati mogućnost za dalja unapređenja algoritma za proračun optimalnih parametara u cilju efikasnijeg proračuna.
- Uvođenje naprednih algoritama veštačke inteligencije - algoritmi veštačke inteligencije mogli bi se koristiti za predviđanje i prilagođavanje kretanja simulatora u realnom vremenu.
- Nova eksperimentalna ispitivanja - merenje stvarnih deformacija, vibracija i zazora prilikom dugotrajnog rada platforme, kao i upoređivanje dobijenih podataka sa rezultatima metode konačnih elemenata. Ovo će omogućiti preciznije modele i eventualno modifikovanje mehanizama u skladu sa realnim ponašanjem materijala.
- Pojednostavljivanje arhitekture programa za proračun i analizu - analiza novih programskih jezika i biblioteka koji bi doprineli pojednostavljivanju arhitekture programa u pogledu broja različitih programskih jezika i biblioteka koje se koriste.
- Integracija novih tipova senzora na letelice i poboljšanje bežičnog prenosa podataka - razvoj stabilnijih i bržih komunikacionih protokola, smanjenje kašnjenja u sistemima s bežičnim prenosom slike i inercijalnih podataka, te uvođenje novih senzora za naprednu kontrolu i preciznije prikupljanje podataka.
- Proširenje spektra primene - analiza mogućnosti korišćenja ovakvog mehanizma za virtuelnu realnost, simulaciju brodova ili drumskih vozila, medicinsku rehabilitaciju i druge oblasti gde je potrebno realistično kretanje platforme.

Proširenje primene na druga vozila je prirodan nastavak ovog istraživanja i otvara mnogo mogućnosti za buduća istraživanja i inovacije.

9

Zaključak

U ovoj disertaciji je kao naučni doprinos, prikazan **razvoj mehanizma platforme pristupačnog simulatora leta koji je optimizovan za simuliranje leta i koji zajedno sa ostalim podsistemima može da se primeni pri pilotiranju daljinski upravljanom letelicom u okviru povratne sprege. Ovako formirana povratna sprega sa simulatorom leta omogućava da pilot pri pilotiranju daljinski upravljanom letelicom ima približno isti osećaj kao da se nalazi u samoj letelici.**

Ovaj novi metod za pilotiranje daljinski upravljanom letelicom, koji se delimično zasniva i na osećaju kretanja same letelice, se bazira na razvijenom pristupačnom simulatoru leta čija je geometrija optimizovana za simuliranje leta. Za razliku od postojećih simulatora leta, ovde prikazani je mehatronički jednostavan i isključivo se sastoji iz standardnih industrijskih komponenata koje su široko dostupne na tržištu, što ga čini pristupačnim u odnosu na postojeće. Eksperimentalno je dokazano na konceptu sistema u stvarnoj veličini da je moguće vršiti merenja na letelici, zatim putem bežične komunikacije podatke poslati do simulatora leta i generisati odgovarajuće upravljanje tako da se pilot i pored neminovnih kašnjenja podataka približno oseća kao da se nalazi u samoj letelici. Tokom izrade idejnog rešenja sistema i pripreme eksperimenta se sve vreme težilo tome da se raznim metodama omogući da kašnjenje podataka na kraju bude minimalno.

U okviru disertacije su **uspešno potvrđene sve polazne hipoteze i donose se sledeći zaključci**:

- U poglavlju 2 je istaknuto da se izmeštanjem pilota iz letelice gubi mogućnost da on istovremeno prima razne informacije preko svih svojih čulnih organa. Na osnovu toga se zaključuje da pilotu daljinski upravljanje letelice nije dovoljno da samo vidi sliku sa kamere na letelici kako bi formirao najbolje moguće upravljanje unutar svojih mogućnosti jer bi korišćenjem i ostalih svojih čulnih organa mogao da generiše bolje upravljanje.

- U poglavljima 6 i 7 je dokazano da je izborom i ugradnjom odgovarajućih senzora zajedno sa algoritmima za obradu podataka moguće u realnom vremenu precizno vršiti merenje ubrzanja i orijentacije daljinski upravljane letelice u prostoru.

- U poglavljima 6 i 7 je dokazano da je pomoću digitalnog radio modula moguće uspostaviti bežičnu komunikaciju između pilota i letelice. Ostvarena veza je stabilna i sa dovoljnim protokom za prenos svih potrebnih informacija u oba smera, od letelice ka simulatoru leta i obrnuto.

- Eksperimentalno je dokazano na prototipu sistema u stvarnoj veličini, sa rezultatima prikazanim u poglavlju 7, da neizbežno kašnjenje slike i kretanja simulatora leta u odnosu

na stvarno kretanje letelice ne utiče na uspešnost pilotiranja.

- U poglavljima 4 i 5 je prikazan izbor mehanizma simulatora leta a potom je urađen i proračun optimalnih geometrijskih parametara mehanizma simulatora leta. Na kraju je, eksperimentalno dokazano, sa rezultatima prikazanim u poglavlju 7, da je izborom odgovarajućeg mehanizma i optimizacijom njegove geometrije dobijen adekvatan simulator leta.

- Uspešno je primenjen genetski algoritam, i to realno kodirani mešovito celobrojni genetski algoritam sa ograničenjima, za proračun optimalnih geometrijskih parametara mehanizma simulatora leta kao što je prikazano u poglavlju 5. U okviru ovog proračuna je algoritam uspešno primenjen na složeni nelinearan problem.

- U poglavlju 6 je dokazano da je moguće formirati matematičke modele letelice i pilota kako bi se formiralo predupravljanje za simulator leta.

- Prikazanim poslednjim eksperimentom u 7 je dokazano da je moguće u realnom vremenu formirati upravljanje za simulator leta na osnovu informacija koje su bežičnom komunikacijom dobijene sa daljinski upravljane letelice.

Za potrebe realizacije ove disertacije, potvrđivanja svih polaznih hipoteza i na kraju formiranja novog proizvoda, **postignuti su i sledeći stručni doprinosi**:

- Analiziran je postavljeni problem i izrađeno je idejno rešenje sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

- Urađen je kompletan pregled stanja relevantne tehnike i prikaz dosadašnjih relevantnih istraživanja za pojedinačne delove sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi. Pregled postojeće naučne literature i ostvarenih rezultata, praktičnih, naučnih, tehničkih i drugih dostignuća je dat u poglavlju 3.

- Analizirani su postojeći simulatori leta za avione, urađen je pregled i izvršen je odabir metode za proračun optimalne geometrije mehanizma simulatora leta za simuliranje leta.

- Izvršena je analiza postojećih bespilotnih letelica, urađena je njihova klasifikacija i analizirani su podsistemi neophodni da bi letelica bila deo sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

- Urađen je pregled značajnih standarda i važećih zakona u vezi sa elementima sistema.

- Analizirani su alati za trodimenzionalno modeliranje i kompjutersku grafiku koji su neophodni za realizaciju disertacije. Takođe, analizirani su i kompozitni materijali, 3D štampanje i skeniranje kao i drugi proizvodni procesi i obrade materijala koji se koriste za izradu elemenata sistema. Za električni sistem su analizirane štampane ploče, mikrokontroleri i komunikacije, bežični prenos podataka i programiranje digitalnih procesora. Zatim je urađen pregled dostupnih biblioteka za manipulaciju i obradu video signala kao i analiza mogućnosti upotrebe sistema za virtuelnu i proširenu realnost sa haptičkim uređajima.

- Implementiran je potpuno novi program `JSLAB` kao okruženje za numeričke proračune, koji se pretežno koristio u realizaciji disertacije. Program `JSLAB` je nastao zbog potrebe da se proračuni u ovoj disertaciji izvrše u programskom jeziku koji može da se koristi i u

kasnijim fazama projekta. Program je otvorenog koda i javno je dostupan na `GitHub`-u (za razliku od programa kao što je *Matlab*).

- Razvijen je i uveden novi metod za definisanje geometrije mehanizma simulatora leta zasnovanog na *Stjuartovoj platformi sa obrtnim aktuatorima*. Definisanje geometrije je bazirano na ukupno 12 geometrijskih parametara koji zajedno sa izvedenim i konstantnim parametrima jednoznačno definišu geometriju celog mehanizma simulatora leta.

- Rešeni su inverzni i direktni kinematički problemi za *Stjuartovu platformu sa obrtnim aktuatorima* i napisani su programi koji se baziraju na ovim rešenjima. Za rešavanje inverznog kinematičkog problema je korišćen geometrijski metod, dok se za rešavanje direktnog kinematičkog problema koristi *Njutnova metoda* bazirana na izvedenoj *Jakobijevoj matrici* za iterativno rešavanje sistema nelinearnih jednačina.

- Izvedene su jednačine za određivanje brzine i ubrzanja svih značajnih tačaka *Stjuartove platforme sa obrtnim aktuatorima*.

- Formiran je novi metod za proračun elastičnosti konstrukcije i posledično greške položaja i orijentacije platforme mehanizma u realnom vremenu zasnovan na metodi konačnih elemenata sa Lagranžovim množiocima (za implementaciju zglobova mehanizma). Na osnovu ove metode je moguće uraditi dimenzionisanje segmenata mehanizma i aktuatora. Urađeno je izvođenje jednačine konačnih elemenata na osnovu *Hamiltonovog principa* i zatim su za element tipa *Ojler-Bernulijeva greda*, kao linijski element u prostoru, izvedene jednačine za proračun pomeranja u čvorovima elemenata. Ovaj proračun je kompletno implementiran u programu `JSLAB`. Za potrebe izvođenja jednačina je implementirana simbolična matematika u `JSLAB`-u na osnovu `Pyodide` i `SymPy` biblioteka.

- Rešeni su inverzni statički i inverzni dinamički problem *Stjuartove platforme sa obrtnim aktuatorima* u cilju određivanja potrebnih obrtnih momenata aktuatora.

- Definisani su indikatori performansi mehanizma kao što su maksimalni ostvarivi parametri i zapremina radnog prostora na osnovu kojih je moguće uraditi proračun optimalne geometrije mehanizma za simuliranje leta.

- Formiran je novi algoritam pod nazivom *SpaceSearch* koji predstavlja numerički metod za efikasno određivanje maksimalnih ostvarivih parametara i granica radnog prostora mehanizma. Ovaj algoritam je osmišljen za efikasno istraživanje i analizu N-dimenzionalnih prostora kroz rekurzivnu deobu na manje hiperkocke. Ovaj metod je posebno efikasan za identifikaciju oblasti unutar višedimenzionalnog prostora koje zadovoljavaju specifičan uslov (definisan u okviru funkcije uslova), što je ključno u različitim oblastima kao što su računarska geometrija, optimizacija i simulacija. Pokazano je da je ovaj algoritam daleko vremenski i procesorski efikasniji od iscrpnih metoda. Algoritam je implementiran u programu `JSLAB`.

- Formirana je nova procedura za određivanje radnog prostora mehanizma sa svim ograničenjima. Nakon određivanja diskretnih tačaka u prostoru koje mehanizam može da postigne pomoću *SpaceSearch* algoritma, određuje se granična površ koja obuhvata sve tačke radnog prostora. Kako je ovo u opštem slučaju konkavna površ prvo se vrši *Deloneova triangulacija* tačaka kojom se dobija konveksna površ a zatim se metodom pod nazivom *Alfa oblici* određuje konkavna površ. Ova procedura je implementirana korišćenjem `C++` biblioteke `CGAL` u programu `JSLAB`.

- Poslednji, 12. geometrijski parametar za slučaj kada se koristi fiksni ugao ugradnje sfernih zglobova gornjih poluga može značajno pojednostaviti izradu mehanizma. Nakon dodavanja ovog parametra su unete potrebne izmene u funkcije kojima se proveravaju ograničenja mehanizma.

- Izveden je analitički proračun minimalnog rastojanja između donjih poluga mehanizma kojim se proverava da li može doći do sudaranja između poluga mehanizma za određene vrednosti geometrijskih parametara mehanizma.

- Napisane su `C++` i `JSLAB` klase za proračun svih karakteristika mehanizma simulatora leta i za trodimenzionalnu vizualizaciju njegovog kinematskog modela.

- U programu `JSLAB` je implementiran kompletan realno kodirani mešovito celobrojni genetski algoritam sa ograničenjima. Tokom procesa proračuna optimalnih geometrijskih parametara mehanizma je za ograničenja korišćen *metod penalizacije* dok su se najbolje jedinke nepromenjene prenosile u sledeću generaciju na osnovu *eltizma*. Generisanje populacije se vršilo uniformnom raspodelom, za izbor jedinki za ukrštanje se koristila *turnirska selekcija sa dve jedinke* dok se ukrštanje baziralo na *Laplasovoj raspodeli*. Mutacija je implementirana korišćenjem *raspodele stepenovanja*. Implementirane su funkcije za *prikaz toka proračuna* i *proveru kriterijuma za prekid procesa*. Zbog toga što neke promenljive mogu imati samo celobrojne vrednosti, napisana je funkcija za *ograničavanje na celobrojne vrednosti*. Vrednost za penalizaciju na osnovu evaluacije funkcije ograničenja je dobijena normalizacijom vrednosti. Ovaj složen oblik algoritma se pokazao kao uspešan u pronalaženju globalnog minimuma funkcije prilagođenosti. Razmatrani proračun je vremenski efikasan i može iskoristiti mogućnosti paralelnog izvršavanja na procesorima što podržava i implementacija u programu `JSLAB`.

- Definisane su funkcije ograničenja i prilagođenosti kojima se vrši pronalaženje rešenja tokom procesa proračuna optimalne geometrije mehanizma za simuliranje leta. Određeni su parametri kojima se podešava algoritam proračuna. Kompletan postupak proračuna je implementiran u programu `JSLAB`.

- Projektovan je i proizveden umanjeni prototip simulatora leta sa oznakom `MP-6DOF-M`. Pored konstrukcije je razvijen i njegov električni sistem sa svim programima neophodnim za njegovu kontrolu. Napravljena je tehnička dokumentacija za njegovu proizvodnju.

- Definisan je i implementiran novi binarni komunikacioni protokol `PRDC_SCBP`. Ovaj protokol je dizajniran za serijsku komunikaciju između uređaja, omogućavajući pouzdanu razmenu podataka uz mehanizme za obeležavanje početka i kraja paketa, enkodiranje specijalnih znakova i proveru integriteta podataka putem kontrolne sume. Protokol je implementiran u programskim jezicima `JavaScript`, `GNU Octave` i `Arduino` (`C++`).

- Izvršena je materijalizacija mehanizma, u smislu pretvaranja kinematičkog modela u trodimenzionalni model sa svim propratnim informacijama na osnovu kojih ga je moguće izraditi. Za dimenzionisanje pojedinačnih delova je korišćen kriterijum relativno malih elastičnih deformacija konstrukcije. Određene su odgovarajuće cevi i materijal od kojih su one izrađene tako da je na kraju dobijeno maksimalno pomeranje manje od usvojenog maksimalno dozvoljenog i veoma značajno je što je dobijeni mehanizam proizvodljiv.

- Projektovan je i proizveden prototip simulatora leta u stvarnoj veličini sa oznakom `MP-6DOF-XXX`. Pored konstrukcije je razvijen i njegov električni sistem sa svim programima neophodnim za njegovu kontrolu. Napravljena je tehnička dokumentacija za njegovu proizvodnju (električna šema, tehnički crteži i modeli). Definisani su parametri za programiranje frekventnih regulatora, definisana su značenja kodova statusa i definisane su poruke za komunikaciju sa mehanizom.

- Razvijene su nove elektronike (ploče) `Servo Board` i `Control Board` kako bi se omogućila istovremena kontrola šest aktuatora i time rad prototipa `MP-6DOF-XXX`.

- Napravljen je novi beskontaktni apsolutni senzor ugla sa elektronikom `Angle Sensor` sa kojim se komunicira putem robusne `Polu-dupleks RS485` komunikacije. Na elektronici se koriste konektori otporni na vibracije.

- Projektovani su i proizvedeni pomoćni alati za jednostavnije i preciznije zavarivanje, sklapanje i za kalibraciju mehanizma pomoću tehnologije 3D štampanja.

- Formirana je nova procedura kojom se programskim jezikom `JavaScript` iz `JSLAB`-a upravlja programskim paketom `FreeCAD` u cilju formiranja trodimenzionalnog *CAD modela* mehanizma. Formirana je klasa `FreeCADApplication` i radni prostor `ExternalAPI` korišćenjem programskog jezika `Python`. Ovakav model je veoma koristan prilikom verifikacije konfiguracije ali i za integraciju simulatora leta sa ostalim sistemima u fazi konceptualnog dizajna.

- Projektovana je i urađena modifikacija standardnih zglobnih ležaja koji se koriste kao sferni zglobovi tako da je na kraju moguće ugaono pomeranje u zglobovima povećano sa $\pm 15^{\circ}$ na $\pm 37^{\circ}$.

- Napisani su svi potrebni programi, oni za mikrokontrolere pisani programskim jezikom `C++` i desktop aplikacije pisane u programskim jezicima `JavaScript`, `C` i `C++`. Svi ovi programi zajedno omogućavaju upotrebu simulatora leta u okviru sistema ali i kao zasebnog proizvoda.

- Napisan je novi program `MotionPlatformCI` koji ima napredan grafički korisnički interfejs za kontrolu kretanja mehanizma i praćenje njegovog stanja.

- Napravljen je novi simulator leta sa šest stepeni slobode koji je kao zaseban proizvod performansama i ekonomski konkurentan na tržištu. Čak se i umanjeni prototip kao proizvod može ponuditi na tržištu pre svega za edukativne svrhe i istraživanja u domenu robotike i programiranja.

- Projektovan je i proizveden kontroler inverzne kinematike `MP-6DOF-CTRLR`, koji preko `USB` komunikacije šalje mehanizmu komande za kontrolu.

- Projektovan je i proizveden kontroler direktne kinematike `MP-6DOF-M-CTRLR`, koji preko `USB` komunikacije šalje mehanizmu komande za kontrolu.

- Definisan je osnovni filter kretanja (ispiranja) za slučajeve kada simulator ne može da dostigne određenu poziciju zbog zasićenja nekih stepeni slobode, tako da se ipak obezbedi osećaj kao da je ostvareno pravo kretanje.

- Nabavljena je jedna, a projektovano je i proizvedeno još pet bespilotnih letelica koje su korišćene u ovom istraživanju. Pri projektovanju bespilotnih letelica za ovaj sistem je

akcenat stavljen na električni sistem (sa potrebnim programima) za merenje parametara kretanja i bežičnu komunikaciju za slanje podataka do simulatora leta.

- Razvijen je sistem koji omogućava bežični prenos slike sa kamere koja se nalazi na bespilotnoj letelici i informacija o kretanju letelice do upravljačke jedinice simulatora leta. Prenos informacija je omogućen radio komunikacijom dok su orijentacija i ubrzanje izmereni pomoću inercijalnog mernog sistema. Sistem takođe koristi dodatne senzore kao što su `GNSS` prijemnik, magnetometar i barometar čime se poboljšava preciznost merenja parametara kretanja letelice.

- Analiziran je autopilot program `ArduPilot` kao najnapredniji, najpotpuniji i najpouzdaniji softver za autopilot otvorenog koda koji je javno dostupan. U cilju proširivanja funkcionalnosti je analizirano kompajliranje programa, definisanje nove ploče (hardver) i standardni putevi za proširenje programa. Program je moguće kompajlirati i za testiranje softvera u petlji (`SITL`) i moguće je koristiti njegove biblioteke za programiranje pomoćnih periferija za autopilot pomoću potprograma `AP_Periph`.

- U programu `JSLAB` je implementirano čitanje dnevnika leta autopilota `ArduPilot`.

- Implementirana je nova poruka u okviru `MAVLink` komunikacionog protokola koja omogućava slanje samo potrebnih parametara za sistem čime se značajno smanjuje potreban protok putem radio veze. Nova implementirana poruka `BASIC_REMILP` zahteva samo 55% protoka koji je potreban da bi se isto uradilo sa standardnim `MAVLink` porukama.

- Implementirane su virtuelne letelice (bazirane na programima `FlightGear`, `AirSim`, `ArduPilot SITL` ili kompatibilne sa `MAVLink` komunikacionim protokolom kao što je `VehicleSimulator`) pomoću kojih je moguće vršiti obuku operatera ali i testirati programe, i autopilota i daljinskog upravljača.

- U programu `JSLAB` je implementirano povezivanje sa letelicom putem `MAVLink` komunikacionog protokola i prikazivanje lokacije i ugla skretanja letelice na 2D mapi sveta kao i prikaz osnovnih parametara letelice na primarnom ekranu leta. Ovo je najbitniji instrument sa indikacijom veštačkog horizonta (uglovi propinjanja i valjanja), ugla skretanja, brzine kretanja u odnosu na zemlju, brzine kretanja u odnosu na vazduh, nadmorske visine leta i brzine penjanja.

- Implementiran je prikaz video snimka u realnom vremenu unutar virtuelne realnosti (testiranu na `Oculus Quest 2` sistemu).

- U programu `JSLAB` je implementiran `RTSP server` za slanje video snimka. Ovakav server se koristi za slanje slika sa kamere letelice do daljinskog upravljača.

- Formiran je matematički model letelice i pilota kako bi se formiralo predupravljanje za simulator leta.

- Formirana je nova pilotska kabina koja omogućava upravljanje daljinski upravljanom letelicom sa mehanizma simulatora leta. Urađena je konačna integracija simulatora leta, pilotske kabine i bespilotne letelice u sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.

- U programu JSLAB je implementiran RTSP klijent za prijem video snimka. Ovakav klijent se koristi za prijem slika sa kamere letelice na daljinskom upravljaču.
- Projektovan je i napravljen je uređaj za električnu promenu antena.
- Projektovan je i napravljen je uređaj za automatsko usmeravanje antena ka letelici.
- Urađene su numeričke simulacije mehanizma simulatora leta, pomoću Three.js biblioteke u JSLAB-u je urađena vizualizacija mehanizma
- Urađena je numerička simulacija mehanizma pomoću OpenModelica programa u cilju određivanja potrebnog obrtnog momenta aktuatora. Korišćenjem biblioteke za rad sa sistemima tela je formiran model mehanizma simulatora leta i urađeno je određivanje potrebnih obrtnih momenta aktuatora. Pored toga je napisan i program koji vrši uporednu analizu rezultata koji se dobijaju nakon OpenModelica simulacija i korišćenjem JSLAB skripti zasnovanih na rešenjima datim u ovoj disertaciji. Nakon uporedne analize je zaključeno da dolazi do poklapanja rezultata uz očekivanu veliku razliku u potrebnom vremenu za izvršavanje proračuna. Skripte za proračun u JSLAB-u su specijalno napisane za ovaj problem dok je OpenModelica predviđena da bude univerzalan alat za rešavanje ovakvih problema.
- Implementirane su specijalne funkcije za komunikaciju između OpenModelica i JSLAB programskih paketa (modulom om-link), tako da je moguće raditi automatsku varijaciju parametara modela.
- Urađena je numerička i eksperimentalna verifikacija rešenja inverznog i direktnog kinematičkog problema. Numerička verifikacija je urađena proverom u programu JSLAB dok je eksperimentalna verifikacija izvršena merenjem ostvarenih položaja sa fotografija i sa 3D modela koji su dobijeni 3D skeniranjem.
- Pomoću umanjenog prototipa simulatora leta MP-6DOF-M, letelice QFLY250 i sistema za digitalni bežični prenos podataka je urađeno osnovno ispitivanje koncepta sistema za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi.
- Urađena su ispitivanja kašnjenja telemetrije pri korišćenju TCP-a lokalno i pri korišćenju TCP-a preko radio veze. Kašnjenje telemetrije direktno utiče i na kašnjenje kretanja simulatora u odnosu na letelicu.
- Urađena su ispitivanja kašnjenja video-snimka pri korišćenju WebRTC-a, pri korišćenju RTSP-a i pri korišćenju RTSP-a sa LAN kamerom. Kašnjenje video-snimka je manje značajno od kašnjenja telemetrije jer direktno utiče samo na kašnjenje prikaza slike operateru.
- Urađena je eksperimentalna validacija celog sistema u stvarnoj veličini. U realnom vremenu je informacija o kretanju bespilotne letelice preneta do pilota koji daljinski upravlja letelicom i omogućeno je da se pilot shodno tome odgovarajuće kreće pomoću simulatora leta sa šest stepeni slobode kretanja.

Konačni zaključak jeste da je **ostvaren očekivani rezultat disertacije, napravljen je i eksperimentalno verifikovan novi proizvod i to sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi**. Ovaj sistem se sastoji od pristupačnog simulatora leta, pilotske kabine i bespilotne letelice (sa svim odgovarajućim podsistemima) koji su specijalno za potrebe ovog sistema projektovani, proizvedeni i eksperimentalno ispitani.

Literatura

[1] N. Tesla, "Method of and apparatus for controlling mechanism of moving vessels or vehicles," U.S. Patent 613 809, Nov. 8, 1898.

[2] R. K. Barnhart, D. M. Marshall, i E. J. Shappee, *Introduction to Unmanned Aircraft Systems*. CRC Press, 2021, ISBN: 9780367366599.

[3] P. A. Hancock, D. A. Vincenzi, J. A. Wise, i M. Mouloua, *Human Factors in Simulation and Training*. CRC Press, 2008, ISBN: 9781420072839.

[4] E. A. Link, "Combination training device for student aviators and entertainment apparatus," U.S. Patent 1 825 462, Sep. 29, 1931.

[5] "Link-trener an-2550-1 | muzej vazduhoplovstva." Pristupljeno 01.06.2023. (2023), DOSTUPNO NA: http://www.muzejvazduhoplovstva.mod.gov.rs/cir/244/link-trener-an-2550-1-244.

[6] "Behind the scenes of boeing's dreamliner project (photos)." Pristupljeno 01.06.2023. (2023), DOSTUPNO NA: https://www.cnet.com/pictures/behind-the-scenes-of-boeings-dreamliner-project-photos/.

[7] D. Stewart, "A platform with six degrees of freedom," *Proceedings of the Institution of Mechanical Engineers*, vol. 180, no. 15, pp. 371–386, Jun. 1965. DOI: 10.1243/PIME_PROC_1965_180_029_02.

[8] E. Fichter, "A stewart platform-based manipulator: General theory and practical construction," *Int. J. Robot. Res.*, vol. 5, pp. 157–182, 1986.

[9] X. Gao, D. Lei, Q. Liao, i G. Zhang, "Generalized stewart-gough platforms and their direct kinematics," *IEEE Trans. Rob.*, vol. 21, pp. 141–151, 2005. DOI: 10.1109/TRO.2004.835456.

[10] R. L. Williams *et al.*, "Kinematics of an in-parallel actuated manipulator based on the stewart platform mechanism," NASA, Tech. Rep., 1992.

[11] J. J. Gil, I. Zabalza, J. Ros, J. M. Pintor, i J. M. Jiménez, "Kinematics and dynamics of a 6-rus hunt-type parallel manipulator by using natural coordinates," u *On Advances in Robot Kinematics*, J. Lenarčič i C. Galletti, Eds., Dordrecht: Springer Netherlands, 2004, pp. 329–335, ISBN: 978-1-4020-2249-4.

[12] J. Lee i Z. Geng, "A dynamic model of a flexible stewart platform," *Comput. Struct.*, vol. 48, pp. 367–374, 1993.

[13] B. Dasgupta i T. Mruthyunjaya, "Closed-form dynamic equations of the general stewart platform through the newton-euler approach," *Mech. Mach. Theory*, vol. 33, pp. 993–1012, 1998.

[14] B. Dasgupta i T. Mruthyunjaya, "A newton-euler formulation for the inverse dynamics of the stewart platform manipulator," *Mech. Mach. Theory*, vol. 33, pp. 1135–1152, 1998.

[15] B. Dasgupta i P. Choudhury, "A general strategy based on the newton-euler approach for the dynamic formulation of parallel manipulators," *Mech. Mach. Theory*, vol. 34, pp. 801–824, 1999.

[16] Z. Ji, "Analysis of design parameters in platform manipulators," *J. Mech. Des.*, vol. 118, pp. 526–531, 1996.

[17] C. Hua, C. Weishan, i L. Junkao, "Optimal design of stewart platform safety mechanism," *Chin. J. Aeronaut.*, vol. 20, pp. 370–377, 4 2008. DOI: 10.1016/S1000-9361(07)60057-0.

[18] Q. Jiang i C. Gosselin, "The maximal singularity-free workspace of the gough-stewart platform for a given orientation," *J. Mech. Des.*, vol. 130, pp. 112304-1–112304-8, 2008. DOI: 10.1115/1.2976452.

[19] Q. Jiang i C. Gosselin, "Maximal singularity-free total orientation workspace of the gough-stewart platform," *J. Mech. Robot.*, vol. 1, pp. 034501-1–034501-4, 2009. DOI: 10.1115/1.3147200.

[20] B. Güneri, "Complete dynamic analysis of stewart platform including singularity detection," Dokuz EylüI University, Izmir, Türkey, 2007.

[21] Q. Jiang, "Singularity-free workspace analysis and geometric optimization of parallel mechanisms," Ph.D. thesis, Faculté des Sciences et de Génie, Université Laval, Québec, Canada, 2008.

[22] B. Dasgupta i T. Mruthyunjaya, "The stewart platform manipulator: A review," *Mech. Mach. Theory*, vol. 35, pp. 15–40, 2000.

[23] M. Furqan, M. Suhaib, i N. Ahmad, "Studies on stewart platform manipulator: A review," *J. Mech. Sci. Technol.*, vol. 31, pp. 4459–4470, 2017. DOI: 10.1007/s12206-017-0846-1.

[24] K. Harib i K. Srinivasan, "Kinematic and dynamic analysis of stewart platform based machine tool structures manipulator," *Robotica*, vol. 21, pp. 541–554, 2003. DOI: 10.1017/S0263574703005046.

[25] M. Wapler, V. Urban, T. Weisner, J. Stallkamp, M. Durr, i A. Hiller, "A stewart platform for precision surgery," *Trans. Inst. Meas. Control.*, vol. 25, pp. 329–334, 2003.

[26] W. Chi, H. Ma, C. Wang, i T. Zhao, "Research on control of stewart platform integrating small attitude maneuver and vibration isolation for high-precision payloads on spacecraft," *Aerospace*, vol. 8, p. 333, 2021. DOI: 10.3390/aerospace8110333.

[27] Y. Wang, M. Soltani, D. Hussain, i R. Christensen, "Design and implementation of attitude stabilization system for marine satellite tracking antenna," *Electronics*, vol. 7, p. 398, 2018. DOI: 10.3390/electronics7120398.

[28] J. Velasco, I. Calvo, O. Barambones, P. Venegas, i C. Napole, "Experimental validation of a sliding mode control for a stewart platform used in aerospace inspection applications," *Mathematics*, vol. 8, p. 2051, 2020. DOI: 10.339/math8112051.

[29] M. Neitmann, T. Rothe, E. Kappel, i C. Hühne, "Assembly of compliant structures with autonomous industrial mobile manipulators (aimm) using an end effector with active deformation compensation for the assembly of flaps," *Machines*, vol. 10, p. 291, 2022. DOI: 10.3390/machines10050291.

[30] Z. Liu, C. Cai, M. Yang, i Y. Zhang, "Testing of a mems dynamic inclinometer using the stewart platform," *Sensors*, vol. 19, p. 4233, 2019.

[31] Z. Kausar, M. Shah, Z. Masood, *et al.*, "Energy efficient parallel configuration based six degree of freedom machining bed," *Energies*, vol. 14, p. 2642, 2021. DOI: 10.3390/en14092642.

[32] A. Bo, L. Casas, G. Cucho-Padin, M. Hayashibe, i D. Elias, "Control strategies for gait tele-rehabilitation system based on parallel robotics," *Appl. Sci.*, vol. 11, p. 11 095, 2021. DOI: 10.3390/app112311095.

[33] S. Summa, R. Gori, L. Freda, E. Castelli, i M. Petrarca, "Development of a dynamic oriented rehabilitative integrated system (doris) and preliminary tests," *Sensors*, vol. 19, p. 3402, 2019. DOI: 10.3390/s19153402.

[34] T. Stenzel, M. Sajkowski, J. Hetmańczyk, i T. Boratyński, "Application of 6-dof parallel manipulator for optoelectronic surveillance systems development," *Proc. IEEE 19th Int. Power Electron. Motion Control Conf. (PEMC)*, pp. 677–681, 2021. DOI: 10.1109/PEMC48073.2021.9432510.

[35] G. Zhan, S. Niu, W. Zhang, *et al.*, "A docking mechanism based on a stewart platform and its tracking control based on information fusion algorithm," *Sensors*, vol. 22, p. 770, 2022. DOI: 10.3390/s22030770.

[36] A. Campos, J. Quintero, R. Saltaren, M. Ferre, i R. Aracil, "An active helideck testbed for floating structures based on a stewart-gough platform," *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 22–26, 2008.

[37] Y. Patel i P. George, "Parallel manipulators applications—a survey," *Mod. Mech. Eng.*, vol. 2, pp. 57–64, 2012. DOI: 10.4236/mme.2012.23008.

[38] R. Stoughton i T. Arai, "A modified stewart platform manipulator with improved dexterity," *IEEE Trans. Robot. Autom.*, vol. 9, pp. 166–173, 1993.

[39] Z. Xie, G. Li, G. Liu, i J. Zhao, "Optimal design of a stewart platform using the global transmission index under determinate constraint of workspace," *Adv. Mech. Eng.*, vol. 9, pp. 1–14, 2017.

[40] M. Cardona, "A new approach for the forward kinematics of general stewart-gough platforms," u *Proc. IEEE Thirty Fifth Central American and Panama Convention (CONCAPAN XXXV)*, Nice, France, 2015, pp. 11–13.

[41] H. Guo i H. Li, "Dynamic analysis and simulation of a six degree of freedom stewart platform manipulator," *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 220, pp. 61–72, 2006.

[42] M.-Y. Wei, "Design and implementation of inverse kinematics and motion monitoring system for 6dof platform," *Appl. Sci.*, vol. 11, p. 9330, 2021. DOI: 10.3390/app1199330.

[43] M. Zhu, C. Huang, S. Song, i D. Gong, "Design of a gough–stewart platform based on visual servoing controller," *Sensors*, vol. 22, p. 2523, 2022. DOI: 10.3390/s22072523.

[44] B. St-Onge i C. Gosselin, "Singularity analysis and representation of the general gough-stewart platform," *Int. J. Robot. Res.*, vol. 19, pp. 271–288, 2000. DOI: 10.1177/02783640022066860.

[45] F. Molina, J. Rosario, i O. Sanchez, "Simulation environment proposal, analysis and control of a stewart platform manipulator," u *Proc. 7th Brazilian Conference on Dynamics, Control and Applications (DINCON 2008)*, Sao Paulo, Brazil, 2008, pp. 7–9.

[46] B. Inner i S. Kucuk, "A novel kinematic design, analysis and simulation tool for general stewart platforms," *Simulation*, vol. 89, pp. 876–897, 2013. DOI: 10.1177/0037549713482733.

[47] C. Zhou i Y. Fang, "Design and analysis for a three-rotational-dof flight simulator of fighter-aircraft," *Chin. J. Mech. Eng.*, vol. 31, p. 55, 2018. DOI: 10.1186/s10033-018-0256-z.

[48] S. Advani, R. Hosmann, i N. Haeck, "Integrated design of a motion cueing algorithm and motion-base mechanism for a wright flyer simulator," u *Proc. AIAA Modeling and Simulation Technologies Conference and Exhibit*, Monterey, CA, USA, 2002, pp. 5–10.

[49] M. Jones, "Motion cueing optimisation applied to rotorcraft flight simulation," *CEAS Aeronaut. J.*, vol. 8, pp. 523–539, 2017. DOI: 10.1007/s13272-017-0256-1.

[50] M.-Y. Wei, "Design of a dsp-based motion-cueing algorithm using the kinematic solution for the 6-dof motion platform," *Aerospace*, vol. 9, p. 203, 2022. DOI: 10.3390/aerospace9040203.

[51] S. Wang i L. Fu, "Predictive washout filter design for vr-based motion simulator," u *Proc. IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 2004, pp. 6291–6295.

[52] C. Liao, C. Huang, i W. Chieng, "A novel washout filter design for a six degree-of-freedom motion simulator," *JSME Int. J. Ser. C*, vol. 47, pp. 626–636, 2004.

[53] H. Asadi, S. Mohamed, K. Nelson, S. Nahavandi, i D. Zadeh, "Human perception-based washout filtering using genetic algorithm," u *Proc. LNCS 22nd International Conference on Neural Information Processing (ICONIP 2015)*, Istanbul, Turkey, 2015, pp. 9–12. DOI: 10.1007/978-3-319-26535-3_46.

[54] C. J. Gutridge, "Three degree-of-freedom simulator motion cueing using classical washout filters and acceleration feedback," M.Sc. thesis, Virginia Polytechnic Institute & State University, Blacksburg, Virginia, 2004.

[55] M. Eftekhari i H. Karimpour, "Emulation of pilot control behavior across a stewart platform simulator," *Robotica*, vol. 36, pp. 588–606, 2018. DOI: 10.1017/S0263574717000662.

[56] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985. DOI: 10.1177/027836498500400201.

[57] T. Huang, D. Whitehouse, i J. Wang, "The local dexterity, optimal architecture and design criteria of parallel machine tools," *Ann. CIRP*, vol. 47, pp. 347–351, 1998.

[58] J. Merlet, "Jacobian, manipulability, condition number, and accuracy of parallel robots," *J. Mech. Des.*, vol. 128, pp. 199–206, 2006. DOI: 10.1115/1.212174.

[59] G. Liu, S. Zheng, X. Liu, Y. Wang, i J. Han, "Formulating an invariant manipulability index of gough-stewart platform," *Adv. Mater. Res.*, vol. 479–481, pp. 2321–2326, 2012. DOI: 10.4028/www.scientific.net/AMR.479-481.2321.

[60] I. Fassi, G. Legnani, i D. Tosi, "Geometrical conditions for the design of partial or full isotropic hexapods," *J. Robot. Syst.*, vol. 22, pp. 507–518, 2005. DOI: 10.1002/rob.20074.

[61] Y. Su, B. Duan, i C. Zheng, "Genetic design of kinematically optimal fine tuning stewart platform for large spherical radio telescope," *Mechatronics*, vol. 11, pp. 821–835, 2001.

[62] Z. Gao, D. Zhang, i Y. Ge, "Design optimization of a spatial six degree-of-freedom parallel manipulator based on artificial intelligence approaches," *Robot. Comput. Integr. Manuf.*, vol. 26, pp. 180–189, 2010. DOI: 10.1016/j.rcim.2009.07.002.

[63] A. Joumah i C. Albitar, "Design optimization of 6-rus parallel manipulator using hybrid algorithm," *Int. J. Inf. Technol. Comput. Sci.*, vol. 2, pp. 83–95, 2018. DOI: 10.5815/ijitcs.2018.02.08.

[64] H. Zamani, H. Nadimi-Shahraki, i H. Gandomi, "Qana: Quantum-based avian navigation optimizer algorithm," *Eng. Appl. Artif. Intell.*, vol. 104, p. 104 314, 2021. DOI: 10.1016/j.engappai.2021.104314.

[65] H. Zamani, H. Nadimi-Shahraki, i H. Gandomi, "Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 392, p. 114 616, 2022. DOI: 10.1016/j.cma.2022.114616.

[66] H. Nadimi-Shahraki i H. Zamani, "Dmde: Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimization," *Expert Syst. Appl.*, vol. 198, p. 116 895, 2022. DOI: 10.1016/j.engappai.2021.104314.

[67] K. Deep, K. Singh, M. Kansal, i C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Appl. Math. Comput.*, vol. 212, pp. 505–518, 2009. DOI: 10.1016/j.amc.2009.02.044.

[68] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, pp. 311–338, 2000.

[69] K. Deep i M. Tahakur, "A new crossover operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 895–911, 2007. DOI: 10.1016/j.amc.2006.10.047.

[70] K. Deep i M. Tahakur, "A new mutation operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 193, no. 1, pp. 211–230, 2007. DOI: 10.1016/j.amc.2007.03.046.

[71] A. Bakar, L. Ke, H. Liu, Z. Xu, i D. Wen, "Design of low altitude long endurance solar-powered uav using genetic algorithm," *Aerospace*, vol. 8, p. 228, 2021. DOI: 10.3390/aerospace8080228.

[72] S. Bhattacharya, H. Hatwal, i A. Ghosh, "On the optimum design of stewart platform type parallel manipulators," *Robotica*, vol. 13, pp. 133–140, 1995. DOI: 10.1017/S026357470001763X.

[73] F. Molina, J. Rosario, i D. Dumur, "Multi-objective design of parallel manipulator using global indices," *Open Mech. Eng. J.*, vol. 4, pp. 37–47, 2010.

[74] A. Ríos, E. Hernández, i S. Valdez, "A two-stage mono- and multi-objective method for the optimization of general ups parallel manipulators," *Mathematics*, vol. 9, p. 543, 2021. DOI: 10.3390/math9050543.

[75] D. Chatterjee i A. Ghosal, "Design of a semi-regular stewart platform manipulator for a desired workspace," u *Proc. 14th National Conference on Machines and Mechanisms (NaCoMM-09)*, Durgapur, India, 2009, pp. 17–18.

[76] S. Advani, M. Nahon, i J. Albronda, "Optimization of six-degrees-of-freedom motion systems for flight simulators," u *Proc. Modeling and Simulation Technologies Conference*, New Orleans, LA, USA, 1997, pp. 12–22. DOI: 10.2514/6.1997-3504.

[77] S. Advani, D. Giovannetti, i M. Blum, "Design of a hexapod motion cueing system for the nasa ames vertical motion simulator," u *Proc. AIAA Modeling and Simulation Technologies Conference and Exhibit*, Monterey, CA, USA, 2002, pp. 5–8.

[78] M. Pestana, "Flying unmanned aircraft: A pilot's perspective," u *Infotech@Aerospace 2011*, 2012. DOI: 10.2514/6.2011-1490.

[79] N. E. Johnson, "Unmanned aerial vehicle (uav)," u *Encyclopedia of Systems and Control*, Springer, 2021, pp. 2388–2393.

[80] H. R. Everett, *Unmanned Systems of World Wars I and II*. The MIT Press, 2015, ISBN: 9780262029223.

[81] "Mq-9 reaper." Pristupljeno 05.05.2023. (2023), DOSTUPNO NA: https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104470/mq-9-reaper/.

[82] "The untold casualties of the drone war." Pristupljeno 01.06.2023. (2023), DOSTUPNO NA: https://www.rollingstone.com/culture/culture-news/the-untold-casualties-of-the-drone-war-67029/.

[83] "Vrabac." Pristupljeno 01.12.2023. (2023), DOSTUPNO NA: https://www.yugoimport.com/sites/default/files/documents/2023-12/Vrabac%20eng..pdf.

[84] "Pegaz." Pristupljeno 01.12.2023. (2023), DOSTUPNO NA: https://www.yugoimport.com/sites/default/files/documents/2023-12/Pegaz%20srp..pdf.

[85] "Ika-bombarder." Pristupljeno 12.01.2023. (2023), DOSTUPNO NA: https://pr-dc.com/ika-bomber.

[86] "Direktorat civilnog vazduhoplovstva republike srbije." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://cad.gov.rs/.

[87] "Ratel." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://ratel.rs/cyr/page/cyr-zakon-elektronske-komunikacije.

[88] "Vojna kontrola kvaliteta." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://www.vkk.mod.gov.rs/.

[89] "Tehnički opitni centar." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: http://www.toc.vs.rs/.

[90] "Freecad." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://github.com/FreeCAD/FreeCAD.

[91] D. Petrašinović, A. Grbović, M. Dinulović, i M. Petrašinovć, *Stajni trap letelica*. Univerzitet u Beogradu, Mašinski fakultet, Beograd, 2017.

[92] "Cgal." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://github.com/cgal/cgal.

[93] "Meshlab." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://www.meshlab.net/.

[94] "Blender." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://github.com/blender/blender.

[95] "Makehuman." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://static.makehumancommunity.org/makehuman.html.

[96] "Cesiumjs." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://cesium.com/platform/cesiumjs/.

[97] "Unreal engine." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://www.unrealengine.com/.

[98] "Inkscape." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://inkscape.org/.

[99] "Gimp." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://www.gimp.org/.

[100] T. Clyne i D. Hull, *An Introduction to Composite Materials*, 3rd ed. Cambridge University Press, 2019, ISBN: 9780521860956.

[101] "Prusaslicer." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://www.prusa3d.com/page/prusaslicer_424/.

[102] "Slic3r." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://slic3r.org/.

[103] "Kicad." Pristupljeno 25.05.2023. (2023), DOSTUPNO NA: https://www.kicad.org/.

[104] F. of Wireless Communication, *David Tse and Pramod Viswanath*. Cambridge University Press, 2005, ISBN: 9780521845274.

[105] "Openhd." Pristupljeno 15.03.2023. (2023), DOSTUPNO NA: https://github.com/OpenHD/OpenHD.

[106] "Ruby fpv." Pristupljeno 15.03.2023. (2023), DOSTUPNO NA: https://rubyfpv.com/.

[107] "Wfb-ng." Pristupljeno 15.03.2023. (2023), DOSTUPNO NA: https://github.com/svpcom/wfb-ng.

[108] "Elrs." Pristupljeno 15.03.2023. (2023), DOSTUPNO NA: https://www.expresslrs.org/.

[109] "Openipc." Pristupljeno 15.03.2023. (2023), DOSTUPNO NA: https://openipc.org/.

[110] B. W. Kernighan i D. M. Ritchie, *The C programming language*. Pearson Education Asia, 2016.

[111] B. Stroustrup, *The C++ programming language*. Pearson Education, 2013.

[112] B. Stroustrup, *A Tour of C++*. Addison-Wesley Professional, 2022.

[113] "Eigen." Pristupljeno 01.01.2023. (2023), DOSTUPNO NA: https://eigen.tuxfamily.org/.

[114] M. Grunwald, *Human haptic perception: Basics and applications*. Springer Science & Business Media, 2008.

[115] C. Hatzfeld i T. A. Kern, *Engineering haptic devices*. Springer, 2016.

[116] J. Podobnik i M. Mihelj, *Haptics for virtual reality and teleoperation*. Springer, 2012.

[117] J. Hamill i K. M. Knutzen, *Biomechanical basis of human movement*. Lippincott Williams & Wilkins, 2015.

[118] A. Girdler i O. Georgiou, "Mid-air haptics in aviation - creating the sensation of touch where there is nothing but thin air," *CoRR*, 2020.

[119] H. A. Malik, S. Rasool, A. Maqsood, i R. Riaz, "Effect of haptic feedback on pilot/operator performance during flight simulation," *Applied Sciences*, vol. 10, no. 11, p. 3877, 2020.

[120] P. Zikmund, M. Horpatzká, L. Dubnickỳ, M. Macík, i I. Jebáček, "Pilot-aircraft haptic feedback tests," *Aircraft Engineering and Aerospace Technology*, vol. 92, no. 9, pp. 1407–1412, 2020.

[121] "Pr-dc jslab - javascript laboratory environment." Pristupljeno 04.11.2024. (2024), DOSTUPNO NA: https://github.com/PR-DC/JSLAB/.

[122] D. Milutinović, *Industrijski roboti*. Beograd: Univerzitet u Beogradu, Mašinski fakultet, 2024, ISBN: 978-86-6060-175-1.

[123] "Hydraulic simulation table." Pristupljeno 01.06.2023. (2023), DOSTUPNO NA: https://www.moog.co.in/products/simulation-tables/hydraulic-simulation-table.html.

[124] "Precision linear actuators." Pristupljeno 01.06.2023. (2023), DOSTUPNO NA: https://www.thomsonlinear.com/downloads/actuators/Precision_Linear_Actuators_bruk.pdf.

[125] "Motion platform emotion-2700." Pristupljeno 01.06.2023. (2023), DOSTUPNO NA: https://www.red-dot.org/project/emotion-2700-24006-24005.

[126] V. Čović i M. Lazarević, *Mehanika robota*. Beograd: Univerzitet u Beogradu, Mašinski fakultet, 2009.

[127] F. Szufnarowski, "Stewart platform with fixed rotary actuators: A low cost design study," *Advances in medical Robotics*, no. 4, 2013.

[128] A. Cvetković i M. Spalević, *Numeričke metode*. Univerzitet u Beogradu, Mašinski fakultet, Beograd, 2013.

[129] G. Liu i S. Quek, *The Finite Element Method: A Practical Course*. Butterworth-Heinemann, 2003.

[130] R. D. Cook *et al.*, *Concepts and applications of finite element analysis*. John wiley & sons, 2007.

[131] N. Anđelić i M. Milovančević, *Otpornost materijala*. Univerzitet u Beogradu, Mašinski fakultet, Beograd, 2010.

[132] A. Ferreira, *MATLAB Codes for Finite Element Analysis*. Beograd: Springer, 2009.

[133] R. Jovanović, *Matlab i Simulink u automatskom upravljanju*. Beograd: Univerzitet u Beogradu, Mašinski fakultet, 2016.

[134] "Cgal 5.1 - 3d alpha shapes: User manual." Pristupljeno 01.10.2020. (2020), DOSTUPNO NA: https://doc.cgal.org/latest/Alpha_shapes_3/index.html.

[135] D. Ružuić, R. Čukić, M. Dunjić, M. Milovančević, N. Anđelić, i V. Milošević-Mitić, *Otpornos materijala tablice*. Beograd: Univerzitet u Beogradu, Mašinski fakultet, 2007, ISBN: 978-86-7083-606-8.

[136] V. Buljak, *Inverse Analzses with Model Reduction*. Springer, 2012.

[137] M. Caran, N. Todorović, J. Milivojević, i D. Milojković, "Genetski algoritam," Seminarski rad, Matematički fakultet, 2020.

[138] A. Đorđević, M. Stefanović, S. Arsovski, M. Erić, A. Aleksić, i S. Nestić, "Optimizacija obradnog procesa struganja korišćenjem genetskog algoritma," u *40 Nacionalna konferencija o kvalitetu*, Fakultet inženjerskih nauka Univerziteta u Kragujevcu, 2013.

[139] Chiaravalli, *Chm worm geared motors and worm gear units*, Dostupno na: https://chiaravalli.pl/wp-content/uploads/Chiaravalli-Katalog-Przekladnie_slimakowe_CHM-1.pdf (pristupljeno 16.01.2024).

[140] E. E. Motors, *General catalogue*, Dostupno na: https://www.intercom.si/wp-content/uploads/2018/06/ELVEM-cat-gen-IT-EN-2018.pdf (pristupljeno 16.01.2024).

[141] "En600-2s0015 sah electronics." Pristupljeno 01.06.2024. (2024), DOSTUPNO NA: http://www.sah.rs/en600-2s0015.html.

[142] D. Petrašinović, B. Rašuo, i N. Petrašinović, "Extended finite element method (xfem) ap-plied to aircraft duralumin spar fatigue life estimation," *Technical Gazette*, vol. 19, pp. 557–562, 2012.

[143] A. Grbović, G. Kastratović, A. Sedmak, I. Balać, i M. Popović, "Fatigue crack paths in light aircraft wing spars," *International Journal of Fatigue*, vol. 123, pp. 96–104, 2019. DOI: 10.1016/j.ijfatigue.2019.02.013.

[144] "Pr-dc half-duplex rs485 communication library for arduino_core_stm32." Pristupljeno 15.09.2023. (2023), DOSTUPNO NA: https://github.com/PR-DC/PRDC_RS485HD_STM32.

[145] "Pr-dc vfd freeport protocol implemented in arduino library." Pristupljeno 15.09.2023. (2023), DOSTUPNO NA: https://github.com/PR-DC/PRDC_FreePort.

[146] W. Rahiman i K. Chen, "Imu and compass sensors for wireless control of 6 dof parallel manipulator," *International Journal of Advancements in Electronics and Electrical Engineering – IJAEEE*, vol. 3, no. 3, pp. 126–132, 2014, ISSN: 2319-7498.

[147] S. K. Advani, "The kinematic design of flight simulator motion bases," Delft University of Technology, Delft, 1998.

[148] N. Semiconductor, *Nrf24l01 single chip 2.4ghz transceiver product specification*, Dostupno na: https://www.nordicsemi.com/chi/content/download/2730/34105/file/nRF24L01_Product_Specification_v2_0.pdf (pristupljeno 16.01.2018).

[149] R. Boucher, D. Rich, H. Crane, i C. Matheny, "A method for measuring the product of inertia and the inclination of the principal longitudinal axis of inertia of an airplane," NACA, Washington, USA, Tech. Rep., 1954.

[150] A. Kotikalpudi, B. Taylor, C. Moreno, i et al., "Swing tests for estimation of moments of inertia," *Aerospace Engineering and Mechanics, University of Minnesota, USA*, 2013.

[151] "Ardupilot project." Pristupljeno 15.06.2024. (2024), DOSTUPNO NA: https://github.com/ArduPilot/ardupilot.

[152] H. Bin i A. Justice, "The design of an unmanned aerial vehicle based on the ardupilot," *Indian Journal of Science and Technology*, vol. 2, no. 4, pp. 12–15, 2009.

[153] "Connect escs and motors — copter documentation." Pristupljeno 15.06.2024. (2024), DOSTUPNO NA: https://ardupilot.org/copter/docs/connect-escs-and-motors.html.

[154] "Ardupilot - setting up the waf build environment on windows using cygwin." Pristupljeno 15.06.2024. (2024), DOSTUPNO NA: https://ardupilot.org/dev/docs/building-setup-windows-cygwin.html#building-setup-windows-cygwin.

[155] S. Baldi, D. Sun, X. Xia, G. Zhou, i D. Liu, "Ardupilot-based adaptive autopilot: Architecture and software-in-the-loop experiments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 5, pp. 4473–4485, 2022.

[156] "Jsdataflashparser." Pristupljeno 15.06.2024. (2024), DOSTUPNO NA: https://github.com/Williangalvani/JsDataflashParser.

[157] "Mavlink developer guide." Pristupljeno 15.06.2024. (2024), DOSTUPNO NA: https://mavlink.io/en/.

[158] S. Atoev, K.-R. Kwon, S.-H. Lee, i K.-S. Moon, "Data analysis of the mavlink communication protocol," u *2017 International Conference on Information Science and Communications Technologies (ICISCT)*, IEEE, 2017, pp. 1–3.

[159] Y.-M. Kwon, J. Yu, B.-M. Cho, Y. Eun, i K.-J. Park, "Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles," *IEEE Access*, vol. 6, pp. 43 203–43 212, 2018.

[160] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, i M. Khalgui, "Micro air vehicle link (mavlink) in a nutshell: A survey," *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019.

[161] A. Allouch, O. Cheikhrouhhou, A. Kouba, M. Khalgui, i T. Abbes, "Mavsec: Securing the mavlink protocol for ardupilot / px4 unmanned aerial systems," u *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2019, pp. 621–628.

[162] M. E. Mora-Soto, J. Maldonado-Romo, A. Rodríguez-Molina, i M. Aldape-Pérez, "Building a realistic virtual simulator for unmanned aerial vehicle teleoperation," *Applied Sciences*, vol. 11, no. 24, p. 12 018, 2021.

[163] C. Caceres, D. Amaya, i J. M. Rosário, "Simulation, model and control of a quadcopter ar drone 2.0," *International review of mechanical engineering*, vol. 10, no. 3, 2016.

[164] D. Alaez, X. Olaz, M. Prieto, P. Porcellinis, i J. Villadangos, "Hil flight simulator for vtol-uav pilot training using x-plane," *Information*, vol. 13, p. 585, 2022. DOI: 10.3390/info13120585.

[165] "About flightgear." Pristupljeno 02.05.2024. (2024), DOSTUPNO NA: https://www.flightgear.org/about/.

[166] S. Shah, D. Dey, C. Lovett, i A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," u *Field and Service Robotics: Results of the 11th International Conference*, Springer, 2018, pp. 621–635.

[167] "Airsim." Pristupljeno 01.06.2024. (2024), DOSTUPNO NA: https://microsoft.github.io/AirSim/.

[168] C. Ma, Y. Zhou, i Z. Li, "A new simulation environment based on airsim, ros, and px4 for quadcopter aircrafts," u *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, 2020, pp. 486–490.

[169] "Sitl simulator ardupilot." Pristupljeno 01.06.2024. (2024), DOSTUPNO NA: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[170] "Ika-trainer | pr-dc." Pristupljeno 01.06.2024. (2024), DOSTUPNO NA: https://pr-dc.com/ika-trainer.

[171] D. T. McRuer, D. Graham, E. Krendel, i W. Reisener, "Human pilot dynamics in compensatory systems," Air Force Flight Dynamics Lab, Tech. Rep., 1965.

[172] D. McRuer, I. Ashkenas, i D. Graham, *Aircraft Dynamics and Automatic Control*. Princeton University Press, 1973, ISBN: 9780691600383.

[173] D. T. McRuer i E. Krendel, "Mathematical models of human pilot behavior," Advisory Group on Aerospace Research and Development, Tech. Rep., 1974.

[174] M. Jirgl, J. Boril, i R. Jalovecky, "Statistical evaluation of pilot's behavior models parameters connected to military flight training," *Energies*, vol. 13, no. 17, p. 4452, 2020.

[175] J. A. Schroeder, "Helicopter flight simulation motion platform requirements," NASA, Tech. Rep., 1999.

[176] M. Nenadović, *Stabilnost i upravljivost letelica - Drugi deo (Uzdužna stabilnost aviona)*. Beograd, SFRJ: Savezni sekretarijat za narodnu odbranu, 1984.

[177] M. Nenadović, *Stabilnost i upravljivost letelica - Prvi deo*. Beograd, SFRJ: Savezni sekretarijat za narodnu odbranu, 1981.

[178] R. C. Nelson, *Flight Stability and Automatic Control*. Mcgraw-Hill College, 1998.

[179] "Rfc 7826." Pristupljeno 01.06.2024. (2024), DOSTUPNO NA: `https://datatracker.ietf.org/doc/html/rfc7826`.

[180] "Rfc 3550." Pristupljeno 01.06.2024. (2024), DOSTUPNO NA: `https://datatracker.ietf.org/doc/html/rfc3550`.

[181] S. C. Tartari Filho i E. L. L. Cabral, "Kinematics and workspace analysis of a parallel architecture robot: The hexa," u *18th International Congress of Mechanical Engineering, Ouro Preto, MG*, 2005.

[182] "Three.js - javascript 3d library." Pristupljeno 10.04.2023. (2023), DOSTUPNO NA: `https://github.com/mrdoob/three.js`.

[183] M. Otter, H. Elmqvist, i S. E. Mattsson, "The new modelica multibody library," u *3rd International modelica conference*, 2003, pp. 311–330.

[184] "Modelica.mechanics.multibody." Pristupljeno 10.09.2020. (2020), DOSTUPNO NA: `https://build.openmodelica.org/Documentation/Modelica.Mechanics.MultiBody.html`.

[185] P. Campanini, "Object-oriented modelling and simulation of parallel manipulators," M.Sc. thesis, Politecnico Di Milano, School of Industrial and Information Engineering, Milano, Italy, 2018.

[186] Y. K. Asalani, "Development of hexapod manipulator as motion platform system for generic flight dynamics simulator," Institut Teknologi Bandung, Faculty of Mechanical and Aerospace Engineering, 2018.

[187] "Kinovea: Video solution for sport analysis. capture, inspect, compare, annotate and measure technical performances." Pristupljeno 20.04.2023. (2023), DOSTUPNO NA: `https://github.com/Kinovea/Kinovea`.

[188] P. Li, A. Ghasemi, W. Xie, i W. Tian, "Visual sensor-based dynamic identification of a 6-rss parallel robot," u *2019 International Conference on Control, Automation and Information Sciences (ICCAIS)*, IEEE, 2019, pp. 1–6.

[189] P. Li, "Visual calibration, identification and control of 6-rss parallel robots," Ph.D. thesis, Concordia University, Montréal, Québec, Canada, 2020.

[190] P. Li, T. Shu, W.-F. Xie, i W. Tian, "Dynamic visual servoing of a 6-rss parallel robot based on optical cmm," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, p. 40, 2021.

[191] R. C. Lemes, M. M. Souza, E. M. Belo, i J. H. Bidinotto, "Latency on a stewart platform using washout filter," *The Aeronautical Journal*, vol. 122, no. 1252, pp. 1003–1019, 2018.

A

JSLAB klasa MP_6RSS_SOLVER

U listingu A.1 je dat kompletan kod JSLAB klase `SOLVER_MP_6RSS.jsl` koja implementira sve neophodne metode za rešavanje inverznog i direktnog kinematičkog problema ali i metode za izračunavanje indikatora performansi i optimizaciju geometrije *6RSS mehanizma*.

```
/**
 * @file Motion Platform 6RSS Mechanism solver
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

/**
 * Class for solving the Motion Platform 6RSS Mechanism.
 */
class MP_6RSS_SOLVER {

  /**
   * Constructor for the 6RSS solver.
   * @param {Object} MP - Mechanism parameters.
   */
  constructor(MP) {
    this.MP = MP;
    var m = 6;

    MP.m = m;

    MP.sigma_l = [-1, 1, -1, 1, -1, 1];
    MP.beta_s = [pi / 6, pi / 6, 5 / 6 * pi, 5 / 6 * pi, -pi / 2, -pi / 2];
    MP.nu_ll = minus(MP.beta_s, multiplyEl(MP.sigma_l, pi/2));

    MP.tau_r = 1E-9; // [-] relative error of external parameters
    MP.tau_a = 1E-6; // [-] absoulte error of external parameters
    MP.n_iter_max = 50; // [-] max number of iterations for direct kinematic

    MP.IK_FLAG = false; // [-] Inverse kinematic success flag

    this.state = {
      'NORMAL_STATE': 0,
      'GET_ALPHA_FAILED': 1,
      'LAMBDA_P_ERROR': 2,
      'LAMBDA_B_ERROR': 3,
      'LEVER_COLLISION': 4,
```

```
40        'MU_UL_ERROR': 5,
41        'GET_PARAEMTERS_FAILED': 6,
42        'JACOBIAN_INVERSE_FAILED': 7,
43        'GET_PARAEMTERS_MAX_ITER': 8
44      };
45
46      MP.alpha = zeros(m);  // [deg] angles of the lower levers
47      MP.alpha_o = zeros(m);  // [deg] previous values
48      MP.alpha_w_in = zeros(m);  // [deg] desired angles of the lower levers
49      MP.x = zeros(m);  // [mm and deg] translation and rotation [x, y, z, phi,
            theta, psi]
50      MP.x_o = zeros(m);  // previous values
51
52      MP.delta_p = zeros(m);
53      MP.delta_b = zeros(m);
54      MP.beta_ll = zeros(m);
55      MP.beta_jp = zeros(m);
56      MP.i_m = zeros(3 * m);
57      MP.i_ll = zeros(3 * m);
58      MP.i_ll0 = zeros(3 * m);
59      MP.i_jp = zeros(3 * m);
60      MP.i_jpb = zeros(3 * m);
61
62      MP.z0 = 0;
63      MP.z1 = 0;
64      MP.P = zeros(3 * m);
65      MP.Pb = zeros(3 * m);
66      MP.PO = zeros(2 *3);
67      MP.POb = zeros(2 *3);
68      MP.B = zeros(3 * m);
69      MP.A = zeros(3 * m);
70      MP.BPb = zeros(3 * m);
71      MP.l = zeros(m);
72      MP.D = zeros(m);
73      MP.E = zeros(m);
74      MP.F = zeros(m);
75      MP.G = zeros(m);
76      MP.R = zeros(3 * 3);
77      MP.RP = zeros(3 * m);
78      MP.RPO = zeros(3 * m);
79      MP.Ri_jp = zeros(3 * m);
80      MP.Rm1_i_ll = zeros(3 * m);
81      MP.Rm2_i_ll = zeros(3 * m);
82      MP.Rm1_BA = zeros(3 * m);
83      MP.Rm2_BA = zeros(3 * m);
84      MP.BA_0 = zeros(3 * m);
85      MP.PbA = zeros(3 * m);
86      MP.lambda_p = zeros(m);
87      MP.lambda_b = zeros(m);
88      MP.i_ul = zeros(3 * m);
89      MP.m_ul = zeros(3 * m);
90      MP.tau_r0 = 0;
91      MP.dalpha = zeros(m);
92      MP.dalpha_r = zeros(m);
93      MP.dx = zeros(m);
94      MP.m_m = zeros(3 * m);
95      MP.BA = zeros(3 * m);
96      MP.PObPb = zeros(3 * m);
97      MP.J_ai = zeros(m);
```

```
98      MP.J_a = zeros(2 * 3 * m);
99      MP.invJ_ai = zeros(m);
100     MP.invJ_a = zeros(2 * 3 * m);
101     MP.J = zeros(2 * 3 * m);
102     MP.J_x = zeros(2 * 3 * m);
103     MP.invJ_x = zeros(2 * 3 * m);
104     MP.e = createFilledArray(m, this.state['NORMAL_STATE']);
105     MP.n_iter = 0;
106
107     this.iterationCallback;
108   }
109
110   /**
111    * Calculates the geometry for the 6RSS mechanism.
112    */
113   getGeometry() {
114     var MP = this.MP;
115     var m = MP.m;
116
117     var cepsilon = cosd(MP.epsilon);
118     var sepsilon = sind(MP.epsilon);
119
120     for(var j = 0; j < m; j++) {
121       MP.beta_ll[j] = MP.nu_ll[j] + MP.tau_ll * MP.sigma_l[j] * d2r;
122       MP.beta_jp[j] = MP.beta_s[j] + MP.tau_jp * MP.sigma_l[j] * d2r;
123       MP.delta_p[j] = MP.beta_s[j] + MP.gamma_p * MP.sigma_l[j] * d2r;
124       MP.delta_b[j] = MP.beta_s[j] + MP.gamma_b * MP.sigma_l[j] * d2r;
125
126       MP.i_m[j * 3] = -MP.sigma_l[j] * cos(MP.beta_ll[j] + pi / 2 * MP.sigma_l[j
          ]) * cepsilon;
127       MP.i_m[j * 3 + 1] = -MP.sigma_l[j] * sin(MP.beta_ll[j] + pi / 2 * MP.
          sigma_l[j]) * cepsilon;
128       MP.i_m[j * 3 + 2] = -MP.sigma_l[j] * sepsilon;
129
130       MP.i_ll[j * 3] = cos(MP.beta_ll[j] + pi / 2 * MP.sigma_l[j]) * cosd(MP.
          epsilon + MP.mu_ll);
131       MP.i_ll[j * 3 + 1] = sin(MP.beta_ll[j] + pi / 2 * MP.sigma_l[j]) * cosd(MP
          .epsilon + MP.mu_ll);
132       MP.i_ll[j * 3 + 2] = sind(MP.epsilon + MP.mu_ll);
133
134       MP.i_jp[j * 3] = cos(MP.beta_jp[j]) * cosd(MP.mu_jp);
135       MP.i_jp[j * 3 + 1] = sin(MP.beta_jp[j]) * cosd(MP.mu_jp);
136       MP.i_jp[j * 3 + 2] = sind(MP.mu_jp);
137
138       MP.P[j * 3] = MP.R_p * cos(MP.delta_p[j]);
139       MP.P[j * 3 + 1] = MP.R_p * sin(MP.delta_p[j]);
140       MP.P[j * 3 + 2] = 0;
141
142       MP.B[j * 3] = MP.R_b * cos(MP.delta_b[j]);
143       MP.B[j * 3 + 1] = MP.R_b * sin(MP.delta_b[j]);
144       MP.B[j * 3 + 2] = 0;
145
146       MP.A[j * 3] = MP.B[j * 3] + MP.a * cos(MP.beta_ll[j]);
147       MP.A[j * 3 + 1] = MP.B[j * 3 + 1] + MP.a * sin(MP.beta_ll[j]);
148       MP.A[j * 3 + 2] = MP.B[j * 3 + 2];
149
150       for(var i = 0; i < 3; i++) {
151         MP.BA_0[j * 3 + i] = MP.A[j * 3 + i] - MP.B[j * 3 + i];
152       }
```

```
153     }
154
155     MP.i_ll0 = [...MP.i_ll];
156
157     MP.Rm2_i_ll = cross3D(MP.i_m, MP.i_ll0, m);
158     MP.Rm1_i_ll = cross3D(MP.i_m, MP.Rm2_i_ll, m);
159     MP.Rm2_BA = cross3D(MP.i_m, MP.BA_0, m);
160     MP.Rm1_BA = cross3D(MP.i_m, MP.Rm2_BA, m);
161
162     MP.z1 = pow(MP.s, 2) - (pow(MP.P[0] - MP.A[0], 2) + pow(MP.P[1] - MP.A[1],
          2));
163     MP.z0;
164     if(MP.z1 > 0) {
165       MP.z0 = sqrt(MP.z1) + MP.A[2];
166     } else {
167       MP.z0 = Infinity;
168     }
169
170     MP.i_jpb = [...MP.i_jp];
171     MP.Pb = [...MP.P];
172
173     for(var j = 0; j < m; j++) {
174       MP.Pb[j * 3 + 2] = MP.Pb[j * 3 + 2] + MP.z0;
175     }
176
177     for(var i = 0; i < 3; i++) {
178       MP.PO[i] = MP.cg[i];
179       MP.PO[i + 3] = 0;
180     }
181   }
182
183   /**
184    * Updates the geometry parameters of the mechanism based on the input array.
185    * @param {Array<number>} x - An array containing the new geometry parameters.
186    */
187   updateGeometry(x) {
188     MP.a = x[0];
189     MP.s = x[1];
190     MP.R_p = x[2];
191     MP.R_b = x[3];;
192     MP.gamma_p = x[4];
193     MP.gamma_b = x[5];
194     MP.epsilon = x[6];
195     MP.tau_ll = x[7];;
196     MP.tau_jp = x[8];
197     MP.mu_ll = x[9];
198     MP.mu_jp = x[10];
199     this.getGeometry();
200   }
201
202   /**
203    * Sets the mechanism parameters based on input.
204    * @param {Array} x_in - Input values for parameters.
205    */
206   setParameters(x_in) {
207     MP.x_o = [...MP.x];
208     MP.x = [...x_in];
209     this.getAlpha();
210   }
```

```
211
212
213   /**
214    * Sets the angles for the lower levers.
215    * @param {Array} alpha_in - Input angles.
216    * @param {boolean} [debug=false] - Debug flag.
217    * @param {string} [log] - Log output.
218    * @returns {string|undefined} - Debug log if debug is enabled.
219    */
220   setAlpha(alpha_in, debug = false, log) {
221     MP.alpha_w_in = [...alpha_in];
222     return this.getParameters(debug, log);
223   }
224
225   /**
226    * Retrieves the current angles of the lower levers.
227    */
228   getAlpha() {
229     var MP = this.MP;
230     var m = MP.m;
231
232     MP.alpha_o = [...MP.alpha];
233
234     var cphi = cosd(MP.x[3]);
235     var ctheta = cosd(MP.x[4]);
236     var cpsi = cosd(MP.x[5]);
237     var sphi = sind(MP.x[3]);
238     var stheta = sind(MP.x[4]);
239     var spsi = sind(MP.x[5]);
240
241     MP.R = [
242       cpsi * ctheta, spsi * ctheta, -stheta,
243       -spsi * cphi + cpsi * stheta * sphi, cpsi * cphi + spsi * stheta * sphi,
              ctheta * sphi,
244       spsi * sphi + cpsi * stheta * cphi, -cpsi * sphi + spsi * stheta * cphi,
              ctheta * cphi
245     ];
246
247     MP.Ri_jp = multiply(MP.R, MP.i_jp, 3, 3, m);
248     MP.RP = multiply(MP.R, MP.P, 3, 3, m);
249     MP.RPO = multiply(MP.R, MP.PO, 3, 3, 2);
250
251     for(var j = 0; j < 2; j++) {
252       for(var i = 0; i < 3; i++) {
253         MP.POb[j * 3 + i] = MP.x[i] + MP.RPO[j * 3 + i];
254       }
255       MP.POb[j * 3 + 2] += MP.z0;
256     }
257
258     for(var j = 0; j < m; j++) {
259       for(var i = 0; i < 3; i++) {
260         MP.Pb[j * 3 + i] = MP.x[i] + MP.RP[j * 3 + i];
261         MP.BPb[j * 3 + i] = MP.Pb[j * 3 + i] - MP.B[j * 3 + i];
262         MP.i_jpb[j * 3 + i] = MP.Ri_jp[j * 3 + i];
263       }
264
265       MP.Pb[j * 3 + 2] += MP.z0;
266       MP.BPb[j * 3 + 2] += MP.z0;
267
```

```
268       MP.l[j] = sqrt(
269         pow(MP.BPb[j * 3], 2) + pow(MP.BPb[j * 3 + 1], 2) + pow(MP.BPb[j * 3 +
              2], 2)
270       );
271       MP.D[j] = pow(MP.l[j], 2) - (pow(MP.s, 2) - pow(MP.a, 2)) -
272         2 * ((MP.Rm1_BA[j * 3] + MP.BA_0[j * 3]) * MP.BPb[j * 3] +
273           (MP.Rm1_BA[j * 3 + 1] + MP.BA_0[j * 3 + 1]) * MP.BPb[j * 3 + 1] +
274           (MP.Rm1_BA[j * 3 + 2] + MP.BA_0[j * 3 + 2]) * MP.BPb[j * 3 + 2]);
275       MP.E[j] = 2 * (MP.Rm2_BA[j * 3] * MP.BPb[j * 3] +
276         MP.Rm2_BA[j * 3 + 1] * MP.BPb[j * 3 + 1] +
277         MP.Rm2_BA[j * 3 + 2] * MP.BPb[j * 3 + 2]);
278       MP.F[j] = -2 * (MP.Rm1_BA[j * 3] * MP.BPb[j * 3] +
279         MP.Rm1_BA[j * 3 + 1] * MP.BPb[j * 3 + 1] +
280         MP.Rm1_BA[j * 3 + 2] * MP.BPb[j * 3 + 2]);
281       MP.G[j] = MP.D[j] / sqrt(pow(MP.E[j], 2) + pow(MP.F[j], 2));
282       MP.alpha[j] = asin(MP.G[j]) - atan2(MP.F[j], MP.E[j]);
283
284       var calpha = cos(MP.alpha[j]);
285       var salpha = sin(MP.alpha[j]);
286
287       for(var i = 0; i < 3; i++) {
288         MP.A[j * 3 + i] = MP.B[j * 3 + i] + MP.BA_0[j * 3 + i] +
289           (1 - calpha) * MP.Rm1_BA[j * 3 + i] +
290           salpha * MP.Rm2_BA[j * 3 + i];
291         MP.i_ll[j * 3 + i] = MP.i_ll0[j * 3 + i] +
292           (1 - calpha) * MP.Rm1_i_ll[j * 3 + i] +
293           salpha * MP.Rm2_i_ll[j * 3 + i];
294       }
295
296       if(abs(MP.G[j]) <= 1) {
297         MP.e[j] = this.state['NORMAL_STATE'];
298         MP.IK_FLAG = true;
299       } else {
300         MP.e[j] = this.state['GET_ALPHA_FAILED'];
301         MP.x = [...MP.x_o];
302         MP.alpha = [...MP.alpha_o];
303       }
304     }
305   }
306
307   /**
308    * Computes the Jacobian matrix.
309    */
310   getJacobian() {
311     var MP = this.MP;
312     var m = MP.m;
313
314     for(var k = 0; k < 3 * m; k++) {
315       MP.i_ul[k] = (MP.Pb[k] - MP.A[k]) / MP.s;
316       MP.BA[k] = MP.A[k] - MP.B[k];
317     }
318
319     for(var j = 0; j < m; j++) {
320       for(var i = 0; i < 3; i++) {
321         MP.PObPb[j * 3 + i] = MP.Pb[j * 3 + i] - MP.POb[1 * 3 + i];
322       }
323     }
324
325     MP.m_ul = cross3D(MP.PObPb, MP.i_ul, m);
```

```
326       for(var j = 0; j < 3; j++) {
327         for(var i = 0; i < m; i++) {
328           MP.J_x[j * m + i] = MP.i_ul[j + i * 3];
329           MP.J_x[(j + 3) * m + i] = MP.m_ul[j + i * 3];
330         }
331       }
332
333       MP.invJ_x = inv(MP.J_x, m);
334       if(!MP.invJ_x) {
335         fill(this.state['JACOBIAN_INVERSE_FAILED'], MP.e, m);
336         return;
337       }
338
339       MP.m_m = cross3D(MP.i_m, MP.BA, m);
340       MP.J_ai = dotColumn(MP.i_ul, MP.m_m, 3, m);
341       MP.J_a = diag(MP.J_ai, m); // J_a
342       MP.invJ_ai = reciprocal(MP.J_ai, m);
343       MP.invJ_a = diag(MP.invJ_ai, m); // invJ_a
344
345       MP.J = multiply(MP.invJ_x, MP.J_a, 2*3, m, 2*3); // Jacobian = inv(J_x)*J_a
346     }
347
348     /**
349      * Calculates mechanism parameters based on current state.
350      * @param {boolean} [debug=false] - Debug flag.
351      * @param {string} [log] - Log output.
352      * @returns {string|undefined} - Debug log if debug is enabled.
353      */
354     getParameters(debug = false, log) {
355       var MP = this.MP;
356       var m = MP.m;
357
358       MP.x_o = [...MP.x];
359       MP.alpha_o = [...MP.alpha];
360       MP.n_iter = 0;
361
362       for(var j = 0; j < m; j++) {
363         MP.dalpha[j] = MP.alpha[j] - MP.alpha_w_in[j];
364       }
365       MP.tau_r0 = MP.tau_r * norm(MP.dalpha, m);
366
367       if(debug) {
368         log += disp(' -----');
369       }
370       while(norm(MP.dalpha, m) > MP.tau_r0 + MP.tau_a) {
371         if(MP.n_iter >= MP.n_iter_max) {
372           fill(this.state['GET_PARAEMTERS_MAX_ITER'], MP.e, m);
373           break;
374         } else {
375           MP.n_iter++;
376         }
377         if(debug) {
378           log += disp(' n_iter = ' + MP.n_iter);
379           log += dispRowVector('x', MP.x, m);
380         }
381
382         this.getAlpha();
383         if(any(MP.e, m)) {
384           fill(this.state['GET_PARAEMTERS_FAILED'], MP.e, m);
```

```
385         break;
386       }
387
388       this.getJacobian();
389       if(any(MP.e, m)) {
390         break;
391       }
392       if(debug) {
393         log += dispMatrix('J_x', MP.J_x, 2 * 3, m);
394         log += dispMatrix('J_a', MP.J_a, 2 * 3, m);
395         log += dispMatrix('J', MP.J, 2 * 3, m);
396       }
397
398       // dx^k = J*dalpha;
399       for(var j = 0; j < m; j++) {
400         MP.dalpha[j] = MP.alpha[j] * r2d - MP.alpha_w_in[j];
401         MP.dalpha_r[j] = MP.alpha[j] - MP.alpha_w_in[j] * d2r;
402       }
403       MP.dx = multiply(MP.J, MP.dalpha_r, 2*3, m, 1);
404       if(debug) {
405         log += dispRowVector('dalpha', MP.dalpha, m);
406         log += dispRowVector('dx', MP.dx, m);
407         log += disp(' -----');
408       }
409
410       // x^{k+1} = x^k+dx^k
411       for(var j = 0; j < 3; j++) {
412         MP.x[j] = MP.x[j] - MP.dx[j];
413         MP.x[j + 3] =  MP.x[j + 3] - MP.dx[j + 3] * r2d;
414       }
415
416       if(isFunction(this.iterationCallback)) {
417         this.iterationCallback();
418       }
419     }
420     if(any(MP.e, m)) {
421       MP.x = [...MP.x_o];
422       MP.alpha = [...MP.alpha_o];
423     }
424
425     if(debug) {
426       return log;
427     }
428     return true;
429   }
430
431   /**
432    * Checks joint constraints and angles.
433    */
434   checkJoints() {
435     var MP = this.MP;
436     var m = MP.m;
437
438     if(MP.IK_FLAG) {
439       for(var k = 0; k < 3 * m; k++) {
440         MP.PbA[k] = MP.A[k] - MP.Pb[k];
441       }
442
443       for(var j = 0; j < m; j++) {
```

```
        MP.lambda_p[j] = abs(acos(
          (MP.PbA[j * 3] * MP.i_jpb[j * 3] +
           MP.PbA[j * 3 + 1] * MP.i_jpb[j * 3 + 1] +
           MP.PbA[j * 3 + 2] * MP.i_jpb[j * 3 + 2]) / MP.s
        ) * r2d - 90.0);
        MP.lambda_b[j] = abs(acos(
          (MP.PbA[j * 3] * MP.i_ll[j * 3] +
           MP.PbA[j * 3 + 1] * MP.i_ll[j * 3 + 1] +
           MP.PbA[j * 3 + 2] * MP.i_ll[j * 3 + 2]) / MP.s
        ) * r2d - 90.0);

        if(MP.lambda_b[j] > MP.lambda_max) {
          MP.e[j] = this.state['LAMBDA_B_ERROR'];
        } else if(MP.lambda_p[j] > MP.lambda_max) {
          MP.e[j] = this.state['LAMBDA_P_ERROR'];
        }
      }
    }
  }

  /**
   * Gets the allowed actuator load.
   * @param {number} F_r - Force reference.
   * @param {number} a - Parameter a.
   * @param {number} b - Parameter b.
   * @param {number} x - Parameter x.
   * @returns {Array} - Allowed actuator loads.
   */
  getActuatorAllowedLoad(F_r, a, b, x) {
    return [F_r * a / (b + x), F_r / 5];
  }

  /**
   * Calculates the actuator load.
   * @returns {Array} - Calculated actuator loads.
   */
  getActuatorLoad() {
    var MP = this.MP;

    var F_a = sumCol(multiplyEl(MP.i_m, MP.F), 3, 6);
    var F_av = multiplyEl(MP.i_m, repRow(F_a, 3));
    var F_rv = minus(MP.F, F_av);
    var F_r = normCol(F_rv, 3, 6);
    return [F_r, F_a];
  }

  /**
   * Computes the forces in the mechanism.
   * @returns {Array} - Calculated forces.
   */
  getForces() {
    var MP = this.MP;
    var m = MP.m;

    MP.i_ul = scale(minus(MP.Pb, MP.A), 1/ MP.s);
    for(var j = 0; j < m; j++) {
      for(var i = 0; i < 3; i++) {
        MP.PObPb[j * 3 + i] = MP.Pb[j * 3 + i] - MP.POb[1 * 3 + i];
      }
```

```
503       }
504
505       MP.m_ul = cross3D(MP.PObPb, MP.i_ul, m);
506       for(var j = 0; j < 3; j++) {
507         for(var i = 0; i < m; i++) {
508           MP.J_x[j * m + i] = MP.i_ul[j + i * 3];
509           MP.J_x[(j + 3) * m + i] = MP.m_ul[j + i * 3];
510         }
511       }
512
513       var W = scale(MP.g, MP.M);
514       MP.F_x = [...W, ...cross3D(multiply(MP.R, MP.cg, 3, 3, 1), W, 1)];
515       var f = linsolve(transpose(MP.J_x, 6, 6), MP.F_x, 6);
516       MP.F = scale(multiplyEl(MP.i_ul, repRow(f, 3)), -1);
517       return MP.F;
518     }
519
520     /**
521      * Computes the forces on the platform.
522      * @returns {Array} - Calculated platform forces.
523      */
524     getForcesPlatform() {
525       var MP = this.MP;
526       var m = MP.m;
527
528       var W = scale(MP.g, MP.M);
529       var Rp = transpose(MP.R, 3, 3);
530       var Wp = multiply(Rp, W, 3, 3, 1);
531       var F_xp = [...Wp, ...cross3D(MP.cg, Wp, 1)];
532
533       var i_ulp = multiply(Rp, scale(minus(MP.Pb, MP.A), 1/ MP.s), 3, 3, 6);
534       var m_ulp = cross3D(MP.P, i_ulp, m);
535       var J_xp = zeros(6*6);
536       for(var j = 0; j < 3; j++) {
537         for(var i = 0; i < m; i++) {
538           J_xp[j * m + i] = i_ulp[j + i * 3];
539           J_xp[(j + 3) * m + i] = m_ulp[j + i * 3];
540         }
541       }
542       var f = linsolve(transpose(J_xp, 6, 6), F_xp, 6);
543       MP.F = scale(multiply(MP.R, multiplyEl(i_ulp, repRow(f, 3)), 3, 3, 6), -1);
544       return MP.F;
545     }
546
547     /**
548      * Calculates torques based on applied forces.
549      * @returns {Array} - Calculated torques.
550      */
551     getTorquesByForce() {
552       var MP = this.MP;
553
554       var rm = scale(minus(MP.B, MP.A), 1/1000); // [m]
555       var M = cross3D(rm, MP.F, 6);
556       MP.M_m = scale(sumCol(multiplyEl(MP.i_m, M), 3, 6), -1); // [Nm]
557       return MP.M_m;
558     }
559
560     /**
561      * Calculates the torques in the mechanism.
```

```
562     * @returns {Array} - Calculated torques.
563     */
564   getTorques() {
565     var MP = this.MP;
566 
567     var W = scale(MP.g, MP.M);
568     MP.F_x = [...W, ...cross3D(multiply(MP.R, MP.cg, 3, 3, 1), W, 1)];
569     MP.M_m = scale(multiply(transpose(MP.J, 6, 6), scale(MP.F_x, 1/1000), 6, 6,
          1), -1); // [Nm]
570     return MP.M_m;
571   }
572 
573   /**
574     * Computes the normalized Jacobian matrix.
575     */
576   getJacobianNormalized() {
577     var MP = this.MP;
578     var m = MP.m;
579 
580     MP.J_xn = zeros(6*6);
581     for(var j = 0; j < 3; j++) {
582       for(var i = 0; i < m; i++) {
583         MP.J_xn[j * m + i] = MP.J_x[j * m + i];
584         MP.J_xn[(j + 3) * m + i] = MP.J_x[(j + 3) * m + i]/MP.R_p;
585       }
586     }
587     MP.J_an = scale(MP.J_a, 1/MP.a);
588     var invJ_xn = inv(MP.J_xn, m);
589     MP.J_n = multiply(invJ_xn, MP.J_an, m, m, m);
590   }
591 
592   /**
593     * Calculates the manipulability of the mechanism.
594     */
595   getManipulability() {
596     var MP = this.MP;
597 
598     MP.w = abs(det(reshape(MP.J_n, 6, 6)));
599   }
600 
601   /**
602     * Computes the dexterity of the mechanism.
603     */
604   getDexterity() {
605     var MP = this.MP;
606 
607     var e = eigs(reshape(multiply(transpose(MP.J_n, 6, 6), MP.J_n, 6, 6, 6), 6,
          6)).values;
608     var k = sqrt(max(e))/sqrt(min(e));
609     MP.eta = 1/k;
610   }
611 
612   /**
613     * Checks for mechanical intersection numerically based on operational
          parameters.
614     * @param {Object} OP - Optimization parameters.
615     * @returns {Array<number>} An array of intersection constraint values.
616     */
617   checkIntersectionNumerical(OP) {
```

```
618     var alpha_i = range(-pi/2, pi/2, pi / OP.Na);
619     var A = (alpha, m) => elementWise((i) => MP.B[m * 3 + i] +
620       MP.BA_0[m * 3 + i] + (1 - cos(alpha)) * MP.Rm1_BA[m * 3 + i] +
621       sin(alpha) * MP.Rm2_BA[m * 3 + i], [0, 1, 2]);
622     var B = (m) => [MP.B[m * 3], MP.B[m * 3 + 1], MP.B[m * 3 + 2]];
623
624     var r1 = elementWise((a) => norm(minus(A(a, 0), A(a, 1))), alpha_i);
625     var r2 = elementWise((a) => norm(minus(A(a, 0), A(a, 5))), alpha_i);
626     var r3 = norm(minus(B(0), B(1)));
627     var r4 = norm(minus(B(0), B(5)));
628     return [OP.dr-min(r1),
629         OP.dr-min(r2),
630         OP.dr-r3,
631         OP.dr-r4];
632   }
633
634   /**
635    * Computes the constraint values for the optimization problem based on
          geometry and operational parameters.
636    * @param {Array<number>} x - An array of geometry parameters.
637    * @param {Object} OP - Optimization parameters.
638    * @param {Object} crawler - Crawler object.
639    * @returns {Array<number>} An array of constraint values.
640    */
641   constraints(x, OP, crawler) {
642     this.updateGeometry(x);
643     var MP = this.MP;
644
645     var c1 = this.checkIntersectionNumerical(OP);
646
647     var flag = zeros(OP.N);
648     var M_m = NaNs(5*OP.N);
649     var F_a = NaNs(5*OP.N);
650     var F_r = NaNs(5*OP.N);
651     var rm = [];
652     var lambda_ujm = zeros(OP.N);
653     var lambda_ljm = zeros(OP.N);
654
655     for(var i = 0; i < OP.N; i++) {
656       this.setParameters([...OP.x[i]]);
657       if(!any(MP.e)) {
658         this.checkJoints();
659         lambda_ujm[i] = max(MP.lambda_p); lambda_ljm[i] = max(MP.lambda_b);
660         if(lambda_ujm[i] < MP.lambda_max && lambda_ljm[i] < MP.lambda_max) {
661           this.getJacobian();
662           if(!any(MP.e)) {
663             flag[i] = false;
664             this.getForces();
665             this.getTorquesByForce();
666             M_m[i] = max(abs(MP.M_m));
667             var [F_ri, F_ai] = this.getActuatorLoad();
668             F_r[i] = max(abs(F_ri));
669             F_a[i] = max(abs(F_ai));
670             var res = this.findMaxParameters(
671               crawler, OP.x_lim, OP.k, [3, 4], false);
672             rm[i] = res[0];
673           } else {
674             flag[i] = true;
675           }
```

```
        } else {
          flag[i] = true;
        }
      } else {
        flag[i] = true;
      }
      if(flag[i]) {
        M_m[i] = NaN;
      }
    }

    for(var i = 0; i < OP.N; i++) {
      for(var j = 0; j < 2; j++) {
        if(!flag[i]) {
          var x = [...OP.x[i]];
          x[3] = rm[i][0][j];
          this.setParameters(x);
          this.getForces();
          this.getTorquesByForce();
          M_m[OP.N*(j+1)+i] = max(abs(MP.M_m));
          var [F_ri, F_ai] = this.getActuatorLoad();
          F_r[OP.N*(j+1)+i] = max(abs(F_ri));
          F_a[OP.N*(j+1)+i] = max(abs(F_ai));
        }
      }
      for(var j = 0; j < 2; j++) {
        if(!flag[i]) {
          var x = [...OP.x[i]];
          x[4] = rm[i][1][j];
          this.setParameters(x);
          this.getForces();
          this.getTorquesByForce();
          M_m[OP.N*(j+3)+i] = max(abs(MP.M_m));
          var [F_ri, F_ai] = this.getActuatorLoad();
          F_r[OP.N*(j+3)+i] = max(abs(F_ri));
          F_a[OP.N*(j+3)+i] = max(abs(F_ai));
        }
      }
    }

    var c2 = minus(M_m, OP.M_max);
    var c3 = minus(F_r, OP.Frd_R);
    var c4 = minus(F_a, OP.Fad_R);
    var c5 = [...minus(lambda_ujm, MP.lambda_max),
          ...minus(lambda_ljm, MP.lambda_max)];
    return [...c1, ...c2, ...c3, ...c4, ...c5];
  }

  /**
   * Computes the fitness value for the optimization based on geometry and
      optimization parameters.
   * @param {Array<number>} x - An array of geometry parameters.
   * @param {Object} OP - Optimization parameters.
   * @param {Object} crawler - Crawler object.
   * @returns {number} The computed fitness value.
   */
  fitness(x, OP, crawler) {
    var MP = this.MP;
    this.updateGeometry(x);
```

```
734
735     var eta = zeros(OP.N);
736     var phi = zeros(OP.N);
737     var theta = zeros(OP.N);
738     var rm = [];
739     var x_gmax = [OP.x_gmax[3], OP.x_gmax[4]];
740     var flag = false;
741
742     for(var i = 0; i < OP.N; i++) {
743       this.setParameters([...OP.x[i]]);
744       this.checkJoints();
745       if(!any(MP.e)) {
746         this.getJacobian();
747         if(!any(MP.e)) {
748           flag = false;
749           this.getJacobianNormalized();
750           this.getDexterity();
751           eta[i] = MP.eta;
752           var res = this.findMaxParameters(
753             crawler, OP.x_lim, OP.k, [3, 4], false);
754           rm[i] = res[0];
755           rm[i][0] = rm[i][0].map((val, idx) =>
756             (val > x_gmax[idx] ? x_gmax[idx] : val));
757           rm[i][1] = rm[i][1].map((val, idx) =>
758             (val < -x_gmax[idx] ? -x_gmax[idx] : val));
759           phi[i] = (rm[i][0][0] - rm[i][0][1]) / 2;
760           theta[i] = (rm[i][1][0] - rm[i][1][1]) / 2;
761         } else {
762           flag = true;
763         }
764       } else {
765         flag = true;
766       }
767       if(flag) {
768         eta[i] = 0;
769         phi[i] = 0;
770         theta[i] = 0;
771       }
772     }
773
774     return (1-sum(multiplyEl([mean(eta),
775       ...divideEl([mean(phi), mean(theta)], x_gmax)], OP.b)))*100;
776   }
777
778   /**
779    * Resets the mechanism state.
780    */
781   reset() {
782     var MP = this.MP;
783     var m = MP.m;
784
785     fill(0.0, MP.x, m);
786     fill(0.0, MP.x_o, m);
787     fill(0.0, MP.alpha_w_in, m);
788     fill(0.0, MP.alpha, m);
789     fill(0.0, MP.alpha_o, m);
790   }
791
792   /**
```

```
793     * Finds the maximum parameters in parallel for a given configuration.
794     * @param {string} id - The identifier for the configuration to load.
795     * @param {Array.<Array.<number>>} x_lim - The limits of the search space,
          where each sub-array represents [start, end] for a dimension.
796     * @param {Array.<Array.<number>>} k - The parameters to optimize.
797     * @returns {Promise<Array>} A promise that resolves to the aggregated maximum
          parameters.
798     */
799    async findAllMaxParametersParallel(id, x_lim, k) {
800      var res = await parallel.parfor(0, 5, 1, parallel.getProcessorsNum(),
801        {id, x_lim, k},
802        function() {
803        var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
804        var MP = new MP_6RSS_MECHANISM();
805
806        var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
807        var solver = new MP_6RSS_SOLVER(MP);
808        var crawler = spaceSearch();
809
810        solver.getGeometry();
811      }, function(i) {
812        MP.x = zeros(6);
813        return solver.findMaxParameter(crawler, i, x_lim, k);
814      });
815      return this.getAllMaxParamters(res);
816    }
817
818    /**
819     * Finds the maximum parameters based on the crawler.
820     * @param {Object} crawler - Crawler object.
821     * @param {Array} x_lim - Limits for x.
822     * @param {Array} k - Parameters k.
823     * @returns {Array} - Maximum parameters and related data.
824     */
825    findAllMaxParameters(crawler, x_lim, k) {
826      this.MP.x = zeros(6);
827      return this.findMaxParameters(crawler, x_lim, k, range(0, 5));
828    }
829
830    /**
831     * Finds the maximum parameter values for the specified indices within given
          limits.
832     * @param {Object} crawler - The crawler instance used to perform the search.
833     * @param {Array} x_lim - An array of parameter limits.
834     * @param {Array} k - An array of scaling factors or step sizes.
835     * @param {Array<number>} ids - An array of parameter indices to optimize.
836     * @param {boolean} [change_z=true] - Indicates whether to change the z
          parameter.
837     * @returns {Array} The maximum parameters found for the specified indices.
838     */
839    findMaxParameters(crawler, x_lim, k, ids, change_z = true) {
840      var res = [];
841      var x = [...this.MP.x];
842      for(var p = 0; p < ids.length; p++) {
843        this.setParameters(x);
844        var i = ids[p];
845        res[p] = this.findMaxParameter(crawler, i, x_lim, k, change_z);
846      }
847      return this.getMaxParamters(res, ids, change_z);
```

```
848   }
849
850   /**
851    * Finds the maximum parameter values for a given parameter index within
          specified limits.
852    * @param {Object} crawler - The crawler instance used to perform the search.
853    * @param {number} i - The index of the parameter to optimize.
854    * @param {Array} x_lim - An array of parameter limits.
855    * @param {Array} k - An array of scaling factors or step sizes.
856    * @param {boolean} [change_z=true] - Indicates whether to change the z
          parameter.
857    * @returns {Array} An array containing Nin and Nout, representing the inner
          and outer parameter sets.
858    */
859   findMaxParameter(crawler, i, x_lim, k, change_z = true) {
860     var x_in = [...this.MP.x];
861     var obj = this;
862     var d = k.length;
863     var Nin, Nout;
864     if(i != 2 && change_z) {
865       var x_limi = reshape(concatRow(2, x_lim[i], x_lim[2]), 2, 2);
866       var ki = reshape(concatCol(d, column(k, i), column(k, 2)), d, 2);
867       [Nin, Nout] = crawler.run(x_limi, ki, function(x) {
868         var p = x_in;
869         p[i] = x[0];
870         p[2] = x[1];
871         obj.setParameters(p);
872         obj.checkJoints();
873         return !any(obj.MP.e);
874       });
875     } else {
876       [Nin, Nout] = crawler.run([x_lim[i]], column(k, i).map((a) => [a]),
              function(x) {
877         var p = x_in;
878         p[i] = x[0];
879         obj.setParameters(p);
880         obj.checkJoints();
881         return !any(obj.MP.e);
882       });
883     }
884     return [Nin, Nout];
885   }
886
887   /**
888    * Gets the maximum parameters based on the crawler results.
889    * @param {Array} res - Results.
890    * @returns {Array} - Maximum parameters and related data.
891    */
892   getAllMaxParamters(res) {
893     return getMaxParamters(res, range(0, 5));
894   }
895
896   /**
897    * Processes the results from parameter search to extract maximum and minimum
          parameter values.
898    * @param {Array} res - The results from `findMaxParameter` for each parameter
          index.
899    * @param {Array<number>} ids - The indices of parameters that were optimized.
```

```
900     * @param {boolean} [change_z=true] - Indicates whether to consider the z
          parameter.
901     * @returns {Array} An array containing maximum and minimum parameter values,
          z-values, count `j`, and combined `N`.
902     */
903    getMaxParamters(res, ids, change_z = true) {
904      var r = zeros(ids.length, 2);
905      var zr = zeros(ids.length, 2);
906      var N = [];
907      var j = 0;
908      for(var p = 0 ; p < ids.length; p++) {
909        var i = ids[p];
910        var [Nin, Nout] = res[p];
911        N[p] = [Nin, Nout];
912        j += Nin.length+Nout.length;
913        var xi = column(Nin, 0);
914        var [x_max, i_max] = maxi(xi);
915        var [x_min, i_min] = mini(xi);
916        r[p][0] = x_max;
917        r[p][1] = x_min;
918        if(i != 2 && change_z) {
919          var zi = column(Nin, 1);
920          zr[p][0] = zi[i_max];
921          zr[p][1] = zi[i_min];
922        } else {
923          zr[p][0] = 0;
924          zr[p][1] = 0;
925        }
926      }
927      return [r, zr, j, N];
928    }
929
930    /**
931     * Sets the callback function for iterations.
932     * @param {Function} callback - Callback function.
933     */
934    setIterationCallback(callback) {
935      this.iterationCallback = callback;
936    }
937
938    /**
939     * Runs a regression test and logs the results.
940     * @param {string} [filename='regression_results.txt'] - Output file name for
          results.
941     */
942    runRegressionTest(filename = 'regression_results.txt') {
943      var MP = this.MP;
944      var m = MP.m;
945
946      // Log file
947      var log = '';
948
949      // Uglovi donjih poluga
950      var alpha_w_in = [10, -2, 3, 4, -4, 1];
951
952      // Proracun
953      log += disp("\n Full regression test");
954
955      // Input
```

```
956       log += disp("\n --------------------\n Input\n");
957       log += dispRowVector('alpha_w', alpha_w_in, m);
958
959       // getGeometry()
960       log += disp("\n --------------------\n getGeometry()\n");
961       this.getGeometry();
962
963       log += dispMatrix('Rm2_i_ll', MP.Rm2_i_ll, 3, m);
964       log += dispMatrix('Rm1_i_ll', MP.Rm1_i_ll, 3, m);
965       log += dispMatrix('i_ll0', MP.i_ll0, 3, m);
966       log += dispMatrix('P', MP.P, 3, m);
967       log += dispMatrix('B', MP.B, 3, m);
968       log += dispMatrix('A', MP.A, 3, m);
969       log += dispMatrix('Rm2_BA', MP.Rm2_BA, 3, m);
970       log += dispMatrix('Rm1_BA', MP.Rm1_BA, 3, m);
971       log += dispRowVector('e', MP.e, m);
972
973       // setAlpha()
974       log += disp("\n --------------------\n setAlpha()\n");
975       log = this.setAlpha(alpha_w_in, true, log);
976
977       log += dispMatrix('J_x', MP.J_x, 2*3, m);
978       log += dispMatrix('invJ_x', MP.invJ_x, 2*3, m);
979       log += dispMatrix('J_a', MP.J_a, 2*3, m);
980       log += dispMatrix('J', MP.J, 2*3, m);
981       log += dispRowVector('x', MP.x, m);
982       log += disp(' n_iter = ' + MP.n_iter);
983       log += dispRowVector('e', MP.e, m);
984
985       // checkJoints()
986       log += disp("\n --------------------\n checkJoints()\n");
987       this.checkJoints();
988
989       log += dispRowVector('lambda_p', MP.lambda_p, m);
990       log += dispRowVector('lambda_b', MP.lambda_b, m);
991       log += dispRowVector('e', MP.e, m);
992
993       // setParameters()
994       log += disp("\n --------------------\n setParameters()\n");
995       this.setParameters(MP.x);
996
997       log += dispMatrix('R', MP.R, 3, 3);
998       log += dispMatrix('Pb', MP.Pb, 3, m);
999       log += dispMatrix('A', MP.A, 3, m);
1000      log += dispMatrix('i_jpb', MP.i_jpb, 3, m);
1001      log += dispMatrix('i_ll', MP.i_ll, 3, m);
1002      log += dispRowVector('alpha', multiplyEl(MP.alpha, r2d), m);
1003      log += dispRowVector('e', MP.e, m);
1004
1005      // Kraj programa
1006      log += disp("\n --------------------");
1007
1008      // Write log file
1009      writeFile(filename, log);
1010
1011      this.reset();
1012    }
1013
1014    /**
```

```
1015      * Logs the current state of the mechanism.
1016      * @param {string} [filename='current_state.txt'] - Output file name for
              current state.
1017      */
1018     printState(filename = 'current_state.txt') {
1019       var MP = this.MP;
1020       var m = MP.m;
1021 
1022       // Log file
1023       var log = '';
1024 
1025       // Current state
1026       log += disp("\n Current state\n\n --------------------");
1027 
1028       log += dispRowVector('alpha', multiplyEl(MP.alpha, r2d), m);
1029       log += dispRowVector('e', MP.e, m);
1030       log += dispRowVector('x', MP.x, m);
1031       log += dispRowVector('x_o', MP.x_o, m);
1032       log += dispRowVector('alpha_w', MP.alpha_w_in, m);
1033       log += dispMatrix('J_x', MP.J_x, 2 * 3, m);
1034       log += dispMatrix('invJ_x', MP.invJ_x, 2 * 3, m);
1035       log += dispMatrix('J_a', MP.J_a, 2 * 3, m);
1036       log += dispMatrix('J', MP.J, 2 * 3, m);
1037       log += dispRowVector('x', MP.x, m);
1038       log += disp(" n_iter = " + MP.n_iter);
1039 
1040       // Write log file
1041       writeFile(filename, log);
1042     }
1043 }
1044 
1045 exports.MP_6RSS_SOLVER = MP_6RSS_SOLVER;
```

Listing A.1 - SOLVER_MP_6RSS.jsl

B

C++ modul SolverMP6RSS

U sledećim listinzima su dati svi neophodni kodovi za `C++` modul `SolverMP6RSS` koji se poziva iz `JSLAB`-a.

Za kompajliranje modula koristi se `JSLAB` funkcija `compileNapi()`. Nakon kompajliranja, modul se može koristiti kao običan `Node.js` modul (korišćenjem funkcije `require()`).

Prva datoteka, `package.json` (listing B.1), sadrži osnovne informacije o paketu i njegovoj instalaciji, uključujući definisanje drugih potrebnih paketa.

Datoteka `binding.gyp` (listing B.2) pruža konfiguraciju za `node-gyp`, alat koji omogućava kompajliranje `C++` koda u binarni modul kompatibilan sa `Node.js` okruženjem.

Header datoteka `solver-mp-6rss.h` (listing B.3) sadrži deklaraciju klase i metoda koje se koriste za rešavanje inverznog i direktnog kinematičkog problema *6RSS mehanizma*.

Izvorna datoteka `solver-mp-6rss.cpp` (listing B.4) sadrži implementaciju funkcija i metoda deklarisanih u `solver-mp-6rss.h`.

Datoteka `math-functions.h` (listing B.5) sadrži deklaracije pomoćnih matematičkih funkcija koje se koriste u okviru algoritma. Ovo uključuje funkcije za vektorsku i matričnu algebru.

U datoteci `print-functions.h` (listing B.6) nalaze se pomoćne funkcije za formatiranje i ispis rezultata. Ove funkcije olakšavaju praćenje toka izvršavanja programa i uklanjanje grešaka.

```
{
  "dependencies": {
    "node-addon-api": "8.2.1"
  }
}
```

Listing B.1 - `package.json`

```
{
  "targets": [
    {
      "target_name": "solver_mp_6rss",
      "sources": [
        "solver-mp-6rss.cpp"
      ],
      "include_dirs": [
        "<(module_root_dir)",
        "<!@(node -p \"require('node-addon-api').include\")",
        "<(module_root_dir)/../eigen-3.4.0/"
      ],
      "cflags!": [
```

```
        "-fno-exceptions"
      ],
      "cflags_cc!": [
        "-fno-exceptions"
      ],
      "defines": [
        "NAPI_DISABLE_CPP_EXCEPTIONS"
      ],
      "msvs_settings": {
        "VCCLCompilerTool": {
          "AdditionalOptions": [
            "-std:c++17"
          ]
        }
      }
    }
  ]
}
```

Listing B.2 - `binding.gyp`

```
// SolverMP6RSS - solver-mp-6rss.h
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// ---------------------

#ifndef SOLVER_MP_6RSS_H
#define SOLVER_MP_6RSS_H

//#define DEBUG_SOLVER_MP_6RSS
//#define DEBUG_SOLVER_MP_6RSS_LEVEL 0
//#define PROFILE_SOLVER_MP_6RSS

#include <napi.h>
#include <chrono>
#include <thread>
#include <Windows.h>
#include <ctime>
#include <iomanip>
#include <sstream>
#include <string>
#include <fstream>
#include <iostream>
#include <filesystem>
#include <Eigen/Dense>

namespace solver_mp_6rss_ns {

using namespace std;
using namespace std::chrono;
using namespace Eigen;

// Basic variables
#define pi 3.14159265358979323846 // [rad] mathematical constant pi
#define d2r pi/180.0 // [-] deg to rad conversion constant
#define r2d 180.0/pi // [-] rad to deg conversion constant
const double inf = numeric_limits<double>::infinity(); // [-] infinity value

#define m 6 // [-] number of actuators
```

```
40
41 enum SOLVER_STATE {
42   NORMAL_STATE = 0,
43   GET_ALPHA_FAILED,
44   LAMBDA_P_ERROR,
45   LAMBDA_B_ERROR,
46   LEVER_COLLISION,
47   MU_UL_ERROR,
48   GET_PARAEMTERS_FAILED,
49   JACOBIAN_INVERSE_FAILED,
50   GET_PARAEMTERS_MAX_ITER
51 };
52
53 class SolverMP6RSS : public Napi::ObjectWrap<SolverMP6RSS> {
54  public:
55   static Napi::Object Init(Napi::Env env, Napi::Object exports);
56   SolverMP6RSS(const Napi::CallbackInfo& info);
57   ~SolverMP6RSS();
58
59   Napi::Value setAlphaJS(const Napi::CallbackInfo& info);
60   Napi::Value getCurrentAlphaJS(const Napi::CallbackInfo& info);
61   Napi::Value getIterParamsJS(const Napi::CallbackInfo& info);
62   Napi::Value setParametersJS(const Napi::CallbackInfo& info);
63   Napi::Value getCurrentParametersJS(const Napi::CallbackInfo& info);
64   void getGeometryJS(const Napi::CallbackInfo& info);
65   void setIterParamsJS(const Napi::CallbackInfo& info);
66   void resetJS(const Napi::CallbackInfo& info);
67   void runRegressionTestJS(const Napi::CallbackInfo& info);
68   void printStateJS(const Napi::CallbackInfo& info);
69
70   void setAlpha(double* , bool debug = false, ofstream* of = nullptr);
71   void getAlpha();
72   void getGeometry();
73   void getJacobian();
74   void getParameters(bool debug = false, ofstream* of = nullptr);
75   void setParameters(double* );
76   void checkJoints();
77   void reset();
78   void runRegressionTest(string filename = "regression_results.txt");
79   void printState(string filename = "current_state.txt");
80
81   double a; // [mm] length of the lower lever from Bi to Ai
82   double s; // [mm] length of the upper lever from Ai to Pi
83   double R_p; // [mm] platform radius from O1 to Pi
84   double R_b; // [mm] base radius from O to Bi
85   double gamma_p; // [deg] angle between platform joints
86   double gamma_b; // [deg] angle between base rotary joints
87   double epsilon; // [deg] angle of the motor axis relative to the horizontal
       plane
88   double tau_ll; // [deg] deviation of the lower lever angle from the given
       value
89   double tau_jp; // [deg] deviation of the platform joints angle from the given
       value
90   double mu_ll; // [deg] angle of the lower lever joint axis and motor plane
91   double mu_jp; // [deg] angle of the platform joint axis and the horizontal
       plane
92
93   double lambda_max = 35; // [deg] max allowed joints angle
94
```

```
95    double cg[3] = {0, 0, 0}; // [mm] center of gravity position
96
97    double tau_r = (double)1E-6; // [-] relative error of external parameters
98    double tau_a = (double)1E-4; // [-] absoulte error of external parameters
99    uint8_t n_iter_max = 20; // [-] max number of iterations for direct kinematic
100
101   bool IK_FLAG = false; // [-] Inverse kinematic success flag
102   double sigma_l[m] = {-1, 1, -1, 1, -1, 1};
103   double beta_s[m] = {(double)pi/6, (double)pi/6, 5*(double)pi/6, 5*(double)pi
          /6, -(double)pi/2, -(double)pi/2};
104   double nu_ll[m] = {2*(double)pi/3, -(double)pi/3, -2*(double)pi/3, (double)pi
          /3, 0, (double)pi};
105
106   double alpha[m] = {0}; // [deg] angles of the lower levers
107   double alpha_o[m] = {0}; // [deg] previous values
108   double alpha_w_in[m] = {0}; // [deg] desired angles of the lower levers
109   double x[m] = {0}; // [mm and deg] translation and rotation [x, y, z, phi,
          theta, psi]
110   double x_o[m] = {0}; // previous values
111
112   double delta_p[m], delta_b[m], beta_ll[m], beta_jp[m];
113   double i_m[3*m], i_ll[3*m], i_ll0[3*m], i_jp[3*m], i_jpb[3*m];
114   double calpha[m], salpha[m], cepsilon, sepsilon;
115   double z0, z1, P[3*m], Pb[3*m], PO[2*3], POb[2*3], B[3*m], A[3*m];
116   double BPb[3*m], l[m], D[m], E[m], F[m], G[m];
117   double R[3*3], RP[3*m], RPO[3*m], Ri_jp[3*m];
118   double Rm1_i_ll[3*m], Rm2_i_ll[3*m], Rm1_BA[3*m], Rm2_BA[3*m], BA_0[3*m];
119   double PbA[3*m], lambda_p[m], lambda_b[m];
120   double cpsi, ctheta, cphi, spsi, stheta, sphi;
121   double i_ul[3*m], m_ul[3*m], tau_r0, dalpha[m], dalpha_r[m], dx[m];
122   double m_m[3*m], BA[3*m], PObPb[3*m];
123   double J_ai[m], J_a[2*3*m], invJ_ai[m], invJ_a[2*3*m];
124   double J[2*3*m], J_x[2*3*m], invJ_x[2*3*m];
125   uint8_t e[m] = {NORMAL_STATE}, n_iter;
126 };
127
128 }// namespace solver_mp_6rss_ns
129
130 #endif // SOLVER_MP_6RSS_H
```

Listing B.3 - `solver-mp-6rss.h`

```
1  // SolverMP6RSS - solver-mp-6rss.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "solver-mp-6rss.h"
8
9  #include "math-functions.h"
10 #include "print-functions.h"
11
12 namespace solver_mp_6rss_ns {
13
14 #ifdef PROFILE_SOLVER_MP_6RSS
15 // Function to start the timer and return the start time
16 time_point<steady_clock> tic() {
17     return steady_clock::now();
18 }
```

```
19
20 // Function to stop the timer and return the elapsed time
21 long toc(const time_point<steady_clock>& startTime) {
22     return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
23 }
24 #endif
25
26 // Function to get currebt time
27 std::string getCurrentTime() {
28   // get current time
29   auto now = system_clock::now();
30
31   // get number of milliseconds for the current second
32   // (remainder after division into seconds)
33   auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
34
35   // convert to std::time_t in order to convert to std::tm (broken time)
36   auto timer = system_clock::to_time_t(now);
37
38   // convert to broken time
39   std::tm bt = *std::localtime(&timer);
40
41   std::ostringstream oss;
42
43   oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
44   oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
45
46   return oss.str();
47 }
48
49 // Function to console log data
50 int consoleLog(uint8_t level, const char* format, ...) {
51 #ifdef DEBUG_SOLVER_MP_6RSS_LEVEL
52   if(level <= DEBUG_SOLVER_MP_6RSS_LEVEL) {
53     printf("\033[0;33m[%s SolverMP6RSS]\033[0m ", getCurrentTime().c_str());
54     va_list vl;
55     va_start(vl, format);
56     auto ret = vprintf(format, vl);
57     va_end(vl);
58     printf("\n");
59     return ret;
60   }
61 #endif
62   return 0;
63 }
64
65 // SolverMP6RSS()
66 // Object constructor
67 // -------------------
68 SolverMP6RSS::SolverMP6RSS(const Napi::CallbackInfo& info)
69     : Napi::ObjectWrap<SolverMP6RSS>(info) {
70 #ifdef DEBUG_SOLVER_MP_6RSS
71   consoleLog(0, "Called constructor");
72 #endif
73
74   // Set geometry
75   getGeometryJS(info);
76 }
77
```

```
78  // ~SolverMP6RSS()
79  // Object destructor
80  // --------------------
81  SolverMP6RSS::~SolverMP6RSS() {
82  }
83
84  // Init() function
85  // --------------------
86  Napi::Object SolverMP6RSS::Init(Napi::Env env, Napi::Object exports) {
87    Napi::Function func =
88      DefineClass(env,
89        "SolverMP6RSS",
90         {InstanceMethod("printState", &SolverMP6RSS::printStateJS),
91          InstanceMethod("runRegressionTest", &SolverMP6RSS::runRegressionTestJS),
92          InstanceMethod("setParameters", &SolverMP6RSS::setParametersJS),
93          InstanceMethod("getCurrentParameters", &SolverMP6RSS::
                getCurrentParametersJS),
94          InstanceMethod("setAlpha", &SolverMP6RSS::setAlphaJS),
95          InstanceMethod("getCurrentAlpha", &SolverMP6RSS::getCurrentAlphaJS),
96          InstanceMethod("getIterParams", &SolverMP6RSS::getIterParamsJS),
97          InstanceMethod("setIterParams", &SolverMP6RSS::setIterParamsJS),
98          InstanceMethod("getGeometry", &SolverMP6RSS::getGeometryJS),
99          InstanceMethod("reset", &SolverMP6RSS::resetJS)
100        });
101
102   Napi::FunctionReference* constructor = new Napi::FunctionReference();
103   *constructor = Napi::Persistent(func);
104   env.SetInstanceData(constructor);
105
106   exports.Set("SolverMP6RSS", func);
107   return exports;
108 }
109
110 // getGeometryJS() function
111 // --------------------
112 void SolverMP6RSS::getGeometryJS(const Napi::CallbackInfo& info) {
113 #ifdef DEBUG_SOLVER_MP_6RSS
114   consoleLog(0, "Call to getGeometryJS()");
115 #endif
116
117   Napi::Object MP = info[0].As<Napi::Object>();
118
119   a = MP.Get("a").As<Napi::Number>().DoubleValue();
120   s = MP.Get("s").As<Napi::Number>().DoubleValue();
121   R_p = MP.Get("R_p").As<Napi::Number>().DoubleValue();
122   R_b = MP.Get("R_b").As<Napi::Number>().DoubleValue();
123   gamma_p = MP.Get("gamma_p").As<Napi::Number>().DoubleValue();
124   gamma_b = MP.Get("gamma_b").As<Napi::Number>().DoubleValue();
125   epsilon = MP.Get("epsilon").As<Napi::Number>().DoubleValue();
126   tau_ll = MP.Get("tau_ll").As<Napi::Number>().DoubleValue();
127   tau_jp = MP.Get("tau_jp").As<Napi::Number>().DoubleValue();
128   mu_ll = MP.Get("mu_ll").As<Napi::Number>().DoubleValue();
129   mu_jp = MP.Get("mu_jp").As<Napi::Number>().DoubleValue();
130   lambda_max = MP.Get("lambda_max").As<Napi::Number>().DoubleValue();
131
132   Napi::Array array = MP.Get("cg").As<Napi::Array>();
133   for(uint8_t i = 0; i < 3; i++) {
134     alpha_w_in[i] = array.Get(i).ToNumber().DoubleValue();
135   }
```

```
136
137   getGeometry();
138 }
139
140 // setIterParamsJS() function
141 // --------------------
142 void SolverMP6RSS::setIterParamsJS(const Napi::CallbackInfo& info) {
143 #ifdef DEBUG_SOLVER_MP_6RSS
144   consoleLog(0, "Call to setIterParamsJS()");
145 #endif
146
147   tau_r = info[0].As<Napi::Number>().DoubleValue();
148   tau_a = info[1].As<Napi::Number>().DoubleValue();
149   n_iter_max = info[2].As<Napi::Number>().Uint32Value();
150 }
151
152 // getIterParamsJS() function
153 // --------------------
154 Napi::Value SolverMP6RSS::getIterParamsJS(const Napi::CallbackInfo& info) {
155 #ifdef DEBUG_SOLVER_MP_6RSS
156   consoleLog(0, "Call to getIterParamsJS()");
157 #endif
158
159   Napi::Object result = Napi::Object::New(info.Env());
160   result.Set("tau_r", Napi::Number::New(info.Env(), tau_r));
161   result.Set("tau_a", Napi::Number::New(info.Env(), tau_a));
162   result.Set("n_iter_max", Napi::Number::New(info.Env(), n_iter_max));
163   return result;
164 }
165
166 // getCurrentAlphaJS() function
167 // --------------------
168 Napi::Value SolverMP6RSS::getCurrentAlphaJS(const Napi::CallbackInfo& info) {
169 #ifdef DEBUG_SOLVER_MP_6RSS
170   consoleLog(0, "Call to getCurrentAlphaJS()");
171 #endif
172
173   Napi::Object result = Napi::Object::New(info.Env());
174   Napi::Array alpha_out = Napi::Array::New(info.Env());
175   Napi::Array e_out = Napi::Array::New(info.Env());
176   for(uint8_t i = 0; i < m; i++) {
177     alpha_out.Set(i, alpha[i]*r2d);
178     e_out.Set(i, e[i]);
179   }
180   result.Set("alpha", alpha_out);
181   result.Set("e", e_out);
182   return result;
183 }
184
185 // setParametersJS() function
186 // --------------------
187 Napi::Value SolverMP6RSS::setParametersJS(const Napi::CallbackInfo& info) {
188 #ifdef DEBUG_SOLVER_MP_6RSS
189   consoleLog(1, "Call to setParametersJS()");
190 #endif
191
192   copyElements(x, x_o, m);
193
194   Napi::Array array = info[0].As<Napi::Array>();
```

```
195   for(uint8_t i = 0; i < m; i++) {
196     x[i] = array.Get(i).ToNumber().DoubleValue();
197   }
198
199   getAlpha();
200
201   Napi::Object result = Napi::Object::New(info.Env());
202   Napi::Array alpha_out = Napi::Array::New(info.Env());
203   Napi::Array e_out = Napi::Array::New(info.Env());
204   for(uint8_t i = 0; i < m; i++) {
205     alpha_out.Set(i, alpha[i]*r2d);
206     e_out.Set(i, e[i]);
207   }
208   result.Set("alpha", alpha_out);
209   result.Set("e", e_out);
210   return result;
211 }
212
213 // setAlphaJS() function
214 // --------------------
215 Napi::Value SolverMP6RSS::setAlphaJS(const Napi::CallbackInfo& info) {
216 #ifdef DEBUG_SOLVER_MP_6RSS
217   consoleLog(1, "Call to setAlphaJS()");
218 #endif
219
220   Napi::Array array = info[0].As<Napi::Array>();
221   for(uint8_t i = 0; i < m; i++) {
222     alpha_w_in[i] = array.Get(i).ToNumber().DoubleValue();
223   }
224
225   getParameters();
226
227   Napi::Object result = Napi::Object::New(info.Env());
228   Napi::Array x_out = Napi::Array::New(info.Env());
229   Napi::Array e_out = Napi::Array::New(info.Env());
230   for(uint8_t i = 0; i < m; i++) {
231     x_out.Set(i, x[i]);
232     e_out.Set(i, e[i]);
233   }
234   result.Set("x", x_out);
235   result.Set("e", e_out);
236   return result;
237 }
238
239 // getCurrentParametersJS() function
240 // --------------------
241 Napi::Value SolverMP6RSS::getCurrentParametersJS(const Napi::CallbackInfo& info)
      {
242 #ifdef DEBUG_SOLVER_MP_6RSS
243   consoleLog(0, "Call to getCurrentParametersJS()");
244 #endif
245
246   Napi::Object result = Napi::Object::New(info.Env());
247   Napi::Array x_out = Napi::Array::New(info.Env());
248   Napi::Array e_out = Napi::Array::New(info.Env());
249   for(uint8_t i = 0; i < m; i++) {
250     x_out.Set(i, x[i]);
251     e_out.Set(i, e[i]);
252   }
```

```
253   result.Set("x", x_out);
254   result.Set("e", e_out);
255   return result;
256 }
257
258 // resetJS() function
259 // --------------------
260 void SolverMP6RSS::resetJS(const Napi::CallbackInfo& info) {
261 #ifdef DEBUG_SOLVER_MP_6RSS
262   consoleLog(0, "Call to reset()");
263 #endif
264
265   reset();
266 }
267
268 // runRegressionTestJS() function
269 // --------------------
270 void SolverMP6RSS::runRegressionTestJS(const Napi::CallbackInfo& info) {
271 #ifdef DEBUG_SOLVER_MP_6RSS
272   consoleLog(0, "Call to runRegressionTestJS()");
273 #endif
274
275   if(info.Length() == 1) {
276     runRegressionTest(info[0].As<Napi::String>());
277   } else {
278     runRegressionTest();
279   }
280 }
281
282 // printStateJS() function
283 // --------------------
284 void SolverMP6RSS::printStateJS(const Napi::CallbackInfo& info) {
285 #ifdef DEBUG_SOLVER_MP_6RSS
286   consoleLog(0, "Call to printStateJS()");
287 #endif
288
289   if(info.Length() == 1) {
290     printState(info[0].As<Napi::String>());
291   } else {
292     printState();
293   }
294 }
295
296 // getGeometry() function
297 // --------------------
298 void SolverMP6RSS::getGeometry() {
299 #ifdef DEBUG_SOLVER_MP_6RSS
300   consoleLog(0, "Call to getGeometry()");
301 #endif
302
303   cepsilon = cos(epsilon*d2r);
304   sepsilon = sin(epsilon*d2r);
305
306   for(uint8_t j = 0; j < m; j++) {
307     beta_ll[j] = nu_ll[j]+tau_ll*sigma_l[j]*d2r;
308     beta_jp[j] = beta_s[j]+tau_jp*sigma_l[j]*d2r;
309     delta_p[j] = beta_s[j]+gamma_p*sigma_l[j]*d2r;
310     delta_b[j] = beta_s[j]+gamma_b*sigma_l[j]*d2r;
311
```

```
312       i_m[j*3] = -sigma_l[j]*cos(beta_ll[j]+pi/2*sigma_l[j])*cepsilon;
313       i_m[j*3+1] = -sigma_l[j]*sin(beta_ll[j]+pi/2*sigma_l[j])*cepsilon;
314       i_m[j*3+2] = -sigma_l[j]*sepsilon;
315       i_ll[j*3] = cos(beta_ll[j]+pi/2*sigma_l[j])*cos((epsilon+mu_ll)*d2r);
316       i_ll[j*3+1] = sin(beta_ll[j]+pi/2*sigma_l[j])*cos((epsilon+mu_ll)*d2r);
317       i_ll[j*3+2] = sin((epsilon+mu_ll)*d2r);
318       i_jp[j*3] = cos(beta_jp[j])*cos(mu_jp*d2r);
319       i_jp[j*3+1] = sin(beta_jp[j])*cos(mu_jp*d2r);
320       i_jp[j*3+2] = sin(mu_jp*d2r);
321       P[j*3] = R_p*cos(delta_p[j]);
322       P[j*3+1] = R_p*sin(delta_p[j]);
323       P[j*3+2] = 0;
324       B[j*3] = R_b*cos(delta_b[j]);
325       B[j*3+1] = R_b*sin(delta_b[j]);
326       B[j*3+2] = 0;
327
328       A[j*3] = B[j*3]+a*cos(beta_ll[j]);
329       A[j*3+1] = B[j*3+1]+a*sin(beta_ll[j]);
330       A[j*3+2] = B[j*3+2];
331
332       for(uint8_t i = 0; i < 3; i++) {
333         BA_0[j*3+i] = A[j*3+i]-B[j*3+i];
334       }
335     }
336     copyElements(i_ll, i_ll0, 3*m);
337
338     cross3D(i_m, i_ll0, Rm2_i_ll, m);
339     cross3D(i_m, Rm2_i_ll, Rm1_i_ll, m);
340     cross3D(i_m, BA_0, Rm2_BA, m);
341     cross3D(i_m, Rm2_BA, Rm1_BA, m);
342
343     z1 = pow(s, 2)-(pow(P[0]-A[0], 2)+pow(P[1]-A[1], 2));
344     if(z1 > 0) {
345       z0 = sqrt(z1)+A[2];
346     } else {
347       z0 = inf;
348     }
349
350     copyElements(i_jp, i_jpb, 3*m);
351     copyElements(P, Pb, 3*m);
352
353     for(uint8_t j = 0; j < m; j++) {
354       Pb[j*3+2] = Pb[j*3+2]+z0;
355     }
356
357     for(uint8_t i = 0; i < 3; i++) {
358       PO[i] = cg[i];
359       PO[i+3] = 0;
360     }
361   }
362
363   // setParameters() function
364   // --------------------
365   void SolverMP6RSS::setParameters(double *x_in) {
366   #ifdef DEBUG_SOLVER_MP_6RSS
367     consoleLog(1, "Call to setParameters()");
368   #endif
369
370     copyElements(x, x_o, m);
```

```
371   for(uint8_t i = 0; i < m; i++) {
372     x[i] = x_in[i];
373   }
374
375   getAlpha();
376 }
377
378 // setAlpha() function
379 // --------------------
380 void SolverMP6RSS::setAlpha(double *alpha_in, bool debug, ofstream* of) {
381 #ifdef DEBUG_SOLVER_MP_6RSS
382   consoleLog(1, "Call to setAlphaJS()");
383 #endif
384
385   for(uint8_t i = 0; i < m; i++) {
386     alpha_w_in[i] = alpha_in[i];
387   }
388   getParameters(debug, of);
389 }
390
391 // getAlpha() function
392 // --------------------
393 void SolverMP6RSS::getAlpha() {
394 #ifdef DEBUG_SOLVER_MP_6RSS
395   consoleLog(1, "Call to getAlpha()");
396 #endif
397
398   copyElements(alpha, alpha_o, m);
399
400   cphi = cos(x[3]*d2r);
401   ctheta = cos(x[4]*d2r);
402   cpsi = cos(x[5]*d2r);
403   sphi = sin(x[3]*d2r);
404   stheta = sin(x[4]*d2r);
405   spsi = sin(x[5]*d2r);
406
407   R[0] = cpsi*ctheta;
408   R[1] = spsi*ctheta;
409   R[2] = -stheta;
410   R[3] = -spsi*cphi+cpsi*stheta*sphi;
411   R[4] = cpsi*cphi+spsi*stheta*sphi;
412   R[5] = ctheta*sphi;
413   R[6] = spsi*sphi+cpsi*stheta*cphi;
414   R[7] = -cpsi*sphi+spsi*stheta*cphi;
415   R[8] = ctheta*cphi;
416
417   multiply(R, i_jp, Ri_jp, 3, 3, m);
418   multiply(R, P, RP, 3, 3, m);
419   multiply(R, PO, RPO, 3, 3, 2);
420
421   for(uint8_t j = 0; j < 2; j++) {
422     for(uint8_t i = 0; i < 3; i++) {
423       POb[j*3+i] = x[i]+RPO[j*3+i];
424     }
425     POb[j*3+2] = POb[j*3+2]+z0;
426   }
427
428   for(uint8_t j = 0; j < m; j++) {
429     for(uint8_t i = 0; i < 3; i++) {
```

```
430       Pb[j*3+i] = x[i]+RP[j*3+i];
431       BPb[j*3+i] = Pb[j*3+i]-B[j*3+i];
432       i_jpb[j*3+i] = Ri_jp[j*3+i];
433     }
434
435     Pb[j*3+2] = Pb[j*3+2]+z0;
436     BPb[j*3+2] = BPb[j*3+2]+z0;
437
438     l[j] = sqrt(pow(BPb[j*3], 2)+pow(BPb[j*3+1], 2)+pow(BPb[j*3+2], 2));
439     D[j] = pow(l[j], 2)-(pow(s, 2)-pow(a, 2))-2*((Rm1_BA[j*3]+BA_0[j*3])*BPb[j
          *3]+\
440       (Rm1_BA[j*3+1]+BA_0[j*3+1])*BPb[j*3+1]+(Rm1_BA[j*3+2]+BA_0[j*3+2])*BPb[j
            *3+2]);
441     E[j] = 2*(Rm2_BA[j*3]*BPb[j*3]+Rm2_BA[j*3+1]*BPb[j*3+1]+Rm2_BA[j*3+2]*BPb[j
          *3+2]);
442     F[j] = -2*(Rm1_BA[j*3]*BPb[j*3]+Rm1_BA[j*3+1]*BPb[j*3+1]+Rm1_BA[j*3+2]*BPb[j
          *3+2]);
443     G[j] = D[j]/sqrt(pow(E[j], 2)+pow(F[j], 2));
444     alpha[j] = asin(G[j])-atan2(F[j], E[j]);
445
446     calpha[j] = cos(alpha[j]);
447     salpha[j] = sin(alpha[j]);
448
449     for(uint8_t i = 0; i < 3; i++) {
450       A[j*3+i] = B[j*3+i]+BA_0[j*3+i]+(1-calpha[j])*Rm1_BA[j*3+i]+\
451         salpha[j]*Rm2_BA[j*3+i];
452       i_ll[j*3+i] = i_ll0[j*3+i]+(1-calpha[j])*Rm1_i_ll[j*3+i]+\
453         salpha[j]*Rm2_i_ll[j*3+i];
454     }
455
456     if(fabs(G[j]) <= 1) {
457       e[j] = NORMAL_STATE;
458       IK_FLAG = true;
459     } else {
460       e[j] = GET_ALPHA_FAILED;
461       copyElements(x_o, x, m);
462       copyElements(alpha_o, alpha, m);
463     }
464   }
465 }
466
467 // getJacobian() function
468 // --------------------
469 void SolverMP6RSS::getJacobian() {
470 #ifdef DEBUG_SOLVER_MP_6RSS
471   consoleLog(1, "Call to getJacobian()");
472 #endif
473
474   for(uint8_t k = 0; k < 3*m; k++) {
475     i_ul[k] = (Pb[k]-A[k])/s;
476     BA[k] = A[k]-B[k];
477   }
478
479   for(uint8_t j = 0; j < m; j++) {
480     for(uint8_t i = 0; i < 3; i++) {
481       PObPb[j*3+i] = Pb[j*3+i]-POb[1*3+i];
482     }
483   }
484
```

```
485   cross3D(PObPb, i_ul, m_ul, m);
486   for(uint8_t j = 0; j < 3; j++) {
487     for(uint8_t i = 0; i < m; i++) {
488       J_x[j*m+i] = i_ul[j+i*3];
489       J_x[(j+3)*m+i] = m_ul[j+i*3]; // J_x
490     }
491   }
492
493   copyElements(J_x, invJ_x, 2*3*m);
494   Eigen::Map<Matrix<double,m,m,ColMajor>> invJ_x_e(invJ_x);
495
496   if(invJ_x_e.determinant() != 0) {
497     invJ_x_e = invJ_x_e.inverse();
498   } else {
499     fill(JACOBIAN_INVERSE_FAILED, e, m);
500     return;
501   }
502
503   cross3D(i_m, BA, m_m, m);
504   dotColumn(i_ul, m_m, J_ai, 3, m);
505   diag(J_ai, J_a, m); // J_a
506   reciprocal(J_ai, invJ_ai, m);
507   diag(invJ_ai, invJ_a, m); // invJ_a
508
509   multiply(invJ_x, J_a, J, m, 2*3, m); // Jacobian = inv(J_x)*J_a
510 }
511
512 // getParameters() function
513 // ---------------------
514 void SolverMP6RSS::getParameters(bool debug, ofstream* of) {
515 #ifdef DEBUG_SOLVER_MP_6RSS
516   consoleLog(1, "Call to getParameters()");
517 #endif
518
519   copyElements(x, x_o, m);
520   copyElements(alpha, alpha_o, m);
521   n_iter = 0;
522
523   for(uint8_t j = 0; j < m; j++) {
524     dalpha[j] = alpha[j]-alpha_w_in[j];
525   }
526   tau_r0 = tau_r*norm(dalpha, m);
527
528   if(debug) {
529     *of << " -----\n";
530   }
531   while(norm(dalpha, m) > tau_r0+tau_a) {
532     if(n_iter >= n_iter_max) {
533       fill(GET_PARAEMTERS_MAX_ITER, e, m);
534       break;
535     } else {
536       n_iter++;
537     }
538     if(debug) {
539       *of << " n_iter = " << num2str("%u", n_iter) << "\n";
540       *of << dispRowVector("x", x, m);
541     }
542
543     getAlpha();
```

```
544       if(any(e, m)) {
545         fill(GET_PARAEMTERS_FAILED, e, m);
546         break;
547       }
548       getJacobian();
549       if(any(e, m)) {
550         break;
551       }
552       if(debug) {
553         *of << dispMatrix("J_x", J_x, 2 * 3, m);
554         *of << dispMatrix("J_a", J_a, 2 * 3, m);
555         *of << dispMatrix("J", J, 2 * 3, m);
556       }
557
558       // dx^k = J*dalpha;
559       for(uint8_t j = 0; j < m; j++) {
560         dalpha[j] = alpha[j]*r2d-alpha_w_in[j];
561         dalpha_r[j] = alpha[j]-alpha_w_in[j]*d2r;
562       }
563       multiply(J, dalpha_r, dx, 2*3, m, 1);
564       if(debug) {
565         *of << dispRowVector("dalpha", dalpha, m);
566         *of << dispRowVector("dx", dx, m);
567         *of << " -----\n";
568       }
569
570       // x^{k+1} = x^k+dx^k
571       for(uint8_t j = 0; j < 3; j++) {
572         x[j] = x[j]-dx[j];
573         x[j+3] = x[j+3]-dx[j+3]*r2d;
574       }
575     }
576     if(any(e, m)) {
577       copyElements(x_o, x, m);
578       copyElements(alpha_o, alpha, m);
579     }
580   }
581
582   // checkJoints() function
583   // ---------------------
584   void SolverMP6RSS::checkJoints() {
585   #ifdef DEBUG_SOLVER_MP_6RSS
586     consoleLog(1, "Call to checkJoints()");
587   #endif
588
589     if(IK_FLAG) {
590       for(uint8_t k = 0; k < 3*m; k++) {
591           PbA[k] = A[k]-Pb[k];
592       }
593
594       for(uint8_t j = 0; j < m; j++) {
595         lambda_p[j] = acos((PbA[j*3]*i_jpb[j*3]+PbA[j*3+1]*i_jpb[j*3+1]+ \
596                PbA[j*3+2]*i_jpb[j*3+2])/s)*180.0/pi-90.0;
597         lambda_b[j] = acos((PbA[j*3]*i_ll[j*3]+PbA[j*3+1]*i_ll[j*3+1]+ \
598                PbA[j*3+2]*i_ll[j*3+2])/s)*180.0/pi-90.0;
599
600         if(fabs(lambda_b[j]) > lambda_max) {
601           e[j] = LAMBDA_B_ERROR;
602         } else if(fabs(lambda_p[j]) > lambda_max) {
```

```
603           e[j] = LAMBDA_P_ERROR;
604         }
605       }
606     }
607 }
608
609 // reset() function
610 // --------------------
611 void SolverMP6RSS::reset() {
612 #ifdef DEBUG_SOLVER_MP_6RSS
613   consoleLog(0, "Call to reset()");
614 #endif
615
616   fill(0.0, x, m);
617   fill(0.0, x_o, m);
618   fill(0.0, alpha_w_in, m);
619   fill(0.0, alpha, m);
620   fill(0.0, alpha_o, m);
621 }
622
623 // runRegressionTest() function
624 // --------------------
625 void SolverMP6RSS::runRegressionTest(string filename) {
626 #ifdef DEBUG_SOLVER_MP_6RSS
627   consoleLog(0, "Call to runRegressionTest()");
628 #endif
629
630   // Open file
631   ofstream of;
632   of.open(filename);
633
634   // Uglovi donjih poluga
635   alpha_w_in[0] = 10;
636   alpha_w_in[1] = -2;
637   alpha_w_in[2] = 3;
638   alpha_w_in[3] = 4;
639   alpha_w_in[4] = -4;
640   alpha_w_in[5] = 1;
641
642   // Proracun
643   of << "\n Full regression test\n";
644
645   // Input
646   of <<  "\n --------------------\n Input\n\n";
647
648   of << dispRowVector("alpha_w", alpha_w_in, m);
649
650   // getGeometry()
651   of <<  "\n --------------------\n getGeometry()\n\n";
652   getGeometry();
653
654   of << dispMatrix("Rm2_i_ll", Rm2_i_ll, 3, m);
655   of << dispMatrix("Rm1_i_ll", Rm1_i_ll, 3, m);
656   of << dispMatrix("i_ll0", i_ll0, 3, m);
657   of << dispMatrix("P", P, 3, m);
658   of << dispMatrix("B", B, 3, m);
659   of << dispMatrix("A", A, 3, m);
660   of << dispMatrix("Rm2_BA", Rm2_BA, 3, m);
661   of << dispMatrix("Rm1_BA", Rm1_BA, 3, m);
```

```
662   of << dispRowVector("e", e, m);
663
664   // setAlpha()
665   of <<  "\n --------------------\n setAlpha()\n\n";
666
667   setAlpha(alpha_w_in, true, &of);
668
669   of << dispMatrix("J_x", J_x, 2*3, m);
670   of << dispMatrix("invJ_x", invJ_x, 2*3, m);
671   of << dispMatrix("J_a", J_a, 2*3, m);
672   of << dispMatrix("J", J, 2*3, m);
673   of << dispRowVector("x", x, m);
674   of << " n_iter = " << num2str("%u", n_iter) << "\n";
675   of << dispRowVector("e", e, m);
676
677   // checkJoints()
678   of <<  "\n --------------------\n checkJoints()\n\n";
679
680   checkJoints();
681
682   of << dispRowVector("lambda_p", lambda_p, m);
683   of << dispRowVector("lambda_b", lambda_b, m);
684   of << dispRowVector("e", e, m);
685
686   // setParameters()
687   of <<  "\n --------------------\n setParameters()\n\n";
688
689   setParameters(x);
690
691   of << dispMatrix("R", R, 3, 3);
692   of << dispMatrix("Pb", Pb, 3, m);
693   of << dispMatrix("A", A, 3, m);
694   of << dispMatrix("i_jpb", i_jpb, 3, m);
695   of << dispMatrix("i_ll", i_ll, 3, m);
696
697   double alpha_r[m];
698   for(uint8_t j = 0; j < m; j++) {
699     alpha_r[j] = alpha[j]*r2d;
700   }
701   of << dispRowVector("alpha", alpha_r, m);
702   of << dispRowVector("e", e, m);
703
704   // Kraj programa
705   of << "\n -------------------- \n";
706
707   // Reset values
708   fill(0.0, x, m);
709   fill(0.0, x_o, m);
710   fill(0.0, alpha_w_in, m);
711   fill(0.0, alpha, m);
712   fill(0.0, alpha_o, m);
713
714   // Close file
715   of.close();
716 }
717
718 // printState() function
719 // --------------------
720 void SolverMP6RSS::printState(string filename) {
```

```
721 #ifdef DEBUG_SOLVER_MP_6RSS
722   consoleLog(0, "Call to printState()");
723 #endif
724
725   // Open file
726   ofstream of;
727   of.open(filename);
728
729   // Current state
730   of << "\n Current state\n\n --------------------\n";
731
732   double alpha_r[m];
733   for(uint8_t j = 0; j < m; j++) {
734     alpha_r[j] = alpha[j]*r2d;
735   }
736
737   of << dispRowVector("alpha", alpha_r, m);
738   of << dispRowVector("e", e, m);
739   of << dispRowVector("x", x, m);
740   of << dispRowVector("x_o", x_o, m);
741   of << dispRowVector("alpha_w", alpha_w_in, m);
742   of << dispMatrix("J_x", J_x, 2*3, m);
743   of << dispMatrix("invJ_x", invJ_x, 2*3, m);
744   of << dispMatrix("J_a", J_a, 2*3, m);
745   of << dispMatrix("J", J, 2*3, m);
746   of << dispRowVector("x", x, m);
747   of << " n_iter = " << num2str("%u", n_iter) << "\n";
748
749   // Close file
750   of.close();
751 }
752
753 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
754   return SolverMP6RSS::Init(env, exports);
755 }
756
757 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
758
759 }  // namespace solver_mp_6rss_ns
```

Listing B.4 - `solver-mp-6rss.cpp`

```
1  // SolverMP6RSS - math-functions.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef MATH_FUNCTIONS_H
8  #define MATH_FUNCTIONS_H
9
10 namespace solver_mp_6rss_ns {
11
12 // cross3D function
13 // Calculate cross product
14 // --------------------
15 void cross3D(double *A_in, double *B_in, double *C_out, uint8_t M) {
16   for(uint8_t j = 0; j < M; j++) {
17     C_out[j*3] = A_in[j*3+1]*B_in[j*3+2]-A_in[j*3+2]*B_in[j*3+1];
18     C_out[j*3+1] = A_in[j*3+2]*B_in[j*3]-A_in[j*3]*B_in[j*3+2];
```

```
19      C_out[j*3+2] = A_in[j*3]*B_in[j*3+1]-A_in[j*3+1]*B_in[j*3];
20    }
21  }
22
23  // norm function
24  // Calculate vector norm
25  // --------------------
26  double norm(double *A_in, uint8_t N) {
27    double S = 0;
28    for(uint8_t i = 0; i < N; i++) {
29      S += pow(A_in[i],2);
30    }
31    return sqrt(S);
32  }
33
34  // dot function
35  // Calculate vector dot product
36  // --------------------
37  double dot(double *A_in, double *B_in, uint8_t N) {
38    double Sout = 0;
39    for(uint8_t i = 0; i < N; i++) {
40       Sout += A_in[i]*B_in[i];
41    }
42    return sqrt(Sout);
43  }
44
45  // dotColumn function
46  // Calculate vector dot product for matrix colums
47  // --------------------
48  void dotColumn(double *A_in, double *B_in, double *C_in, uint8_t N, uint8_t M) {
49    for(uint8_t j = 0; j < M; j++) {
50     C_in[j] = 0;
51      for(uint8_t i = 0; i < N; i++) {
52        C_in[j] += A_in[j*N+i]*B_in[j*N+i];
53      }
54    }
55  }
56
57  // all function
58  // Check if all elements of vector are true
59  // --------------------
60  bool all(bool *A_in, uint8_t N) {
61    for(uint8_t i = 0; i < N; i++) {
62      if(A_in[i] != 1) {
63        return false;
64      }
65    }
66    return true;
67  }
68
69  bool all(uint8_t *A_in, uint8_t N) {
70    for(uint8_t i = 0; i < N; i++) {
71      if(A_in[i] != 1) {
72        return false;
73      }
74    }
75    return true;
76  }
77
```

```
78 // any function
79 // Check if any element of vector is true
80 // --------------------
81 bool any(bool *A_in, uint8_t N) {
82   for(uint8_t i = 0; i < N; i++) {
83     if(A_in[i]) {
84       return true;
85     }
86   }
87   return false;
88 }
89
90 bool any(uint8_t *A_in, uint8_t N) {
91   for(uint8_t i = 0; i < N; i++) {
92     if(A_in[i] > 0) {
93       return true;
94     }
95   }
96   return false;
97 }
98
99 bool any(double *A_in, uint8_t N) {
100   for(uint8_t i = 0; i < N; i++) {
101     if(A_in[i] > 0) {
102       return true;
103     }
104   }
105   return false;
106 }
107
108 bool any(uint8_t *A_in, uint8_t x, uint8_t N) {
109   for(uint8_t i = 0; i < N; i++) {
110     if(A_in[i] == x) {
111       return true;
112     }
113   }
114   return false;
115 }
116
117 // anyg function
118 // Check if any element of vector is greater than value
119 // --------------------
120 bool anyg(uint8_t *A_in, uint8_t x, uint8_t N) {
121   for(uint8_t i = 0; i < N; i++) {
122     if(A_in[i] > x) {
123       return true;
124     }
125   }
126   return false;
127 }
128
129 // anyl function
130 // Check if any element of vector is less than value
131 // --------------------
132 bool anyl(uint8_t *A_in, uint8_t x, uint8_t N) {
133   for(uint8_t i = 0; i < N; i++) {
134     if(A_in[i] < x) {
135       return true;
136     }
```

```
137   }
138   return false;
139 }
140 
141 // fill function
142 // fill vector with value
143 // --------------------
144 void fill(bool x, bool *A_out, uint8_t N) {
145   for(uint8_t i = 0; i < N; i++) {
146     A_out[i] = x;
147   }
148 }
149 
150 void fill(uint8_t x, uint8_t *A_out, uint8_t N) {
151   for(uint8_t i = 0; i < N; i++) {
152     A_out[i] = x;
153   }
154 }
155 
156 void fill(double x, double *A_out, uint8_t N) {
157   for(uint8_t i = 0; i < N; i++) {
158     A_out[i] = x;
159   }
160 }
161 
162 // diag function
163 // Form diagonal matrix from vector
164 // --------------------
165 void diag(double *A_in, double *B_out, uint8_t N) {
166   fill(0.0, B_out, N*N);
167   for(uint8_t i = 0; i < N; i++) {
168     B_out[i+i*N] = A_in[i];
169   }
170 }
171 
172 // scale function
173 // Scale vector with value
174 // --------------------
175 void scale(double *A_in, double x, double *B_out, uint8_t N) {
176   for(uint8_t i = 0; i < N; i++) {
177     B_out[i] = A_in[i]*x;
178   }
179 }
180 
181 // reciprocal function
182 // Reciprocal vector
183 // --------------------
184 void reciprocal(double *A_in, double *B_out, uint8_t N) {
185   for(uint8_t i = 0; i < N; i++) {
186     B_out[i] = 1/A_in[i];
187   }
188 }
189 
190 // multiply function
191 // Matrix multiply
192 // --------------------
193 void multiply(double *A_in, double *B_in, double *C_out,
194     uint8_t N, uint8_t M, uint8_t P) {
195   memset(C_out, 0, M*P*sizeof(C_out));
```

```
196   for(uint8_t i = 0; i < N; i++) {
197     for(uint8_t j = 0; j < M; j++) {
198       for(uint8_t k = 0; k < P; k++) {
199         C_out[k*N+i] += A_in[j*N+i]*B_in[k*M+j];
200       }
201     }
202   }
203 }
204
205 // transpose function
206 // Matrix transpose
207 // --------------------
208 void transpose(double *A_in, double *B_out,uint8_t N, uint8_t M) {
209   for(uint8_t i = 0; i < M; i++) {
210     for(uint8_t j = 0; j < N; j++) {
211       B_out[j*M+i] = A_in[i*N+j];
212     }
213   }
214 }
215
216 // copy function
217 // Matrix copy
218 // --------------------
219 void copyElements(double *A_in, double *B_out, uint8_t N) {
220   memcpy(B_out, A_in, N * sizeof(double));
221 }
222
223 } // namespace solver_mp_6rss_ns
224
225 #endif // MATH_FUNCTIONS_H
```

Listing B.5 - `math-functions.h`

```
1  // SolverMP6RSS - print-functions.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef PRINT_FUNCTIONS_H
8  #define PRINT_FUNCTIONS_H
9
10 namespace solver_mp_6rss_ns {
11
12 // num2str function
13 // Number to string
14 // --------------------
15 string num2str(char prec[], float A) {
16   char buff[100];
17   snprintf(buff, sizeof(buff), prec, A);
18   return buff;
19 }
20
21 string num2str(char prec[], double A) {
22   char buff[100];
23   snprintf(buff, sizeof(buff), prec, A);
24   return buff;
25 }
26
27 string num2str(char prec[], uint8_t A) {
```

```
28    char buff[100];
29    snprintf(buff, sizeof(buff), prec, A);
30    return buff;
31  }
32
33  string num2str(char prec[], uint16_t A) {
34    char buff[100];
35    snprintf(buff, sizeof(buff), prec, A);
36    return buff;
37  }
38
39  string num2str(char prec[], uint32_t A) {
40    char buff[100];
41    snprintf(buff, sizeof(buff), prec, A);
42    return buff;
43  }
44
45  string num2str(char prec[], int8_t A) {
46    char buff[100];
47    snprintf(buff, sizeof(buff), prec, A);
48    return buff;
49  }
50
51  string num2str(char prec[], int16_t A) {
52    char buff[100];
53    snprintf(buff, sizeof(buff), prec, A);
54    return buff;
55  }
56
57  string num2str(char prec[], int32_t A) {
58    char buff[100];
59    snprintf(buff, sizeof(buff), prec, A);
60    return buff;
61  }
62
63  // dispMatrix function
64  // Matrix display
65  // --------------------
66  string dispMatrix(string varname, float *A_in, uint8_t N,  uint8_t M) {
67    string str;
68    str += " " + varname + " = [\n";
69    for(uint8_t i = 0; i < N; i++) {
70      str += "  ";
71      for(uint8_t j = 0; j < M; j++) {
72        str += num2str("%.5f", A_in[j*N+i]);
73        str += " ";
74      }
75      str += "\n";
76    }
77    str += " ]\n";
78    return str;
79  }
80
81  string dispMatrix(string varname, double *A_in, uint8_t N,  uint8_t M) {
82    string str;
83    str += " " + varname + " = [\n";
84    for(uint8_t i = 0; i < N; i++) {
85      str += "  ";
86      for(uint8_t j = 0; j < M; j++) {
```

```
87        str += num2str("%.8f", A_in[j*N+i]);
88        str += " ";
89      }
90      str += "\n";
91    }
92    str += " ]\n";
93    return str;
94  }
95
96  // dispRowVector function
97  // Vector display
98  // --------------------
99  string dispRowVector(string varname, double *A_in, uint8_t M) {
100   string str;
101   str += " " + varname + " = [ ";
102   for(uint8_t j = 0; j < M; j++) {
103     str += num2str("%.8f", A_in[j]);
104     str += " ";
105   }
106   str += "]\n";
107   return str;
108 }
109
110 string dispRowVector(string varname, float *A_in, uint8_t M) {
111   string str;
112   str += " " + varname + " = [ ";
113   for(uint8_t j = 0; j < M; j++) {
114     str += num2str("%.5f", A_in[j]);
115     str += " ";
116   }
117   str += "]\n";
118   return str;
119 }
120
121 string dispRowVector(string varname, uint8_t *A_in, uint8_t M) {
122   string str;
123   str += " " + varname + " = [ ";
124   for(uint8_t j = 0; j < M; j++) {
125     str += num2str("%u", A_in[j]);
126     str += " ";
127   }
128   str += "]\n";
129   return str;
130 }
131
132 string dispRowVector(string varname, uint16_t *A_in, uint8_t M) {
133   string str;
134   str += " " + varname + " = [ ";
135   for(uint8_t j = 0; j < M; j++) {
136     str += num2str("%u", A_in[j]);
137     str += " ";
138   }
139   str += "]\n";
140   return str;
141 }
142
143 string dispRowVector(string varname, uint32_t *A_in, uint8_t M) {
144   string str;
145   str += " " + varname + " = [ ";
```

```
146   for(uint8_t j = 0; j < M; j++) {
147     str += num2str("%u", A_in[j]);
148     str += " ";
149   }
150   str += "]\n";
151   return str;
152 }
153
154 string dispRowVector(string varname, int8_t *A_in, uint8_t M) {
155   string str;
156   str += " " + varname + " = [ ";
157   for(uint8_t j = 0; j < M; j++) {
158     str += num2str("%i", A_in[j]);
159     str += " ";
160   }
161   str += "]\n";
162   return str;
163 }
164
165 string dispRowVector(string varname, int16_t *A_in, uint8_t M) {
166   string str;
167   str += " " + varname + " = [ ";
168   for(uint8_t j = 0; j < M; j++) {
169     str += num2str("%i", A_in[j]);
170     str += " ";
171   }
172   str += "]\n";
173   return str;
174 }
175
176 string dispRowVector(string varname, int32_t *A_in, uint8_t M) {
177   string str;
178   str += " " + varname + " = [ ";
179   for(uint8_t j = 0; j < M; j++) {
180     str += num2str("%i", A_in[j]);
181     str += " ";
182   }
183   str += "]\n";
184   return str;
185 }
186
187 } // namespace solver_mp_6rss_ns
188
189 #endif // PRINT_FUNCTIONS_H
```

Listing B.6 - `print-functions.h`

C

JSLAB klasa MP_6RSS_PLOTTER

Metodom `show()` zajedno sa ostatkom klase `MP_6RSS_PLOTTER` (listing C.1), se prikazuje osnovni kinematički model mehanizma simulatora leta. Ova klasa implementira i sve druge metode za vizuelizaciju mehanizma i rezultata proračuna (kao što je prikaz granične površi radnog prostora mehanizma).

```
/**
 * @file Motion Platform 6RSS Mechanism plotter
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

/**
 * Class for Motion Platform 6RSS Mechanism plotter.
 */
class MP_6RSS_PLOTTER {

  /**
   * Constructor for the 6RSS plotter.
   * @param {Object} MP - Mechanism parameters.
   */
  constructor(MP) {
    this.MP = MP;
    this.traces = [];
    this.updateCallbacks = [];

    this.c1 = rgb2hex(0, 0.6, 1);
    this.c2 = rgb2hex(0.35, 0.35, 0.35);
    this.c3 = rgb2hex(0, 0, 0);
    this.c4 = rgb2hex(1, 0.6, 0);

    this.line_width = 5;
    this.marker_color = 5;
    this.s = 1.2; // scale

    this.head_scale = 0.3;
    this.head_angleFactor = 0.4;
  }

  /**
   * Plots the base of the 6RSS mechanism.
   */
```

```
39    plotBase() {
40      var MP = this.MP;
41      var a = MP.a;
42      var B = reshape(MP.B, 3, 6);
43
44      this.updateCallbacks.push('updateBase');
45
46      // Base up
47      this.traces.push({
48        x: [...B[0], B[0][0]],
49        y: [...B[1], B[1][0]],
50        z: [...B[2], B[2][0]],
51        type: 'scatter3d',
52        mode: 'lines',
53        width: this.line_width,
54        color: this.c1,
55        name: 'b',
56        'showLegend': false
57      });
58
59      // Base down
60      this.traces.push({
61        x: [...B[0], B[0][0]],
62        y: [...B[1], B[1][0]],
63        z: minus([...B[2], B[2][0]], a * 1.1),
64        type: 'scatter3d',
65        mode: 'lines',
66        width: this.line_width,
67        color: this.c1,
68        name: 'b',
69        'showLegend': false
70      });
71
72      for(var i = 0; i < MP.m; i++) {
73        // Base connection
74        this.traces.push({
75          x: [B[0][i], B[0][i]],
76          y: [B[1][i], B[1][i]],
77          z: [B[2][i], B[2][i] - a * 1.1],
78          type: 'scatter3d',
79          mode: 'lines',
80          width: this.line_width,
81          color: this.c1,
82          name: 'b',
83          'showLegend': false
84        });
85      }
86    }
87
88    /**
89     * Plots the entire 6RSS mechanism.
90     */
91    plotMechanism() {
92      var MP = this.MP;
93      var B = reshape(MP.B, 3, 6);
94      var Pb = reshape(MP.Pb, 3, 6);
95      var A = reshape(MP.A, 3, 6);
96
97      this.updateCallbacks.push('updateMechanism');
```

```
98
99      this.plotBase();
100
101     for(var i = 0; i < MP.m; i++) {
102       // Linear leg
103       this.traces.push({
104         x: [B[0][i], Pb[0][i]],
105         y: [B[1][i], Pb[1][i]],
106         z: [B[2][i], Pb[2][i]],
107         type: 'scatter3d',
108         mode: 'lines',
109         width: this.line_width/2,
110         color: this.c3,
111         name: 'BP',
112         line: {
113           dash: 'dash'
114         },
115         'showLegend': false
116       });
117
118       // Lower leg
119       this.traces.push({
120         x: [B[0][i], A[0][i]],
121         y: [B[1][i], A[1][i]],
122         z: [B[2][i], A[2][i]],
123         type: 'scatter3d',
124         mode: 'lines',
125         width: this.line_width,
126         color: this.c2,
127         name: 'BA',
128         'showLegend': false
129       });
130
131       // Upper leg
132       this.traces.push({
133         x: [A[0][i], Pb[0][i]],
134         y: [A[1][i], Pb[1][i]],
135         z: [A[2][i], Pb[2][i]],
136         type: 'scatter3d',
137         mode: 'lines',
138         width: this.line_width,
139         color: this.c2,
140         name: 'AP',
141         'showLegend': false
142       });
143     }
144
145     // Platform
146     this.traces.push({
147       x: [...Pb[0], Pb[0][0]],
148       y: [...Pb[1], Pb[1][0]],
149       z: [...Pb[2], Pb[2][0]],
150       type: 'scatter3d',
151       mode: 'lines',
152       width: this.line_width,
153       color: this.c1,
154       name: 'p',
155       'showLegend': false
156     });
```

```
157
158     // A
159     this.traces.push({
160       x: A[0],
161       y: A[1],
162       z: A[2],
163       type: 'scatter3d',
164       mode: 'markers',
165       width: this.line_width,
166       color: this.c1,
167       name: 'A',
168       marker: {
169         color: this.c4,
170         size: this.marker_color,
171         symbol: 'circle',
172         line: {
173           color: this.c2,
174           width: this.line_width/2
175         },
176       },
177       'showLegend': false
178     });
179
180     // P
181     this.traces.push({
182       x: Pb[0],
183       y: Pb[1],
184       z: Pb[2],
185       type: 'scatter3d',
186       mode: 'markers',
187       width: this.line_width,
188       color: this.c1,
189       name: 'P',
190       marker: {
191         color: this.c4,
192         size: this.marker_color,
193         symbol: 'circle',
194         line: {
195           color: this.c2,
196           width: this.line_width/2
197         },
198       },
199       'showLegend': false
200     });
201
202     // B
203     this.traces.push({
204       x: B[0],
205       y: B[1],
206       z: B[2],
207       type: 'scatter3d',
208       mode: 'markers',
209       width: this.line_width,
210       color: this.c1,
211       name: 'B',
212       marker: {
213         color: this.c2,
214         size: this.marker_color,
215         symbol: 'circle',
```

```
        line: {
          color: this.c2,
          width: this.line_width/2
        },
      },
      'showLegend': false,
    });
  }

  /**
   * Plots the names of points in the mechanism.
   * @param {Object} opts_in - Optional parameters for plotting.
   */
  plotPointNames(opts_in) {
    var MP = this.MP;

    this.updateCallbacks.push('updatePointNames');

    var opts = {
      text_dx: MP.a/3,
      text_dy: MP.a/3,
      text_dz: MP.a/3
    };
    if(typeof opts_in == 'object') {
      Object.assign(opts, opts_in);
    }

    var B = reshape(MP.B, 3, 6);
    var Pb = reshape(MP.Pb, 3, 6);
    var A = reshape(MP.A, 3, 6);

    // A
    this.traces.push({
      x: plus(A[0], opts.text_dx),
      y: plus(A[1], opts.text_dy),
      z: plus(A[2], opts.text_dz),
      type: 'scatter3d',
      mode: 'text',
      text: range(1, 6).map((e) => 'A'+e),
      textposition: 'center middle',
      textfont: {
        size: 18,
        color: '#f00'
      },
      'showLegend': false
    });

    // P
    this.traces.push({
      x: plus(Pb[0], opts.text_dx),
      y: plus(Pb[1], opts.text_dy),
      z: plus(Pb[2], opts.text_dz),
      type: 'scatter3d',
      mode: 'text',
      text: range(1, 6).map((e) => 'P'+e),
      textposition: 'center middle',
      textfont: {
        size: 18,
        color: '#f00'
```

```
275       },
276       'showLegend': false
277     });
278
279     // B
280     this.traces.push({
281       x: plus(B[0], opts.text_dx),
282       y: plus(B[1], opts.text_dy),
283       z: plus(B[2], opts.text_dz),
284       type: 'scatter3d',
285       mode: 'text',
286       text: range(1, 6).map((e) => 'B'+e),
287       textposition: 'center middle',
288       textfont: {
289         size: 18,
290         color: '#f00'
291       },
292       'showLegend': false
293     });
294   }
295
296   /**
297    * Plots the joint vectors of the mechanism.
298    */
299   plotJointVectors() {
300     var MP = this.MP;
301
302     this.updateCallbacks.push('updateJointVectors');
303
304     var i_m = reshape(scale(MP.i_m, MP.a/2), 3, 6);
305     var i_ll = reshape(scale(MP.i_ll, MP.a/2), 3, 6);
306     var i_jpb = reshape(scale(MP.i_jpb, MP.a/2), 3, 6);
307     var B = reshape(MP.B, 3, 6);
308     var Pb = reshape(MP.Pb, 3, 6);
309     var A = reshape(MP.A, 3, 6);
310
311     var i_m_vectors = createVectors3D(B[0], B[1], B[2],
312       i_m[0], i_m[1], i_m[2], this.head_scale, this.head_angleFactor, {width: 6,
              color: '#f00'});
313     this.traces.push(i_m_vectors.line);
314     this.traces.push(i_m_vectors.head);
315
316     var i_ll_vectors = createVectors3D(A[0], A[1], A[2],
317       i_ll[0], i_ll[1], i_ll[2], this.head_scale, this.head_angleFactor, {width:
              6, color: '#f00'});
318     this.traces.push(i_ll_vectors.line);
319     this.traces.push(i_ll_vectors.head);
320
321     var i_jpb_vectors = createVectors3D(Pb[0], Pb[1], Pb[2],
322       i_jpb[0], i_jpb[1], i_jpb[2], this.head_scale, this.head_angleFactor, {
            width: 6, color: '#f00'});
323     this.traces.push(i_jpb_vectors.line);
324     this.traces.push(i_jpb_vectors.head);
325   }
326
327   /**
328    * Plots the current iteration of the mechanism.
329    */
330   plotIteration() {
```

```
331       var MP = this.MP;
332       var B = reshape(MP.B, 3, 6);
333       var Pb = reshape(MP.Pb, 3, 6);
334       var A = reshape(MP.A, 3, 6);
335
336       for(var i = 0; i < MP.m; i++) {
337         // Lower leg
338         this.traces.push({
339           x: [B[0][i], A[0][i]],
340           y: [B[1][i], A[1][i]],
341           z: [B[2][i], A[2][i]],
342           type: 'scatter3d',
343           mode: 'lines',
344           width: 1,
345           color: '#f00',
346           name: 'BA',
347           'showLegend': false
348         });
349
350         // Upper leg
351         this.traces.push({
352           x: [A[0][i], Pb[0][i]],
353           y: [A[1][i], Pb[1][i]],
354           z: [A[2][i], Pb[2][i]],
355           type: 'scatter3d',
356           mode: 'lines',
357           width: 1,
358           color: '#f00',
359           name: 'AP',
360           'showLegend': false
361         });
362       }
363
364       // Platform
365       this.traces.push({
366         x: [...Pb[0], Pb[0][0]],
367         y: [...Pb[1], Pb[1][0]],
368         z: [...Pb[2], Pb[2][0]],
369         type: 'scatter3d',
370         mode: 'lines',
371         width: 1,
372         color: '#f00',
373         name: 'p',
374         'showLegend': false
375       });
376     }
377
378     /**
379      * Plots the loads applied to the mechanism using FEM.
380      * @param {boolean} [deformed=false] - Flag to indicate if deformation is
          shown.
381      * @param {Object} opts_in - Optional parameters for plotting.
382      */
383     plotLoadsFEM(deformed = false, opts_in) {
384       var opts = {
385         displacements_scale: 100,
386         forces_scale: 1/8,
387         torques_scale: 2
388       };
```

```
389     if(typeof opts_in == 'object') {
390       Object.assign(opts, opts_in);
391     }
392
393     var F = getSub(solver_fem.FEM.F, range(0, solver_fem.FEM.dof-1));
394     var nc = nc = scale(solver_fem.FEM.nc, 1000);
395     if(deformed) {
396       nc = plus(nc, scale(solver_fem.FEM.DT, 1000*opts.displacements_scale));
397     }
398
399     // Forces
400     var c = [];
401     var f = [];
402     for(var i = 0; i < solver_fem.FEM.dof ; i += 6) {
403       if(norm([F[i], F[i+1], F[i+2]]) > 0.001) {
404         c.push(nc[i / 2], nc[i / 2 + 1], nc[i / 2 + 2]);
405         f.push(F[i], F[i + 1], F[i + 2]);
406       }
407     }
408     if(c.length > 0) {
409       c = reshape(c, 3, c.length / 3);
410       f = reshape(scale(f, opts.forces_scale), 3, f.length / 3);
411       var F_vectors = createVectors3D(c[0], c[1], c[2],
412         f[0], f[1], f[2], this.head_scale, this.head_angleFactor, {width: 6,
              color: '#00f'});
413       F_vectors.line.name = "Sile";
414       F_vectors.line.showLegend = true;
415       this.traces.push(F_vectors.line);
416       this.traces.push(F_vectors.head);
417     }
418
419     // Torques
420     var c = [];
421     var T = [];
422     for(var i = 0; i < solver_fem.FEM.dof ; i += 6) {
423       if(norm([F[i + 3], F[i + 4], F[i + 5]]) > 0.001) {
424         c.push(nc[i / 2], nc[i / 2 + 1], nc[i / 2 + 2]);
425         T.push(F[i + 3], F[i + 4], F[i + 5]);
426       }
427     }
428     if(c.length > 0) {
429       c = reshape(c, 3, c.length / 3);
430       T = reshape(scale(T, opts.torques_scale), 3, T.length / 3);
431       var T_vectors = createVectors3D(c[0], c[1], c[2],
432         T[0], T[1], T[2], this.head_scale, this.head_angleFactor, {width: 6,
              color: '#f00'});
433       T_vectors.line.name = "Momenti";
434       T_vectors.line.showLegend = true;
435       this.traces.push(T_vectors.line);
436       this.traces.push(T_vectors.head);
437     }
438   }
439
440   /**
441    * Plots the FEM model of the mechanism.
442    * @param {Object} solver_fem - FEM solver instance.
443    * @param {Object} opts_in - Optional parameters for plotting.
444    */
445   plotModelFEM(solver_fem, opts_in) {
```

```
446     this.updateCallbacks.push('updateModelFEM');
447
448     var opts = {
449       text_dx: MP.a/3,
450       text_dy: MP.a/3,
451       text_dz: MP.a/3
452     };
453     if(typeof opts_in == 'object') {
454       Object.assign(opts, opts_in);
455     }
456
457     this.plotBase();
458
459     // Elements
460     var nn = solver_fem.FEM.nn;
461     var en = solver_fem.FEM.en;
462     var nc = scale(solver_fem.FEM.nc, 1000);
463     var enc = [];
464     for(var i = 0; i < solver_fem.FEM.ne; i++) {
465       var n1 = en[2 * i] - 1; // Node 1
466       var n2 = en[2 * i + 1] - 1; // Node 2
467       enc.push(nc[3 * n1], nc[3 * n1 + 1], nc[3 * n1 + 2],
468               nc[3 * n2], nc[3 * n2 + 1], nc[3 * n2 + 2],
469               null, null, null);
470     }
471     enc = reshape(enc, 3, enc.length / 3);
472
473     this.traces.push({
474       x: enc[0],
475       y: enc[1],
476       z: enc[2],
477       type: 'scatter3d',
478       mode: 'lines+markers',
479       width: this.line_width,
480       color: this.c2,
481       name: 'Elementi',
482       marker: {
483         color: this.c2,
484         size: this.marker_color,
485         symbol: 'circle',
486         line: {
487           color: this.c2,
488           width: this.line_width/2
489         },
490       },
491       'showLegend': true
492     });
493
494     this.plotLoadsFEM(false, opts);
495
496     // Node text
497     nc = reshape(nc, 3, nn);
498     var text_dx = createFilledArray(nn, opts.text_dx);
499     var text_dy = createFilledArray(nn, opts.text_dy);
500     var text_dz = createFilledArray(nn, opts.text_dz);
501     for(var i = 0; i < solver_fem.FEM.nec ; i++) {
502       var n2 = solver_fem.FEM.ec[2 * i + 1] - 1; // Node 2
503       text_dx[n2] = -opts.text_dx;
504     }
```

```
505
506       this.traces.push({
507         x: plus(nc[0], text_dx),
508         y: plus(nc[1], text_dy),
509         z: plus(nc[2], text_dz),
510         type: 'scatter3d',
511         mode: 'text',
512         text: range(1, nn),
513         textposition: 'center middle',
514         textfont: {
515           size: 18,
516           color: '#f00'
517         },
518         'showLegend': false
519       });
520     }
521
522     /**
523      * Plots the FEM analysis results of the mechanism.
524      * @param {Object} solver_fem - FEM solver instance.
525      * @param {Object} opts_in - Optional parameters for plotting.
526      */
527     plotResultFEM(solver_fem, opts_in) {
528       this.updateCallbacks.push('updateResultFEM');
529
530       var opts = {
531         displacements_scale: 100,
532         forces_scale: 1/8,
533         torques_scale: 2
534       };
535       if(typeof opts_in == 'object') {
536         Object.assign(opts, opts_in);
537       }
538
539       this.plotBase();
540
541       // Elements and deformed elements
542       var en = solver_fem.FEM.en;
543       var nc = scale(solver_fem.FEM.nc, 1000);
544       var dnc = plus(nc, scale(solver_fem.FEM.DT, 1000*opts.displacements_scale));
545       var enc = [];
546       var denc = [];
547       for(var i = 0; i < solver_fem.FEM.ne; i++) {
548         var n1 = en[2 * i] - 1; // Node 1
549         var n2 = en[2 * i + 1] - 1; // Node 2
550         enc.push(nc[3 * n1], nc[3 * n1 + 1], nc[3 * n1 + 2],
551                  nc[3 * n2], nc[3 * n2 + 1], nc[3 * n2 + 2],
552                  null, null, null);
553         denc.push(dnc[3 * n1], dnc[3 * n1 + 1], dnc[3 * n1 + 2],
554                   dnc[3 * n2], dnc[3 * n2 + 1], dnc[3 * n2 + 2],
555                   null, null, null);
556       }
557       enc = reshape(enc, 3, enc.length / 3);
558       denc = reshape(denc, 3, denc.length / 3);
559
560       this.traces.push({
561         x: enc[0],
562         y: enc[1],
563         z: enc[2],
```

```
564       type: 'scatter3d',
565       mode: 'lines',
566       width: this.line_width,
567       color: this.c2,
568       name: 'Nedeformisan  ',
569       line: {
570         dash: 'dash'
571       }
572     });
573
574     this.traces.push({
575       x: denc[0],
576       y: denc[1],
577       z: denc[2],
578       type: 'scatter3d',
579       mode: 'lines+markers',
580       width: this.line_width,
581       color: '#0f0',
582       name: 'Deformisan  ',
583       marker: {
584         color: '#0f0',
585         size: this.marker_color,
586         symbol: 'circle',
587         line: {
588           color: '#0f0',
589           width: this.line_width/2
590         },
591       }
592     });
593
594     // Load forces and torques
595     this.plotLoadsFEM(true, opts);
596
597     // Reaction forces
598     var R = solver_fem.FEM.R;
599     var c = [];
600     var f = [];
601     for(var i = 0; i < solver_fem.FEM.rdof.length / 6; i++) {
602       var n = 8 + i * 4; // Node
603       if(norm([R[i * 6], R[i * 6 + 1], R[i * 6 + 2]]) > 0.001) {
604         c.push(dnc[n * 3], dnc[n * 3 + 1], dnc[n * 3 + 2]);
605         f.push(R[i * 6], R[i * 6 + 1], R[i * 6 + 2]);
606       }
607     }
608     if(c.length > 0) {
609       c = reshape(c, 3, c.length / 3);
610       f = reshape(scale(f, opts.forces_scale), 3, f.length / 3);
611       var F_vectors = createVectors3D(c[0], c[1], c[2],
612         f[0], f[1], f[2], this.head_scale, this.head_angleFactor, {width: 6,
              color: '#00fffd'});
613       F_vectors.line.name = "Reakcije sile";
614       F_vectors.line.showLegend = true;
615       this.traces.push(F_vectors.line);
616       this.traces.push(F_vectors.head);
617     }
618
619     // Reaction torques
620     var c = [];
621     var T = [];
```

```
622     for(var i = 0; i < solver_fem.FEM.rdof.length / 6; i++) {
623       var n = 8 + i * 4; // Node
624       if(norm([R[i * 6 + 3], R[i * 6 + 4], R[i * 6 + 5]]) > 0.001) {
625         c.push(dnc[n * 3], dnc[n * 3 + 1], dnc[n * 3 + 2]);
626         T.push(R[i * 6 + 3], R[i * 6 + 4], R[i * 6 + 5]);
627       }
628     }
629     if(c.length > 0) {
630       c = reshape(c, 3, c.length / 3);
631       T = reshape(scale(T, opts.torques_scale), 3, T.length / 3);
632       var T_vectors = createVectors3D(c[0], c[1], c[2],
633         T[0], T[1], T[2], this.head_scale, this.head_angleFactor, {width: 6,
              color: '#ff8a00'});
634       T_vectors.line.name = "Reakcije momenti";
635       T_vectors.line.showLegend = true
636       this.traces.push(T_vectors.line);
637       this.traces.push(T_vectors.head);
638     }
639
640     // Joints reactions
641     var Rz = solver_fem.FEM.Rz;
642     var c = [];
643     var f = [];
644     for(var i = 0; i < Rz.length / 3 / 2; i++) {
645       var n = i + 2; // Node
646       if(norm([Rz[i * 2 * 3], Rz[i * 2 * 3 + 1], Rz[i * 2 * 3 + 2]]) > 0.001) {
647         c.push(dnc[n * 3], dnc[n * 3 + 1], dnc[n * 3 + 2]);
648         f.push(Rz[i * 2 * 3], Rz[i * 2 * 3 + 1], Rz[i * 2 * 3 + 2]);
649       }
650     }
651     if(c.length > 0) {
652       c = reshape(c, 3, c.length / 3);
653       f = reshape(scale(f, opts.forces_scale), 3, f.length / 3);
654       var F_vectors = createVectors3D(c[0], c[1], c[2],
655         f[0], f[1], f[2], this.head_scale, this.head_angleFactor, {width: 6,
              color: '#7100ff'});
656       F_vectors.line.name = "Reakcije zglobovi";
657       F_vectors.line.showLegend = true;
658       this.traces.push(F_vectors.line);
659       this.traces.push(F_vectors.head);
660     }
661   }
662
663   /**
664    * Displays the plotted mechanism.
665    * @param {number} [fid=1] - Figure identifier.
666    */
667   show(fid = 1) {
668     this.fid = fid;
669     var D = max([this.MP.R_b, this.MP.R_p]);
670     if(!this.traces.length) {
671       this.plotMechanism();
672     }
673
674     figure(fid);
675     plot(this.traces, {'showLegend': false,
676      'font': {'family': 'LatinModern', 'size': 14},
677      'legend': {'font': {'family': 'LatinModern', 'size': 14}},
678      'legendLocation': 'north', 'legendOrientation': 'h'});
```

```
679       xlim([-D * this.s, D * this.s]);
680       ylim([-D * this.s, D * this.s]);
681       zlim([-this.MP.a * this.s, [this.MP.s + this.MP.cg[2]] * this.s]);
682       xlabel('x [mm]');
683       ylabel('y [mm]');
684       zlabel('z [mm]');
685     }
686
687     /**
688      * Displays the legend on the plot.
689      */
690     showLegend() {
691       figure(this.fid);
692       legend('on');
693     }
694
695     /**
696      * Displays the workspace as a mesh.
697      * @param {Array} vertices - Vertex coordinates.
698      * @param {Array} faces - Face indices.
699      * @param {number} [fid=1] - Figure identifier.
700      */
701     showWorkspace(vertices, faces, fid = 1) {
702       figure(fid);
703       plot([
704         {x: column(vertices, 0), y: column(vertices, 1), z: column(vertices, 2),
705          i: column(faces, 0), j: column(faces, 1), k: column(faces, 2),
706          intensity: column(vertices, 2), colorscale: 'Jet',
707          lighting: { ambient: 1 }, type: 'mesh3d', colorbar: { title: 'z [mm]' }}
708       ], {'showLegend': false, 'font': {'family': 'LatinModern', 'size': 14}});
709       xlabel('x [mm]');
710       ylabel('y [mm]');
711       zlabel('z [mm]');
712     }
713
714     /**
715      * Resets the plotter by clearing all traces and callbacks.
716      */
717     reset() {
718       this.traces = [];
719       this.updateCallbacks = [];
720     }
721 }
722
723 exports.MP_6RSS_PLOTTER = MP_6RSS_PLOTTER;
```

Listing C.1 - PLOTTER_MP_6RSS.jsl

D

Skripta solveInverseKinematics.jsl

Skriptom `solveInverseKinematics.jsl`, koja je u celosti data u listingu D.1, se prvo definišu geometrijski parametri mehanizma simulatora leta, zatim se rešava inverzni kinematički problem a zatim i prikazuje grafik sa uprošćenim kinematičkim modelom ukoliko konfiguracija može da ostvari zadat položaj i orijentaciju.

```
/**
 * @file Solve Inverse Kinematics
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x = [ // translaWcija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  0 // [deg] psi
];
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

var { MP_6RSS_PLOTTER } = require('./PLOTTER_MP_6RSS.jsl');
var plotter = new MP_6RSS_PLOTTER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
solver.getGeometry();

// Odredjivanje potrebnog ugla alpha za date podatke
disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
solver.setParameters(x);

```

```
39 // Provera uglova u zglobovim
40 disp(' Provera uglova u zglobovima...');
41 solver.checkJoints();
42
43 if(any(MP.e)) {
44   disp(' Neuspešan proračun!');
45   dispRowVector('x', MP.x, MP.m);
46   dispRowVector('e', MP.e, MP.m);
47 } else {
48   disp(' Uspešan proračun!');
49   dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);
50
51   disp(' Prikaz mehanizma...');
52   plotter.plotMechanism();
53   plotter.plotPointNames();
54   plotter.plotJointVectors();
55   plotter.show();
56 }
57
58 // - Kraj proračuna
59 disp(' Program je uspešno izvršen... ');
60 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
61 disp(' -------------------- ');
```

Listing D.1 - `solveInverseKinematics.jsl`

E

Skripta solveForwardKinematics.jsl

Skriptom `solveForwardKinematics.jsl`, koja je u celosti data u listingu E.1, se prvo definišu geometrijski parametri mehanizma simulatora leta, zatim se rešava direktni kinematički problem Njutnovom metodom za sistem nelinearnih jednačina. Definiše se početna pretpostavka za položaj i orijentaciju platforme i željene vrednosti, sada uglova otklona donjih poluga. Pri rešavanju direktnog kinematičkog problema, kao što je već pokazano, je neophodno rešavati i inverzni kinematički problem. Na kraju izvršavanja programa se u komandnom prozoru programskog paketa `JSLAB` ispisuju rezultati i ukoliko postoji rešenje formira se grafik sa uprošćenim kinematičkim modelom mehanizma.

```
/**
 * @file Solve Forward Kinematics
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var alpha = [ // [deg] uglovi donjih poluga
  34.18945660,
  -20.81030658,
  34.18945660,
  -20.81030658,
  34.18945660,
  -20.81030658
];
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

var { MP_6RSS_PLOTTER } = require('./PLOTTER_MP_6RSS.jsl');
var plotter = new MP_6RSS_PLOTTER(MP);

solver.setIterationCallback(function() {
  plotter.plotIteration();
});
```

```
34
35 // Odredjivanje geometrijskih karakteristika
36 disp(' Odredjivanje geometrijskih karakteristika...');
37 solver.getGeometry();
38
39 // Odredjivanje spoljašnjih parametara na osnovu uglova alpha
40 disp(' Odredjivanje spoljašnjih parametara na osnovu uglova alpha...');
41 solver.setAlpha(alpha);
42
43 // Provera uglova u zglobovim
44 disp(' Provera uglova u zglobovima...');
45 solver.checkJoints();
46
47 if(any(MP.e)) {
48   disp(' Neuspešan proračun!');
49   dispRowVector('alpha', multiplyEl(MP.alpha_w_in, r2d), MP.m);
50   dispRowVector('e', MP.e, MP.m);
51 } else {
52   disp(' Uspešan proračun!');
53   dispRowVector('x', MP.x, MP.m);
54
55   disp(' Prikaz mehanizma...');
56   plotter.plotMechanism();
57   plotter.show();
58 }
59
60 // - Kraj proračuna
61 disp(' Program je uspešno izvršen... ');
62 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
63 disp(' -------------------- ');
```

Listing E.1 - `solveForwardKinematics.jsl`

F

Skripta calculateVelocitiesAndAccelerations.jsl

Skriptom `calculateVelocitiesAndAccelerations.jsl`, koja je u celosti data u listingu F.1, se na osnovu definisanih uglova donjih poluga, ugaonih brzina i ugaonih ubrzanja aktuatora određuju brzine i ubrzanja svih značajnih tačaka mehanizma. Na kraju izvršavanja programa se u komandnom prozoru programskog paketa `JSLAB` ispisuju rezultati koji su dati u nastavku.

```
/**
 * @file Calculate velocities and accelerations
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x = [ // translacija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  0 // [deg] psi
];
var alpha = [30, 30, 30, 30, 30, 30]; // deg
var omega = [10, -5, -5, 5, -5, 10]; // rad/s
var epsilon = [0, 0, 0, 0, 0, 0]; // rad/s/s
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

var { MP_6RSS_PLOTTER } = require('./classes/PLOTTER_MP_6RSS.jsl');
var plotter = new MP_6RSS_PLOTTER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
```

```
36 solver.getGeometry();
37
38 // Odredjivanje potrebnog ugla alpha za date podatke
39 disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
40 solver.setParameters(x);
41 solver.getJacobian();
42 solver.setAlpha(alpha);
43
44 // Provera uglova u zglobovim
45 disp(' Provera uglova u zglobovima...');
46 solver.checkJoints();
47
48 if(any(MP.e)) {
49   disp(' Neuspešan proračun!');
50   dispRowVector('x', MP.x, MP.m);
51   dispRowVector('e', MP.e, MP.m);
52 } else {
53   disp(' Uspešan proračun!');
54   dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);
55   dispRowVector('x', MP.x, MP.m);
56   disp('Potreban broj iteracija: ' + MP.n_iter);
57
58   disp(' Prikaz mehanizma...');
59   plotter.plotMechanism();
60   plotter.show();
61 }
62
63 dispMatrix('B', transpose(MP.B, MP.m, 3), MP.m, 3);
64 dispMatrix('A', transpose(MP.A, MP.m, 3), MP.m, 3);
65 dispMatrix('P', transpose(MP.P, MP.m, 3), MP.m, 3);
66 dispMatrix('Pb', transpose(MP.Pb, MP.m, 3), MP.m, 3);
67 dispRowVector('CG', getSub(MP.POb, range(0, 2)), 3);
68
69 // Promena jedinica iz mm u m
70 var BA = divideEl(minus(MP.A, MP.B), 1000);
71 var P = divideEl(MP.P, 1000);
72 var cg = divideEl(multiply(MP.R, MP.cg, 3, 3, 1), 1000);
73 var s = MP.s/1000;
74 var AP = divideEl(minus(MP.Pb, MP.A), 1000);
75 var OpP = divideEl(minus(MP.Pb,
76   repCol(getSub(MP.POb, range(3, 5)), MP.m)), 1000);
77 var J_a = divideEl(MP.J_a, 1000);
78 var J_x = cloneArray(MP.J_x);
79 setSub(J_x, range(18, 35), divideEl(getSub(J_x, range(18, 35)), 1000));
80 var invJ_x = inv(J_x, MP.m);
81 var J = multiply(invJ_x, J_a, MP.m, MP.m, MP.m);
82
83 // Brzina kraja poluge
84 disp(' Odredjivanje brzine kraja donje poluge...');
85 var v_A = multiplyEl(cross3D(MP.i_m, BA, MP.m), repRow(omega, 3));
86 dispMatrix('v_A', transpose(v_A, MP.m, 3), MP.m, 3);
87
88 // Brzina tacaka na platformi
89 disp(' Odredjivanje brzina tacaka na platformi...');
90
91 // dot_x = J*omega = J*dot_alpha
92 var dot_x = multiply(J, omega, 6, 6, 1);
93 var v_p = getSub(dot_x, range(0, 2));
94 var omega_p = getSub(dot_x, range(3, 5));
```

```
95  dispRowVector('v_p', v_p, 3);
96  dispRowVector('omega_p', omega_p, 3);
97
98  // v_P = v_p + (omega_p x CG_P)
99  var v_P = plus(repCol(v_p, MP.m), cross3D(repCol(omega_p, MP.m), OpP, MP.m));
100 dispMatrix('v_P', transpose(v_P, MP.m, 3), MP.m, 3);
101
102 // v_CG = v_p + (omega_p x CG_P)
103 var v_CG = plus(v_p, cross3D(omega_p, cg, 1));
104 dispRowVector('v_CG', v_CG, 3);
105
106 // Ugaona brzina gornje poluge
107 var omega_ul = scale(cross3D(minus(v_P, v_A), AP, MP.m), -1/pow(s, 2));
108 dispMatrix('omega_ul', transpose(omega_ul, MP.m, 3), MP.m, 3);
109
110 // Ubrzanje kraja poluge
111 disp(' Odredjivanje ubrzanja kraja donje poluge...');
112
113 // A = r*i_t*epsilon + r*i_r*omega^2
114 var a_A = plus(multiplyEl(repRow(epsilon, 3), cross3D(MP.i_m, BA, MP.m)),
      cross3D(multiplyEl(repRow(omega, 3), MP.i_m), cross3D(multiplyEl(repRow(omega
      , 3), MP.i_m), BA, MP.m), MP.m));
115 dispMatrix('a_A', transpose(a_A, MP.m, 3), MP.m, 3);
116 // Prvi izvod Jakobijan matrice
117 var dot_J_x = transpose(concatRow(6, cross3D(omega_ul, MP.i_ul, MP.m),
118   plus(cross3D(cross3D(repCol(omega_p, MP.m), OpP, MP.m), MP.i_ul, MP.m),
119   cross3D(OpP, cross3D(omega_ul, MP.i_ul, MP.m), MP.m))), MP.m, MP.m);
120
121 var dot_J_ai = plus(dot(cross3D(omega_ul, MP.i_ul, MP.m),
122   cross3D(MP.i_m, BA, MP.m), 6), dot(MP.i_ul, cross3D(MP.i_m,
123   cross3D(multiplyEl(repRow(omega, 3), MP.i_m), BA, MP.m), MP.m), 6));
124 var dot_J_a = diag(dot_J_ai, MP.m);
125
126 var dot2_x = multiply(invJ_x, plus(
127     multiply(dot_J_a, omega, MP.m, MP.m, 1),
128     multiply(J_a, epsilon, MP.m, MP.m, 1),
129     scale(multiply(dot_J_x, dot_x, MP.m, MP.m, 1), -1)
130   ), MP.m, MP.m, 1);
131
132 var a_p = getSub(dot2_x, range(0, 2));
133 var epsilon_p = getSub(dot2_x, range(3, 5));
134 dispRowVector('a_p', a_p, 3);
135 dispRowVector('epsilon_p', epsilon_p, 3);
136
137 // Ubrzanja tacaka na platformi
138 disp(' Odredjivanje ubrzanja tacaka na platformi...');
139
140 // a_P = a_p + epsilon_p x MP.P + omega_p x (omega_p x P)
141 var a_P = plus(repCol(a_p, MP.m), cross3D(repCol(epsilon_p, MP.m), P, MP.m),
142   cross3D(repCol(omega_p, MP.m), cross3D(repCol(omega_p, MP.m), P, MP.m), MP.m))
      ;
143 dispMatrix('a_P', transpose(a_P, MP.m, 3), MP.m, 3);
144
145 // a_CG = a_p + epsilon_p x MP.cg + omega_p x (omega_p x CG)
146 var a_CG = plus(a_p, cross3D(epsilon_p, cg, 1),
147   cross3D(omega_p, cross3D(omega_p, cg, 1)));
148 dispRowVector('a_CG', a_CG, 3);
149
150 // Ugaono ubrzanje gornje poluge
```

```
151 var epsilon_ul = scale(cross3D(minus(a_P, a_A), AP, MP.m), -1/pow(s, 2));
152 dispMatrix('epsilon_ul', transpose(epsilon_ul, MP.m, 3), MP.m, 3);
153
154 // Statika
155 var W = scale(MP.g, MP.M);
156 var F_x = concat(W, cross3D(multiply(MP.R, cg, 3, 3, 1), W, 1));
157 var M_m = scale(multiply(transpose(J, 6, 6), F_x, 6, 6, 1), -1); // [Nm]
158 dispRowVector('Statika M_m', M_m, MP.m);
159
160 // Dinamika
161 var F = scale(minus(MP.g, a_CG), MP.M);
162 var T = minus(cross3D(cg, F, 1),
163   plus(multiply(MP.I, epsilon_p, 3, 3, 1),
164   cross3D(omega_p, multiply(MP.I, omega_p, 3, 3, 1), 1)));
165 var F_x = concat(F, T);
166 var M_m = scale(multiply(transpose(J, 6, 6), F_x, 6, 6, 1), -1);
167 dispRowVector('Dinamika M_m', M_m, MP.m);
168
169 // - Kraj proračuna
170 disp(' Program je uspešno izvršen... ');
171 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
172 disp(' -------------------- ');
```

Listing F.1 - calculateVelocitiesAndAccelerations.jsl

```
 --- calculateVelocitiesAndAccelerations.jsl ---
 Određivanje geometrijskih karakteristika...
 Određivanje potrebnih uglova otklona donjih poluga...
 Provera uglova u zglobovima...
 Uspešan proračun!
 alpha = [ 30.00000002 30.00000002 30.00000002 30.00000002
    30.00000002 30.00000002 ]
 x = [ 0.00000000 -0.00000000 129.24715436 0.00000000 -0.00000000
     -0.00000000 ]
Potreban broj iteracija: 4
 Prikaz mehanizma...
 B = [
  697.33628866 61.00901992 0.00000000
  401.50350545 573.40643100 0.00000000
  -401.50350545 573.40643100 0.00000000
  -697.33628866 61.00901992 0.00000000
  -295.83278322 -634.41545093 0.00000000
  295.83278322 -634.41545093 0.00000000
 ]
 A = [
  578.80517895 258.52754041 97.14324523
  513.29400705 371.99621861 97.14324523
  -513.29400705 371.99621861 97.14324523
  -578.80517895 258.52754041 97.14324523
  -65.51117190 -630.52375902 97.14324523
  65.51117190 -630.52375902 97.14324523
 ]
 P = [
  475.44490588 -118.54172884 0.00000000
```

```
 135.06230435 471.01823101 0.00000000
 -135.06230435 471.01823101 0.00000000
 -475.44490588 -118.54172884 0.00000000
 -340.38260152 -352.47650217 0.00000000
 340.38260152 -352.47650217 0.00000000
]
Pb = [
 475.44490588 -118.54172884 1017.54284330
 135.06230435 471.01823101 1017.54284330
 -135.06230435 471.01823101 1017.54284330
 -475.44490588 -118.54172884 1017.54284330
 -340.38260152 -352.47650217 1017.54284330
 340.38260152 -352.47650217 1017.54284330
]
CG = [ -0.00000000 -0.00000000 1317.54284330 ]
Određivanje brzine kraja donje poluge...
v_A = [
 -1.09897407 -1.48701479 1.68257036
 0.91863981 0.10411603 -0.84128518
 -0.91863981 0.10411603 -0.84128518
 0.54948703 -0.74350739 0.84128518
 0.36915277 -0.84762343 -0.84128518
 0.73830555 1.69524685 1.68257036
]
Određivanje brzina tacaka na platformi...
v_p = [ 1.81617013 -2.62317403 0.40253076 ]
omega_p = [ -2.81922535 -2.54557703 1.93438467 ]
v_P = [
 2.04547543 -1.70348069 1.94700824
 0.90503968 -2.36191158 -0.58156428
 0.90503968 -2.88443648 -1.26918727
 2.04547543 -3.54286737 -0.47355502
 2.49799527 -3.28160492 0.52977133
 2.49799527 -1.96474314 2.26271159
]
v_CG = [ 1.05249702 -1.77740643 0.40253076 ]
omega_ul = [
 0.09952373 2.92148242 1.20804925
 2.29544891 0.08571713 0.93407654
 2.70829081 1.84036000 -1.31094972
 3.07231563 1.51280936 0.27474864
 2.62145409 2.33625004 0.07711316
 3.52996000 1.46015346 -1.49530358
]
Određivanje ubrzanja kraja donje poluge...
a_A = [
 11.85311097 -19.75185205 -9.71432452
 -2.79476254 5.03525531 -2.42858113
 2.79476254 5.03525531 -2.42858113
```

```
 -2.96327774 -4.93796301 -2.42858113
 -5.75804028 -0.09729230 -2.42858113
 23.03216113 -0.38916919 -9.71432452
]
a_p = [ -11.52819001 -3.36946375 -14.12587991 ]
epsilon_p = [ 12.72971106 -5.19552984 0.00000000 ]
Određivanje ubrzanja tacaka na platformi...
a_P = [
 -17.23881710 1.42833097 -15.17380214
 -9.52848237 -7.90632621 -10.48414416
 -6.76732090 -9.84489040 -10.41446917
 -7.51900548 -5.39578234 -14.92853297
 -10.57843205 -1.69183183 -16.78936843
 -17.53708220 3.19371729 -16.96496261
]
a_CG = [ -14.72288886 -8.66561463 -18.45427811 ]
epsilon_ul = [
 -17.43563070 -27.34049200 -13.15886156
 11.11374837 -9.24458235 5.56170290
 12.90490143 -5.78042173 -4.68128610
 5.13472443 -2.90109154 -1.76514526
 -2.52536397 -8.38405678 1.77859007
 -5.31370731 -35.34692200 12.26499995
]
Statika M_m = [ 59.23240202 59.23240202 59.23240202 59.23240202
   59.23240202 59.23240202 ]
Dinamika M_m = [ 392.46963264 506.29741863 -379.67334318
   -118.92521746 -169.37774398 -543.95365569 ]
Program je uspešno izvršen...
Potrebno vreme: 0.15 sekundi
--------------------
```

G

Određivanje matrice krutosti elementa Ojler-Bernulijeva greda

Skriptom `tensionShape.jsl`, koja je u celosti data u listingu G.1, se izvodi matrica krutosti elementa za slučaj zatezanja i uvijanja grede i prikazuje grafik funkcija oblika.

Skriptom `bendingShapeXY.jsl`, koja je u celosti data u listingu G.2, se izvodi matrica krutosti elementa za slučaj savijanja grede u xy ravni i prikazuje grafik funkcija oblika.

Skriptom `bendingShapeXZ.jsl`, koja je u celosti data u listingu G.3, se izvodi matrica krutosti elementa za slučaj savijanja grede u xz ravni i prikazuje grafik funkcija oblika.

U prikazanim skriptama se koristi biblioteka `sym` za simboličku matematiku u programu JSLAB.

```
1  /**
2   * @file Tension shape function
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8  close('all'); clear; clc; tic;
9  disp(' --- ' + jsl_file_name + ' ---');
10
11 var le, x, A, E, G, J;
12 var p, P, invP, N, B;
13 var k_int, k_e_tension, k_e_torsion;
14 var xi = range(0, 1, 0.01);
15
16 sym.load().then(function() {
17   [le, x, A, E, G, J] = sym.syms(['le', 'x', 'A', 'E', 'G', 'J']);
18
19   p = sym.mat([[1, x]]);
20   P = sym.mat([[1, 0], [1, le]]);
21
22   invP = sym.inv(P);
23   N = sym.mul(p, invP);
24   B = sym.diff(N, 'x');
25
26   k_int = sym.intg(sym.mul(sym.transp(B), B), x, [0, le]);
27
28   k_e_tension = sym.mul(A, E, k_int);
29   k_e_torsion = sym.mul(G, J, k_int);
30
```

```
  Ni = sym.subs(sym.subs(N, le, 1), x, xi).toNumeric();
  var N_flat = Ni.flat();

  sym.showLatex(N);
  sym.showLatex(k_e_tension);
  sym.showLatex(k_e_torsion);

  // Prikaz grafika
  figure(1);
  plot([
    {x: xi, y: column(N_flat, 0), name: 'N<sub>1</sub>', width: 3},
    {x: xi, y: column(N_flat, 1), name: 'N<sub>2</sub>', width: 3}
  ], {font: {family: 'LatinModern', size: 18}, legendLocation: 'east'});
  xlabel("x [L]");
  ylabel("N [-]");
  printFigure(pwd+'zatezanjeN', {type: 'pdf', size: [560, 420]});

  // - Kraj proračuna
  disp(' Program je uspešno izvršen... ');
  disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
  disp(' -------------------- ');
});
```

Listing G.1 - `tensionShape.jsl`

```
/**
 * @file Bending in XY plane shape function
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');

var le, x, E, Iz;
var p, P, invP, N, d2N;
var k_int, k_e_stretching, k_e_torsion;
var xi = range(0, 1, 0.01);

sym.load().then(function() {
  [le, x, E, Iz] = sym.syms(['le', 'x', 'E', 'Iz']);

  P = sym.mat([
    [1, 0, 0, 0],
    [0, 1, 0, 0],
    [1, le, sym.pow(le, 2), sym.pow(le, 3)],
    [0, 1, sym.mul(2, le), sym.mul(3, sym.pow(le, 2))]
  ]);
  p = sym.mat([[1, x, sym.pow(x, 2), sym.pow(x, 3)]]);

  invP = sym.inv(P);
  N = sym.mul(p, invP);
  d2N = sym.diff(N, 'x', 2);

  k_int = sym.mul(E, Iz, sym.intg(sym.mul(sym.transp(d2N), d2N), x, [0, le]));


  Ni = sym.subs(sym.subs(N, le, 1), x, xi).toNumeric();
  var N_flat = Ni.flat();
```

```
36
37    sym.showLatex(N);
38    sym.showLatex(k_int);
39
40    // Prikaz grafika
41    figure(1);
42    plot([
43      {x: xi, y: column(N_flat, 0), name: 'N<sub>1</sub>', width: 3},
44      {x: xi, y: column(N_flat, 1), name: 'N<sub>2</sub>', width: 3},
45      {x: xi, y: column(N_flat, 2), name: 'N<sub>3</sub>', width: 3},
46      {x: xi, y: column(N_flat, 3), name: 'N<sub>4</sub>', width: 3}
47    ], {font: {family: 'LatinModern', size: 18}, legendLocation: 'east'});
48    xlabel("x [L]");
49    ylabel("N [-]");
50    printFigure(pwd+'savijanjeXY', {type: 'pdf', size: [560, 420]});
51
52    // - Kraj proračuna
53    disp(' Program je uspešno izvršen... ');
54    disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
55    disp(' -------------------- ');
56  });
```

Listing G.2 - bendingShapeXY.jsl

```
1   /**
2    * @file Bending in XZ plane shape function
3    * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4    * Doktorska disertacija
5    * Mašinski fakultet Univerziteta u Beogradu
6    * @version 0.0.1
7    */
8   close('all'); clear; clc; tic;
9   disp(' --- ' + jsl_file_name + ' ---');
10
11  var le, x, E, Iy;
12  var p, P, invP, N, d2N;
13  var k_int, k_e_stretching, k_e_torsion;
14  var xi = range(0, 1, 0.01);
15
16  sym.load().then(function() {
17    [le, x, E, Iy] = sym.syms(['le', 'x', 'E', 'Iy']);
18
19    P = sym.mat([
20      [1, 0, 0, 0],
21      [0, -1, 0, 0],
22      [1, le, sym.pow(le, 2), sym.pow(le, 3)],
23      [0, -1, sym.mul(-2, le), sym.mul(-3, sym.pow(le, 2))]
24    ]);
25    p = sym.mat([[1, x, sym.pow(x, 2), sym.pow(x, 3)]]);
26
27    invP = sym.inv(P);
28    N = sym.mul(p, invP);
29    d2N = sym.diff(N, 'x', 2);
30
31    k_int = sym.mul(E, Iy, sym.intg(sym.mul(sym.transp(d2N), d2N), x, [0, le]));
32
33
34    Ni = sym.subs(sym.subs(N, le, 1), x, xi).toNumeric();
35    var N_flat = Ni.flat();
36
```

```
  sym.showLatex(N);
  sym.showLatex(k_int);

  // Prikaz grafika
  figure(1);
  plot([
    {x: xi, y: column(N_flat, 0), name: 'N<sub>1</sub>', width: 3},
    {x: xi, y: column(N_flat, 1), name: 'N<sub>2</sub>', width: 3},
    {x: xi, y: column(N_flat, 2), name: 'N<sub>3</sub>', width: 3},
    {x: xi, y: column(N_flat, 3), name: 'N<sub>4</sub>', width: 3}
  ], {font: {family: 'LatinModern', size: 18}, legendLocation: 'east'});
  xlabel("x [L]");
  ylabel("N [-]");
  printFigure(pwd+'savijanjeXZ', {type: 'pdf', size: [560, 420]});

  // - Kraj proračuna
  disp(' Program je uspešno izvršen... ');
  disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
  disp(' -------------------- ');
});
```

Listing G.3 - `bendingShapeXZ.jsl`

H

JSLAB klasa MP_6RSS_FEM_SOLVER

U listingu H.1 je dat kompletan kod JSLAB klase SOLVER_FEM_MP_6RSS.jsl koja implementira sve neophodne metode opisane u poglavlju 4.7.3 za metod konačnih elemenata primenjen na mehanizam simulatora leta.

```
1  /**
2   * @file Motion Platform 6RSS Mechanism FEM solver
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8
9  /**
10  * Class for Motion Platform 6RSS Mechanism FEM solver .
11  */
12 class MP_6RSS_FEM_SOLVER {
13
14   /**
15    * Constructor for the 6RSS FEM solver.
16    * @param {Object} MP - Mechanism parameters.
17    */
18   constructor(MP) {
19     this.MP = MP;
20   }
21
22   /**
23    * Constructs the FEM model with given parameters.
24    * @param {Object} FEM - FEM data structure.
25    * @param {Object} M - Material properties.
26    * @param {Object} PP - Section properties.
27    */
28   construct(FEM, M, PP) {
29     this.FEM = FEM;
30     this.M = M;
31     this.PP = PP;
32
33     var nn = FEM.nc.length / 3; // broj cvorova
34     var nec = FEM.ec.length / 2; // broj veza cvorova
35     FEM.nn = nn;
36     FEM.nec = nec;
37     FEM.ne = this.FEM.en.length / 2;
38
39     FEM.dof = 6 * nn; // broj stepeni slobode
```

```
40      FEM.D = zeros(FEM.dof); // pomeranja
41      FEM.DT = zeros(FEM.dof / 2); // translatorna pomeranja
42      FEM.DR = zeros(FEM.dof / 2); // ugaona pomeranja
43      FEM.F = zeros(FEM.dof); // opterecenja
44      FEM.R = zeros(FEM.dof); // reakcije
45      FEM.K = zeros(FEM.dof * FEM.dof); // matrica krutosti
46      FEM.C = zeros(6 * 6 * nn); // matrica konstanti
47
48      // Stepeni slobode cvorova
49      FEM.rdof = indexOfAll(FEM.bc, 1); // ograniceni stepeni slobode
50      FEM.fdof = indexOfAll(FEM.bc, 0); // slobodni stepeni slobode
51
52      // Povezivanje cvorova
53      var ecdof = 3;
54      FEM.ecdof = ecdof;
55      FEM.Np = nec * ecdof;
56      FEM.lmn = range(FEM.dof, FEM.Np + FEM.dof - 1);
57
58      FEM.Q = zeros(FEM.Np);
59      FEM.C = zeros(FEM.Np * FEM.dof);
60      var I3 = eye(3);
61      var A = concatCol(3, I3, scale(I3, -1));
62
63      for(var i = 0; i < nec; i++) {
64        var n1 = FEM.ec[i * 2] - 1; // Node 1
65        var n2 = FEM.ec[i * 2 + 1] - 1; // Node 2
66        setSub(FEM.C, index(range(i * ecdof, i * ecdof + 2),
67          concatCol(ecdof, range(6 * n1, 6 * n1 + 2),
68          range(6 * n2, 6 * n2+2)), FEM.Np), A);
69      }
70
71      // Karakteristike materijala i poprecnog preseka
72      M.G = M.E / (2 * (1 + M.ni)); // Modul klizanja
73      if(PP.type == 1) {
74        // Cev kruznog poprecnog preseka
75        PP.D = PP.D / 1000; // [m]
76        PP.t = PP.t / 1000; // [m]
77        PP.A = (PP.D ** 2 - (PP.D - 2 * PP.t) ** 2) * pi / 4;
78        PP.Iy = (PP.D ** 4 - (PP.D - 2 * PP.t) ** 4) * pi / 64;
79        PP.Iz = PP.Iy;
80        PP.J = (PP.D ** 4 - (PP.D - 2 * PP.t) ** 4) * pi / 32;
81      }
82
83      // Matrica krutosti
84      this.stiffnessBeam3D();
85    }
86
87    /**
88     * Calculates the stiffness matrix for 3D beam elements.
89     */
90    stiffnessBeam3D() {
91      var E = this.M.E;
92      var G = this.M.G;
93      var A = this.PP.A;
94      var Iz = this.PP.Iz;
95      var Iy = this.PP.Iy;
96      var J = this.PP.J;
97      var nc = this.FEM.nc;
98      var ne = this.FEM.ne;
```

```
99      var en = this.FEM.en;
100
101     for(var i = 0; i < ne; i++) {
102       var n1 = en[2 * i] - 1; // Node 1
103       var n2 = en[2 * i + 1] - 1; // Node 2
104       var edof = concatCol(1, range(6 * n1, 6 * n1 + 5), range(6 * n2, 6 * n2 +
            5));
105       var edof_ind = index(edof, edof, this.FEM.dof);
106
107       var dx = nc[3 * n2] - nc[3 * n1];
108       var dy = nc[3 * n2 + 1] - nc[3 * n1 + 1];
109       var dz = nc[3 * n2 + 2] - nc[3 * n1 + 2];
110
111       var L_e = sqrt(pow(dx, 2) + pow(dy, 2) + pow(dz, 2));
112
113       var ki = [E * A / L_e, 12 * E * Iz / (L_e ** 3), 6 * E * Iz / (L_e ** 2),
            4 * E * Iz / L_e,
114             2 * E * Iz / L_e, 12 * E * Iy / (L_e ** 3), 6 * E * Iy / (L_e ** 2)
                ,
115             4 * E * Iy / L_e, 2 * E * Iy / L_e, G * J / L_e];
116
117
118       var a = diag([ki[0], ki[1], ki[5]], 3);
119       var b = [0, 0, 0, 0, 0, -ki[2], 0, ki[6], 0];
120       var c = diag([ ki[9], ki[7], ki[3]], 3);
121       var d = diag([-ki[9], ki[8], ki[4]], 3);
122
123       var b_T = transpose(b, 3, 3);
124
125       var k_e = concatRow(12,
126         concatCol(3, a, b, scale(a, -1), b),
127         concatCol(3, b_T, c, b, d),
128         concatCol(3, scale(transpose(a, 3, 3), -1), b_T, a, scale(b, -1)),
129         concatCol(3, b_T, transpose(d, 3, 3), scale(b_T, -1), c));
130
131       var r;
132       if(nc[3 * n1] === nc[3 * n2] &&
133          nc[3 * n1 + 1] === nc[3 * n2 + 1]) {
134         if(nc[3 * n2 + 2] > nc[3 * n1 + 2]) {
135           r = [0, 0, -1, 0, 1, 0, 1, 0, 0];
136         } else {
137           r = [0, 0, 1, 0, 1, 0, -1, 0, 0];
138         }
139       } else {
140         var CXx = dx / L_e;
141         var CYx = dy / L_e;
142         var CZx = dz / L_e;
143         var D = sqrt(CXx * CXx + CYx * CYx);
144         var CXy = -CYx / D;
145         var CYy = CXx / D;
146         var CZy = 0;
147         var CXz = -CXx * CZx / D;
148         var CYz = -CYx * CZx / D;
149         var CZz = D;
150         r = [CXx, CXy, CXz, CYx, CYy, CYz, CZx, CZy, CZz];
151       }
152
153       var T = concatRow(12,
154         concatCol(3, r, zeros(3 * 9)),
```

```
155         concatCol(3, zeros(3 * 3), r, zeros(3 * 6)),
156         concatCol(3, zeros(3 * 6), r, zeros(3 * 3)),
157         concatCol(3, zeros(3 * 9), r));
158
159       var k_e_global = multiply(multiply(transpose(T, 12, 12), k_e, 12, 12, 12),
              T, 12, 12, 12);
160       setSub(this.FEM.K, edof_ind, plus(getSub(this.FEM.K, edof_ind), k_e_global
              ));
161     }
162   }
163
164   /**
165    * Computes the reactions based on the FEM model.
166    */
167   getReactions() {
168     var FEM = this.FEM;
169     FEM.Fn = scale(MP.g, MP.M);
170     setSub(FEM.F, [0, 1, 2], FEM.Fn);
171
172     FEM.Kp = concatRow(FEM.dof + FEM.Np,
173       concatCol(FEM.dof, FEM.K, transpose(FEM.C, FEM.dof, FEM.Np)),
174       concatCol(FEM.Np, FEM.C, zeros(FEM.Np * FEM.Np)));
175     FEM.Fp = concatRow(1, FEM.F, FEM.Q);
176
177
178     var dofs = concatCol(1, FEM.fdof, FEM.lmn);
179     var N = dofs.length;
180     var K_r = getSub(FEM.Kp, index(dofs, dofs, FEM.dof + FEM.Np));
181
182     var D1 = linsolve(K_r, getSub(FEM.Fp, dofs), N);
183     setSub(FEM.D, dofs, D1);
184     var F = multiply(FEM.Kp, FEM.D, FEM.dof + FEM.Np, FEM.dof + FEM.Np, 1);
185     FEM.R = getSub(F, FEM.rdof);
186     FEM.Rz = getSub(FEM.D, range(FEM.dof, FEM.dof + FEM.Np - 1));
187
188     FEM.DT = getSub(FEM.D, index([0, 1, 2], range(0, FEM.dof / 6 - 1), 6));
189     FEM.DR = getSub(FEM.D, index([3, 4, 5], range(0, FEM.dof / 6 - 1), 6));
190   }
191
192   /**
193    * Calculates the torques in the mechanism.
194    */
195   getTorques() {
196     var MP = this.MP;
197     var FEM = this.FEM;
198
199     var M = getSub(FEM.R, index([3, 4, 5], range(0, 5), 6));
200     FEM.M_m = sumCol(multiplyEl(M, MP.i_m), 3, 6); // [Nm]
201   }
202
203   /**
204    * Displays the results of the FEM analysis.
205    */
206   dispResults() {
207     var n = repRow(range(1, FEM.nn), 3);
208     disp(' Pomeranja');
209     disp(' Čvor | Komponenta | Pomeranje [mm]');
210     for(var i = 0; i < FEM.DT.length; i++) {
211       disp(sprintf(' %3d | %3d | %.10f', n[i], i % 3 + 1, FEM.DT[i]*1000));
```

```
212       }
213
214       disp(' Čvor | Komponenta | Pomeranje [deg]');
215       for(var i = 0; i < FEM.DR.length; i++) {
216         disp(sprintf(' %3d | %3d | %.10f', n[i], i % 3 + 1, FEM.DR[i]*r2d));
217       }
218
219       // Prikaz vrednosti reakcija
220       disp(' Reakcije veze');
221       var n = repRow(range(9, 35, 4), 6);
222       disp(' Cvor | Komponenta | Reakcije [N i Nm]');
223       for(var i = 0; i < FEM.R.length; i++) {
224         disp(sprintf(' %3d | %3d | %.10f', n[i], i % 6 + 1, FEM.R[i]));
225       }
226       disp(' Potrebni momenti motora: ');
227       for(var i = 0; i < 6; i++) {
228         disp(sprintf(' %.5f', FEM.M_m[i]) + ' Nm');
229       }
230       disp(' Ukupni potrebni moment motora: ' + num2str(sum(abs(FEM.M_m))) + ' Nm'
            );
231     }
232   }
233
234   exports.MP_6RSS_FEM_SOLVER = MP_6RSS_FEM_SOLVER;
```

Listing H.1 - SOLVER_FEM_MP_6RSS.jsl

I

Skripta solveInverseStaticFEM.jsl

Skriptom `solveInverseStaticFEM.jsl`, koja je u celosti data u listingu I.1, se prvo definišu geometrijski parametri mehanizma simulatora leta, zatim se rešava inverzni kinematički problem i potom se metodom konačnih elemenata određuju pomeranja u čvorovima elemenata. Prvo se formira matrica krutosti za konkretan slučaj pozicije i orijentacije platforme, zatim se definišu spoljašnja opterećenja, odgovarajuće provere za sferne zglobove i izračunavaju pomeranja. Nakon toga je moguće odrediti i reakcije veza iz kojih je moguće izraziti potrebni moment za obrtne aktuatore. Na kraju izvršavanja programa se u komandnom prozoru programskog paketa `JSLAB` ispisuju rezultati i formira se grafik sa prikazom modela mehanizma.

Metodom `dispResults()` klase `MP_6RSS_FEM_SOLVER` (data u prilogu H), se prikazuju vrednosti translatornih i ugaonih pomeranja za svaki čvor kao i vrednosti reakcija veza u čvorovima kojima su ograničeni određeni stepeni slobode kretanja.

Metodama `plotModelFEM()` i `plotResultFEM()` klase `MP_6RSS_PLOTTER` (data u prilogu C), se na prikaz osnovnog kinematičkog modela mehanizma simulatora leta dodaje prikaz konačnih elemenata u nedeformisanom i deformisanom stanju kao i vektori spoljašnjih opterećenja i reakcija veza.

```
/**
 * @file Solve Inverse Static FEM
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x = [ // translacija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  -30 // [deg] psi
];

// - Karakteristike materijala
var M = {
  E: 210e9, // Modul elasticnosti
```

```
24    ni: 0.3 // Poasonov koeficijent
25  };
26  // - Karaktetristike poprecnog preseka
27  var PP = {
28    type: 1, // kruzna cev
29    D: 50, // precnik cevi
30    t: 2 // debljina zida cevi
31  };
32
33  // - FEM
34  var FEM = {};
35  // Cvorovi elemenata
36  // en(element,[cvor1,cvor2]);
37  FEM.en = [1, 2,
38          2, 3,
39          2, 4,
40          2, 5,
41          2, 6,
42          2, 7,
43          2, 8,
44          3, 4,
45          4, 5,
46          5, 6,
47          6, 7,
48          7, 8,
49          8, 3,
50          9, 10,
51          11, 12,
52          13, 14,
53          15, 16,
54          17, 18,
55          19, 20,
56          21, 22,
57          23, 24,
58          25, 26,
59          27, 28,
60          29, 30,
61          31, 32];
62
63  // Povezivanje cvorova
64  // ec(cvor1,cvor2);
65  FEM.ec = [10, 11,
66          12, 3,
67          14, 15,
68          16, 4,
69          18, 19,
70          20, 5,
71          22, 23,
72          24, 6,
73          26, 27,
74          28, 7,
75          30, 31,
76          32, 8];
77
78  // Konturni uslovi
79  // 0 - slobodan
80  // 1 - ogranicen
81  FEM.bc = zeros(32*6);
82  setSub(FEM.bc, index(range(0, 5),
```

```
  range(8, FEM.bc.length / 6 - 1, 4), 6), repRow(ones(6), 6));
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

var { MP_6RSS_PLOTTER } = require('./classes/PLOTTER_MP_6RSS.jsl');
var plotter1 = new MP_6RSS_PLOTTER(MP);
var plotter2 = new MP_6RSS_PLOTTER(MP);

var { MP_6RSS_FEM_SOLVER } = require('./classes/SOLVER_FEM_MP_6RSS.jsl');
var solver_fem = new MP_6RSS_FEM_SOLVER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
solver.getGeometry();

// Odredjivanje potrebnog ugla alpha za date podatke
disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
solver.setParameters(x);

// Provera uglova u zglobovim
disp(' Provera uglova u zglobovima...');
solver.checkJoints();

// Koordinate cvorova
FEM.nc = zeros(3*32);
setSub(FEM.nc, index(range(0, 2), range(0, 1), 3), MP.POb);
setSub(FEM.nc, index(range(0, 2), range(2, 7), 3), MP.Pb);
setSub(FEM.nc, index(range(0, 2), range(8, 31, 4), 3), MP.B);
setSub(FEM.nc, index(range(0, 2), range(9, 31, 4), 3), MP.A);
setSub(FEM.nc, index(range(0, 2), range(10, 31, 4), 3), MP.A);
setSub(FEM.nc, index(range(0, 2), range(11, 31, 4), 3), MP.Pb);
FEM.nc = scale(FEM.nc, 1/1000); // [m]

// Formiranje MKE
disp(' Formiranje MKE...');
solver_fem.construct(FEM, M, PP);

// Proracun MKE
disp(' Odredjivanje reakcija...');
solver_fem.getReactions();

// Odredjivanje potrebnog momenta
disp(' Odredjivanje potrebnog momenta...');
solver_fem.getTorques();

if(any(MP.e)) {
  disp(' Neuspešan proračun!');
  dispRowVector('x', MP.x, MP.m);
  dispRowVector('e', MP.e, MP.m);
} else {
  disp(' Uspešan proračun!');
  dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);

  plotter1.plotModelFEM(solver_fem);
```

```
142   plotter2.plotResultFEM(solver_fem);
143
144   disp(' Prikaz mehanizma...');
145   plotter1.show(1);
146   plotter1.showLegend();
147
148   plotter2.show(2);
149   plotter2.showLegend();
150   solver_fem.dispResults();
151 }
152
153 // - Kraj proračuna
154 disp(' Program je uspešno izvršen... ');
155 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
156 disp(' -------------------- ');
```

Listing I.1 - `solveInverseStaticFEM.jsl`

J

JSLAB klasa SPACE_SEARCH

Klasa SPACE_SEARCH, napisana u programu JSLAB, u okviru koje je implementiran algoritam za pretragu, detaljno opisan u okviru potpoglavlja 4.9.1, je u celosti data u listingu J.1.

U listingu J.2 je dat primer za upotrebu ove klase i u njoj implementiranog algoritma.

```
1  /**
2   * @file Space Search
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8
9  /**
10  * Class for N-dimensional Space Search.
11  */
12 class SPACE_SEARCH {
13
14   /**
15    * Displays the size of elements based on bounds and subdivisions.
16    * @param {Array} bounds - Array of [min, max] for each dimension.
17    * @param {Array} subdivisionPerDepth - Subdivision factors per depth and
          dimension.
18    */
19   dispElementSize(bounds, subdivisionPerDepth) {
20     var numDimensions = bounds.length;
21     // Calculate the initial differences between bounds for each dimension
22     var dq = bounds.map((bound, index) => (bound[1] - bound[0]) /
          subdivisionPerDepth[0][index]);
23
24     // Copy dq to dqmin for further subdivision calculations
25     var dqmin = [...dq];
26
27     // Iterate through subdivisionPerDepth starting from the second depth level
28     for (var i = 1; i < subdivisionPerDepth.length; i++) {
29       for (var j = 0; j < numDimensions; j++) {
30         dqmin[j] /= subdivisionPerDepth[i][j];
31       }
32     }
33
34     // Display the results for initial and minimal element dimensions
35     disp(` Dimenzije inicijalnih elemenata: ${dq.map(val => val.toFixed(2)).join
          ('x')}`);
36     disp(` Dimenzije najmanjih elemenata: ${dqmin.map(val => val.toFixed(2)).
```

```
      join('x')}`);
}

/**
 * Splits the search space into smaller intervals for parallel processing.
 * @param {Array.<Array.<number>>} x_lim - The limits of the search space,
    where each sub-array represents [start, end] for a dimension.
 * @param {Array.<Array.<number>>} k - The parameters to optimize.
 * @param {number} N_proc - The number of processors to divide the work among.
 * @returns {Array.<Array.<number>>} An array containing the split search
    spaces and the adjusted parameters.
 */
splitSearchSpace(x_lim, k, N_proc) {
  const [start, end] = x_lim[0];
  const interval = (end - start) / N_proc;

  var k_out = [...k];
  k_out[0][0] = k_out[0][0]/N_proc;

  return [Array.from({ length: N_proc }, (_, i) => [
    [start + i * interval, start + (i + 1) * interval],
    ...x_lim.slice(1)
  ]), k_out];
}

/**
 * Executes a provided function in parallel across the split search spaces.
 * @param {Array.<Array.<number>>} x_lim - The limits of the search space,
    where each sub-array represents [start, end] for a dimension.
 * @param {Array.<Array.<number>>} k - The parameters to optimize.
 * @param {Object} context - The execution context containing necessary
    configurations and states.
 * @param {Function} setup - The setup function to initialize the parallel
    environment.
 * @param {Function} fun - The function to execute in parallel on each split
    of the search space.
 * @returns {Promise<Array.<Array.<number>>>} A promise that resolves to
    arrays of input and output results from the parallel execution.
 */
async runParallel(x_lim, k, context, setup, fun) {
  var N_proc = parallel.getProcessorsNum();

  var funStr = fun.toString();
  var funBody = funStr.slice(funStr.indexOf("{") + 1, funStr.lastIndexOf("}"))
     ;
  var funArgs = funStr.slice(funStr.indexOf("(") + 1, funStr.indexOf(")"));

  var [x_lim_parallel, k_parallel] = this.splitSearchSpace(x_lim, k, N_proc);

  var res = await parallel.parfor(0, N_proc-1, 1, N_proc,
      Object.assign(context, {x_lim_parallel, k_parallel,
      funArgs, funBody}), setup, function(i) {
    var fun = new Function(funArgs, funBody);
    return crawler.run(x_lim_parallel[i], k_parallel, fun);
  });
  var Nin = res.map(pair => pair[0]).flat();
  var Nout = res.map(pair => pair[1]).flat();
  return [Nin, Nout];
}
```

```
87
88    /**
89     * Executes the space search algorithm.
90     * @param {number[][]} bounds - Array of [min, max] for each dimension.
91     * @param {number[][]} subdivisionPerDepth - Subdivision factors for each
          depth and dimension.
92     * @param {function} conditionFunction - Function that determines if a point
          satisfies the condition.
93     * @returns {[number[][], number[][]]} - Arrays of points inside and outside
          the condition.
94     */
95    run(bounds, subdivisionPerDepth, conditionFunction) {
96      var numDimensions = bounds.length;
97      var maxDepth = subdivisionPerDepth.length;
98
99      // Initial starting point
100     var startCoordinates = bounds.map(([min]) => min);
101
102     // Calculate initial step sizes (dq)
103     var initialStepSizes = bounds.map(
104       ([min, max], idx) => (max - min) / subdivisionPerDepth[0][idx]
105     );
106
107     // Generate step sizes for each depth
108     var stepSizesPerDepth = [initialStepSizes];
109     for(var depth = 1; depth < maxDepth; depth++) {
110       var previousStepSizes = stepSizesPerDepth[depth - 1];
111       var newStepSizes = previousStepSizes.map(
112         (size, idx) => size / subdivisionPerDepth[depth][idx]
113       );
114       stepSizesPerDepth.push(newStepSizes);
115     }
116
117     // Compute the values at the boundary points
118     var cornerShifts = this.generateCornerShifts(numDimensions);
119     var boundaryPoints = cornerShifts.map(shift => {
120       return shift.map((s, idx) => startCoordinates[idx] + s * (bounds[idx][1] -
            bounds[idx][0]));
121     });
122     var boundaryValues = boundaryPoints.map(point => (conditionFunction(point) ?
          1 : 0));
123
124     // Call makeNodesND
125     var [Nin, Nout] = this.makeNodesND(
126       startCoordinates,
127       boundaryValues,
128       0,
129       stepSizesPerDepth,
130       subdivisionPerDepth,
131       conditionFunction
132     );
133
134     return [Nin, Nout];
135   }
136
137   /**
138    * Recursively creates nodes in N dimensions based on subdivisions.
139    * @param {number[]} startCoordinates - Starting coordinates for the current
          grid.
```

```
140     * @param {number[]} boundaryValues - Values at the boundary points of the
          current hypercube.
141     * @param {number} currentDepth - Current depth of the recursion.
142     * @param {number[][]} stepSizesPerDepth - Step sizes for each depth.
143     * @param {number[][]} subdivisionPerDepth - Subdivision factors for each
          depth.
144     * @param {function} conditionFunction - User-defined condition function.
145     * @returns {[number[][], number[][]]} - Arrays of points inside and outside
          the condition.
146     */
147    makeNodesND(
148      startCoordinates,
149      boundaryValues,
150      currentDepth,
151      stepSizesPerDepth,
152      subdivisionPerDepth,
153      conditionFunction
154    ) {
155      var numDimensions = startCoordinates.length;
156      var maxDepth = subdivisionPerDepth.length; // maximum depth
157      var Nin = [];
158      var Nout = [];
159
160      // Generate coordinate arrays for each dimension
161      var stepSizes = stepSizesPerDepth[currentDepth];
162      var numSteps = subdivisionPerDepth[currentDepth];
163      var coordsArray = [];
164      for(var idx = 0; idx < numDimensions; idx++) {
165        var steps = numSteps[idx];
166        var stepSize = stepSizes[idx];
167        var start = startCoordinates[idx];
168        var arr = Array.from(
169          { length: steps + 1 },
170          (_, i) => start + i * stepSize
171        );
172        coordsArray.push(arr);
173      }
174
175      // Initialize N-dimensional grid
176      var gridShape = coordsArray.map(arr => arr.length);
177      var N = createFilledArray(...gridShape, null);
178
179      // Mark whether a node needs to be calculated
180      var nf = createFilledArray(...gridShape, 1);
181
182      // Set nf to 0 for corner points (boundary points)
183      var cornerIndices = this.getCornerIndices(gridShape);
184      cornerIndices.forEach(indices => {
185        setValueAt(nf, indices, 0);
186      });
187
188      // Set the values at the corner points and add them to Nin or Nout
189      cornerIndices.forEach((indices, idx) => {
190        var coords = indices.map((index, dim) => coordsArray[dim][index]);
191        var value = boundaryValues[idx];
192        setValueAt(N, indices, [...coords, value]);
193
194        // On first level, add corner points to Nin or Nout
195        if(currentDepth == 0) {
```

```
196         if(value) {
197           Nin.push(coords);
198         } else {
199           Nout.push(coords);
200         }
201       }
202     });
203
204     // Now compute the values at all nodes where nf is 1
205     var indicesList = this.generateIndicesList(gridShape, nf);
206     for(var indices of indicesList) {
207       var coords = indices.map((index, dim) => coordsArray[dim][index]);
208       var e = conditionFunction(coords) ? 1 : 0;
209       setValueAt(N, indices, [...coords, e]);
210
211       if(e) {
212         Nin.push(coords);
213       } else {
214         Nout.push(coords);
215       }
216     }
217
218     // Check for further subdivision
219     if(currentDepth < maxDepth - 1) {
220       var innerCubeIndicesList = this.generateInnerCubeIndicesList(gridShape);
221       for(var indices of innerCubeIndicesList) {
222         var cubeCorners = this.getCubeCorners(N, indices);
223         var boundaryValues_sub = cubeCorners.map(corner => corner[corner.length
              - 1]);
224
225         var allSame = boundaryValues_sub.every(val => val === boundaryValues_sub
              [0]);
226         if(!allSame) {
227           var startCoords_sub = cubeCorners[0].slice(0, numDimensions);
228           var [Nin_sub, Nout_sub] = this.makeNodesND(
229             startCoords_sub,
230             boundaryValues_sub,
231             currentDepth + 1,
232             stepSizesPerDepth,
233             subdivisionPerDepth,
234             conditionFunction
235           );
236           Nin = Nin.concat(Nin_sub);
237           Nout = Nout.concat(Nout_sub);
238         }
239       }
240     }
241
242     return [Nin, Nout];
243   }
244
245   /**
246    * Generates all corner shifts for a hypercube in N dimensions.
247    * @param {number} numDimensions - Number of dimensions.
248    * @returns {number[][]} - Array of shifts for each corner.
249    */
250   generateCornerShifts(numDimensions) {
251     var shifts = [];
252     var totalCorners = 1 << numDimensions;
```

```
253     for(var i = 0; i < totalCorners; i++) {
254       var shift = [];
255       for(var j = 0; j < numDimensions; j++) {
256         shift.push((i >> j) & 1);
257       }
258       shifts.push(shift);
259     }
260     return shifts;
261   }
262
263   /**
264    * Retrieves the indices of corner points in the grid.
265    * @param {number[]} gridShape - Shape of the grid.
266    * @returns {number[][]} - List of corner indices.
267    */
268   getCornerIndices(gridShape) {
269     var numDimensions = gridShape.length;
270     var cornerShifts = this.generateCornerShifts(numDimensions);
271     return cornerShifts.map(shift => shift.map((s, idx) => s * (gridShape[idx] -
          1)));
272   }
273
274   /**
275    * Generates a list of indices for nodes that need to be calculated.
276    * @param {number[]} gridShape - Shape of the grid.
277    * @param {Array} nf - N-dimensional array indicating nodes to compute.
278    * @returns {number[][]} - List of node indices where nf is 1.
279    */
280   generateIndicesList(gridShape, nf) {
281     var indicesList = [];
282
283     var generateIndices = (indices, dim) => {
284       if(dim === gridShape.length) {
285         if(getValueAt(nf, indices) === 1) {
286           indicesList.push([...indices]);
287         }
288         return;
289       }
290       for(var i = 0; i < gridShape[dim]; i++) {
291         indices[dim] = i;
292         generateIndices(indices, dim + 1);
293       }
294     };
295     generateIndices(new Array(gridShape.length).fill(0), 0);
296     return indicesList;
297   }
298
299   /**
300    * Generates a list of starting indices for inner hypercubes.
301    * @param {number[]} gridShape - Shape of the grid.
302    * @returns {number[][]} - List of inner cube starting indices.
303    */
304   generateInnerCubeIndicesList(gridShape) {
305     var indicesList = [];
306
307     var generateIndices = (indices, dim) => {
308       if(dim === gridShape.length) {
309         indicesList.push([...indices]);
310         return;
```

```
311         }
312         if(gridShape[dim] <= 1) {
313           // No cubes along this dimension
314           return;
315         }
316         for(var i = 0; i < gridShape[dim] - 1; i++) {
317           indices[dim] = i;
318           generateIndices(indices, dim + 1);
319         }
320       };
321       generateIndices(new Array(gridShape.length).fill(0), 0);
322       return indicesList;
323     }
324
325     /**
326      * Retrieves the corner coordinates and values of a hypercube.
327      * @param {Array} N - N-dimensional grid.
328      * @param {number[]} indices - Starting indices of the hypercube.
329      * @returns {Array} - Array of corner coordinates and values.
330      */
331     getCubeCorners(N, indices) {
332       var numDimensions = indices.length;
333       var cornerShifts = this.generateCornerShifts(numDimensions);
334       var cubeCorners = cornerShifts.map(shift => {
335         var cornerIndices = indices.map((idx, dim) => idx + shift[dim]);
336         return getValueAt(N, cornerIndices);
337       });
338       return cubeCorners;
339     }
340 }
341
342 exports.SPACE_SEARCH = SPACE_SEARCH;
```

Listing J.1 - SPACE_SEARCH.jsl

```
1  /**
2   * @file Example for Space Search
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8  close('all'); clear; clc; tic;
9  disp(' --- ' + jsl_file_name + ' ---');
10 // --------------------
11
12 // Define the condition function
13 var numDimensions = 1; // Change this value for different dimensions
14 var minRadius = 27.3;
15 var maxRadius = 42.3;
16 var bounds = Array.from({ length: numDimensions }, () => [-50, 50]); // N-
       dimensional bounds
17 var subdivisionPerDepth = [
18   Array(numDimensions).fill(20),
19   Array(numDimensions).fill(5),
20   Array(numDimensions).fill(5),
21 ];
22
23 /**
24  * Condition function that checks if a point is within a hyperspherical shell.
```

```
25   * @param {number[]} coordinates - Point in N-dimensional space.
26   * @returns {number} - Returns 1 if the point satisfies the condition, otherwise
         0.
27   */
28  function conditionFunction(coordinates) {
29    var distanceFromOrigin = Math.sqrt(coordinates.reduce((sum, coord) => sum +
         coord * coord, 0));
30    return distanceFromOrigin >= minRadius && distanceFromOrigin <= maxRadius ? 1
         : 0;
31  }
32
33  // Instantiate the SPACE_SEARCH class
34  var { SPACE_SEARCH } = require('./SPACE_SEARCH.jsl');
35  var crawler = new SPACE_SEARCH();
36
37  // Run the solver
38  crawler.dispElementSize(bounds, subdivisionPerDepth);
39  var [pointsInside, pointsOutside] = crawler.run(
40    bounds,
41    subdivisionPerDepth,
42    conditionFunction
43  );
44
45  // Output the results
46  disp(`Total points checked: ${pointsInside.length + pointsOutside.length}`);
47  disp(`Points inside the range: ${pointsInside.length}`);
48  disp(`Points outside the range: ${pointsOutside.length}`);
49
50  if(numDimensions == 1 || numDimensions == 2) {
51    var x_in = column(pointsInside, 0);
52    var y_in = column(pointsInside, 1);
53    var x_out = column(pointsOutside, 0);
54    var y_out = column(pointsOutside, 1);
55    if(numDimensions == 1) {
56      y_in = zeros(x_in.length);
57      y_out = zeros(x_out.length);
58    }
59    figure(1);
60    plot([
61      {x: x_in, y: y_in, name: 'In', color: '#f00', mode: 'markers', type: '
           scattergl'},
62      {x: x_out, y: y_out, name: 'Out', color: '#000', mode: 'markers', type: '
           scattergl'}
63    ], {'font': {family: 'LatinModern', size: 18}});
64    xlabel("x [mm]");
65    ylabel("y [mm]");
66    axis("equal");
67    printFigure(pwd+'spaceSearch'+numDimensions+'D', {type: 'pdf', size: [560,
         420]});
68  }
69
70  // - Kraj proračuna
71  disp(' Program je uspešno izvršen... ');
72  disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
73  disp(' -------------------- ');
```

Listing J.2 - example_SpaceSearch.jsl

K

Skripta findWorkspaceParallel.jsl

Skriptom `findWorkspaceParallel.jsl`, koja je u celosti data u listingu K.1, se vrši određivanje radnog prostora mehanizma simulatora leta prema algoritmu prikazanom u poglavlju 4.9.4.

```
/**
 * @file Find max paramters - parallel
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var solver_type = ['js', 'cpp'][1];
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x_lim = [[-1000, 1000], [-1000, 1000], [-300, 300]]; // granicne vrednosti parametara
var k = [[20, 20, 60],
  createFilledArray(3, 5),
  createFilledArray(3, 2),
  createFilledArray(3, 2)]; // faktori pretrage
var solver_path = "C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS/build/Release/solver_mp_6rss";
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_PLOTTER } = require('./classes/PLOTTER_MP_6RSS.jsl');
var plotter = new MP_6RSS_PLOTTER(MP);

var { SPACE_SEARCH } = require('./classes/SPACE_SEARCH.jsl');
var crawler = new SPACE_SEARCH();
crawler.dispElementSize(x_lim, k);

// Odredjivanje radnog prostora
disp(' Odredjivanje radnog prostora...');
var t1 = tic;
var [Nin, Nout] = await crawler.runParallel(x_lim, k,
    {id, solver_type, solver_path}, function() {
  var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
  var MP = new MP_6RSS_MECHANISM();
```

```
38
39    var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
40    var solver = new MP_6RSS_SOLVER(MP);
41    solver.getGeometry();
42
43    var { SolverMP6RSS } = require(solver_path);
44    var solver_cpp = new SolverMP6RSS(MP);
45
46    var { SPACE_SEARCH } = require('./classes/SPACE_SEARCH.jsl');
47    var crawler = new SPACE_SEARCH();
48  }, function(x) {
49    if(solver_type == 'cpp') {
50      solver_cpp.setParameters([...x, ...zeros(3)]);
51      var res = solver_cpp.checkJoints();
52      return !any(res.e);
53    } else if(solver_type == 'js') {
54      solver.setParameters([...x, ...zeros(3)]);
55      solver.checkJoints();
56      return !any(solver.MP.e);
57    }
58  });
59  disp(' Potrebno vreme za proračun radnog prostora: ' + num2str(toc(t1), 2) + '
        sekundi');
60
61  var j = Nin.length + Nout.length;
62  var RP = Nin;
63
64  // Prikazivanje radnog prostora
65  disp(' Prikazivanje radnog prostora...');
66
67  disp(' Triangulacija tačaka u okviru radnog prostora...');
68  [b, V] = boundary3D(RP);
69  disp(' Zapremina radnog prostora je: ' + num2str(V/1e3, '%.2f') + ' cm^3');
70
71  disp(' Crtanje radnog prostora...');
72  plotter.showWorkspace(RP, b);
73
74  disp(' Broj uradjenih provera, j = ' + j);
75
76  // - Kraj proračuna
77  disp(' Program je uspešno izvršen... ');
78  disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
79  disp(' -------------------- ');
```

Listing K.1 - `findWorkspaceParallel.jsl`

L

JSLAB funkcija boundary3D

Funkcija `boundary3D()` za određivanje granične površi oblaka tačaka, u programu `JSLAB`, je implementirana koristeći `C++` biblioteku `CGAL` i sastavni je deo programa. Pored toga što je funkcija integrisana u program `JSLAB`, ona je ovde izdvojena, prikazana i objašnjena zbog značaja za ovaj rad (data je u listingu L.1).

Da bi se `C++` kod koristio iz `JSLAB`-a, neophodno je kompajlirati ga u `Node.js` modul. Datoteka `binding.gyp` (listing L.2) pruža konfiguraciju za `node-gyp`, alat koji omogućava kompajliranje `C++` koda u binarni modul kompatibilan sa `Node.js` okruženjem i to sa svim potrebnim bibliotekama (u ovom slučaju `CGAL`) i podešavanjima.

Header datoteka `alpha-shape-3d.h` (listing L.3) definiše klasu `AlphaShape3D` i uvozi sve potrebne biblioteke.

Izvorna datoteka `alpha-shape-3d.cpp` (listing L.4) sadrži implementaciju funkcija i metoda deklarisanih u `alpha-shape-3d.h`.

U nastavku su dati listinzi svih neophodnih datoteka.

```
boundary3D(points, shrink = 0.5) {
  var shp = new this.jsl.env.AlphaShape3D();
  shp.newShape(points);

  var Acrit = shp.getCriticalAlpha('one-region');
  var spec = shp.getAlphaSpectrum();

  var idx = spec.indexOf(Acrit);
  var subspec = spec.slice(idx);

  var idx = Math.max(Math.ceil((1 - shrink) * subspec.length) - 1, 0);
  var alphaval = subspec[idx];

  shp.setAlpha(alphaval);
  var V = shp.getVolume();
  var bf = shp.getBoundaryFacets();
  shp = null;
  return [bf, V];
}
```

Listing L.1 - `boundary3D.js`

```
[
  {
    "target_name": "alpha_shape_3d",
    "sources": [
      "cpp/alpha-shape-3d.cpp"
```

```
    ],
    "include_dirs": [
      "<!@(node -p \"require('node-addon-api').include\")",
      "<(module_root_dir)/lib/cgal-6.0.1/include/",
      "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/include",
      "<(module_root_dir)/lib/boost-1.86.0/"
    ],
    "libraries": [
      "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/lib/gmp.lib",
      "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/lib/mpfr.lib"
    ],
    "cflags!": [
      "-fno-exceptions"
    ],
    "cflags_cc!": [
      "-fno-exceptions",
      "-O3",
      "-DNDEBUG"
    ],
    "defines": [
      "NAPI_DISABLE_CPP_EXCEPTIONS"
    ],
    "copies": [
      {
        "destination": "<(module_root_dir)/build/Release",
        "files": [
          "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/bin/gmp-10.dll",
          "<(module_root_dir)/lib/cgal-6.0.1/auxiliary/gmp/bin/mpfr-6.dll"
        ]
      }
    ],
    "msvs_settings": {
      "VCCLCompilerTool": {
        "AdditionalOptions": [
          "-std:c++17",
          "/GR",
          "/EHsc"
        ]
      }
    }
  }
]
```

Listing L.2 - `binding.gyp`

```
// AlphaShape3D - alpha-shape-3d.h
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#ifndef ALPHA_SHAPE_3D_H
#define ALPHA_SHAPE_3D_H

//#define DEBUG_ALPHA_SHAPE_3D
//#define DEBUG_ALPHA_SHAPE_3D_LEVEL 0
//#define PROFILE_ALPHA_SHAPE_3D

#include <napi.h>
#include <chrono>
```

```
16 #include <thread>
17 #include <Windows.h>
18 #include <ctime>
19 #include <iomanip>
20 #include <sstream>
21 #include <string>
22 #include <fstream>
23 #include <iostream>
24 #include <filesystem>
25 #include <cassert>
26 #include <list>
27 #include <vector>
28
29 #include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
30
31 #include <CGAL/Delaunay_triangulation_3.h>
32 #include <CGAL/Triangulation_vertex_base_with_info_3.h>
33 #include <CGAL/Alpha_shape_3.h>
34 #include <CGAL/Alpha_shape_cell_base_3.h>
35 #include <CGAL/Alpha_shape_vertex_base_3.h>
36
37 #include <CGAL/Surface_mesh.h>
38 #include <CGAL/Surface_mesh_simplification/edge_collapse.h>
39 #include <CGAL/Surface_mesh_simplification/Policies/Edge_collapse/
       Edge_count_ratio_stop_predicate.h>
40
41 #include <CGAL/Polygon_mesh_processing/repair_polygon_soup.h>
42 #include <CGAL/Polygon_mesh_processing/orient_polygon_soup.h>
43 #include <CGAL/Polygon_mesh_processing/polygon_soup_to_polygon_mesh.h>
44 #include <CGAL/Polygon_mesh_processing/measure.h>
45 #include <CGAL/Polygon_mesh_processing/repair.h>
46
47 #include <CGAL/Search_traits_3.h>
48 #include <CGAL/Search_traits_adapter.h>
49 #include <CGAL/Orthogonal_k_neighbor_search.h>
50 #include <CGAL/boost/iterator/counting_iterator.hpp>
51
52 namespace alpha_shape_3d_ns {
53
54 using namespace std;
55 using namespace std::chrono;
56
57 namespace SMS = CGAL::Surface_mesh_simplification;
58 namespace PMP = CGAL::Polygon_mesh_processing;
59
60 typedef CGAL::Exact_predicates_inexact_constructions_kernel Gt;
61
62 typedef CGAL::Triangulation_vertex_base_with_info_3<unsigned, Gt> Tvb;
63 typedef CGAL::Alpha_shape_vertex_base_3<Gt, Tvb> Vb;
64 typedef CGAL::Alpha_shape_cell_base_3<Gt> Fb;
65 typedef CGAL::Triangulation_data_structure_3<Vb, Fb> Tds;
66 typedef CGAL::Delaunay_triangulation_3<Gt, Tds, CGAL::Fast_location> Dt;
67 typedef Dt::Point Point;
68
69 class search_map {
70     const std::vector<Point>& points;
71   public:
72     typedef Point value_type;
73     typedef const value_type& reference;
```

```
74      typedef std::size_t key_type;
75      typedef boost::lvalue_property_map_tag category;
76      search_map(const std::vector<Point>& pts):points(pts){}
77      reference operator[](key_type k) const {return points[k];}
78      friend reference get(const search_map& ppmap, key_type i)
79      {return ppmap[i];}
80  };
81
82  typedef CGAL::Alpha_shape_3<Dt> As3;
83  typedef CGAL::Surface_mesh<Point> Mesh;
84  typedef std::vector<std::size_t> CGAL_Polygon;
85
86  typedef CGAL::Search_traits_3<Gt> Trb;
87  typedef CGAL::Search_traits_adapter<std::size_t, search_map, Trb> Traits;
88  typedef CGAL::Orthogonal_k_neighbor_search<Traits>  K_neighbor_search;
89  typedef K_neighbor_search::Tree Tree;
90
91  template <typename T>
92  class CustomMatrix {
93    private:
94      std::vector<T> data;
95      uint32_t rows;
96      uint32_t cols;
97
98    public:
99      CustomMatrix() : rows(0), cols(0) {}
100     CustomMatrix(uint32_t r, uint32_t c) : rows(r), cols(c), data(r * c) {}
101
102     T& operator()(uint32_t i, uint32_t j) {
103       return data[i * cols + j];
104     }
105
106     const T& operator()(uint32_t i, uint32_t j) const {
107       return data[i * cols + j];
108     }
109
110     uint32_t numRows() const { return rows; }
111     uint32_t numCols() const { return cols; }
112
113     void resize(uint32_t r, uint32_t c) {
114       rows = r;
115       cols = c;
116       data.resize(r * c);
117     }
118 };
119 typedef CustomMatrix<double> Matrix;
120
121 class AlphaShape3D : public Napi::ObjectWrap<AlphaShape3D> {
122  public:
123   static Napi::Object Init(Napi::Env env, Napi::Object exports);
124   AlphaShape3D(const Napi::CallbackInfo& info);
125   ~AlphaShape3D();
126
127   Matrix inputPoints;
128   std::vector<Point> Points;
129   std::vector<std::pair<Point, unsigned>> Vertices;
130   double getAlpha(void);
131   void setAlpha(double);
132   double numRegions(void);
```

```
133   Matrix getAlphaSpectrum(void);
134   double getCriticalAlpha(std::string);
135   double getSurfaceArea(void);
136   double getVolume(void);
137   Matrix getBoundaryFacets(void);
138   Matrix getBoundaryFacets(std::string);
139   void writeBoundaryFacets(std::string);
140   Matrix checkInShape(Matrix);
141   Matrix getTriangulation(void);
142   std::pair<Matrix, Matrix> getNearestNeighbor(Matrix);
143   std::pair<Matrix, Matrix> getSimplifiedShape(double);
144   std::pair<Matrix, Matrix> getSimplifiedShape();
145   std::pair<Matrix, Matrix> getSimplifiedShape(std::string);
146   std::pair<Matrix, Matrix> getSimplifiedShape(double, std::string);
147   std::pair<Matrix, Matrix> removeUnusedPoints(Matrix, Matrix);
148   void writeOff(std::string, Matrix, Matrix);
149
150   // New JavaScript wrapper methods
151   void NewShapeJS(const Napi::CallbackInfo& info);
152   Napi::Value GetAlphaJS(const Napi::CallbackInfo& info);
153   void SetAlphaJS(const Napi::CallbackInfo& info);
154   Napi::Value GetNumRegionsJS(const Napi::CallbackInfo& info);
155   Napi::Value GetAlphaSpectrumJS(const Napi::CallbackInfo& info);
156   Napi::Value GetCriticalAlphaJS(const Napi::CallbackInfo& info);
157   Napi::Value GetSurfaceAreaJS(const Napi::CallbackInfo& info);
158   Napi::Value GetVolumeJS(const Napi::CallbackInfo& info);
159   Napi::Value GetBoundaryFacetsJS(const Napi::CallbackInfo& info);
160   void WriteBoundaryFacetsJS(const Napi::CallbackInfo& info);
161   Napi::Value CheckInShapeJS(const Napi::CallbackInfo& info);
162   void WriteOffJS(const Napi::CallbackInfo& info);
163   Napi::Value GetTriangulationJS(const Napi::CallbackInfo& info);
164   Napi::Value GetNearestNeighborJS(const Napi::CallbackInfo& info);
165   Napi::Value GetSimplifiedShapeJS(const Napi::CallbackInfo& info);
166   Napi::Value RemoveUnusedPointsJS(const Napi::CallbackInfo& info);
167
168 private:
169   As3 *alphaShape;
170   Dt *delaunayTriangulation;
171   Matrix triangulationMatrix;
172   std::size_t numAlphaValues;
173   Mesh surface_mesh;
174 };
175
176 }// namespace alpha_shape_3d_ns
177
178 #endif // ALPHA_SHAPE_3D_H
```

Listing L.3 - `alpha-shape-3d.h`

```
1  // AlphaShape3D - alpha-shape-3d.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "alpha-shape-3d.h"
8
9  namespace alpha_shape_3d_ns {
10
11 #ifdef PROFILE_SOLVER_MP_6RSS
```

```
12 // Function to start the timer and return the start time
13 time_point<steady_clock> tic() {
14   return steady_clock::now();
15 }
16
17 // Function to stop the timer and return the elapsed time
18 long toc(const time_point<steady_clock>& startTime) {
19   return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
20 }
21 #endif
22
23 // Function to get current time
24 std::string getCurrentTime() {
25   // get current time
26   auto now = system_clock::now();
27
28   // get number of milliseconds for the current second
29   // (remainder after division into seconds)
30   auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
31
32   // convert to std::time_t in order to convert to std::tm (broken time)
33   auto timer = system_clock::to_time_t(now);
34
35   // convert to broken time
36   std::tm bt = *std::localtime(&timer);
37
38   std::ostringstream oss;
39
40   oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
41   oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
42
43   return oss.str();
44 }
45
46 // Function to console log data
47 int consoleLog(uint8_t level, const char* format, ...) {
48 #ifdef DEBUG_SOLVER_MP_6RSS_LEVEL
49   if(level <= DEBUG_SOLVER_MP_6RSS_LEVEL) {
50     printf("\033[0;33m[%s SolverMP6RSS]\033[0m ", getCurrentTime().c_str());
51     va_list vl;
52     va_start(vl, format);
53     auto ret = vprintf(format, vl);
54     va_end(vl);
55     printf("\n");
56     return ret;
57   }
58 #endif
59   return 0;
60 }
61
62 // AlphaShape3D()
63 // Object constructor
64 // --------------------
65 AlphaShape3D::AlphaShape3D(const Napi::CallbackInfo& info) : Napi::ObjectWrap<
     AlphaShape3D>(info) {
66 #ifdef DEBUG_ALPHA_SHAPE_3D
67   consoleLog(0, "Called constructor");
68 #endif
69   this->alphaShape = nullptr;
```

```
70    this->delaunayTriangulation = nullptr;
71  }
72
73  // ~AlphaShape3D()
74  // Object destructor
75  // --------------------
76  AlphaShape3D::~AlphaShape3D(void) {
77    if(this->delaunayTriangulation){
78      delete this->delaunayTriangulation;
79      this->delaunayTriangulation = nullptr;
80    }
81    if(this->alphaShape){
82      delete this->alphaShape;
83      this->alphaShape = nullptr;
84    }
85  }
86
87  // Init() function
88  // --------------------
89  Napi::Object AlphaShape3D::Init(Napi::Env env, Napi::Object exports) {
90    Napi::Function func = DefineClass(env, "AlphaShape3D", {
91      InstanceMethod("newShape", &AlphaShape3D::NewShapeJS),
92      InstanceMethod("getAlpha", &AlphaShape3D::GetAlphaJS),
93      InstanceMethod("setAlpha", &AlphaShape3D::SetAlphaJS),
94      InstanceMethod("getNumRegions", &AlphaShape3D::GetNumRegionsJS),
95      InstanceMethod("getAlphaSpectrum", &AlphaShape3D::GetAlphaSpectrumJS),
96      InstanceMethod("getCriticalAlpha", &AlphaShape3D::GetCriticalAlphaJS),
97      InstanceMethod("getSurfaceArea", &AlphaShape3D::GetSurfaceAreaJS),
98      InstanceMethod("getVolume", &AlphaShape3D::GetVolumeJS),
99      InstanceMethod("getBoundaryFacets", &AlphaShape3D::GetBoundaryFacetsJS),
100     InstanceMethod("writeBoundaryFacets", &AlphaShape3D::WriteBoundaryFacetsJS),
101     InstanceMethod("checkInShape", &AlphaShape3D::CheckInShapeJS),
102     InstanceMethod("writeOff", &AlphaShape3D::WriteOffJS),
103     InstanceMethod("getTriangulation", &AlphaShape3D::GetTriangulationJS),
104     InstanceMethod("getNearestNeighbor", &AlphaShape3D::GetNearestNeighborJS),
105     InstanceMethod("getSimplifiedShape", &AlphaShape3D::GetSimplifiedShapeJS),
106     InstanceMethod("removeUnusedPoints", &AlphaShape3D::RemoveUnusedPointsJS)
107   });
108   Napi::FunctionReference* constructor = new Napi::FunctionReference();
109   *constructor = Napi::Persistent(func);
110   env.SetInstanceData(constructor);
111
112   exports.Set("AlphaShape3D", func);
113   return exports;
114 }
115
116 // NewShapeJS() function
117 // --------------------
118 void AlphaShape3D::NewShapeJS(const Napi::CallbackInfo& info) {
119 #ifdef DEBUG_ALPHA_SHAPE_3D
120   consoleLog(0, "Called NewShapeJS()");
121 #endif
122
123   Napi::Env env = info.Env();
124   if(info.Length() < 1 || !info[0].IsArray()){
125     Napi::TypeError::New(env, "Array of points expected").
          ThrowAsJavaScriptException();
126     return;
127   }
```

```
128
129   Napi::Array jsPoints = info[0].As<Napi::Array>();
130   uint32_t numPoints = jsPoints.Length();
131
132   // Clear existing data to prevent accumulation
133   this->inputPoints.resize(0, 0);
134   this->Points.clear();
135   this->Vertices.clear();
136
137   // Initialize inputPoints matrix
138   this->inputPoints.resize(numPoints, 3);
139
140   for(uint32_t i = 0; i < numPoints; i++){
141     Napi::Value point = jsPoints[i];
142     if(!point.IsArray()){
143       Napi::TypeError::New(env, "Each point should be an array of 3 numbers").
          ThrowAsJavaScriptException();
144       return;
145     }
146     Napi::Array jsPoint = point.As<Napi::Array>();
147     if(jsPoint.Length() != 3){
148       Napi::TypeError::New(env, "Each point should have exactly 3 coordinates").
          ThrowAsJavaScriptException();
149       return;
150     }
151
152     for(uint32_t j = 0; j < 3; j++){
153       this->inputPoints(i, j) = jsPoint.Get(j).As<Napi::Number>().DoubleValue();
154     }
155   }
156
157 #ifdef DEBUG_ALPHA_SHAPE_3D
158   std::chrono::steady_clock::time_point begin =
159     std::chrono::steady_clock::now();
160 #endif
161
162   uint32_t n = this->inputPoints.numRows();
163
164 #ifdef DEBUG_ALPHA_SHAPE_3D
165   std::cout << "Reading " << n << " points " << std::endl;
166 #endif
167
168   for(std::size_t i = 0; i < n; i++){
169     this->Points.emplace_back(
170       Point(this->inputPoints(i, 0), this->inputPoints(i, 1), this->inputPoints(
          i, 2)));
171     this->Vertices.emplace_back(std::make_pair(this->Points.back(), i));
172   }
173
174 #ifdef DEBUG_ALPHA_SHAPE_3D
175   std::cout << "Computing delaunay triangulation." << std::endl;
176 #endif
177
178   // Delete existing triangulation and alphaShape to prevent memory leaks
179   if(this->delaunayTriangulation){
180     delete this->delaunayTriangulation;
181     this->delaunayTriangulation = nullptr;
182   }
183   if(this->alphaShape){
```

```
184       delete this->alphaShape;
185       this->alphaShape = nullptr;
186     }
187
188     this->delaunayTriangulation = new Dt(this->Vertices.begin(), this->Vertices.
          end());
189
190 #ifdef DEBUG_ALPHA_SHAPE_3D
191     std::cout << "Number of triangulation cells is "
192               << this->delaunayTriangulation->number_of_finite_cells() << std::
                    endl;
193 #endif
194
195     this->triangulationMatrix.resize(this->delaunayTriangulation->
          number_of_finite_cells()*4, 3);
196     uint64_t i_idx = 0;
197     for(Dt::Finite_cells_iterator cit = this->delaunayTriangulation->
          finite_cells_begin();
198         cit != this->delaunayTriangulation->finite_cells_end(); cit++){
199       this->triangulationMatrix(i_idx, 0) = cit->vertex(0)->info();
200       this->triangulationMatrix(i_idx, 1) = cit->vertex(1)->info();
201       this->triangulationMatrix(i_idx, 2) = cit->vertex(2)->info();
202       i_idx++;
203       this->triangulationMatrix(i_idx, 0) = cit->vertex(0)->info();
204       this->triangulationMatrix(i_idx, 1) = cit->vertex(2)->info();
205       this->triangulationMatrix(i_idx, 2) = cit->vertex(3)->info();
206       i_idx++;
207       this->triangulationMatrix(i_idx, 0) = cit->vertex(1)->info();
208       this->triangulationMatrix(i_idx, 1) = cit->vertex(2)->info();
209       this->triangulationMatrix(i_idx, 2) = cit->vertex(3)->info();
210       i_idx++;
211       this->triangulationMatrix(i_idx, 0) = cit->vertex(0)->info();
212       this->triangulationMatrix(i_idx, 1) = cit->vertex(1)->info();
213       this->triangulationMatrix(i_idx, 2) = cit->vertex(3)->info();
214       i_idx++;
215     }
216
217 #ifdef DEBUG_ALPHA_SHAPE_3D
218     std::cout << "Computing alpha shapes." << std::endl;
219 #endif
220     this->alphaShape = new As3(*this->delaunayTriangulation, As3::GENERAL);
221
222     this->numAlphaValues = this->alphaShape->number_of_alphas();
223
224 #ifdef DEBUG_ALPHA_SHAPE_3D
225     std::cout << "Number of alpha values is "
226               << this->numAlphaValues << std::endl;
227     std::cout << "Max alpha value is "
228               << this->alphaShape->get_nth_alpha(this->numAlphaValues) << std::
                    endl;
229     std::cout << "Min of alpha value is "
230               << this->alphaShape->get_nth_alpha(1) << std::endl;
231 #ifdef PROFILE_SOLVER_MP_6RSS
232     std::chrono::steady_clock::time_point end =
233       std::chrono::steady_clock::now();
234     std::cout << "Time elapsed = "
235         << std::chrono::duration_cast<std::chrono::milliseconds>
236           (end - begin).count()
237         << " ms" << std::endl;
```

```
238 #endif
239 #endif
240 }
241
242 // GetAlphaJS() function
243 // --------------------
244 Napi::Value AlphaShape3D::GetAlphaJS(const Napi::CallbackInfo& info) {
245 #ifdef DEBUG_ALPHA_SHAPE_3D
246   consoleLog(0, "Called GetAlphaJS()");
247 #endif
248
249   Napi::Env env = info.Env();
250   double result = this->getAlpha();
251   return Napi::Number::New(env, result);
252 }
253
254 // SetAlphaJS() function
255 // --------------------
256 void AlphaShape3D::SetAlphaJS(const Napi::CallbackInfo& info) {
257 #ifdef DEBUG_ALPHA_SHAPE_3D
258   consoleLog(0, "Called SetAlphaJS()");
259 #endif
260
261   Napi::Env env = info.Env();
262   if(info.Length() < 1 || !info[0].IsNumber()){
263     Napi::TypeError::New(env, "Number expected").ThrowAsJavaScriptException();
264   }
265   double alpha = info[0].As<Napi::Number>().DoubleValue();
266   this->setAlpha(alpha);
267 }
268
269 // GetNumRegionsJS() function
270 // --------------------
271 Napi::Value AlphaShape3D::GetNumRegionsJS(const Napi::CallbackInfo& info) {
272 #ifdef DEBUG_ALPHA_SHAPE_3D
273   consoleLog(0, "Called GetNumRegionsJS()");
274 #endif
275
276   Napi::Env env = info.Env();
277   double result = this->numRegions();
278   return Napi::Number::New(env, result);
279 }
280
281 // GetAlphaSpectrumJS() function
282 // --------------------
283 Napi::Value AlphaShape3D::GetAlphaSpectrumJS(const Napi::CallbackInfo& info) {
284 #ifdef DEBUG_ALPHA_SHAPE_3D
285   consoleLog(0, "Called GetAlphaSpectrumJS()");
286 #endif
287
288   Napi::Env env = info.Env();
289   Matrix spectrum = this->getAlphaSpectrum();
290   Napi::Array result = Napi::Array::New(env, spectrum.numCols());
291   for(size_t i = 0; i < spectrum.numCols(); i++){
292     result.Set(i, Napi::Number::New(env, spectrum(0, i)));
293   }
294   return result;
295 }
296
```

```
297 // GetCriticalAlphaJS() function
298 // --------------------
299 Napi::Value AlphaShape3D::GetCriticalAlphaJS(const Napi::CallbackInfo& info) {
300 #ifdef DEBUG_ALPHA_SHAPE_3D
301   consoleLog(0, "Called GetCriticalAlphaJS()");
302 #endif
303
304   Napi::Env env = info.Env();
305   if(info.Length() < 1 || !info[0].IsString()){
306     Napi::TypeError::New(env, "String expected").ThrowAsJavaScriptException();
307     return env.Null();
308   }
309   std::string type = info[0].As<Napi::String>().Utf8Value();
310   double result = this->getCriticalAlpha(type);
311   return Napi::Number::New(env, result);
312 }
313
314 // GetSurfaceAreaJS() function
315 // --------------------
316 Napi::Value AlphaShape3D::GetSurfaceAreaJS(const Napi::CallbackInfo& info) {
317 #ifdef DEBUG_ALPHA_SHAPE_3D
318   consoleLog(0, "Called GetSurfaceAreaJS()");
319 #endif
320
321   Napi::Env env = info.Env();
322   double result = this->getSurfaceArea();
323   return Napi::Number::New(env, result);
324 }
325
326 // GetVolumeJS() function
327 // --------------------
328 Napi::Value AlphaShape3D::GetVolumeJS(const Napi::CallbackInfo& info) {
329 #ifdef DEBUG_ALPHA_SHAPE_3D
330   consoleLog(0, "Called GetVolumeJS()");
331 #endif
332
333   Napi::Env env = info.Env();
334   double result = this->getVolume();
335   return Napi::Number::New(env, result);
336 }
337
338 // GetBoundaryFacetsJS() function
339 // --------------------
340 Napi::Value AlphaShape3D::GetBoundaryFacetsJS(const Napi::CallbackInfo& info) {
341 #ifdef DEBUG_ALPHA_SHAPE_3D
342   consoleLog(0, "Called GetBoundaryFacetsJS()");
343 #endif
344
345   Napi::Env env = info.Env();
346   Matrix facets = this->getBoundaryFacets();
347   Napi::Array result = Napi::Array::New(env, facets.numRows());
348   for(size_t i = 0; i < facets.numRows(); i++){
349     Napi::Array facet = Napi::Array::New(env, 3);
350     for(size_t j = 0; j < 3; j++){
351       facet.Set(j, Napi::Number::New(env, facets(i, j)));
352     }
353     result.Set(i, facet);
354   }
355   return result;
```

```
356 }
357
358 // WriteBoundaryFacetsJS() function
359 // --------------------
360 void AlphaShape3D::WriteBoundaryFacetsJS(const Napi::CallbackInfo& info) {
361 #ifdef DEBUG_ALPHA_SHAPE_3D
362   consoleLog(0, "Called WriteBoundaryFacetsJS()");
363 #endif
364
365   Napi::Env env = info.Env();
366   if(info.Length() < 1 || !info[0].IsString()){
367     Napi::TypeError::New(env, "String expected").ThrowAsJavaScriptException();
368   }
369   std::string filename = info[0].As<Napi::String>().Utf8Value();
370   this->writeBoundaryFacets(filename);
371 }
372
373 // CheckInShapeJS() function
374 // --------------------
375 Napi::Value AlphaShape3D::CheckInShapeJS(const Napi::CallbackInfo& info) {
376 #ifdef DEBUG_ALPHA_SHAPE_3D
377   consoleLog(0, "Called CheckInShapeJS()");
378 #endif
379
380   Napi::Env env = info.Env();
381   if(info.Length() < 1 || !info[0].IsArray()){
382     Napi::TypeError::New(env, "Array expected").ThrowAsJavaScriptException();
383     return env.Null();
384   }
385   Napi::Array jsArray = info[0].As<Napi::Array>();
386   Matrix QP(jsArray.Length(), 3);
387   for(size_t i = 0; i < jsArray.Length(); i++){
388     Napi::Array point = jsArray.Get(i).As<Napi::Array>();
389     for(size_t j = 0; j < 3; j++){
390       QP(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
391     }
392   }
393   Matrix result = this->checkInShape(QP);
394   Napi::Array jsResult = Napi::Array::New(env, result.numRows());
395   for(size_t i = 0; i < result.numRows(); i++){
396     jsResult.Set(i, Napi::Boolean::New(env, result(i, 0) != 0));
397   }
398   return jsResult;
399 }
400
401 // WriteOffJS() function
402 // --------------------
403 void AlphaShape3D::WriteOffJS(const Napi::CallbackInfo& info) {
404 #ifdef DEBUG_ALPHA_SHAPE_3D
405   consoleLog(0, "Called WriteOffJS()");
406 #endif
407
408   Napi::Env env = info.Env();
409   if(info.Length() < 3 || !info[0].IsString() || !info[1].IsArray() || !info[2].
       IsArray()){
410     Napi::TypeError::New(env, "Expected arguments: filename (string), points (
         array), facets (array)").ThrowAsJavaScriptException();
411   }
412
```

```
413   std::string filename = info[0].As<Napi::String>().Utf8Value();
414   Napi::Array jsPoints = info[1].As<Napi::Array>();
415   Napi::Array jsFacets = info[2].As<Napi::Array>();
416
417   Matrix Points(jsPoints.Length(), 3);
418   Matrix bf(jsFacets.Length(), 3);
419
420   for(size_t i = 0; i < jsPoints.Length(); i++){
421     Napi::Array point = jsPoints.Get(i).As<Napi::Array>();
422     for(size_t j = 0; j < 3; j++){
423       Points(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
424     }
425   }
426
427   for(size_t i = 0; i < jsFacets.Length(); i++){
428     Napi::Array facet = jsFacets.Get(i).As<Napi::Array>();
429     for(size_t j = 0; j < 3; j++){
430       bf(i, j) = facet.Get(j).As<Napi::Number>().DoubleValue();
431     }
432   }
433
434   this->writeOff(filename, Points, bf);
435 }
436
437 // GetTriangulationJS() function
438 // --------------------
439 Napi::Value AlphaShape3D::GetTriangulationJS(const Napi::CallbackInfo& info) {
440 #ifdef DEBUG_ALPHA_SHAPE_3D
441   consoleLog(0, "Called GetTriangulationJS()");
442 #endif
443
444   Napi::Env env = info.Env();
445   Matrix triangulation = this->getTriangulation();
446   Napi::Array result = Napi::Array::New(env, triangulation.numRows());
447   for(size_t i = 0; i < triangulation.numRows(); i++){
448     Napi::Array row = Napi::Array::New(env, 3);
449     for(size_t j = 0; j < 3; j++){
450       row.Set(j, Napi::Number::New(env, triangulation(i, j)));
451     }
452     result.Set(i, row);
453   }
454   return result;
455 }
456
457 // GetNearestNeighborJS() function
458 // --------------------
459 Napi::Value AlphaShape3D::GetNearestNeighborJS(const Napi::CallbackInfo& info) {
460 #ifdef DEBUG_ALPHA_SHAPE_3D
461   consoleLog(0, "Called GetNearestNeighborJS()");
462 #endif
463
464   Napi::Env env = info.Env();
465   if(info.Length() < 1 || !info[0].IsArray()){
466     Napi::TypeError::New(env, "Array expected").ThrowAsJavaScriptException();
467     return env.Null();
468   }
469   Napi::Array jsArray = info[0].As<Napi::Array>();
470   Matrix QP(jsArray.Length(), 3);
471   for(size_t i = 0; i < jsArray.Length(); i++){
```

```
472       Napi::Array point = jsArray.Get(i).As<Napi::Array>();
473       for(size_t j = 0; j < 3; j++){
474         QP(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
475       }
476     }
477     std::pair<Matrix, Matrix> result = this->getNearestNeighbor(QP);
478 
479 
480     Napi::Object jsResult = Napi::Object::New(env);
481 
482     Napi::Array indices = Napi::Array::New(env, result.first.numRows());
483     Napi::Array distances = Napi::Array::New(env, result.second.numRows());
484 
485     for(size_t i = 0; i < result.first.numRows(); i++){
486       indices.Set(i, Napi::Number::New(env, result.first(i, 0)));
487       distances.Set(i, Napi::Number::New(env, result.second(i, 0)));
488     }
489 
490     jsResult.Set("indices", indices);
491     jsResult.Set("distances", distances);
492     return jsResult;
493   }
494 
495   // GetSimplifiedShapeJS() function
496   // --------------------
497   Napi::Value AlphaShape3D::GetSimplifiedShapeJS(const Napi::CallbackInfo& info) {
498   #ifdef DEBUG_ALPHA_SHAPE_3D
499     consoleLog(0, "Called GetSimplifiedShapeJS()");
500   #endif
501 
502     Napi::Env env = info.Env();
503     std::pair<Matrix, Matrix> result;
504 
505     if(info.Length() == 0){
506       result = this->getSimplifiedShape();
507     }
508     else if(info.Length() == 1){
509       if(info[0].IsNumber()){
510         double stop_ratio = info[0].As<Napi::Number>().DoubleValue();
511         result = this->getSimplifiedShape(stop_ratio);
512       }
513       else if(info[0].IsString()){
514         std::string filename = info[0].As<Napi::String>().Utf8Value();
515         result = this->getSimplifiedShape(filename);
516       }
517       else{
518         Napi::TypeError::New(env, "Invalid argument type").
              ThrowAsJavaScriptException();
519         return env.Null();
520       }
521     }
522     else if(info.Length() == 2 && info[0].IsNumber() && info[1].IsString()){
523       double stop_ratio = info[0].As<Napi::Number>().DoubleValue();
524       std::string filename = info[1].As<Napi::String>().Utf8Value();
525       result = this->getSimplifiedShape(stop_ratio, filename);
526     }
527     else{
528       Napi::TypeError::New(env, "Invalid arguments").ThrowAsJavaScriptException();
529       return env.Null();
```

```
530   }
531
532   Napi::Object jsResult = Napi::Object::New(env);
533   Napi::Array points = Napi::Array::New(env, result.first.numRows());
534   Napi::Array facets = Napi::Array::New(env, result.second.numRows());
535
536   for(size_t i = 0; i < result.first.numRows(); i++){
537     Napi::Array point = Napi::Array::New(env, 3);
538     for(size_t j = 0; j < 3; j++){
539       point.Set(j, Napi::Number::New(env, result.first(i, j)));
540     }
541     points.Set(i, point);
542   }
543
544   for(size_t i = 0; i < result.second.numRows(); i++){
545     Napi::Array facet = Napi::Array::New(env, 3);
546     for(size_t j = 0; j < 3; j++){
547       facet.Set(j, Napi::Number::New(env, result.second(i, j)));
548     }
549     facets.Set(i, facet);
550   }
551
552   jsResult.Set("points", points);
553   jsResult.Set("facets", facets);
554   return jsResult;
555 }
556
557 // RemoveUnusedPointsJS() function
558 // ---------------------
559 Napi::Value AlphaShape3D::RemoveUnusedPointsJS(const Napi::CallbackInfo& info) {
560 #ifdef DEBUG_ALPHA_SHAPE_3D
561   consoleLog(0, "Called RemoveUnusedPointsJS()");
562 #endif
563
564   Napi::Env env = info.Env();
565   if(info.Length() < 2 || !info[0].IsArray() || !info[1].IsArray()){
566     Napi::TypeError::New(env, "Two arrays expected").ThrowAsJavaScriptException
          ();
567     return env.Null();
568   }
569
570   Napi::Array jsPoints = info[0].As<Napi::Array>();
571   Napi::Array jsFacets = info[1].As<Napi::Array>();
572
573   Matrix Pi(jsPoints.Length(), 3);
574   Matrix bfi(jsFacets.Length(), 3);
575
576   for(size_t i = 0; i < jsPoints.Length(); i++){
577     Napi::Array point = jsPoints.Get(i).As<Napi::Array>();
578     for(size_t j = 0; j < 3; j++){
579       Pi(i, j) = point.Get(j).As<Napi::Number>().DoubleValue();
580     }
581   }
582
583   for(size_t i = 0; i < jsFacets.Length(); i++){
584     Napi::Array facet = jsFacets.Get(i).As<Napi::Array>();
585     for(size_t j = 0; j < 3; j++){
586       bfi(i, j) = facet.Get(j).As<Napi::Number>().DoubleValue();
587     }
```

```
588   }
589
590   std::pair<Matrix, Matrix> result = this->removeUnusedPoints(Pi, bfi);
591
592   Napi::Object jsResult = Napi::Object::New(env);
593   Napi::Array points = Napi::Array::New(env, result.first.numRows());
594   Napi::Array facets = Napi::Array::New(env, result.second.numRows());
595
596   for(size_t i = 0; i < result.first.numRows(); i++){
597     Napi::Array point = Napi::Array::New(env, 3);
598     for(size_t j = 0; j < 3; j++){
599       point.Set(j, Napi::Number::New(env, result.first(i, j)));
600     }
601     points.Set(i, point);
602   }
603
604   for(size_t i = 0; i < result.second.numRows(); i++){
605     Napi::Array facet = Napi::Array::New(env, 3);
606     for(size_t j = 0; j < 3; j++){
607       facet.Set(j, Napi::Number::New(env, result.second(i, j)));
608     }
609     facets.Set(i, facet);
610   }
611
612   jsResult.Set("points", points);
613   jsResult.Set("facets", facets);
614   return jsResult;
615 }
616
617 // getAlpha() function
618 // --------------------
619 double AlphaShape3D::getAlpha(void) {
620 #ifdef DEBUG_ALPHA_SHAPE_3D
621   consoleLog(0, "Called getAlpha()");
622 #endif
623
624   return this->alphaShape->get_alpha();
625 }
626
627 // setAlpha() function
628 // --------------------
629 void AlphaShape3D::setAlpha(double alpha) {
630 #ifdef DEBUG_ALPHA_SHAPE_3D
631   consoleLog(0, "Called setAlpha()");
632 #endif
633
634   this->surface_mesh.clear();
635   this->alphaShape->set_alpha(alpha);
636
637 #ifdef DEBUG_ALPHA_SHAPE_3D
638   std::cout << "Number of solid components for alpha " << alpha
639       << " is " << this->numRegions() << std::endl;
640   std::list<As3::Cell_handle>     cells;
641   std::list<As3::Facet>           facets;
642   std::list<As3::Edge>            edges;
643   std::list<As3::Vertex_handle>   vertices;
644   this->alphaShape->get_alpha_shape_cells(std::back_inserter(cells),
645       As3::INTERIOR);
646   this->alphaShape->get_alpha_shape_facets(std::back_inserter(facets),
```

```
647         As3::REGULAR);
648     this->alphaShape->get_alpha_shape_facets(std::back_inserter(facets),
649         As3::SINGULAR);
650     this->alphaShape->get_alpha_shape_edges(std::back_inserter(edges),
651         As3::SINGULAR);
652     this->alphaShape->get_alpha_shape_vertices(std::back_inserter(vertices),
653         As3::REGULAR);
654     std::cout << "Number of interior tetrahedra is "
655               << cells.size() << std::endl;
656     std::cout << "Number of boundary facets is "
657               << facets.size() << std::endl;
658     std::cout << "Number of singular edges is "
659               << edges.size() << std::endl;
660     std::cout << "Number of singular vertices is "
661               << vertices.size() << std::endl;
662     std::cout << "Boundary surface construction." << std::endl;
663 #endif
664
665     std::vector<As3::Facet> bfacets;
666     this->alphaShape->get_alpha_shape_facets(std::back_inserter(bfacets),
667       As3::REGULAR);
668
669     std::size_t nbf = bfacets.size();
670     std::vector<CGAL_Polygon> polygons;
671     CGAL_Polygon p;
672
673     for(std::size_t i = 0; i < nbf; i++){
674       if(this->alphaShape->classify(bfacets[i].first) != As3::EXTERIOR)
675         bfacets[i] = this->alphaShape->mirror_facet(bfacets[i]);
676
677       int32_t indices[3] = {
678         (bfacets[i].second + 1) % 4,
679         (bfacets[i].second + 2) % 4,
680         (bfacets[i].second + 3) % 4,
681       };
682
683       // Consistent orientation
684       if(bfacets[i].second % 2 == 0) std::swap(indices[0], indices[1]);
685
686       p.clear();
687       for(uint8_t j = 0; j < 3; j++){
688         p.push_back(bfacets[i].first->vertex(indices[j])->info());
689       }
690       polygons.push_back(p);
691     }
692
693     PMP::polygon_soup_to_polygon_mesh(this->Points,
694       polygons, this->surface_mesh);
695 }
696
697 // numRegions() function
698 // --------------------
699 double AlphaShape3D::numRegions(void) {
700 #ifdef DEBUG_ALPHA_SHAPE_3D
701   consoleLog(0, "Called numRegions()");
702 #endif
703
704   return this->alphaShape->number_of_solid_components();
705 }
```

```
706
707 // getAlphaSpectrum() function
708 // --------------------
709 Matrix AlphaShape3D::getAlphaSpectrum() {
710 #ifdef DEBUG_ALPHA_SHAPE_3D
711   consoleLog(0, "Called getAlphaSpectrum()");
712 #endif
713
714   Matrix a(1, this->numAlphaValues);
715   for(uint32_t i = 0; i < this->numAlphaValues; i++){
716     a(0, i) = this->alphaShape->get_nth_alpha(i + 1);
717   }
718   return a;
719 }
720
721 // getCriticalAlpha() function
722 // --------------------
723 double AlphaShape3D::getCriticalAlpha(std::string type) {
724 #ifdef DEBUG_ALPHA_SHAPE_3D
725   consoleLog(0, "Called getCriticalAlpha()");
726 #endif
727
728   if(type == "all-points"){
729     return this->alphaShape->find_alpha_solid();
730   }
731   else if(type == "one-region"){
732     return *this->alphaShape->find_optimal_alpha(1);
733   }
734   else{
735     return nan("");
736   }
737 }
738
739 // getSurfaceArea() function
740 // --------------------
741 double AlphaShape3D::getSurfaceArea(void) {
742 #ifdef DEBUG_ALPHA_SHAPE_3D
743   consoleLog(0, "Called getSurfaceArea()");
744 #endif
745
746   return PMP::area(this->surface_mesh);
747 }
748
749 // getVolume() function
750 // --------------------
751 double AlphaShape3D::getVolume(void) {
752 #ifdef DEBUG_ALPHA_SHAPE_3D
753   consoleLog(0, "Called getVolume()");
754 #endif
755
756   return PMP::volume(this->surface_mesh);
757 }
758
759 // getBoundaryFacets() function
760 // --------------------
761 Matrix AlphaShape3D::getBoundaryFacets(void) {
762 #ifdef DEBUG_ALPHA_SHAPE_3D
763   consoleLog(0, "Called getBoundaryFacets()");
764 #endif
```

```
765
766   Matrix bf(this->surface_mesh.number_of_faces(), 3);
767   for(Mesh::Face_index face_index : this->surface_mesh.faces()){
768     CGAL::Vertex_around_face_circulator<Mesh>
769       vcirc(this->surface_mesh.halfedge(face_index), this->surface_mesh);
770     bf(face_index.idx(), 0) = *vcirc++;
771     bf(face_index.idx(), 1) = *vcirc++;
772     bf(face_index.idx(), 2) = *vcirc++;
773   }
774   return bf;
775 }
776
777 // getBoundaryFacets() function with filename
778 // --------------------
779 Matrix AlphaShape3D::getBoundaryFacets(std::string filename) {
780   Matrix bf = this->getBoundaryFacets();
781   this->writeOff(filename, this->inputPoints, bf);
782   return bf;
783 }
784
785 // writeBoundaryFacets() function
786 // --------------------
787 void AlphaShape3D::writeBoundaryFacets(std::string filename) {
788 #ifdef DEBUG_ALPHA_SHAPE_3D
789   consoleLog(0, "Called writeBoundaryFacets()");
790 #endif
791
792   this->writeOff(filename, this->inputPoints, this->getBoundaryFacets());
793 }
794
795 // checkInShape() function
796 // --------------------
797 Matrix AlphaShape3D::checkInShape(Matrix QP) {
798 #ifdef DEBUG_ALPHA_SHAPE_3D
799   consoleLog(0, "Called checkInShape()");
800 #endif
801
802   Matrix tf(QP.numRows(), 1);
803   for(uint32_t i = 0; i < QP.numRows(); i++){
804     tf(i, 0) = this->alphaShape->classify(Point(QP(i, 0), QP(i, 1), QP(i, 2)));
805   }
806   return tf;
807 }
808
809 // getTriangulation() function
810 // --------------------
811 Matrix AlphaShape3D::getTriangulation(void) {
812 #ifdef DEBUG_ALPHA_SHAPE_3D
813   consoleLog(0, "Called getTriangulation()");
814 #endif
815
816   return this->triangulationMatrix;
817 }
818
819 // getNearestNeighbor() function
820 // --------------------
821 std::pair<Matrix, Matrix> AlphaShape3D::getNearestNeighbor(Matrix QP) {
822 #ifdef DEBUG_ALPHA_SHAPE_3D
823   consoleLog(0, "Called getNearestNeighbor()");
```

```
824 #endif
825
826   Matrix I(QP.numRows(), 1);
827   Matrix D(QP.numRows(), 1);
828
829   Mesh surface_mesh_s(this->surface_mesh);
830   PMP::remove_isolated_vertices(surface_mesh_s);
831
832   std::vector<Point> points;
833   std::vector<uint32_t> vs;
834   for(Mesh::Vertex_index i : vertices(surface_mesh_s)){
835     if(!surface_mesh_s.is_removed(i)){
836       points.push_back(surface_mesh_s.point(i));
837       vs.push_back(i);
838     }
839   }
840
841   search_map map(points);
842   Tree tree(boost::counting_iterator<std::size_t>(0),
843     boost::counting_iterator<std::size_t>(
844     vertices(surface_mesh_s).size()),
845     Tree::Splitter(), Traits(map));
846   K_neighbor_search::Distance tr_dist(map);
847
848   for(uint32_t i = 0; i < QP.numRows(); i++){
849     K_neighbor_search search(tree, Point(QP(i, 0), QP(i, 1), QP(i, 2)),
850       1, 0, true, tr_dist);
851     I(i, 0) = vs[search.begin()->first];
852     D(i, 0) =
853       tr_dist.inverse_of_transformed_distance(search.begin()->second);
854   }
855   return std::make_pair(I, D);
856 }
857
858 // getSimplifiedShape() function with stop_ratio
859 // --------------------
860 std::pair<Matrix, Matrix>
861     AlphaShape3D::getSimplifiedShape(double stop_ratio) {
862 #ifdef DEBUG_ALPHA_SHAPE_3D
863   consoleLog(0, "Called getSimplifiedShape()");
864 #endif
865
866   Mesh surface_mesh_s(this->surface_mesh);
867   SMS::Edge_count_ratio_stop_predicate<Mesh> stop(stop_ratio);
868   uint32_t r = SMS::edge_collapse(surface_mesh_s, stop);
869   PMP::remove_isolated_vertices(surface_mesh_s);
870   surface_mesh_s.collect_garbage();
871
872 #ifdef DEBUG_ALPHA_SHAPE_3D
873   std::cout << "Number of edges removed is " << r << std::endl
874             << "Number of  final edges is "
875             << surface_mesh_s.number_of_edges() << std::endl;
876 #endif
877
878   Matrix Points(surface_mesh_s.number_of_vertices(), 3);
879   Matrix bf(surface_mesh_s.number_of_faces(), 3);
880
881   for(Mesh::Vertex_index vertex_index : surface_mesh_s.vertices()){
882     Point p = surface_mesh_s.point(vertex_index);
```

```
883       Points(vertex_index.idx(), 0) = p[0];
884       Points(vertex_index.idx(), 1) = p[1];
885       Points(vertex_index.idx(), 2) = p[2];
886     }
887
888     for(Mesh::Face_index face_index : surface_mesh_s.faces()){
889       CGAL::Vertex_around_face_circulator<Mesh>
890         vcirc(surface_mesh_s.halfedge(face_index), this->surface_mesh);
891       bf(face_index.idx(), 0) = *vcirc++;
892       bf(face_index.idx(), 1) = *vcirc++;
893       bf(face_index.idx(), 2) = *vcirc++;
894     }
895     return std::make_pair(Points, bf);
896   }
897
898   // getSimplifiedShape() function without parameters
899   // --------------------
900   std::pair<Matrix, Matrix>
901       AlphaShape3D::getSimplifiedShape() {
902     double stop_ratio = 0.05;
903     return this->getSimplifiedShape(stop_ratio);
904   }
905
906   // getSimplifiedShape() function with filename
907   // --------------------
908   std::pair<Matrix, Matrix>
909       AlphaShape3D::getSimplifiedShape(std::string filename) {
910     double stop_ratio = 0.05;
911     std::pair<Matrix, Matrix> ret = this->getSimplifiedShape(stop_ratio);
912     this->writeOff(filename, ret.first, ret.second);
913     return ret;
914   }
915
916   // getSimplifiedShape() function with stop_ratio and filename
917   // --------------------
918   std::pair<Matrix, Matrix>
919       AlphaShape3D::getSimplifiedShape(double stop_ratio,
920       std::string filename) {
921     std::pair<Matrix, Matrix> ret = this->getSimplifiedShape(stop_ratio);
922     this->writeOff(filename, ret.first, ret.second);
923     return ret;
924   }
925
926   // removeUnusedPoints() function
927   // --------------------
928   std::pair<Matrix, Matrix>
929       AlphaShape3D::removeUnusedPoints(Matrix Pi, Matrix bfi) {
930   #ifdef DEBUG_ALPHA_SHAPE_3D
931     consoleLog(0, "Called removeUnusedPoints()");
932   #endif
933
934     Mesh surface_mesh_s;
935     std::vector<CGAL_Polygon> polygons;
936     CGAL_Polygon p;
937     std::vector<Point> points;
938
939   #ifdef DEBUG_ALPHA_SHAPE_3D
940     std::cout << "Boundary surface reconstruction. " << std::endl;
941   #endif
```

```
942
943   uint32_t n = Pi.numRows();
944   uint32_t nbf = bfi.numRows();
945
946   for(std::size_t i = 0; i < n; i++){
947     points.push_back(Point(Pi(i, 0), Pi(i, 1), Pi(i, 2)));
948   }
949
950   for(std::size_t i = 0; i < nbf; i++){
951     p.clear();
952     for(uint8_t j = 0; j < 3; j++){
953       p.push_back(bfi(i, j));
954     }
955     polygons.push_back(p);
956   }
957
958   PMP::orient_polygon_soup(points, polygons);
959   PMP::repair_polygon_soup(points, polygons);
960   PMP::polygon_soup_to_polygon_mesh(points,
961     polygons, surface_mesh_s);
962   surface_mesh_s.collect_garbage();
963
964   Matrix Points(surface_mesh_s.number_of_vertices(), 3);
965   Matrix bf(surface_mesh_s.number_of_faces(), 3);
966
967   for(Mesh::Vertex_index vertex_index : surface_mesh_s.vertices()){
968     Point p = surface_mesh_s.point(vertex_index);
969     Points(vertex_index.idx(), 0) = p[0];
970     Points(vertex_index.idx(), 1) = p[1];
971     Points(vertex_index.idx(), 2) = p[2];
972   }
973
974   for(Mesh::Face_index face_index : surface_mesh_s.faces()){
975     CGAL::Vertex_around_face_circulator<Mesh>
976       vcirc(surface_mesh_s.halfedge(face_index), this->surface_mesh);
977     bf(face_index.idx(), 0) = *vcirc++;
978     bf(face_index.idx(), 1) = *vcirc++;
979     bf(face_index.idx(), 2) = *vcirc++;
980   }
981   return std::make_pair(Points, bf);
982 }
983
984 // writeOff() function
985 // --------------------
986 void AlphaShape3D::writeOff(std::string filename, Matrix Points, Matrix bf) {
987 #ifdef DEBUG_ALPHA_SHAPE_3D
988   consoleLog(0, "Called writeOff()");
989 #endif
990
991   uint32_t n = Points.numRows();
992   uint32_t nbf = bf.numRows();
993
994   std::stringstream pts;
995   std::stringstream ind;
996
997   for(std::size_t i = 0; i < n; i++){
998     pts << Points(i, 0) << " " << Points(i, 1) << " " << Points(i, 2) << std::
        endl;
999   }
```

```
1000
1001    for(std::size_t i = 0; i < nbf; i++){
1002      ind << "3 " << (uint64_t)bf(i, 0) << " " << (uint64_t)bf(i, 1)
1003        << " " << (uint64_t)bf(i, 2) << std::endl;
1004    }
1005
1006    std::ofstream of(filename);
1007    CGAL::set_ascii_mode(of);
1008    of << "OFF" << std::endl << n << " " << nbf << " 0" << std::endl;
1009    of << pts.str();
1010    of << ind.str();
1011    of.close();
1012  }
1013
1014  Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
1015    return AlphaShape3D::Init(env, exports);
1016  }
1017
1018  NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
1019
1020  }  // namespace alpha_shape_3d_ns
```

Listing L.4 - `alpha-shape-3d.cpp`

M

Skripta leversMinDistance.jsl

Skriptom `leversMinDistance.jsl`, koja je u celosti data u listingu M.1, se vrši provera rastojanja između diskova koje svojim rotacijama definišu donje poluge. U podglavlju 4.11 je definisan algoritam funkcije `disksDistance3D()` koja se ovde koristi.

```
/**
 * @file Solve Inverse Kinematics
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x = [ // translaWcija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  0 // [deg] psi
];
const N_angular = 201; // Number of angular points
const N_r = 21; // Number of radial points
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

var { MP_6RSS_PLOTTER } = require('./classes/PLOTTER_MP_6RSS.jsl');
var plotter = new MP_6RSS_PLOTTER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
solver.getGeometry();

// Odredjivanje potrebnog ugla alpha za date podatke
disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
solver.setParameters(x);
```

```
40
41 // Provera uglova u zglobovim
42 disp(' Provera uglova u zglobovima...');
43 solver.checkJoints();
44
45 function disksDistance3D(O1, n1, r1, O2, n2, r2) {
46   let L_min = NaN;
47   let compute_case = 0;
48
49   // 1) Intersect or not intersect planes
50   const { P, i, flag } = planesIntersection(O1, n1, O2, n2);
51
52   if(flag == 0) {
53     // The planes do intersect in a line (P, i)
54     const { d: d1, P1: P1_ } = pointLineDistance(O2, P, i);
55     const { d: d2, P1: P2_ } = pointLineDistance(O1, P, i);
56
57     // Intersect disk 2 & plane 1
58     const { P1: P3, P2: P4, flag: flag1 } = lineCircleIntersection(P, i, O2, r2)
          ;
59
60     if(flag1 > 1) {
61       // No intersection between disk 2 and plane 1
62       if(
63         abs(r1 - r2) <= EPS &&
64         norm(minus(P1_, P2_)) <= EPS &&
65         abs(norm(minus(P1_, O1)) - norm(minus(P2_, O2))) <= EPS
66       ) {
67         // Symmetric disks
68         const Pc = P1_; // same as P2_, presumably
69         const i_O1Pc = scale(minus(Pc, O1), 1.0 / norm(minus(Pc, O1)));
70         const i_O2Pc = scale(minus(Pc, O2), 1.0 / norm(minus(Pc, O2)));
71
72         const Pc1 = plus(O1, scale(i_O1Pc, r1));
73         const Pc2 = plus(O2, scale(i_O2Pc, r2));
74         L_min = norm(minus(Pc1, Pc2));
75         compute_case = 1;
76       } else {
77         compute_case = 2;
78       }
79     } else {
80       // Disk 2 intersects or touches the line
81       const { P1: P5, P2: P6, flag: flag2 } = lineCircleIntersection(P, i, O1,
            r1);
82       if(flag2 > 1) {
83         // No intersection for disk 1 & plane 2
84         compute_case = 3;
85       } else {
86         // Both disks might intersect or touch or not
87         if(!flag1 && !flag2) {
88           // Both intersect => possible overlap check
89           const { A, B, flag: f_e, id1, id2 } = linesOverlap(P3, P4, P5, P6);
90           if(!f_e) {
91             // There's an overlap
92             // Original code sets L_min = d1-r1 + d2-r2
93             L_min = (d1 - r1) + (d2 - r2);
94             compute_case = 4;
95           } else {
96             // No overlap of the arcs => no intersection
```

```
            compute_case = 5;
          }
        } else if(!flag1 && flag2 == 1) {
          // Disk1 is tangent => check if tangent point is on the other arc
          if(isPointOnLine(P5, P3, P4)) {
            // Disks are touching
            L_min = 0;
            compute_case = 6;
          } else {
            // No intersection
            compute_case = 7;
          }
        } else if(!flag2 && flag1 == 1) {
          if(isPointOnLine(P3, P5, P6)) {
            // Disks are touching
            L_min = 0;
            compute_case = 8;
          } else {
            // No intersection
            compute_case = 9;
          }
        } else if(flag2 && flag1 == 1) {
          if(norm(minus(P3, P5)) <= EPS) {
            // Disks are touching
            L_min = 0;
            compute_case = 10;
          } else {
            // No intersection
            compute_case = 11;
          }
        }
      }
    }
  } else if(flag == 1) {
    // The planes are the same => treat this as a 2D disk/disk intersection
    const d = norm(minus(O2, O1));
    if(d < (r1 + r2)) {
      if(abs(r1 - r2) < EPS && d <= EPS) {
        // same disk, same location, same radius
        compute_case = 12;
      } else if(r1 > r2 && d < (r1 - r2)) {
        // disk2 inside disk1
        compute_case = 13;
      } else if(r2 > r1 && d < (r2 - r1)) {
        // disk1 inside disk2
        compute_case = 14;
      } else {
        // intersection of two disks
        compute_case = 15;
      }
      L_min = d - (r1 + r2);
    } else if(abs(d - (r1 + r2)) <= EPS) {
      // disks are touching
      L_min = 0;
      compute_case = 16;
    } else {
      // no intersection
      L_min = d - (r1 + r2);
      compute_case = 17;
```

```
    }
  } else {
    // flag == 2 -> parallel planes
    const d1 = norm(minus(O2, O1)); // distance between centers
    // project (O2 - O1) onto n1
    const proj = scale(n1, dot(minus(O2, O1), n1) / dot(n1, n1));
    const d2 = norm(proj); // distance between planes
    // distance in the plane = sqrt(d1^2 - d2^2)
    const d = sqrt(d1 * d1 - d2 * d2);

    if(d <= (r1 + r2)) {
      // one disk above the other
      L_min = d2;
      compute_case = 18;
    } else {
      // side by side
      const dist = (d - (r1 + r2));
      L_min = sqrt(dist * dist + d2 * d2);
      compute_case = 19;
    }
  }
  return { L_min, compute_case };
}

if(any(MP.e)) {
  disp(' Neuspešan proračun!');
  dispRowVector('x', MP.x, MP.m);
  dispRowVector('e', MP.e, MP.m);
} else {
  disp(' Uspešan proračun!');
  dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);

  disp(' Prikaz mehanizma...');

  plotter.s = 1.5;
  plotter.plotMechanism();

  var i_m = reshape(MP.i_m, 3, 6);
  var B = reshape(MP.B, 3, 6);

  var r = MP.a;
  var O1 = [B[0][1], B[1][1], B[2][1]];
  var O2 = [
    [B[0][0], B[1][0], B[2][0]],
    [B[0][2], B[1][2], B[2][2]]
  ];
  var n1 = [i_m[0][1], i_m[1][1], i_m[2][1]];
  var n2 = [
    [i_m[0][0], i_m[1][0], i_m[2][0]],
    [i_m[0][2], i_m[1][2], i_m[2][2]]
  ];

  var p_n1 = createVectors3D(O1[0], O1[1], O1[2], n1[0], n1[1], n1[2],
    0.3, 0.4, {width: 6, color: '#00b'});4
  var p_D1 = createDisks3D(O1[0], O1[1], O1[2], r, n1[0], n1[1], n1[2],
    {width: 6, color: '#00b'});
  p_D1.line.width = 6;
  p_D1.area.opacity = 0.2;

```

```
215   plotter.traces.push(p_n1.line);
216   plotter.traces.push(p_n1.head);
217   plotter.traces.push(p_D1.line);
218   plotter.traces.push(p_D1.area);
219
220   var D1 = disk(r, N_angular, N_r);
221   var R1 = getRotationMatrix([0, 0, 1], normalizeVector(n1));
222   D1 = transform(D1, 1, R1, O1);
223
224   for(var k = 0; k < n2.length; k++) {
225     var p_n2 = createVectors3D(O2[k][0], O2[k][1], O2[k][2],
226       n2[k][0], n2[k][1], n2[k][2], 0.3, 0.4, {width: 6, color: '#0b0'});
227     var p_D2 = createDisks3D(O2[k][0], O2[k][1], O2[k][2], r,
228       n2[k][0], n2[k][1], n2[k][2], {width: 6, color: '#0b0'});
229     p_D2.line.width = 6;
230     p_D2.area.opacity = 0.2;
231
232     plotter.traces.push(p_n2.line);
233     plotter.traces.push(p_n2.head);
234     plotter.traces.push(p_D2.line);
235     plotter.traces.push(p_D2.area);
236
237     var D2 = disk(r, N_angular, N_r);
238     var R2 = getRotationMatrix([0, 0, 1], normalizeVector(n2[k]));
239     D2 = transform(D2, 1, R2, O2[k]);
240
241     var { L: L_min_num, P1, P2} = minPointsDistance3D(D1, D2);
242
243     plotter.traces.push(createLines3D(P1[0], P1[1], P1[2],
244        P2[0], P2[1], P2[2], {width: 6, color: '#f00'}));
245     plotter.traces.push(createPoints3D([P1[0], P2[0]], [P1[1], P2[1]],
246       [P1[2], P2[2]], {width: 6, color: '#f00'}));
247
248     var { L_min, compute_case } =
249       disksDistance3D(O1, n1, r, O2[k], n2[k], r);
250
251     disp(' -------------------- ');
252     disp(' Proračun ' + (k+1));
253     disp(' Analitičko rastojanje L_min = ' + L_min);
254     disp(' Numeričko rastojanje L_min = ' + L_min_num);
255     disp(' Slučaj proračuna = ' + compute_case);
256
257     if(isNaN(L_min)) {
258       disp(' Polozaj poluga nije implementiran proracunom!');
259     } else if(L_min < MP.dr_min) {
260       disp(' Rastojanje manje od dozvoljenog!');
261     }
262   }
263   plotter.show();
264 }
265
266 // - Kraj proračuna
267 disp(' Program je uspešno izvršen... ');
268 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
269 disp(' -------------------- ');
```

Listing M.1 - `leversMinDistance.jsl`

N

Metoda checkJointsFixed

Metodom `checkJointsFixed()`, koja je u celosti data u listingu N.1, se vrši provera da li uglovi u zglobovima premašuju dozvoljene ali tako da se u obzir uzima i fiksni ugao ugradnje sfernih zglobova gornjih poluga, prema algoritmu prikazanom u poglavlju 4.10.

```
checkJointsFixed() {
  var MP = this.MP;
  var m = MP.m;

  if(MP.IK_FLAG) {
    for(var k = 0; k < 3 * m; k++) {
      MP.PbA[k] = MP.A[k] - MP.Pb[k];
    }
    for(var j = 0; j < m; j++) {
      MP.lambda_p[j] = acos(
        (MP.PbA[j * 3] * MP.i_jpb[j * 3] +
         MP.PbA[j * 3 + 1] * MP.i_jpb[j * 3 + 1] +
         MP.PbA[j * 3 + 2] * MP.i_jpb[j * 3 + 2]) / MP.s
      ) * r2d - 90.0;
      MP.lambda_b[j] = acos(
        (MP.PbA[j * 3] * MP.i_ll[j * 3] +
         MP.PbA[j * 3 + 1] * MP.i_ll[j * 3 + 1] +
         MP.PbA[j * 3 + 2] * MP.i_ll[j * 3 + 2]) / MP.s
      ) * r2d - 90.0;
      if(abs(MP.lambda_b[j]) > MP.lambda_max) {
        MP.e[j] = this.state['LAMBDA_B_ERROR'];
      } else if(abs(MP.lambda_p[j]) > MP.lambda_max) {
        MP.e[j] = this.state['LAMBDA_P_ERROR'];
      }
    }

    if(isNaN(MP.mu_ul)) return;

    var i_jpb_p = minus(MP.i_jpb,
      multiplyEl(repRow(sumCol(multiplyEl(MP.i_jpb, MP.i_ul), 3, MP.m), 3), MP.
          i_ul));
    i_jpb_p = divideEl(i_jpb_p, repRow(elementWise((a) => sqrt(a),
      sumCol(elementWise((b) => pow(b, 2), i_jpb_p), 3, MP.m)), 3));
    var i_ll_p = minus(MP.i_ll,
      multiplyEl(repRow(sumCol(multiplyEl(MP.i_ll, MP.i_ul), 3, MP.m), 3), MP.
          i_ul));
    i_ll_p = divideEl(i_ll_p, repRow(elementWise((a) => sqrt(a),
      sumCol(elementWise((b) => pow(b, 2), i_ll_p), 3, MP.m)), 3));
    var lambda_pj = elementWise((a) => abs(acosd(a)-90),
```

```
38        sumCol(multiplyEl(MP.i_ul, MP.i_jpb), 3, MP.m));
39      var mu_pj_max = elementWise((a) => acosd(cosd(MP.lambda_max)/cosd(a)),
          lambda_pj);
40      var lambda_bj = elementWise((a) => abs(acosd(a)-90),
41        sumCol(multiplyEl(MP.i_ul, MP.i_ll), 3, MP.m));
42      var mu_bj_max = elementWise((a) => acosd(cosd(MP.lambda_max)/cosd(a)),
          lambda_bj);
43      var mu_ul_opt = multiplyEl(elementWise((a, b) => atan2d(a, b),
44        sumCol(multiplyEl(cross3D(i_jpb_p, i_ll_p, MP.m), MP.i_ul), 3, MP.m),
45        sumCol(multiplyEl(i_jpb_p, i_ll_p, MP.m), 3, MP.m)), MP.sigma_l);
46      var mu_ul_max = plus(mu_ul_opt, mu_pj_max, mu_bj_max);
47      var mu_ul_min = minus(mu_ul_opt, mu_pj_max, mu_bj_max);
48
49      setSubB(MP.e, elementWise((a, b) =>
50        a < MP.mu_ul || b > MP.mu_ul, mu_ul_max, mu_ul_min),
51        this.state['MU_UL_ERROR']);
52    }
53  }
```

Listing N.1 - `checkJoints()` za fiksni ugao ugradnje sfernih zglobova

O

Skripta analyticStructuralAnalysis.jsl

Skriptom `analyticStructuralAnalysis.jsl`, koja je u celosti data u listingu O.1, se vrši analitička provera čvrstoće poluga Stjuartovog mehanizma sa obrtnim aktuatorima. U poglavlju 4.12 je definisan algoritam koji se ovde koristi.

```
/**
 * @file Analytic structural analysis of Rotary Stewart Platform 6RUS levers
 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');

// Ulazni podaci
var F = 4448; // [N] opterećenje
var s = 130;  // [cm] duzina gornje poluge
var a = 25;   // [cm] duzina donje poluge
var D_ll = 6;  // [cm]  precnik cevi donje poluge
var delta_ll = 1; // [cm] debljina zida cevi donje poluge
var D_ul = 5.1;    // [cm] precnik cevi gornje poluge
var delta_ul = 0.8; // [cm] debljina zida cevi gornje poluge
var alfa = 15; // [deg] ugao napadne linije sile za donju polugu

// Materijal S325
var sigma_max = 3250; // [daN/cm^2] Maksimalni dozvoljeni normalni napon
var E = 2100000; // [daN/cm^2] Modul elastičnosti

// - Donja poluga

// Karakteristike poprečnog preseka
var d_ll = D_ll - 2 * delta_ll; // [cm]
var A_ll = (pow(D_ll, 2) - pow(d_ll, 2)) * pi / 4; // [cm^2]
var Sx_ll = (pow(D_ll, 3) - pow(d_ll, 3)) / 12; // [cm^3]
var Ix_ll = (pow(D_ll, 4) - pow(d_ll, 4)) * pi / 64; // [cm^4]
var Wx_ll = (pow(D_ll, 4) - pow(d_ll, 4)) * pi / (32 * D_ll); // [cm^3]

// Proračun napona
var T = F / 10 * cosd(alfa); // [daN]
var N = F / 10 * sind(alfa); // [daN]
var M = a * T;  // [daNcm]
var sigma_ll = N / A_ll + M / Wx_ll;
var tau_ll = (T * Sx_ll) / (Ix_ll * (D_ll - d_ll)/2);
```

```
var sigma_ll_e = sigma_ll / 2 + sqrt(pow(sigma_ll, 2) + 4 * pow(tau_ll, 2)) / 2;
var s_ll = sigma_max / sigma_ll_e;

// Prikaz rezultata
disp(" --------------------");
disp(" Donja poluga");
disp(` Maksimalni ekvivalenti napon: ${sigma_ll_e.toFixed(2)} MPa.`);
disp(` Maksimalni tangencijalni napon: ${tau_ll.toFixed(2)} MPa.`);
disp(` Stepen sigurnosti: ${s_ll.toFixed(2)}.`);
if(s_ll < 1) {
  disp(" Stepen sigurnosti nije zadovoljavajuć!");
}

var x = [0, 0, a, a];
var data = await plotBeamDiagrams([
  {
    x: x,
    y: [0, N, N, 0],
    xlabel: ['x', 'cm'],
    ylabel: ['N', 'daN'],
    title: 'Aksijana sila'
  },
  {
    x: x,
    y: [0, T, T, 0],
    xlabel: ['x', 'cm'],
    ylabel: ['T', 'daN'],
    title: 'Transverzalna sila'
  },
  {
    x: x,
    y: [0, T * a, 0, 0],
    xlabel: ['x', 'cm'],
    ylabel: ['Ms', 'daNcm'],
    title: 'Moment savijanja'
  },
], {
  font: {family: 'LatinModern', size: 18}
});
data.extrems.forEach(row => disp(row));
data.context.setTitle('Donja poluga dijagrami');

// - Gornja poluga

// Karakteristike poprečnog preseka
var d_ul = D_ul - 2 * delta_ul;
var A_ul = (pow(D_ul, 2) - pow(d_ul, 2)) * pi / 4; // [cm^2]
var i_ul = sqrt(pow(D_ul, 2) + pow(d_ul, 2)) / 4;    // [cm]
var lambda_ul = s / i_ul;

// Proračun napona
var N = F / 10; // [daN]
var lambda_p = pi * sqrt(E / sigma_max);
var sigma_ul = N / A_ul; // [daN/cm^2]

let sigma_kr;
if(lambda_ul >= lambda_p) {
  sigma_kr = pow(pi, 2) * E / pow(lambda_ul, 2);
} else {
```

```
99    error('Nije implementiran slučaj!');
100 }
101
102 // Prikaz rezultata
103 disp(" --------------------");
104 disp(" Gornja poluga");
105 disp(` Kriticni napon: ${sigma_kr.toFixed(2)} MPa.`);
106 disp(` Maksimalni napon: ${sigma_ul.toFixed(2)} MPa.`);
107 disp(` Stepen sigurnosti: ${(sigma_kr / sigma_ul).toFixed(2)}.`);
108 if ((sigma_kr / sigma_ul) < 1) {
109   disp(" Stepen sigurnosti nije zadovoljavajuć!");
110 }
111
112 var x = [0, 0, s, s];
113 var data = await plotBeamDiagrams([
114   {
115     x: x,
116     y: [0, N, N, 0],
117     xlabel: ['x', 'cm'],
118     ylabel: ['N', 'daN'],
119     title: 'Aksijana sila'
120   },
121   {
122     x: x,
123     y: [0, 0, 0, 0],
124     xlabel: ['x', 'cm'],
125     ylabel: ['T', 'daN'],
126     title: 'Transverzalna sila'
127   },
128   {
129     x: x,
130     y: [0, 0, 0, 0],
131     xlabel: ['x', 'cm'],
132     ylabel: ['Ms', 'daNcm'],
133     title: 'Moment savijanja'
134   },
135 ], {
136   font: {family: 'LatinModern', size: 18}
137 });
138 data.extrems.forEach(row => disp(row));
139 data.context.setTitle('Gornja poluga dijagrami');
140 disp(" --------------------");
141
142 // End of program
143 disp(' The program was executed successfully... ');
144 disp(' Execution time: ' + num2str(toc(), 2) + ' seconds');
145 disp(' -------------------- ');
```

Listing O.1 - analyticStructuralAnalysis.jsl

Nakon izvršavanja je u komandnom prozoru ispisan sledeći tekst:

```
1  --- analyticStructuralAnalysis.jsl ---
2  --------------------
3  Donja poluga
4  Maksimalni ekvivalenti napon: 655.86 MPa.
5  Maksimalni tangencijalni napon: 106.60 MPa.
6  Stepen sigurnosti: 4.96.
7  Aksijana sila: N_max = 115.12 daN, x = 0.00 cm
```

```
Transverzalna sila: T_max = 429.64 daN, x = 0.00 cm
Moment savijanja: Ms_max = 10741.10 daNcm, x = 0.00 cm
--------------------
Gornja poluga
Kriticni napon: 2932.63 MPa.
Maksimalni napon: 41.16 MPa.
Stepen sigurnosti: 71.25.
Aksijana sila: N_max = 444.80 daN, x = 0.00 cm
Transverzalna sila: T_max = 0.00 daN, x = 0.00 cm
Moment savijanja: Ms_max = 0.00 daNcm, x = 0.00 cm
--------------------
The program was executed successfully...
Execution time: 1.05 seconds
--------------------
```

P

JSLAB klasa OPTIM_RCMIGA

Klasa `OPTIM_RCMIGA`, napisana u programu `JSLAB`, u okviru koje je implementiran *realno kodirani mešovito celobrojni genetski algoritam sa ograničenjima*, detaljno opisan u okviru poglavlja 5, je u celosti data u listingu P.1.

U listingu P.2 je dat primer za upotrebu ove klase i u njoj implementiranog algoritma.

```
/**
 * @file Real Coded Mixed Integer Genetic Algorithm
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

/**
 * Class for Real Coded Mixed Integer Genetic Algorithm - RCMIGA.
 */
class OPTIM_RCMIGA {

  /**
   * Creates an instance of OPTIM_RCMIGA.
   * @param {Object} problem - The optimization problem definition.
   * @param {Object} opts - Configuration options for the algorithm.
   */
  constructor(problem, opts) {
    this.flag = 'preinit';
    this.problem = problem;
    this.opts = opts;
    this.state = {};
    this.selection = {};
    this.solution = {};
    this.constrained = problem.constrained;
    this.stoped = 0;

    this.checkInputs();
    if(this.bounded) {
      this.opts.lbm = repCol(this.opts.lb, opts.PopulationSize);
      this.opts.ubm = repCol(this.opts.ub, opts.PopulationSize);
    }

    this.opts.a = opts.a ?? 0;
    this.opts.b_real = opts.b_real ?? 0.15;
    this.opts.b_integer = opts.b_integer ?? 0.35;
    this.opts.p_real = opts.p_real ?? 10;
```

```
39      this.opts.p_integer = opts.p_integer ?? 4;
40      this.opts.UseVectorized = opts.UseVectorized ?? false;
41      this.opts.UseParallel = opts.UseParallel ?? false;
42      this.opts.Display = opts.Display ?? 'iter';
43
44      // Inicijalizacija
45      this.flag = 'init';
46      this.outputFcn();
47      this.state.StartTime = tic;
48      this.state.StallTime = tic;
49      this.state.StallGenerations = 1;
50      this.state.RandSeed = 'rcmiga';
51      this.state.Generation = 0;
52
53      this.rand = seedRandom(this.state.RandSeed);
54    }
55
56    /**
57     * Executes the optimization process.
58     */
59    async run(parallel_context, parallel_setup_fun) {
60      if(this.opts.UseParallel) {
61        this.parallel_context = parallel_context;
62        this.parallel_setup_fun = parallel_setup_fun;
63        parallel.terminate();
64      }
65
66      this.initState();
67      this.initPopulation();
68      await this.updateState();
69
70      this.flag = 'iter';
71      this.outputFcn();
72      this.createPlotFcn();
73      this.plotFcn();
74
75      // Optimizacija
76      var nvars = this.problem.nvars;
77      var newPopulation = zeros(nvars * this.opts.PopulationSize);
78      var stop = 0;
79      while(!stop) { // Petlja za optimizaciju
80        // Merenje vremena
81        this.state.GenTime = tic;
82
83        // Elitizam
84        if(this.state.ReproductionCount.elite > 0) {
85          var [S, I] = sorti(this.state.FunVal);
86          this.selection.current_elite = getSub(I, this.selection.elite);
87          setSub(newPopulation, index(range(0, nvars - 1),
88            this.selection.elite, nvars), getSub(this.state.Population,
89            index(range(0, nvars - 1), this.selection.current_elite, nvars)));
90        }
91
92        //  Selekcija
93        this.state.parents = this.selectionFcn();
94
95        // Ukrstanje
96        setSub(newPopulation, index(range(0, nvars - 1),
97          this.selection.children, nvars), this.crossoverFcn());
```

```
98
99        // Mutacija
100       setSub(newPopulation, index(range(0, nvars - 1),
101         this.selection.mutants, nvars), this.mutationFcn());
102
103       // Provera granica
104       this.state.Population = [...newPopulation];
105       this.integerRestriction();
106       if(this.bounded) {
107         this.checkBounds();
108       }
109
110       // Odredjivanje vrednosti funkcije prilagodjenosti i ogranicenja
111       this.state.Generation = this.state.Generation + 1;
112       await this.updateState();
113
114       // Prikaz stanja
115       stop = this.stoppingCriteria(); // Kriterijumi zaustavljanja
116       this.outputFcn();
117       this.plotFcn();
118       await waitMSeconds(5);
119     }
120
121     this.solution.generations = this.state.Generation;
122     var [x_min, I] = mini(this.state.FunVal);
123     this.solution.feasible = 1;
124     if(this.constrained && this.state.ConSumVal[I] > 0) {
125       this.solution.feasible = 0;
126     }
127     this.solution.x = getSub(this.state.Population,
128       index(range(0, nvars - 1), I, nvars));
129     this.solution.fval = this.problem.fitnessfcn(this.solution.x);
130     this.solution.StoppingCriteria = stop;
131
132     this.flag = 'done';
133     this.outputFcn();
134   }
135
136   /**
137    * Validates the input parameters and options.
138    */
139   checkInputs() {
140     if(typeof this.problem.nonlconfcn === 'function') {
141       this.constrained = 1;
142     }
143     if(typeof this.problem.IntCon === 'undefined') {
144       this.problem.IntCon = [];
145     }
146
147     this.bounded = true;
148     if(typeof this.opts.lb === 'undefined' || typeof this.opts.ub === 'undefined
          ') {
149       this.bounded = false;
150       if(typeof this.opts.InitalUnboundedRange === 'undefined') {
151         this.opts.InitalUnboundedRange = createFilledArray(this.problem.nvars,
              [0, 1]);
152       }
153     } else if((this.problem.lb && this.problem.lb.length != this.problem.nvars)
          ||
```

```
154         (this.problem.ub && this.problem.ub.length != this.problem.nvars)) {
155       throw new Error('Problem bounds have invalid dimension!');
156     }
157
158     if(this.opts.UseVectorized) {
159       if(this.opts.UseParallel == true){
160         disp('Option UseParallel is ignored while option UseVectorized is true.'
              );
161       } else if (this.opts.UseParallel != false){
162         disp('Option UseParallel must be true or false.');
163       }
164     } else {
165       if(![true, false].includes(this.opts.UseParallel)) {
166         throw new Error('Option UseParallel must be true or false!');
167       }
168     }
169   }
170
171   /**
172    * Creates the initial population.
173    */
174   creationFcn() {
175     return this.creationMixedUniform();
176   }
177
178   /**
179    * Selects individuals from the population.
180    */
181   selectionFcn() {
182     return this.binaryTournamentSelection();
183   }
184
185   /**
186    * Performs crossover between selected parents.
187    */
188   crossoverFcn() {
189     return this.laplaceMixedCrossover();
190   }
191
192   /**
193    * Mutates individuals in the population.
194    */
195   mutationFcn() {
196     return this.powerMixedMutation();
197   }
198
199   /**
200    * Initializes the plotting function.
201    */
202   createPlotFcn() {
203
204   }
205
206   /**
207    * Updates the graphical plot with current optimization status.
208    */
209   plotFcn() {
210     this.displayPlot();
211   }
```

```
212
213   /**
214    * Renders the optimization plot.
215    */
216   displayPlot() {
217
218   }
219
220   /**
221    * Handles button events to stop the optimization process.
222    */
223   buttonCallback() {
224     this.stoped = 1;
225   }
226
227   /**
228    * Manages the output display based on configuration.
229    */
230   outputFcn() {
231     if(['iter', 'final'].includes(this.opts.Display)) {
232       this.displayOutput();
233     }
234   }
235
236   /**
237    * Displays the current state of optimization.
238    */
239   displayOutput() {
240     if(this.opts.Display === 'iter') {
241       switch(this.flag) {
242         case 'init':
243           disp(' Optimization is initialized!');
244           if(this.opts.UseVectorized) {
245             disp(' Vectorized functions evaluation in use.');
246           } else if(this.opts.UseParallel){
247             disp(' Parallel functions evaluation in use.');
248           }
249           break;
250         case 'iter':
251           if(this.state.Generation === 0 || this.state.Generation % 20 === 0) {
252             if(this.constrained) {
253               dispMonospaced('\n                          Best        Max
                                 Stall');
254               dispMonospaced('   Generation           f(x)     Constraint
                                  Generations');
255             } else {
256               dispMonospaced('\n                          Best       Stall');
257               dispMonospaced('   Generation           f(x)    Generations');
258             }
259           }
260
261           let fval_s = num2str(this.state.Best[this.state.Generation], 5);
262           if(this.constrained) {
263             if(!this.state.Feasible[this.state.Generation]) {
264               fval_s += '*';
265             }
266             dispMonospaced(
267               `   ${num2str(this.state.Generation, 0).padStart(10)}   ${
268               fval_s.padStart(12)}   ${
```

```
269               num2str(max(this.state.ConSumVal), 2).padStart(12)}   ${
270               num2str(this.state.StallGenerations, 0).padStart(12)}`
271             );
272           } else {
273             dispMonospaced(
274               `   ${num2str(this.state.Generation, 0).padStart(10)}   ${
275               fval_s.padStart(12)}   ${
276               num2str(this.state.StallGenerations, 0).padStart(12)}`
277             );
278           }
279           break;
280         case 'done':
281           var str = '\nOptimization is done';
282           if(this.constrained) {
283             if(this.state.Feasible[this.state.Generation]) {
284               str += ', solution found';
285             } else {
286               str += ', no feasible solution found';
287             }
288           }
289           disp(` ${str}, stopping criteria = ${this.solution.StoppingCriteria}!\
                n`);
290           break;
291       }
292     } else if(this.opts.Display === 'final') {
293       if(this.flag === 'done') {
294         var str = '\nOptimization is done';
295         if(this.constrained) {
296           if(this.state.Feasible[this.state.Generation]) {
297             str += ', solution found';
298           } else {
299             str += ', no feasible solution found';
300           }
301         }
302         disp(` ${str}, stopping criteria = ${this.solution.StoppingCriteria}!\n
              `);
303       }
304     }
305   }
306
307   /**
308    * Creates the initial population with mixed uniform distribution.
309    */
310   creationMixedUniform() {
311     var lb = this.opts.lb;
312     var ub = this.opts.ub;
313     if(!this.bounded) {
314       lb = this.opts.InitalUnboundedRange.map((v) => v[0]);
315       ub = this.opts.InitalUnboundedRange.map((v) => v[1]);
316     }
317     var N = this.problem.IntCon.length;
318     var population = arrayRand(lb, ub,
319       this.problem.nvars, this.opts.PopulationSize, this.rand);
320     if(N) {
321       var r = arrayRandi([0, 1], N, this.opts.PopulationSize, this.rand);
322       var I = index(this.problem.IntCon,
323         range(0, this.opts.PopulationSize - 1), this.problem.nvars);
324
325       setSub(population, I, plus(fix(getSub(population, I)), r));
```

```
326       }
327       return population;
328     }
329
330     /**
331      * Performs binary tournament selection of parents.
332      */
333     binaryTournamentSelection() {
334       var parents = zeros(this.state.nParents);
335       var r1 = arrayRandi([0, this.opts.PopulationSize-1], 2,
336         this.state.nParents, this.rand);
337       var r2 = arrayRandi([0, this.opts.PopulationSize-1], 2,
338         this.state.nParents, this.rand);
339       var neq = elementWise((a, b) => a > b, r1, r2);
340       var I1 = indexOfAll(neq, true); // closer r2
341       var I2 = indexOfAll(neq, false); // closer r1
342       setSub(parents, I1, getSub(r2, I1));
343       setSub(parents, I2, getSub(r1, I2));
344       return parents;
345     }
346
347     /**
348      * Executes Laplace mixed crossover to generate offspring.
349      */
350     laplaceMixedCrossover() {
351       var N = this.problem.IntCon.length;
352       var nvars = this.problem.nvars;
353       var iParents = this.state.parents;
354       var nChildren = this.state.ReproductionCount.children;
355       var nParents = this.state.parentsCount.crossover;
356       var sParents = this.state.parentsSelection.crossover;
357
358       var b = scale(ones(nvars * nChildren/2), this.opts.b_real);
359       if(N) {
360         setSub(b, index(this.problem.IntCon, range(0, nChildren/2 - 1), nvars),
361           scale(ones(N * nChildren/2), this.opts.b_integer));
362       }
363       var u = arrayRand(zeros(nvars), ones(nvars), nvars, nChildren/2, this.rand);
364       var r = arrayRand(zeros(nvars), ones(nvars), nvars, nChildren/2, this.rand);
365       var beta = elementWise((x, y) => this.opts.a-x*log10(y), b, u); // a-b*log10
            (u)
366       var I = indexOfAll(r.map((ri) => ri > 0.5), true);
367       setSub(beta, I, elementWise((x, y) => this.opts.a+x*log10(y),
368         getSub(b, I),  getSub(u, I))); // a+b*log10(u)
369
370       var mother = getSub(this.state.Population, index(range(0, nvars - 1),
371         getSub(iParents, getSub(sParents, range(0, nParents/2-1))), nvars));
372       var father = getSub(this.state.Population, index(range(0, nvars - 1),
373         getSub(iParents, getSub(sParents, range(nParents/2, nParents-1))), nvars))
            ;
374       var children = plus(concatRow(nvars, mother, father),
375         elementWise((a, b, c) => a*abs(b-c),
376         repCol(beta, 2), repCol(mother, 2), repCol(father, 2)));
377       return children;
378     }
379
380     /**
381      * Applies power mixed mutation to selected individuals.
382      */
```

```
383   powerMixedMutation() {
384     var N = this.problem.IntCon.length;
385     var nvars = this.problem.nvars;
386     var nMutants = this.state.ReproductionCount.mutants;
387     var sParents = this.state.parentsSelection.mutation;
388
389     var p = scale(ones(nvars * nMutants), this.opts.p_real);
390     if(N) {
391       setSub(p, index(this.problem.IntCon, range(0, nMutants - 1), nvars),
392         scale(ones(N * nMutants), this.opts.p_integer));
393     }
394     var s1 = arrayRand(zeros(nvars), ones(nvars), nvars, nMutants, this.rand);
395     var r = arrayRand(zeros(nvars), ones(nvars), nvars, nMutants, this.rand);
396     var s = elementWise((a, b) => pow(a, b), s1, p);
397
398     var parents = getSub(this.state.Population,
399       getSub(sParents, index(range(0, nvars - 1), sParents, nvars)));
400     var mutants;
401
402     if(!this.bounded) {
403       mutants = elementWise((a, b, c) => c < 0.5 ? a + b : a - b, parents, s, r)
404     } else {
405       var ubm = repCol(this.opts.ub, nMutants);
406       var lbm = repCol(this.opts.lb, nMutants);
407       var t = elementWise((a, b, c) => (a - b)/(c - a), parents, this.opts.lb,
          this.opts.ub);
408       mutants = elementWise((a, b, c) => a+b*(a-c), parents, s, lbm);
409
410       var I = indexOfAll(elementWise((a, b) => a < b, t, r), true);
411       setSub(mutants, I, elementWise((a, b, c) => a-b*(c-a),
412         getSub(parents, I), getSub(s, I), getSub(ubm, I)));
413     }
414
415     return mutants;
416   }
417
418   /**
419    * Applies Gaussian mutation to selected individuals.
420    */
421   gaussianMutation() {
422
423   }
424
425   /**
426    * Ensures integer constraints are met for specific variables.
427    */
428   integerRestriction() {
429     var N = this.problem.IntCon.length;
430     if(N) {
431       var I = index(this.problem.IntCon,
432         [...this.selection.children, ...this.selection.mutants], this.problem.
            nvars);
433       var p = getSub(this.state.Population, I);
434
435       var r = arrayRandi([0, 1], N,
436         this.state.ReproductionCount.children +
437         this.state.ReproductionCount.mutants, this.rand);
438
439       var Is = indexOfAll(neg(isInteger(p)), true);
```

```
440       setSub(p, Is, plus(fix(getSub(p, Is)), getSub(r, Is)));
441       setSub(this.state.Population, I, p);
442     }
443   }
444
445   /**
446    * Checks and enforces variable bounds within the population.
447    */
448   checkBounds() {
449     this.state.Population = elementWise((a, b, c) => min([max([a, b]), c]),
450       this.state.Population, this.opts.lbm, this.opts.ubm);
451   }
452
453   /**
454    * Evaluates whether stopping criteria have been met.
455    * @returns {number} Code indicating the reason to stop or continue.
456    */
457   stoppingCriteria() {
458     var stop = 0;
459     var gen = this.state.Generation;
460
461     if(gen > 0) {
462       if(this.state.Best[gen] < this.state.Best[gen - 1] ||
463           !this.state.Feasible[gen] ||
464           this.state.Feasible[gen] > this.state.Feasible[gen - 1]) {
465         this.state.StallTime = tic;
466         this.state.StallGenerations = 1;
467       } else {
468         this.state.StallGenerations = this.state.StallGenerations + 1;
469       }
470     }
471
472     if(gen >= (this.opts.MaxGenerations - 1)) {
473       // 1 - Maksimalni broj generacija ostvaren
474       stop = 1;
475     } else if(toc(this.state.StartTime) >= this.opts.MaxTime) {
476       // 2 - Proteklo maksimalno vreme trajanja optimizacije
477       stop = 2;
478     } else if(this.state.Feasible[gen] &&
479         this.state.Best[gen] <= this.opts.FitnessLimit) {
480       // 3 - Funkcija prilagodjenosti dostiglja ciljanu vrednost
481       stop = 3;
482     } else if(gen > (this.opts.MaxStallGenerations - 2) &&
483         this.state.Feasible[gen - (this.opts.MaxStallGenerations - 2)] &&
484         this.state.Feasible[gen] &&
485         ((this.state.Best[gen - (this.opts.MaxStallGenerations - 2)] -
486         this.state.Best[gen]) <= this.opts.FunctionTolerance)) {
487       // 4 - Promena vrednosti funkcija manja od tolerancije
488       stop = 4;
489     } else if(this.state.StallGenerations >= this.opts.MaxStallGenerations) {
490       // 5 - Bez promene vrednosti funkcije kroz odredjeni broj generacija
491       stop = 5;
492     } else if(toc(this.state.StallTime) >= this.opts.MaxStallTime) {
493       // 6 - Bez promene vrednosti funkcije odredjeno vreme
494       stop = 6;
495     } else if(this.stoped) {
496       // 7 - Korisnik zaustavlja proces
497       stop = 7;
498     }
```

```
499       return stop;
500     }
501
502     /**
503      * Initializes the population for the optimization process.
504      */
505     initPopulation() {
506       this.state.Population = this.creationFcn();
507       this.integerRestriction();
508       if(this.bounded) {
509         this.checkBounds();
510       }
511     }
512
513     /**
514      * Initializes the internal state of the algorithm.
515      */
516     initState() {
517       // Merenje vremena
518       this.state.GenTime = tic;
519
520       this.state.FunVal = zeros(this.opts.PopulationSize);
521
522       this.state.ConSumVal = zeros(this.opts.PopulationSize);
523       this.state.Best = zeros(this.opts.MaxGenerations);
524       this.state.Feasible = ones(this.opts.MaxGenerations);
525       this.state.Time = zeros(this.opts.MaxGenerations);
526       this.state.x = zeros(this.opts.MaxGenerations * this.problem.nvars);
527
528       this.state.ReproductionCount = {};
529       this.state.ReproductionCount.elite = this.opts.EliteCount;
530       this.state.ReproductionCount.children = fix(this.opts.CrossoverFraction *
531         (this.opts.PopulationSize - this.opts.EliteCount));
532
533       if(mod(this.state.ReproductionCount.children, 2) != 0) {
534         this.state.ReproductionCount.children = this.state.ReproductionCount.
              children - 1;
535       }
536
537       this.state.ReproductionCount.mutants = this.opts.PopulationSize -
538         (this.state.ReproductionCount.elite + this.state.ReproductionCount.
              children);
539
540       this.state.parentsCount = {};
541       this.state.parentsCount.crossover = this.state.ReproductionCount.children;
542       this.state.parentsCount.mutation = this.state.ReproductionCount.mutants;
543       this.state.nParents = this.state.parentsCount.crossover +
544         this.state.parentsCount.mutation;
545
546       this.state.parentsSelection = {};
547       this.state.parentsSelection.crossover = range(0,
548         this.state.parentsCount.crossover - 1);
549       this.state.parentsSelection.mutation = plus(
550         this.state.parentsCount.crossover,
551         range(0, this.state.parentsCount.mutation - 1));
552
553       this.selection.elite = range(0, this.state.ReproductionCount.elite - 1);
554       this.selection.children = plus(this.state.ReproductionCount.elite,
555         range(0, this.state.ReproductionCount.children - 1));
```

```
556       this.selection.mutants = plus((this.state.ReproductionCount.elite +
557         this.state.ReproductionCount.children),
558         range(0, this.state.ReproductionCount.mutants - 1));
559
560       if(this.constrained) {
561         this.state.ConSize = [
562           this.problem.nonlconfcn(zeros(this.problem.nvars)).length, this.opts.
                PopulationSize];
563
564         this.state.ConVal = zeros(this.state.ConSize[0] * this.state.ConSize[1]);
565         this.state.ConNormVal = zeros(this.state.ConSize[0] * this.state.ConSize
              [1]);
566       }
567     }
568
569     /**
570      * Updates the current state of the algorithm based on evaluations.
571      */
572     async updateState() {
573       await this.evalFitnessFcn();
574       if(this.constrained) {
575         await this.evalConstraintsFcn();
576         this.updatePenalty();
577       }
578       this.state.Time[this.state.Generation] = toc(this.state.GenTime);
579       var i;
580       [this.state.Best[this.state.Generation], i] = mini(this.state.FunVal);
581       setSub(this.state.x, index(range(0, this.problem.nvars - 1),
582          this.state.Generation,  this.problem.nvars),
583          getSub(this.state.Population,
584          index(range(0, this.problem.nvars - 1), i,  this.problem.nvars)));
585
586       if(this.constrained && this.state.ConSumVal[i] > 0) {
587         this.state.Feasible[this.state.Generation] = 0;
588       }
589     }
590
591     /**
592      * Updates penalty values based on constraint violations.
593      */
594     updatePenalty() {
595       if(any(this.state.ConSumVal.map((a) => a == 0))) {
596         if(this.state.Generation == 0) {
597           var [s, i] = sorti(this.state.FunVal);
598           this.selection.current_elite = getSub(i, this.selection.elite);
599         }
600         var fmin;
601         if(this.state.ReproductionCount.elite > 0) {
602           // Ne moze da bude bolje od najgore elitne jedinke
603           fmin = max(getSub(this.state.FunVal, this.selection.current_elite));
604         } else {
605           // Ne moze da bude bolje od najbolje jedinike
606           fmin = min(getSub(this.state.FunVal, this.selection.current_elite));
607         }
608         var I1 = indexOfAll(elementWise((a, b) => a > 0 && b <= fmin,
609           this.state.ConSumVal, this.state.FunVal), true);
610         var I2 = indexOfAll(elementWise((a, b) => a > 0 && b > fmin,
611           this.state.ConSumVal, this.state.FunVal), true);
612         setSub(this.state.FunVal, I1, plus(fmin,
```

```
613         getSub(this.state.ConSumVal, I1)));
614       setSub(this.state.FunVal, I2, plus(getSub(this.state.FunVal, I2),
615         getSub(this.state.ConSumVal, I2)));
616     } else {
617       this.state.FunVal = this.state.ConSumVal;
618     }
619   }
620
621   /**
622    * Evaluates the fitness function for the current population.
623    */
624   async evalFitnessFcn() {
625     var obj = this;
626     var nvars = this.problem.nvars;
627
628     var p;
629     var n;
630     if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
631       p = getSub(this.state.Population, index(range(0, nvars - 1),
632         [...this.selection.children, ...this.selection.mutants], nvars));
633       n = this.state.ReproductionCount.children + this.state.ReproductionCount.
           mutants;
634     } else {
635       p = this.state.Population;
636       n = this.opts.PopulationSize;
637     }
638
639     var val;
640     if(this.opts.UseVectorized) {
641       val = this.problem.fitnessfcn(p);
642     } else {
643       if(this.opts.UseParallel) {
644         val = await parallel.parfor(0, n-1, 1, parallel.getProcessorsNum(),
645           this.parallel_context, this.parallel_setup_fun, `function(i) {
646             var fun = ${obj.problem.fitnessfcn.toString()};
647             return fun(getSub(${JSON.stringify(p)}, index(range(0, ${nvars} - 1)
                 , i, ${nvars})));
648         }`);
649       } else {
650         val = zeros(n);
651         for(var i = 0; i < n; i++) {
652           val[i] = this.problem.fitnessfcn(getSub(p, index(range(0, nvars - 1),
                i, nvars)));
653         }
654       }
655     }
656
657     if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
658       this.state.FunVal = [...getSub(this.state.FunVal, this.selection.
           current_elite), ...val];
659     } else {
660       this.state.FunVal = val;
661     }
662   }
663
664   /**
665    * Evaluates constraint functions for the current population.
666    */
667   async evalConstraintsFcn() {
```

```
668     var obj = this;
669     var nvars = this.problem.nvars;
670
671     var p;
672     var n;
673     var m = this.state.ConSize[0];
674     if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
675       p = getSub(this.state.Population, index(range(0, nvars - 1),
676         [...this.selection.children, ...this.selection.mutants], nvars));
677       n = this.state.ReproductionCount.children + this.state.ReproductionCount.
             mutants;
678     } else {
679       p = this.state.Population;
680       n = this.opts.PopulationSize;
681     }
682
683     var val;
684     if(this.opts.UseVectorized) {
685       val = obj.problem.nonlconfcn(p);
686     } else {
687       if(this.opts.UseParallel) {
688         val = await parallel.parfor(0, n-1, 1, parallel.getProcessorsNum(),
689           this.parallel_context, this.parallel_setup_fun, `function(i) {
690             var fun = ${obj.problem.nonlconfcn.toString()};
691             return fun(getSub(${JSON.stringify(p)}, index(range(0, ${nvars} - 1)
                    , i, ${nvars})));
692         }`);
693         val = val.flat();
694       } else {
695         m = this.state.ConSize[0];
696         val = zeros(m * n);
697         for(var i = 0; i < n; i++) {
698           setSub(val, index(range(0, m - 1), i, m),
699             this.problem.nonlconfcn(getSub(p,
700             index(range(0, nvars - 1), i, nvars))));
701         }
702       }
703     }
704     if(this.state.ReproductionCount.elite > 0 && this.state.Generation > 0) {
705       this.state.ConVal = concatCol(m, getSub(this.state.ConVal,
706         index(range(0, m - 1), this.selection.current_elite, m)), val);
707     } else {
708       this.state.ConVal = val;
709     }
710     this.normConstraintsFcn();
711     this.state.ConSumVal = sumCol(this.state.ConNormVal, m, this.opts.
          PopulationSize);
712   }
713
714   /**
715    * Normalizes constraint values to maintain consistent scaling.
716    */
717   normConstraintsFcn() {
718     var ConNormVal = this.state.ConVal;
719     var sNaN = indexOfAll(isNaN(ConNormVal), true);
720     var sInf = indexOfAll(isInfinity(ConNormVal), true);
721     var sNeg = indexOfAll(isNegative(ConNormVal), true);
722
723     setSub(ConNormVal, sNaN, zeros(sNaN.length));
```

```
724     setSub(ConNormVal, sInf, zeros(sInf.length));
725     setSub(ConNormVal, sNeg, zeros(sNeg.length));
726 
727     var m = this.state.ConSize[0];
728     var n = this.opts.PopulationSize;
729 
730     if(this.state.Generation == 0 ||
731         (this.state.Generation > 0 &&
732         this.state.Feasible[this.state.Generation-1])) {
733 
734       // Use first values of the first generation for normalization of
735       // constraints until there is a feasible solution
736       var l = elementWise((a) => sqrt(a),
737         sumRow(elementWise((a) => pow(a, 2), ConNormVal), m, n));
738       var I = indexOfAll(l, 0);
739       setSub(l, I, ones(I.length));
740       if(isEmpty(this.l)) {
741         this.l = l;
742       } else {
743         var s = indexOfAll(elementWise((a, b) => a > b, this.l, l), true);
744         setSub(this.l, s, getSub(l, s));
745       }
746 
747       this.lm = repCol(this.l, n);
748     }
749 
750     setSub(ConNormVal, sNaN, getSub(this.lm, sNaN));
751     setSub(ConNormVal, sInf, getSub(this.lm, sInf));
752     this.state.ConNormVal = divideEl(ConNormVal, this.lm);
753   }
754 }
755 
756 exports.OPTIM_RCMIGA = OPTIM_RCMIGA;
```

Listing P.1 - `RCMIGA.jsl`

```
1  /**
2   * @file Example for RCMIGA
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8  close('all'); clear; clc; tic;
9  disp(' --- ' + jsl_file_name + ' ---');
10 // --------------------
11 
12 var { OPTIM_RCMIGA } = require('./RCMIGA.jsl');
13 
14 var f_sph = (x) => { // funkcija prilagodjenosti
15   return x.reduce((sum, xi) => sum + pow(xi - 5, 2), 0);
16 };
17 
18 var f_cons = (x) => { // funkcija ogranicenja
19   return minus(x, 4);
20 };
21 
22 // Definicija problema
23 var problem = {};
24 problem.fitnessfcn = f_sph;
```

```
problem.nonlconfcn = f_cons;
problem.nvars = 5; // broj promenljivih
problem.lb = [1, 1, 1, 1, 1]; // donja granica
problem.ub = [12, 15, 20, 25, 12]; // gornja granica
problem.IntCon = [0, 1, 2, 3, 4]; // celobrojne promenljive

// Definicija parametara
var opts = {};
opts.PopulationSize = 100;
opts.EliteCount = 10;
opts.CrossoverFraction = 0.9;
opts.MaxGenerations = 300;
opts.MaxStallGenerations = 100;
opts.FitnessLimit = 0;
opts.FunctionTolerance = 1e-6;
opts.MaxTime = 12 * 60 * 60;
opts.MaxStallTime = 30;
opts.UseVectorized = false; // true | [ false ]
opts.UseParallel = false; // true | [ false ]
opts.Display = 'iter'; // [ iter ] | final | off

var ga = new OPTIM_RCMIGA(problem, opts);
await ga.run();

// - Kraj proračuna
disp(' Program je uspešno izvršen... ');
disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
disp(' -------------------- ');
```

Listing P.2 - `example_RCMIGA.jsl`

Q

Skripte za primenu postupka proračuna optimalne geometrije

U listingu Q.1 je data JSLAB skripta za primenu postupka proračuna optimalne geometrije mehanizma simulatora leta kao što je opisano u poglavlju 5. Isto je urađeno i u skripti datoj u listingu Q.2 ali sa paralelizacijom proračuna.

```
/**
 * @file Mechanisam optimization with RCMIGA
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var copy_solver = false;
var copy_eigen = false;
var compile_solver = false;
var libs_path = "C:/_Electron_lib/";
var solver_path = "C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS/build/Release/
    solver_mp_6rss";
var OP = {
  // Postavke optimizacije
  N_p: 5000, // velicina populacije
  N_g: 1000, // maksimalni broj generacija
  N_e: 500, // broj elitnih
  t_max: 12 * 60 * 60, // [s] maksimalno vreme
  t_s: 1 * 60 * 60, // [s] maksimalno vreme bez promene
  N_gs: 300, // maksimalni broj generacija bez promene
  FL: 0, // zeljena vrednost funkcije prilagodjenosti
  FT: 1e-9, // maksimalna tolerancija vrednosti funkcije
  CF: 0.3, // deo populacije koju crossover funkcija pravi

  // Postavke objektivne funkcije
  b: [0.4, 0.3, 0.3], // vektor za bodovanje[eta, phi, theta]
  x_gmax: [500, 500, 250, 35, 35, 30], // max vrednosti parametara za bodovanje
  x_lim: [ // granicne vrednosti parametara
    [-1000, 1000],
    [-1000, 1000],
    [-500, 500],
    [-60, 60],
```

```
    [-60, 60],
    [-90, 90]
  ],
  k: [ // faktori pretrage
    [10, 10, 10, 6, 6, 9],
    createFilledArray(6, 5),
    createFilledArray(6, 5),
    createFilledArray(6, 2),
    createFilledArray(6, 2),
    createFilledArray(6, 2)
  ],

  // Nelinearna ogranicenja
  M_max: 200, // [Nm] raspolozivi moment reduktora
  x_R: 110, // [mm] polozaj napadne tacke sile duz vratila
  a_R: 162, // [mm] a konstanta reduktora
  b_R: 122, // [mm] b konstanta reduktora
  Fr_R: 5000, // [N] maksimalna dozvoljena radijalna sila
  dr: 100, // [mm] minimalno rastojanje izmedju poluga
  Na: 50, // [-] broj podela poluge

  // Tacke za proveru radnog prostora
  x_max: 150, // min i max x koordinata cetvorostrane prizme
  y_max: 150, // min i max y koordinata cetvorostrane prizme
  z_max: 75, // min i max z koordinata cetvorostrane prizme
  kg: [ // Granice konfiguracija
    [100, 10, 400], // a[mm]
    [600, 10, 1000], // s[mm]
    [400, 10, 800],  // R_p[mm]
    [700, 10, 1500], // R_b[mm]
    [5, 1, 55],      // gama_p[deg]
    [5, 1, 55],      // gama_b[deg]
    [-90, 1, 90],    // eps[deg]
    [-180, 1, 180],  // tau_ll[deg]
    [-90, 1, 90],    // tau_jp[deg]
    [-90, 1, 90],    // mi_ll[deg]
    [-90, 1, 90]     // mi_jp[deg]
  ]
};
var MP = {
  M: 150,
  cg: [0, 0, 300],
  g: [0, 0, -9.81],
  lambda_max: 35, // [deg] dozvoljeni uglovi u zglobovima
};
// --------------------

// Copy and compile cpp module
if(copy_solver) {
  mkdir('C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS');
  disp(' Downloading SOLVER_MP_6RSS');
  copyDir(libs_path+'SOLVER_MP_6RSS', 'C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS
      ');
}
if(copy_eigen) {
  disp(' Downloading eigen-3.4.0');
  copyDir7z(libs_path+'eigen-3.4.0.7z', 'C:/NodeJS/SolverMP6RSS/lib/');
}
if(compile_solver) {
```

```
94    disp(' Compiling SOLVER_MP_6RSS');
95    compileNapi('C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS');
96  }
97
98  // Load solvers
99  disp(' Loading solvers...');
100
101 var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
102 var solver = new MP_6RSS_SOLVER(MP);
103 var { SolverMP6RSS } = require(solver_path);
104 var solver_cpp = new SolverMP6RSS(MP);
105
106 var { SPACE_SEARCH } = require('./classes/SPACE_SEARCH.jsl');
107 var crawler = new SPACE_SEARCH();
108
109 var { OPTIM_RCMIGA } = require('./classes/RCMIGA.jsl');
110
111 // - Optimizacija
112 // Korak promene parametara za optimizaciju
113 crawler.dispElementSize(OP.x_lim, OP.k);
114
115 // Odredjivanje dozvoljenih sila za reduktor
116 disp(' Odredjivanje dozvoljenih sila za reduktor...');
117 [OP.Frd_R, OP.Fad_R] = solver.getActuatorAllowedLoad(OP.Fr_R, OP.a_R, OP.b_R, OP
      .x_R);
118 disp(' Dozvoljena radijalna sila reduktora: ' + num2str(OP.Frd_R, 2) + ' N');
119 disp(' Dozvoljena aksijalna sila reduktora: ' + num2str(OP.Fad_R, 2) + ' N');
120
121 var [xi, yi, zi] = meshgrid(
122   range(-OP.x_max, OP.x_max, OP.x_max),
123   range(-OP.y_max, OP.y_max, OP.y_max),
124   range(-OP.z_max, OP.z_max, OP.z_max));
125 OP.N = 3 ** 3; // broj tacaka za proracun
126 xi = xi.flat(2); yi = yi.flat(2); zi = zi.flat(2);
127 OP.x = reshape(concatCol(xi.length,
128   xi, yi, zi, zeros(3 * xi.length)), OP.N, 6);
129
130 var lb = zeros(OP.kg.length, 1);
131 var ub = zeros(OP.kg.length, 1);
132 for(var i = 0; i < OP.kg.length; i++) {
133   lb[i] = 0;
134   ub[i] = (OP.kg[i][2] - OP.kg[i][0]) / OP.kg[i][1];
135 }
136
137 // Definicija problema
138 OP.kgn = [];
139 for(var i = 0; i < OP.kg.length; i++) {
140   OP.kgn[i] = range(OP.kg[i][0], OP.kg[i][2], OP.kg[i][1]);
141 }
142
143 var mapping = (x) => {
144   var x_real = [...x];
145   for(var i = 0; i < x.length; i++) {
146     x_real[i] = OP.kgn[i][x[i]];
147   }
148   return x_real;
149 };
150
151 var f_cons = (x) => {
```

```
  return solver.constraints(mapping(x), OP, crawler);
};

var f_sph = (x) => {
  return solver.fitness(mapping(x), OP, crawler);
};

var problem = {};
problem.fitnessfcn = f_sph;
problem.nonlconfcn = f_cons;
problem.nvars = OP.kg.length; // broj promenljivih
problem.lb = lb; // donja granica
problem.ub = ub; // gornja granica
problem.IntCon = range(0, OP.kg.length - 1); // celobrojne promenljive

// Definicija parametara
var opts = {};
opts.PopulationSize = OP.N_p;
opts.EliteCount = OP.N_e;
opts.CrossoverFraction = OP.CF;
opts.MaxGenerations = OP.N_g;
opts.MaxStallGenerations = OP.N_gs;
opts.FitnessLimit = OP.FL;
opts.FunctionTolerance = OP.FT;
opts.MaxTime = OP.t_max;
opts.MaxStallTime = OP.t_s;
opts.UseVectorized = false; // true | [ false ]
opts.UseParallel = false; // true | [ false ]
opts.Display = 'iter'; // [ iter ] | final | off

var t1 = tic;
var ga = new OPTIM_RCMIGA(problem, opts);
await ga.run();
var xbest = mapping(ga.solution.x);
OP.y_of = ga.solution.fval;
OP.solution = ga.solution;
OP.state = ga.state;

disp(' --------------------');
disp(' Kraj RCMIGA optimizacije - ' + getDateTimeFull());
disp(' --------------------');
disp(' Penalty value: ' + num2str(OP.y_of));

MP.a = xbest[0];
MP.s = xbest[1];
MP.R_p = xbest[2];
MP.R_b = xbest[3];
MP.gamma_p = xbest[4];
MP.gamma_b = xbest[5];
MP.epsilon = xbest[6];
MP.tau_ll = xbest[7];
MP.tau_jp = xbest[8];
MP.mu_ll = xbest[9];
MP.mu_jp = xbest[10];

var filename = pwd + 'results/ga_' + getDateTimeStr() + '.json';
save(filename, 'MP', 'OP');

disp(' Podaci sacuvani u: '+ filename);
```

```
211
212 disp(' Dobijena konfiguracija:');
213 disp(' MP.a = ' + num2str(MP.a, '%.0f') + ' mm');
214 disp(' MP.s = ' + num2str(MP.s, '%.0f') + ' mm');
215 disp(' MP.R_p = ' + num2str(MP.R_p, '%.0f') + ' mm');
216 disp(' MP.R_b = ' + num2str(MP.R_b, '%.0f') + ' mm');
217 disp(' MP.gamma_p = ' + num2str(MP.gamma_p, '%.0f') + ' deg');
218 disp(' MP.gamma_b = ' + num2str(MP.gamma_b, '%.0f') + ' deg');
219 disp(' MP.epsilon = ' + num2str(MP.epsilon, '%.0f') + ' deg');
220 disp(' MP.tau_ll = ' + num2str(MP.tau_ll, '%.0f') + ' deg');
221 disp(' MP.tau_jp = ' + num2str(MP.tau_jp, '%.0f') + ' deg');
222 disp(' MP.mu_ll = ' + num2str(MP.mu_ll, '%.0f') + ' deg');
223 disp(' MP.mu_jp = ' + num2str(MP.mu_jp, '%.0f') + ' deg');
224 disp(' --------------------');
225 disp(' Potrebno vreme za optimizaciju: ' + num2str(toc(t1), 2) + ' sekundi');
226
227 // - Kraj proračuna
228 disp(' Program je uspešno izvršen... ');
229 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
230 disp(' -------------------- ');
```

Listing Q.1 - optimizationRCMIGA.jsl

```
1  /**
2   * @file Mechanisam optimization with parallel RCMIGA
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8  close('all'); clear; clc; tic;
9  disp(' --- ' + jsl_file_name + ' ---');
10 // - Ulazni podaci
11 var copy_solver = false;
12 var copy_eigen = false;
13 var compile_solver = false;
14 var libs_path = "C:/_Electron_lib/";
15 var solver_path = "C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS/build/Release/
       solver_mp_6rss";
16 var OP = {
17   // Postavke optimizacije
18   N_p: 5000, // velicina populacije
19   N_g: 1000, // maksimalni broj generacija
20   N_e: 500, // broj elitnih
21   t_max: 12 * 60 * 60, // [s] maksimalno vreme
22   t_s: 1 * 60 * 60, // [s] maksimalno vreme bez promene
23   N_gs: 300, // maksimalni broj generacija bez promene
24   FL: 0, // zeljena vrednost funkcije prilagodjenosti
25   FT: 1e-9, // maksimalna tolerancija vrednosti funkcije
26   CF: 0.3, // deo populacije koju crossover funkcija pravi
27
28   // Postavke objektivne funkcije
29   b: [0.4, 0.3, 0.3], // vektor za bodovanje[eta, phi, theta]
30   x_gmax: [500, 500, 250, 35, 35, 30], // max vrednosti parametara za bodovanje
31   x_lim: [ // granicne vrednosti parametara
32     [-1000, 1000],
33     [-1000, 1000],
34     [-500, 500],
35     [-60, 60],
36     [-60, 60],
```

```
    [-90, 90]
  ],
  k: [ // faktori pretrage
    [10, 10, 10, 6, 6, 9],
    createFilledArray(6, 5),
    createFilledArray(6, 5),
    createFilledArray(6, 2),
    createFilledArray(6, 2),
    createFilledArray(6, 2)
  ],

  // Nelinearna ogranicenja
  M_max: 200, // [Nm] raspolozivi moment reduktora
  x_R: 110, // [mm] polozaj napadne tacke sile duz vratila
  a_R: 162, // [mm] a konstanta reduktora
  b_R: 122, // [mm] b konstanta reduktora
  Fr_R: 5000, // [N] maksimalna dozvoljena radijalna sila
  dr: 100, // [mm] minimalno rastojanje izmedju poluga
  Na: 50, // [-] broj podela poluge

  // Tacke za proveru radnog prostora
  x_max: 150, // min i max x koordinata cetvorostrane prizme
  y_max: 150, // min i max y koordinata cetvorostrane prizme
  z_max: 75, // min i max z koordinata cetvorostrane prizme
  kg: [ // Granice konfiguracija
    [100, 10, 400], // a[mm]
    [600, 10, 1000], // s[mm]
    [400, 10, 800],  // R_p[mm]
    [700, 10, 1500], // R_b[mm]
    [5, 1, 55],      // gama_p[deg]
    [5, 1, 55],      // gama_b[deg]
    [-90, 1, 90],    // eps[deg]
    [-180, 1, 180],  // tau_ll[deg]
    [-90, 1, 90],    // tau_jp[deg]
    [-90, 1, 90],    // mi_ll[deg]
    [-90, 1, 90]     // mi_jp[deg]
  ]
};
var MP = {
  M: 150,
  cg: [0, 0, 300],
  g: [0, 0, -9.81],
  lambda_max: 35, // [deg] dozvoljeni uglovi u zglobovima
};
// --------------------

// Copy and compile cpp module
if(copy_solver) {
  mkdir('C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS');
  disp(' Downloading SOLVER_MP_6RSS');
  copyDir(libs_path+'SOLVER_MP_6RSS', 'C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS
      ');
}
if(copy_eigen) {
  disp(' Downloading eigen-3.4.0');
  copyDir7z(libs_path+'eigen-3.4.0.7z', 'C:/NodeJS/SolverMP6RSS/lib/');
}
if(compile_solver) {
  disp(' Compiling SOLVER_MP_6RSS');
```

```
  compileNapi('C:/NodeJS/SolverMP6RSS/lib/SOLVER_MP_6RSS');
}

// Load solvers
disp(' Loading solvers...');

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);
var { SolverMP6RSS } = require(solver_path);
var solver_cpp = new SolverMP6RSS(MP);

var { SPACE_SEARCH } = require('./classes/SPACE_SEARCH.jsl');
var crawler = new SPACE_SEARCH();

var { OPTIM_RCMIGA } = require('./classes/RCMIGA.jsl');

// - Optimizacija
// Korak promene parametara za optimizaciju
crawler.dispElementSize(OP.x_lim, OP.k);

// Odredjivanje dozvoljenih sila za reduktor
disp(' Odredjivanje dozvoljenih sila za reduktor...');
[OP.Frd_R, OP.Fad_R] = solver.getActuatorAllowedLoad(OP.Fr_R, OP.a_R, OP.b_R, OP
    .x_R);
disp(' Dozvoljena radijalna sila reduktora: ' + num2str(OP.Frd_R, 2) + ' N');
disp(' Dozvoljena aksijalna sila reduktora: ' + num2str(OP.Fad_R, 2) + ' N');

var [xi, yi, zi] = meshgrid(
  range(-OP.x_max, OP.x_max, OP.x_max),
  range(-OP.y_max, OP.y_max, OP.y_max),
  range(-OP.z_max, OP.z_max, OP.z_max));
OP.N = 3 ** 3; // broj tacaka za proracun
xi = xi.flat(2); yi = yi.flat(2); zi = zi.flat(2);
OP.x = reshape(concatCol(xi.length,
  xi, yi, zi, zeros(3 * xi.length)), OP.N, 6);

var lb = zeros(OP.kg.length, 1);
var ub = zeros(OP.kg.length, 1);
for(var i = 0; i < OP.kg.length; i++) {
  lb[i] = 0;
  ub[i] = (OP.kg[i][2] - OP.kg[i][0]) / OP.kg[i][1];
}

// Definicija problema
OP.kgn = [];
for(var i = 0; i < OP.kg.length; i++) {
  OP.kgn[i] = range(OP.kg[i][0], OP.kg[i][2], OP.kg[i][1]);
}

var mapping = (x) => {
  var x_real = [...x];
  for(var i = 0; i < x.length; i++) {
    x_real[i] = OP.kgn[i][x[i]];
  }
  return x_real;
};

var f_cons = (x) => {
  return solver.constraints(mapping(x), OP, crawler);
```

```
};

var f_sph = (x) => {
  return solver.fitness(mapping(x), OP, crawler);
};

var problem = {};
problem.fitnessfcn = f_sph;
problem.nonlconfcn = f_cons;
problem.nvars = OP.kg.length; // broj promenljivih
problem.lb = lb; // donja granica
problem.ub = ub; // gornja granica
problem.IntCon = range(0, OP.kg.length - 1); // celobrojne promenljive

// Definicija parametara
var opts = {};
opts.PopulationSize = OP.N_p;
opts.EliteCount = OP.N_e;
opts.CrossoverFraction = OP.CF;
opts.MaxGenerations = OP.N_g;
opts.MaxStallGenerations = OP.N_gs;
opts.FitnessLimit = OP.FL;
opts.FunctionTolerance = OP.FT;
opts.MaxTime = OP.t_max;
opts.MaxStallTime = OP.t_s;
opts.UseVectorized = false; // true | [ false ]
opts.UseParallel = true; // true | [ false ]
opts.Display = 'iter'; // [ iter ] | final | off

var t1 = tic;
var ga = new OPTIM_RCMIGA(problem, opts);
await ga.run({
  MP: MP,
  OP: OP,
  solver_path: solver_path
}, function() {
  var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
  var solver = new MP_6RSS_SOLVER(MP);

  var { SolverMP6RSS } = require(solver_path);
  //var solver_cpp = new SolverMP6RSS(MP);

  var { SPACE_SEARCH } = require('./classes/SPACE_SEARCH.jsl');
  var crawler = new SPACE_SEARCH();

  var mapping = (x) => {
    var x_real = [...x];
    for(var i = 0; i < x.length; i++) {
      x_real[i] = OP.kgn[i][x[i]];
    }
    return x_real;
  };
});
var xbest = mapping(ga.solution.x);
OP.y_of = ga.solution.fval;
OP.solution = ga.solution;
OP.state = ga.state;

disp(' --------------------');
```

```
212 disp(' Kraj RCMIGA optimizacije - ' + getDateTimeFull());
213 disp(' --------------------');
214 disp(' Penalty value: ' + num2str(OP.y_of));
215
216 MP.a = xbest[0];
217 MP.s = xbest[1];
218 MP.R_p = xbest[2];
219 MP.R_b = xbest[3];
220 MP.gamma_p = xbest[4];
221 MP.gamma_b = xbest[5];
222 MP.epsilon = xbest[6];
223 MP.tau_ll = xbest[7];
224 MP.tau_jp = xbest[8];
225 MP.mu_ll = xbest[9];
226 MP.mu_jp = xbest[10];
227
228 var filename = pwd + 'results/ga_' + getDateTimeStr() + '.json';
229 save(filename, 'MP', 'OP');
230
231 disp(' Podaci sacuvani u: '+ filename);
232
233 disp(' Dobijena konfiguracija:');
234 disp(' MP.a = ' + num2str(MP.a, '%.0f') + ' mm');
235 disp(' MP.s = ' + num2str(MP.s, '%.0f') + ' mm');
236 disp(' MP.R_p = ' + num2str(MP.R_p, '%.0f') + ' mm');
237 disp(' MP.R_b = ' + num2str(MP.R_b, '%.0f') + ' mm');
238 disp(' MP.gamma_p = ' + num2str(MP.gamma_p, '%.0f') + ' deg');
239 disp(' MP.gamma_b = ' + num2str(MP.gamma_b, '%.0f') + ' deg');
240 disp(' MP.epsilon = ' + num2str(MP.epsilon, '%.0f') + ' deg');
241 disp(' MP.tau_ll = ' + num2str(MP.tau_ll, '%.0f') + ' deg');
242 disp(' MP.tau_jp = ' + num2str(MP.tau_jp, '%.0f') + ' deg');
243 disp(' MP.mu_ll = ' + num2str(MP.mu_ll, '%.0f') + ' deg');
244 disp(' MP.mu_jp = ' + num2str(MP.mu_jp, '%.0f') + ' deg');
245 disp(' --------------------');
246 disp(' Potrebno vreme za optimizaciju: ' + num2str(toc(t1), 2) + ' sekundi');
247
248 // - Kraj proračuna
249 disp(' Program je uspešno izvršen... ');
250 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
251 disp(' -------------------- ');
```

Listing Q.2 - optimizationParallelRCMIGA.jsl

R

Konfiguracije za MP-6DOF-M i MP-6DOF-XXX

Konfiguracije klase `MP_6RSS_MECHANISM` za mehanizme `MP-6DOF-M` i `MP-6DOF-XXX` su date u listinzima R.1 i R.2.

```
/**
 * @file Motion Platform 6RSS Mechanism configuration for MP_6DOF_M
 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
 * PR-DC, Republic of Serbia
 * info@pr-dc.com
 * @version 0.0.1
 */

/**
 * Class for Motion Platform 6RSS Mechanism configs.
 */
class MP_6RSS_MECHANISM {
  constructor() {
    // Konfiguracija mehanizma
    this.a = 25; // [mm] duzina donje poluge od Bi do Ai
    this.s = 112; // [mm] duzina gornje poluge od Ai do Pi
    this.R_p = 63.5; // [mm] poluprecnik platforme od O1 do Pi
    this.R_b = 80.972; // [mm] poluprecnik baze od O do Bi
    this.gamma_p = 60 - 24.371; // [deg] ugao izmedju zglobova platforme
    this.gamma_b = 60 - 30.972; // [deg] ugao izmedju obrtnih zglobova baze
    this.epsilon = 0; // [deg] ugao ose motora u odnosu na horizontalnu ravan
    this.tau_ll = 0; // [deg] odstupanje ugla donje poluge od zadatog
    this.tau_jp = 0; // [deg] odstupanje ugla zglobova platforme od zadatog
    this.mu_ll = 0; // [deg] ugao ose zgloba donje poluge i ravani motora
    this.mu_jp = 0; // [deg] ugao ose zgloba platforme i horizontalne ravani
    this.mu_ul = NaN; // [deg] ugao ugradnje zglobova gornje poluge

    // Inercijalne karakteristike mehanizma
    this.m = 0.5; // [kg] masa tereta i platforme
    this.cg = [0, 0, 0]; // [mm] koordinate tezista tereta
    this.g = [0, 0, -9.81]; // [m/s^2] gravitaciono ubrzanje
    var I = [0, 0, 0, 0, 0, 0]; // Momenti inercije[Ixx, Iyy, Izz, Ixy, Ixz, Iyz
        ]
    this.I = [I[0], I[3], I[4], I[3], I[1], I[5], I[4], I[5], I[2]];

    // Ogranicenja mehanizma
    this.lambda_max = 35; // [deg] dozvoljeni uglovi u zglobovima
    this.dr_min = 5; // [mm] minimalno rastojanje izmedju poluga
```

```
38    }
39  }
40
41  exports.MP_6RSS_MECHANISM = MP_6RSS_MECHANISM;
```

Listing R.1 - MP_6DOF_M.jsl

```
1   /**
2    * @file Motion Platform 6RSS Mechanism configuration for MP_6DOF_XXX
3    * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
4    * PR-DC, Republic of Serbia
5    * info@pr-dc.com
6    * @version 0.0.1
7    */
8
9   /**
10   * Class for Motion Platform 6RSS Mechanism configs.
11   */
12  class MP_6RSS_MECHANISM {
13    constructor() {
14      // Konfiguracija mehanizma
15      this.a = 250; // [mm] duzina donje poluge od Bi do Ai
16      this.s = 1000; // [mm] duzina gornje poluge od Ai do Pi
17      this.R_p = 490; // [mm] poluprecnik platforme od O1 do Pi
18      this.R_b = 700; // [mm] poluprecnik baze od O do Bi
19      this.gamma_p = 44; // [deg] ugao izmedju zglobova platforme
20      this.gamma_b = 25; // [deg] ugao izmedju obrtnih zglobova baze
21      this.epsilon = 39; // [deg] ugao ose motora u odnosu na horizontalnu ravan
22      this.tau_ll = 19; // [deg] odstupanje ugla donje poluge od zadatog
23      this.tau_jp = -81; // [deg] odstupanje ugla zglobova platforme od zadatog
24      this.mu_ll = -31; // [deg] ugao ose zgloba donje poluge i ravani motora
25      this.mu_jp = 14; // [deg] ugao ose zgloba platforme i horizontalne ravani
26      this.mu_ul = NaN; // [deg] ugao ugradnje zglobova gornje poluge
27
28      // Inercijalne karakteristike mehanizma
29      this.M = 150; // [kg] masa tereta i platforme
30      this.cg = [0, 0, 300]; // [mm] koordinate tezista tereta
31      this.g = [0, 0, -9.81]; // [m/s^2] gravitaciono ubrzanje
32      var I = [60, 60, 60, 0, 0, 0]; // Momenti inercije [Ixx, Iyy, Izz, Ixy, Ixz,
            Iyz]
33      this.I = [I[0], I[3], I[4], I[3], I[1], I[5], I[4], I[5], I[2]];
34
35      // Ogranicenja mehanizma
36      this.lambda_max = 35; // [deg] dozvoljeni uglovi u zglobovima
37      this.dr_min = 100; // [mm] minimalno rastojanje izmedju poluga
38
39      this.lb = -50;
40    }
41  }
42
43  exports.MP_6RSS_MECHANISM = MP_6RSS_MECHANISM;
```

Listing R.2 - MP_6DOF_XXX.jsl

S

Program za upravljanje MP-6DOF-M

U listingu S.1 je dat `Arduino` program za upravljanje umanjenim prototipom simulatora leta *MP-6DOF-M*. Program je detaljno opisan u poglavlju 6.1.1.

```
// Control board program for MP-6DOF-M
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------
// Board: "Arduino AVT boards"
// Board part number: "Arduino Uno"
// --------------------

// Optional debug flags
//#define TERMINAL
//#define CONTROL_TEST

// Library
// --------------------
#ifdef TERMINAL
  // PRDC SCTP
  // Author: PRDC
  #include <PRDC_SCTP.h>
#else
  // PRDC SCBP
  // Author: PRDC
  #include <PRDC_SCBP.h>
#endif

// Servo
// Author: Arduino
// https://www.arduino.cc/en/reference/servo
#include <Servo.h>

// Define variables
// --------------------
// Servo configuration
#define m 6
#define N_c 3 // number of configuration parameters
#define HEARTBEAT_TIME 1000 // [ms]
#define LED_TIME 1000 // [ms] LED time difference

Servo motor[m];
int8_t pins[] = {5, 6, 9, 10, 11, 3};
```

```
41 uint16_t W_l[] = {300, 300, 300, 300, 300, 300};
42 uint16_t W_h[] = {1500, 1500, 1500, 1500, 1500, 1500};
43 uint16_t W_u[] = {2700, 2700, 2700, 2700, 2700, 2700};
44 float W_out[] = {1500, 1500, 1500, 1500, 1500, 1500};
45 float ip_lim[] = {1500, 1500, 1500, 1500, 1500, 1500};
46 int8_t alpha_s[m] = {1, -1, 1, -1, 1, -1};
47 float alpha_w_in[m] = {0, 0, 0, 0, 0, 0}; // [deg] zeljeni uglovi donjih poluga
48 float alpha_w[m];
49 bool CONFIGS_AVAILABLE = false; // motors configuration data available flag
50 bool STATES_AVAILABLE = false; // motors controls data available flag
51 bool CONTROLS_AVAILABLE = false; // motors controls data available flag
52 uint8_t states[m] = {1, 1, 1, 1, 1, 1}; // [-] motors state
53 uint8_t statesInPC[m] = {1, 1, 1, 1, 1, 1}; // [-] motors state from PC
54 uint16_t configsInPC[m*N_c]; // [-] configuration from PC
55 uint16_t configsSaved[m*N_c]; // [-] configuration from PC
56 uint16_t paramInPC[m]; // Parameters in vector
57
58 // Serial communications
59 #ifdef TERMINAL
60   PRDC_SCTP PC_COM;
61 #else
62   PRDC_SCBP PC_COM;
63 #endif
64 #define PC_USB_BAUDRATE 115200
65 #define ANGLE_MAX 360.0
66 #define ANGLE_C 180.0
67 #define ANGLE_MIN 0.0
68 #define protocolMax 65534
69
70 // Define packets
71 struct __attribute__((packed)) packet_t {
72   uint16_t ID;
73   uint16_t P[6];
74 };
75 union dataIn_t{
76   packet_t p;
77   uint8_t bytes[sizeof(packet_t)];
78 } dataInPC = {0, 0};
79 union dataOut_t{
80   packet_t p;
81   uint8_t bytes[sizeof(packet_t)];
82 } dataOutPC = {0, 0};
83
84 // setup function
85 // --------------------
86 void setup() {
87   // Define pin modes
88   pinMode(LED_BUILTIN, OUTPUT);
89
90   // Communication settings
91   PC_COM.begin(Serial, PC_USB_BAUDRATE);
92
93   #ifdef CONTROL_TEST
94     // Controls test
95     paramInPC[0] = 22767;
96     paramInPC[1] = 22767;
97     paramInPC[2] = 42767;
98     paramInPC[3] = 42767;
99     paramInPC[4] = 32767;
```

```
100       paramInPC[5] = 32767;
101       forwardKinematic();
102       controlMotors();
103       CONTROLS_AVAILABLE = false;
104       Serial.println("CONTROL TEST");
105       Serial.print("alpha_w = [ ");
106       for(int j = 0; j < m; ++j) {
107         Serial.print(alpha_w_in[j]);
108         Serial.print(" ");
109       }
110       Serial.println("]");
111       Serial.print("W_out = [ ");
112       for(int j = 0; j < m; ++j) {
113         Serial.print(W_out[j]);
114         Serial.print(" ");
115       }
116       Serial.println("]");
117
118       for(uint8_t j = 0; j < m; j++) {
119         configsInPC[j*N_c] = 300;
120         configsInPC[j*N_c+1] = 1500;
121         configsInPC[j*N_c+2] = 2700;
122       }
123
124       setConfigs();
125       Serial.print("W_l = [ ");
126       for(int j = 0; j < m; ++j) {
127         Serial.print(W_l[j]);
128         Serial.print(" ");
129       }
130       Serial.println("]");
131       Serial.print("W_h = [ ");
132       for(int j = 0; j < m; ++j) {
133         Serial.print(W_h[j]);
134         Serial.print(" ");
135       }
136       Serial.println("]");
137       Serial.print("W_u = [ ");
138       for(int j = 0; j < m; ++j) {
139         Serial.print(W_u[j]);
140         Serial.print(" ");
141       }
142       Serial.println("]");
143     #endif
144
145     // Serial data input
146     getControlsPC();
147
148     // Motors configuration
149     if(STATES_AVAILABLE) {
150       STATES_AVAILABLE = false;
151       setStates();
152     }
153     if(CONFIGS_AVAILABLE) {
154       CONFIGS_AVAILABLE = false;
155       setConfigs();
156     }
157
158     // Servo setup
```

```
159   for(int j = 0; j < m; ++j) {
160     motor[j].attach(pins[j]);
161     W_out[j] = W_h[j];
162     if(states[j]) {
163       motor[j].writeMicroseconds(W_out[j]);
164     }
165   }
166
167   // Enable motors
168   fill(1, states, m);
169 }
170
171 // loop function
172 // --------------------
173 void loop() {
174   // Serial data input
175   getControlsPC();
176
177   // Motors configuration
178   if(STATES_AVAILABLE) {
179     STATES_AVAILABLE = false;
180     setStates();
181   }
182   if(CONFIGS_AVAILABLE) {
183     CONFIGS_AVAILABLE = false;
184     setConfigs();
185   }
186
187   // Control motors
188   if(CONTROLS_AVAILABLE) {
189     CONTROLS_AVAILABLE = false;
190     controlMotors();
191   }
192
193   // Send heartbeat
194   sendHeartbeat();
195
196   // Blink LED
197   blink();
198 }
199
200 // getControlsPC function
201 // Get controls from PC
202 // --------------------
203 void getControlsPC() {
204   bool flagParamPC = false;
205   while(PC_COM.availableData() > 2) {
206     flagParamPC = false;
207     bool flagInPC = PC_COM.readPacket(dataInPC.bytes, sizeof(dataInPC));
208     if(flagInPC) {
209       if(dataInPC.p.ID == 11) {
210         // Mechanism controls
211         flagParamPC = true;
212          for(uint8_t j = 0; j < m; j++) {
213           paramInPC[j] = dataInPC.p.P[j];
214         }
215       } else if(dataInPC.p.ID == 1) {
216         // Enable/Disable actuator command
217         for(uint8_t j = 0; j < m; j++) {
```

```
218             statesInPC[j] = dataInPC.p.P[j];
219           }
220           STATES_AVAILABLE = true;
221         } else if(dataInPC.p.ID == 7 || dataInPC.p.ID == 8 || dataInPC.p.ID == 9)
              {
222           for(uint8_t j = 0; j < m; j++) {
223             configsInPC[j*N_c+(dataInPC.p.ID-7)] = dataInPC.p.P[j];
224           }
225           CONFIGS_AVAILABLE = true;
226         }
227       }
228     }
229
230     if(flagParamPC) {
231       flagParamPC = false;
232       if(dataInPC.p.ID == 11) {
233         // Forward kinematic
234         forwardKinematic();
235       }
236     }
237   }
238
239   // forwardKinematic function
240   // Calculate forward kinematic parameters
241   // --------------------
242   void forwardKinematic() {
243     CONTROLS_AVAILABLE = true;
244     for(uint8_t j = 0; j < m; j++) {
245       #ifdef TERMINAL
246         alpha_w_in[j] = paramInPC[j];
247       #else
248         alpha_w_in[j] =((float)paramInPC[j]/(float)protocolMax)* \
249           (ANGLE_MAX-ANGLE_MIN)+ANGLE_MIN;
250       #endif
251       if(alpha_w_in[j] > ANGLE_MAX) {
252         alpha_w_in[j] = ANGLE_MAX;
253       } else if(alpha_w_in[j] < ANGLE_MIN){
254         alpha_w_in[j] = ANGLE_MIN;
255       }
256       alpha_w_in[j] = alpha_w_in[j]-ANGLE_C;
257     }
258   }
259
260   // controlMotors function
261   // Send controls to motors
262   // --------------------
263   void controlMotors() {
264     for(uint8_t j = 0; j < m; j++) {
265       if(states[j]) {
266         alpha_w[j] = alpha_w_in[j];
267         if(alpha_s[j]*alpha_w[j] > 0) {
268           W_out[j] = W_h[j]+(W_u[j]-W_h[j])*(alpha_s[j]*alpha_w[j]/90.0);
269         } else {
270           W_out[j] = W_h[j]-(W_h[j]-W_l[j])*(-alpha_s[j]*alpha_w[j]/90.0);
271         }
272         motor[j].writeMicroseconds(W_out[j]);
273       }
274     }
275   }
```

```
276
277 // setState function
278 // Set motor state
279 // --------------------
280 void setStates() {
281   for(uint8_t j = 0; j < m; j++) {
282     states[j] = statesInPC[j];
283   }
284 }
285
286 // setConfig function
287 // Set motor configuration
288 // --------------------
289 void setConfigs() {
290   for(uint8_t i = 0; i < N_c; i++) {
291     bool flagSaveConfig = false;
292     for(uint8_t j = 0; j < m; j++) {
293       if(configsSaved[j*N_c+i] != configsInPC[j*N_c+i]) {
294         flagSaveConfig = true;
295         break;
296       }
297     }
298     if(flagSaveConfig) {
299       for(uint8_t j = 0; j < m; j++) {
300         configsSaved[j*N_c+i] = configsInPC[j*N_c+i];
301       }
302       if(i == 0) { // ID == 7
303         for(uint8_t j = 0; j < m; j++) {
304           W_l[j] = (float)configsInPC[j*N_c+i];
305         }
306       } else if(i == 1) { // ID == 8
307         for(uint8_t j = 0; j < m; j++) {
308           W_h[j] = (float)configsInPC[j*N_c+i];
309         }
310       } else if(i == 2) { // ID == 9
311         for(uint8_t j = 0; j < m; j++) {
312           W_u[j] = (float)configsInPC[j*N_c+i];
313         }
314       }
315     }
316   }
317   controlMotors();
318 }
319
320 // fill function
321 // fill vector with value
322 // --------------------
323 void fill(uint8_t x, uint8_t *A_out, uint8_t N) {
324   for(uint8_t i = 0; i < N; i++) {
325     A_out[i] = x;
326   }
327 }
328
329 // sendHeartbeat function
330 // Send heartbeat
331 // --------------------
332 void sendHeartbeat() {
333   static uint32_t t_heartbeat = millis();
334
```

```
335   if((millis()-t_heartbeat) > HEARTBEAT_TIME) {
336     t_heartbeat = millis();
337     PC_COM.sendPacket(dataOutPC.bytes, sizeof(dataOutPC));
338   }
339 }
340 
341 // blink function
342 // Blink LED at right time
343 // --------------------
344 void blink() {
345   static uint32_t t_blink = millis();
346   static bool s = false;
347 
348   if((millis()-t_blink) > LED_TIME) {
349     t_blink = millis();
350     s = !s;
351     digitalWrite(LED_BUILTIN, s);
352   }
353 }
```

Listing S.1 - MP-6DOF-M_CB.ino

T

Protokol za komunikaciju PRDC_SCBP

U ovom prilogu je data implementacija komunikacionog protokola `PRDC_SCBP` (*Serial Communication Binary Protocol*), koji je u programskom jeziku `JavaScript` implementiran kroz klasu `PRDC_SCBP_CODEC`. Pored implementacije u programskom jeziku `JavaScript`, priložena je implementacija i u programskom jeziku `Octave` ali i odgovarajuća `Arduino` biblioteka.

T.1 JavaScript implementacija

U okviru listinga T.5 je dat primer za korišćenje, dok je implementacija data u okviru T.1. Program iz T.2 pojednostavljuje komunikaciju sa serijskim uređajima povezanim na računar, u T.3 je data implementacija za `Node.js` dok je u T.4 data implementacja za `web`.

```
/**
 * @file Serial Communication Binary Protocol Codec
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

class PRDC_SCBP_CODEC {

  /**
   * Initializes the codec with default settings.
   */
  constructor() {
    this.M1; // start bit - marker (<)
    this.M2; // stop bit - marker (>)
    this.M3; // encode bit - marker (_)
    this.k; // counter
    this.encode_flag; // encode flag
    this.cks_in; // input checksum
    this.buffer_in; // input buffer
    this.buffer_in_read_ind; // read index for buffer_in
    this.data_in_temp; // temporary input data vector
    this.onDiscardedData;
    this.max_buffer_size = 2**20;
  }

  /**
```

```
29      * Prepares the codec for operation by initializing control characters and
            buffers.
30      */
31     begin() {
32       this.M1 = 60; // '<'.charCodeAt()
33       this.M2 = 62; // '>'.charCodeAt()
34       this.M3 = 95; // '_'.charCodeAt()
35       this.k = 0;
36       this.kr = 0;
37       this.k_t = 0;
38       this.encode_flag = 0;
39       this.cks_in = [0, 0];
40       this.data_in_temp = [];
41       this.buffer_in = [];
42       this.buffer_in_read_ind = 0; // Initialize read index
43       this.discarded_buffer = [];
44       this.discarded_buffer_max_length = 512;
45     }
46
47     /**
48      * Adds raw data bytes to the input buffer for processing.
49      * @param {Array} data - The raw data bytes to be processed.
50      */
51     bufferInData(data) {
52       if(data.length > 10000) {
53         this.buffer_in = this.buffer_in.concat(data);
54       } else {
55         this.buffer_in.push(...data);
56       }
57     }
58
59     /**
60      * Clears the input buffer and returns its contents as a string.
61      * @returns {string} The contents of the input buffer converted to a string.
62      */
63     showBuffer() {
64       var data_in = String.fromCharCode(...this.buffer_in.slice(this.
            buffer_in_read_ind));
65       this.buffer_in = [];
66       this.buffer_in_read_ind = 0;
67       this.discarded_buffer = [];
68       return data_in;
69     }
70
71     /**
72      * Returns the current contents of the input buffer as a string without
            clearing it.
73      * @returns {string} The current contents of the input buffer converted to a
            string.
74      */
75     peekBuffer() {
76       var data_in = String.fromCharCode(...this.buffer_in.slice(this.
            buffer_in_read_ind));
77       return data_in;
78     }
79
80     /**
81      * Returns the number of bytes available in the input buffer.
```

```
82     * @returns {number} The number of bytes available to read from the input
           buffer.
83     */
84    availableData() {
85      return this.buffer_in.length - this.buffer_in_read_ind;
86    }
87
88    /**
89     * Attempts to read a complete packet from the input buffer based on the
           protocol framing markers.
90     * @param {number} N_in - The expected number of data bytes in the packet,
           excluding markers and checksum.
91     * @returns {Array} An array containing the decoded data packet (if available)
            and a flag indicating if a packet was successfully read.
92     */
93    readPacket(N_in) {
94      if(!Number.isFinite(N_in)) {
95        console.error("Packet length must be a finite number.");
96      }
97      var flag_in = 0;
98      var data_in = [];
99      var p = 0;
100     if(this.buffer_in.length > this.buffer_in_read_ind) {
101       for(p = this.buffer_in_read_ind; p < this.buffer_in.length; p++) {
102         var b = this.buffer_in[p];
103         this.kr = this.kr + 1;
104         if(b == this.M1) {
105           // Packet start
106           this.k = 1;
107           this.kr = 1;
108         } else if(b == this.M2) {
109           if(this.k == (N_in+3)) {
110             // Packet end
111             var cks = this.getCKS(this.data_in_temp, N_in);
112             if(cks[0] == this.cks_in[0] && cks[1] == this.cks_in[1]) {
113               flag_in = 1;
114               data_in = this.data_in_temp.slice(0, N_in);
115
116               this.discarded_buffer = this.discarded_buffer.slice(0, -this.kr);
117               if(this.discarded_buffer.length > 0) {
118                 this.discardedCallback(String.fromCharCode(...this.
                      discarded_buffer));
119                 this.discarded_buffer = [];
120               }
121               p++; // Move to next byte after packet end
122               break;
123             }
124           }
125           this.k = 1;
126         } else if(this.k < (N_in+3) && this.k > 0) {
127           // Packet payload or checksum
128           if(b == this.M3) {
129             // Encoded value
130             this.encode_flag = 1;
131           } else {
132             if(this.encode_flag) {
133               // Decoding value
134               this.encode_flag = 0;
135               if(b == 3) {
```

```
136                 b = this.M3;
137               } else if(b == 2) {
138                 b = this.M2;
139               } else if(b == 1) {
140                 b = this.M1;
141               }
142             }
143
144             if(this.k < (N_in+1)) {
145               this.data_in_temp[this.k-1] = b;
146             } else {
147               this.cks_in[this.k-N_in-1] = b;
148             }
149             this.k = this.k+1;
150           }
151         }
152
153         this.discarded_buffer.push(b);
154         if(this.discarded_buffer.length > this.discarded_buffer_max_length) {
155           this.discardedCallback(String.fromCharCode(...this.discarded_buffer));
156           this.discarded_buffer = [];
157         }
158       }
159       this.buffer_in_read_ind = p; // Update read index
160       // Periodically clean up the buffer to prevent it from growing
          indefinitely
161       if(this.buffer_in_read_ind >= this.buffer_in.length) {
162         // All data has been processed; reset the buffer
163         this.buffer_in = [];
164         this.buffer_in_read_ind = 0;
165       } else if(this.buffer_in_read_ind > this.max_buffer_size) {
166         // Remove processed data
167         this.buffer_in = this.buffer_in.slice(this.buffer_in_read_ind);
168         this.buffer_in_read_ind = 0;
169       }
170     }
171     return [data_in, flag_in];
172   }
173
174   /**
175    * Encodes a packet of data for transmission, including framing markers and
          checksum.
176    * @param {Array} data_out - The data packet to encode for transmission.
177    * @param {number} N_out - The number of data bytes in the packet.
178    * @returns {Array} The encoded data packet ready for transmission.
179    */
180   sendPacket(data_out, N_out) {
181     var cks = this.getCKS(data_out, N_out);
182     var data_out_temp = [...data_out, ...cks];
183     var buffer_out = [];
184     var p = 0;
185     buffer_out[p] = this.M1;
186     for(var i = 0; i < N_out+2; i++) {
187       var b = data_out_temp[i];
188       p = p+1;
189       if(b == this.M3) {
190         buffer_out[p] = this.M3;
191         p = p+1;
192         buffer_out[p] = 3;
```

```
193         } else if(b == this.M2) {
194           buffer_out[p] = this.M3;
195           p = p+1;
196           buffer_out[p] = 2;
197         } else if(b == this.M1) {
198           buffer_out[p] = this.M3;
199           p = p+1;
200           buffer_out[p] = 1;
201         } else {
202           buffer_out[p] = b;
203         }
204       }
205       p = p+1;
206       buffer_out[p] = this.M2;
207       return buffer_out;
208     }
209
210     /**
211      * Calculates the checksum for a given array of data bytes.
212      * @param {Array} data - The data bytes for which to calculate the checksum.
213      * @param {number} N - The number of bytes in the data array.
214      * @returns {Array} A two-element array containing the calculated checksum
           bytes.
215      */
216     getCKS(data, N) {
217       var sum1 = 0;
218       var sum2 = 0;
219       for(var i = 0; i < N; i++){
220         sum1 = sum1 + data[i];
221         sum2 = sum2 + sum1;
222       }
223       return [sum1 % 256, sum2 % 256];
224     }
225
226     /**
227      * Clears the input buffer, discarding any unread data.
228      */
229     flushInput() {
230       this.buffer_in = [];
231       this.buffer_in_read_ind = 0;
232     }
233
234     /**
235      * Executes a callback or logs a string when data is discarded.
236      * @param {string} string - The discarded data as a string.
237      */
238     discardedCallback(string) {
239       if(typeof this.onDiscardedData === 'function') {
240         this.onDiscardedData(string);
241       } else {
242         console.log(string);
243       }
244     }
245 }
246
247 exports.PRDC_SCBP_CODEC = PRDC_SCBP_CODEC;
```

Listing T.1 - PRDC_SCBP_CODEC.js

```
1 /**
```

```
 * @file Serial Communication Binary Protocol library
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

var codec_path = "./../PRDC_SCBP_CODEC/PRDC_SCBP_CODEC";
var backend_web_path = "./PRDC_SCBP_WEB";
var backend_node_path = "./PRDC_SCBP_NODE";
if(typeof getInternalModulePath == "function") {
  codec_path = getInternalModulePath('PRDC_SCBP_CODEC', 'lib/PRDC_SCBP_CODEC');
  backend_web_path = getInternalModulePath('PRDC_SCBP_WEB', 'lib/PRDC_SCBP');
  backend_node_path = getInternalModulePath('PRDC_SCBP_NODE', 'lib/PRDC_SCBP');
}
const { PRDC_SCBP_CODEC } = require(codec_path);

if(config && hasKey(config, 'SCBP_BACKEND')) {
  if(config.SCBP_BACKEND == 'web') {
    var { PRDC_SCBP_BACKEND_WEB } = require(backend_web_path);
  } else if(config.SCBP_BACKEND == 'node') {
    var { PRDC_SCBP_BACKEND_NODE } = require(backend_node_path);
  }
} else {
  var { PRDC_SCBP_BACKEND_NODE } = require(backend_node_path);
}

/**
 * Class for Serial Communication Binary Protocol.
 */
class PRDC_SCBP {

  /**
   * Initializes the protocol handler, preparing it for communication over a
      serial port.
   */
  constructor() {
    this.codec; // codec object
    this.backend; // backend object

    this.disconnectCallback = function() {};
    this.connectCallback = function() {};

    if(config && hasKey(config, 'SCBP_BACKEND')) {
      if(config.SCBP_BACKEND == 'web') {
        this.backend = new PRDC_SCBP_BACKEND_WEB(this);
      } else if(config.SCBP_BACKEND == 'node') {
        this.backend = new PRDC_SCBP_BACKEND_NODE(this);
      }
    } else {
      this.backend = new PRDC_SCBP_BACKEND_NODE(this);
    }
  }

  /**
   * Initializes serial communication with the specified port and baud rate, and
      sets up callbacks.
   * @param {string} port - The path to the serial port.
   * @param {number} baudrate - The baud rate for the serial communication.
```

```
59      * @param {Function} [callback] - Optional callback function for handling
            errors during port initialization.
60      */
61     begin(...args) {
62       // SCBP CODEC
63       this.codec = new PRDC_SCBP_CODEC();
64       this.codec.begin();
65
66       this.backend.begin(...args);
67     }
68
69     /**
70      * Retrieves and clears the current data buffer, returning its contents.
71      * @returns {string} The contents of the data buffer as a string.
72      */
73     showBuffer() {
74       return this.codec.showBuffer();
75     }
76
77     /**
78      * Peeks at the current data buffer contents without clearing it.
79      * @returns {string} The contents of the data buffer as a string.
80      */
81     peekBuffer() {
82       return this.codec.peekBuffer();
83     }
84
85     /**
86      * Checks the amount of available data in the buffer.
87      * @returns {number} The number of bytes available in the buffer.
88      */
89     availableData() {
90       return this.codec.availableData();
91     }
92
93     /**
94      * Checks if the serial port is currently open.
95      * @returns {boolean} True if the port is open, false otherwise.
96      */
97     isOpen() {
98       return this.backend.isOpen();
99     }
100
101    /**
102     * Reads a packet of data from the buffer, attempting to decode it according
            to the SCBP.
103     * @param {number} N_in - The expected size of the packet to read.
104     * @returns {Array} An array containing the decoded packet data, if available.
105     */
106    readPacket(N_in) {
107      return this.codec.readPacket(N_in);
108    }
109
110    /**
111     * Encodes and sends a packet of data over the serial port.
112     * @param {Array} data_out - The packet data to send.
113     * @param {number} N_out - The size of the packet.
114     */
115    sendPacket(data_out, N_out) {
```

```
116       this.backend.sendPacket(data_out, N_out);
117     }
118
119     /**
120      * Clears the input buffer, discarding any unread data.
121      */
122     flushInput() {
123       this.codec.flushInput();
124     }
125
126     /**
127      * Closes the serial port connection.
128      */
129     close() {
130       this.backend.close();
131     }
132
133     /**
134      * Sets the callback function to execute when the serial port is disconnected.
135      * @param {Function} callback - The callback function to execute on
           disconnection.
136      */
137     setDisconnectCallback(callaback) {
138       this.disconnectCallback = callaback;
139     }
140
141     /**
142      * Sets the callback function to execute when the serial port is successfully
           connected.
143      * @param {Function} callback - The callback function to execute on connection
           .
144      */
145     setConnectCallback(callaback) {
146       this.connectCallback = callaback;
147     }
148
149     /**
150      * Sets a callback function to handle discarded data from a codec.
151      * @param {Function} callback - The function to execute when data is discarded
           .
152      * @param {number} [discarded_buffer_max_length=512] - Maximum length of the
           buffer for discarded data.
153      */
154     setDiscardedDataCallback(callback, discarded_buffer_max_length = 512) {
155       this.codec.onDiscardedData = callback;
156       this.codec.discarded_buffer_max_length = discarded_buffer_max_length;
157     }
158 }
159
160 exports.PRDC_SCBP = PRDC_SCBP;
```

Listing T.2 - PRDC_SCBP.js

```
1  /**
2   * @file Serial Communication Binary Protocol library backend NODE
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
```

```
8
9  var { SerialPort } = global.require('serialport');
10
11 /**
12  * Class for Serial Communication Binary Protocol backend NODE.
13  */
14 class PRDC_SCBP_BACKEND_NODE {
15
16   /**
17    * Initializes the protocol handler, preparing it for communication over a
          serial port.
18    */
19   constructor(protocol) {
20     this.protocol = protocol;
21     this.port; // serial port
22     this.baudrate; // serial baud rate
23     this.s; // serial object
24     this.begin_flag = false; // flag for begin
25   }
26
27   /**
28    * Initializes serial communication with the specified port and baud rate, and
          sets up callbacks.
29    * @param {string} port - The path to the serial port.
30    * @param {number} baudrate - The baud rate for the serial communication.
31    * @param {Function} [callback] - Optional callback function for handling
          errors during port initialization.
32    */
33   begin(...args) {
34     this.port = args[0];
35     this.baudrate = args[1];
36     var callback;
37     if(args.length > 2) {
38       callback = args[2];
39     }
40     this.begin_flag = true;
41
42     this.s = new SerialPort({path: this.port,
43        baudRate: this.baudrate,
44        dataBits: 8,
45        parity: 'none',
46        stopBits: 1,
47        flowControl: false
48     }, function(err) {
49       if(err) {
50         if(typeof callback === 'function') {
51           callback(err);
52         } else {
53           console.log('Error: ' + err.message);
54         }
55       }
56     });
57
58     var obj = this;
59     this.s.on('open', function() {
60       try {
61         obj.s.set({
62           dtr: true,
63           rts: false
```

```
64          });
65          obj.protocol.connectCallback();
66        } catch(err) {
67          if(typeof callback === 'function') {
68            callback(err);
69          } else {
70            console.log('Error: ' + err.message);
71          }
72        }
73      });
74      this.s.on('data', function(data) {
75        obj.protocol.codec.bufferInData(data);
76      });
77      this.s.on('error', function(err) {
78        if(typeof callback === 'function') {
79          callback(err);
80        } else {
81          console.log('Error: ' + err.message);
82        }
83      });
84      this.s.on('close', function(err) {
85        obj.protocol.disconnectCallback(err);
86      });
87    }
88
89    /**
90     * Checks if the serial port is currently open.
91     * @returns {boolean} True if the port is open, false otherwise.
92     */
93    isOpen() {
94      return this.begin_flag && this.s.isOpen;
95    }
96
97    /**
98     * Encodes and sends a packet of data over the serial port.
99     * @param {Array} data_out - The packet data to send.
100    * @param {number} N_out - The size of the packet.
101    */
102   sendPacket(data_out, N_out) {
103     this.s.write(this.protocol.codec.sendPacket(data_out, N_out));
104   }
105
106   /**
107    * Closes the serial port connection.
108    */
109   close() {
110     try {
111       this.s.close();
112     } catch(e) {};
113   }
114 }
115
116 exports.PRDC_SCBP_BACKEND_NODE = PRDC_SCBP_BACKEND_NODE;
```

Listing T.3 - PRDC_SCBP_NODE.js

```
1  /**
2   * @file Serial Communication Binary Protocol library backend WEB
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
```

```
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8
9   /**
10  * Class for Serial Communication Binary Protocol backend WEB.
11  */
12  class PRDC_SCBP_BACKEND_WEB {
13
14    /**
15     * Initializes the protocol handler, preparing it for communication over a
          serial port.
16     */
17    constructor(protocol) {
18      this.protocol = protocol;
19      this.device; // serial port
20      this.baudrate; // serial baud rate
21      this.s; // serial object
22      this.reader; // reader object
23      this.writer; // writer object
24      this.begin_flag = false; // flag for begin
25      this.active = false;
26    }
27
28    /**
29     * Initializes serial communication with the specified port and baud rate, and
           sets up callbacks.
30     * @param {string} port - The path to the serial port.
31     * @param {number} baudrate - The baud rate for the serial communication.
32     * @param {Function} [callback] - Optional callback function for handling
          errors during port initialization.
33     */
34    async begin(...args) {
35      var obj = this;
36      this.device = args[0];
37      this.baudrate = args[1];
38      var callback;
39      if(args.length > 2) {
40        callback = args[2];
41      }
42
43      if(Object.hasOwn(this.device, 'vid') && Object.hasOwn(this.device, 'pid') &&
           isNumeric(this.baudrate)) {
44        try {
45          var ports = await navigator.serial.getPorts();
46          obj.s = ports.filter(function(p) {
47            var info = p.getInfo();
48            return info.usbVendorId == obj.device.vid && info.usbProductId == obj.
                 device.pid;
49          });
50          if(!(obj.s && obj.s.length)) {
51            if(typeof callback === 'function') {
52              callback('Device not found!');
53            } else {
54              console.log('Error: Device not found!');
55            }
56          } else {
57            obj.s = obj.s[0];
58            await obj.s.open({ baudRate: obj.baudrate });
```

```
59            await obj.s.setSignals({
60              dataTerminalReady: true,
61              requestToSend: false
62            });
63
64            obj.reader = obj.s.readable.getReader();
65            obj.writer = obj.s.writable.getWriter();
66
67            obj.active = true;
68            obj.begin_flag = true;
69            obj.protocol.connectCallback();
70
71            obj.readLoop();
72          }
73        } catch (err) {
74          if (typeof callback === 'function') {
75            callback(err);
76          } else {
77            console.log('Error: ' + err.message);
78          }
79        }
80      } else {
81        if(typeof callback === 'function') {
82          callback('Device selection failed!');
83        } else {
84          console.log('Error: Device selection failed!');
85        }
86      }
87    }
88
89    /**
90     * Continuous reading from the serial port.
91     */
92    async readLoop() {
93      var obj = this;
94      while(this.s.readable && this.active) {
95        try {
96          const { value, done } = await obj.reader.read();
97          if(done) {
98            break;
99          }
100         if(value) {
101           obj.protocol.codec.bufferInData(value);
102         }
103       } catch(err) {
104         obj.protocol.disconnectCallback(err);
105         obj.close();
106       }
107     }
108
109     this.reader.releaseLock();
110   }
111
112   /**
113    * Checks if the serial port is currently open.
114    * @returns {boolean} True if the port is open, false otherwise.
115    */
116   isOpen() {
117     return this.begin_flag && this.s && this.s.readable;
```

```
118   }
119
120   /**
121    * Encodes and sends a packet of data over the serial port.
122    * @param {Array} data_out - The packet data to send.
123    * @param {number} N_out - The size of the packet.
124    */
125   async sendPacket(data_out, N_out) {
126     var obj = this;
127     try {
128       obj.writer.write(new Uint8Array(obj.protocol.codec.sendPacket(data_out,
          N_out)));
129     } catch (err) {
130       obj.protocol.disconnectCallback(err);
131       obj.close();
132     }
133   }
134
135   /**
136    * Closes the serial port connection.
137    */
138   async close() {
139     this.active = false;
140     if(this.reader) {
141       try {
142         await this.reader.cancel();
143       } catch(err) {}
144       this.reader.releaseLock();
145     }
146     if(this.writer) {
147       this.writer.releaseLock();
148     }
149     if(this.s) {
150       try {
151         await this.s.close();
152       } catch(err) {}
153     }
154   }
155 }
156
157 exports.PRDC_SCBP_BACKEND_WEB = PRDC_SCBP_BACKEND_WEB;
```

Listing T.4 - PRDC_SCBP_WEB.js

```
1  /**
2   * @file Example for Serial Communication Binary Protocol library
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8
9  const { ipcRenderer } = require('electron');
10
11 // Toggle the DevTools
12 document.addEventListener('keydown', function (e) {
13   if (e.which === 123) {
14     ipcRenderer.send('toggle-dev-tools');
15   }
16 });
```

```
17
18 const COM = require('./PRDC_SCBP');
19 const P = require('./PRDC_PACKET');
20
21 var packetStruct = [
22    ['UInt16LE','p1'],['UInt32LE','p2'],['Int16LE','p36',4]
23 ];
24 P.makePacket(packetStruct);
25 var inFun = P.getInputCast();
26 var outFun = P.getOutputCast();
27 var N = P.bytes; // length of packet payload
28
29 var PORT;
30 var baudrate = 115200;
31 if (process.platform == 'win32') {
32   PORT = 'COM5';
33 } else {
34   PORT = '/dev/serial0';
35 }
36
37 COM.begin(PORT, baudrate);
38
39 var i = 0;
40 setInterval(function(){
41   i = i+1;
42   if(i > 255) {
43     i = 0;
44   }
45   COM.sendPacket(outFun([i, 2, 3, 4, 5, 1000]), N);
46 }, 1000);
47
48 setInterval(function(){
49   while(COM.availableData() > 10) {
50     var [data_in, flag_in] = COM.readPacket(N);
51     console.log('dataIn = ['+inFun(data_in)+']');
52     console.log('flagIn = '+flag_in);
53   }
54 }, 1000);
```

Listing T.5 - `example_PRDC_SCBP.js`

T.2 Octave implementacija

U okviru listinga T.7 je dat primer za korišćenje, dok je `Octave` implementacija data u okviru T.6.

```
1  % Serial Communication Binary Protocol library
2  % @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  % Doktorska disertacija
4  % Mašinski fakultet Univerziteta u Beogradu
5  % --------------------
6
7  classdef PRDC_SCBP_CLASS < handle
8     properties (Access = private)
9        PORT; % serial port
10       baudrate; % serial baud rate
11       M1; % start bit - marker (<)
12       M2; % stop bit - marker (>)
```

```
13        M3; % encode bit - marker (_)
14        s; % serial object
15        k; % counter
16        encodeFlag; % encode flag
17        cksIn; % input checksum
18        bufferIn; % input buffer
19        dataInTemp; % temporary input data vector
20    end
21    methods
22        function obj = begin(obj, PORT, baudrate)
23          % PORT - Serial port
24          % baudrate - Serial baud rate
25          obj.PORT = PORT;
26          obj.baudrate = baudrate;
27
28          obj.M1 = '<';
29          obj.M2 = '>';
30          obj.M3 = '_';
31          obj.k = 0;
32          obj.encodeFlag = 0;
33          obj.cksIn = zeros(1, 2);
34          obj.dataInTemp = [];
35          obj.bufferIn = [];
36
37          if(isOctave)
38            obj.s = serial(["\\\\.\\" obj.PORT], obj.baudrate, 5);
39            set(obj.s, "dataterminalready", "On") % Enables DTR line
40            set(obj.s, "requesttosend", "Off") % Disable RTS line
41          else
42            obj.s = serial(obj.PORT, 'BaudRate', obj.baudrate, 'TimeOut', 5);
43          end
44          fopen(obj.s);
45        end
46
47        function N = availableData(obj)
48          Nin = get(obj.s, 'BytesAvailable');
49          if(Nin > 0)
50            if(isOctave)
51              data = srl_read(obj.s, Nin);
52              obj.bufferIn = [obj.bufferIn, data];
53            else
54              obj.bufferIn = [obj.bufferIn; fread(obj.s, Nin, 'uint8')];
55            end
56          end
57          N = length(obj.bufferIn);
58        end
59
60        function [dataIn] = showBuffer(obj)
61          dataIn = char(obj.bufferIn);
62          obj.bufferIn = [];
63        end
64
65        function [dataIn, flagIn] = readPacket(obj, Nin)
66          flagIn = 0;
67          dataIn = zeros(1, Nin);
68          if(obj.availableData > 0)
69            for p = 1:length(obj.bufferIn)
70              b = obj.bufferIn(p);
71              if(b == obj.M1)
```

```
72              obj.k = 1;
73            elseif(b == obj.M2)
74             if(obj.k == (Nin+3))
75                 cks = obj.getCKS(obj.dataInTemp(1:Nin), Nin);
76                 if(cks(1) == obj.cksIn(1) && cks(2) == obj.cksIn(2))
77                    flagIn = 1;
78                    dataIn(1:Nin) = uint8(obj.dataInTemp(1:Nin));
79                     break;
80                 end
81             end
82             obj.k = 1;
83            elseif(b == obj.M3)
84              obj.encodeFlag = 1;
85            elseif(obj.k < (Nin+3) && obj.k > 0)
86              if(obj.encodeFlag)
87                obj.encodeFlag = 0;
88                if(b == 3)
89                  b = obj.M3;
90                elseif(b == 2)
91                  b = obj.M2;
92                elseif(b == 1)
93                  b = obj.M1;
94                end
95              end
96
97              if(obj.k < (Nin+1))
98                obj.dataInTemp(obj.k) = b;
99                obj.k = obj.k+1;
100             else
101               obj.cksIn(obj.k-Nin) = b;
102               obj.k = obj.k+1;
103             end
104           end
105         end
106         obj.bufferIn(1:p) = [];
107       end
108     end
109
110     function sendPacket(obj, dataOut, Nout)
111       cks = obj.getCKS(dataOut, Nout);
112       dataOutTemp = [dataOut, cks];
113       bufferOut = zeros(1, 256);
114       p = 1;
115       bufferOut(p) = obj.M1;
116       for i = 1:Nout+2
117         b = dataOutTemp(i);
118           p = p+1;
119         if(b == obj.M3)
120           bufferOut(p) = obj.M3;
121           p = p+1;
122           bufferOut(p) = 3;
123         elseif(b == obj.M2)
124           bufferOut(p) = obj.M3;
125           p = p+1;
126           bufferOut(p) = 2;
127         elseif(b == obj.M1)
128           bufferOut(p) = obj.M3;
129           p = p+1;
130           bufferOut(p) = 1;
```

```
131           else
132             bufferOut(p) = b;
133           end
134         end
135         p = p+1;
136         bufferOut(p) = obj.M2;
137         if(isOctave)
138           srl_write(obj.s, uint8(bufferOut(1:p)));
139         else
140           fwrite(obj.s, uint8(bufferOut(1:p)));
141         end
142       end
143
144       function CKS = getCKS(~, data, N)
145         CKS = zeros(1, 2);
146         for i = 1:N
147             CKS(1) = CKS(1) + data(i);
148             CKS(2) = CKS(2) + CKS(1);
149         end
150         CKS(1) = mod(CKS(1), 256);
151         CKS(2) = mod(CKS(2), 256);
152       end
153
154       function flushInput(obj)
155         obj.bufferIn = [];
156         flushinput(obj.s)
157       end
158
159       function close(obj)
160         fclose(obj.s);
161       end
162     end
163 end
```

Listing T.6 - PRDC_SCBP_CLASS.m

```
1  % Example program for PRDC_SCBP
2  % Serial Communication Binary Protocol library
3  % @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4  % Doktorska disertacija
5  % Mašinski fakultet Univerziteta u Beogradu
6  % --------------------
7
8  % Clear program
9  clear all, clc, close all
10
11 % Common functions path
12 lib = 'Z:\PRDC\PROGRAMI\_Octave_lib\';
13 addpath([lib 'PRDC_COMMON']);
14 addpath([lib 'PRDC_SCBP']);
15 addpath([lib 'PRDC_PACKET']);
16
17 % Get serial devices
18 if(isOctave)
19   pkg('load', 'instrument-control');
20 end
21 PORT = getSerialPort('Instrument');
22
23 % Protocol packet
24 packetStruct = {
```

```
    {'uint16', 'p1'}, {'uint32', 'p2'}, {'int16', 'p36', 4}
};
P = PRDC_PACKET_CLASS;
P.makePacket(packetStruct);
inFun = P.getInputCast();
outFun = P.getOutputCast();
N = P.bytes; % length of packet payload

% Define PRDC_SCBP protocol object
INST_COM = PRDC_SCBP_CLASS;
INST_COM.begin(PORT, 115200);

% Main loop exit button
fs = [165, 50];
ss = get(0, 'Screensize');
f = figure('Color', 'w', 'MenuBar', 'none', 'ToolBar', 'none', ...
    'Name', mfilename, 'NumberTitle', 'off', 'Resize', 'off', ...
    'Position', [(ss(3)-fs(1))/2, (ss(4)-fs(2))/2, fs(1), fs(2)]);
buttonHandle = uicontrol(f, 'Style', 'PushButton', ...
   'String', 'Stop loop', 'Callback', 'delete(gcbf)', ...
   'Position', [5, 5, 155, 45], 'FontSize', 14, 'FontName', 'Arial', ...
   'FontWeight', 'bold', 'ForegroundColor', [.3, .3, .3]);
drawnow

% Main loop
disp([' ----- ' mfilename ' ----- ']);
while(1)
  flagIn = 0;
  INST_COM.sendPacket(outFun([20000, 10, 10, 10, 10, 10]), N);
  while(INST_COM.availableData > 10) % read packets
    [dataIn, flagIn] = INST_COM.readPacket(N);
  end
  if(flagIn)
    disp([' dataIn = [' num2str(inFun(dataIn)) ']']);
  end

  drawnow
  if(~ishandle(buttonHandle))
    break
  end
end
disp(' --------------------');
delete(f);

% Close serial communication
INST_COM.close;
```

Listing T.7 - `example_PRDC_SCBP.m`

T.3 Arduino biblioteka

U okviru listinga T.10 je dat primer za korišćenje, dok je `Arduino` implementacija biblioteke data u okviru T.8 i T.9.

```
// Serial Communication Binary Protocol library
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
```

```
// --------------------

#ifndef _PRDC_SCBP_H_
#define _PRDC_SCBP_H_
#include "Arduino.h"

#define PRDC_SCBP_M1 0x3C // Start bit - marker (<)
#define PRDC_SCBP_M2 0x3E // Stop bit - marker (>)
#define PRDC_SCBP_M3 0x5F // Encode bit - marker (_)

#ifdef SCBP_MAX_PACKET_SIZE
  #define _SCBP_MAX_PACKET_SIZE SCBP_MAX_PACKET_SIZE
#endif
#ifndef SCBP_MAX_PACKET_SIZE
  #define _SCBP_MAX_PACKET_SIZE 256
#endif

#ifndef ARDUINO_ARCH_ESP32
  #include <SoftwareSerial.h>
#endif

class PRDC_SCBP {
  public:
    void begin(HardwareSerial& serial, unsigned long baudrate);
#ifdef RS485HD_STM32_AVAILABLE
    void begin(PRDC_RS485HD_STM32& serial, unsigned long baudrate){
      serial.begin(baudrate);
      begin((Stream&)serial);
    }
#endif
#if defined (USBCON) && defined(USBD_USE_CDC)
    void begin(USBSerial& serial, unsigned long baudrate);
#endif
#ifndef ARDUINO_ARCH_ESP32
    void begin(SoftwareSerial& serial, unsigned long baudrate);
#endif
    void begin(Stream& stream);
    void begin(unsigned long baudrate){
      Serial.begin(baudrate);
      begin((Stream&)Serial);
    }
    void begin() {
      begin((Stream&)Serial);
    }
    bool readPacket(uint8_t* dataIn, uint16_t Nin);
    void sendPacket(uint8_t* dataOut, uint16_t Nout);
    uint32_t availableData();
    void flushInput();

  private:
    Stream* _serial;
    uint8_t dataInTemp[_SCBP_MAX_PACKET_SIZE];  // data in buffer
    uint16_t k = 0;
    bool encodeFlag = 0;
    uint8_t cksIn[2];
    void getCKS(uint8_t* CKS, uint8_t* data, uint16_t N);
    bool decodePacket(uint8_t* dataIn, uint16_t Nin);
    void encodePacket(uint8_t* dataOut, uint16_t Nout);
};
```

```
64 #endif // _PRDC_SCBP_H_
```

Listing T.8 - PRDC_SCBP.h

```
1  // Serial Communication Binary Protocol library
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "PRDC_SCBP.h"
8
9  // begin() function
10 // Begin serial communication
11 // --------------------
12 void PRDC_SCBP::begin(Stream& stream) {
13         _serial = &stream;
14 }
15
16 void PRDC_SCBP::begin(HardwareSerial& serial, unsigned long baudrate){
17   serial.begin(baudrate);
18   begin((Stream&)serial);
19 }
20
21 #if defined (USBCON) && defined(USBD_USE_CDC)
22 void PRDC_SCBP::begin(USBSerial& serial, unsigned long baudrate){
23   serial.begin(baudrate);
24   begin((Stream&)serial);
25 }
26 #endif
27
28 #ifndef ARDUINO_ARCH_ESP32
29 void PRDC_SCBP::begin(SoftwareSerial& serial, unsigned long baudrate){
30   serial.begin(baudrate);
31   begin((Stream&)serial);
32 }
33 #endif
34
35 // getCKS() function
36 // Calculate Fletcher checksum
37 // --------------------
38 void PRDC_SCBP::getCKS(uint8_t* cks, uint8_t* data, uint16_t N) {
39   memset(cks, 0, 2);
40   for (uint16_t i = 0; i < N; i++) {
41     cks[0] += data[i];
42     cks[1] += cks[0];
43   }
44 }
45
46 // availableData() function
47 // Get number of available bytes
48 // --------------------
49 uint32_t PRDC_SCBP::availableData() {
50   return _serial->available();
51 }
52
53 // readPacket() function
54 // Read packet
55 // --------------------
56 bool PRDC_SCBP::readPacket(uint8_t* dataIn, uint16_t Nin) {
```

```
57    bool flagIn = 0;
58    if(_serial->available()) {
59      flagIn = decodePacket(dataIn, Nin);
60    }
61    return flagIn;
62  }
63
64  // decodePacket() function
65  // Decode packet
66  // --------------------
67  bool PRDC_SCBP::decodePacket(uint8_t* dataIn, uint16_t Nin) {
68    bool flagIn = 0;
69    uint8_t cks[2];
70
71    while(_serial->available()) {
72      uint8_t b = _serial->read();
73      if(b == PRDC_SCBP_M1) {
74        k = 0;
75      } else if(b == PRDC_SCBP_M2) {
76        if(k == (Nin+2)) {
77          getCKS(cks, dataInTemp, Nin);
78          if(cks[0] == cksIn[0] && cks[1] == cksIn[1]) {
79            flagIn = 1;
80            for (uint16_t i = 0; i < Nin; i++) {
81              dataIn[i] = dataInTemp[i];
82            }
83            break;
84          }
85        }
86        k = 0;
87      } else if(b == PRDC_SCBP_M3) {
88          encodeFlag = 1;
89      } else if(k < (Nin+2)) {
90        if(encodeFlag) {
91          encodeFlag = 0;
92          if(b == 0x03) {
93            b = PRDC_SCBP_M3;
94          } else if(b == 0x02) {
95            b = PRDC_SCBP_M2;
96          } else if(b == 0x01) {
97            b = PRDC_SCBP_M1;
98          }
99        }
100       if(k < Nin) {
101         ((uint8_t*)dataInTemp)[k] = b;
102         ++k;
103       } else {
104         cksIn[k-Nin] = b;
105         ++k;
106       }
107     }
108   }
109
110   return flagIn;
111 }
112
113 // sendPacket() function
114 // Send packet
115 // --------------------
```

```
116 void PRDC_SCBP::sendPacket(uint8_t* dataOut, uint16_t Nout) {
117   uint8_t cks[2];
118   uint8_t dataOutTemp[Nout+2];
119   for (uint16_t i = 0; i < Nout; i++) {
120     dataOutTemp[i] = dataOut[i];
121   }
122 
123   getCKS(cks,dataOut,Nout);
124   ((uint8_t*)dataOutTemp)[Nout] = cks[0];
125   ((uint8_t*)dataOutTemp)[Nout+1] = cks[1];
126 
127   _serial->write((byte)PRDC_SCBP_M1);
128   encodePacket(dataOutTemp,Nout);
129   _serial->write((byte)PRDC_SCBP_M2);
130 }
131 
132 // encodePacket() function
133 // Encode packet
134 // --------------------
135 void PRDC_SCBP::encodePacket(uint8_t* dataOut, uint16_t Nout) {
136   for (uint16_t i = 0; i < Nout+2; i++) {
137     uint8_t b = ((uint8_t*)dataOut)[i];
138     if(b == PRDC_SCBP_M1) {
139       _serial->write((byte)PRDC_SCBP_M3);
140       _serial->write((byte)0x01);
141     } else if (b == PRDC_SCBP_M2) {
142       _serial->write((byte)PRDC_SCBP_M3);
143       _serial->write((byte)0x02);
144     } else if (b == PRDC_SCBP_M3) {
145       _serial->write((byte)PRDC_SCBP_M3);
146       _serial->write((byte)0x03);
147     } else {
148       _serial->write(b);
149     }
150   }
151 }
152 
153 // flushInput() function
154 // Flush input data
155 // --------------------
156 void PRDC_SCBP::flushInput() {
157   while(_serial->available()) {
158     _serial->read();
159   }
160 }
```

Listing T.9 - PRDC_SCBP.cpp

```
1  // Test SCBP
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6 
7  // Library
8  // --------------------
9  // PRDC SCBP
10 // Author: PRDC
11 #include <PRDC_SCBP.h>
12 
```

```
13 // Define variables
14 // --------------------
15 #define SERIAL_BAUDRATE 115200
16 PRDC_SCBP PC_COM;
17 bool flag;
18
19 // Define packet
20 struct __attribute__((packed)) packet_t {
21   uint16_t p1;
22   uint32_t p2;
23   int16_t p36[4];
24 };
25 union dataOut_t{
26   packet_t p;
27   uint8_t bytes[sizeof(packet_t)];
28 } dataOut = {1,2,3,4,5,10000};
29 union dataIn_t{
30   packet_t p;
31   uint8_t bytes[sizeof(packet_t)];
32 } dataIn = {0,0,0,0,0,0};
33
34 // setup function
35 // --------------------
36 void setup() {
37   PC_COM.begin(Serial,SERIAL_BAUDRATE);
38 }
39
40 // loop function
41 // --------------------
42 void loop(){
43   delay(1000);
44   while(PC_COM.availableData() > 10) { // read packets
45     flag = PC_COM.readPacket(dataIn.bytes,sizeof(dataIn));
46   }
47   if(flag) {
48     dataOut.p.p1 = dataIn.p.p1;
49   }
50   PC_COM.sendPacket(dataOut.bytes,sizeof(dataOut));
51 }
```

Listing T.10 - `example_SCBP.ino`

U

Pomoćna biblioteka PRDC_PACKET

U listingu U.1 je data pomoćna biblioteka `PRDC_PACKET` koja se koristi kako bi se formirali paketi u programskom jeziku `JavaScript`, a za potrebe komunikacionog protokola `PRDC_SCBP`.

```
/**
 * @file Packet library
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

/**
 * Class for packet creation.
 */
class PRDC_PACKET {

  /**
   * Initializes a new packet structure.
   */
  constructor() {
    this.pl; // packet payload
    this.fn; // packet payload filednames
    this.N; // number of variables
    this.len; // packet length
    this.bytes; // number of bytes in packet
  }

  /**
   * Constructs a packet layout based on specified fields and their types.
   * @param {Array} A - Array defining packet fields and their attributes.
   */
  makePacket(A) {
    this.pl = {};
    this.N = A.length;
    this.fn = [];
    var n = 0, m = 0, q = 0, nb, ne;
    for(var i = 0; i < this.N; i++) {
      switch(A[i][0]) {
        case 'Int8':
        case 'Uint8':
          nb = 1;
          break;
        case 'Int16LE':
```

```
41          case 'UInt16LE':
42          case 'Int16BE':
43          case 'UInt16BE':
44            nb = 2;
45            break;
46          case 'FloatLE':
47          case 'Int32LE':
48          case 'UInt32LE':
49          case 'FloatBE':
50          case 'Int32BE':
51          case 'UInt32BE':
52            nb = 4;
53            break;
54          case 'DoubleLE':
55          case 'BigInt64LE':
56          case 'BigUInt64LE':
57          case 'DoubleBE':
58          case 'BigInt64BE':
59          case 'BigUInt64BE':
60            nb = 8;
61            break;
62          default:
63            console.error('Unknown numeric data type: ' + A[i][0]);
64            break;
65        }
66        if(A[i].length < 3) {
67          ne = 1;
68        } else {
69          ne = A[i][2];
70        }
71        this.pl[A[i][1]] = {'type':A[i][0],
72          'i':this.range(q,(q+ne)),
73          'j':this.range(n,(n+nb*ne)),
74          'bytes':nb,'N':ne};
75        q = q+ne;
76        n = n+nb*ne;
77        m = m+ne;
78      }
79      for(var fieldname in this.pl){
80        this.fn.push(fieldname);
81      }
82      this.bytes = n;
83      this.len = m;
84    }
85
86    /**
87     * Retrieves the index of a field within the packet.
88     * @param {...any} args - Field name and optionally the element index for
          array types.
89     * @returns {number|Array<number>} Field index or array of indices.
90     */
91    field(...args) {
92      var i;
93      if(args.length > 1) {
94        i = this.pl[args[0]].i;
95        i = i[args[1]];
96      } else {
97        i = this.pl[args[0]].i;
98      }
```

```
    return i;
  }

  /**
   * Extracts the data for a given field from a packet.
   * @param {Array} data - The packet data as an array.
   * @param {string} field_in - The field name.
   * @returns {Array} The extracted data for the specified field.
   */
  getFieldData(data, field_in) {
    var ind = this.field(field_in);
    return data.slice(ind[0], ind[ind.length-1]+1);
  }

  /**
   * Retrieves the raw byte indices for a field within the packet.
   * @param {...any} args - Field name and optionally the element index for
       array types.
   * @returns {Array<number>} Array of byte indices for the field.
   */
  fieldRaw(...args) {
    var j;
    if(args.length > 1) {
      j = this.pl[args[0]].j;
      j = j.slice((args[1]-1)*this.pl[args[0]].bytes+1,args[1]*this.pl[args[0]].
          bytes);
    } else {
      j = this.pl[args[0]].j;
    }
    return j;
  }

  /**
   * Generates a function to cast input data to the packet structure.
   * @returns {Function} A function that casts input to the structured packet
       data.
   */
  getInputCast() {
    var str = 'var buf = Buffer.from(x); return [';
    for(var i = 0; i < this.N; i++) {
      var bytes = this.pl[this.fn[i]].bytes;
      var type = this.pl[this.fn[i]].type;
      var j = this.pl[this.fn[i]].j;
      for(var p = 0; p < this.pl[this.fn[i]].N; p++) {
        var ind = j[p*bytes];
        str += 'buf.read'+type+'('+ ind + ')';
        if(p < (this.pl[this.fn[i]].N-1)) {
          str += ',';
        }
      }
      if(i < (this.N-1)) {
        str += ',';
      }
    }
    if(this.N > 0) {
      return new Function('x', str+']');
    } else {
      return new Function('x', 'return x');
    }
```

```
155   }
156
157   /**
158    * Generates a function to cast packet data to output format.
159    * @returns {Function} A function that converts structured packet data to
          output format.
160    */
161   getOutputCast() {
162     var str = 'var buf = Buffer.alloc('+this.bytes+'); ';
163     var m = 0;
164     for(var i = 0; i < this.N; i++) {
165       var bytes = this.pl[this.fn[i]].bytes;
166       var type = this.pl[this.fn[i]].type;
167       var j = this.pl[this.fn[i]].j;
168       for(var p = 0; p < this.pl[this.fn[i]].N; p++) {
169         var ind = j[p*bytes];
170         str += 'buf.write'+type+'(x[' + m + '],'+ ind + ');';
171         m++;
172       }
173
174     }
175     if(this.N > 0) {
176       return new Function('x', str+' return [...buf];');
177     } else {
178       return new Function('x', 'return x');
179     }
180   }
181
182   /**
183    * Creates an array or a single value representing a range.
184    * @param {number} start - The start of the range.
185    * @param {number} end - The end of the range.
186    * @returns {Array<number>|number} An array of numbers or a single number if
          start equals end.
187    */
188   range(start, end) {
189     if(start == end) {
190       return start;
191     }
192     return new Array(end - start).fill().map(function(d, i) { return i + start;
          });
193   }
194 }
195
196 if(typeof exports != 'undefined') {
197   exports.PRDC_PACKET = PRDC_PACKET;
198 } else {
199   global.PRDC_PACKET_CLASS = PRDC_PACKET;
200 }
```

Listing U.1 - PRDC_PACKET.js

V

FreeCAD radni prostor ExternalAPI

U poglavlju 6.1.2.2 je detaljno opisan novi *FreeCAD* radni prostor pod nazivom `ExternalAPI`, dok su u nastavku ovog priloga dati listinzi (V.1, V.2, V.3, V.4, V.5, V.6 i V.7) svih neophodnih datoteka za ovaj radni prostor.

Skripte su date u listinzima V.8, V.9, V.10 i V.11.

```
# -*- coding: utf-8 -*-
# This file is always executed when FreeCAD starts
# @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
# Doktorska disertacija
# Mašinski fakultet Univerziteta u Beogradu
# --------------------

"""@package Init
This file is always executed when FreeCAD starts. Currently there is no action.
"""
```

Listing V.1 - `Init.py`

```
# -*- coding: utf-8 -*-
# This file is executed when FreeCAD starts in GUI mode
# @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
# Doktorska disertacija
# Mašinski fakultet Univerziteta u Beogradu
# --------------------

"""@package InitGui
This file is executed when FreeCAD starts in GUI mode. It adds a preference page
    and checks if the program is ready and then runs TCP Server.
"""

import FreeCAD as App
import FreeCADGui as Gui
import resource_utils

# Add preference page
Gui.addPreferencePage(
    resource_utils.uiPath("eapi_prefs.ui"),"ExternalAPI"
    )
Gui.addIconPath(resource_utils.iconPath(""))

# Check if program is ready and run TCP Server
preferences = App.ParamGet("User parameter:BaseApp/Preferences/Mod/ExternalAPI")
if preferences.GetBool('runServer',True):
```

```
25   import check_ready
```

Listing V.2 - `InitGui.py`

```
1  # -*- coding: utf-8 -*-
2  # Checks if the program is ready to start the TCP server
3  # @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4  # Doktorska disertacija
5  # Mašinski fakultet Univerziteta u Beogradu
6  # --------------------
7
8  """@package check_ready
9  Checks if the program is ready to start the TCP server.
10 """
11
12 import FreeCAD as App
13 import FreeCADGui as Gui
14 from PySide2 import QtCore
15
16 mw = Gui.getMainWindow()
17
18 def onStart():
19     """!
20     Checks if workbenches are activated.
21     """
22     start = False
23     try:
24         mw.workbenchActivated
25         start = True
26     except AttributeError:
27         pass
28     if start:
29         timer.stop()
30         timer.deleteLater()
31         import tcp_server
32
33 def onPreStart():
34     """!
35     CChecks if the event loop is ready.
36     """
37     if App.Version()[1] < "17":
38         onStart()
39     else:
40         if mw.property("eventLoop"):
41             onStart()
42
43 timer = QtCore.QTimer()
44 timer.timeout.connect(onPreStart)
45 timer.start(500)
```

Listing V.3 - `check_ready.py`

```
1  # -*- coding: utf-8 -*-
2  # Utility functions for program resources
3  # @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4  # Doktorska disertacija
5  # Mašinski fakultet Univerziteta u Beogradu
6  # --------------------
7
8  """@package resource_utils
```

```
Utility functions for program resources.
"""

from os import path

resources_path = path.join(path.dirname(path.realpath(__file__)), 'Resources')
icons_path = path.join(resources_path, 'Icons')
ui_path = path.join(resources_path, 'UI')
scripts_path = path.join(resources_path, 'Scripts')

def iconPath(name):
    """!
    Function to obtain icon file path.

    """
    f = path.join(icons_path, name)

    return f

def uiPath(name):
    """!
    Function to obtain ui file path.

    """
    f = path.join(ui_path, name)

    return f

def scriptsPath(name):
    """!
    Function to obtain script file path.

    """
    f = path.join(scripts_path, name)

    return f
```

Listing V.4 - `resource_utils.py`

```
# -*- coding: utf-8 -*-
# Starts TCP Server and maintains communication
# @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
# Doktorska disertacija
# Mašinski fakultet Univerziteta u Beogradu
# --------------------

"""@package tcp_server
Starts TCP Server and maintains communication.
"""

import os
import random
import sys
import FreeCAD as App
from PySide2.QtCore import (Signal, QByteArray, QDataStream, QIODevice)
from PySide2.QtNetwork import (QHostAddress, QNetworkInterface, QTcpServer)
from message_parser import inputParser

# Time to wait in milliseconds - used to wait for incoming message etc.
WAIT_TIME_MS = 30000
```

```
22
23 class RunTCPServer():
24     """!
25     TCP Server class.
26     """
27     def __init__(self, parent=None):
28         """!
29         Starts TCP Server at specific address and port.
30         """
31         self.tcpServer = QTcpServer()
32         for ipAddress in QNetworkInterface.allAddresses():
33             if ipAddress != QHostAddress.LocalHost and ipAddress.toIPv4Address()
                    != 0:
34                 break
35         else:
36             ipAddress = QHostAddress(QHostAddress.LocalHost)
37
38         preferences = App.ParamGet("User parameter:BaseApp/Preferences/Mod/
               ExternalAPI")
39         port = int(preferences.GetString('serverPort'))
40         if not self.tcpServer.listen(QHostAddress.Any, port):
41             App.Console.PrintLog("@ExternalAPI: Unable to start TCP Server: %s.\
                   n" % self.tcpServer.errorString())
42             return
43
44         ipAddress = ipAddress.toString()
45         App.Console.PrintLog("@ExternalAPI: TCP Server is running on\nIP: %s\
               nport: %d\n\n" % (ipAddress, self.tcpServer.serverPort()))
46         self.tcpServer.newConnection.connect(self.communication)
47
48     def communication(self):
49         """
50         Maintains communication over TCP.
51         """
52         # Get a QTcpSocket from the QTcpServer
53         tcpSocket = self.tcpServer.nextPendingConnection()
54
55         # Wait for an incoming message
56         if not tcpSocket.waitForReadyRead(msecs=WAIT_TIME_MS):
57             App.Console.PrintError("@ExternalAPI: No request sent.\n")
58             tcpSocket.disconnectFromHost()
59             return
60
61         # Make an input data stream
62         instr = QDataStream(tcpSocket)
63         instr.setVersion(QDataStream.Qt_5_0)
64         instr.setByteOrder(QDataStream.BigEndian)
65
66         # Try to read the message size
67         if tcpSocket.bytesAvailable() < 2:
68             App.Console.PrintError("@ExternalAPI: Received message "
69                                    + "has too few bytes.\n")
70             tcpSocket.disconnectFromHost()
71             return
72         blockSize = instr.readUInt16()
73         App.Console.PrintLog("@ExternalAPI: Received message with payload size:
               "+str(blockSize)+"\n");
74
75         # Check message is sent complete
```

```
76          if tcpSocket.bytesAvailable() < blockSize:
77              App.Console.PrintError("@ExternalAPI: Received message has less
                    bytes "
78                                     + "then it's supposed to.\n")
79              tcpSocket.disconnectFromHost()
80              return
81
82          # Read message and inform about it
83          message = instr.readRawData(blockSize).decode("ascii")
84          App.Console.PrintLog("@ExternalAPI: Message in = "
85                               + message + "\n")
86
87          # Parse message and make response
88          response = inputParser(message)
89
90          # Prepare the data block to send back and inform about it
91          response_b = response.encode("ascii");
92          App.Console.PrintLog("@ExternalAPI: Response = " + response + " \n")
93          block = QByteArray(
94                  len(response_b).to_bytes(2, byteorder='big')
95                  + response_b)
96          outstr = QDataStream(block, QIODevice.WriteOnly)
97          outstr.setVersion(QDataStream.Qt_5_0)
98
99          # get the connection ready for clean up
100         tcpSocket.disconnected.connect(tcpSocket.deleteLater)
101
102         # now send the QByteArray.
103         tcpSocket.write(block)
104
105         # now disconnect connection.
106         tcpSocket.disconnectFromHost()
107
108 server = RunTCPServer()
```

Listing V.5 - `tcp_server.py`

```
1  # -*- coding: utf-8 -*-
2  # Message parser script
3  # @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4  # Doktorska disertacija
5  # Mašinski fakultet Univerziteta u Beogradu
6  # --------------------
7
8  """@package message_parser
9  Message parser script.
10 """
11 import FreeCAD as App
12 import FreeCADGui as Gui
13 import sys
14
15 def inputParser(message):
16     """!
17     Input message parser
18
19     @param message: Message to parse.
20
21     @return Returns a response message.
22     """
23     try:
```

```
24        params = message.split("|")
25        Nin = len(params);
26        if Nin >= 2:
27            if params[0] == "PING":
28                return "MSG|Server ready!"
29
30            elif params[0] == "CMD":
31                if params[1] == "QUIT":
32                    #Gui.getMainWindow().close()
33                    Gui.getMainWindow().deleteLater()
34                    return "MSG|FreeCAD closed!"
35
36                elif params[1] == "OPEN":
37                    if Nin == 3:
38                        from pathlib import Path
39                        if Path(params[2]).is_file():
40                            App.open(params[2])
41                            return "MSG|Document openend!"
42
43                        else:
44                           return "ERR|File does not exist!"
45
46                elif params[1] == "IMPORT":
47                    if Nin == 3:
48                        from pathlib import Path
49                        import ImportGui
50                        if Path(params[2]).is_file():
51                            ImportGui.open(params[2])
52                            return "MSG|Document openend!"
53
54                        else:
55                           return "ERR|File does not exist!"
56
57                elif params[1] == "NEW":
58                    if Nin == 3:
59                        document = App.newDocument(params[2])
60                    else:
61                        document = App.newDocument()
62
63                    App.setActiveDocument(document.Name)
64                    App.ActiveDocument=App.getDocument(document.Name)
65                    Gui.ActiveDocument=Gui.getDocument(document.Name)
66                    document.Label = document.Name
67                    return "DAT|"+document.Name
68
69                elif params[1] == "SAVE":
70                    document = App.ActiveDocument
71                    App.getDocument(document.Name).save()
72                    return "MSG|Document saved!"
73
74                elif params[1] == "SAVEAS":
75                    document = App.ActiveDocument
76                    App.getDocument(document.Name).saveAs(params[2])
77                    return "MSG|Document saved as stated!"
78
79                elif params[1] == "CLOSE":
80                    App.closeDocument(App.ActiveDocument.Name)
81                    return "MSG|Document closed!"
82
```

```
83          elif params[0] == "SCRIPT":
84              from pathlib import Path
85              from resource_utils import scriptsPath
86
87              if Path(params[1]).is_file():
88                  script_name = Path(params[1]).stem
89                  script_path = params[1]
90              elif Path(scriptsPath(params[1])+".py").is_file():
91                  script_name = params[1]
92                  script_path = scriptsPath(params[1])+".py"
93              else:
94                  return "ERR|Unknown script!"
95
96              import importlib
97              import time
98              t = time.process_time()
99
100             spec = importlib.util.spec_from_file_location(script_name,
                    script_path)
101             script = importlib.util.module_from_spec(spec)
102             sys.modules[script_name] = script
103             spec.loader.exec_module(script)
104
105             msg = script.init(params)
106             elapsed_time = time.process_time() - t
107             App.Console.PrintLog("@ExternalAPI: Elapsed time = "+str(
                    elapsed_time)+"\n")
108             return msg
109
110         elif params[0] == "EVAL":
111             if Nin == 2:
112                 exec(params[1])
113                 return "MSG|Command evaluated!"
114
115         elif params[0] == "MSR":
116             if params[1] == "A":
117                 objs = getObjects()
118                 A = 0
119                 for i in objs:
120                     A = A + i.Shape.Area
121                 return "DAT|"+str(A)
122
123             if params[1] == "V":
124                 objs = getObjects()
125                 V = 0
126                 for i in objs:
127                     V = V + i.Shape.Volume
128                 return "DAT|"+str(V)
129
130     return "ERR|Unknown command!"
131
132 except Exception as e:
133     exc_type, exc_obj, exc_tb = sys.exc_info()
134     App.Console.PrintLog("@ExternalAPI: Error = "+str(e)+"\n")
135     App.Console.PrintError("    Error line number = "+str(exc_tb.tb_lineno)+
            "\n")
136     return "ERR|"+str(e)
137
138 def getObjects():
```

```
    """!
    Visible objects selector

    @return Returns visible objects.
    """
    Gui.Selection.clearSelection()
    for obj in App.ActiveDocument.Objects:
        if obj.ViewObject.isVisible():
            Gui.Selection.addSelection(obj)
    return Gui.Selection.getSelection()
```

Listing V.6 - message_parser.py

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Gui::Dialog::DlgSettingsExternalAPI</class>
 <widget class="QWidget" name="Gui::Dialog::DlgSettingsExternalAPI">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>452</width>
    <height>589</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>ExternalAPI settings</string>
  </property>
  <layout class="QGridLayout" name="gridLayout1">
   <property name="margin">
    <number>9</number>
   </property>
   <property name="spacing">
    <number>6</number>
   </property>
   <item row="2" column="0">
    <spacer>
     <property name="orientation">
      <enum>Qt::Vertical</enum>
     </property>
     <property name="sizeHint" stdset="0">
      <size>
       <width>20</width>
       <height>40</height>
      </size>
     </property>
    </spacer>
   </item>
   <item row="0" column="0">
    <widget class="QGroupBox" name="groupBox_1">
     <property name="title">
      <string>Server settings</string>
     </property>
     <layout class="QGridLayout" name="gridLayout_1">
      <item row="0" column="0">
       <widget class="Gui::PrefCheckBox" name="checkBox_1">
        <property name="toolTip">
         <string>Run TCP Server of ExternalAPI on start up.</string>
        </property>
        <property name="text">
```

```
48          <string>Run server</string>
49         </property>
50         <property name="checked">
51          <bool>true</bool>
52         </property>
53         <property name="prefEntry" stdset="0">
54          <cstring>runServer</cstring>
55         </property>
56         <property name="prefPath" stdset="0">
57          <cstring>Mod/ExternalAPI</cstring>
58         </property>
59        </widget>
60       </item>
61       <item row="1" column="0">
62        <layout class="QGridLayout" name="gridLayout_2">
63         <item row="0" column="0">
64          <widget class="QLabel" name="label_1">
65           <property name="text">
66            <string>Server port: </string>
67           </property>
68          </widget>
69         </item>
70         <item row="0" column="1">
71          <widget class="Gui::PrefLineEdit" name="lineEdit_1">
72           <property name="toolTip">
73            <string>Port for TCP Server of ExternalAPI</string>
74           </property>
75           <property name="text">
76            <string>11077</string>
77           </property>
78           <property name="prefEntry" stdset="0">
79            <cstring>serverPort</cstring>
80           </property>
81           <property name="prefPath" stdset="0">
82            <cstring>Mod/ExternalAPI</cstring>
83           </property>
84          </widget>
85         </item>
86        </layout>
87       </item>
88      </layout>
89     </widget>
90    </item>
91    <item row="1" column="0">
92      <widget class="QLabel" name="label_2">
93       <property name="text">
94        <string>Please restart FreeCAD for changes to take effect!</string>
95       </property>
96      </widget>
97    </item>
98   </layout>
99  </widget>
100  <layoutdefault spacing="6" margin="11"/>
101  <customwidgets>
102   <customwidget>
103    <class>Gui::PrefLineEdit</class>
104    <extends>QLineEdit</extends>
105    <header>Gui/PrefWidgets.h</header>
106   </customwidget>
```

```
  <customwidget>
   <class>Gui::PrefCheckBox</class>
   <extends>QCheckBox</extends>
   <header>Gui/PrefWidgets.h</header>
  </customwidget>
 </customwidgets>
 <resources/>
 <connections/>
</ui>
```

Listing V.7 - `eapi_prefs.ui`

```
# -*- coding: utf-8 -*-
# Script to create cylindrical beams
# @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
# Doktorska disertacija
# Mašinski fakultet Univerziteta u Beogradu
# --------------------

"""@package MakeBeams
Script to create cylindrical beams.
"""

def init(params):
    """!
    Script main function.
    @param params: Param params[2] is a file with data.
    @return Returns a response message.
    """
    if len(params) == 3:
        import os.path
        if os.path.isfile(params[2]):
            import FreeCAD as App
            import FreeCADGui as Gui
            from math import pi
            import json

            with open(params[2]) as f:
                data = json.load(f)
                if len(data["Diameters"]) == 1 or len(data["Diameters"]) == len(
                    data["Coordinates"]):
                    if len(data["Diameters"]) == 1:
                        flag = 1
                    else:
                        flag = 0
                    vz = App.Base.Vector(0,0,1)
                    for idx, nodes in enumerate(data["Coordinates"]):
                        c = App.ActiveDocument.addObject("Part::Cylinder","
                            Cylinder")
                        n1 = App.Base.Vector(nodes[0:3])
                        n2 = App.Base.Vector(nodes[3:6])
                        vb = n2-n1
                        c.Placement.Base = n1
                        if(vz.getAngle(vb) == pi):
                            c.Placement.Rotation.Axis = App.Base.Vector(1,0,0)
                            c.Placement.Rotation.Angle = pi
                        else:
                            c.Placement.Rotation.Axis = vz.cross(vb)
                            c.Placement.Rotation.Angle = vz.getAngle(vb)

```

```
                        c.Height = vb.Length
                        if flag:
                            c.Radius = data["Diameters"][0]/2
                        else:
                            c.Radius = data["Diameters"][idx]/2

                    App.ActiveDocument.recompute()
                    Gui.SendMsgToActiveView("ViewFit")
                    return "MSG|Beams created!"

                return "ERR|Data dimensions mismatch!"

        return "ERR|File does not exist!"

    return "ERR|Argument count mismatch!"
```

Listing V.8 - MakeBeams.py

```
# -*- coding: utf-8 -*-
# Script to apply fusion command to all visible objects
# @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
# Doktorska disertacija
# Mašinski fakultet Univerziteta u Beogradu
# --------------------

"""@package MakeFusion
Script to apply fusion command to all visible objects.
"""

def init(params):
    """!
    Script main function.
    @return Returns a response message.
    """
    import FreeCAD as App
    import FreeCADGui as Gui

    Gui.Selection.clearSelection()
    for obj in App.ActiveDocument.Objects:
        if obj.ViewObject.isVisible():
            Gui.Selection.addSelection(obj)
    objs = Gui.Selection.getSelection()

    f = App.activeDocument().addObject("Part::MultiFuse","Fusion")
    f.Shapes = objs
    App.ActiveDocument.recompute()
    return "MSG|Fusion created!"
```

Listing V.9 - MakeFusion.py

```
# -*- coding: utf-8 -*-
# Script to create spherical nodes
# @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
# Doktorska disertacija
# Mašinski fakultet Univerziteta u Beogradu
# --------------------

"""@package MakeNodes
Script to create spherical nodes.
"""
```

```
11
12 def init(params):
13     """!
14     Script main function.
15     @param params: Param params[2] is a file with data.
16     @return Returns a response message.
17     """
18     if len(params) == 3:
19         import os.path
20         if os.path.isfile(params[2]):
21             import FreeCAD as App
22             import FreeCADGui as Gui
23             import json
24
25             with open(params[2]) as f:
26                 data = json.load(f)
27                 if len(data["Diameters"]) == 1 or len(data["Diameters"]) == len(
                       data["Coordinates"]):
28                     if len(data["Diameters"]) == 1:
29                         flag = 1
30                     else:
31                         flag = 0
32
33                     for idx, node in enumerate(data["Coordinates"]):
34                         s = App.ActiveDocument.addObject("Part::Sphere","Sphere"
                               )
35                         s.Placement.Base = node
36                         if flag:
37                             s.Radius = data["Diameters"][0]/2
38                         else:
39                             s.Radius = data["Diameters"][idx]/2
40
41                     App.ActiveDocument.recompute()
42                     Gui.SendMsgToActiveView("ViewFit")
43                     return "MSG|Nodes created!"
44
45                 return "ERR|Data dimensions mismatch!"
46
47         return "ERR|File does not exist!"
48
49     return "ERR|Argument count mismatch!"
```

Listing V.10 - MakeNodes.py

```
1  # -*- coding: utf-8 -*-
2  # Script to show the message box with the message inside
3  # @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4  # Doktorska disertacija
5  # Mašinski fakultet Univerziteta u Beogradu
6  # --------------------
7
8  """@package showMsg
9  Script to show the message box with the message inside.
10 """
11
12 def init(params):
13     """!
14     Script main function.
15     @param params: Param params[2] is a title, params[3] is a message.
16     @return Returns a response message.
```

```
17      """
18      if len(params) == 4:
19          from PySide2 import QtCore, QtWidgets
20          diag = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information, params
                [2], params[3])
21          diag.exec_()
22          return "MSG|Dialog shown!"
23
24      return "ERR|Argument count mismatch!"
```

Listing V.11 - showMsg.py

W

Skripta createModelFreeCAD.jsl

Skriptom `createModelFreeCAD.jsl`, koja je u celosti data u listingu W.1, se pomoću *FreeCAD* radnog prostora `ExternalAPI` modelira kinematski model mehanizma.

```
/**
 * @file Create model in FreeCAD
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x = [ // translaWcija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  0 // [deg] psi
];
var d1 = 20;
var d2 = 10;
var D = 40;
var part = 'MP_6DOF_XXX'; // naziv modela
var exe = 'C:/Program Files/FreeCAD 0.21/bin/FreeCAD.exe'; // putanja do
    programa
var port = 11077; // port TCP servera
var host = 'localhost'; // adresa TCP servera
var model = pwd + '/resources/' + part + '.FCStd';
var timeout = 300000; // [ms] timeout
var startup_timeout = 120000; // [ms] start up timeout
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
```

```
40 solver.getGeometry();
41
42 // Odredjivanje potrebnog ugla alpha za date podatke
43 disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
44 solver.setParameters(x);
45
46 // Provera uglova u zglobovim
47 disp(' Provera uglova u zglobovima...');
48 solver.checkJoints();
49
50 if(any(MP.e)) {
51   disp(' Neuspešan proračun!');
52   dispRowVector('x', MP.x, MP.m);
53   dispRowVector('e', MP.e, MP.m);
54 } else {
55   disp(' Uspešan proračun!');
56   dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);
57
58   disp(' Modeliranje geometrije mehanizma leta...');
59
60   // Variables translation for initial node and line calculations
61   var Pb1 = plus(scale(MP.i_jpb, MP.lb), MP.Pb);
62   var B1 = plus(scale(MP.i_jpb, MP.lb), MP.B);
63   var A2 = plus(scale(MP.i_ll, MP.lb), MP.A);
64   var A1 = plus(minus(MP.A, MP.B), B1);
65
66   // Nodes and diameters array
67   var nodes = [
68     ...MP.A,
69     ...A1,
70     ...MP.B,
71     ...B1,
72     ...MP.Pb,
73     ...Pb1,
74     ...A2
75   ];
76   nodes = reshape(transpose(nodes, nodes.length / 3, 3) , nodes.length / 3, 3);
77   D = Array(nodes.length).fill(D);
78
79   // Lines definitions
80   var lines1 = concatCol(7*6,
81     concatRow(6, MP.A, MP.B),
82     concatRow(6, MP.A, A1),
83     concatRow(6, B1, A1),
84     concatRow(6, MP.B, B1),
85     concatRow(6, MP.A, MP.Pb),
86     concatRow(6, MP.Pb, Pb1),
87     concatRow(6, MP.A, A2)
88   );
89   d1 = Array(lines1.length / 6).fill(d1);
90
91   // Additional lines definitions
92   var lines2 = concatCol(2*6,
93     concatRow(6, MP.B, movelr(MP.B, 3)),
94     concatRow(6, MP.Pb, movelr(MP.Pb, 3))
95   );
96   d2 = Array(lines2.length / 6).fill(d2);
97
98   // Combine lines and diameters
```

```
  var lines = [...lines1, ...lines2];
  lines = reshape(transpose(lines, lines.length / 6, 6) , lines.length / 6, 6);
  var d = [...d1, ...d2];

  // Generate JSON for nodes
  var nodesFile = `${pwd}/resources/MP_nodes.json`;
  var nodesData = {
    'Coordinates': nodes,
    'Diameters': D
  };
  writeFile(nodesFile, JSON.stringify(nodesData));

  // Generate JSON for beams
  var beamsFile = `${pwd}/resources/MP_beams.json`;
  var beamsData = {
    'Coordinates': lines,
    'Diameters': d
  };
  writeFile(beamsFile, JSON.stringify(beamsData));

  // Run FreeCADLink
  await freecad_link.start(exe, {
    port: port,
    host: host,
    timeout: timeout,
    startup_timeout: startup_timeout
  }); // Start FreeCAD programa

  await freecad_link.newDocument(part);
  await freecad_link.callScript('MakeNodes', nodesFile, timeout);
  await freecad_link.callScript('MakeBeams', beamsFile, timeout);
  await freecad_link.callScript('MakeFusion', [], timeout);
  await freecad_link.saveAs(model, timeout);
  // await freecad_link.quit(); // Optionally close program

  deleteFile(nodesFile);
  deleteFile(beamsFile);
}

// - Kraj proračuna
disp(' Program je uspešno izvršen... ');
disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
disp(' -------------------- ');
```

Listing W.1 - `createModelFreeCAD.jsl`

X

Programi za upravljanje MP-6DOF-XXX

U ovom prilogu su dati svi programi za mikrokontrolere električnog sistema prototipa simulatora leta u stvarnoj veličini *MP-6DOF-XXX*. Svi programi sa njihovim modulima i bibliotekama su detaljno opisani u poglavlju 6.1.2.

X.1 Control Board

Glavni deo programa `MP-6DOF-XXX_CB` za elektroniku `Control Board` je dat u listingu X.1, dok su u svim ostalim listinzima dati kodovi za module i konfiguracione fajlove (listinzi X.2, X.3, X.4, X.5, X.6, X.7, X.8, X.9, X.10, X.11, X.12, X.13, X.14, X.15, X.16, X.17, X.18 i X.19).

```
1  // Control Board program
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // --------------------
6  // Board: "Generic STM32H7 Series"
7  // Board part number: "Generic H743VITx"
8  // U(S)ART support: "Enabled (generic 'Serial')"
9  // USB Support (if available): "CDC (generic 'Serial' supersede U(S)ART)"
10 // USB speed (if available): "Low/Full Speed"
11 // Optimize: "Fastest (-O3)"
12 // Debug symbols and core logs: "None"
13 // C Runtime Library: "Newlib Nano (default)"
14 // Upload method: "STM32CubeProgrammer (SWD)"
15 // --------------------
16
17 // Configuration
18 #include "config.h"
19
20 // Helper files
21 #include "math_helper.h"
22 #include "print_helper.h"
23
24 // Classes
25 #include "BASIC_IO.h"
26 #include "MISSION_CONTROLLER.h"
27 #include "SERVO_BOARD.h"
28 #include "MOTION_PLANNER.h"
29 #include "MECHANISM_6RSS.h"
30
31 // Define variables
```

```
32 // --------------------
33 uint8_t state = UNINITIALIZED_STATE;
34
35 uint8_t SB_state[N_ACT] = {0};
36 float alpha_m[N_ACT] = {0};
37 float temp[N_ACT] = {0};
38 float f_out[N_ACT] = {0};
39
40 uint8_t SBs_enable = 0;
41 float alpha_w_in[N_ACT] = {0};
42 float x_w_in[N_ACT] = {0};
43 float f_max_in = 0;
44 float Kp_in = 0;
45 float Ki_in = 0;
46 float Kd_in = 0;
47
48 float alpha_w[N_ACT] = {0};
49 float f_max = 0;
50 float Kp = 0;
51 float Ki = 0;
52 float Kd = 0;
53
54 uint32_t t_wait_SBs = millis();
55 uint32_t t_wake_SBs[N_ACT];
56 bool wait_SBs_flag[N_ACT] = {false};
57
58 HardwareSerial HS_SBs[] = {
59   HardwareSerial(RX1_PIN, TX1_PIN),
60   HardwareSerial(RX2_PIN, TX2_PIN),
61   HardwareSerial(RX3_PIN, TX3_PIN),
62   HardwareSerial(RX4_PIN, TX4_PIN),
63   HardwareSerial(RX5_PIN, TX5_PIN),
64   HardwareSerial(RX6_PIN, TX6_PIN)
65 };
66
67 HardwareSerial HS_MC(MAIN_RX_PIN, MAIN_TX_PIN);
68
69 BasicIO IO;
70 MissionController MC;
71 ServoBoard SBs[] = {
72   ServoBoard(),
73   ServoBoard(),
74   ServoBoard(),
75   ServoBoard(),
76   ServoBoard(),
77   ServoBoard()
78 };
79 MotionPlanner MP;
80 Mechanism6RSS M6RSS;
81
82 dataOutMC_t dataOutMC;
83 dataOutSB_t dataOutSB;
84
85 bool DISABLE_SERVO[] = {
86   DISABLE_SERVO_1,
87   DISABLE_SERVO_2,
88   DISABLE_SERVO_3,
89   DISABLE_SERVO_4,
90   DISABLE_SERVO_5,
```

```
91    DISABLE_SERVO_6
92  };
93
94  // Regression test function
95  #ifdef RUN_REGRESSION_TEST
96  #include "regression_test.h"
97  #endif
98
99  // Debug
100 #ifdef DEBUG_MAIN
101 #include "debug.h"
102 #endif
103
104 // setup() function
105 // --------------------
106 void setup() {
107   // Communication settings
108 #if defined(DEBUG_MAIN) || defined(RUN_REGRESSION_TEST) || defined(WAIT_SERIAL)
109   SerialUSB.begin(MC_USB_BAUDRATE);
110
111 #ifdef WAIT_SERIAL
112   uint32_t t = millis();
113   while(!SerialUSB.available() && ((millis()-t) < WAIT_SERIAL_TIME)) {};
114 #endif
115
116 #ifdef RUN_REGRESSION_TEST
117   // Run regression test
118   runRegressionTest();
119 #endif
120 #endif
121
122   // Submodules setup
123 #ifdef DEBUG_MAIN
124   SerialUSB.print("@main [");
125   SerialUSB.print(millis());
126   SerialUSB.println("]: Submodules setup");
127 #endif
128   IO.setup();
129   MC.setup(&SerialUSB, &HS_MC);
130   for(uint8_t j = 0; j < N_ACT; j++) {
131     SBs[j].setup(&HS_SBs[j], j+1, M6RSS.alpha_calib[j], -M6RSS.sigma_l[j]);
132   }
133   MP.setup();
134   M6RSS.setup();
135
136   // Set state
137   state = WAITING_SBS_START;
138 }
139
140 // loop() function
141 // --------------------
142 void loop(){
143 #ifdef DEBUG_LOOP_TIME
144   static uint32_t t_loop = micros();
145
146   SerialUSB.print("@main: Loop time: ");
147   SerialUSB.println(micros()-t_loop);
148   t_loop = micros();
149 #endif
```

```
150
151   IO.loop();
152   MC.read();
153   for(uint8_t j = 0; j < N_ACT; j++) {
154     if(!DISABLE_SERVO[j]) {
155       if(SBs[j].read()) {
156         SBs[j].getData(&SB_state[j], &alpha_m[j], &temp[j], &f_out[j]);
157
158         // Read ESTOP on SB3 inputs 1 and 4
159         if(j == 2 && ((SBs[j].switches & 0x01) || (SBs[j].switches & 0x08))) {
160           IO.state = ESTOP_PRESSED;
161           IO.disableVFDs();
162         }
163       }
164     } else {
165       SBs[j].flushInput();
166     }
167   }
168   if(MC.new_pid) {
169     MC.getCoeffsPID(&f_max_in, &Kp_in, &Ki_in, &Kd_in);
170   }
171
172   // Run state machine
173   runStateMachine();
174
175 #ifdef DEBUG_MAIN
176   // Print status
177   printStatus();
178 #endif
179
180   // Send heartbeat
181   sendHeartbeat();
182
183   // Send status
184   sendStatus();
185 }
186
187 // runStateMachine() function
188 // Run state machine
189 // --------------------
190 void runStateMachine() {
191   // Check subsystems state
192   checkState();
193
194   if(state <= NORMAL_STATE) {
195     switch(state) {
196       case WAITING_SBS_START:
197         waitingSBsStart();
198         break;
199       case WAITING_SBS:
200         waitingSBs();
201         break;
202       case SERVO_HOMING_START:
203         servoHomingStart();
204         break;
205       case SERVO_HOMING:
206         servoHoming();
207         break;
208       case NORMAL_STATE:
```

```
209           normalOperation();
210           break;
211       }
212     } else {
213       onErrorState();
214     }
215   }
216
217   // checkState() function
218   // Check if no errors
219   // --------------------
220   void checkState() {
221   #ifdef CHECK_MEASURED_POSITION
222     static uint32_t t_check_measured_position = millis();
223
224     bool angle_flag = true;
225     for(uint8_t j = 0; j < N_ACT; j++) {
226       if(fabs(alpha_m[j]-alpha_w[j]) > ALPHA_CHECK_MEASURED_THRESHOLD) {
227         angle_flag = false;
228         break;
229       }
230     }
231
232     if(angle_flag) {
233       t_check_measured_position = millis();
234     } else {
235       M6RSS.setDesiredAlphas(alpha_m);
236       M6RSS.getParameters();
237       M6RSS.checkJoints();
238       if(any(M6RSS.e, N_ACT)) {
239         if((millis()-t_check_measured_position) > CHECK_MEASURED_POSITION_TIMEOUT)
                {
240           state = MEASURED_POSITION_ERROR;
241         }
242       } else {
243         t_check_measured_position = millis();
244       }
245     }
246   #endif
247
248     if(state <= NORMAL_STATE) { // Prevents error override
249       if(IO.state > NORMAL_STATE) {
250         state = IO.state;
251       } else if(MC.state > NORMAL_STATE) {
252         state = MC.state;
253       } else if(SBs[0].state > NORMAL_STATE) {
254         state = SBs[0].state;
255       } else if(SBs[1].state > NORMAL_STATE) {
256         state = SBs[1].state;
257       } else if(SBs[2].state > NORMAL_STATE) {
258         state = SBs[2].state;
259       } else if(SBs[3].state > NORMAL_STATE) {
260         state = SBs[3].state;
261       } else if(SBs[4].state > NORMAL_STATE) {
262         state = SBs[4].state;
263       } else if(SBs[5].state > NORMAL_STATE) {
264         state = SBs[5].state;
265       } else if(MP.state > NORMAL_STATE) {
266         state = MP.state;
```

```
267     } else if(M6RSS.state > NORMAL_STATE) {
268       state = M6RSS.state;
269     }
270   }
271 }
272 
273 // sendDataSBs() function
274 // Send data to SBs
275 // --------------------
276 void sendDataSBs() {
277   static uint32_t t_send = micros();
278 
279   if((micros()-t_send) > SB_CONTROL_TIME) {
280 #ifdef DEBUG_SBS_SEND_TIME
281   SerialUSB.print("@main: SBs send time: ");
282   SerialUSB.println(micros()-t_send);
283 #endif
284     t_send = micros();
285 
286     for(uint8_t j = 0; j < N_ACT; j++) {
287       if(!DISABLE_SERVO[j]) {
288         dataOutSB.p.enable = SBs_enable;
289         dataOutSB.p.alpha_w = alpha_w[j];
290         dataOutSB.p.f_max = f_max;
291         dataOutSB.p.Kp = Kp;
292         dataOutSB.p.Ki = Ki;
293         dataOutSB.p.Kd = Kd;
294         SBs[j].sendData(dataOutSB);
295       }
296     }
297   }
298 }
299 
300 // sendHeartbeat function
301 // Send heartbeat
302 // --------------------
303 void sendHeartbeat() {
304   static uint32_t t_heartbeat = millis();
305 
306   if((millis()-t_heartbeat) > HEARTBEAT_TIME) {
307     t_heartbeat = millis();
308 
309     dataOutMC.p.ID = 0;
310     dataOutMC.p.P[0] = 0;
311     dataOutMC.p.P[1] = 0;
312     dataOutMC.p.P[2] = 0;
313     dataOutMC.p.P[3] = 0;
314     dataOutMC.p.P[4] = 0;
315     dataOutMC.p.P[5] = 0;
316 
317 #if defined(USE_MC_USB) || defined(USE_MC_USB_MONITOR)
318     MC.sendDataUSB(dataOutMC);
319 #endif
320 #ifdef USE_MC_RS485FD
321     MC.sendDataRS485FD(dataOutMC);
322 #endif
323   }
324 }
325 
```

```
326 // sendStatus() function
327 // Send status data
328 // --------------------
329 void sendStatus() {
330   static uint32_t t_send = millis();
331 
332   if((millis()-t_send) > STATUS_TIME) {
333     t_send = millis();
334 
335     for(uint8_t s = 1; s < 8; s++) {
336       dataOutMC.p.ID = 100+s;
337       switch(dataOutMC.p.ID) {
338         case 101:
339           dataOutMC.p.P[0] = state;
340           dataOutMC.p.P[1] = IO.state;
341           dataOutMC.p.P[2] = MC.state;
342           dataOutMC.p.P[3] = MP.state;
343           dataOutMC.p.P[4] = M6RSS.state;
344           dataOutMC.p.P[5] = 0;
345           break;
346         case 102:
347           for(uint8_t j = 0; j < N_ACT; j++) {
348             dataOutMC.p.P[j] =  M6RSS.e[j];
349           }
350           break;
351         case 103:
352           for(uint8_t j = 0; j < N_ACT; j++) {
353             dataOutMC.p.P[j] = SB_state[j];
354           }
355           break;
356         case 104:
357           for(uint8_t j = 0; j < N_ACT; j++) {
358             dataOutMC.p.P[j] = alpha_m[j];
359           }
360           break;
361         case 105:
362           for(uint8_t j = 0; j < N_ACT; j++) {
363             dataOutMC.p.P[j] = temp[j];
364           }
365           break;
366         case 106:
367           for(uint8_t j = 0; j < N_ACT; j++) {
368             dataOutMC.p.P[j] = f_out[j];
369           }
370           break;
371         case 107:
372           dataOutMC.p.P[0] = f_max_in*1000.0;
373           dataOutMC.p.P[1] = Kp_in*1000.0;
374           dataOutMC.p.P[2] = Ki_in*1000.0;
375           dataOutMC.p.P[3] = Kd_in*1000.0;
376           break;
377       }
378 
379 #if defined(USE_MC_USB) || defined(USE_MC_USB_MONITOR)
380       MC.sendDataUSB(dataOutMC);
381 #endif
382 #ifdef USE_MC_RS485FD
383       MC.sendDataRS485FD(dataOutMC);
384 #endif
```

```
385       }
386     }
387   }
388
389   // waitingSBsStart() function
390   // Waiting SBs start
391   // --------------------
392   void waitingSBsStart() {
393   #ifdef DEBUG_MAIN
394     SerialUSB.print("@main [");
395     SerialUSB.print(millis());
396     SerialUSB.println("]: Waiting for SBs...");
397   #endif
398
399     IO.enableVFDs();
400     dataOutSB.p.enable = 0;
401     dataOutSB.p.alpha_w = 0;
402     dataOutSB.p.f_max = 0;
403     dataOutSB.p.Kp = 0;
404     dataOutSB.p.Ki = 0;
405     dataOutSB.p.Kd = 0;
406     fill(millis(), t_wake_SBs, N_ACT);
407     state = WAITING_SBS;
408   }
409
410   // waitingSBs() function
411   // Waiting SBs
412   // --------------------
413   void waitingSBs() {
414     if(!all(wait_SBs_flag, N_ACT)) {
415       for(uint8_t j = 0; j < N_ACT; j++) {
416         if(DISABLE_SERVO[j]) {
417           wait_SBs_flag[j] = true;
418           SBs[j].state = NORMAL_STATE;
419           SBs[j].flushInput();
420         } else {
421           if(!wait_SBs_flag[j]) {
422             if(SBs[j].N_in > 0) {
423   #ifdef DEBUG_MAIN
424               SerialUSB.print("@main [");
425               SerialUSB.print(millis());
426               SerialUSB.print("]: SB ");
427               SerialUSB.print(j+1);
428               SerialUSB.println(" booted!");
429   #endif
430               SBs[j].state = NORMAL_STATE;
431               wait_SBs_flag[j] = true;
432             }
433
434             // Send wake command
435             if((millis()-t_wake_SBs[j]) > SERVO_BOARD_WAKE_TIME) {
436               t_wake_SBs[j] = millis();
437               SBs[j].sendData(dataOutSB);
438             }
439
440             // Check for wait timeout
441             if((millis()-t_wait_SBs) > SERVO_BOARD_WAIT_TIMEOUT_TIME) {
442   #ifdef DEBUG_MAIN
443               SerialUSB.print("@main: SB ");
```

```
444             SerialUSB.print(j+1);
445             SerialUSB.println(" wait timeout!");
446 #endif
447             SBs[j].state = SERVO_BOARD_WAIT_TIMEOUT + (j+1);
448             wait_SBs_flag[j] = true;
449           }
450         }
451       }
452     }
453   } else {
454     state = SERVO_HOMING_START;
455     for(uint8_t j = 0; j < N_ACT; j++) {
456       SBs[j].resetTimeout();
457     }
458   }
459 }
460
461 // servoHomingStart() function
462 // Servo homing start
463 // --------------------
464 void servoHomingStart() {
465 #ifdef DEBUG_MAIN
466   SerialUSB.print("@main [");
467   SerialUSB.print(millis());
468   SerialUSB.println("]: Servos homing...");
469 #endif
470   SBs_enable = 1;
471   f_max = HOMING_FMAX;
472   Kp = HOMING_KP;
473   Ki = HOMING_KI;
474   Kd = HOMING_KD;
475   for(uint8_t j = 0; j < N_ACT; j++) {
476     alpha_w[j] = 0;
477   }
478   state = SERVO_HOMING;
479 }
480
481 // servoHoming() function
482 // Servo homing
483 // --------------------
484 void servoHoming() {
485   bool flag = true;
486   for(uint8_t j = 0; j < N_ACT; j++) {
487     if(fabs(alpha_m[j]-alpha_w[j]) > ALPHA_ERROR_THRESHOLD) {
488       flag = false;
489       break;
490     }
491   }
492
493   if(flag) {
494     IO.setNormalBlink();
495     state = NORMAL_STATE;
496   } else {
497     sendDataSBs();
498   }
499 }
500
501 // normalOperation() function
502 // Normal operation
```

```
503 // --------------------
504 void normalOperation() {
505   SBs_enable = 1;
506   f_max = f_max_in;
507   Kp = Kp_in;
508   Ki = Ki_in;
509   Kd = Kd_in;
510   if(MC.new_alpha_w) {
511     MC.getDesiredAlphas(alpha_w_in);
512     for(uint8_t j = 0; j < N_ACT; j++) {
513       alpha_w[j] = alpha_w_in[j];
514     }
515     M6RSS.setDesiredAlphas(alpha_w);
516     M6RSS.getParameters();
517     M6RSS.checkJoints();
518 #ifdef DEBUG_MECHANISM_6RSS_CALC_IO
519     SerialUSB.print("@main: alpha_w = [ ");
520     dispRowVector(alpha_w, N_ACT);
521     SerialUSB.println("]");
522     SerialUSB.print("@main: x = [ ");
523     dispRowVector(M6RSS.x, N_ACT);
524     SerialUSB.println("]");
525     SerialUSB.print("@main: e = [ ");
526     dispRowVector(M6RSS.e, N_ACT);
527     SerialUSB.println("]");
528 #endif
529   } else if(MC.new_x_w) {
530     MC.getDesiredParameters(x_w_in);
531     M6RSS.setDesiredParameters(x_w_in);
532     M6RSS.getAlpha();
533     M6RSS.checkJoints();
534     for(uint8_t j = 0; j < N_ACT; j++) {
535       alpha_w[j] = M6RSS.alpha[j]*R2D;
536     }
537 #ifdef DEBUG_MECHANISM_6RSS_CALC_IO
538     SerialUSB.print("@main: x_w_in = [ ");
539     dispRowVector(x_w_in, N_ACT);
540     SerialUSB.println("]");
541     SerialUSB.print("@main: alpha_w = [ ");
542     dispRowVector(alpha_w, N_ACT);
543     SerialUSB.println("]");
544     SerialUSB.print("@main: e = [ ");
545     dispRowVector(M6RSS.e, N_ACT);
546     SerialUSB.println("]");
547 #endif
548   }
549
550   if(!any(M6RSS.e, N_ACT)) {
551     sendDataSBs();
552   }
553 }
554
555 // onErrorState() function
556 // On error state
557 // --------------------
558 void onErrorState() {
559   if(IO.VFDS_enabled) {
560     SBs_enable = 0;
561     sendDataSBs();
```

```
562       IO.disableVFDs();
563     }
564 }
```

Listing X.1 - MP-6DOF-XXX_CB.ino

```
1  // Control Board configuration parameters
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef CONTROL_BOARD_CONFIG_H
8  #define CONTROL_BOARD_CONFIG_H
9
10 // Optional flags
11 //#define RUN_REGRESSION_TEST
12 //#define WAIT_SERIAL
13 //#define DEBUG_MAIN
14 //#define DEBUG_LOOP_TIME
15 //#define DEBUG_SBS_SEND_TIME
16 //#define DEBUG_SERVO_BOARD
17 //#define DEBUG_SERVO_BOARD_PACKETS
18 //#define DEBUG_SERVO_BOARD_ACK_TIMEOUT
19 //#define DEBUG_MECHANISM_6RSS
20 //#define DEBUG_MISSION_CONTROLLER
21 //#define DEBUG_BASIC_IO
22 //#define DEBUG_MOTION_PLANNER
23 //#define DEBUG_MECHANISM_6RSS_CALC_IO
24
25 #define USE_MC_RS485FD
26 //#define USE_MC_USB
27 #define USE_MC_USB_MONITOR
28 #define CHECK_MEASURED_POSITION
29
30 #define DISABLE_SERVO_1 false
31 #define DISABLE_SERVO_2 false
32 #define DISABLE_SERVO_3 false
33 #define DISABLE_SERVO_4 false
34 #define DISABLE_SERVO_5 false
35 #define DISABLE_SERVO_6 false
36
37 #define DEBUG_LEVEL 2
38
39 // Communication
40 #define MC_USB_BAUDRATE 1000000
41 #define MC_RS485FD_BAUDRATE 115200
42 #define SB_BAUDRATE 576000
43
44 // Timings
45 #define LAMP_INIT_TIME 3000 // [ms] Init lamp time difference
46 #define LAMP_NORMAL_TIME 1000 // [ms] Nominal lamp time difference
47 #define LAMP_ERROR_TIME 50 // [ms] Error lamp time difference
48 #define PRINT_STATUS_TIME 1000 // [ms]
49 #define STATUS_TIME 100 // [ms]
50 #define HEARTBEAT_TIME 1000 // [ms]
51 #define LED_TIME 100 // [ms]
52 #define SERVO_BOARD_WAKE_TIME 500 // [ms]
53 #define SERVO_BOARD_WAIT_TIMEOUT_TIME 30000 // [ms]
54 #define SERVO_BOARD_READ_TIMEOUT_TIME 500 // [ms]
```

```
55 #define SB_CONTROL_TIME 5000 // [us]
56 #define WAIT_SERIAL_TIME 15000 // [ms]
57 #define CHECK_MEASURED_POSITION_TIMEOUT 1000 // [ms]
58 #define ESTOP_READ_TIMEOUT 300 // [ms]
59
60 // Homing
61 #define HOMING_FMAX 3.0
62 #define HOMING_KP 1.0
63 #define HOMING_KI 0.0
64 #define HOMING_KD 0.0
65
66 // Mechanism
67 #define N_DIM 3 // [-] number of dimensions
68 #define N_ACT 6 // [-] number of actuators
69 #define M_a 250.0 // [mm] duzina donje poluge od Bi do Ai
70 #define M_s 1000.0 // [mm] duzina gornje poluge od Ai do Pi
71 #define M_R_p 490.0 // [mm] poluprecnik platforme od O1 do Pi
72 #define M_R_b 700.0 // [mm] poluprecnik baze od O do Bi
73 #define M_gamma_p 44.0 // [deg] ugao izmedju zglobova platforme
74 #define M_gamma_b 25.0 // [deg] ugao izmedju obrtnih zglobova baze
75 #define M_epsilon 39.0 // [deg] ugao ose motora u odnosu na horizontalnu ravan
76 #define M_tau_ll 19.0 // [deg] odstupanje ugla donje poluge od zadatog
77 #define M_tau_jp -81.0 // [deg] odstupanje ugla zglobova platforme od zadatog
78 #define M_mu_ll -31.0 // [deg] ugao ose zgloba donje poluge i ravani motora
79 #define M_mu_jp 14.0 // [deg] ugao ose zgloba platforme i horizontalne ravani
80
81 #define LAMBDA_MAX 35.0 // [deg] max allowed joints angle
82 #define TAU_R 1E-6 // [-] relative error of external parameters
83 #define TAU_A 1E-4 // [-] absoulte error of external parameters
84 #define DK_MAX_ITER 20 // [-] max number of iterations for direct kinematic
85
86 #define ALPHA_CALIB_1 -10.3 // [deg]
87 #define ALPHA_CALIB_2 -10.5 // [deg]
88 #define ALPHA_CALIB_3 7.0 // [deg]
89 #define ALPHA_CALIB_4 -1.0 // [deg]
90 #define ALPHA_CALIB_5 -21.5 // [deg]
91 #define ALPHA_CALIB_6 9.3 // [deg]
92
93 #define ALPHA_ERROR_THRESHOLD 0.5 // [deg]
94 #define ALPHA_CHECK_MEASURED_THRESHOLD 1 // [deg]
95
96 #define COG_X 0.0 // [mm]
97 #define COG_Y 0.0 // [mm]
98 #define COG_Z 500.0 // [mm]
99
100 #define PROTOCOL_MAX 65534
101 #define IP_POROTOCOL_LIM 180
102 #define EP_REL_IN_MAX 200
103 #define EP_REL_IN_MIN 0
104 #define EP_REL_IN_C 100
105
106 #define EP_LIM_MAX_1 400.0
107 #define EP_LIM_MAX_2 400.0
108 #define EP_LIM_MAX_3 300.0
109 #define EP_LIM_MAX_4 45.0
110 #define EP_LIM_MAX_5 45.0
111 #define EP_LIM_MAX_6 45.0
112
113 #define EP_LIM_MIN_1 -400.0
```

```
114 #define EP_LIM_MIN_2 -400.0
115 #define EP_LIM_MIN_3 -200.0
116 #define EP_LIM_MIN_4 -45.0
117 #define EP_LIM_MIN_5 -45.0
118 #define EP_LIM_MIN_6 -45.0
119
120 #define EP_PROTOCOL_LIM_1 1000.0
121 #define EP_PROTOCOL_LIM_2 1000.0
122 #define EP_PROTOCOL_LIM_3 500.0
123 #define EP_PROTOCOL_LIM_4 45.0
124 #define EP_PROTOCOL_LIM_5 45.0
125 #define EP_PROTOCOL_LIM_6 45.0
126
127 #define IP_LIM_MAX_1 120.0
128 #define IP_LIM_MAX_2 120.0
129 #define IP_LIM_MAX_3 120.0
130 #define IP_LIM_MAX_4 120.0
131 #define IP_LIM_MAX_5 120.0
132 #define IP_LIM_MAX_6 120.0
133
134 #define IP_LIM_MIN_1 -120.0
135 #define IP_LIM_MIN_2 -120.0
136 #define IP_LIM_MIN_3 -120.0
137 #define IP_LIM_MIN_4 -120.0
138 #define IP_LIM_MIN_5 -120.0
139 #define IP_LIM_MIN_6 -120.0
140
141 // Other
142 #define ANGLE_MAX 360.0
143 #define ANGLE_C 180.0
144 #define ANGLE_MIN 0.0
145
146 // Math
147 #define D2R PI/180.0 // [-] deg to rad conversion constant
148 #define R2D 180.0/PI // [-] rad to deg conversion constant
149
150 // Pins
151 #define LED_PIN PB12
152 #define ENABLE_VFD_PIN PE11
153 #define LAMP_PIN PE13
154
155 #define MAIN_RX_PIN PB11
156 #define MAIN_TX_PIN PB10
157
158 #define RX1_PIN PA3
159 #define TX1_PIN PA2
160 #define RX2_PIN PB5
161 #define TX2_PIN PB_6_ALT1
162 #define RX3_PIN PE0
163 #define TX3_PIN PE1
164 #define RX4_PIN PC11
165 #define TX4_PIN PC10
166 #define RX5_PIN PC7
167 #define TX5_PIN PC6
168 #define RX6_PIN PB15
169 #define TX6_PIN PB14
170
171 // Enums
172 enum CB_STATE {
```

```
173   UNINITIALIZED_STATE,
174   WAITING_SBS_START,
175   WAITING_SBS,
176   SERVO_HOMING_START,
177   SERVO_HOMING,
178   NORMAL_STATE,
179   ESTOP_PRESSED,
180   MEASURED_POSITION_ERROR,
181   SERVO_BOARD_WAIT_TIMEOUT = 110,
182   SERVO_BOARD_READ_TIMEOUT = 120,
183   SERVO_BOARD_ERROR = 130
184 };
185
186 enum SB_STATE {
187   SB_UNINITIALIZED_STATE = 0,
188   SB_DISABLED_STATE,
189   SB_NORMAL_STATE,
190   SB_ANGLE_SENSOR_WAIT_TIMEOUT,
191   SB_VFD_WAIT_TIMEOUT,
192   SB_VFD_READ_TIMEOUT,
193   SB_VFD_VALID_READ_TIMEOUT,
194   SB_ANGLE_SENSOR_READ_TIMEOUT,
195   SB_VFD_ERROR = 100
196 };
197
198 enum MP_STATE {
199   MP_NORMAL_STATE = 0,
200   MP_GET_ALPHA_FAILED,
201   MP_LAMBDA_P_ERROR,
202   MP_LAMBDA_B_ERROR,
203   MP_LEVER_COLLISION,
204   MP_MU_UL_ERROR,
205   MP_GET_PARAEMTERS_FAILED,
206   MP_JACOBIAN_INVERSE_FAILED,
207   MP_GET_PARAEMTERS_MAX_ITER
208 };
209 #endif
```

Listing X.2 - MP-6DOF-XXX_CB config.h

```
1 -DSERIAL_RX_BUFFER_SIZE=1024 -DSERIAL_TX_BUFFER_SIZE=1024
```

Listing X.3 - MP-6DOF-XXX_CB build_opt.h

```
1  // BasicIO - BASIC_IO.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef BASIC_IO_H
8  #define BASIC_IO_H
9
10 #include "config.h" // Configuration
11
12 #include <Arduino.h>
13
14 class BasicIO {
15  public:
16   BasicIO();
```

```
17    ~BasicIO();
18
19    uint32_t state = UNINITIALIZED_STATE;
20
21    void setup();
22    void loop();
23    void enableVFDs();
24    void disableVFDs();
25    void setNormalBlink();
26    void setErrorBlink();
27    void blinkLamp();
28    void blinkLEDs();
29
30    uint32_t dt_lamp = LAMP_INIT_TIME;
31
32    bool VFDS_enabled = false;
33    uint32_t t_VFDS_enabled;
34  };
35
36  #endif // BASIC_IO_H
```

Listing X.4 - MP-6DOF-XXX_CB BASIC_IO.h

```
1   // BasicIO - BASIC_IO.cpp
2   // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Mašinski fakultet Univerziteta u Beogradu
5   // --------------------
6
7   #include "BASIC_IO.h"
8
9   // BasicIO()
10  // Object constructor
11  // --------------------
12  BasicIO::BasicIO() {}
13
14  // ~BasicIO()
15  // Object destructor
16  // --------------------
17  BasicIO::~BasicIO() {}
18
19  // setup() function
20  // --------------------
21  void BasicIO::setup() {
22  #ifdef DEBUG_BASIC_IO
23    if(DEBUG_LEVEL > 0) {
24      SerialUSB.println("@BasicIO: setup()");
25    }
26  #endif
27
28    // Set pin modes
29    pinMode(LED_PIN, OUTPUT);
30    pinMode(ENABLE_VFD_PIN, OUTPUT);
31    pinMode(LAMP_PIN, OUTPUT);
32
33    // Initial otuputs
34    digitalWrite(LED_PIN, HIGH);
35    digitalWrite(ENABLE_VFD_PIN, LOW);
36    digitalWrite(LAMP_PIN, HIGH);
37
```

```
38    this->state = NORMAL_STATE;
39  }
40
41  // loop() function
42  // --------------------
43  void BasicIO::loop() {
44  #ifdef DEBUG_BASIC_IO
45    if(DEBUG_LEVEL > 1) {
46      SerialUSB.println("@BasicIO: loop()");
47    }
48  #endif
49
50    this->blinkLamp();
51    this->blinkLEDs();
52  }
53
54  // enableVFDs() function
55  // Send command to relay to enable VFDs
56  // --------------------
57  void BasicIO::enableVFDs() {
58  #ifdef DEBUG_BASIC_IO
59    if(DEBUG_LEVEL > 0) {
60      SerialUSB.println("@BasicIO: enableVFDs()");
61    }
62  #endif
63
64    if(!this->VFDS_enabled) {
65      this->t_VFDS_enabled = millis();
66      this->VFDS_enabled = true;
67      digitalWrite(ENABLE_VFD_PIN, HIGH);
68    }
69  }
70
71  // disableVFDs() function
72  // Send command to relay to disable VFDs
73  // --------------------
74  void BasicIO::disableVFDs() {
75  #ifdef DEBUG_BASIC_IO
76    if(DEBUG_LEVEL > 0) {
77      SerialUSB.println("@BasicIO: disableVFDs()");
78    }
79  #endif
80
81    if(VFDS_enabled) {
82      this->VFDS_enabled = false;
83      digitalWrite(ENABLE_VFD_PIN, LOW);
84    }
85    this->setErrorBlink();
86  }
87
88  // setNormalBlink() function
89  // Set normal blink
90  // --------------------
91  void BasicIO::setNormalBlink() {
92  #ifdef DEBUG_BASIC_IO
93    if(DEBUG_LEVEL > 0) {
94      SerialUSB.println("@BasicIO: setNormalBlink()");
95    }
96  #endif
```

```
97
98    this->dt_lamp = LAMP_NORMAL_TIME;
99  }
100
101 // setErrorBlink() function
102 // Set error blink
103 // --------------------
104 void BasicIO::setErrorBlink() {
105 #ifdef DEBUG_BASIC_IO
106   if(DEBUG_LEVEL > 0) {
107     SerialUSB.println("@BasicIO: setErrorBlink()");
108   }
109 #endif
110
111   this->dt_lamp = LAMP_ERROR_TIME;
112 }
113
114 // blinkLamp() function
115 // Blink lamp at right time
116 // --------------------
117 void BasicIO::blinkLamp() {
118 #ifdef DEBUG_BASIC_IO
119   if(DEBUG_LEVEL > 2) {
120     SerialUSB.println("@BasicIO: blinkLamp()");
121   }
122 #endif
123   static uint32_t t_lamp = millis();
124   static bool s = false;
125
126   if((millis()-t_lamp) > this->dt_lamp) {
127     t_lamp = millis();
128     s = !s;
129     digitalWrite(LAMP_PIN, s);
130   }
131 }
132
133 // blinkLEDs() function
134 // Blink LEDs at right time
135 // --------------------
136 void BasicIO::blinkLEDs() {
137 #ifdef DEBUG_BASIC_IO
138   if(DEBUG_LEVEL > 2) {
139     SerialUSB.println("@BasicIO: blinkLEDs()");
140   }
141 #endif
142   static uint8_t s = 0;
143   static uint32_t t_led = millis();
144
145   if((millis()-t_led) > LED_TIME) {
146     t_led = millis();
147     s = !s;
148     digitalWrite(LED_PIN, s);
149   }
150 }
```

Listing X.5 - MP-6DOF-XXX_CB BASIC_IO.cpp

```
1 // Mechanism6RSS - MECHANISM_6RSS.h
2 // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3 // Doktorska disertacija
```

```
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef MECHANISM_6RSS_H
8  #define MECHANISM_6RSS_H
9
10 #include "config.h" // Configuration
11
12 #include <Arduino.h>
13
14 #include "math_helper.h" // Math helper
15
16 class Mechanism6RSS {
17  public:
18   Mechanism6RSS();
19   ~Mechanism6RSS();
20
21   uint32_t state = UNINITIALIZED_STATE;
22
23   void setup();
24   void reset();
25   void getGeometry();
26   void getAlpha();
27   void getJacobian();
28   void getParameters();
29   void checkJoints();
30   void setDesiredAlphas(float32_t*);
31   void setDesiredParameters(float32_t*);
32
33   bool IK_flag = false; // [-] Inverse kinematic success flag
34   float32_t sigma_l[N_ACT] = {-1, 1, -1, 1, -1, 1};
35   float32_t nu_ll[N_ACT] = {2*PI/3, -PI/3, -2*PI/3, PI/3, 0, PI};
36   float32_t beta_s[N_ACT] = {PI/6, PI/6, 5*PI/6, 5*PI/6, -PI/2, -PI/2};
37
38   float32_t alpha[N_ACT] = {0}; // [deg] uglovi donjih poluga
39   float32_t alpha_m[N_ACT] = {0}; // [deg] izmereni uglovi donjih poluga
40   float32_t alpha_w[N_ACT] = {0}; // [deg] zeljeni uglovi donjih poluga
41   float32_t x[N_ACT] = {0}; // [mm i deg] translacija i rotacija [x,y,z,phi,
        theta,psi]
42   float32_t x_o[N_ACT]; // stare vrednosti
43
44   float32_t delta_p[N_ACT], delta_b[N_ACT], beta_ll[N_ACT], beta_jp[N_ACT];
45   float32_t i_m[N_DIM*N_ACT], i_ll[N_DIM*N_ACT], i_ll0[N_DIM*N_ACT], i_jp[N_DIM*
        N_ACT], i_jpb[N_DIM*N_ACT];
46   float32_t calpha[N_ACT], salpha[N_ACT], cepsilon, sepsilon;
47   float32_t z0, z1, P[N_DIM*N_ACT], Pb[N_DIM*N_ACT], PO[2*N_DIM], POb[2*N_DIM],
        B[N_DIM*N_ACT], A[N_DIM*N_ACT];
48   float32_t BPb[N_DIM*N_ACT], l[N_ACT], D[N_ACT], E[N_ACT], F[N_ACT], G[N_ACT];
49   float32_t R[N_DIM*N_DIM], RP[N_DIM*N_ACT],RPO[N_DIM*N_ACT], Ri_jp[N_DIM*N_ACT
        ];
50   float32_t Rm1_i_ll[N_DIM*N_ACT], Rm2_i_ll[N_DIM*N_ACT],Rm1_BA[N_DIM*N_ACT],
        Rm2_BA[N_DIM*N_ACT],BA_0[N_DIM*N_ACT];
51   float32_t PbA[N_DIM*N_ACT], lambda_p[N_ACT], lambda_b[N_ACT];
52   float32_t cpsi, ctheta, cphi, spsi, stheta, sphi;
53   float32_t i_ul[N_DIM*N_ACT], m_ul[N_DIM*N_ACT], tau_r0, dalpha[N_ACT],
        dalpha_r[N_ACT], dx[N_ACT];
54   float32_t m_m[N_DIM*N_ACT], BA[N_DIM*N_ACT], PObPb[N_DIM*N_ACT];
55   float32_t J_ai[N_ACT], J_a[N_ACT*N_ACT], invJ_ai[N_ACT], invJ_a[N_ACT*N_ACT];
56   float64_t J[N_ACT*N_ACT], J_x_f64[N_ACT*N_ACT], J_x_f64_t[N_ACT*N_ACT],
```

```
      invJ_x_f64[N_ACT*N_ACT];
  uint8_t e[N_ACT], n_iter;

  float32_t alpha_calib[N_ACT] =
    {ALPHA_CALIB_1, ALPHA_CALIB_2, ALPHA_CALIB_3, ALPHA_CALIB_4, ALPHA_CALIB_5,
       ALPHA_CALIB_6};

  arm_matrix_instance_f64 J_x = {N_ACT, N_ACT, J_x_f64_t};
  arm_matrix_instance_f64 invJ_x = {N_ACT, N_ACT, invJ_x_f64};
  arm_status inv_status;

  float32_t cg[N_DIM] = {COG_X, COG_Y, COG_Z};
};

#endif // MECHANISM_6RSS_H
```

Listing X.6 - MP-6DOF-XXX_CB MECHANISM_6RSS.h

```
// Mechanism6RSS - MECHANISM_6RSS.cpp
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#include "MECHANISM_6RSS.h"

// Mechanism6RSS()
// Object constructor
// --------------------
Mechanism6RSS::Mechanism6RSS() {}

// ~Mechanism6RSS()
// Object destructor
// --------------------
Mechanism6RSS::~Mechanism6RSS() {}

// setup() function
// --------------------
void Mechanism6RSS::setup() {
#ifdef DEBUG_MECHANISM_6RSS
  if(DEBUG_LEVEL > 0) {
    SerialUSB.println("@Mechanism6RSS: setup()");
  }
#endif

  // Get mechanism geometry
  this->getGeometry();

  // Get mechanism parameters
  this->getParameters();

  // Check mechanism joints
  this->checkJoints();

  this->state = NORMAL_STATE;
}

// reset() function
// --------------------
void Mechanism6RSS::reset() {
```

```
43 #ifdef DEBUG_MECHANISM_6RSS
44   if(DEBUG_LEVEL > 0) {
45     SerialUSB.println("@Mechanism6RSS: reset()");
46   }
47 #endif
48
49   fill(0.0, this->x, N_ACT);
50   fill(0.0, this->x_o, N_ACT);
51   fill(0.0, this->alpha_w, N_ACT);
52   fill(0.0, this->alpha_m, N_ACT);
53   fill(0.0, this->alpha, N_ACT);
54 }
55
56 // setDesiredAlphas() function
57 // --------------------
58 void Mechanism6RSS::setDesiredAlphas(float32_t* alpha_w_in) {
59 #ifdef DEBUG_MECHANISM_6RSS
60   if(DEBUG_LEVEL > 1) {
61     SerialUSB.println("@Mechanism6RSS: setDesiredAlphas()");
62   }
63 #endif
64
65   arm_copy_f32(alpha_w_in, this->alpha_w, N_ACT);
66 }
67
68 // setDesiredParameters() function
69 // --------------------
70 void Mechanism6RSS::setDesiredParameters(float32_t* x_w_in) {
71 #ifdef DEBUG_MECHANISM_6RSS
72   if(DEBUG_LEVEL > 1) {
73     SerialUSB.println("@Mechanism6RSS: setDesiredParameters()");
74   }
75 #endif
76
77   arm_copy_f32(x_w_in, this->x, N_ACT);
78 }
79
80 // getGeometry() function
81 // --------------------
82 void Mechanism6RSS::getGeometry() {
83 #ifdef DEBUG_MECHANISM_6RSS
84   if(DEBUG_LEVEL > 0) {
85     SerialUSB.println("@Mechanism6RSS: getGeometry()");
86   }
87 #endif
88
89   this->cepsilon = cos(M_epsilon*D2R);
90   this->sepsilon = sin(M_epsilon*D2R);
91
92   for(uint8_t j = 0; j < N_ACT; j++) {
93     this->beta_ll[j] = this->nu_ll[j]+M_tau_ll*this->sigma_l[j]*D2R;
94     this->beta_jp[j] = this->beta_s[j]+M_tau_jp*this->sigma_l[j]*D2R;
95     this->delta_p[j] = this->beta_s[j]+M_gamma_p*this->sigma_l[j]*D2R;
96     this->delta_b[j] = this->beta_s[j]+M_gamma_b*this->sigma_l[j]*D2R;
97
98     this->i_m[j*N_DIM] = -this->sigma_l[j]*cos(this->beta_ll[j]+PI/2*this->
        sigma_l[j])*this->cepsilon;
99     this->i_m[j*N_DIM+1] = -this->sigma_l[j]*sin(this->beta_ll[j]+PI/2*this->
        sigma_l[j])*this->cepsilon;
```

```
100       this->i_m[j*N_DIM+2] = -this->sigma_l[j]*this->sepsilon;
101       this->i_ll[j*N_DIM] = cos(this->beta_ll[j]+PI/2*this->sigma_l[j])*cos((
              M_epsilon+M_mu_ll)*D2R);
102       this->i_ll[j*N_DIM+1] = sin(this->beta_ll[j]+PI/2*this->sigma_l[j])*cos((
              M_epsilon+M_mu_ll)*D2R);
103       this->i_ll[j*N_DIM+2] = sin((M_epsilon+M_mu_ll)*D2R);
104       this->i_jp[j*N_DIM] = cos(this->beta_jp[j])*cos(M_mu_jp*D2R);
105       this->i_jp[j*N_DIM+1] = sin(this->beta_jp[j])*cos(M_mu_jp*D2R);
106       this->i_jp[j*N_DIM+2] = sin(M_mu_jp*D2R);
107       this->P[j*N_DIM] = M_R_p*cos(this->delta_p[j]);
108       this->P[j*N_DIM+1] = M_R_p*sin(this->delta_p[j]);
109       this->P[j*N_DIM+2] = 0;
110       this->B[j*N_DIM] = M_R_b*cos(this->delta_b[j]);
111       this->B[j*N_DIM+1] = M_R_b*sin(this->delta_b[j]);
112       this->B[j*N_DIM+2] = 0;
113
114       this->A[j*N_DIM] = this->B[j*N_DIM]+M_a*cos(this->beta_ll[j]);
115       this->A[j*N_DIM+1] = this->B[j*N_DIM+1]+M_a*sin(this->beta_ll[j]);
116       this->A[j*N_DIM+2] = this->B[j*N_DIM+2];
117
118       for(uint8_t i = 0; i < N_DIM; i++) {
119         this->BA_0[j*N_DIM+i] = this->A[j*N_DIM+i]-this->B[j*N_DIM+i];
120       }
121     }
122     arm_copy_f32(this->i_ll, this->i_ll0, N_DIM*N_ACT);
123
124     cross3D(this->i_m, this->i_ll0, this->Rm2_i_ll, N_DIM, N_ACT);
125     cross3D(this->i_m, this->Rm2_i_ll, this->Rm1_i_ll, N_DIM, N_ACT);
126     cross3D(this->i_m, this->BA_0, this->Rm2_BA, N_DIM, N_ACT);
127     cross3D(this->i_m, this->Rm2_BA, this->Rm1_BA, N_DIM, N_ACT);
128
129     this->z1 = pow(M_s, 2)-(pow(this->P[0]-this->A[0], 2)+pow(this->P[1]-this->A
            [1], 2));
130     if(this->z1 > 0) {
131       this->z0 = sqrt(this->z1)+this->A[2];
132     } else {
133       this->z0 = 1.0/0.0;
134     }
135
136     arm_copy_f32(this->i_jp, this->i_jpb, N_DIM*N_ACT);
137     arm_copy_f32(this->P, this->Pb, N_DIM*N_ACT);
138
139     for(uint8_t j = 0; j < N_ACT; j++) {
140       this->Pb[j*N_DIM+2] = this->Pb[j*N_DIM+2]+this->z0;
141     }
142
143     for(uint8_t i = 0; i < N_DIM; i++) {
144       this->PO[i] = this->cg[i];
145       this->PO[i+N_DIM] = 0;
146     }
147   }
148
149   // getAlpha() function
150   // --------------------
151   void Mechanism6RSS::getAlpha() {
152   #ifdef DEBUG_MECHANISM_6RSS
153     if(DEBUG_LEVEL > 1) {
154       SerialUSB.println("@Mechanism6RSS: getAlpha()");
155     }
```

```
156 #endif
157
158   this->cphi = cos(this->x[3]*D2R);
159   this->ctheta = cos(this->x[4]*D2R);
160   this->cpsi = cos(this->x[5]*D2R);
161   this->sphi = sin(this->x[3]*D2R);
162   this->stheta = sin(this->x[4]*D2R);
163   this->spsi = sin(this->x[5]*D2R);
164
165   R[0] = this->cpsi*this->ctheta;
166   R[1] = this->spsi*this->ctheta;
167   R[2] = -this->stheta;
168   R[3] = -this->spsi*this->cphi+this->cpsi*this->stheta*this->sphi;
169   R[4] = this->cpsi*this->cphi+this->spsi*this->stheta*this->sphi;
170   R[5] = this->ctheta*this->sphi;
171   R[6] = this->spsi*this->sphi+this->cpsi*this->stheta*this->cphi;
172   R[7] = -this->cpsi*this->sphi+this->spsi*this->stheta*this->cphi;
173   R[8] = this->ctheta*this->cphi;
174
175   multiply(this->R, this->i_jp, this->Ri_jp, N_DIM, N_DIM, N_ACT);
176   multiply(this->R, this->P, this->RP, N_DIM, N_DIM, N_ACT);
177   multiply(this->R, this->PO, this->RPO, N_DIM, N_DIM, 2);
178
179   for(uint8_t j = 0; j < 2; j++) {
180     for(uint8_t i = 0; i < N_DIM; i++) {
181       this->POb[j*N_DIM+i] = this->x[i]+this->RPO[j*N_DIM+i];
182     }
183     this->POb[j*N_DIM+2] = this->POb[j*N_DIM+2]+this->z0;
184   }
185
186   for(uint8_t j = 0; j < N_ACT; j++) {
187     for(uint8_t i = 0; i < N_DIM; i++) {
188       this->Pb[j*N_DIM+i] = this->x[i]+this->RP[j*N_DIM+i];
189       this->BPb[j*N_DIM+i] = this->Pb[j*N_DIM+i]-this->B[j*N_DIM+i];
190       this->i_jpb[j*N_DIM+i] = this->Ri_jp[j*N_DIM+i];
191     }
192
193     this->Pb[j*N_DIM+2] = this->Pb[j*N_DIM+2]+this->z0;
194     this->BPb[j*N_DIM+2] = this->BPb[j*N_DIM+2]+this->z0;
195
196     this->l[j] = sqrt(pow(this->BPb[j*N_DIM], 2)+pow(this->BPb[j*N_DIM+1], 2)+
         pow(this->BPb[j*N_DIM+2], 2));
197     this->D[j] = pow(this->l[j], 2)-(pow(M_s, 2)-pow(M_a, 2))-2*((this->Rm1_BA[j
         *N_DIM]+this->BA_0[j*N_DIM])*this->BPb[j*N_DIM]+\
198       (this->Rm1_BA[j*N_DIM+1]+this->BA_0[j*N_DIM+1])*this->BPb[j*N_DIM+1]+(this
           ->Rm1_BA[j*N_DIM+2]+this->BA_0[j*N_DIM+2])*this->BPb[j*N_DIM+2]);
199     this->E[j] = 2*(this->Rm2_BA[j*N_DIM]*this->BPb[j*N_DIM]+this->Rm2_BA[j*
         N_DIM+1]*this->BPb[j*N_DIM+1]+this->Rm2_BA[j*N_DIM+2]*this->BPb[j*N_DIM
         +2]);
200     this->F[j] = -2*(this->Rm1_BA[j*N_DIM]*this->BPb[j*N_DIM]+this->Rm1_BA[j*
         N_DIM+1]*this->BPb[j*N_DIM+1]+this->Rm1_BA[j*N_DIM+2]*this->BPb[j*N_DIM
         +2]);
201     this->G[j] = this->D[j]/sqrt(pow(this->E[j], 2)+pow(this->F[j], 2));
202     this->alpha[j] = asin(this->G[j])-atan2(this->F[j], this->E[j]);
203
204     this->calpha[j] = cos(this->alpha[j]);
205     this->salpha[j] = sin(this->alpha[j]);
206
207     for(uint8_t i = 0; i < N_DIM; i++) {
```

```
208         this->A[j*N_DIM+i] = this->B[j*N_DIM+i]+this->BA_0[j*N_DIM+i]+(1-this->
                calpha[j])*this->Rm1_BA[j*N_DIM+i]+\
209           this->salpha[j]*this->Rm2_BA[j*N_DIM+i];
210         this->i_ll[j*N_DIM+i] = this->i_ll0[j*N_DIM+i]+(1-this->calpha[j])*this->
                Rm1_i_ll[j*N_DIM+i]+\
211           this->salpha[j]*this->Rm2_i_ll[j*N_DIM+i];
212       }
213
214       if(fabs(this->G[j]) <= 1) {
215         this->e[j] = MP_NORMAL_STATE;
216         this->IK_flag = true;
217       } else {
218         this->e[j] = MP_GET_ALPHA_FAILED;
219       }
220     }
221   }
222
223   // getJacobian() function
224   // ----------------------
225   void Mechanism6RSS::getJacobian() {
226   #ifdef DEBUG_MECHANISM_6RSS
227     if(DEBUG_LEVEL > 1) {
228       SerialUSB.println("@Mechanism6RSS: getJacobian()");
229     }
230   #endif
231
232     for(uint8_t k = 0; k < N_DIM*N_ACT; k++) {
233       this->i_ul[k] = (this->Pb[k]-this->A[k])/M_s;
234       this->BA[k] = this->A[k]-this->B[k];
235     }
236
237     for(uint8_t j = 0; j < N_ACT; j++) {
238       for(uint8_t i = 0; i < N_DIM; i++) {
239         this->PObPb[j*N_DIM+i] = this->Pb[j*N_DIM+i]-this->POb[1*N_DIM+i];
240       }
241     }
242
243     cross3D(this->PObPb, this->i_ul, this->m_ul, N_DIM, N_ACT);
244     for(uint8_t j = 0; j < N_DIM; j++) {
245       for(uint8_t i = 0; i < N_ACT; i++) {
246         this->J_x_f64[j*N_ACT+i] = this->i_ul[j+i*N_DIM];
247         this->J_x_f64[(j+N_DIM)*N_ACT+i] = this->m_ul[j+i*N_DIM]; // J_x
248       }
249     }
250
251     for(uint8_t k = 0; k < N_ACT*N_ACT; k++) {
252       this->J_x_f64_t[k] = this->J_x_f64[k];
253     }
254     this->inv_status = arm_mat_inverse_f64(&this->J_x, &this->invJ_x);  // invJ_x
255     if(this->inv_status != ARM_MATH_SUCCESS) {
256       fill(MP_JACOBIAN_INVERSE_FAILED, this->e, N_ACT);
257       return;
258     }
259
260     cross3D(this->i_m, this->BA, this->m_m, N_DIM, N_ACT);
261     dotColumn(this->i_ul, this->m_m, this->J_ai, N_DIM, N_ACT);
262     diag(this->J_ai, this->J_a, N_ACT); // J_a
263     reciprocal(this->J_ai, this->invJ_ai, N_ACT);
264     diag(this->invJ_ai, this->invJ_a, N_ACT); // invJ_a
```

```
265
266   multiply(this->invJ_x_f64, this->J_a, this->J, N_ACT, N_ACT, N_ACT); //
          Jacobian = inv(J_x)*J_a
267 }
268
269 // getParameters() function
270 // --------------------
271 void Mechanism6RSS::getParameters() {
272 #ifdef DEBUG_MECHANISM_6RSS
273   if(DEBUG_LEVEL > 1) {
274     SerialUSB.println("@Mechanism6RSS: getParameters()");
275   }
276 #endif
277
278   arm_copy_f32(this->x, this->x_o, N_ACT);
279
280   this->n_iter = 0;
281   for(uint8_t j = 0; j < N_ACT; j++) {
282     this->dalpha[j] = this->alpha_m[j]-this->alpha_w[j];
283   }
284   this->tau_r0 = TAU_R*norm(this->dalpha, N_ACT);
285   while(norm(this->dalpha, N_ACT) > this->tau_r0+TAU_A) {
286     if(this->n_iter >= DK_MAX_ITER) {
287         fill(MP_GET_PARAEMTERS_MAX_ITER, this->e, N_ACT);
288         break;
289     } else {
290       this->n_iter++;
291     }
292
293     this->getAlpha();
294     if(any(this->e, N_ACT)) {
295       fill(MP_GET_PARAEMTERS_FAILED, this->e, N_ACT);
296       break;
297     }
298     this->getJacobian();
299     if(any(this->e, N_ACT)) {
300       break;
301     }
302
303     // dx^k = J*dalpha;
304     for(uint8_t j = 0; j < N_ACT; j++) {
305       this->dalpha[j] = this->alpha[j]*R2D-this->alpha_w[j];
306       this->dalpha_r[j] = this->alpha[j]-this->alpha_w[j]*D2R;
307     }
308     multiply(this->J, this->dalpha_r, this->dx, N_ACT, N_ACT, 1);
309
310     // x^{k+1} = x^k+dx^k
311     for(uint8_t j = 0; j < 3; j++) {
312       this->x[j] = this->x[j]-this->dx[j];
313       this->x[j+3] = this->x[j+3]-this->dx[j+3]*R2D;
314     }
315   }
316   if(any(this->e, N_ACT)) {
317     arm_copy_f32(this->x_o, this->x, N_ACT);
318   }
319 }
320
321 // checkJoints() function
322 // --------------------
```

```
323 void Mechanism6RSS::checkJoints() {
324 #ifdef DEBUG_MECHANISM_6RSS
325   if(DEBUG_LEVEL > 1) {
326     SerialUSB.println("@Mechanism6RSS: checkJoints()");
327   }
328 #endif
329 
330   if(this->IK_flag) {
331     for(uint8_t k = 0; k < N_DIM*N_ACT; k++) {
332         this->PbA[k] = this->A[k]-this->Pb[k];
333     }
334 
335     for(uint8_t j = 0; j < N_ACT; j++) {
336       this->lambda_p[j] = acos((this->PbA[j*N_DIM]*this->i_jpb[j*N_DIM]+this->
          PbA[j*N_DIM+1]*this->i_jpb[j*N_DIM+1]+ \
337             this->PbA[j*N_DIM+2]*this->i_jpb[j*N_DIM+2])/M_s)*180.0/PI-90.0;
338       this->lambda_b[j] = acos((this->PbA[j*N_DIM]*this->i_ll[j*N_DIM]+this->PbA
          [j*N_DIM+1]*this->i_ll[j*N_DIM+1]+ \
339             this->PbA[j*N_DIM+2]*this->i_ll[j*N_DIM+2])/M_s)*180.0/PI-90.0;
340 
341       if(fabs(this->lambda_b[j]) > LAMBDA_MAX) {
342         e[j] = MP_LAMBDA_B_ERROR;
343       } else if(fabs(this->lambda_p[j]) > LAMBDA_MAX) {
344         e[j] = MP_LAMBDA_P_ERROR;
345       }
346     }
347   }
348 }
```

Listing X.7 - MP-6DOF-XXX_CB MECHANISM_6RSS.cpp

```
1  // MissionController - MISSION_CONTROLLER.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6 
7  #ifndef MISSION_CONTROLLER_H
8  #define MISSION_CONTROLLER_H
9 
10 #include "config.h" // Configuration
11 
12 #include <Arduino.h>
13 
14 #include <PRDC_SCBP.h>
15 
16 struct __attribute__((packed)) packetInMC_t {
17   uint16_t ID;
18   uint16_t P[6];
19   uint32_t N;
20 };
21 union dataInMC_t{
22   packetInMC_t p;
23   uint8_t bytes[sizeof(packetInMC_t)];
24 };
25 
26 struct __attribute__((packed)) packetOutMC_t {
27   uint8_t ID;
28   float P[6];
29   uint32_t N;
```

```
30 };
31 union dataOutMC_t{
32   packetOutMC_t p;
33   uint8_t bytes[sizeof(packetOutMC_t)];
34 };
35
36 class MissionController {
37  public:
38   MissionController();
39   ~MissionController();
40
41   uint32_t state = UNINITIALIZED_STATE;
42   float alpha_w[N_ACT];
43   float x_w[N_ACT];
44   bool new_alpha_w;
45   bool new_x_w;
46   float f_max;
47   float Kp;
48   float Ki;
49   float Kd;
50   uint32_t N_USB_in = 0;
51   uint32_t N_RS485FD_in = 0;
52   uint32_t N_USB_out = 0;
53   uint32_t N_RS485FD_out = 0;
54   bool new_pid;
55
56   float EP_LIM_MAX[6] = {EP_LIM_MAX_1, EP_LIM_MAX_2, EP_LIM_MAX_3, EP_LIM_MAX_4,
         EP_LIM_MAX_5, EP_LIM_MAX_6};
57   float EP_LIM_MIN[6] = {EP_LIM_MIN_1, EP_LIM_MIN_2, EP_LIM_MIN_3, EP_LIM_MIN_4,
         EP_LIM_MIN_5, EP_LIM_MIN_6};
58   float EP_PROTOCOL_LIM[6] = {EP_PROTOCOL_LIM_1, EP_PROTOCOL_LIM_2,
        EP_PROTOCOL_LIM_3, EP_PROTOCOL_LIM_4, EP_PROTOCOL_LIM_5, EP_PROTOCOL_LIM_6
         };
59   float IP_LIM_MAX[6] = {IP_LIM_MAX_1, IP_LIM_MAX_2, IP_LIM_MAX_3, IP_LIM_MAX_4,
         IP_LIM_MAX_5, IP_LIM_MAX_6};
60   float IP_LIM_MIN[6] = {IP_LIM_MIN_1, IP_LIM_MIN_2, IP_LIM_MIN_3, IP_LIM_MIN_4,
         IP_LIM_MIN_5, IP_LIM_MIN_6};
61
62   void setup(USBSerial*, HardwareSerial*);
63   bool read();
64   void flushInput();
65   void handleMessage();
66   void sendDataUSB(dataOutMC_t);
67   void sendDataRS485FD(dataOutMC_t);
68
69   void getDesiredAlphas(float*);
70   void getDesiredParameters(float*);
71   void getCoeffsPID(float*, float*, float*, float*);
72
73   USBSerial* USB;
74   HardwareSerial* HS_RS485FD;
75   PRDC_SCBP COM_USB;
76   PRDC_SCBP COM_RS485FD;
77   dataInMC_t dataIn;
78   dataOutMC_t dataOut;
79 };
80
81 #endif // MISSION_CONTROLLER_H
```

Listing X.8 - MP-6DOF-XXX_CB MISSION_CONTROLLER.h

```
1  // MissionController - MISSION_CONTROLLER.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "MISSION_CONTROLLER.h"
8
9  // MissionController()
10 // Object constructor
11 // --------------------
12 MissionController::MissionController() {}
13
14 // ~MissionController()
15 // Object destructor
16 // --------------------
17 MissionController::~MissionController() {}
18
19 // setup() function
20 // --------------------
21 void MissionController::setup(USBSerial* USB_in, HardwareSerial* HS_RS485FD_in)
       {
22 #ifdef DEBUG_MISSION_CONTROLLER
23   if(DEBUG_LEVEL > 0) {
24     SerialUSB.println("@MissionController: setup()");
25   }
26 #endif
27
28   this->USB = USB_in;
29   this->HS_RS485FD = HS_RS485FD_in;
30
31   // CB communication
32 #if defined(USE_MC_USB) || defined(USE_MC_USB_MONITOR)
33   this->COM_USB.begin(*this->USB, MC_USB_BAUDRATE);
34 #endif
35 #ifdef USE_MC_RS485FD
36   this->COM_RS485FD.begin(*this->HS_RS485FD, MC_RS485FD_BAUDRATE);
37 #endif
38
39   this->state = NORMAL_STATE;
40 }
41
42 // getControlCB() function
43 // Get control from CB
44 // --------------------
45 bool MissionController::read() {
46 #ifdef DEBUG_MISSION_CONTROLLER
47   if(DEBUG_LEVEL > 1) {
48     SerialUSB.println("@MissionController: read()");
49   }
50 #endif
51
52   bool got_message = false;
53
54 #ifdef USE_MC_USB
55   while(this->COM_USB.availableData() > 0) {
56     bool flagIn = this->COM_USB.readPacket(this->dataIn.bytes, sizeof(this->
         dataIn));
57     if(flagIn) {
```

```
58        this->handleMessage();
59        got_message = true;
60        this->N_USB_in += 1;
61      }
62    }
63  #endif
64
65  #ifdef USE_MC_RS485FD
66    while(this->COM_RS485FD.availableData() > 0) {
67      bool flagIn = this->COM_RS485FD.readPacket(this->dataIn.bytes, sizeof(this->
          dataIn));
68      if(flagIn) {
69        this->handleMessage();
70        got_message = true;
71        this->N_RS485FD_in += 1;
72      }
73    }
74  #endif
75    return got_message;
76  }
77
78  // flushInput() function
79  // Flush input
80  // --------------------
81  void MissionController::flushInput() {
82  #ifdef DEBUG_MISSION_CONTROLLER
83    if(DEBUG_LEVEL > 0) {
84      SerialUSB.println("@MissionController: flushInput()");
85    }
86  #endif
87
88    this->COM_USB.flushInput();
89    this->COM_RS485FD.flushInput();
90  }
91
92  // handleMessage() function
93  // Handle input message
94  // --------------------
95  void MissionController::handleMessage() {
96  #ifdef DEBUG_MISSION_CONTROLLER
97    if(DEBUG_LEVEL > 1) {
98      SerialUSB.println("@MissionController: handleMessage()");
99    }
100 #endif
101
102   if(this->dataIn.p.ID == 11) {
103     // Absolute internal parameters
104     for(uint8_t j = 0; j < N_ACT; j++) {
105       // Read input
106       this->alpha_w[j] =((float)this->dataIn.p.P[j]/PROTOCOL_MAX)* \
107         (ANGLE_MAX-ANGLE_MIN)+ANGLE_MIN;
108
109       // Calculate real value
110       if(this->alpha_w[j] > ANGLE_MAX) {
111         this->alpha_w[j] = ANGLE_MAX;
112       } else if(alpha_w[j] < ANGLE_MIN){
113         this->alpha_w[j] = ANGLE_MIN;
114       }
115       this->alpha_w[j] = this->alpha_w[j]-ANGLE_C;
```

```
116       }
117       this->new_alpha_w = true;
118     } else if(this->dataIn.p.ID == 12) {
119       // Relative external parameters
120       for(uint8_t j = 0; j < N_ACT; j++) {
121         this->x_w[j] = ((float)this->dataIn.p.P[j]/PROTOCOL_MAX* \
122           (EP_REL_IN_MAX-EP_REL_IN_MIN)+EP_REL_IN_MIN-EP_REL_IN_C)/EP_REL_IN_C;
123         if(this->x_w[j] > 0) {
124           this->x_w[j] =  this->x_w[j]*this->EP_LIM_MAX[j];
125         } else {
126           this->x_w[j] = -this->x_w[j]*this->EP_LIM_MIN[j];
127         }
128       }
129       this->new_x_w = true;
130     } else if(this->dataIn.p.ID == 13) {
131       // Absolute external parameters
132       for(uint8_t j = 0; j < N_ACT; j++) {
133         this->x_w[j] = (float)this->dataIn.p.P[j]/PROTOCOL_MAX*2*this->
              EP_PROTOCOL_LIM[j]-this->EP_PROTOCOL_LIM[j];
134       }
135       this->new_x_w = true;
136     } else if(this->dataIn.p.ID == 201) {
137       // PID configs
138       this->f_max = (float)this->dataIn.p.P[0]/1000.0;
139       this->Kp = (float)this->dataIn.p.P[1]/1000.0;
140       this->Ki = (float)this->dataIn.p.P[2]/1000.0;
141       this->Kd = (float)this->dataIn.p.P[3]/1000.0;
142       this->new_pid = true;
143     }
144   }
145
146   // getDesiredAlphas() function
147   // Handle input message for absolute internal parameters
148   // --------------------
149   void MissionController::getDesiredAlphas(float* alpha_w_out) {
150   #ifdef DEBUG_MISSION_CONTROLLER
151     if(DEBUG_LEVEL > 1) {
152       SerialUSB.println("@MissionController: getDesiredAlphas()");
153     }
154   #endif
155
156     for(uint8_t j = 0; j < N_ACT; j++) {
157       alpha_w_out[j] = this->alpha_w[j];
158     }
159     this->new_alpha_w = false;
160   }
161
162   // getDesiredParameters() function
163   // Handle input message for relative external parameters
164   // --------------------
165   void MissionController::getDesiredParameters(float* x_w_out) {
166   #ifdef DEBUG_MISSION_CONTROLLER
167     if(DEBUG_LEVEL > 1) {
168       SerialUSB.println("@MissionController: getDesiredParameters()");
169     }
170   #endif
171
172     for(uint8_t j = 0; j < N_ACT; j++) {
173       x_w_out[j] = this->x_w[j];
```

```
174   }
175   this->new_x_w = false;
176 }
177
178 // getCoeffsPID() function
179 // Get coeffs PID
180 // --------------------
181 void MissionController::getCoeffsPID(float* f_max_out, float* Kp_out, float*
      Ki_out, float* Kd_out) {
182 #ifdef DEBUG_MISSION_CONTROLLER
183   if(DEBUG_LEVEL > 1) {
184     SerialUSB.println("@MissionController: getCoeffsPID()");
185   }
186 #endif
187
188   *f_max_out = this->f_max;
189   *Kp_out = this->Kp;
190   *Ki_out = this->Ki;
191   *Kd_out = this->Kd;
192   this->new_pid = false;
193 }
194
195 // sendDataUSB() function
196 // Send data over USB
197 // --------------------
198 void MissionController::sendDataUSB(dataOutMC_t dataOut) {
199 #ifdef DEBUG_MISSION_CONTROLLER
200   if(DEBUG_LEVEL > 1) {
201     SerialUSB.println("@MissionController: sendData()");
202   }
203 #endif
204
205 #if defined(USE_MC_USB) || defined(USE_MC_USB_MONITOR)
206   this->N_USB_out = this->N_USB_out+1;
207   dataOut.p.N = this->N_USB_out;
208   this->COM_USB.sendPacket(dataOut.bytes, sizeof(dataOut));
209 #endif
210 }
211
212 // sendDataRS485FD() function
213 // Send data over RS485FD
214 // --------------------
215 void MissionController::sendDataRS485FD(dataOutMC_t dataOut) {
216 #ifdef DEBUG_MISSION_CONTROLLER
217   if(DEBUG_LEVEL > 1) {
218     SerialUSB.println("@MissionController: sendData()");
219   }
220 #endif
221
222 #ifdef USE_MC_RS485FD
223   this->N_RS485FD_out = this->N_RS485FD_out+1;
224   dataOut.p.N = this->N_RS485FD_out;
225   this->COM_RS485FD.sendPacket(dataOut.bytes, sizeof(dataOut));
226 #endif
227 }
```

Listing X.9 - MP-6DOF-XXX_CB MISSION_CONTROLLER.cpp

```
1 // MOTION_PLANNER - MOTION_PLANNER.h
2 // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
```

```
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#ifndef MOTION_PLANNER_H
#define MOTION_PLANNER_H

#include "config.h" // Configuration

#include <Arduino.h>

#include <PRDC_SCBP.h>

class MotionPlanner {
 public:
  MotionPlanner();
  ~MotionPlanner();

  uint32_t state = UNINITIALIZED_STATE;

  void setup();
  void loop();
};

#endif // MOTION_PLANNER_H
```

Listing X.10 - MP-6DOF-XXX_CB MOTION_PLANNER.h

```
// MOTION_PLANNER - MOTION_PLANNER.cpp
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#include "MOTION_PLANNER.h"

// MotionPlanner()
// Object constructor
// --------------------
MotionPlanner::MotionPlanner() {}

// ~MotionPlanner()
// Object destructor
// --------------------
MotionPlanner::~MotionPlanner() {}

// setup() function
// --------------------
void MotionPlanner::setup() {
#ifdef DEBUG_MOTION_PLANNER
  if(DEBUG_LEVEL > 0) {
    SerialUSB.println("@MotionPlanner: setup()");
  }
#endif

  this->state = NORMAL_STATE;
}

// loop() function
// --------------------
```

```
33 void MotionPlanner::loop() {
34 #ifdef DEBUG_MOTION_PLANNER
35   if(DEBUG_LEVEL > 1) {
36     SerialUSB.println("@MotionPlanner: loop()");
37   }
38 #endif
39
40 }
```

Listing X.11 - MP-6DOF-XXX_CB MOTION_PLANNER.cpp

```
1  // ServoBoard - SERVO_BOARD.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef SERVO_BOARD_H
8  #define SERVO_BOARD_H
9
10 #include "config.h" // Configuration
11
12 #include <Arduino.h>
13
14 #include <HardwareSerial.h>
15
16 #include <PRDC_SCBP.h>
17
18 struct __attribute__((packed)) packetInSB_t {
19   uint8_t state;
20   float alpha_m;
21   float temp;
22   float f_out;
23   uint8_t switches;
24   uint32_t N;
25 };
26 union dataInSB_t{
27   packetInSB_t p;
28   uint8_t bytes[sizeof(packetInSB_t)];
29 };
30
31 struct __attribute__((packed)) packetOutSB_t {
32   uint8_t enable;
33   float alpha_w;
34   float f_max;
35   float Kp;
36   float Ki;
37   float Kd;
38   uint32_t N;
39 };
40 union dataOutSB_t{
41   packetOutSB_t p;
42   uint8_t bytes[sizeof(packetOutSB_t)];
43 };
44
45 class ServoBoard {
46  public:
47   ServoBoard();
48   ~ServoBoard();
49
```

```
50    uint32_t state = UNINITIALIZED_STATE;
51
52    uint8_t id = 0;
53    float alpha_calib = 0;
54    int8_t alpha_sign = 1;
55    uint32_t n_no_message = 0;
56
57    void setup(HardwareSerial*, uint8_t, float, int8_t);
58    bool read();
59    void resetTimeout();
60    void flushInput();
61    void getData(uint8_t*, float*, float*, float*);
62    void sendData(dataOutSB_t);
63
64    float applyCalibration(float, bool);
65
66    HardwareSerial* HS;
67    PRDC_SCBP COM;
68    dataInSB_t dataIn;
69
70    bool new_data = false;
71    uint8_t state_in;
72    float alpha_m;
73    float temp;
74    float f_out;
75    uint8_t switches = 0;
76    uint32_t N_in = 0;
77    uint32_t N_out = 0;
78    uint32_t t_sent = 0;
79
80    uint32_t t_new_message;
81 };
82
83 #endif // SERVO_BOARD_H
```

Listing X.12 - MP-6DOF-XXX_CB SERVO_BOARD.h

```
1  // ServoBoard - SERVO_BOARD.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "SERVO_BOARD.h"
8
9  // ServoBoard()
10 // Object constructor
11 // --------------------
12 ServoBoard::ServoBoard() {}
13
14 // ~ServoBoard()
15 // Object destructor
16 // --------------------
17 ServoBoard::~ServoBoard() {}
18
19 // setup() function
20 // --------------------
21 void ServoBoard::setup(HardwareSerial* HS_in, uint8_t id_in, float
       alpha_calib_in, int8_t alpha_sign_in) {
22 #ifdef DEBUG_SERVO_BOARD
```

```
23    if(DEBUG_LEVEL > 0) {
24      SerialUSB.print("@ServoBoard ");
25      SerialUSB.print(this->id);
26      SerialUSB.println(": setup()");
27    }
28  #endif
29
30    this->id = id_in;
31    this->alpha_calib = alpha_calib_in;
32    this->alpha_sign = alpha_sign_in;
33
34    // CB communication
35    this->HS = HS_in;
36    this->COM.begin(*this->HS, SB_BAUDRATE);
37
38    this->state = UNINITIALIZED_STATE;
39  }
40
41  // getControlCB() function
42  // Get control from CB
43  // --------------------
44  bool ServoBoard::read() {
45  #ifdef DEBUG_SERVO_BOARD
46    if(DEBUG_LEVEL > 1) {
47      SerialUSB.print("@ServoBoard ");
48      SerialUSB.print(this->id);
49      SerialUSB.println(": read()");
50    }
51  #endif
52
53    bool got_message = false;
54
55    while(this->COM.availableData() > 0) {
56      bool flagIn = this->COM.readPacket(this->dataIn.bytes, sizeof(this->dataIn))
          ;
57      if(flagIn) {
58        this->t_new_message = millis();
59        this->new_data = true;
60        this->state_in = this->dataIn.p.state;
61        this->alpha_m = this->applyCalibration(this->dataIn.p.alpha_m, true);
62        this->temp = this->dataIn.p.temp;
63        this->f_out = this->dataIn.p.f_out;
64        this->switches = this->dataIn.p.switches;
65        this->N_in = this->dataIn.p.N;
66
67        this->n_no_message = 0;
68        if(this->state_in > SB_NORMAL_STATE && this->state == NORMAL_STATE) {
69          this->state = SERVO_BOARD_ERROR + this->id;
70        }
71        got_message = true;
72
73  #ifdef DEBUG_SERVO_BOARD_PACKETS
74        SerialUSB.print(millis());
75        SerialUSB.print(" - Got new SB");
76        SerialUSB.print(this->id);
77        SerialUSB.print(" packet ");
78        SerialUSB.println(dataIn.p.N);
79  #endif
80      }
```

```
81    }
82
83    if(!got_message) {
84      this->n_no_message = this->n_no_message+1;
85    }
86
87    // Check for read timeout
88    if(this->state == NORMAL_STATE) {
89      if((millis()-this->t_new_message) > SERVO_BOARD_READ_TIMEOUT_TIME && this->
          n_no_message > 1) {
90  #ifdef DEBUG_SERVO_BOARD
91      if(DEBUG_LEVEL > 0) {
92        SerialUSB.print("@ServoBoard ");
93        SerialUSB.print(this->id);
94        SerialUSB.println(": read timeout!");
95      }
96  #endif
97        this->state = SERVO_BOARD_READ_TIMEOUT + this->id;
98      }
99    }
100
101   return got_message;
102 }
103
104 // resetTimeout() function
105 // Reset timeout
106 // --------------------
107 void ServoBoard::resetTimeout() {
108 #ifdef DEBUG_SERVO_BOARD
109   if(DEBUG_LEVEL > 0) {
110     SerialUSB.print("@ServoBoard ");
111     SerialUSB.print(this->id);
112     SerialUSB.println(": resetTimeout()");
113   }
114 #endif
115
116   this->t_new_message = millis();
117 }
118
119 // flushInput() function
120 // Flush input
121 // --------------------
122 void ServoBoard::flushInput() {
123 #ifdef DEBUG_SERVO_BOARD
124   if(DEBUG_LEVEL > 0) {
125     SerialUSB.print("@ServoBoard ");
126     SerialUSB.print(this->id);
127     SerialUSB.println(": flushInput()");
128   }
129 #endif
130
131   this->COM.flushInput();
132 }
133
134 // getData() function
135 // Get data
136 // --------------------
137 void ServoBoard::getData(uint8_t* state_in_out, float* alpha_m_out, float*
      temp_out, float* f_out_out) {
```

```
138 #ifdef DEBUG_SERVO_BOARD
139   if(DEBUG_LEVEL > 1) {
140     SerialUSB.print("@ServoBoard ");
141     SerialUSB.print(this->id);
142     SerialUSB.println(": getData()");
143   }
144 #endif
145
146   *state_in_out = this->state_in;
147   *alpha_m_out = this->alpha_m;
148   *temp_out = this->temp;
149   *f_out_out = this->f_out;
150   this->new_data = false;
151 }
152
153 // sendData() function
154 // Send data
155 // --------------------
156 void ServoBoard::sendData(dataOutSB_t dataOut) {
157 #ifdef DEBUG_SERVO_BOARD
158   if(DEBUG_LEVEL > 1) {
159     SerialUSB.print("@ServoBoard ");
160     SerialUSB.print(this->id);
161     SerialUSB.println(": sendData()");
162   }
163 #endif
164
165   this->N_out = this->N_out+1;
166   dataOut.p.N = this->N_out;
167   dataOut.p.alpha_w = this->applyCalibration(dataOut.p.alpha_w, false);
168   this->COM.sendPacket(dataOut.bytes, sizeof(dataOut));
169   this->t_sent = micros();
170 }
171
172 // applyCalibration() function
173 // Appy calibration to angle
174 // --------------------
175 float ServoBoard::applyCalibration(float alpha, bool inv) {
176 #ifdef DEBUG_SERVO_BOARD
177   if(DEBUG_LEVEL > 1) {
178     SerialUSB.print("@ServoBoard ");
179     SerialUSB.print(this->id);
180     SerialUSB.println(": applyCalibration()");
181   }
182 #endif
183
184   double new_alpha;
185   if(!inv) {
186     new_alpha = (alpha+this->alpha_calib)*this->alpha_sign;
187   } else {
188     new_alpha = this->alpha_sign*alpha-this->alpha_calib;
189   }
190
191   new_alpha = fmod(new_alpha, 360.0);
192
193   // Map to the range -180 to 180 degrees
194   if (new_alpha <= -180.0) {
195       new_alpha += 360.0;
196   } else if (new_alpha > 180.0) {
```

```
197         new_alpha -= 360.0;
198     }
199     return new_alpha;
200 }
```

Listing X.13 - MP-6DOF-XXX_CB SERVO_BOARD.cpp

```
1  // Math helper - math_helper.h
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef MATH_HELPER_H
8  #define MATH_HELPER_H
9
10 #include <Arduino.h>
11
12 // CMSIS DSP Library
13 // Author: ARM KEIL
14 // http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html
15 #include <CMSIS_DSP.h>
16
17 void cross3D(float32_t *A_in ,float32_t *B_in, float32_t *C_out, uint8_t N,
       uint8_t M);
18 float32_t norm(float32_t *A_in, uint8_t N);
19 float32_t dot(float32_t *A_in, float32_t *B_in, uint8_t N);
20 void dotColumn(float32_t *A_in, float32_t *B_in, float32_t *C_in, uint8_t N,
       uint8_t M);
21
22 bool all(bool *A_in, uint8_t N);
23 bool all(uint8_t *A_in, uint8_t N);
24
25 bool any(bool *A_in, uint8_t N);
26 bool any(uint8_t *A_in, uint8_t N);
27 bool any(float32_t *A_in, uint8_t N);
28 bool any(uint8_t *A_in, uint8_t x, uint8_t N);
29
30 bool anyg(uint8_t *A_in, uint8_t x, uint8_t N);
31 bool anyl(uint8_t *A_in, uint8_t x, uint8_t N);
32
33 void fill(bool x, bool *A_out, uint8_t N);
34 void fill(uint8_t x, uint8_t *A_out, uint8_t N);
35 void fill(uint32_t x, uint32_t *A_out, uint8_t N);
36 void fill(float32_t x, float32_t *A_out, uint8_t N);
37
38 void diag(float32_t *A_in, float32_t *B_out, uint8_t N);
39 void scale(float32_t *A_in, float32_t x, float32_t *B_out, uint8_t N);
40 void reciprocal(float32_t *A_in, float32_t *B_out, uint8_t N);
41
42 void multiply(float32_t *A_in, float32_t *B_in, float32_t *C_out, uint8_t N,
       uint8_t M, uint8_t P);
43 void multiply(float32_t *A_in, float32_t *B_in, float64_t *C_out, uint8_t N,
       uint8_t M, uint8_t P);
44 void multiply(float64_t *A_in, float32_t *B_in, float32_t *C_out, uint8_t N,
       uint8_t M, uint8_t P);
45 void multiply(float64_t *A_in, float32_t *B_in, float64_t *C_out, uint8_t N,
       uint8_t M, uint8_t P);
46 void multiply(float64_t *A_in, float64_t *B_in, float32_t *C_out, uint8_t N,
       uint8_t M, uint8_t P);
```

```
47
48 void transpose(double *A_in, double *B_out,uint8_t N, uint8_t M);
49
50 #endif // MATH_HELPER_H
```

Listing X.14 - MP-6DOF-XXX_CB math_helper.h

```
1  // Math helper - math_helper.cpp
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // ---------------------
6
7  #include "math_helper.h"
8
9  // cross3D() function
10 // Calculate cross product
11 // ---------------------
12 void cross3D(float32_t *A_in ,float32_t *B_in, float32_t *C_out, uint8_t N,
       uint8_t M) {
13   for(uint8_t j = 0; j < M; j++) {
14     C_out[j*N] = A_in[j*N+1]*B_in[j*N+2]-A_in[j*N+2]*B_in[j*N+1];
15     C_out[j*N+1] = A_in[j*N+2]*B_in[j*N]-A_in[j*N]*B_in[j*N+2];
16     C_out[j*N+2] = A_in[j*N]*B_in[j*N+1]-A_in[j*N+1]*B_in[j*N];
17   }
18 }
19
20 // norm() function
21 // Calculate vector norm
22 // ---------------------
23 float32_t norm(float32_t *A_in, uint8_t N) {
24   float32_t S = 0;
25   for(uint8_t i = 0; i < N; i++) {
26     S += pow(A_in[i],2);
27   }
28   return sqrt(S);
29 }
30
31 // dot() function
32 // Calculate vector dot product
33 // ---------------------
34 float32_t dot(float32_t *A_in, float32_t *B_in, uint8_t N) {
35   float32_t Sout = 0;
36   for(uint8_t i = 0; i < N; i++) {
37      Sout += A_in[i]*B_in[i];
38   }
39   return sqrt(Sout);
40 }
41
42 // dotColumn() function
43 // Calculate vector dot product for matrix colums
44 // ---------------------
45 void dotColumn(float32_t *A_in, float32_t *B_in, float32_t *C_in, uint8_t N,
       uint8_t M) {
46   for(uint8_t j = 0; j < M; j++) {
47    C_in[j] = 0;
48     for(uint8_t i = 0; i < N; i++) {
49       C_in[j] += A_in[j*N+i]*B_in[j*N+i];
50     }
51   }
```

```
52  }
53
54  // all() function
55  // Check if all elements of vector are true
56  // --------------------
57  bool all(bool *A_in, uint8_t N) {
58    for(uint8_t i = 0; i < N; i++) {
59      if(A_in[i] != 1) {
60        return false;
61      }
62    }
63    return true;
64  }
65
66  bool all(uint8_t *A_in, uint8_t N) {
67    for(uint8_t i = 0; i < N; i++) {
68      if(A_in[i] != 1) {
69        return false;
70      }
71    }
72    return true;
73  }
74
75  // any() function
76  // Check if any element of vector is true
77  // --------------------
78  bool any(bool *A_in, uint8_t N) {
79    for(uint8_t i = 0; i < N; i++) {
80      if(A_in[i]) {
81        return true;
82      }
83    }
84    return false;
85  }
86
87  bool any(uint8_t *A_in, uint8_t N) {
88    for(uint8_t i = 0; i < N; i++) {
89      if(A_in[i] > 0) {
90        return true;
91      }
92    }
93    return false;
94  }
95
96  bool any(float32_t *A_in, uint8_t N) {
97    for(uint8_t i = 0; i < N; i++) {
98      if(A_in[i] > 0) {
99        return true;
100     }
101   }
102   return false;
103 }
104
105 bool any(uint8_t *A_in, uint8_t x, uint8_t N) {
106   for(uint8_t i = 0; i < N; i++) {
107     if(A_in[i] == x) {
108       return true;
109     }
110   }
```

```
111   return false;
112 }
113
114 // anyg() function
115 // Check if any element of vector is greater than value
116 // --------------------
117 bool anyg(uint8_t *A_in, uint8_t x, uint8_t N) {
118   for(uint8_t i = 0; i < N; i++) {
119     if(A_in[i] > x) {
120       return true;
121     }
122   }
123   return false;
124 }
125
126 // anyl() function
127 // Check if any element of vector is less than value
128 // --------------------
129 bool anyl(uint8_t *A_in, uint8_t x, uint8_t N) {
130   for(uint8_t i = 0; i < N; i++) {
131     if(A_in[i] < x) {
132       return true;
133     }
134   }
135   return false;
136 }
137
138 // fill() function
139 // fill vector with value
140 // --------------------
141 void fill(bool x, bool *A_out, uint8_t N) {
142   for(uint8_t i = 0; i < N; i++) {
143     A_out[i] = x;
144   }
145 }
146
147 void fill(uint8_t x, uint8_t *A_out, uint8_t N) {
148   for(uint8_t i = 0; i < N; i++) {
149     A_out[i] = x;
150   }
151 }
152
153 void fill(uint32_t x, uint32_t *A_out, uint8_t N) {
154   for(uint8_t i = 0; i < N; i++) {
155     A_out[i] = x;
156   }
157 }
158
159 void fill(float32_t x, float32_t *A_out, uint8_t N) {
160   for(uint8_t i = 0; i < N; i++) {
161     A_out[i] = x;
162   }
163 }
164
165 // diag() function
166 // Form diagonal matrix from vector
167 // --------------------
168 void diag(float32_t *A_in, float32_t *B_out, uint8_t N) {
169   fill(0.0, B_out, N*N);
```

```
170   for(uint8_t i = 0; i < N; i++) {
171     B_out[i+i*N] = A_in[i];
172   }
173 }
174 
175 // scale() function
176 // Scale vector with value
177 // --------------------
178 void scale(float32_t *A_in, float32_t x, float32_t *B_out, uint8_t N) {
179   for(uint8_t i = 0; i < N; i++) {
180     B_out[i] = A_in[i]*x;
181   }
182 }
183 
184 // reciprocal() function
185 // Reciprocal vector
186 // --------------------
187 void reciprocal(float32_t *A_in, float32_t *B_out, uint8_t N) {
188   for(uint8_t i = 0; i < N; i++) {
189     B_out[i] = 1/A_in[i];
190   }
191 }
192 
193 // multiply() function
194 // Matrix multiply
195 // --------------------
196 void multiply(float32_t *A_in, float32_t *B_in, float32_t *C_out,
197     uint8_t N, uint8_t M, uint8_t P) {
198   arm_fill_f32(0, C_out, M*P);
199   for(uint8_t i = 0; i < N; i++) {
200     for(uint8_t j = 0; j < M; j++) {
201       for(uint8_t k = 0; k < P; k++) {
202         C_out[k*N+i] += A_in[j*N+i]*B_in[k*M+j];
203       }
204     }
205   }
206 }
207 void multiply(float32_t *A_in, float32_t *B_in, float64_t *C_out,
208     uint8_t N, uint8_t M, uint8_t P) {
209   for(uint8_t p = 0; p < M*P; p++) {
210     C_out[p] = 0;
211   }
212   for(uint8_t i = 0; i < N; i++) {
213     for(uint8_t j = 0; j < M; j++) {
214       for(uint8_t k = 0; k < P; k++) {
215         C_out[k*N+i] += A_in[j*N+i]*B_in[k*M+j];
216       }
217     }
218   }
219 }
220 void multiply(float64_t *A_in, float32_t *B_in, float32_t *C_out,
221     uint8_t N, uint8_t M, uint8_t P) {
222   for(uint8_t p = 0; p < M*P; p++) {
223     C_out[p] = 0;
224   }
225   for(uint8_t i = 0; i < N; i++) {
226     for(uint8_t j = 0; j < M; j++) {
227       for(uint8_t k = 0; k < P; k++) {
228         C_out[k*N+i] += A_in[j*N+i]*B_in[k*M+j];
```

```
      }
    }
  }
}

void multiply(float64_t *A_in, float32_t *B_in, float64_t *C_out,
    uint8_t N, uint8_t M, uint8_t P) {
  for(uint8_t p = 0; p < M*P; p++) {
    C_out[p] = 0;
  }
  for(uint8_t i = 0; i < N; i++) {
    for(uint8_t j = 0; j < M; j++) {
      for(uint8_t k = 0; k < P; k++) {
        C_out[k*N+i] += A_in[j*N+i]*B_in[k*M+j];
      }
    }
  }
}

void multiply(float64_t *A_in, float64_t *B_in, float32_t *C_out,
    uint8_t N, uint8_t M, uint8_t P) {
  for(uint8_t p = 0; p < M*P; p++) {
    C_out[p] = 0;
  }
  for(uint8_t i = 0; i < N; i++) {
    for(uint8_t j = 0; j < M; j++) {
      for(uint8_t k = 0; k < P; k++) {
        C_out[k*N+i] += A_in[j*N+i]*B_in[k*M+j];
      }
    }
  }
}

// transpose() function
// Matrix transpose
// --------------------
void transpose(double *A_in, double *B_out,uint8_t N, uint8_t M) {
  for(uint8_t i = 0; i < M; i++) {
    for(uint8_t j = 0; j < N; j++) {
      B_out[j*M+i] = A_in[i*N+j];
    }
  }
}
```

Listing X.15 - MP-6DOF-XXX_CB math_helper.cpp

```
// Print helper - print_helper.h
// Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Masinski fakultet Univerziteta u Beogradu
// --------------------

#ifndef PRINT_HELPER_H
#define PRINT_HELPER_H

#include "config.h" // Configuration

#include <Arduino.h>

// CMSIS DSP Library
```

```
15 // Author: ARM KEIL
16 // http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html
17 #include <CMSIS_DSP.h>
18
19 void dispMatrix(float32_t *A_in, uint8_t N,  uint8_t M);
20 void dispMatrix(float64_t *A_in, uint8_t N,  uint8_t M);
21
22 void dispRowVector(float32_t *A_in, uint8_t M);
23 void dispRowVector(uint8_t *A_in, uint8_t M);
24 void dispRowVector(uint16_t *A_in, uint8_t M);
25 void dispRowVector(uint32_t *A_in, uint8_t M);
26 void dispRowVector(int8_t *A_in, uint8_t M);
27 void dispRowVector(int16_t *A_in, uint8_t M);
28 void dispRowVector(int32_t *A_in, uint8_t M);
29
30 void printVars();
31 void printStatus();
32
33 #endif // PRINT_HELPER_H
```

Listing X.16 - MP-6DOF-XXX_CB print_helper.h

```
1  // Print helper - print_helper.cpp
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "print_helper.h"
8
9  // dispMatrix() function
10 // Matrix display
11 // --------------------
12 void dispMatrix(float32_t *A_in, uint8_t N,  uint8_t M) {
13   for(uint8_t i = 0; i < N; i++) {
14     SerialUSB.print("  ");
15     for(uint8_t j = 0; j < M; j++) {
16       SerialUSB.print(A_in[j*N+i], 5);
17       SerialUSB.print(" ");
18     }
19     SerialUSB.println();
20   }
21 }
22
23 void dispMatrix(float64_t *A_in, uint8_t N,  uint8_t M) {
24   for(uint8_t i = 0; i < N; i++) {
25     SerialUSB.print("  ");
26     for(uint8_t j = 0; j < M; j++) {
27       SerialUSB.print(A_in[j*N+i], 8);
28       SerialUSB.print(" ");
29     }
30     SerialUSB.println();
31   }
32 }
33
34 // dispRowVector() function
35 // Vector display
36 // --------------------
37 void dispRowVector(float32_t *A_in, uint8_t M) {
38   for(uint8_t j = 0; j < M; j++) {
```

```
39      SerialUSB.print(A_in[j], 5);
40      SerialUSB.print(" ");
41    }
42  }
43
44  void dispRowVector(uint8_t *A_in, uint8_t M) {
45    for(uint8_t j = 0; j < M; j++) {
46      SerialUSB.print(A_in[j]);
47      SerialUSB.print(" ");
48    }
49  }
50
51  void dispRowVector(uint16_t *A_in, uint8_t M) {
52    for(uint8_t j = 0; j < M; j++) {
53      SerialUSB.print(A_in[j]);
54      SerialUSB.print(" ");
55    }
56  }
57
58  void dispRowVector(uint32_t *A_in, uint8_t M) {
59    for(uint8_t j = 0; j < M; j++) {
60      SerialUSB.print(A_in[j]);
61      SerialUSB.print(" ");
62    }
63  }
64
65  void dispRowVector(int8_t *A_in, uint8_t M) {
66    for(uint8_t j = 0; j < M; j++) {
67      SerialUSB.print(A_in[j]);
68      SerialUSB.print(" ");
69    }
70  }
71
72  void dispRowVector(int16_t *A_in, uint8_t M) {
73    for(uint8_t j = 0; j < M; j++) {
74      SerialUSB.print(A_in[j]);
75      SerialUSB.print(" ");
76    }
77  }
78
79  void dispRowVector(int32_t *A_in, uint8_t M) {
80    for(uint8_t j = 0; j < M; j++) {
81      SerialUSB.print(A_in[j]);
82      SerialUSB.print(" ");
83    }
84  }
```

Listing X.17 - MP-6DOF-XXX_CB print_helper.cpp

```
1  // Debug - debug.h
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef DEBUG_H
8  #define DEBUG_H
9
10 #ifdef DEBUG_MAIN
11 // printVars() function
```

```
12 // Print variable values
13 // --------------------
14 void printVars() {
15   // t
16   SerialUSB.print("t = ");
17   SerialUSB.println(millis());
18
19   // state
20   SerialUSB.print("state = ");
21   SerialUSB.println(state);
22
23   // IO.state
24   SerialUSB.print("IO.state = ");
25   SerialUSB.println(IO.state);
26
27   // MC.state
28   SerialUSB.print("MC.state = ");
29   SerialUSB.println(MC.state);
30
31   // SB_state
32   SerialUSB.print("SB_state = [ ");
33   dispRowVector(SB_state, N_ACT);
34   SerialUSB.println("]");
35
36   // MP.state
37   SerialUSB.print("MP.state = ");
38   SerialUSB.println(MP.state);
39
40   // M6RSS.state
41   SerialUSB.print("M6RSS.state = ");
42   SerialUSB.println(M6RSS.state);
43
44   // SBs_enable
45   SerialUSB.print("SBs_enable = ");
46   SerialUSB.println(SBs_enable);
47
48   // f_max
49   SerialUSB.print("f_max = ");
50   SerialUSB.println(f_max);
51
52   // Kp
53   SerialUSB.print("Kp = ");
54   SerialUSB.println(Kp);
55
56   // Ki
57   SerialUSB.print("Ki = ");
58   SerialUSB.println(Ki);
59
60   // Kd
61   SerialUSB.print("Kd = ");
62   SerialUSB.println(Kd);
63
64   // alpha_m
65   SerialUSB.print("alpha_m = [ ");
66   dispRowVector(alpha_m, N_ACT);
67   SerialUSB.println("]");
68
69   // temp
70   SerialUSB.print("temp = [ ");
```

```
71    dispRowVector(temp, N_ACT);
72    SerialUSB.println("]");
73
74    // f_out
75    SerialUSB.print("f_out = [ ");
76    dispRowVector(f_out, N_ACT);
77    SerialUSB.println("]");
78
79    // alpha_w
80    SerialUSB.print("alpha_w = [ ");
81    dispRowVector(alpha_w, N_ACT);
82    SerialUSB.println("]");
83  }
84
85  // printStatus() function
86  // Print variable values at right time
87  // --------------------
88  void printStatus() {
89    static uint32_t t_send = millis();
90
91    if((millis()-t_send) > PRINT_STATUS_TIME) {
92      t_send = millis();
93      printVars();
94    }
95  }
96  #endif
97
98  #endif // DEBUG_H
```

Listing X.18 - MP-6DOF-XXX_CB debug.h

```
1   // Regression test functions
2   // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Masinski fakultet Univerziteta u Beogradu
5   // --------------------
6
7   // Print functions
8   // --------------------
9   #include "print_helper.h"
10
11  // runRegressionTest() function
12  // Run regression test and print results
13  // --------------------
14  void runRegressionTest() {
15    // Uglovi donjih poluga
16    alpha_w[0] = 10;
17    alpha_w[1] = -2;
18    alpha_w[2] = 3;
19    alpha_w[3] = 4;
20    alpha_w[4] = -4;
21    alpha_w[5] = 1;
22
23    M6RSS.setDesiredAlphas(alpha_w);
24
25    // Proracun
26    SerialUSB.println();
27    SerialUSB.println(" Full regression test");
28
29    // Input
```

```
30    SerialUSB.println();
31    SerialUSB.println(" --------------------");
32    SerialUSB.println(" Input");
33    SerialUSB.println();
34
35
36    SerialUSB.print(" alpha_w = [ ");
37    dispRowVector(alpha_w, N_ACT);
38    SerialUSB.println("]");
39
40    // getGeometry()
41    SerialUSB.println();
42    SerialUSB.println(" --------------------");
43    SerialUSB.println(" getGeometry()");
44    SerialUSB.println();
45    M6RSS.getGeometry();
46
47    SerialUSB.println(" Rm2_i_ll = [ ");
48    dispMatrix(M6RSS.Rm2_i_ll, 3, N_ACT);
49    SerialUSB.println(" ]");
50    SerialUSB.println(" Rm1_i_ll = [ ");
51    dispMatrix(M6RSS.Rm1_i_ll, 3, N_ACT);
52    SerialUSB.println(" ]");
53
54    SerialUSB.println(" P = [ ");
55    dispMatrix(M6RSS.P, 3, N_ACT);
56    SerialUSB.println(" ]");
57    SerialUSB.println(" B = [ ");
58    dispMatrix(M6RSS.B, 3, N_ACT);
59    SerialUSB.println(" ]");
60    SerialUSB.println(" A = [ ");
61    dispMatrix(M6RSS.A, 3, N_ACT);
62    SerialUSB.println(" ]");
63
64    SerialUSB.println(" Rm2_BA = [ ");
65    dispMatrix(M6RSS.Rm2_BA, 3, N_ACT);
66    SerialUSB.println(" ]");
67    SerialUSB.println(" Rm1_BA = [ ");
68    dispMatrix(M6RSS.Rm1_BA, 3, N_ACT);
69    SerialUSB.println(" ]");
70
71    SerialUSB.print(" e = [ ");
72    dispRowVector(M6RSS.e, N_ACT);
73    SerialUSB.println("]");
74
75    // getParameters()
76    SerialUSB.println();
77    SerialUSB.println(" --------------------");
78    SerialUSB.println(" getParameters()");
79    SerialUSB.println();
80
81    M6RSS.getParameters();
82
83    SerialUSB.println(" J_x = [ ");
84    dispMatrix(M6RSS.J_x_f64, N_ACT, N_ACT);
85    SerialUSB.println(" invJ_x = [ ");
86    dispMatrix(M6RSS.invJ_x_f64, N_ACT, N_ACT);
87    SerialUSB.println(" J_a = [ ");
88    dispMatrix(M6RSS.J_a, N_ACT, N_ACT);
```

```
89    SerialUSB.println(" J = [ ");
90    dispMatrix(M6RSS.J, N_ACT, N_ACT);
91    SerialUSB.println(" ]");
92    SerialUSB.print(" x = [ ");
93    dispRowVector(M6RSS.x, N_ACT);
94    SerialUSB.println("]");
95    SerialUSB.print(" n_iter = ");
96    SerialUSB.println(M6RSS.n_iter);
97    SerialUSB.print(" e = [ ");
98    dispRowVector(M6RSS.e, N_ACT);
99    SerialUSB.println("]");
100
101   // checkJoints()
102   SerialUSB.println();
103   SerialUSB.println(" --------------------");
104   SerialUSB.println(" checkJoints()");
105   SerialUSB.println();
106
107   M6RSS.checkJoints();
108
109   SerialUSB.print(" lambda_p = [ ");
110   dispRowVector(M6RSS.lambda_p, N_ACT);
111   SerialUSB.println("]");
112   SerialUSB.print(" lambda_b = [ ");
113   dispRowVector(M6RSS.lambda_b, N_ACT);
114   SerialUSB.println("]");
115   SerialUSB.print(" e = [ ");
116   dispRowVector(M6RSS.e, N_ACT);
117   SerialUSB.println("]");
118
119   // getAlpha()
120   SerialUSB.println();
121   SerialUSB.println(" --------------------");
122   SerialUSB.println(" getAlpha()");
123   SerialUSB.println();
124
125   M6RSS.getAlpha();
126
127   SerialUSB.println(" R = [ ");
128   dispMatrix(M6RSS.R, 3, 3);
129   SerialUSB.println(" ]");
130
131   SerialUSB.println(" Pb = [ ");
132   dispMatrix(M6RSS.Pb, 3, N_ACT);
133   SerialUSB.println(" ]");
134   SerialUSB.println(" A = [ ");
135   dispMatrix(M6RSS.A, 3, N_ACT);
136   SerialUSB.println(" ]");
137
138   SerialUSB.println(" i_jpb = [ ");
139   dispMatrix(M6RSS.i_jpb, 3, N_ACT);
140   SerialUSB.println(" ]");
141   SerialUSB.println(" i_ll = [ ");
142   dispMatrix(M6RSS.i_ll, 3, N_ACT);
143   SerialUSB.println(" ]");
144
145   SerialUSB.print(" alpha = [ ");
146   for(uint8_t j = 0; j < N_ACT; j++) {
147     SerialUSB.print(M6RSS.alpha[j]*R2D, 5);
```

```
148       SerialUSB.print(" ");
149     }
150     SerialUSB.println("]");
151     SerialUSB.print(" e = [ ");
152     dispRowVector(M6RSS.e, N_ACT);
153     SerialUSB.println("]");
154 
155     // Kraj programa
156     SerialUSB.println();
157     SerialUSB.println(" --------------------");
158 
159     M6RSS.reset();
160     fill(0, alpha_w, N_ACT);
161 }
```

Listing X.19 - `MP-6DOF-XXX_CB regression_test.h`

X.2 Servo Board

Glavni deo programa `MP-6DOF-XXX_SB` za elektroniku `Servo Board` je dat u listingu X.20, dok su u svim ostalim listinzima dati kodovi za module i konfiguracione fajlove (listinzi X.21, X.22, X.23, X.24, X.25, X.26, X.27, X.28, X.29, X.30 i X.31).

```
1  // Servo Board program
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6  // Board: "PRDC STM32 boards"
7  // Board part number: "platformController"
8  // U(S)ART support: "Enabled (generic 'Serial')"
9  // USB Support (if available): "CDC (generic 'Serial' supersede U(S)ART)"
10 // USB speed (if available): "Low/Full Speed"
11 // Optimize: "Fastest (-O3)"
12 // Debug symbols and core logs: "None"
13 // C Runtime Library: "Newlib Nano (default)"
14 // Upload method: "STM32CubeProgrammer (SWD)"
15 // --------------------
16 
17 // Configuration
18 #include "config.h"
19 
20 // Classes
21 #include "BASIC_IO.h"
22 #include "ANGLE_SENSOR.h"
23 #include "CONTROL_BOARD.h"
24 #include "ANGLE_CONTROLLER.h"
25 
26 // Libraries
27 // --------------------
28 // PRDC RS485 STM32 Half Duplex
29 // Author: PR-DC
30 #include <PRDC_RS485HD_STM32.h>
31 
32 // PRDC VFD ENC EN600
33 // Author: PR-DC
34 #include <PRDC_VFD_ENC_EN600.h>
35 
```

```
36 // Define variables
37 // --------------------
38 uint8_t state = UNINITIALIZED_STATE;
39 uint8_t enable;
40
41 double alpha_w = 0; // [deg] current desired angle
42 double alpha_m = 0; // [deg] measured angle
43 double temp = 0; // [Hz]
44 double f_out = 0; // [Hz]
45 float f_max = 0;
46 double Kp = 0;
47 double Ki = 0;
48 double Kd = 0;
49
50 BasicIO IO;
51 AngleSensor AS;
52 ControlBoard CB;
53 AngleController AC;
54 PRDC_VfdEncEN600 VFD;
55
56 dataOutCB_t dataOutCB;
57
58 TwoWire DAC_I2C(MCP_SDA_PIN, MCP_SCL_PIN);
59 PRDC_RS485HD_STM32 VFD_RS485(VFD_RX_PIN, VFD_TX_PIN);
60 PRDC_RS485HD_STM32 AS_RS485(COMM_RX_PIN, COMM_TX_PIN);
61 HardwareSerial CB_HS(RS485FD_RX_PIN, RS485FD_TX_PIN);
62
63 // Debug
64 #ifdef DEBUG_MAIN
65 #include "debug.h"
66 #endif
67
68 // setup() function
69 // --------------------
70 void setup() {
71   // Set state
72   state = DISABLED_STATE;
73
74   IO.setup();
75
76   // Communication settings
77   SerialUSB.begin(PC_USB_BAUDRATE);
78 #ifdef WAIT_SERIAL
79   uint32_t t = millis();
80   while(!SerialUSB.available() && ((millis()-t) < WAIT_SERIAL_TIME)) {};
81 #endif
82
83   // Submodules setup
84 #ifdef DEBUG_MAIN
85   SerialUSB.println("@main: Submodules setup");
86 #endif
87   AS.setup(&AS_RS485, COMM_DE_PIN, COMM_RE_PIN);
88   CB.setup(&CB_HS);
89   VFD.setup(&DAC_I2C, &VFD_RS485, VFD_DE_PIN, VFD_RE_PIN, VFD_FWD_PIN,
       VFD_REV_PIN);
90   AC.setup();
91
92   // Startup
93 #ifdef DEBUG_MAIN
```

```
94    SerialUSB.println("@main: Waiting for VFD");
95  #endif
96    VFD.wait();
97  #ifdef DEBUG_MAIN
98    SerialUSB.println("@main: Waiting for Angle Sensor");
99  #endif
100   AS.wait();
101 #ifdef DEBUG_MAIN
102   SerialUSB.println("@main: Waiting for Control Board");
103 #endif
104   CB.wait();
105
106   // Flush inputs
107   AS.flushInput();
108   VFD.flushInput();
109   CB.flushInput();
110
111   // Init IO inputs
112   IO.initInputs();
113
114   // Reset timeouts
115   AS.resetTimeout();
116   VFD.resetTimeout();
117
118   // Check state
119   checkState();
120 }
121
122 // loop() function
123 // --------------------
124 void loop(){
125 #ifdef DEBUG_LOOP_TIME
126   static uint32_t t_loop = micros();
127
128   SerialUSB.print("@main: Loop time: ");
129   SerialUSB.println(micros()-t_loop);
130   t_loop = micros();
131 #endif
132
133   IO.loop();
134
135   // Read Control Board
136   if(CB.read()) {
137     CB.getData(&enable, &alpha_w, &f_max, &Kp, &Ki, &Kd);
138   }
139   // Read Angle sensor
140   if(AS.read()) {
141     AS.getData(&alpha_m, &temp);
142   }
143   VFD.read();
144   checkState();
145
146   if(state <= NORMAL_STATE) {
147     if(enable) {
148       if(state != NORMAL_STATE) {
149         state = NORMAL_STATE;
150         AC.resume();
151       }
152
```

```
153         if(AC.update(alpha_m, alpha_w, f_max, Kp, Ki, Kd)) {
154           f_out = AC.getControl();
155           VFD.setControl(f_out);
156         }
157       } else if(state != DISABLED_STATE) {
158         state = DISABLED_STATE;
159         AC.pause();
160       }
161     }
162
163 #ifdef DEBUG_MAIN
164     // Print status
165     printStatus();
166 #endif
167
168     // Send status
169     sendStatus();
170 }
171
172 // checkState() function
173 // Check if no errors
174 // --------------------
175 void checkState() {
176     if(state <= NORMAL_STATE) { // Prevents error override
177       if(CB.state > NORMAL_STATE) {
178         state = CB.state;
179       } else if(AS.state > NORMAL_STATE) {
180         state = AS.state;
181       } else if(VFD.state > NORMAL_STATE) {
182         state = VFD.state;
183       } else if(AC.state > NORMAL_STATE) {
184         state = AC.state;
185       }
186     }
187 }
188
189 // sendStatus() function
190 // Send status data
191 // --------------------
192 void sendStatus() {
193     static uint32_t t_send = micros();
194
195     if((micros()-t_send) > STATUS_TIME) {
196       t_send = micros();
197       dataOutCB.p.state = state;
198       dataOutCB.p.alpha_m = (float)alpha_m;
199       dataOutCB.p.temp = (float)temp;
200       dataOutCB.p.f_out = (float)f_out;
201       dataOutCB.p.switches = IO.switches;
202       CB.sendData(dataOutCB);
203     }
204 }
```

Listing X.20 - `MP-6DOF-XXX_SB.ino`

```
1 // Servo CoProcessor configuration parameters
2 // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3 // Doktorska disertacija
4 // Mašinski fakultet Univerziteta u Beogradu
5 // --------------------
```

```
6
7  #ifndef SERVO_BOARD_CONFIG_H
8  #define SERVO_BOARD_CONFIG_H
9
10 // Optional flags
11 //#define WITHOUT_ERROR_CHECKING
12 //#define WAIT_SERIAL
13 //#define DEBUG_MAIN
14 //#define DEBUG_LOOP_TIME
15 //#define DEBUG_CB_SEND_TIME
16 //#define DEBUG_VFD_ENC_EN600
17 //#define DEBUG_ANGLE_CONTROLLER
18 //#define DEBUG_ANGLE_SENSOR
19 //#define DEBUG_CONTROL_BOARD
20 //#define DEBUG_CONTROL_BOARD_PACKETS
21 //#define DEBUG_MOTOR_STALL_CHECK
22 //#define DEBUG_INPUT_READ
23 //#define DEBUG_INPUT_READ_INIT
24
25 //#define DEBUG_LEVEL 1
26
27 // Communication
28 #define PC_USB_BAUDRATE 1000000
29 #define CB_BAUDRATE 576000
30 #define AS_BAUDRATE 576000
31 #define VFD_RS485_BAUDRATE 57600 // [bps] VDF communication baudrate
32 #define VFD_RS485_MODE RS485_7E1 // [-] VDF serial com mode
33
34 // Timings
35 #define VFD_DELAY_TIME 3000 // [ms]
36 #define VFD_WAIT_TIMEOUT_TIME 60000 // [ms]
37 #define VFD_STATE_TIMEOUT_TIME 50 // [ms]
38 #define VFD_VALID_READ_TIMEOUT_TIME 800 // [ms]
39 #define VFD_READ_TIMEOUT_TIME 400 // [ms]
40 #define ANGLE_SENSOR_WAIT_TIMEOUT_TIME 10000 // [ms]
41 #define ANGLE_SENSOR_READ_TIMEOUT_TIME 50 // [ms]
42 #define PRINT_STATUS_TIME 1000 // [ms]
43 #define STATUS_TIME 10000 // [us]
44 #define LED_TIME 100 // [ms]
45 #define MOTOR_STALL_TIME 2000 // [ms]
46 #define MOTOR_STALL_CHECK_TIME 100 // [ms]
47 #define WAIT_SERIAL_TIME 15000 // [ms]
48 #define READ_INPUTS_TIME 2 // [ms]
49
50 // Angle controller
51 #define ANGLE_PID_SAMPLE_TIME 10 // [ms]
52
53 // Other
54 #define ACTUATOR_RPM 1410.0 // [rpm] nominal number of rotations per minute (50
       Hz)
55 #define ACTUATOR_I 50.0 // [-] worm gear ration
56
57 #define DAC_ADDR 0x60
58 #define DAC_MAX 4095 // [-] Max DAC output
59 #define VFD_F_MIN 0.1 // [Hz]
60 #define VFD_F_MAX 50.0 // [Hz]
61 #define ANGLE_MIN 0.0 // [deg]
62 #define ANGLE_MAX 360.0 // [deg]
63
```

```
64 #define ANGLE_ERROR_THRESHOLD 5.0 // [deg]
65 #define ANGLE_ERROR_RESET_THRESHOLD 1.0 // [deg]
66 #define ANGLE_VELOCITY_THRESHOLD 1.0 // [deg/s]
67
68 #define SWITCH_NUMBER 4
69 #define SWITCH_READINGS_NUMBER 5
70
71 // Enums
72 enum SB_STATE {
73   UNINITIALIZED_STATE = 0,
74   DISABLED_STATE,
75   NORMAL_STATE,
76   ANGLE_SENSOR_WAIT_TIMEOUT,
77   VFD_WAIT_TIMEOUT,
78   VFD_READ_TIMEOUT,
79   VFD_VALID_READ_TIMEOUT,
80   ANGLE_SENSOR_READ_TIMEOUT,
81   MOTOR_STALL,
82   VFD_ERROR = 100
83 };
84
85 #endif
```

Listing X.21 - MP-6DOF-XXX_SB config.h

```
1 -DSERIAL_RX_BUFFER_SIZE=1024 -DSERIAL_TX_BUFFER_SIZE=1024
```

Listing X.22 - MP-6DOF-XXX_SB build_opt.h

```
1  // BasicIO - BASIC_IO.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef BASIC_IO_H
8  #define BASIC_IO_H
9
10 #include "config.h" // Configuration
11
12 #include <Arduino.h>
13
14 class BasicIO {
15  public:
16   BasicIO();
17   ~BasicIO();
18
19   uint32_t state = UNINITIALIZED_STATE;
20
21   void setup();
22   void initInputs();
23   void loop();
24   void readInputs();
25   void blinkLEDs();
26
27   uint32_t switch_pins[SWITCH_NUMBER] = {
28     SWITCH_1_PIN,
29     SWITCH_2_PIN,
30     SWITCH_3_PIN,
31     SWITCH_4_PIN
```

```
32    };
33    uint8_t switch_readings[SWITCH_NUMBER] = {0};
34    bool switch_state[SWITCH_NUMBER] = {0};
35    uint8_t switches;
36  };
37
38  #endif // BASIC_IO_H
```

Listing X.23 - MP-6DOF-XXX_SB BASIC_IO.h

```
1   // BasicIO - BASIC_IO.cpp
2   // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Mašinski fakultet Univerziteta u Beogradu
5   // ---------------------
6
7   #include "BASIC_IO.h"
8
9   // BasicIO()
10  // Object constructor
11  // ---------------------
12  BasicIO::BasicIO() {}
13
14  // ~BasicIO()
15  // Object destructor
16  // ---------------------
17  BasicIO::~BasicIO() {}
18
19  // setup() function
20  // ---------------------
21  void BasicIO::setup() {
22  #ifdef DEBUG_BASIC_IO
23    if(DEBUG_LEVEL > 0) {
24      SerialUSB.println("@BasicIO: setup()");
25    }
26  #endif
27
28    // Set pin modes
29    pinMode(LED_1_PIN, OUTPUT);
30    pinMode(LED_2_PIN, OUTPUT);
31    pinMode(LED_3_PIN, OUTPUT);
32    pinMode(SWITCH_1_PIN, INPUT_PULLUP);
33    pinMode(SWITCH_2_PIN, INPUT_PULLUP);
34    pinMode(SWITCH_3_PIN, INPUT_PULLUP);
35    pinMode(SWITCH_4_PIN, INPUT_PULLUP);
36
37    // Initial otuputs
38    digitalWrite(LED_1_PIN, HIGH);
39    digitalWrite(LED_2_PIN, HIGH);
40    digitalWrite(LED_3_PIN, HIGH);
41
42    this->state = NORMAL_STATE;
43  }
44
45  // initInputs() function
46  // ---------------------
47  void BasicIO::initInputs() {
48
49    // Initial inputs
50    for(uint8_t j = 0; j < SWITCH_READINGS_NUMBER*2; j++) {
```

```
51      delay(READ_INPUTS_TIME);
52      for(uint8_t i = 0; i < SWITCH_NUMBER; i++) {
53        if(digitalRead(this->switch_pins[i])) {
54          if(this->switch_readings[i] < SWITCH_READINGS_NUMBER) {
55            this->switch_readings[i]++;
56          }
57        } else {
58          if(this->switch_readings[i] > 0) {
59            this->switch_readings[i]--;
60          }
61        }
62      }
63    }
64
65    for(uint8_t i = 0; i < SWITCH_NUMBER; i++) {
66  #ifdef DEBUG_INPUT_READ_INIT
67      uint8_t readings = this->switch_readings[i];
68  #endif
69      if(this->switch_readings[i] > SWITCH_READINGS_NUMBER/2.0) {
70        this->switch_state[i] = true;
71        this->switch_readings[i] = SWITCH_READINGS_NUMBER;
72      } else {
73        this->switch_state[i] = false;
74        this->switch_readings[i] = 0;
75      }
76  #ifdef DEBUG_INPUT_READ_INIT
77      SerialUSB.print("@BasicIO: Initialized input ");
78      SerialUSB.print(i);
79      SerialUSB.print(" to ");
80      SerialUSB.print(this->switch_state[i]);
81      SerialUSB.print(" (");
82      SerialUSB.print(readings);
83      SerialUSB.println(")");
84  #endif
85    }
86  #ifdef DEBUG_INPUT_READ_INIT
87    SerialUSB.println();
88  #endif
89  }
90
91  // loop() function
92  // --------------------
93  void BasicIO::loop() {
94  #ifdef DEBUG_BASIC_IO
95    if(DEBUG_LEVEL > 1) {
96      SerialUSB.println("@BasicIO: loop()");
97    }
98  #endif
99
100   this->readInputs();
101   this->blinkLEDs();
102 }
103
104 // readInputs() function
105 // Read all pots
106 // --------------------
107 void BasicIO::readInputs() {
108 #ifdef DEBUG_BASIC_IO
109   if(DEBUG_LEVEL > 1) {
```

```
110     SerialUSB.println("@BasicIO: readInputs()");
111   }
112 #endif
113 
114   static uint32_t t_read = 0;
115   if((millis()-t_read) > READ_INPUTS_TIME) {
116     t_read = millis();
117     // Read switches
118     for(uint8_t i = 0; i < SWITCH_NUMBER; i++) {
119       if(digitalRead(this->switch_pins[i])) {
120         if(this->switch_readings[i] < SWITCH_READINGS_NUMBER) {
121           this->switch_readings[i]++;
122         }
123       } else {
124         if(this->switch_readings[i] > 0) {
125           this->switch_readings[i]--;
126         }
127       }
128       if(this->switch_readings[i] == 0 && this->switch_state[i]) {
129         this->switch_state[i] = false;
130       } else if(this->switch_readings[i] == SWITCH_READINGS_NUMBER && !this->
            switch_state[i]) {
131         this->switch_state[i] = true;
132       }
133 #ifdef DEBUG_INPUT_READ
134       SerialUSB.print("@BasicIO: Input ");
135       SerialUSB.print(i);
136       SerialUSB.print(" state ");
137       SerialUSB.print(this->switch_state[i]);
138       SerialUSB.print(" (");
139       SerialUSB.print(this->switch_readings[i]);
140       SerialUSB.println(")");
141 #endif
142     }
143 #ifdef DEBUG_INPUT_READ
144     SerialUSB.println();
145 #endif
146   }
147 
148   for(uint8_t i = 0; i < SWITCH_NUMBER; i++) {
149     if(this->switch_state[i]) {
150       this->switches &= ~(1 << i);
151     } else {
152       this->switches |= 1 << i;
153     }
154   }
155 }
156 
157 // blinkLEDs() function
158 // Blink LEDs at right time
159 // --------------------
160 void BasicIO::blinkLEDs() {
161 #ifdef DEBUG_BASIC_IO
162   if(DEBUG_LEVEL > 2) {
163     SerialUSB.println("@BasicIO: blinkLEDs()");
164   }
165 #endif
166   static uint8_t s = 0;
167   static uint32_t t_led = millis();
```

```

  if((millis()-t_led) > LED_TIME) {
    t_led = millis();

    s += 1;
    if(s > 2) {
      s = 0;
    }
    if(s == 0) {
      digitalWrite(LED_1_PIN, HIGH);
      digitalWrite(LED_2_PIN, LOW);
      digitalWrite(LED_3_PIN, LOW);
    } else if(s == 1) {
      digitalWrite(LED_1_PIN, LOW);
      digitalWrite(LED_2_PIN, HIGH);
      digitalWrite(LED_3_PIN, LOW);
    } else {
      digitalWrite(LED_1_PIN, LOW);
      digitalWrite(LED_2_PIN, LOW);
      digitalWrite(LED_3_PIN, HIGH);
    }
  }
}
```

Listing X.24 - MP-6DOF-XXX_SB BASIC_IO.cpp

```
// AngleController - ANGLE_CONTROLLER.h
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#ifndef ANGLE_CONTROLLER_H
#define ANGLE_CONTROLLER_H

#include "config.h" // Configuration

#include <Arduino.h>

#include <PID_v1.h>

class AngleController {
 public:
  AngleController();
  ~AngleController();

  uint32_t state = UNINITIALIZED_STATE;
  bool paused = false;

  void setup();
  bool update(double, double, float, double, double, double);
  void pause();
  void resume();
  double getControl();

  PID* ANGLE_PID;

  double angle_m;
  double frequency_output;
  double angle_w;
```

```

  float f_max;
  double Kp = 0;
  double Ki = 0;
  double Kd = 0;
};

#endif // ANGLE_CONTROLLER_H
```

Listing X.25 - MP-6DOF-XXX_SB ANGLE_CONTROLLER.h

```
// AngleController - ANGLE_CONTROLLER.cpp
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#include "ANGLE_CONTROLLER.h"

// AngleController()
// Object constructor
// --------------------
AngleController::AngleController() {}

// ~AngleController()
// Object destructor
// --------------------
AngleController::~AngleController() {}

// setup() function
// --------------------
void AngleController::setup() {
#ifdef DEBUG_ANGLE_CONTROLLER
  if(DEBUG_LEVEL > 0) {
    SerialUSB.println("@AngleController: setup()");
  }
#endif

  this->ANGLE_PID = new PID(&angle_m, &frequency_output, &angle_w, Kp, Ki, Kd,
     DIRECT);
  this->ANGLE_PID->SetOutputLimits(-VFD_F_MAX, VFD_F_MAX);
  this->ANGLE_PID->SetSampleTime(ANGLE_PID_SAMPLE_TIME);
  this->ANGLE_PID->SetMode(AUTOMATIC);
  this->state = NORMAL_STATE;
}

// update() function
// --------------------
bool AngleController::update(double angle_m_in, double angle_w_in, float
    f_max_in, double Kp_in, double Ki_in, double Kd_in) {
#ifdef DEBUG_ANGLE_CONTROLLER
  if(DEBUG_LEVEL > 1) {
    SerialUSB.println("@AngleController: update()");
  }
#endif

#ifndef WITHOUT_ERROR_CHECKING
  static uint32_t t_update = millis();
  static double angle_m_in_last = 0;
  static uint32_t angle_error_time = 0;
```

```
48
49    if((millis() - t_update) > MOTOR_STALL_CHECK_TIME) {
50      uint32_t dt = millis() - t_update;
51      t_update = millis();
52
53      double angle_error = fabs(angle_w_in-angle_m_in);
54      double angle_change = fabs(angle_m_in-angle_m_in_last);
55      angle_m_in_last = angle_m_in;
56      double angle_velocity = angle_change/(float)dt*1E3; // [deg/s]
57
58      if(angle_error > ANGLE_ERROR_THRESHOLD) {
59        angle_error_time += dt;
60      }
61
62      if(angle_error_time > MOTOR_STALL_TIME) {
63        this->state = MOTOR_STALL;
64      } else if(angle_velocity > ANGLE_VELOCITY_THRESHOLD || angle_error <
          ANGLE_ERROR_RESET_THRESHOLD) {
65        angle_error_time = 0;
66      }
67
68  #ifdef DEBUG_MOTOR_STALL_CHECK
69      SerialUSB.print("@AngleController: angle_error = ");
70      SerialUSB.println(angle_error);
71      SerialUSB.print("@AngleController: angle_change = ");
72      SerialUSB.println(angle_change);
73      SerialUSB.print("@AngleController: angle_velocity = ");
74      SerialUSB.println(angle_velocity);
75      SerialUSB.print("@AngleController: angle_error_time = ");
76      SerialUSB.println(angle_error_time);
77  #endif
78    }
79  #endif
80
81    this->angle_m = angle_m_in;
82    this->angle_w = angle_w_in;
83    this->f_max = f_max_in;
84    this->Kp = Kp_in;
85    this->Ki = Ki_in;
86    this->Kd = Kd_in;
87
88    this->ANGLE_PID->SetOutputLimits(-this->f_max, this->f_max);
89    this->ANGLE_PID->SetTunings(this->Kp, this->Ki, this->Kd);
90
91    if(this->paused) {
92      return false;
93    }
94    return this->ANGLE_PID->Compute();
95  }
96
97  // getControl() function
98  // Return control value
99  // --------------------
100 double AngleController::getControl() {
101 #ifdef DEBUG_ANGLE_CONTROLLER
102   if(DEBUG_LEVEL > 1) {
103     SerialUSB.println("@AngleController: getControl()");
104   }
105 #endif
```

```
106
107   return this->frequency_output;
108 }
109
110 // pause() function
111 // Pause controller
112 // --------------------
113 void AngleController::pause() {
114 #ifdef DEBUG_ANGLE_CONTROLLER
115   if(DEBUG_LEVEL > 0) {
116     SerialUSB.println("@AngleController: pause()");
117   }
118 #endif
119
120   this->paused = true;
121   this->ANGLE_PID->SetMode(MANUAL);
122 }
123
124 // resume() function
125 // Resume controller
126 // --------------------
127 void AngleController::resume() {
128 #ifdef DEBUG_ANGLE_CONTROLLER
129   if(DEBUG_LEVEL > 0) {
130     SerialUSB.println("@AngleController: resume()");
131   }
132 #endif
133
134   this->paused = false;
135   this->ANGLE_PID->SetMode(AUTOMATIC);
136 }
```

Listing X.26 - MP-6DOF-XXX_SB ANGLE_CONTROLLER.cpp

```
1  // AngleSensor - ANGLE_SENSOR.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef ANGLE_SENSOR_H
8  #define ANGLE_SENSOR_H
9
10 #include "config.h" // Configuration
11
12 #include <Arduino.h>
13
14 #include <PRDC_RS485HD_STM32.h>
15 #include <PRDC_SCBP.h>
16
17 struct __attribute__((packed)) packetInAS_t {
18   float angle;
19   float temp;
20   uint32_t N;
21 };
22 union dataInAS_t{
23   packetInAS_t p;
24   uint8_t bytes[sizeof(packetInAS_t)];
25 };
26
```

```
27 struct __attribute__((packed)) packetOutAS_t {
28   uint8_t enable;
29 };
30 union dataOutAS_t{
31   packetOutAS_t p;
32   uint8_t bytes[sizeof(packetOutAS_t)];
33 };
34
35 class AngleSensor {
36  public:
37   AngleSensor();
38   ~AngleSensor();
39
40   uint32_t state = UNINITIALIZED_STATE;
41
42   void setup(PRDC_RS485HD_STM32*, uint32_t, uint32_t);
43   bool read();
44   void resetTimeout();
45   void flushInput();
46   void getData(double*, double*);
47   void wait();
48
49   PRDC_RS485HD_STM32* RS485;
50   PRDC_SCBP COM;
51   dataOutAS_t dataOut;
52   dataInAS_t dataIn;
53
54   bool new_angle = false;
55   float angle = 0;
56   float temp = 0;
57   uint32_t N_in = 0;
58   uint32_t n_no_message = 0;
59
60   uint32_t t_new_message;
61 };
62
63 #endif // ANGLE_SENSOR_H
```

Listing X.27 - MP-6DOF-XXX_SB ANGLE_SENSOR.h

```
1  // AngleSensor - ANGLE_SENSOR.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "ANGLE_SENSOR.h"
8
9  // AngleSensor()
10 // Object constructor
11 // --------------------
12 AngleSensor::AngleSensor() {}
13
14 // ~AngleSensor()
15 // Object destructor
16 // --------------------
17 AngleSensor::~AngleSensor() {}
18
19 // setup() function
20 // --------------------
```

```
21 void AngleSensor::setup(PRDC_RS485HD_STM32* RS485_in, uint32_t PIN_DE, uint32_t
       PIN_nRE) {
22 #ifdef DEBUG_ANGLE_SENSOR
23   if(DEBUG_LEVEL > 0) {
24     SerialUSB.println("@AngleSensor: setup()");
25   }
26 #endif
27
28   this->RS485 = RS485_in;
29
30   // AS communication
31   this->RS485->setPins(PIN_DE, PIN_nRE);
32   this->COM.begin(*this->RS485, AS_BAUDRATE);
33 }
34
35 // read() function
36 // --------------------
37 bool AngleSensor::read() {
38 #ifdef DEBUG_ANGLE_SENSOR
39   if(DEBUG_LEVEL > 1) {
40     SerialUSB.println("@AngleSensor: loop()");
41   }
42 #endif
43
44   bool got_message = false;
45
46   while(this->COM.availableData() > 0) {
47     bool flagIn = this->COM.readPacket(this->dataIn.bytes, sizeof(this->dataIn))
          ;
48     if(flagIn) {
49       this->t_new_message = millis();
50       this->new_angle = true;
51       this->angle = this->dataIn.p.angle;
52       this->temp = this->dataIn.p.temp;
53       this->N_in = this->dataIn.p.N;
54
55       this->n_no_message = 0;
56       got_message = true;
57     }
58   }
59
60   if(!got_message) {
61     this->n_no_message = this->n_no_message+1;
62   }
63
64 #ifndef WITHOUT_ERROR_CHECKING
65   if(this->state == NORMAL_STATE) {
66     // Check for angle sensor communication timeout
67     if((millis()-this->t_new_message) > ANGLE_SENSOR_READ_TIMEOUT_TIME && this->
          n_no_message > 1) {
68       this->state = ANGLE_SENSOR_READ_TIMEOUT;
69     }
70   }
71 #endif
72
73   return got_message;
74 }
75
76 // resetTimeout() function
```

```
77 // Reset timeout
78 // --------------------
79 void AngleSensor::resetTimeout() {
80 #ifdef DEBUG_ANGLE_SENSOR
81   if(DEBUG_LEVEL > 0) {
82     SerialUSB.println("@AngleSensor: resetTimeout()");
83   }
84 #endif
85
86   this->t_new_message = millis();
87 }
88
89 // flushInput() function
90 // Flush input
91 // --------------------
92 void AngleSensor::flushInput() {
93 #ifdef DEBUG_ANGLE_SENSOR
94   if(DEBUG_LEVEL > 0) {
95     SerialUSB.println("@AngleSensor: flushInput()");
96   }
97 #endif
98
99   this->COM.flushInput();
100 }
101
102 // getAngle() function
103 // Get angle
104 // --------------------
105 void AngleSensor::getData(double* angle_out, double* temp_out) {
106 #ifdef DEBUG_ANGLE_SENSOR
107   if(DEBUG_LEVEL > 1) {
108     SerialUSB.println("@AngleSensor: getAngle()");
109   }
110 #endif
111   *angle_out = this->angle;
112   *temp_out = this->temp;
113   this->new_angle = false;
114 }
115
116 // wait() function
117 // Wait for angle sensor to be active
118 // --------------------
119 void AngleSensor::wait() {
120 #ifdef DEBUG_ANGLE_SENSOR
121   if(DEBUG_LEVEL > 0) {
122     SerialUSB.println("@AngleSensor: wait()");
123   }
124 #endif
125
126   uint32_t t_wait = millis();
127   uint32_t t_enable = millis();
128
129   while(true) {
130     if(this->read()) {
131       break;
132     }
133
134     // Check for wait timeout
135     if((millis()-t_wait) > ANGLE_SENSOR_WAIT_TIMEOUT_TIME) {
```

```
136       t_wait = millis();
137       this->state = ANGLE_SENSOR_WAIT_TIMEOUT;
138       return;
139     }
140   }
141   this->state = NORMAL_STATE;
142 }
```

Listing X.28 - MP-6DOF-XXX_SB ANGLE_SENSOR.cpp

```
1  // ControlBoard - CONTROL_BOARD.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef CONTROL_BOARD_H
8  #define CONTROL_BOARD_H
9
10 #include "config.h" // Configuration
11
12 #include <Arduino.h>
13
14 #include <PRDC_SCBP.h>
15
16 struct __attribute__((packed)) packetInCB_t {
17   uint8_t enable;
18   float alpha_w;
19   float f_max;
20   float Kp;
21   float Ki;
22   float Kd;
23   uint32_t N;
24 };
25 union dataInCB_t{
26   packetInCB_t p;
27   uint8_t bytes[sizeof(packetInCB_t)];
28 };
29
30 struct __attribute__((packed)) packetOutCB_t {
31   uint8_t state;
32   float alpha_m;
33   float temp;
34   float f_out;
35   uint8_t switches;
36   uint32_t N;
37 };
38 union dataOutCB_t{
39   packetOutCB_t p;
40   uint8_t bytes[sizeof(packetOutCB_t)];
41 };
42
43 class ControlBoard {
44  public:
45   ControlBoard();
46   ~ControlBoard();
47
48   uint32_t state = UNINITIALIZED_STATE;
49
50   void setup(HardwareSerial*);
```

```
51    bool read();
52    void flushInput();
53    void getData(uint8_t*, double*, float*, double*, double*, double*);
54    void wait();
55    void sendData(dataOutCB_t);
56    void setData(dataOutCB_t);
57
58    HardwareSerial* HS;
59    PRDC_SCBP COM;
60    dataInCB_t dataIn;
61    dataOutCB_t dataOut;
62
63    bool new_data = false;
64    uint8_t enable;
65    double alpha_w;
66    float f_max;
67    double Kp;
68    double Ki;
69    double Kd;
70    uint32_t N_in = 0;
71    uint32_t N_out = 0;
72  };
73
74  #endif // CONTROL_BOARD_H
```

Listing X.29 - MP-6DOF-XXX_SB CONTROL_BOARD.h

```
1   // ControlBoard - CONTROL_BOARD.cpp
2   // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Mašinski fakultet Univerziteta u Beogradu
5   // --------------------
6
7   #include "CONTROL_BOARD.h"
8
9   // ControlBoard()
10  // Object constructor
11  // --------------------
12  ControlBoard::ControlBoard() {}
13
14  // ~ControlBoard()
15  // Object destructor
16  // --------------------
17  ControlBoard::~ControlBoard() {}
18
19  // setup() function
20  // --------------------
21  void ControlBoard::setup(HardwareSerial* HS_in) {
22  #ifdef DEBUG_CONTROL_BOARD
23    if(DEBUG_LEVEL > 0) {
24      SerialUSB.println("@ControlBoard: setup()");
25    }
26  #endif
27
28    this->HS = HS_in;
29
30    // CB communication
31    this->COM.begin(*this->HS, CB_BAUDRATE);
32  }
33
```

```
34 // getControlCB() function
35 // Get control from CB
36 // --------------------
37 bool ControlBoard::read() {
38 #ifdef DEBUG_CONTROL_BOARD
39   if(DEBUG_LEVEL > 1) {
40     SerialUSB.println("@ControlBoard: read()");
41   }
42 #endif
43
44   bool got_message = false;
45
46   while(this->COM.availableData() > 0) {
47     bool flagIn = this->COM.readPacket(this->dataIn.bytes, sizeof(this->dataIn))
         ;
48     if(flagIn) {
49       this->new_data = true;
50       this->enable = dataIn.p.enable;
51       this->alpha_w = dataIn.p.alpha_w;
52       this->f_max = dataIn.p.f_max;
53       this->Kp = dataIn.p.Kp;
54       this->Ki = dataIn.p.Ki;
55       this->Kd = dataIn.p.Kd;
56       this->N_in = dataIn.p.N;
57       got_message = true;
58 #ifdef DEBUG_CONTROL_BOARD_PACKETS
59       SerialUSB.print(millis());
60       SerialUSB.print(" - Got new CB packet ");
61       SerialUSB.println(dataIn.p.N);
62 #endif
63     }
64   }
65
66   return got_message;
67 }
68
69 // flushInput() function
70 // Flush input
71 // --------------------
72 void ControlBoard::flushInput() {
73 #ifdef DEBUG_CONTROL_BOARD
74   if(DEBUG_LEVEL > 0) {
75     SerialUSB.println("@ControlBoard: flushInput()");
76   }
77 #endif
78
79   this->COM.flushInput();
80 }
81
82 // getData() function
83 // Get data
84 // --------------------
85 void ControlBoard::getData(uint8_t* enable_out, double* alpha_w_out, float*
     f_max_out, double* Kp_out, double* Ki_out, double* Kd_out) {
86 #ifdef DEBUG_CONTROL_BOARD
87   if(DEBUG_LEVEL > 1) {
88     SerialUSB.println("@ControlBoard: getData()");
89   }
90 #endif
```

```
91
92    *enable_out = this->enable;
93    *alpha_w_out = this->alpha_w;
94    *f_max_out = this->f_max;
95    *Kp_out = this->Kp;
96    *Ki_out = this->Ki;
97    *Kd_out = this->Kd;
98    this->new_data = false;
99  }
100
101 // wait() function
102 // Wait for control board to be active
103 // --------------------
104 void ControlBoard::wait() {
105 #ifdef DEBUG_CONTROL_BOARD
106   if(DEBUG_LEVEL > 0) {
107     SerialUSB.println("@ControlBoard: wait()");
108   }
109 #endif
110
111   while(true) {
112     if(this->read()) {
113       break;
114     }
115   }
116   this->state = NORMAL_STATE;
117 }
118
119 // setData() function
120 // Set data
121 // --------------------
122 void ControlBoard::setData(dataOutCB_t dataOut_in) {
123 #ifdef DEBUG_CONTROL_BOARD
124   if(DEBUG_LEVEL > 1) {
125     SerialUSB.println("@ControlBoard: setData()");
126   }
127 #endif
128   dataOut.p.state = dataOut_in.p.state;
129   dataOut.p.alpha_m = dataOut_in.p.alpha_m;
130   dataOut.p.temp = dataOut_in.p.temp;
131   dataOut.p.f_out = dataOut_in.p.f_out;
132   dataOut.p.switches = dataOut_in.p.switches;
133 }
134
135 // sendData() function
136 // Send data
137 // --------------------
138 void ControlBoard::sendData(dataOutCB_t dataOut) {
139 #ifdef DEBUG_CONTROL_BOARD
140   if(DEBUG_LEVEL > 1) {
141     SerialUSB.println("@ControlBoard: sendData()");
142   }
143 #endif
144
145 #ifdef DEBUG_CB_SEND_TIME
146   static uint32_t t_send = micros();
147
148   SerialUSB.print("@ControlBoard: Send time: ");
149   SerialUSB.println(micros()-t_send);
```

```
150 #endif
151
152   this->N_out = this->N_out+1;
153   dataOut.p.N = this->N_out;
154   this->COM.sendPacket(dataOut.bytes, sizeof(dataOut));
155 }
```

Listing X.30 - MP-6DOF-XXX_SB CONTROL_BOARD.cpp

```
1  // Debug - debug.h
2  // Autor: Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Masinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef DEBUG_H
8  #define DEBUG_H
9
10 #ifdef DEBUG_MAIN
11
12 // printVars() function
13 // Print variable values
14 // --------------------
15 void printVars() {
16   // t
17   SerialUSB.print("t = ");
18   SerialUSB.println(millis());
19
20   // state
21   SerialUSB.print("state = ");
22   SerialUSB.println(state);
23
24   // alpha_m
25   SerialUSB.print("alpha_m = ");
26   SerialUSB.println(alpha_m);
27
28   // temp
29   SerialUSB.print("temp = ");
30   SerialUSB.println(temp);
31
32   // f_out
33   SerialUSB.print("f_out = ");
34   SerialUSB.println(f_out);
35
36   // IO.switches
37   SerialUSB.print("IO.switches = ");
38   SerialUSB.println(IO.switches);
39
40   // enable
41   SerialUSB.print("enable = ");
42   SerialUSB.println(enable);
43
44   // alpha_w
45   SerialUSB.print("alpha_w = ");
46   SerialUSB.println(alpha_w);
47
48   // f_max
49   SerialUSB.print("f_max = ");
50   SerialUSB.println(f_max);
51
```

```
52    // Kp
53    SerialUSB.print("Kp = ");
54    SerialUSB.println(Kp);
55
56    // Ki
57    SerialUSB.print("Ki = ");
58    SerialUSB.println(Ki);
59
60    // Kd
61    SerialUSB.print("Kd = ");
62    SerialUSB.println(Kd);
63  }
64
65  // printStatus() function
66  // Print variable values at right time
67  // --------------------
68  void printStatus() {
69    static uint32_t t_send = millis();
70
71    if((millis()-t_send) > PRINT_STATUS_TIME) {
72      t_send = millis();
73      printVars();
74    }
75  }
76  #endif
77
78  #endif // DEBUG_H
```

Listing X.31 - MP-6DOF-XXX_SB debug.h

U listinzima X.32 i X.33 dati fajlovi biblioteke PRDC_VFD_ENC_EN600.

```
1   // PRDC_VfdEncEN600 - PRDC_VFD_ENC_EN600.h
2   // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Mašinski fakultet Univerziteta u Beogradu
5   // --------------------
6
7   #ifndef PRDC_VFD_ENC_EN600_H
8   #define PRDC_VFD_ENC_EN600_H
9
10  #include "config.h"
11
12  #include <Arduino.h>
13  #include <Wire.h>
14
15  #include <PRDC_RS485HD_STM32.h>
16  #include <PRDC_FreePort.h>
17  #include <PRDC_MCP4725.h>
18
19  #define VFD_RS485_BAUDRATE 57600 // [bps] VDF communication baudrate
20  #define VFD_RS485_MODE RS485_7E1 // [-] VDF serial com mode
21  #define DAC_MAX 4095 // [-] DAC MAX
22
23  class PRDC_VfdEncEN600 {
24   public:
25    PRDC_VfdEncEN600();
26    ~PRDC_VfdEncEN600();
27
28    uint32_t state = UNINITIALIZED_STATE;
29
```

```
30    void setup(TwoWire*, PRDC_RS485HD_STM32*, uint32_t, uint32_t, uint32_t,
          uint32_t);
31    bool read();
32    void resetTimeout();
33    void flushInput();
34    void handleMessage();
35    void getState();
36    void wait();
37    void setControl(double);
38    void changeDirPinState();
39
40    PRDC_RS485HD_STM32* RS485;
41    PRDC_FreePort VFD;
42    PRDC_MCP4725 DAC;
43    TwoWire* DAC_I2C;
44
45    uint32_t PIN_FWD;
46    uint32_t PIN_REV;
47    uint32_t PIN_SPD;
48
49    uint8_t FP_DA; // Auxilary device address (decimal)
50    char FP_DR[3]; // Auxilary device response (ASCII)
51    uint8_t FP_FI; // Failure index (decimal)
52    char FP_CI[3]; // Command index (ASCII)
53    uint16_t FP_RD; // Run data (decimal)
54
55    uint32_t n_no_message = 0;
56
57    double f_out;
58
59    bool FWD_STATE = LOW; // [-] FWD pin state
60    bool FWD_STATE_new = LOW; // [-] FWD pin new state
61    bool REV_STATE = LOW; // [-] REV pin state
62    bool REV_STATE_new = LOW; // [-] REV pin new state
63
64    uint32_t t_new_message;
65    uint32_t t_new_valid;
66
67    uint32_t N_in;
68    uint32_t N_out;
69 };
70
71 #endif // PRDC_VFD_ENC_EN600_H
```

Listing X.32 - PRDC_VFD_ENC_EN600.h

```
1  // PRDC_VfdEncEN600 - PRDC_VFD_ENC_EN600.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "PRDC_VFD_ENC_EN600.h"
8
9  // PRDC_VfdEncEN600()
10 // Object constructor
11 // --------------------
12 PRDC_VfdEncEN600::PRDC_VfdEncEN600() {}
13
14 // ~PRDC_VfdEncEN600()
```

```
15 // Object destructor
16 // --------------------
17 PRDC_VfdEncEN600::~PRDC_VfdEncEN600() {}
18
19 // setup() function
20 // --------------------
21 void PRDC_VfdEncEN600::setup(TwoWire* DAC_I2C_in, PRDC_RS485HD_STM32* RS485_in,
       uint32_t PIN_DE, uint32_t PIN_nRE, uint32_t PIN_FWD_in, uint32_t PIN_REV_in)
       {
22 #ifdef DEBUG_VFD_ENC_EN600
23   if(DEBUG_LEVEL > 0) {
24     SerialUSB.println("@VfdEncEN600: setup()");
25   }
26 #endif
27
28   this->RS485 = RS485_in;
29
30   this->PIN_FWD = PIN_FWD_in;
31   this->PIN_REV = PIN_REV_in;
32
33   this->DAC_I2C = DAC_I2C_in;
34
35   // Set pin modes
36   pinMode(this->PIN_FWD, OUTPUT);
37   pinMode(this->PIN_REV, OUTPUT);
38
39   // Initial otuputs
40   digitalWrite(this->PIN_FWD, LOW);
41   digitalWrite(this->PIN_REV, LOW);
42
43   // VFD communication
44   this->RS485->setPins(PIN_DE, PIN_nRE);
45   this->RS485->begin(VFD_RS485_BAUDRATE, VFD_RS485_MODE);
46   this->VFD.attach(*this->RS485);
47
48   // DAC begin
49   this->DAC.begin(DAC_ADDR, DAC_I2C);
50   this->DAC.setVoltage(0, false);
51 }
52
53 // loop() function
54 // --------------------
55 bool PRDC_VfdEncEN600::read() {
56 #ifdef DEBUG_VFD_ENC_EN600
57   if(DEBUG_LEVEL > 1) {
58     SerialUSB.println("@VfdEncEN600: loop()");
59   }
60 #endif
61
62   this->getState();
63
64   bool got_message = false;
65
66   while(this->VFD.availableData() > 0) {
67     bool flagIn = VFD.readData(&this->FP_DA, this->FP_DR, &this->FP_FI, this->
         FP_CI, &this->FP_RD);
68     if(flagIn) {
69       this->t_new_message = millis();
70       this->handleMessage();
```

```
71
72        this->n_no_message = 0;
73        got_message = true;
74      }
75    }
76
77    if(!got_message) {
78      this->n_no_message = this->n_no_message+1;
79    }
80
81  #ifndef WITHOUT_ERROR_CHECKING
82    if(this->state == NORMAL_STATE) {
83      // Check for VFD communication timeout
84      if((millis()-this->t_new_message) > VFD_READ_TIMEOUT_TIME && this->
           n_no_message > 1) {
85        this->state = VFD_READ_TIMEOUT;
86      }
87
88      // Check for VFD valid communication timeout
89      if((millis()-this->t_new_valid) > VFD_VALID_READ_TIMEOUT_TIME && this->
           n_no_message > 1) {
90        this->state = VFD_VALID_READ_TIMEOUT;
91      }
92    }
93  #endif
94
95    return got_message;
96  }
97
98  // resetTimeout() function
99  // Reset timeout
100 // --------------------
101 void PRDC_VfdEncEN600::resetTimeout() {
102 #ifdef DEBUG_VFD_ENC_EN600
103   if(DEBUG_LEVEL > 0) {
104     SerialUSB.println("@VfdEncEN600: resetTimeout()");
105   }
106 #endif
107
108   this->t_new_message = millis();
109   this->t_new_valid = millis();
110 }
111
112 // flushInput() function
113 // Flush input
114 // --------------------
115 void PRDC_VfdEncEN600::flushInput() {
116 #ifdef DEBUG_VFD_ENC_EN600
117   if(DEBUG_LEVEL > 0) {
118     SerialUSB.println("@VfdEncEN600: flushInput()");
119   }
120 #endif
121
122   this->VFD.flushInput();
123 }
124
125 // handleMessage() function
126 // Handle input message
127 // --------------------
```

```
128 void PRDC_VfdEncEN600::handleMessage() {
129 #ifdef DEBUG_VFD_ENC_EN600
130   if(DEBUG_LEVEL > 1) {
131     SerialUSB.println("@VfdEncEN600: handleMessage()");
132   }
133 #endif
134
135   this->N_in = this->N_in+1;
136
137 #ifndef WITHOUT_ERROR_CHECKING
138   if(this->FP_DR[0] != '1' || this->FP_DR[1] == '0') {
139     if(this->state == NORMAL_STATE) {
140       // VDF reports error
141       this->state = VFD_ERROR+this->FP_FI;
142     }
143   } else {
144     this->t_new_valid = millis();
145   }
146 #endif
147
148 #ifdef DEBUG_VFD_ENC_EN600
149   if(DEBUG_LEVEL > 0) {
150     SerialUSB.print(" Address: ");
151     SerialUSB.println(this->FP_DA);
152     SerialUSB.print(" Response: ");
153     SerialUSB.println(this->FP_DR);
154     SerialUSB.print(" Failure index: ");
155     SerialUSB.println(this->FP_FI);
156     SerialUSB.print(" Command index: ");
157     SerialUSB.println(this->FP_CI);
158     SerialUSB.print(" Run data: ");
159     SerialUSB.println(this->FP_RD);
160     SerialUSB.println();
161   }
162 #endif
163 }
164
165 // getState() function
166 // Check if VFD is functional
167 // --------------------
168 void PRDC_VfdEncEN600::getState() {
169 #ifdef DEBUG_VFD_ENC_EN600
170   if(DEBUG_LEVEL > 1) {
171     SerialUSB.println("@VfdEncEN600: getState()");
172   }
173 #endif
174
175   static uint32_t t_send = millis();
176
177   // Send packet to VFD
178   if((millis()-t_send) > VFD_STATE_TIMEOUT_TIME) {
179     t_send = millis();
180
181     this->N_out = this->N_out+1;
182     this->VFD.sendData(1, 10, 0, 0); // Look up auxiliary motor state
183
184 #ifdef DEBUG_VFD_ENC_EN600
185     if(DEBUG_LEVEL > 0) {
186       SerialUSB.println(" 010A0000");
```

```
187     }
188 #endif
189   }
190 }
191
192 // wait() function
193 // Wait for VFD boot up
194 // --------------------
195 void PRDC_VfdEncEN600::wait() {
196 #ifdef DEBUG_VFD_ENC_EN600
197   if(DEBUG_LEVEL > 0) {
198     SerialUSB.println("@VfdEncEN600: wait()");
199   }
200 #endif
201
202   uint32_t t_wait = millis();
203
204   while(true) {
205     if(this->read()) {
206       delay(VFD_DELAY_TIME);
207       break;
208     }
209
210     // Check for VFD wait timeout
211     if((millis()-t_wait) > VFD_WAIT_TIMEOUT_TIME) {
212       this->state = VFD_WAIT_TIMEOUT;
213       return;
214     }
215   }
216   this->state = NORMAL_STATE;
217 }
218
219 // setControl() function
220 // Set control frequency
221 // ---------------------
222 void PRDC_VfdEncEN600::setControl(double f_out_in) {
223   this->f_out = f_out_in;
224
225 #ifndef PREVENT_DAC_OUTPUT
226   // Set DAC output
227   this->DAC.setVoltage(abs(this->f_out)/VFD_F_MAX*DAC_MAX, false);
228 #endif
229
230   // Change direction pin state
231   if(abs(this->f_out) > VFD_F_MIN) {
232     if(this->f_out > 0) {
233       // FWD
234       this->FWD_STATE_new = HIGH;
235       this->REV_STATE_new = LOW;
236     } else {
237       // REV
238       this->FWD_STATE_new = HIGH;
239       this->REV_STATE_new = HIGH;
240     }
241   } else {
242     // Stop
243     this->FWD_STATE_new = LOW;
244     this->FWD_STATE_new = LOW;
245   }
```

```
246   this->changeDirPinState();
247 }
248
249 // changeDirPinState() function
250 // Change FWD and REV pins states
251 // --------------------
252 void PRDC_VfdEncEN600::changeDirPinState() {
253 #ifdef DEBUG_VFD_ENC_EN600
254   if(DEBUG_LEVEL > 1) {
255     SerialUSB.println("@VfdEncEN600: changePinState()");
256   }
257 #endif
258
259   if(this->FWD_STATE_new != this->FWD_STATE) {
260     this->FWD_STATE = this->FWD_STATE_new;
261     digitalWrite(this->PIN_FWD, this->FWD_STATE);
262   }
263   if(this->REV_STATE_new != this->REV_STATE) {
264     this->REV_STATE = this->REV_STATE_new;
265     digitalWrite(this->PIN_REV, this->REV_STATE);
266   }
267 }
```

Listing X.33 - PRDC_VFD_ENC_EN600.cpp

X.3 Angle Sensor

Program PRDC_AngleSensor za elektroniku Angle Sensor je dat u listingu X.34, dok su u listinzima X.35 i X.36 dati fajlovi biblioteke PRDC_TLE5012B.

```
1  // PRDC_AngleSensor program
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6  // Board: "PRDC STM32 boards"
7  // Board part number: "angleSense"
8  // U(S)ART support: "Enabled (generic 'Serial')"
9  // USB Support (if available): "None"
10 // Optimize: "Smallest (-Os)"
11 // Debug symbols and core logs: "None"
12 // C Runtime Library: "Newlib Nano (default)"
13 // Upload method: "STM32CubeProgrammer (SWD)"
14 // --------------------
15
16 // Libraries
17 // --------------------
18 // PRDC RS485 STM32 Half Duplex
19 // Author: PR-DC
20 #include <PRDC_RS485HD_STM32.h>
21
22 // PRDC SCBP
23 // Author: PRDC
24 #include <PRDC_SCBP.h>
25
26 // PRDC_TLE5012B
27 // Author: PR-DC
28 #include <PRDC_TLE5012B.h>
```

```
29
30 // Define variables
31 // --------------------
32 #define LED_TIME 100 // [ms] time difference for LED state change
33
34 // Communication
35 #define SCP_BAUDRATE 576000
36 PRDC_RS485HD_STM32 SCP_HS(RS485_RX_PIN, RS485_TX_PIN);
37 PRDC_SCBP SCP_COM;
38
39 struct __attribute__((packed)) packetIn_t {
40   uint8_t enable;
41 };
42 union dataIn_t{
43   packetIn_t p;
44   uint8_t bytes[sizeof(packetIn_t)];
45 } dataIn;
46
47 struct __attribute__((packed)) packetOut_t {
48   float angle;
49   float temp;
50   uint32_t N;
51 };
52 union dataOut_t{
53   packetOut_t p;
54   uint8_t bytes[sizeof(packetOut_t)];
55 } dataOut;
56
57 // Angle sensor
58 PRDC_TLE5012B AngleSensor = PRDC_TLE5012B();
59 errorType checkError = NO_ERROR;
60
61 // Control variables
62 bool SEND_ANGLE = false;
63 uint32_t N = 0;
64
65 // setup function
66 // --------------------
67 void setup() {
68   // Define pin modes
69   pinMode(LED_SIGNAL_PIN, OUTPUT);
70
71   // Communication setup
72   SCP_HS.setPins(RS485_DE_PIN, RS485_nRE_PIN);
73   SCP_COM.begin(SCP_HS, SCP_BAUDRATE);
74
75   // Angle sensor setup
76   checkError = AngleSensor.begin();
77 }
78
79 // loop function
80 // --------------------
81 void loop() {
82   // Send angle
83   sendAngle();
84
85   // Blink LED
86   blinkLED();
87 }
```

```
88
89  // sendAngle() function
90  // Send angle
91  // --------------------
92  void sendAngle(){
93    double angle;
94    double T;
95
96    N = N+1;
97    checkError = AngleSensor.getAngleValue(angle);
98    checkError = AngleSensor.getTemperature(T);
99    dataOut.p.angle = (float)angle;
100   dataOut.p.temp = (float)T;
101   dataOut.p.N = N;
102   SCP_COM.sendPacket(dataOut.bytes, sizeof(dataOut));
103 }
104
105 // blinkLED() function
106 // Blink LED at right time
107 // --------------------
108 void blinkLED(){
109   static uint32_t t_led = millis();
110   static bool s = 0;
111
112   if((millis()-t_led) > LED_TIME) {
113     t_led = millis();
114     s = !s;
115     digitalWrite(LED_SIGNAL_PIN, s);
116   }
117 }
```

Listing X.34 - PRDC_AngleSensor.ino

```
1  // TLE5012B Angle Sensor library
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef _PRDC_TLE5012B_H_
8  #define _PRDC_TLE5012B_H_
9
10 #include "SPI.h"
11 #include "Arduino.h"
12
13 // TLE5012B Library debug
14 //#define DEBUG_TLE5012B
15
16 // SPI communication settings
17 #define TLE5012B_DEFAULT_SPI_FREQUENCY  1000000U
18 #define TLE5012B_DEFAULT_SPI            SPI
19 #define TLE5012B_DEFAULT_CS             PIN_SPI_SS
20 #define TLE5012B_DEFAULT_MISO           PIN_SPI_MISO
21 #define TLE5012B_DEFAULT_MOSI           PIN_SPI_MOSI
22 #define TLE5012B_DEFAULT_SCK            PIN_SPI_SCK
23
24 // Error masks for safety words
25 #define SYSTEM_ERROR_MASK                       0x4000
26 #define INTERFACE_ERROR_MASK            0x2000
27 #define INV_ANGLE_ERROR_MASK            0x1000
```

```
28
29 // Commands for read
30 #define READ_STA_CMD_NOSAFETY    0x8000
31 #define READ_STA_CMD                                    0x8001
32 #define READ_ACTIV_STA_CMD              0x8011
33 #define READ_ANGLE_VAL_CMD              0x8021
34 #define READ_ANGLE_SPD_CMD              0x8031
35 #define READ_ANGLE_REV_CMD              0x8041
36 #define READ_TEMP_CMD                                   0x8051
37 #define READ_INTMODE_1                          0x8061
38 #define READ_SIL                                                0x8071
39 #define READ_INTMODE_2                          0x8081
40 #define READ_INTMODE_3                  0x8091
41 #define READ_OFFSET_X                                   0x80A1
42 #define READ_OFFSET_Y                                   0x80B1
43 #define READ_SYNCH                                              0x80C1
44 #define READ_IFAB                                               0x80D1
45 #define READ_INTMODE_4                          0x80E1
46 #define READ_TEMP_COEFF                         0x80F1
47 #define READ_RAW_X_CMD                          0x8101
48 #define READ_RAW_Y_CMD                          0x8111
49
50 // Commands for updated read
51 #define READ_UPD_STA_CMD                        0x8401
52 #define READ_UPD_ANGLE_VAL_CMD          0x8421
53 #define READ_UPD_ANGLE_SPD_CMD          0x8431
54 #define READ_UPD_ANGLE_REV_CMD          0x8441
55
56 #define READ_BLOCK_CRC                          0x8088
57
58 // Commands for write
59 #define WRITE_STA                                 0x5001
60 #define WRITE_ACTIV_STA                         0x0011
61 #define WRITE_INTMODE_1                         0x5061
62 #define WRITE_SIL                                       0x5071
63 #define WRITE_INTMODE_2                         0x5081
64 #define WRITE_INTMODE_3                         0x5091
65 #define WRITE_OFFSET_X                          0x50A1
66 #define WRITE_OFFSET_Y                          0x50B1
67 #define WRITE_SYNCH                                     0x50C1
68 #define WRITE_IFAB                                      0x50D1
69 #define WRITE_INTMODE_4                         0x50E1
70 #define WRITE_TEMP_COEFF                        0x50F1
71
72 // Mask to check if the command want the value in the register or the value in
       the update buffer
73 #define CHECK_CMD_UPDATE                        0x0400
74
75 // Values used for calculating the CRC
76 #define CRC_POLYNOMIAL                          0x1D
77 #define CRC_SEED                                        0xFF
78 #define CRC_NUM_REGISTERS                       8
79
80 // Values used to calculate 15 bit signed int sent by the sensor
81 #define DELETE_BIT_15                           0x7FFF
82 #define CHANGE_UINT_TO_INT_15           32768
83 #define CHECK_BIT_14                            0x4000
84
85 // Values used to calculate 9 bit signed int sent by the sensor
```

```
86 #define DELETE_7BITS                                    0x01FF
87 #define CHANGE_UNIT_TO_INT_9            512
88 #define CHECK_BIT_9                                          0x0100
89
90 // Values used to for final calculations of angle speed, revolutions, range and
       value
91 #define POW_2_15                                             32768.0
92 #define POW_2_7                                              128.0
93 #define ANGLE_360_VAL                              360.0
94
95 // Values used to calculate the temperature
96 #define TEMP_OFFSET                                       152.0
97 #define TEMP_DIV                                           2.776
98
99 #define GET_BIT_14_4                               0x7FF0
100
101 // Trigger delay time
102 #define TRIGGER_DELAY     5
103
104 // Write read delay
105 #define WRITE_READ_DELAY        1
106
107 enum slaveNum {
108   TLE5012B_S0 = 0x0000,
109   TLE5012B_S1 = 0x2000,
110   TLE5012B_S2 = 0x4000,
111   TLE5012B_S3 = 0x6000
112 };
113
114 // Error types from safety word
115 enum errorType {
116         NO_ERROR = 0x00,
117         SYSTEM_ERROR = 0x01,
118         INTERFACE_ACCESS_ERROR = 0x02,
119         INVALID_ANGLE_ERROR = 0x03,
120         CRC_ERROR = 0xFF
121 };
122
123 class PRDC_TLE5012B {
124   public:
125     //standard constructor
126     PRDC_TLE5012B();
127     PRDC_TLE5012B(SPIClass &conf, uint16_t miso,
128       uint16_t mosi, uint16_t sck);
129     ~PRDC_TLE5012B();
130
131     errorType begin();
132     errorType begin(uint32_t cs);
133     errorType begin(SPIClass &conf, uint16_t miso, uint16_t mosi,
134       uint16_t sck, uint32_t cs);
135     errorType begin(SPIClass &conf, uint16_t miso, uint16_t mosi,
136       uint16_t sck, uint32_t cs, SPISettings &settings);
137     void end();
138
139     errorType resetFirmware();
140
141     errorType readFromSensor(uint16_t command, uint16_t &data);
142     errorType readBlockCRC();
143     errorType readMoreRegisters(uint16_t command, uint16_t data[]);
```

```
144
145     errorType readStatus(uint16_t &data);
146     errorType readAngleValue(int16_t &data);
147     errorType readAngleSpeed(int16_t &data);
148     errorType readAngleRevolution(int16_t &data);
149     errorType readUpdStatus(uint16_t &data);
150     errorType readUpdAngleValue(int16_t &data);
151     errorType readUpdAngleSpeed(int16_t &data);
152     errorType readUpdAngleRevolution(int16_t &data);
153     errorType readTemp(int16_t &data);
154     errorType readRawX(int16_t &data);
155     errorType readRawY(int16_t &data);
156     errorType readActivationStatus(uint16_t &data);
157     errorType readOffsetX(uint16_t &data);
158     errorType readOffsetY(uint16_t &data);
159     errorType readIntMode1(uint16_t &data);
160     errorType readSIL(uint16_t &data);
161     errorType readIntMode2(uint16_t &data);
162     errorType readIntMode3(uint16_t &data);
163     errorType readSynch(uint16_t &data);
164     errorType readIFAB(uint16_t &data);
165     errorType readIntMode4(uint16_t &data);
166     errorType readTempCoeff(uint16_t &data);
167
168     errorType writeToSensor(uint16_t command,
169       uint16_t dataToWrite, bool changeCRC);
170
171     errorType writeStatus(uint16_t dataToWrite);
172     errorType writeSlaveNumber(uint16_t dataToWrite);
173     errorType writeActivationStatus(uint16_t dataToWrite);
174     errorType writeIntMode1(uint16_t dataToWrite);
175     errorType writeSIL(uint16_t dataToWrite);
176     errorType writeIntMode2(uint16_t dataToWrite);
177     errorType writeIntMode3(uint16_t dataToWrite);
178     errorType writeOffsetX(uint16_t dataToWrite);
179     errorType writeOffsetY(uint16_t dataToWrite);
180     errorType writeSynch(uint16_t dataToWrite);
181     errorType writeIFAB(uint16_t dataToWrite);
182     errorType writeIntMode4(uint16_t dataToWrite);
183     errorType writeTempCoeff(uint16_t dataToWrite);
184     errorType writeUpdateCRC(uint16_t dataToWrite);
185
186     errorType getAngleSpeed(double &angleSpeed);
187     errorType getAngleValue(double &angleValue);
188     errorType getNumRevolutions(int16_t &numRev);
189     errorType getUpdAngleSpeed(double &angleSpeed);
190     errorType getUpdAngleValue(double &angleVal);
191     errorType getUpdNumRevolutions(int16_t &numRev);
192     errorType getTemperature(double &temp);
193     errorType getAngleRange(double &angleRange);
194
195     void triggerUpdate();
196
197   private:
198     SPIClass* _spi;
199     SPISettings _spiSettings;
200     uint16_t _CS;
201     uint16_t _MOSI;
202     uint16_t _MISO;
```

```
203     uint16_t _SCK;
204 
205     void beginTransaction();
206     void endTransaction();
207 
208     uint16_t _registers[CRC_NUM_REGISTERS];
209 
210     errorType regularCrcUpdate();
211     errorType checkSafety(uint16_t safety, uint16_t command,
212       uint16_t* readreg, uint16_t length);
213     void resetSafety();
214 };
215 
216 #endif // _PRDC_TLE5012B_H_
```

Listing X.35 - PRDC_TLE5012B.h

```
1  // TLE5012B Angle Sensor library
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6  
7  #include "PRDC_TLE5012B.h"
8  
9  #ifdef DEBUG_TLE5012B
10   char* uint16tToBinaryString(uint16_t decimalnum) {
11     char *bufferOut = (char *) malloc(17);
12     uint8_t i = 0;
13     uint8_t rem = 0;
14 
15     bufferOut[16] = '\0';
16     memset(bufferOut, '0', 16);
17     while(decimalnum != 0) {
18       rem = decimalnum%2;
19       decimalnum = decimalnum / 2;
20       if(rem) {
21         bufferOut[15-i] = '1';
22       } else {
23         bufferOut[15-i] = '0';
24       }
25       i++;
26     }
27     return bufferOut;
28   }
29 #endif
30 
31 // PRDC_TLE5012B()
32 // Object constructor
33 // --------------------
34 PRDC_TLE5012B::PRDC_TLE5012B() {
35         _spi = &TLE5012B_DEFAULT_SPI;
36         _spiSettings = SPISettings(TLE5012B_DEFAULT_SPI_FREQUENCY,
37     MSBFIRST, SPI_MODE1);
38         _CS = TLE5012B_DEFAULT_CS;
39         _MOSI = TLE5012B_DEFAULT_MOSI;
40         _MISO = TLE5012B_DEFAULT_MISO;
41         _SCK = TLE5012B_DEFAULT_SCK;
42 }
43 
```

```
44 // ~PRDC_TLE5012B()
45 // Object destructor
46 // --------------------
47 PRDC_TLE5012B::~PRDC_TLE5012B() {
48         _spi->end();
49 }
50
51 // beginTransaction() function
52 // Begin SPI transaction
53 // --------------------
54 inline void PRDC_TLE5012B::beginTransaction() {
55   _spi->beginTransaction(_spiSettings);
56   digitalWrite(_CS, LOW);
57 }
58
59 // endTransaction() function
60 // End SPI transaction
61 // --------------------
62 inline void PRDC_TLE5012B::endTransaction() {
63   digitalWrite(_CS, HIGH);
64   _spi->endTransaction();
65 }
66
67 // begin() function
68 // Begin communication
69 // --------------------
70 errorType PRDC_TLE5012B::begin() {
71         return begin(SPI, _MISO, _MOSI,
72     _SCK, _CS, _spiSettings);
73 }
74
75 errorType PRDC_TLE5012B::begin(uint32_t cs) {
76         return begin(SPI, _MISO, _MOSI, _SCK, cs, _spiSettings);
77 }
78
79 errorType PRDC_TLE5012B::begin(SPIClass &spi, uint16_t miso,
80     uint16_t mosi, uint16_t sck, uint32_t cs) {
81         return begin(spi, miso, mosi, sck, cs, _spiSettings);
82 }
83
84 errorType PRDC_TLE5012B::begin(SPIClass &spi, uint16_t miso,
85     uint16_t mosi, uint16_t sck, uint32_t cs, SPISettings &settings) {
86   #ifdef DEBUG_TLE5012B
87     Serial.println("begin()");
88   #endif
89
90         _MOSI = mosi;
91         _MISO = miso;
92         _SCK = sck;
93         _spi = &spi;
94
95         _CS = cs;
96         _spiSettings = settings;
97
98   // Select device
99         pinMode(_CS, OUTPUT);
100        digitalWrite(_CS, HIGH);
101
102  _spi->begin();
```

```
103
104   return readBlockCRC();
105 }
106
107 // resetFirmware() function
108 // Resets firmware
109 // --------------------
110 errorType PRDC_TLE5012B::resetFirmware() {
111   #ifdef DEBUG_TLE5012B
112     Serial.println("resetFirmware()");
113   #endif
114
115   return writeActivationStatus(0x401);
116 }
117
118 // getFirstByte() function
119 // Gets the first byte of a 2 byte word
120 // --------------------
121 uint8_t getFirstByte(uint16_t twoByteWord) {
122         return (uint8_t) (twoByteWord >> 8);
123 }
124
125 // getSecondByte() function
126 // Gets the second byte of the 2 byte word
127 // --------------------
128 uint8_t getSecondByte(uint16_t twoByteWord) {
129         return (uint8_t) twoByteWord;
130 }
131
132 // crcCalc() function
133 // Function for calculation the CRC
134 // --------------------
135 uint8_t crcCalc(uint8_t *data ,uint8_t length) {
136   uint32_t crc;
137   int16_t i,bit;
138
139   crc = CRC_SEED;
140   for(i = 0; i<length; i++) {
141     crc ^= data[i];
142     for(bit = 0; bit < 8; bit++) {
143       if((crc & 0x80) != 0) {
144         crc <<= 1;
145         crc ^= CRC_POLYNOMIAL;
146       } else {
147         crc <<= 1;
148       }
149     }
150   }
151   return (~crc) & CRC_SEED;
152 }
153
154 // triggerUpdate() function
155 // Triggers an update
156 // --------------------
157 void PRDC_TLE5012B::triggerUpdate() {
158   #ifdef DEBUG_TLE5012B
159     Serial.println("triggerUpdate()");
160   #endif
161
```

```
162         digitalWrite(_SCK, LOW);
163         digitalWrite(_MOSI,  HIGH);
164         digitalWrite(_CS, LOW);
165   delayMicroseconds(TRIGGER_DELAY);
166         digitalWrite(_CS, HIGH);
167 }
168
169 // checkSafety() function
170 // Check safery word
171 // After every transaction with the TLE5012B, a safety word is
172 // returned to check the validity of the value received.
173 // This is the structure of safety word, in which the numbers
174 // represent the bit position in 2 bytes.
175 // 15 - indication of chip reset or watchdog overflow:
176 //      0 reset occured, 1 no reset
177 // 14 - 0 System error, 1 No error
178 // 13 - 0 Infterface access error, 1 No error
179 // 12 - 0 Invalid angle value, 1 valid angle value
180 // 11:8 - Sensor number response indicator,
181 //        the sensor number bit is pulled low and other bits are high
182 // 7:0 - CRC
183 //
184 // A CRC needs to be calculated using all the data sent and received
185 // (i.e. the command and the value return from the register, which is 4 bytes),
186 // and needs to be checked with the CRC sent in the safety word.
187 // --------------------
188 errorType PRDC_TLE5012B::checkSafety(uint16_t safety, uint16_t command,
189     uint16_t* readreg, uint16_t length) {
190         errorType errorCheck ;
191   #ifdef DEBUG_TLE5012B
192     Serial.println("checkSafety()");
193   #endif
194
195         if(!((safety) & SYSTEM_ERROR_MASK))     {
196                 errorCheck = SYSTEM_ERROR;
197                 resetSafety();
198         }       else if(!((safety) & INTERFACE_ERROR_MASK))     {
199                 errorCheck = INTERFACE_ACCESS_ERROR;
200         }       else if(!((safety) & INV_ANGLE_ERROR_MASK)) {
201                 errorCheck = INVALID_ANGLE_ERROR;
202         }       else {
203                 uint16_t lengthOfTemp = length*2 + 2;
204                 uint8_t temp[lengthOfTemp];
205
206                 temp[0] = getFirstByte(command);
207                 temp[1] = getSecondByte(command);
208
209                 for(uint16_t i = 0; i<length; i++)      {
210                         temp[2 + 2*i] = getFirstByte(readreg[i]);
211                         temp[2 + 2*i+1] = getSecondByte(readreg[i]);
212                 }
213
214                 uint8_t crcReceivedFinal = getSecondByte(safety);
215
216                 uint8_t crc = crcCalc(temp, lengthOfTemp);
217                 if(crc == crcReceivedFinal) {
218                         errorCheck = NO_ERROR;
219                 }       else {
220                         errorCheck = CRC_ERROR;
```

```
221                         resetSafety();
222                 }
223         }
224
225   #ifdef DEBUG_TLE5012B
226     Serial.print("errorCheck = ");
227     Serial.println(errorCheck);
228   #endif
229         return errorCheck;
230 }
231
232 // resetSafety() function
233 // when an error occurs in the safety word, the error bit remains
234 // 0(error), until the status register is read again.
235 // Flushes out safety errors, that might have occured by reading the
236 // register without a safety word.
237 // --------------------
238 void PRDC_TLE5012B::resetSafety() {
239   #ifdef DEBUG_TLE5012B
240     Serial.println("resetSafety()");
241   #endif
242
243         triggerUpdate();
244   beginTransaction();
245         _spi->transfer16(READ_STA_CMD);
246         _spi->transfer16(0xFFFF);
247         _spi->transfer16(0xFFFF);
248   endTransaction();
249 }
250
251 // readFromSensor() function
252 // General read function for reading registers from the TLE5012B.
253 // Structure of command word, the numbers represent the bit position
254 // of the 2 byte command
255 // 15 - 0 write, 1 read
256 // 14:11 -  0000 for default operational access for addresses
257 // between 0x00 - 0x04,
258 // 1010 for configuration access for addresses between 0x05 - 0x11
259 // 10 - 0 access to current value, 1 access to value in update buffer
260 // 9:4 - access to 6 bit register address
261 // 3:0 - 4 bit number of data words.
262 // --------------------
263 errorType PRDC_TLE5012B::readFromSensor(uint16_t command, uint16_t &data) {
264   #ifdef DEBUG_TLE5012B
265     Serial.println("readFromSensor()");
266     Serial.print("command = 0x");
267     Serial.println(command, HEX);
268   #endif
269
270         if(command & CHECK_CMD_UPDATE)  {
271                 triggerUpdate();
272         }
273         beginTransaction();
274         _spi->transfer16(command);
275   delayMicroseconds(WRITE_READ_DELAY);
276         uint16_t regVal = _spi->transfer16(0xFFFF);
277         uint16_t safety = _spi->transfer16(0xFFFF);
278         endTransaction();
279   #ifdef DEBUG_TLE5012B
```

```
280     Serial.print("regVal = ");
281     char* str = uint16tToBinaryString(regVal);
282     Serial.println(str);
283     free(str);
284   #endif
285 
286         errorType checkError = checkSafety(safety, command, &regVal, 1);
287         if(checkError != NO_ERROR)      {
288                 data = 0;
289                 return checkError;
290         }       else {
291                 data = regVal;
292                 return NO_ERROR;
293         }
294 }
295 
296 // readBlockCRC() function
297 // Reads the block of registers from addresses 08 - 0F in order
298 // to figure out the CRC.
299 // --------------------
300 errorType PRDC_TLE5012B::readBlockCRC() {
301         uint16_t safety = 0;
302 
303         beginTransaction();
304         _spi->transfer16(READ_BLOCK_CRC);
305   delayMicroseconds(WRITE_READ_DELAY);
306         for(uint8_t i=0; i < CRC_NUM_REGISTERS; i++)    {
307                 _registers[i] = _spi->transfer16(0xFFFF);
308         }
309         safety = _spi->transfer16(0xFFFF);
310         endTransaction();
311 
312         errorType checkError = checkSafety(safety, READ_BLOCK_CRC,
313       _registers, CRC_NUM_REGISTERS);
314         return checkError;
315 }
316 
317 
318 // readMoreRegisters() function
319 // Used to read 1 or more than 1 consecutive registers
320 // --------------------
321 errorType PRDC_TLE5012B::readMoreRegisters(uint16_t command,
322     uint16_t data[]) {
323         uint16_t lengthOfResponse = command & (0x000F);
324         uint16_t safety = 0;
325 
326         beginTransaction();
327         _spi->transfer16(command);
328   delayMicroseconds(WRITE_READ_DELAY);
329         for(uint8_t i=0; i < lengthOfResponse; i++) {
330                 data[i] = _spi->transfer16(0xFFFF);
331         }
332         safety = _spi->transfer16(0xFFFF);
333         endTransaction();
334 
335         errorType checkError = checkSafety(safety, command, data,
            lengthOfResponse);
336         return checkError;
337 }
```

```
338
339 // readStatus() function
340 // Read status register
341 // --------------------
342 errorType PRDC_TLE5012B::readStatus(uint16_t &data) {
343   return readFromSensor(READ_STA_CMD, data);
344 }
345
346 // readUpdStatus() function
347 // Trigger update than read status register
348 // --------------------
349 errorType PRDC_TLE5012B::readUpdStatus(uint16_t &data) {
350         return readFromSensor(READ_UPD_STA_CMD, data);
351 }
352
353 // readActivationStatus() function
354 // Read activation status register
355 // --------------------
356 errorType PRDC_TLE5012B::readActivationStatus(uint16_t &data) {
357         return readFromSensor(READ_ACTIV_STA_CMD, data);
358 }
359
360 // readAngleValue() function
361 // Read angle value
362 // --------------------
363 errorType PRDC_TLE5012B::readAngleValue(int16_t &data) {
364         uint16_t rawData = 0;
365         errorType status = readFromSensor(READ_ANGLE_VAL_CMD, rawData);
366         if(status != NO_ERROR)  {
367                 return status;
368         }
369
370         rawData = (rawData & (DELETE_BIT_15));
371         if(rawData & CHECK_BIT_14) {
372                 rawData = rawData - CHANGE_UINT_TO_INT_15; // Negative data
373         }
374         data = rawData;
375         return NO_ERROR;
376 }
377
378 // readAngleSpeed() function
379 // Read angle speed value
380 // --------------------
381 errorType PRDC_TLE5012B::readAngleSpeed(int16_t &data) {
382         uint16_t rawData = 0;
383         errorType status =  readFromSensor(READ_ANGLE_SPD_CMD, rawData);
384         if(status != NO_ERROR)  {
385                 return status;
386         }
387
388         rawData = (rawData & (DELETE_BIT_15));
389         if(rawData & CHECK_BIT_14)      {
390                 rawData = rawData - CHANGE_UINT_TO_INT_15; // Negative data
391         }
392         data = rawData;
393         return NO_ERROR;
394 }
395
396 // readAngleRevolution() function
```

```
397 // Read angle revolution value
398 // --------------------
399 errorType PRDC_TLE5012B::readAngleRevolution(int16_t &data) {
400         uint16_t rawData = 0;
401         errorType status =      readFromSensor(READ_ANGLE_REV_CMD, rawData);
402         if(status != NO_ERROR)  {
403                 return status;
404         }
405
406         rawData = (rawData & (DELETE_7BITS));
407         if(rawData & CHECK_BIT_9) {
408                 rawData = rawData - CHANGE_UNIT_TO_INT_9; // Negative data
409         }
410         data = rawData;
411         return NO_ERROR;
412 }
413
414 // readTemp() function
415 // Read temperature value
416 // --------------------
417 errorType PRDC_TLE5012B::readTemp(int16_t &data) {
418         uint16_t rawData = 0;
419         errorType status =      readFromSensor(READ_TEMP_CMD, rawData);
420         if(status != NO_ERROR)  {
421                 data = 0;
422                 return status;
423         }
424
425         rawData = (rawData & (DELETE_7BITS));
426         if(rawData & CHECK_BIT_9) {
427                 rawData = rawData - CHANGE_UNIT_TO_INT_9; // Negative data
428         }
429         data = rawData;
430         return NO_ERROR;
431 }
432
433 // readRawX() function
434 // Read rawX value
435 // --------------------
436 errorType PRDC_TLE5012B::readRawX(int16_t &data) {
437         uint16_t rawData = 0;
438         errorType status =      readFromSensor(READ_RAW_X_CMD, rawData);
439         if(status != NO_ERROR)  {
440                 data = 0;
441                 return status;
442         }
443
444         data = rawData;
445         return NO_ERROR;
446 }
447
448 // readRawY() function
449 // Read rawY value
450 // --------------------
451 errorType PRDC_TLE5012B::readRawY(int16_t &data) {
452         uint16_t rawData = 0;
453         errorType status =      readFromSensor(READ_RAW_Y_CMD, rawData);
454         if(status != NO_ERROR)  {
455                 data = 0;
```

```
456                 return status;
457         }
458 
459         data = rawData;
460         return NO_ERROR;
461 }
462 
463 // readUpdAngleValue() function
464 // Trigger update than read angle value
465 // --------------------
466 errorType PRDC_TLE5012B::readUpdAngleValue(int16_t &data) {
467         uint16_t rawData = 0;
468         errorType status = readFromSensor(READ_UPD_ANGLE_VAL_CMD, rawData);
469         if(status != NO_ERROR)  {
470                 data = 0;
471                 return status;
472         }
473 
474         rawData = (rawData & (DELETE_BIT_15));
475         if(rawData & CHECK_BIT_14)      {
476                 rawData = rawData - CHANGE_UINT_TO_INT_15; // Negative data
477         }
478         data = rawData;
479         return NO_ERROR;
480 }
481 
482 // readUpdAngleSpeed() function
483 // Trigger update than read angle speed value
484 // --------------------
485 errorType PRDC_TLE5012B::readUpdAngleSpeed(int16_t &data) {
486         uint16_t rawData = 0;
487         errorType status =  readFromSensor(READ_UPD_ANGLE_SPD_CMD, rawData);
488         if(status != NO_ERROR)  {
489                 data = 0;
490                 return status;
491         }
492 
493         rawData = (rawData & (DELETE_BIT_15));
494         if(rawData & CHECK_BIT_14)      {
495                 rawData = rawData - CHANGE_UINT_TO_INT_15; // Negative data
496         }
497         data = rawData;
498         return NO_ERROR;
499 }
500 
501 // readUpdAngleRevolution() function
502 // Trigger update than read angle revolution value
503 // --------------------
504 errorType PRDC_TLE5012B::readUpdAngleRevolution(int16_t &data) {
505         uint16_t rawData = 0;
506 
507         errorType status = readFromSensor(READ_UPD_ANGLE_REV_CMD, rawData);
508         if(status != NO_ERROR) {
509                 data = 0;
510                 return status;
511         }
512 
513         rawData = (rawData & (DELETE_7BITS));
514         if(rawData & CHECK_BIT_9)       {
```

```
515                     rawData = rawData - CHANGE_UNIT_TO_INT_9; // Negative data
516             }
517             data = rawData;
518             return NO_ERROR;
519     }
520
521     // readIntMode1() function
522     // Read IntMode1 register
523     // --------------------
524     errorType PRDC_TLE5012B::readIntMode1(uint16_t &data) {
525             return readFromSensor(READ_INTMODE_1, data);
526     }
527
528     // readSIL() function
529     // Read SIL register
530     // --------------------
531     errorType PRDC_TLE5012B::readSIL(uint16_t &data) {
532             return readFromSensor(READ_SIL, data);
533     }
534
535     // readIntMode2() function
536     // Read IntMode2 register
537     // --------------------
538     errorType PRDC_TLE5012B::readIntMode2(uint16_t &data) {
539             return readFromSensor(READ_INTMODE_2, data);
540     }
541
542     // readIntMode3() function
543     // Read IntMode3 register
544     // --------------------
545     errorType PRDC_TLE5012B::readIntMode3(uint16_t &data) {
546             return readFromSensor(READ_INTMODE_3, data);
547     }
548
549     // readOffsetX() function
550     // Read OffsetX register
551     // --------------------
552     errorType PRDC_TLE5012B::readOffsetX(uint16_t &data) {
553             return readFromSensor(READ_OFFSET_X, data);
554     }
555
556     // readOffsetY() function
557     // Read OffsetY register
558     // --------------------
559     errorType PRDC_TLE5012B::readOffsetY(uint16_t &data) {
560             return readFromSensor(READ_OFFSET_Y, data);
561     }
562
563     // readSynch() function
564     // Read Synch register
565     // --------------------
566     errorType PRDC_TLE5012B::readSynch(uint16_t &data) {
567             return readFromSensor(READ_SYNCH, data);
568     }
569
570     // readIFAB() function
571     // Read IFAB register
572     // --------------------
573     errorType PRDC_TLE5012B::readIFAB(uint16_t &data) {
```

```
574         return readFromSensor(READ_IFAB, data);
575 }
576
577 // readIntMode4() function
578 // Read IntMode4 register
579 // --------------------
580 errorType PRDC_TLE5012B::readIntMode4(uint16_t &data) {
581         return readFromSensor(READ_INTMODE_4, data);
582 }
583
584 // readTempCoeff() function
585 // Read temperature coef from register
586 // --------------------
587 errorType PRDC_TLE5012B::readTempCoeff(uint16_t &data) {
588         return readFromSensor(READ_TEMP_COEFF, data);
589 }
590
591 // regularCrcUpdate() function
592 // This function is called each time any register in the
593 // range 08 - 0F(first byte) is changed.
594 // It calculates the new CRC based on the value of all the
595 // registers and then stores the value in 0F(second byte)
596 // --------------------
597 errorType PRDC_TLE5012B::regularCrcUpdate() {
598   #ifdef DEBUG_TLE5012B
599     Serial.println("regularCrcUpdate()");
600   #endif
601
602         readBlockCRC();
603         uint8_t temp[16];
604         for(uint8_t i = 0; i < CRC_NUM_REGISTERS; i++) {
605                 temp[2*i] = getFirstByte(_registers[i]);
606                 temp[(2*i)+1] = getSecondByte(_registers[i]);
607         }
608         uint8_t crc = crcCalc(temp, 15);
609         uint16_t firstTempByte = (uint16_t) temp[14];
610         uint16_t secondTempByte = (uint16_t) crc;
611         uint16_t regVal = (firstTempByte << 8) | secondTempByte;
612         _registers[7] = regVal;
613         return writeUpdateCRC(regVal);
614 }
615
616 // writeToSensor() function
617 // General write function for writing registers from the TLE5012B
618 // --------------------
619 errorType PRDC_TLE5012B::writeToSensor(uint16_t command,
620     uint16_t dataToWrite, bool changeCRC) {
621         uint16_t safety = 0;
622   #ifdef DEBUG_TLE5012B
623     Serial.println("writeToSensor()");
624     Serial.print("command = 0x");
625     Serial.println(command);
626     Serial.print("regVal = ");
627     char* str = uint16tToBinaryString(dataToWrite);
628     Serial.println(str);
629     free(str);
630   #endif
631
632         beginTransaction();
```

```
633         _spi->transfer16(command);
634         _spi->transfer16(dataToWrite);
635   delayMicroseconds(WRITE_READ_DELAY);
636         safety = _spi->transfer16(0xFFFF);
637         endTransaction();
638 
639         uint16_t data[1] = {dataToWrite};
640         errorType checkError = checkSafety(safety, command, data, 1);
641 
642         // Update CRC if we write to a register, which changes the CRC.
643         if(changeCRC)   {
644                 checkError = regularCrcUpdate();
645         }
646         return checkError;
647 }
648 
649 // writeStatus() function
650 // Write to status register
651 // --------------------
652 errorType PRDC_TLE5012B::writeStatus(uint16_t dataToWrite) {
653         return writeToSensor(WRITE_STA, dataToWrite, false);
654 }
655 
656 // writeSlaveNumber() function
657 // Write to slave number to status register
658 // --------------------
659 errorType PRDC_TLE5012B::writeSlaveNumber(uint16_t dataToWrite) {
660         return writeToSensor(WRITE_STA, dataToWrite, false);
661 }
662 
663 // writeActivationStatus() function
664 // Write to activation status register
665 // --------------------
666 errorType PRDC_TLE5012B::writeActivationStatus(uint16_t dataToWrite) {
667          return writeToSensor(WRITE_ACTIV_STA, dataToWrite, false);
668 }
669 
670 // writeIntMode1() function
671 // Write to IntMode1 register
672 // --------------------
673 errorType PRDC_TLE5012B::writeIntMode1(uint16_t dataToWrite) {
674         return writeToSensor(WRITE_INTMODE_1, dataToWrite, false);
675 }
676 
677 // writeSIL() function
678 // Write to SIL register
679 // --------------------
680 errorType PRDC_TLE5012B::writeSIL(uint16_t dataToWrite) {
681         return writeToSensor(WRITE_SIL, dataToWrite, false);
682 }
683 
684 // writeIntMode2() function
685 // Write to IntMode2 register
686 // --------------------
687 errorType PRDC_TLE5012B::writeIntMode2(uint16_t dataToWrite) {
688         return writeToSensor(WRITE_INTMODE_2, dataToWrite, true);
689 }
690 
691 // writeIntMode3() function
```

```
692 // Write to IntMode3 register
693 // --------------------
694 errorType PRDC_TLE5012B::writeIntMode3(uint16_t dataToWrite) {
695         return writeToSensor(WRITE_INTMODE_3, dataToWrite, true);
696 }
697
698 // writeOffsetX() function
699 // Write to OffsetX register
700 // --------------------
701 errorType PRDC_TLE5012B::writeOffsetX(uint16_t dataToWrite) {
702         return writeToSensor(WRITE_OFFSET_X, dataToWrite, true);
703 }
704
705 // writeOffsetY() function
706 // Write to OffsetY register
707 // --------------------
708 errorType PRDC_TLE5012B::writeOffsetY(uint16_t dataToWrite) {
709         return writeToSensor(WRITE_OFFSET_Y, dataToWrite, true);
710 }
711
712 // writeSynch() function
713 // Write to Synch register
714 // --------------------
715 errorType PRDC_TLE5012B::writeSynch(uint16_t dataToWrite) {
716         return writeToSensor(WRITE_SYNCH, dataToWrite, true);
717 }
718
719 // writeIFAB() function
720 // Write to IFAB register
721 // --------------------
722 errorType PRDC_TLE5012B::writeIFAB(uint16_t dataToWrite) {
723         return writeToSensor(WRITE_IFAB, dataToWrite, true);
724 }
725
726 // writeIntMode4() function
727 // Write to IntMode4 register
728 // --------------------
729 errorType PRDC_TLE5012B::writeIntMode4(uint16_t dataToWrite) {
730         return writeToSensor(WRITE_INTMODE_4, dataToWrite, true);
731 }
732
733 // writeTempCoeff() function
734 // Write to temperature coef to register
735 // --------------------
736 errorType PRDC_TLE5012B::writeTempCoeff(uint16_t dataToWrite) {
737         return writeToSensor(WRITE_TEMP_COEFF, dataToWrite, true);
738 }
739
740 // writeUpdateCRC() function
741 // Update the CRC in the register 0F(second byte)
742 // --------------------
743 errorType PRDC_TLE5012B::writeUpdateCRC(uint16_t dataToWrite) {
744   #ifdef DEBUG_TLE5012B
745     Serial.println("writeUpdateCRC()");
746   #endif
747
748         uint16_t safety = 0;
749         uint16_t readreg = 0;
750         uint16_t data[1] = {dataToWrite};
```

```
751
752         triggerUpdate();
753
754         beginTransaction();
755         _spi->transfer16(WRITE_TEMP_COEFF);
756         _spi->transfer16(dataToWrite);
757   delayMicroseconds(WRITE_READ_DELAY);
758         safety = _spi->transfer16(0xFFFF);
759         endTransaction();
760
761         errorType checkError = checkSafety(safety, WRITE_TEMP_COEFF, data, 1);
762         checkError = readStatus(readreg);
763         if(readreg & 0x0008) {
764                 checkError = regularCrcUpdate();
765         }
766         return checkError;
767 }
768
769 // getAngleValue() function
770 // Returns the angle value
771 // --------------------
772 errorType PRDC_TLE5012B::getAngleValue(double &angleValue) {
773         int16_t rawAnglevalue = 0;
774         errorType checkError = readAngleValue(rawAnglevalue);
775         if(checkError != NO_ERROR) {
776                 return checkError;
777         }
778
779         angleValue = (ANGLE_360_VAL / POW_2_15) * ((double)rawAnglevalue);
780         return NO_ERROR;
781 }
782
783 // calculateAngleSpeed() function
784 // Calculate the Angle Speed as per the data sheet
785 // --------------------
786 double calculateAngleSpeed(double angRange, int16_t rawAngleSpeed,
787     uint16_t firMD, uint16_t predictionVal) {
788         double finalAngleSpeed;
789         double microsecToSec = 0.000001;
790         double firMDVal;
791
792         if(firMD == 1) {
793                 firMDVal = 42.7;
794         }       else if(firMD == 0)     {
795                 firMDVal = 21.3;
796         }       else if(firMD == 2)     {
797                 firMDVal = 85.3;
798         }       else if(firMD == 3)     {
799                 firMDVal = 170.6;
800         }       else {
801                 firMDVal = 0;
802         }
803
804         finalAngleSpeed = ((angRange / POW_2_15) * ((double)rawAngleSpeed)) /
805     (((double)predictionVal) * firMDVal * microsecToSec);
806         return finalAngleSpeed;
807 }
808
809 // getAngleSpeed() function
```

```
810 // Returns the angle speed
811 // --------------------
812 errorType PRDC_TLE5012B::getAngleSpeed(double &finalAngleSpeed) {
813         int16_t rawAngleSpeed = 0;
814         double angleRange = 0.0;
815         uint16_t firMDVal = 0;
816         uint16_t intMode2Prediction = 0;
817 
818         errorType checkError = readAngleSpeed(rawAngleSpeed);
819         if(checkError != NO_ERROR)      {
820                 return checkError;
821         }
822 
823         checkError = getAngleRange(angleRange);
824         if(checkError != NO_ERROR)      {
825                 return checkError;
826         }
827 
828         checkError = readIntMode1(firMDVal);
829         if(checkError != NO_ERROR)      {
830                 return checkError;
831         }
832         firMDVal >>= 14;
833 
834         checkError = readIntMode2(intMode2Prediction);
835         if(checkError != NO_ERROR)      {
836                 return checkError;
837         }
838 
839         if(intMode2Prediction & 0x0004) {
840                 intMode2Prediction = 3;
841         }       else {
842                 intMode2Prediction = 2;
843         }
844 
845         finalAngleSpeed = calculateAngleSpeed(angleRange, rawAngleSpeed,
846     firMDVal, intMode2Prediction);
847         return NO_ERROR;
848 }
849 
850 // getNumRevolutions() function
851 // Returns the number of revolutions
852 // --------------------
853 errorType PRDC_TLE5012B::getNumRevolutions(int16_t &numRev) {
854         return readAngleRevolution(numRev);
855 }
856 
857 // getUpdAngleValue() function
858 // Triggers update and returns the angle value
859 // --------------------
860 errorType PRDC_TLE5012B::getUpdAngleValue(double &angleValue) {
861         int16_t rawAnglevalue = 0;
862         errorType checkError = readUpdAngleValue(rawAnglevalue);
863         if(checkError != NO_ERROR)      {
864                 return checkError;
865         }
866 
867         angleValue = (ANGLE_360_VAL / POW_2_15) * ((double)rawAnglevalue);
868         return NO_ERROR;
```

```
869 }
870
871 // getUpdAngleSpeed() function
872 // Triggers update and returns the angle speed value
873 // --------------------
874 errorType PRDC_TLE5012B::getUpdAngleSpeed(double &angleSpeed) {
875         int16_t rawAngleSpeed = 0;
876         double angleRange = 0.0;
877         uint16_t firMDVal = 0;
878         uint16_t intMode2Prediction = 0;
879
880         errorType checkError = readUpdAngleSpeed(rawAngleSpeed);
881         if(checkError != NO_ERROR)      {
882                 return checkError;
883         }
884
885         checkError = getAngleRange(angleRange);
886         if(checkError != NO_ERROR)      {
887                 return checkError;
888         }
889
890         checkError = readIntMode1(firMDVal);
891         if(checkError != NO_ERROR)      {
892                 return checkError;
893         }
894
895         checkError = readIntMode2(intMode2Prediction);
896         if(checkError != NO_ERROR) {
897                 return checkError;
898         }
899
900         if(intMode2Prediction & 0x0004) {
901                 intMode2Prediction = 3;
902         }       else {
903                 intMode2Prediction = 2;
904         }
905
906         angleSpeed = calculateAngleSpeed(angleRange, rawAngleSpeed,
907     firMDVal, intMode2Prediction);
908         return NO_ERROR;
909 }
910
911 // getUpdNumRevolutions() function
912 // Triggers update and returns the number of revolutions
913 // --------------------
914 errorType PRDC_TLE5012B::getUpdNumRevolutions(int16_t &numRev) {
915         return readAngleRevolution(numRev);
916 }
917
918 // getAngleRange() function
919 // Returns the angle range
920 // --------------------
921 errorType PRDC_TLE5012B::getAngleRange(double &angleRange) {
922         uint16_t rawData = 0;
923         errorType checkError = readIntMode2(rawData);
924         if(checkError != NO_ERROR)      {
925                 return checkError;
926         }
927
```

```
928         rawData &=  GET_BIT_14_4;
929         rawData >>= 4;
930         angleRange = ANGLE_360_VAL * (POW_2_7 / (double)(rawData));
931         return NO_ERROR;
932 }
933
934 // getTemperature() function
935 // Returns the temperature
936 // --------------------
937 errorType PRDC_TLE5012B::getTemperature(double &temperature) {
938         int16_t rawTemp = 0;
939         errorType checkError = readTemp(rawTemp);
940         if(checkError != NO_ERROR)      {
941                 return checkError;
942         }
943
944         temperature = (rawTemp + TEMP_OFFSET) / (TEMP_DIV);
945         return NO_ERROR;
946 }
```

Listing X.36 - PRDC_TLE5012B.cpp

Y

Skripta flightGearSimulationUDP.jsl

Skriptom `flightGearSimulationUDP.jsl`, koja je u celosti data u listingu Y.1, se program *FlightGear* koristi za formiranje virtuelne letelice.

```
/**
 * @file Flight Gear Simulation over UDP
 * @author Milos Petrasinovic <mpetrasinovic@pr-dc.com>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_M'; // ID mehanizma
var libs_folder = 'Z:/MP/PROGRAMI/_Electron_lib/';
var port = 'COM10';
var baudrate = 115200; // [-] serial communication baudrate
var ep_lim = [[40, 40, 30, 45, 45, 45],
  [-40, -40,-20,-45,-45,-45]];
var ip_lim = [[70, 70, 70, 70, 70, 70],
  [-70, -70, -70, -70, -70, -70]];
var W_l = [420, 500, 400, 500, 390, 500];
var W_h = [1470, 1550, 1450, 1550, 1440, 1550];
var W_u = [2370, 2450, 2350, 2450, 2340, 2450];
var packet_struct_in = [
  ['UInt16LE', 'ID'], ['UInt16LE', 'P', 6]
];
var packet_struct_out = [
  ['UInt16LE', 'ID'], ['UInt16LE', 'P', 6]
];
var k = [0.001, 0.001, 0.001, 1/2, 1/2, 1/2];
var PROTOCOL_MAX = 65534; // [-] max value in protocol
var IP_PROTOCOL_LIM = 180; // [-] max value for internal parameters
var telemetry_hz = 50;
var telemetry_host = '127.0.0.1';
var telemetry_port = 5500;
var fg_path = "C:/Program Files/FlightGear 2020.3/bin/fgfs.exe";
var protocol_type = ['bin', 'asci'][0];
var telemetry_hz_interval = 5; // [s]
// --------------------

// - Motion platform
const { PRDC_PACKET } = require(libs_folder + 'PRDC_PACKET/PRDC_PACKET');
```

```
var P_in = new PRDC_PACKET();
var P_out = new PRDC_PACKET();
P_in.makePacket(packet_struct_in);
P_out.makePacket(packet_struct_out);
var inFun = P_in.getInputCast();
var outFun = P_out.getOutputCast();
var N_in = P_in.bytes;
var N_out = P_out.bytes;

const { PRDC_SCBP } = require(libs_folder + 'PRDC_SCBP/PRDC_SCBP');
var COM = new PRDC_SCBP();

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
solver.getGeometry();

// Open serial port
function serialErrorCallback(e) {
  stop = true;
  error(e);
}
COM.begin(port, baudrate, serialErrorCallback);
addForCleanup(COM, function() {
  if(COM.isOpen()) {
    COM.close();
  }
});

await waitMSeconds(1000);

// Enable motors
var data_out = ones(7);
COM.sendPacket(outFun(data_out), N_out);

// Set lower position pulse width
data_out[0] = 7;
data_out = [data_out[0], ...W_l];
COM.sendPacket(outFun(data_out), N_out);

// Set horizontal position pulse width
data_out[0] = 8;
data_out = [data_out[0], ...W_h];
COM.sendPacket(outFun(data_out), N_out);

// Set upper position pulse width
data_out[0] = 9;
data_out = [data_out[0], ...W_u];
COM.sendPacket(outFun(data_out), N_out);

// Zero motors
data_out = createFilledArray(7, PROTOCOL_MAX/2);
data_out[0] = 11;
COM.sendPacket(outFun(data_out), N_out);
```

```
100
101 // - Flight Gear over UDP
102 var com;
103
104 var pitch;
105 var roll;
106 var yaw_rate;
107 var acc_x;
108 var acc_y;
109 var acc_z;
110
111 var xml_data_asci = '<?xml version="1.0"?>
112
113 <PropertyList>
114
115  <generic>
116
117   <output>
118       <line_separator></line_separator>
119       <var_separator>,</var_separator>
120       <chunk>
121         <name>roll</name>
122         <type>float</type>
123         <format>%.2f</format>
124         <node>/orientation/roll-deg</node>
125       </chunk>
126       <chunk>
127         <name>pitch</name>
128         <type>float</type>
129         <format>%.2f</format>
130         <node>/orientation/pitch-deg</node>
131       </chunk>
132       <chunk>
133         <name>yaw</name>
134         <type>float</type>
135         <format>%.2f</format>
136         <node>/orientation/yaw-rate-degps</node>
137       </chunk>
138       <chunk>
139         <name>x-accel</name>
140         <type>float</type>
141         <format>%.2f</format>
142         <node>/accelerations/pilot/x-accel-fps_sec</node>
143       </chunk>
144       <chunk>
145         <name>y-accel</name>
146         <type>float</type>
147         <format>%.2f</format>
148         <node>/accelerations/pilot/y-accel-fps_sec</node>
149       </chunk>
150       <chunk>
151         <name>z-accel</name>
152         <type>float</type>
153         <format>%.2f</format>
154         <node>/accelerations/pilot/z-accel-fps_sec</node>
155       </chunk>
156   </output>
157
158  </generic>
```

```
159
160 </PropertyList>`;
161
162 var xml_data_bin = `<?xml version="1.0"?>
163
164 <PropertyList>
165
166  <generic>
167
168   <output>
169       <binary_mode>true</binary_mode>
170       <binary_footer>none</binary_footer>
171       <byte_order>host</byte_order>
172       <chunk>
173         <name>roll</name>
174         <type>float</type>
175         <node>/orientation/roll-deg</node>
176       </chunk>
177       <chunk>
178         <name>pitch</name>
179         <type>float</type>
180         <node>/orientation/pitch-deg</node>
181       </chunk>
182       <chunk>
183         <name>yaw</name>
184         <type>float</type>
185         <node>/orientation/yaw-rate-degps</node>
186       </chunk>
187       <chunk>
188         <name>x-accel</name>
189         <type>float</type>
190         <node>/accelerations/pilot/x-accel-fps_sec</node>
191       </chunk>
192       <chunk>
193         <name>y-accel</name>
194         <type>float</type>
195         <node>/accelerations/pilot/y-accel-fps_sec</node>
196       </chunk>
197       <chunk>
198         <name>z-accel</name>
199         <type>float</type>
200         <node>/accelerations/pilot/z-accel-fps_sec</node>
201       </chunk>
202   </output>
203
204  </generic>
205
206 </PropertyList>`;
207 writeFile(pwd + 'resources\\remilp_output_asci.xml', xml_data_asci);
208 writeFile(pwd + 'resources\\remilp_output_bin.xml', xml_data_bin);
209
210 disp('Starting FlightGear...');
211 exec(`"${fg_path}" --generic=socket,out,${telemetry_hz},${telemetry_host},${
    telemetry_port},udp,remilp_output_${protocol_type} --aircraft=c172p --airport
    =KSFO --offset-distance=5 --offset-azimuth=90 --altitude=3000 --heading=270
    --vc=100 --timeofday=noon --enable-terrasync`);
212
213 var N = 0;
214 var com = udpServer(telemetry_port, telemetry_host);
```

```
com.setOnData(function(data) {
  var telemetry_data;
  if(protocol_type == 'bin') {
    telemetry_data = [...(new Float32Array(data.buffer))];
  } else if(protocol_type == 'asci') {
    telemetry_data = data.toString().split(",").map((e) => Number(e));
  }
  if(telemetry_data && telemetry_data.length == 6) {
    pitch = telemetry_data[1];
    roll = telemetry_data[0];
    yaw_rate = telemetry_data[2];
    acc_x = telemetry_data[3];
    acc_y = telemetry_data[4];
    acc_z = telemetry_data[5];
    N += 1;
  }

  var x = [acc_x*k[0], acc_y*k[1], acc_z*k[2],
    pitch*k[3], roll*k[4], yaw_rate*k[5]];
  solver.setParameters(x);
  solver.checkJoints();

  if(!any(MP.e)) {
    let data_out = zeros(7);
    data_out[0] = 11;
    for(var i = 0; i < 6; i++) {
      data_out[i+1] = (MP.alpha[i]*r2d + IP_PROTOCOL_LIM)/(2*IP_PROTOCOL_LIM)*
          PROTOCOL_MAX;
    }
    COM.sendPacket(outFun(data_out), N_out);
  }
});

setInterval(function() {
  disp('Učestanost telemetrije: '+(N/telemetry_hz_interval).toFixed(2)+' Hz');
  N = 0;
}, telemetry_hz_interval * 1000);
```

Listing Y.1 - `flightGearSimulationUDP.jsl`

Z

Skripte ArdupilotLink.jsl i ArdupilotSitlLink.jsl

U listingu Z.1 je data skripta `ArdupilotLink.jsl` za povezivanje sa *ArduPilot* autopilotom i prikazivanje instrumenata.

```
/**
 * @file Ardupilot SITL Link script
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var install_module = true;
var tcp_ip = '192.168.144.1';
var lat = 44.821190;
var lon = 20.434067;
var tcp_port = 20003;
var tcp_timeout = 3000;
// --------------------

// - Map

// Tilesets
// Roadmap
// http://mt0.google.com/vt/lyrs=m&hl=en&x={x}&y={y}&z={z}
// Terrain
// http://mt0.google.com/vt/lyrs=p&hl=en&x={x}&y={y}&z={z}
// Altered roadmap
// http://mt0.google.com/vt/lyrs=r&hl=en&x={x}&y={y}&z={z}
// Satellite only
// http://mt0.google.com/vt/lyrs=s&hl=en&x={x}&y={y}&z={z}
// Terrain only
// http://mt0.google.com/vt/lyrs=t&hl=en&x={x}&y={y}&z={z}
// Hybrid
// http://mt0.google.com/vt/lyrs=y&hl=en&x={x}&y={y}&z={z}
var map = await webmap('http://mt0.google.com/vt/lyrs=y&hl=en&x={x}&y={y}&z={z}'
    );
map.win.setTitle('Map');
map.setCenter(lat, lon);
map.setZoom(13);
```

```
38 var aircraft_marker = map.addMarker(lat, lon);
39 aircraft_marker.setIcon({
40   iconUrl: pwd+'aircraft-marker.svg',
41   iconSize: [50, 50],
42   iconAnchor: [25, 25]
43 });
44
45 // - PFD
46
47 var { PFD } = require(pwd + 'PFD.js');
48 var pfd_win = await openCanvas();
49 pfd_win.setTitle('PFD');
50 var pfd = new PFD(pfd_win);
51
52 // - Mavlink
53
54 if(install_module) {
55   disp('Installing node-mavlink-wrapper');
56   installModule(pwd + 'node-mavlink-wrapper');
57 }
58 var mavlink = require(pwd + 'node-mavlink-wrapper');
59 var mavlink_com;
60
61 var armed = false;
62 var flying = false;
63 var prearm;
64 var flight_mode;
65
66 var reader;
67 var reconnect_timeout;
68 var heartbeat_loop;
69 var connected = false;
70
71 class BasicRemilp extends mavlink.MavLinkData {
72   constructor() {
73     super();
74     this.time_boot_ms = 0;
75     this.ax = 0;
76     this.ay = 0;
77     this.az = 0;
78     this.roll = 0;
79     this.pitch = 0;
80     this.yaw_rate = 0;
81   }
82 }
83 BasicRemilp.MSG_ID = 11050;
84 BasicRemilp.MSG_NAME = 'BASIC_REMILP';
85 BasicRemilp.PAYLOAD_LENGTH = 28;
86 BasicRemilp.MAGIC_NUMBER = 15;
87 BasicRemilp.FIELDS = [
88   new mavlink.MavLinkPacketField('time_boot_ms', 'time_boot_ms', 0, false, 4, '
       uint32_t', 'ms'),
89   new mavlink.MavLinkPacketField('ax', 'ax', 4, false, 4, 'float', ''),
90   new mavlink.MavLinkPacketField('ay', 'ay', 8, false, 4, 'float', ''),
91   new mavlink.MavLinkPacketField('az', 'az', 12, false, 4, 'float', ''),
92   new mavlink.MavLinkPacketField('roll', 'roll', 16, false, 4, 'float', ''),
93   new mavlink.MavLinkPacketField('pitch', 'pitch', 20, false, 4, 'float', ''),
94   new mavlink.MavLinkPacketField('yaw_rate', 'yaw_rate', 24, false, 4, 'float',
       ''),
```

```
95 ];
96
97 mavlink.registerCustomMessageMagicNumber(String(BasicRemilp.MSG_ID), BasicRemilp
       .MAGIC_NUMBER);
98
99 var CUSTOM_REGISTRY = {};
100 CUSTOM_REGISTRY[BasicRemilp.MSG_ID] = BasicRemilp;
101
102 var REGISTRY = {
103   ...mavlink.minimal.REGISTRY,
104   ...mavlink.common.REGISTRY,
105   ...mavlink.ardupilotmega.REGISTRY,
106   ...CUSTOM_REGISTRY
107 };
108
109 var mavlink_protocol = new mavlink.MavLinkProtocolV2(240, 190);
110
111 var MAV_MODE_FLAG = { // Bit mask
112   7: {'name': 'MAV_MODE_FLAG_SAFETY_ARMED', 'short': 'SAFETY_ARMED'},
113   6: {'name': 'MAV_MODE_FLAG_MANUAL_INPUT_ENABLED', 'short': '
          MANUAL_INPUT_ENABLED'},
114   5: {'name': 'MAV_MODE_FLAG_HIL_ENABLED', 'short': 'HIL_ENABLED'},
115   4: {'name': 'MAV_MODE_FLAG_STABILIZE_ENABLED', 'short': 'STABILIZE_ENABLED'},
116   3: {'name': 'MAV_MODE_FLAG_GUIDED_ENABLED', 'short': 'GUIDED_ENABLED'},
117   2: {'name': 'MAV_MODE_FLAG_AUTO_ENABLED', 'short': 'AUTO_ENABLED'},
118   1: {'name': 'MAV_MODE_FLAG_TEST_ENABLED', 'short': 'TEST_ENABLED'},
119   0: {'name': 'MAV_MODE_FLAG_CUSTOM_MODE_ENABLED', 'short': 'CUSTOM_MODE_ENABLED
          '}
120 };
121
122 var MAV_STATE = {
123   0: {'name': 'MAV_STATE_UNINIT', 'short': 'UNINIT'},
124   1: {'name': 'MAV_STATE_BOOT', 'short': 'BOOT'},
125   2: {'name': 'MAV_STATE_CALIBRATING', 'short': 'CALIBRATING'},
126   3: {'name': 'MAV_STATE_STANDBY', 'short': 'STANDBY'},
127   4: {'name': 'MAV_STATE_ACTIVE', 'short': 'ACTIVE'},
128   5: {'name': 'MAV_STATE_CRITICAL', 'short': 'CRITICAL'},
129   6: {'name': 'MAV_STATE_EMERGENCY', 'short': 'EMERGENCY'},
130   7: {'name': 'MAV_STATE_POWEROFF', 'short': 'POWEROFF'},
131   8: {'name': 'MAV_STATE_FLIGHT_TERMINATION', 'short': 'FLIGHT_TERMINATION'},
132 };
133 var MAV_STATE_INV = invertEnum(MAV_STATE, 'short');
134
135 var MAV_SEVERITY = {
136   0: {'name': 'EMERGENCY'},
137   1: {'name': 'ALERT'},
138   2: {'name': 'CRITICAL'},
139   3: {'name': 'ERROR'},
140   4: {'name': 'WARNING'},
141   5: {'name': 'NOTICE'},
142   6: {'name': 'INFO'},
143   7: {'name': 'DEBUG'},
144 };
145
146 var MAV_SYS_STATUS_SENSOR = { // Bit mask
147   0: {'name': 'MAV_SYS_STATUS_SENSOR_3D_GYRO', 'short': '3D_GYRO'},
148   1: {'name': 'MAV_SYS_STATUS_SENSOR_3D_ACCEL', 'short': '3D_ACCEL'},
149   2: {'name': 'MAV_SYS_STATUS_SENSOR_3D_MAG', 'short': '3D_MAG'},
```

```
150     3: {'name': 'MAV_SYS_STATUS_SENSOR_ABSOLUTE_PRESSURE', 'short': '
            ABSOLUTE_PRESSURE'},
151     4: {'name': 'MAV_SYS_STATUS_SENSOR_DIFFERENTIAL_PRESSURE', 'short': '
            DIFFERENTIAL_PRESSURE'},
152     5: {'name': 'MAV_SYS_STATUS_SENSOR_GPS', 'short': 'GPS'},
153     6: {'name': 'MAV_SYS_STATUS_SENSOR_OPTICAL_FLOW', 'short': 'OPTICAL_FLOW'},
154     7: {'name': 'MAV_SYS_STATUS_SENSOR_VISION_POSITION', 'short': 'VISION_POSITION
            '},
155     8: {'name': 'MAV_SYS_STATUS_SENSOR_LASER_POSITION', 'short': 'LASER_POSITION'
            },
156     9: {'name': 'MAV_SYS_STATUS_SENSOR_EXTERNAL_GROUND_TRUTH', 'short': '
            EXTERNAL_GROUND_TRUTH'},
157     10: {'name': 'MAV_SYS_STATUS_SENSOR_ANGULAR_RATE_CONTROL', 'short': '
            ANGULAR_RATE_CONTROL'},
158     11: {'name': 'MAV_SYS_STATUS_SENSOR_ATTITUDE_STABILIZATION', 'short': '
            ATTITUDE_STABILIZATION'},
159     12: {'name': 'MAV_SYS_STATUS_SENSOR_YAW_POSITION', 'short': 'YAW_POSITION'},
160     13: {'name': 'MAV_SYS_STATUS_SENSOR_Z_ALTITUDE_CONTROL', 'short': '
            Z_ALTITUDE_CONTROL'},
161     14: {'name': 'MAV_SYS_STATUS_SENSOR_XY_POSITION_CONTROL', 'short': '
            XY_POSITION_CONTROL'},
162     15: {'name': 'MAV_SYS_STATUS_SENSOR_MOTOR_OUTPUTS', 'short': 'MOTOR_OUTPUTS'},
163     16: {'name': 'MAV_SYS_STATUS_SENSOR_RC_RECEIVER', 'short': 'RC_RECEIVER'},
164     17: {'name': 'MAV_SYS_STATUS_SENSOR_3D_GYRO2', 'short': '3D_GYRO2'},
165     18: {'name': 'MAV_SYS_STATUS_SENSOR_3D_ACCEL2', 'short': '3D_ACCEL2'},
166     19: {'name': 'MAV_SYS_STATUS_SENSOR_3D_MAG2', 'short': '3D_MAG2'},
167     20: {'name': 'MAV_SYS_STATUS_GEOFENCE', 'short': 'GEOFENCE'},
168     21: {'name': 'MAV_SYS_STATUS_AHRS', 'short': 'AHRS'},
169     22: {'name': 'MAV_SYS_STATUS_TERRAIN', 'short': 'TERRAIN'},
170     23: {'name': 'MAV_SYS_STATUS_REVERSE_MOTOR', 'short': 'REVERSE_MOTOR'},
171     24: {'name': 'MAV_SYS_STATUS_LOGGING', 'short': 'LOGGING'},
172     25: {'name': 'MAV_SYS_STATUS_SENSOR_BATTERY', 'short': 'SENSOR_BATTERY'},
173     26: {'name': 'MAV_SYS_STATUS_SENSOR_PROXIMITY', 'short': 'SENSOR_PROXIMITY'},
174     27: {'name': 'MAV_SYS_STATUS_SENSOR_SATCOM', 'short': 'SENSOR_SATCOM'},
175     28: {'name': 'MAV_SYS_STATUS_PREARM_CHECK', 'short': 'PREARM_CHECK'},
176     29: {'name': 'MAV_SYS_STATUS_OBSTACLE_AVOIDANCE', 'short': 'OBSTACLE_AVOIDANCE
            '},
177     30: {'name': 'MAV_SYS_STATUS_SENSOR_PROPULSION', 'short': 'SENSOR_PROPULSION'
            },
178     31: {'name': 'MAV_SYS_STATUS_EXTENSION_USED', 'short': 'EXTENSION_USED'}
179 };
180
181 var FLIGHT_MODE = {
182     0: {'name': 'STABILIZE'},
183     1: {'name': 'ACRO'},
184     2: {'name': 'ALTHOLD'},
185     3: {'name': 'AUTO'},
186     4: {'name': 'GUIDED'},
187     5: {'name': 'LOITER'},
188     6: {'name': 'RTL'},
189     7: {'name': 'CIRCLE'},
190     9: {'name': 'LAND'},
191     11: {'name': 'DRIFT'},
192     13: {'name': 'SPORT'},
193     14: {'name': 'FLIP'},
194     15: {'name': 'AUTOTUNE'},
195     16: {'name': 'POSHOLD'},
196     17: {'name': 'BRAKE'},
197     18: {'name': 'THROW'},
```

```
198     19: {'name': 'AVOID_ADSB'},
199     20: {'name': 'GUIDED_NO_GNSS'},
200     21: {'name': 'SMART_RTL'},
201     22: {'name': 'FLOW_HOLD'},
202     23: {'name': 'FOLLOW'},
203     24: {'name': 'ZIG_ZAG'},
204     25: {'name': 'SYSTEM_ID'},
205     26: {'name': 'HELI_AUTOROTATE'},
206     27: {'name': 'AUTO_RTL'},
207 };
208 var FLIGHT_MODE_INV = invertEnum(FLIGHT_MODE, 'name');
209
210 function setSystemStatus(data) {
211   if(data.customMode != flight_mode) {
212     flight_mode = data.customMode;
213     disp('Flight mode: '+FLIGHT_MODE[flight_mode].name);
214   }
215
216   var mode = getBitFlags(MAV_MODE_FLAG, 'short', data.baseMode);
217
218   if(mode.SAFETY_ARMED != armed) {
219     armed = mode.SAFETY_ARMED;
220     if(armed) {
221       if(data.systemStatus > MAV_STATE_INV["STANDBY"]) {
222         disp('Flying');
223       } else {
224         disp('Armed');
225       }
226     } else {
227       disp('Disarmed');
228     }
229   }
230
231   if(data.systemStatus > MAV_STATE_INV["STANDBY"] && armed) {
232     if(!flying) {
233       flying = true;
234       disp('Flying');
235     }
236   } else {
237     if(flying) {
238       flying = false;
239       disp('Landed');
240     }
241   }
242 }
243
244 function setSysStatusData(data) {
245   var sensors_health_obj = getBitFlags(MAV_SYS_STATUS_SENSOR, 'short', data.
        onboardControlSensorsHealth);
246
247   if(prearm != sensors_health_obj.PREARM_CHECK) {
248     prearm = sensors_health_obj.PREARM_CHECK;
249     if(prearm) {
250       disp('[Prearm] Ready!');
251     } else {
252       disp('[Prearm] Not ready!');
253     }
254   }
255 }
```

```
256
257 function setVfrHudData(data) {
258   pfd.setVerticalSpeed(data.climb);
259   pfd.setGroundspeed(data.groundspeed*3.6); // [km/h]
260   pfd.setAirspeed(data.airspeed*3.6); // [km/h]
261 }
262
263 function setAttitudeData(data) {
264   pfd.setPitch(data.pitch*r2d);
265   pfd.setRoll(data.roll*r2d);
266 }
267
268 function setGlobalPositionIntData(data) {
269   pfd.setAltitude(data.alt/1000);
270   pfd.setHeading(data.hdg/100);
271   aircraft_marker.setPosition(data.lat/1E7, data.lon/1E7);
272   aircraft_marker.setRotation(data.hdg/100);
273 }
274
275 function setBasicRemilpData(data) {
276   //disp(data);
277 }
278
279 function setMessageInterval(id, frequency) {
280   let message = new mavlink.common.CommandLong();
281   message.targetSystem = 1;
282   message.targetComponent = 1;
283   message.command = mavlink.common.MavCmd.SET_MESSAGE_INTERVAL;
284   message._param1 = id;
285   if(frequency == 0) {
286     message._param2 = -1;
287   } else {
288     message._param2 = round(1E6/frequency, 0);
289   }
290   mavlink.send(mavlink_com, message, mavlink_protocol);
291 }
292
293 function setParamById(id, value, type) {
294   var message = new mavlink.common.ParamSet();
295   message.targetSystem = 1;
296   message.targetComponent = 1;
297   message.paramId = id;
298   message.paramValue = value;
299   message.paramType = type;
300   mavlink.send(mavlink_com, message, mavlink_protocol);
301 }
302
303 function setFlightMode(mode) {
304   var message = new mavlink.common.CommandLong();
305   message.targetSystem = 1;
306   message.targetComponent = 1;
307   message.command = mavlink.common.MavCmd.DO_SET_MODE;
308   message._param1 = 1;
309   message._param2 = FLIGHT_MODE_INV[mode];
310   mavlink.send(mavlink_com, message, mavlink_protocol);
311 }
312
313 function armVehicle(cmd = 1) {
314   var message = new mavlink.common.CommandLong();
```

```
315   message.targetSystem = 1;
316   message.targetComponent = 1;
317   message.command = mavlink.common.MavCmd.COMPONENT_ARM_DISARM;
318   message._param1 = cmd;
319   mavlink.send(mavlink_com, message, mavlink_protocol);
320 }
321
322 function sendRcChannelsOverride(channels) {
323   let message = new mavlink.common.RcChannelsOverride();
324   message.targetSystem = 1;
325   message.targetComponent = 1;
326   message.chan1Raw = channels[0];
327   message.chan2Raw = channels[1];
328   message.chan3Raw = channels[2];
329   message.chan4Raw = channels[3];
330   message.chan5Raw = channels[4];
331   message.chan6Raw = channels[5];
332   message.chan7Raw = channels[6];
333   message.chan8Raw = channels[7];
334   message.chan9Raw = channels[8];
335   message.chan10Raw = channels[9];
336   message.chan11Raw = channels[10];
337   message.chan12Raw = channels[11];
338   message.chan13Raw = channels[12];
339   message.chan14Raw = channels[13];
340   message.chan15Raw = channels[14];
341   message.chan16Raw = channels[15];
342   message.chan17Raw = 65535;
343   message.chan18Raw = 65535;
344   mavlink.send(mavlink_com, message, mavlink_protocol);
345 }
346
347 function onAircraftConnected() {
348   setMessageInterval(mavlink.common.SysStatus.MSG_ID, 10);
349   setMessageInterval(mavlink.common.VfrHud.MSG_ID, 10);
350   setMessageInterval(mavlink.common.Attitude.MSG_ID, 10);
351   setMessageInterval(mavlink.common.GlobalPositionInt.MSG_ID, 10);
352
353   // Start sending heartbeat
354   clearIntervalIf(heartbeat_loop);
355   heartbeat_loop = setInterval(function() {
356     let message = new mavlink.minimal.Heartbeat();
357     message.type = 6; // MAV_TYPE_GCS
358     message.autopilot = 8; // MAV_AUTOPILOT_INVALID
359     message.baseMode = 0;
360     message.customMode = 0;
361     message.systemStatus = 0;
362
363     mavlink.send(mavlink_com, message, mavlink_protocol);
364   }, 1000);
365 }
366
367 function connect() {
368   mavlink_com = tcp(tcp_ip, tcp_port, function() {
369     mavlink_com.setTimeout(0);
370     console.log('Connected to TCP telemetry stream');
371   });
372   mavlink_com.setNoDelay(true);
373   mavlink_com.setKeepAlive(true);
```

```
374   mavlink_com.setTimeout(tcp_timeout);
375   mavlink_com.setOnData(function(data) { });
376   mavlink_com.setOnError(function(data) {
377     if(data) {
378       connected = false;
379       mavlink_com.close();
380
381       heartbeat_loop = clearIntervalIf(heartbeat_loop);
382       clearTimeout(reconnect_timeout);
383       reconnect_timeout = setTimeout(function() {
384         connect();
385       }, 100);
386     }
387   });
388
389   reader = mavlink_com.com
390     .pipe(new mavlink.MavLinkPacketSplitter())
391     .pipe(new mavlink.MavLinkPacketParser());
392
393   reader.on('data', packet => {
394     var hit = REGISTRY[packet.header.msgid];
395     if(hit) {
396       var data = packet.protocol.data(packet.payload, hit);
397       var packet_name = data.constructor.name;
398
399       if(!connected) {
400         connected = true;
401         onAircraftConnected();
402       }
403       switch(packet_name) {
404         case 'Heartbeat':
405           setSystemStatus(data);
406           break;
407         case 'SysStatus':
408           setSysStatusData(data);
409           break;
410         case 'VfrHud':
411           setVfrHudData(data);
412           break;
413         case 'Attitude':
414           setAttitudeData(data);
415           break;
416         case 'GlobalPositionInt':
417           setGlobalPositionIntData(data);
418           break;
419         case 'StatusText':
420           disp('From autopilot ['+
421              MAV_SEVERITY[data.severity].name+']: '+data.text);
422           break;
423         case 'BasicRemilp':
424           setBasicRemilpData(data);
425           break;
426         default:
427           disp('Received packet ['+packet_name+']: ' + stringify(data));
428           break;
429       }
430     }
431   });
432 }
```

```
433
434 connect();
```

Listing Z.1 - `ArdupilotLink.jsl`

U listingu Z.2 je data skripta `ArdupilotSitlLink.jsl` za povezivanje sa *ArduPilot* `SITL` autopilotom i prikazivanje instrumenata.

```
1  /**
2   * @file Ardupilot SITL Link script
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8  close('all'); clear; clc; tic;
9  disp(' --- ' + jsl_file_name + ' ---');
10 // - Ulazni podaci
11 var folder = pwd + "sitl";
12 var sitl_exe_name = ["ArduCopter.exe", "arducopter_custom.exe"][1];
13 var model = '+';
14 var params = pwd+'sitl/copter.parm';
15 var lat = 44.821190;
16 var lon = 20.434067;
17 var height = 119;
18 var heading = 90;
19 var install_module = false;
20 var tcp_ip = '127.0.0.1';
21 var tcp_port = 5760;
22 var tcp_timeout = 3000;
23 var gamepad_id = 'Logitech Cordless RumblePad 2 (STANDARD GAMEPAD Vendor: 046d
       Product: c219)';
24 // --------------------
25
26 // - Map
27
28 // Tilesets
29 // Roadmap
30 // http://mt0.google.com/vt/lyrs=m&hl=en&x={x}&y={y}&z={z}
31 // Terrain
32 // http://mt0.google.com/vt/lyrs=p&hl=en&x={x}&y={y}&z={z}
33 // Altered roadmap
34 // http://mt0.google.com/vt/lyrs=r&hl=en&x={x}&y={y}&z={z}
35 // Satellite only
36 // http://mt0.google.com/vt/lyrs=s&hl=en&x={x}&y={y}&z={z}
37 // Terrain only
38 // http://mt0.google.com/vt/lyrs=t&hl=en&x={x}&y={y}&z={z}
39 // Hybrid
40 // http://mt0.google.com/vt/lyrs=y&hl=en&x={x}&y={y}&z={z}
41 var map = await webmap('http://mt0.google.com/vt/lyrs=y&hl=en&x={x}&y={y}&z={z}'
       );
42 map.win.setTitle('Map');
43 map.setCenter(lat, lon);
44 map.setZoom(13);
45 var aircraft_marker = map.addMarker(lat, lon);
46 aircraft_marker.setIcon({
47   iconUrl: pwd+'aircraft-marker.svg',
48   iconSize: [50, 50],
49   iconAnchor: [25, 25]
50 });
51
```

```
52 // - PFD
53
54 var { PFD } = require(pwd + 'PFD.js');
55 var pfd_win = await openCanvas();
56 pfd_win.setTitle('PFD');
57 var pfd = new PFD(pfd_win);
58
59 // - Gamepad
60
61 var gamepad_win_ready = false;
62 var gamepad_win;
63 var doms = {};
64 var gamepad = getGamepad(gamepad_id, 20);
65 gamepad.setOnConnect(async function() {
66   disp('Gamepad ' + gamepad_id + ' found!');
67   gamepad_win = await openWindowBlank();
68
69   gamepad_win.document.body.innerHTML += '<div id="timestamp-cont" style="margin
       : 5px 20px; padding: 3px; border: 2px solid #ddd; border-radius: 5px; font-
      weight:bold; color: #333;">t: <span id="timestamp-value">0</span></div>';
70   for(let i = 0; i < gamepad.data.axes.length; i++) {
71     gamepad_win.document.body.innerHTML += '<div id="axis-'+i+'-cont" style="
         margin: 5px 20px; padding: 3px; border: 2px solid #ddd; border-radius: 5
         px; font-weight:bold; color: #333;">A'+(i+1)+': <span id="axis-'+i+'-
         value">0</span></div>';
72   }
73   for(let i = 0; i < gamepad.data.buttons.length; i++) {
74     gamepad_win.document.body.innerHTML += '<div id="button-'+i+'-cont" style="
         margin: 5px 20px; padding: 3px; border: 2px solid #ddd; border-radius: 5
         px; font-weight:bold; color: #333;">B'+(i+1)+': <span id="button-'+i+'-
         value">0</span></div>';
75   }
76
77   doms.timestamp = gamepad_win.document.getElementById('timestamp-value');
78   for(let i = 0; i < gamepad.data.axes.length; i++) {
79     doms['axis-'+i] = gamepad_win.document.getElementById('axis-'+i+'-value');
80   }
81   for(let i = 0; i < gamepad.data.buttons.length; i++) {
82     doms['button-'+i] = gamepad_win.document.getElementById('button-'+i+'-value'
         );
83   }
84   gamepad_win_ready = true;
85
86   gamepad_win.setSize(400, (1+gamepad.data.axes.length+gamepad.data.buttons.
       length)*31+10+35);
87   gamepad_win.setTitle('Gamepad');
88 });
89
90 gamepad.setOnData(function(data) {
91   var rc_data = zeros(16);
92   rc_data[0] = (data.axes[2])*500+1500;
93   rc_data[1] = (data.axes[3])*500+1500;
94   rc_data[2] = (-data.axes[1])*500+1500;
95   rc_data[3] = (data.axes[0])*500+1500;
96   sendRcChannelsOverride(rc_data);
97
98   if(gamepad_win_ready) {
99     doms.timestamp.innerText = data.timestamp;
100    for(let i = 0; i < data.axes.length; i++) {
```

```
101         doms['axis-'+i].innerText = data.axes[i];
102       }
103       for(let i = 0; i < data.buttons.length; i++) {
104         doms['button-'+i].innerText = '{p: '+data.buttons[i].pressed+', t: '+data.
              buttons[i].touched+', v: '+data.buttons[i].value+'}';
105       }
106     }
107 });
108
109 // - Mavlink
110
111 if(install_module) {
112   installModule(pwd + 'node-mavlink-wrapper');
113 }
114 var mavlink = require(pwd + 'node-mavlink-wrapper');
115 var mavlink_com;
116
117 var armed = false;
118 var flying = false;
119 var prearm;
120 var flight_mode;
121
122 var reader;
123 var reconnect_timeout;
124 var heartbeat_loop;
125 var connected = false;
126
127 class BasicRemilp extends mavlink.MavLinkData {
128   constructor() {
129     super();
130     this.time_boot_ms = 0;
131     this.ax = 0;
132     this.ay = 0;
133     this.az = 0;
134     this.roll = 0;
135     this.pitch = 0;
136     this.yaw_rate = 0;
137   }
138 }
139 BasicRemilp.MSG_ID = 11050;
140 BasicRemilp.MSG_NAME = 'BASIC_REMILP';
141 BasicRemilp.PAYLOAD_LENGTH = 28;
142 BasicRemilp.MAGIC_NUMBER = 15;
143 BasicRemilp.FIELDS = [
144   new mavlink.MavLinkPacketField('time_boot_ms', 'time_boot_ms', 0, false, 4, '
        uint32_t', 'ms'),
145   new mavlink.MavLinkPacketField('ax', 'ax', 4, false, 4, 'float', ''),
146   new mavlink.MavLinkPacketField('ay', 'ay', 8, false, 4, 'float', ''),
147   new mavlink.MavLinkPacketField('az', 'az', 12, false, 4, 'float', ''),
148   new mavlink.MavLinkPacketField('roll', 'roll', 16, false, 4, 'float', ''),
149   new mavlink.MavLinkPacketField('pitch', 'pitch', 20, false, 4, 'float', ''),
150   new mavlink.MavLinkPacketField('yaw_rate', 'yaw_rate', 24, false, 4, 'float',
        ''),
151 ];
152
153 mavlink.registerCustomMessageMagicNumber(String(BasicRemilp.MSG_ID), BasicRemilp
      .MAGIC_NUMBER);
154
155 var CUSTOM_REGISTRY = {};
```

```
156 CUSTOM_REGISTRY[BasicRemilp.MSG_ID] = BasicRemilp;
157
158 var REGISTRY = {
159   ...mavlink.minimal.REGISTRY,
160   ...mavlink.common.REGISTRY,
161   ...mavlink.ardupilotmega.REGISTRY,
162   ...CUSTOM_REGISTRY
163 };
164
165 var mavlink_protocol = new mavlink.MavLinkProtocolV2(240, 190);
166
167 var MAV_MODE_FLAG = { // Bit mask
168   7: {'name': 'MAV_MODE_FLAG_SAFETY_ARMED', 'short': 'SAFETY_ARMED'},
169   6: {'name': 'MAV_MODE_FLAG_MANUAL_INPUT_ENABLED', 'short': '
        MANUAL_INPUT_ENABLED'},
170   5: {'name': 'MAV_MODE_FLAG_HIL_ENABLED', 'short': 'HIL_ENABLED'},
171   4: {'name': 'MAV_MODE_FLAG_STABILIZE_ENABLED', 'short': 'STABILIZE_ENABLED'},
172   3: {'name': 'MAV_MODE_FLAG_GUIDED_ENABLED', 'short': 'GUIDED_ENABLED'},
173   2: {'name': 'MAV_MODE_FLAG_AUTO_ENABLED', 'short': 'AUTO_ENABLED'},
174   1: {'name': 'MAV_MODE_FLAG_TEST_ENABLED', 'short': 'TEST_ENABLED'},
175   0: {'name': 'MAV_MODE_FLAG_CUSTOM_MODE_ENABLED', 'short': 'CUSTOM_MODE_ENABLED
        '}
176 };
177
178 var MAV_STATE = {
179   0: {'name': 'MAV_STATE_UNINIT', 'short': 'UNINIT'},
180   1: {'name': 'MAV_STATE_BOOT', 'short': 'BOOT'},
181   2: {'name': 'MAV_STATE_CALIBRATING', 'short': 'CALIBRATING'},
182   3: {'name': 'MAV_STATE_STANDBY', 'short': 'STANDBY'},
183   4: {'name': 'MAV_STATE_ACTIVE', 'short': 'ACTIVE'},
184   5: {'name': 'MAV_STATE_CRITICAL', 'short': 'CRITICAL'},
185   6: {'name': 'MAV_STATE_EMERGENCY', 'short': 'EMERGENCY'},
186   7: {'name': 'MAV_STATE_POWEROFF', 'short': 'POWEROFF'},
187   8: {'name': 'MAV_STATE_FLIGHT_TERMINATION', 'short': 'FLIGHT_TERMINATION'},
188 };
189 var MAV_STATE_INV = invertEnum(MAV_STATE, 'short');
190
191 var MAV_SEVERITY = {
192   0: {'name': 'EMERGENCY'},
193   1: {'name': 'ALERT'},
194   2: {'name': 'CRITICAL'},
195   3: {'name': 'ERROR'},
196   4: {'name': 'WARNING'},
197   5: {'name': 'NOTICE'},
198   6: {'name': 'INFO'},
199   7: {'name': 'DEBUG'},
200 };
201
202 var MAV_SYS_STATUS_SENSOR = { // Bit mask
203   0: {'name': 'MAV_SYS_STATUS_SENSOR_3D_GYRO', 'short': '3D_GYRO'},
204   1: {'name': 'MAV_SYS_STATUS_SENSOR_3D_ACCEL', 'short': '3D_ACCEL'},
205   2: {'name': 'MAV_SYS_STATUS_SENSOR_3D_MAG', 'short': '3D_MAG'},
206   3: {'name': 'MAV_SYS_STATUS_SENSOR_ABSOLUTE_PRESSURE', 'short': '
        ABSOLUTE_PRESSURE'},
207   4: {'name': 'MAV_SYS_STATUS_SENSOR_DIFFERENTIAL_PRESSURE', 'short': '
        DIFFERENTIAL_PRESSURE'},
208   5: {'name': 'MAV_SYS_STATUS_SENSOR_GPS', 'short': 'GPS'},
209   6: {'name': 'MAV_SYS_STATUS_SENSOR_OPTICAL_FLOW', 'short': 'OPTICAL_FLOW'},
210   7: {'name': 'MAV_SYS_STATUS_SENSOR_VISION_POSITION', 'short': 'VISION_POSITION
```

```
        '},
211     8: {'name': 'MAV_SYS_STATUS_SENSOR_LASER_POSITION', 'short': 'LASER_POSITION'
        },
212     9: {'name': 'MAV_SYS_STATUS_SENSOR_EXTERNAL_GROUND_TRUTH', 'short': '
        EXTERNAL_GROUND_TRUTH'},
213     10: {'name': 'MAV_SYS_STATUS_SENSOR_ANGULAR_RATE_CONTROL', 'short': '
        ANGULAR_RATE_CONTROL'},
214     11: {'name': 'MAV_SYS_STATUS_SENSOR_ATTITUDE_STABILIZATION', 'short': '
        ATTITUDE_STABILIZATION'},
215     12: {'name': 'MAV_SYS_STATUS_SENSOR_YAW_POSITION', 'short': 'YAW_POSITION'},
216     13: {'name': 'MAV_SYS_STATUS_SENSOR_Z_ALTITUDE_CONTROL', 'short': '
        Z_ALTITUDE_CONTROL'},
217     14: {'name': 'MAV_SYS_STATUS_SENSOR_XY_POSITION_CONTROL', 'short': '
        XY_POSITION_CONTROL'},
218     15: {'name': 'MAV_SYS_STATUS_SENSOR_MOTOR_OUTPUTS', 'short': 'MOTOR_OUTPUTS'},
219     16: {'name': 'MAV_SYS_STATUS_SENSOR_RC_RECEIVER', 'short': 'RC_RECEIVER'},
220     17: {'name': 'MAV_SYS_STATUS_SENSOR_3D_GYRO2', 'short': '3D_GYRO2'},
221     18: {'name': 'MAV_SYS_STATUS_SENSOR_3D_ACCEL2', 'short': '3D_ACCEL2'},
222     19: {'name': 'MAV_SYS_STATUS_SENSOR_3D_MAG2', 'short': '3D_MAG2'},
223     20: {'name': 'MAV_SYS_STATUS_GEOFENCE', 'short': 'GEOFENCE'},
224     21: {'name': 'MAV_SYS_STATUS_AHRS', 'short': 'AHRS'},
225     22: {'name': 'MAV_SYS_STATUS_TERRAIN', 'short': 'TERRAIN'},
226     23: {'name': 'MAV_SYS_STATUS_REVERSE_MOTOR', 'short': 'REVERSE_MOTOR'},
227     24: {'name': 'MAV_SYS_STATUS_LOGGING', 'short': 'LOGGING'},
228     25: {'name': 'MAV_SYS_STATUS_SENSOR_BATTERY', 'short': 'SENSOR_BATTERY'},
229     26: {'name': 'MAV_SYS_STATUS_SENSOR_PROXIMITY', 'short': 'SENSOR_PROXIMITY'},
230     27: {'name': 'MAV_SYS_STATUS_SENSOR_SATCOM', 'short': 'SENSOR_SATCOM'},
231     28: {'name': 'MAV_SYS_STATUS_PREARM_CHECK', 'short': 'PREARM_CHECK'},
232     29: {'name': 'MAV_SYS_STATUS_OBSTACLE_AVOIDANCE', 'short': 'OBSTACLE_AVOIDANCE
        '},
233     30: {'name': 'MAV_SYS_STATUS_SENSOR_PROPULSION', 'short': 'SENSOR_PROPULSION'
        },
234     31: {'name': 'MAV_SYS_STATUS_EXTENSION_USED', 'short': 'EXTENSION_USED'}
235 };
236
237 var FLIGHT_MODE = {
238     0: {'name': 'STABILIZE'},
239     1: {'name': 'ACRO'},
240     2: {'name': 'ALTHOLD'},
241     3: {'name': 'AUTO'},
242     4: {'name': 'GUIDED'},
243     5: {'name': 'LOITER'},
244     6: {'name': 'RTL'},
245     7: {'name': 'CIRCLE'},
246     9: {'name': 'LAND'},
247     11: {'name': 'DRIFT'},
248     13: {'name': 'SPORT'},
249     14: {'name': 'FLIP'},
250     15: {'name': 'AUTOTUNE'},
251     16: {'name': 'POSHOLD'},
252     17: {'name': 'BRAKE'},
253     18: {'name': 'THROW'},
254     19: {'name': 'AVOID_ADSB'},
255     20: {'name': 'GUIDED_NO_GNSS'},
256     21: {'name': 'SMART_RTL'},
257     22: {'name': 'FLOW_HOLD'},
258     23: {'name': 'FOLLOW'},
259     24: {'name': 'ZIG_ZAG'},
260     25: {'name': 'SYSTEM_ID'},
```

```
261   26: {'name': 'HELI_AUTOROTATE'},
262   27: {'name': 'AUTO_RTL'},
263 };
264 var FLIGHT_MODE_INV = invertEnum(FLIGHT_MODE, 'name');
265
266 function setSystemStatus(data) {
267   if(data.customMode != flight_mode) {
268     flight_mode = data.customMode;
269     disp('Flight mode: '+FLIGHT_MODE[flight_mode].name);
270   }
271
272   var mode = getBitFlags(MAV_MODE_FLAG, 'short', data.baseMode);
273
274   if(mode.SAFETY_ARMED != armed) {
275     armed = mode.SAFETY_ARMED;
276     if(armed) {
277       if(data.systemStatus > MAV_STATE_INV["STANDBY"]) {
278         disp('Flying');
279       } else {
280         disp('Armed');
281       }
282     } else {
283       disp('Disarmed');
284     }
285   }
286
287   if(data.systemStatus > MAV_STATE_INV["STANDBY"] && armed) {
288     if(!flying) {
289       flying = true;
290       disp('Flying');
291     }
292   } else {
293     if(flying) {
294       flying = false;
295       disp('Landed');
296     }
297   }
298 }
299
300 function setSysStatusData(data) {
301   var sensors_health_obj = getBitFlags(MAV_SYS_STATUS_SENSOR, 'short', data.
        onboardControlSensorsHealth);
302
303   if(prearm != sensors_health_obj.PREARM_CHECK) {
304     prearm = sensors_health_obj.PREARM_CHECK;
305     if(prearm) {
306       disp('[Prearm] Ready!');
307     } else {
308       disp('[Prearm] Not ready!');
309     }
310   }
311 }
312
313 function setVfrHudData(data) {
314   pfd.setVerticalSpeed(data.climb);
315   pfd.setGroundspeed(data.groundspeed*3.6); // [km/h]
316   pfd.setAirspeed(data.airspeed*3.6); // [km/h]
317 }
318
```

```
319 function setAttitudeData(data) {
320   pfd.setPitch(data.pitch*r2d);
321   pfd.setRoll(data.roll*r2d);
322 }
323
324 function setGlobalPositionIntData(data) {
325   pfd.setAltitude(data.alt/1000);
326   pfd.setHeading(data.hdg/100);
327   aircraft_marker.setPosition(data.lat/1E7, data.lon/1E7);
328   aircraft_marker.setRotation(data.hdg/100);
329 }
330
331 function setBasicRemilpData(data) {
332   //disp(data);
333 }
334
335 function setMessageInterval(id, frequency) {
336   let message = new mavlink.common.CommandLong();
337   message.targetSystem = 1;
338   message.targetComponent = 1;
339   message.command = mavlink.common.MavCmd.SET_MESSAGE_INTERVAL;
340   message._param1 = id;
341   if(frequency == 0) {
342     message._param2 = -1;
343   } else {
344     message._param2 = round(1E6/frequency, 0);
345   }
346   mavlink.send(mavlink_com, message, mavlink_protocol);
347 }
348
349 function setParamById(id, value, type) {
350   var message = new mavlink.common.ParamSet();
351   message.targetSystem = 1;
352   message.targetComponent = 1;
353   message.paramId = id;
354   message.paramValue = value;
355   message.paramType = type;
356   mavlink.send(mavlink_com, message, mavlink_protocol);
357 }
358
359 function setFlightMode(mode) {
360   var message = new mavlink.common.CommandLong();
361   message.targetSystem = 1;
362   message.targetComponent = 1;
363   message.command = mavlink.common.MavCmd.DO_SET_MODE;
364   message._param1 = 1;
365   message._param2 = FLIGHT_MODE_INV[mode];
366   mavlink.send(mavlink_com, message, mavlink_protocol);
367 }
368
369 function armVehicle(cmd = 1) {
370   var message = new mavlink.common.CommandLong();
371   message.targetSystem = 1;
372   message.targetComponent = 1;
373   message.command = mavlink.common.MavCmd.COMPONENT_ARM_DISARM;
374   message._param1 = cmd;
375   mavlink.send(mavlink_com, message, mavlink_protocol);
376 }
377
```

```
function sendRcChannelsOverride(channels) {
  let message = new mavlink.common.RcChannelsOverride();
  message.targetSystem = 1;
  message.targetComponent = 1;
  message.chan1Raw = channels[0];
  message.chan2Raw = channels[1];
  message.chan3Raw = channels[2];
  message.chan4Raw = channels[3];
  message.chan5Raw = channels[4];
  message.chan6Raw = channels[5];
  message.chan7Raw = channels[6];
  message.chan8Raw = channels[7];
  message.chan9Raw = channels[8];
  message.chan10Raw = channels[9];
  message.chan11Raw = channels[10];
  message.chan12Raw = channels[11];
  message.chan13Raw = channels[12];
  message.chan14Raw = channels[13];
  message.chan15Raw = channels[14];
  message.chan16Raw = channels[15];
  message.chan17Raw = 65535;
  message.chan18Raw = 65535;
  mavlink.send(mavlink_com, message, mavlink_protocol);
}

function onAircraftConnected() {
  setMessageInterval(mavlink.common.SysStatus.MSG_ID, 10);
  setMessageInterval(mavlink.common.VfrHud.MSG_ID, 10);
  setMessageInterval(mavlink.common.Attitude.MSG_ID, 10);
  setMessageInterval(mavlink.common.GlobalPositionInt.MSG_ID, 10);

  // Start sending heartbeat
  clearIntervalIf(heartbeat_loop);
  heartbeat_loop = setInterval(function() {
    let message = new mavlink.minimal.Heartbeat();
    message.type = 6; // MAV_TYPE_GCS
    message.autopilot = 8; // MAV_AUTOPILOT_INVALID
    message.baseMode = 0;
    message.customMode = 0;
    message.systemStatus = 0;

    mavlink.send(mavlink_com, message, mavlink_protocol);
  }, 1000);
}

function connect() {
  mavlink_com = tcp(tcp_ip, tcp_port, function() {
    mavlink_com.setTimeout(0);
    console.log('Connected to TCP telemetry stream');
  });
  mavlink_com.setNoDelay(true);
  mavlink_com.setKeepAlive(true);
  mavlink_com.setTimeout(tcp_timeout);
  mavlink_com.setOnData(function(data) { });
  mavlink_com.setOnError(function(data) {
    if(data) {
      connected = false;
      mavlink_com.close();

```

```
437       heartbeat_loop = clearIntervalIf(heartbeat_loop);
438       clearTimeout(reconnect_timeout);
439       reconnect_timeout = setTimeout(function() {
440         connect();
441       }, 100);
442     }
443   });
444
445   reader = mavlink_com.com
446     .pipe(new mavlink.MavLinkPacketSplitter())
447     .pipe(new mavlink.MavLinkPacketParser());
448
449   reader.on('data', packet => {
450     var hit = REGISTRY[packet.header.msgid];
451     if(hit) {
452       var data = packet.protocol.data(packet.payload, hit);
453       var packet_name = data.constructor.name;
454
455       if(!connected) {
456         connected = true;
457         onAircraftConnected();
458       }
459       switch(packet_name) {
460         case 'Heartbeat':
461           setSystemStatus(data);
462           break;
463         case 'SysStatus':
464           setSysStatusData(data);
465           break;
466         case 'VfrHud':
467           setVfrHudData(data);
468           break;
469         case 'Attitude':
470           setAttitudeData(data);
471           break;
472         case 'GlobalPositionInt':
473           setGlobalPositionIntData(data);
474           break;
475         case 'StatusText':
476           disp('From autopilot ['+
477              MAV_SEVERITY[data.severity].name+']: '+data.text);
478           break;
479         case 'BasicRemilp':
480           setBasicRemilpData(data);
481           break;
482         default:
483           disp('Received packet ['+packet_name+']: ' + stringify(data));
484           break;
485       }
486     }
487   });
488 }
489
490 // - SITL
491
492 disp('Starting Ardupilot SITL...');
493
494 var [flag1, pids1] = isProgramRunning(sitl_exe_name);
495 var process = spawn(sitl_exe_name,
```

```
496   ['--model', model, '--speedup=1',
497   '--home', lat+','+lon+','+height+','+heading,
498   '--defaults', params], {cwd: folder });
499 var [flag2, pids2] = isProgramRunning(sitl_exe_name);
500 var process_pid = pids2.filter(function(e) {
501   return !pids1.includes(e)
502 });
503
504 process.stdout.on('data', function(data) {
505   disp('SITL stdout: '+ data);
506 });
507
508 process.stderr.on('data', function(data) {
509   disp('SITL stderr: '+ data);
510 });
511
512 process.on('close', function(code) {
513   disp('SITL close: '+ code);
514 });
515
516 connect();
```

Listing Z.2 - `ArdupilotSitlLink.jsl`

U listingu Z.3 je dat kod klase `PFD` za prikaz primarnog ekrana leta.

```
1  /**
2   * @file Primary flight display library
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8
9  /**
10  * Class for primary flight display.
11  */
12 class PFD {
13
14   /**
15    * Initializes a new Primary Flight Display within a specified container.
16    * @param {string} container - The ID of the HTML container where the display
          will be rendered.
17    */
18   constructor(win) {
19
20     // Properties
21     // --------------------
22     this.win = win;
23     this.pitch = 0;
24     this.roll = 0;
25     this.heading = 0;
26     this.airspeed = 0;
27     this.altitude = 0;
28     this.vertical_speed = 0;
29
30     this.last_pitch = 0;
31     this.last_roll = 0;
32     this.last_heading = 0;
33     this.last_airspeed = 0;
34     this.last_altitude = 0;
```

```
35      this.last_vertical_speed = 0;
36
37      this.visible = false;
38
39      // Options
40      // --------------------
41      var width = 250;
42      this.width = width;
43      var height = 330;
44      this.height = height;
45      var header_height = 0;
46      this.header_height = header_height;
47      var font_size = 12;
48      var font_padding = 2;
49      var unit_font_size = 7;
50      var sky_color = "#2e85c7ff";
51      var ground_color = "#5e554fff";
52      var compass_color = '#000000aa';
53      var overlay_color = '#00000033';
54      var center_height = (height-header_height)/2+header_height;
55
56      var overlay_dx = 2;
57      var overlay_radius = 3;
58      var overlay_font_dy = 2;
59
60      var cross_height = 3;
61      var cross_line_width = 20;
62      var cross_text_dy = 1.5;
63      var cross_text_dx = 5;
64
65      var horizon_height = 2;
66
67      var compass_size = 0.55;
68      var compass_radius = compass_size*width/2;
69      var compass_tick_length = 12;
70      var compass_tick_width = 2;
71      var compas_pointer_top_width = 8;
72      var compas_pointer_top_height = 8;
73      var compas_pointer_bottom_width = 8;
74      var compas_pointer_bottom_height = 10;
75      var compass_text_radius = 3;
76      var compass_text_dy = 3;
77      var compass_tick_val = 15;
78
79      var logo_size = 40;
80      var logo_y = 15;
81
82      var pitch_values = [10, 20];
83      var pitch_minor_values = [5, 15];
84      var pitch_value_step = 10;
85      var pitch_lines_dy = 30;
86      var pitch_line_width = 40;
87      var pitch_minor_line_width = 30;
88      var pitch_line_height = 2;
89
90      var roll_values = [10, 20, 30, 45];
91      var roll_minor_values = [5, 15];
92      var roll_size = 0.9;
93      var roll_arc_radius = roll_size*width/2;
```

```
94      var roll_arc_width = 2;
95      var roll_line_width = 12;
96      var roll_minor_line_width = 8;
97      var roll_line_height = 2;
98      var roll_pointer_width = 10;
99      var roll_pointer_height = 12;
100     var roll_text_dy = 3;
101
102     var airspeed_x_size = 0.15;
103     var airspeed_width = airspeed_x_size*width;
104     var airspeed_y_size = 0.4;
105     var airspeed_height = airspeed_y_size*height;
106     var airspeed_ticks = 5; // odd number
107     var airspeed_value_step = 5;
108     var airspeed_line_width = 20;
109     var airspeed_line_height = 2;
110
111     var groundspeed_top = 0.01*height;
112     var groundspeed_x_size = 0.15;
113     var groundspeed_width = groundspeed_x_size*width;
114     var groundspeed_y_size = 0.09;
115     var groundspeed_height = groundspeed_y_size*height;
116
117     var altitude_x_size = 0.13;
118     var altitude_width = altitude_x_size*width;
119     var altitude_y_size = 0.4;
120     var altitude_height = altitude_y_size*height;
121     var altitude_ticks = 11; // odd number
122     var altitude_value_step = 10;
123     var altitude_line_width = 15;
124     var altitude_line_height = 2;
125
126     var vertical_speed_x_size = 0.07;
127     var vertical_speed_width = vertical_speed_x_size*width;
128     var vertical_speed_y_size = 0.4;
129     var vertical_speed_height = vertical_speed_y_size*height;
130     var vertical_speed_ticks = 11; // odd number
131     var vertical_speed_value_step = 2;
132     var vertical_speed_line_width = 7;
133     var vertical_speed_line_height = 2;
134
135     // Constants
136     // --------------------
137     this.pitch_lines_dy = pitch_lines_dy;
138     this.pitch_value_step = pitch_value_step;
139     this.pitch_y = this.pitch_lines_dy/this.pitch_value_step*this.pitch;
140     this.vertical_speed_width = vertical_speed_width;
141     this.airspeed_dy = airspeed_height/airspeed_ticks;
142     this.altitude_dy = altitude_height/altitude_ticks;
143     this.vertical_speed_dy = vertical_speed_height/vertical_speed_ticks;
144     this.airspeed_value_step = airspeed_value_step;
145     this.altitude_value_step = altitude_value_step;
146     this.vertical_speed_value_step = vertical_speed_value_step;
147     this.overlay_dx = overlay_dx;
148
149     var font_height = font_size+font_padding;
150
151     // Base SVG
152     // --------------------
```

```
153     var svg = win.d3.select('#d3-svg')
154        .attr('class', 'pfd-svg')
155        .attr('width', width)
156        .attr('height', height)
157        .attr('viewBox', '0 0 '+width+' '+height)
158        .style('width', '100%')
159        .style('height', '100%');
160     var defs = svg.append("defs");
161
162     // Attitude group
163     // --------------------
164     var attitude = svg.append("g")
165       .attr('class', 'pfd-atitude')
166       .attr('transform', 'translate('+width/2+','+center_height+')');
167
168     var horizon = attitude.append("g")
169       .attr('class', 'pfd-horizon');
170     this.svg_horizon = horizon;
171
172     // Ground
173     var ground = horizon.append('rect')
174       .attr('class', 'pfd-ground')
175       .attr('x', -3*width)
176       .attr('y', 0)
177       .attr('width', 6*width)
178       .attr('height', 4*height)
179       .attr('fill', ground_color);
180
181     // Sky
182     var sky = horizon.append('rect')
183       .attr('class', 'pfd-sky')
184       .attr('x', -3*width)
185       .attr('y', -4*height)
186       .attr('width', 6*width)
187       .attr('height', 4*height)
188       .attr('fill', sky_color);
189
190     // Horizon line
191     var horizon_line = horizon.append('rect')
192       .attr('class', 'pfd-horizon')
193       .attr('x', -3*width)
194       .attr('y', -horizon_height/2)
195       .attr('width', 6*width)
196       .attr('height', horizon_height)
197       .attr('fill', "#ffffffff");
198
199     // Pitch
200     // --------------------
201     for(let i = 0; i < pitch_values.length; i++) {
202       // Pitch ticks top
203       horizon.append("line")
204         .attr("x1", -pitch_line_width/2)
205         .attr("y1", (i+1)*pitch_lines_dy)
206         .attr("x2", pitch_line_width/2)
207         .attr("y2", (i+1)*pitch_lines_dy)
208         .attr("stroke", "#ffffffff")
209         .style("stroke-width", pitch_line_height)
210         .style("stroke-linecap", "round");
211
```

```
212       // Pitch ticks bottom
213       horizon.append("line")
214         .attr("x1", -pitch_line_width/2)
215         .attr("y1", -(i+1)*pitch_lines_dy)
216         .attr("x2", pitch_line_width/2)
217         .attr("y2", -(i+1)*pitch_lines_dy)
218         .attr("stroke", "#ffffffff")
219         .style("stroke-width", pitch_line_height)
220         .style("stroke-linecap", "round");
221     }
222
223     for(let i = 0; i < pitch_minor_values.length; i++) {
224       // Pitch minor ticks top
225       horizon.append("line")
226         .attr("x1", -pitch_minor_line_width/2)
227         .attr("y1", (i+0.5)*pitch_lines_dy)
228         .attr("x2", pitch_minor_line_width/2)
229         .attr("y2", (i+0.5)*pitch_lines_dy)
230         .attr("stroke", "#ffffffff")
231         .style("stroke-width", pitch_line_height)
232         .style("stroke-linecap", "round");
233
234       // Pitch minor ticks bottom
235       horizon.append("line")
236         .attr("x1", -pitch_minor_line_width/2)
237         .attr("y1", -(i+0.5)*pitch_lines_dy)
238         .attr("x2", pitch_minor_line_width/2)
239         .attr("y2", -(i+0.5)*pitch_lines_dy)
240         .attr("stroke", "#ffffffff")
241         .style("stroke-width", pitch_line_height)
242         .style("stroke-linecap", "round");
243     }
244
245     // Roll
246     // --------------------
247     var roll = svg.append("g")
248       .attr('class', 'pfd-roll')
249       .attr('transform', 'translate('+width/2+','+center_height+')');
250
251     var roll_disc = roll.append("g")
252       .attr('class', 'pfd-roll-disc');
253     this.svg_roll_disc = roll_disc;
254
255     var roll_arc_data = win.d3.arc()
256         .innerRadius(roll_arc_radius-roll_arc_width/2)
257         .outerRadius(roll_arc_radius+roll_arc_width/2)
258         .startAngle(-roll_values[roll_values.length-1]*Math.PI/180)
259         .endAngle(roll_values[roll_values.length-1]*Math.PI/180);
260
261     // Roll disc
262     roll_disc.append("path")
263         .attr("class", "arc")
264         .attr("d", roll_arc_data)
265         .attr("fill","#ffffffff");
266
267     // Roll ticks
268     for(var i = 0; i < roll_values.length; i++) {
269       roll_disc.append("line")
270         .attr("x1", 0)
```

```
271         .attr("y1", 0)
272         .attr("x2", roll_line_width)
273         .attr("y2", 0)
274         .attr("stroke", "#ffffffff")
275         .style("stroke-width", roll_line_height)
276         .style("stroke-linecap", "round")
277         .attr("transform", "rotate("+(roll_values[i]-90)+",0,0) translate("+(
                roll_arc_radius)+",0)");
278       roll_disc.append("line")
279         .attr("x1", 0)
280         .attr("y1", 0)
281         .attr("x2", roll_line_width)
282         .attr("y2", 0)
283         .attr("stroke", "#ffffffff")
284         .style("stroke-width", roll_line_height)
285         .style("stroke-linecap", "round")
286         .attr("transform", "rotate("+(-roll_values[i]-90)+",0,0) translate("+(
                roll_arc_radius)+",0)");
287     }
288
289     // Roll minor ticks
290     for(let i = 0; i < roll_minor_values.length; i++) {
291       roll_disc.append("line")
292         .attr("x1", 0)
293         .attr("y1", 0)
294         .attr("x2", roll_minor_line_width)
295         .attr("y2", 0)
296         .attr("stroke", "#ffffffff")
297         .style("stroke-width", roll_line_height)
298         .style("stroke-linecap", "round")
299         .attr("transform", "rotate("+(roll_minor_values[i]-90)+",0,0) translate(
                "+(roll_arc_radius)+",0)");
300       roll_disc.append("line")
301         .attr("x1", 0)
302         .attr("y1", 0)
303         .attr("x2", roll_minor_line_width)
304         .attr("y2", 0)
305         .attr("stroke", "#ffffffff")
306         .style("stroke-width", roll_line_height)
307         .style("stroke-linecap", "round")
308         .attr("transform", "rotate("+(-roll_minor_values[i]-90)+",0,0) translate
                ("+(roll_arc_radius)+",0)");
309     }
310
311     // Roll zero
312     roll_disc.append("polygon")
313      .attr("points", "0,0 "+roll_pointer_width/2+",-"+roll_pointer_height+" -"+
            roll_pointer_width/2+",-"+roll_pointer_height+"")
314      .style("fill", "#ffffffff")
315      .attr("transform", "translate(0,"+(-roll_arc_radius)+")");
316
317     // Compass group
318     // --------------------
319     var compass = svg.append("g")
320       .attr('class', 'pfd-compass')
321       .attr('transform', 'translate('+width/2+','+height+')');
322
323     // Compass background
324     var compass_bg = compass.append('circle')
```

```
325         .attr('class', 'pfd-compass-bg')
326         .attr('cx', 0)
327         .attr('cy', 0)
328         .attr('r', compass_radius)
329         .style('fill', compass_color);
330
331       // Compass text
332       var compass_text_bg = compass.append("rect")
333         .attr("width", 3*font_size)
334         .attr("height", 2.5*font_size)
335         .attr("rx", compass_text_radius)
336         .style('fill', '#000000ff')
337         .attr("transform", "translate(-"+(1.5*font_size)+",-"+(compass_radius+
              font_size+compass_text_dy)+")");
338       var compass_text = compass.append("text")
339         .text('0')
340         .attr("x", 0)
341         .attr("y", 0)
342         .attr("text-anchor", "middle")
343         .attr("font-family", "Roboto, Arial")
344         .attr("font-size", font_size)
345         .attr("font-weight", "bold")
346         .attr("fill", "#ffffffff")
347         .style("text-shadow", "2px 2px 5px #000")
348         .attr("transform", "translate(0,-"+(compass_radius+compass_text_dy)+")");
349       this.svg_compass_text = compass_text;
350
351       // Compass disk
352       var compass_disk = compass.append("g")
353         .attr('class', 'pfd-compass-disk');
354       this.svg_compass_disk = compass_disk;
355
356       // Compass ticks
357       for(var alpha = 0; alpha < 360; alpha = alpha+compass_tick_val) {
358         compass_disk.append("line")
359           .attr("x1", 0)
360           .attr("y1", 0)
361           .attr("x2", compass_tick_length)
362           .attr("y2", 0)
363           .attr("stroke", "#ffffffff")
364           .style("stroke-width", compass_tick_width)
365           .style("stroke-linecap", "round")
366           .attr("transform", "rotate("+alpha+",0,0) translate("+(compass_radius-
              compass_tick_length)+",0)");
367       }
368
369       // Compass poles
370       var compas_poles = ['N', 'E', 'S', 'W'];
371       for(let i = 0; i < 4; i++) {
372         compass_disk.append("text")
373           .text(compas_poles[i])
374           .attr("x", 0)
375           .attr("y", 0)
376           .attr("text-anchor", "middle")
377           .attr("font-family", "Roboto, Arial")
378           .attr("font-size", font_size)
379           .attr("font-weight", "bold")
380           .attr("fill", "#ffffffff")
381           .style("text-shadow", "2px 2px 5px #000")
```

```
382         .attr("transform", "rotate("+(i*90)+",0,0) translate(0,"+(-
                compass_radius+compass_tick_length+font_size)+")");
383     }
384
385     // Compass pointers
386     compass.append("polygon")
387       .attr("points", "0,0 "+compas_pointer_top_width/2+",-"+
             compas_pointer_top_height+" -"+compas_pointer_top_width/2+",-"+
             compas_pointer_top_height+"")
388       .style("fill", "#2e85c7ff")
389       .attr("transform", "translate(0,"+(-compass_radius+
             compas_pointer_top_height+1)+")");
390     compass.append("polygon")
391       .attr("points", "0,0 "+compas_pointer_bottom_width/2+","+
             compas_pointer_bottom_height+" -"+compas_pointer_bottom_width/2+","+
             compas_pointer_bottom_height+"")
392       .style("fill", "#2e85c7ff")
393       .attr("transform", "translate(0,"+(-compass_radius+compass_tick_length+
             font_size+1)+")");
394
395    // Attitude cross
396     // --------------------
397     var cross = svg.append("g")
398       .attr('class', 'pfd-cross')
399       .attr('transform', 'translate('+width/2+','+center_height+')');
400
401     // Pitch cross
402     cross.append("line")
403       .attr("x1", 0.5*cross_line_width)
404       .attr("y1", 0)
405       .attr("x2", 1.5*cross_line_width)
406       .attr("y2", 0)
407       .attr("stroke", "#000000ff")
408       .style("stroke-width", cross_height)
409       .style("stroke-linecap", "round");
410     cross.append("line")
411       .attr("x1", -0.5*cross_line_width)
412       .attr("y1", 0)
413       .attr("x2", -1.5*cross_line_width)
414       .attr("y2", 0)
415       .attr("stroke", "#000000ff")
416       .style("stroke-width", cross_height)
417       .style("stroke-linecap", "round");
418     cross.append("circle")
419       .attr('cx', 0)
420       .attr('cy', 0)
421       .attr('r', cross_height)
422       .style('fill', '#000000ff');
423
424     // Pitch text
425     var pitch_text = cross.append("text")
426       .text("0")
427       .attr("x", 0)
428       .attr("y", -cross_height-cross_text_dy)
429       .attr("text-anchor", "middle")
430       .attr("font-family", "Roboto, Arial")
431       .attr("font-size", font_size)
432       .attr("font-weight", "bold")
433       .attr("fill", "#000000ff");
```

```
434     this.svg_pitch_text = pitch_text;
435
436     // Roll cross
437     cross.append("polygon")
438      .attr("points", "0,0 "+roll_pointer_width/2+","+roll_pointer_height+" -"+
            roll_pointer_width/2+","+roll_pointer_height+"")
439      .style("fill", "#000000ff")
440      .attr("transform", "translate(0,"+(-roll_arc_radius)+")");
441
442     // Roll text
443     var roll_text = cross.append("text")
444       .text("0")
445       .attr("x", 0)
446       .attr("y", -roll_arc_radius-roll_line_width-roll_text_dy)
447       .attr("text-anchor", "middle")
448       .attr("font-family", "Roboto, Arial")
449       .attr("font-size", font_size)
450       .attr("font-weight", "bold")
451       .attr("fill", "#000000ff");
452     this.svg_roll_text = roll_text;
453
454     // Airspeed
455     // --------------------
456     var airspeed_cont = svg.append('rect')
457       .attr('class', 'pfd-airspeed')
458       .attr('x', overlay_dx)
459       .attr('y', center_height-airspeed_height/2)
460       .attr("rx", overlay_radius)
461       .attr('width', airspeed_width)
462       .attr('height', airspeed_height)
463       .attr('fill', overlay_color);
464
465     defs.append("clipPath")
466         .attr("id", "pfd-airspeed-clip")
467         .append("rect")
468         .attr('x', -airspeed_width)
469         .attr('y', -airspeed_height/2)
470         .attr("rx", overlay_radius)
471         .attr('width', airspeed_width-overlay_dx)
472         .attr('height', airspeed_height);
473
474     // Airspeed ticks
475     var airspeed_ticks_wrap_group = svg.append("g")
476       .attr('class', 'pfd-wrap-airspeed-ticks')
477       .attr("clip-path", "url(#pfd-airspeed-clip)")
478       .attr('transform', 'translate('+(overlay_dx+airspeed_width)+','+
            center_height+')');
479     var airspeed_ticks_group = airspeed_ticks_wrap_group.append("g")
480       .attr('class', 'pfd-airspeed-ticks');
481     this.svg_airspeed_ticks_group = airspeed_ticks_group;
482
483     var airspeed_dy = airspeed_height/airspeed_ticks;
484     for(let i = 0; i < airspeed_ticks+2; i++) {
485       airspeed_ticks_group.append("line")
486         .attr("x1", 0)
487         .attr("y1", (i-Math.ceil(airspeed_ticks/2))*airspeed_dy)
488         .attr("x2", -airspeed_line_width)
489         .attr("y2", (i-Math.ceil(airspeed_ticks/2))*airspeed_dy)
490         .attr("stroke", "#ffffffff")
```

```
491         .style("stroke-width", airspeed_line_height)
492         .style("stroke-linecap", "round");
493     }
494
495     // Airspeed text container
496     var airspeed_text_cont = svg.append("polygon")
497       .attr("points", "0,0 0,"+font_height/2+" "+airspeed_width*0.8+","+
              font_height/2+" "+airspeed_width+",0 "+airspeed_width*0.8+",-"+
              font_height/2+" 0,-"+font_height/2)
498       .style("fill", "#000000ff")
499       .attr("transform", "translate("+overlay_dx+","+center_height+")");
500
501     // Airspeed text
502     var airspeed_text = svg.append("text")
503       .text("0")
504       .attr("x", overlay_dx+airspeed_width/2)
505       .attr("y", center_height+font_size/2-overlay_font_dy)
506       .attr("text-anchor", "middle")
507       .attr("font-family", "Roboto, Arial")
508       .attr("font-size", font_size)
509       .attr("font-weight", "bold")
510       .style("text-shadow", "2px 2px 5px #000")
511       .attr("fill", "#ffffffff");
512     this.svg_airspeed_text = airspeed_text;
513
514     // Airspeed unit text background
515     var airspeed_unit_text_cont = svg.append('rect')
516       .attr('x', 2*overlay_dx)
517       .attr('y', center_height+airspeed_height/2-unit_font_size-2*font_padding-
              overlay_font_dy)
518       .attr("rx", overlay_radius)
519       .attr('width', airspeed_x_size*width-2*overlay_dx)
520       .attr('height', unit_font_size+2*font_padding)
521       .attr('fill', "#00000066");
522
523     // Airspeed unit text
524     var airspeed_unit_text = svg.append("text")
525       .text("km/h")
526       .attr("x", overlay_dx+airspeed_width/2)
527       .attr("y", center_height+airspeed_height/2-unit_font_size/2-font_padding)
528       .attr("text-anchor", "middle")
529       .attr("font-family", "Roboto, Arial")
530       .attr("font-size", unit_font_size)
531       .attr("font-weight", "bold")
532       .style("text-shadow", "2px 2px 5px #000")
533       .attr("fill", "#ffffffff");
534
535     // Groundspeed
536     // --------------------
537     // Groundspeed container
538     var groundspeed_cont = svg.append('rect')
539       .attr('class', 'pfd-airspeed')
540       .attr('x', overlay_dx)
541       .attr('y', center_height+airspeed_height/2+groundspeed_top)
542       .attr("rx", overlay_radius)
543       .attr('width', groundspeed_width)
544       .attr('height', groundspeed_height)
545       .attr('fill', overlay_color);
546
```

```
547     // Groundspeed text
548     var groundspeed_text = svg.append("text")
549       .text("0")
550       .attr("x", overlay_dx+groundspeed_width/2)
551       .attr("y", center_height+airspeed_height/2+groundspeed_top+font_height)
552       .attr("text-anchor", "middle")
553       .attr("font-family", "Roboto, Arial")
554       .attr("font-size", font_size)
555       .attr("font-weight", "bold")
556       .style("text-shadow", "2px 2px 5px #000")
557       .attr("fill", "#ffffffff");
558     this.svg_groundspeed_text = groundspeed_text;
559
560     // Groundspeed unit text background
561     var groundspeed_unit_text_cont = svg.append('rect')
562       .attr('x', 2*overlay_dx)
563       .attr('y', center_height+airspeed_height/2+groundspeed_top+font_height+
              font_padding)
564       .attr("rx", overlay_radius)
565       .attr('width', airspeed_x_size*width-2*overlay_dx)
566       .attr('height', unit_font_size+2*font_padding)
567       .attr('fill', "#00000066");
568
569     // Groundspeed unit text
570     var groundspeed_unit_text = svg.append("text")
571       .text("GS [km/h]")
572       .attr("x", overlay_dx+groundspeed_width/2)
573       .attr("y", center_height+airspeed_height/2+groundspeed_top+font_height+
              unit_font_size/2+3*font_padding)
574       .attr("text-anchor", "middle")
575       .attr("font-family", "Roboto, Arial")
576       .attr("font-size", unit_font_size)
577       .attr("font-weight", "bold")
578       .style("text-shadow", "2px 2px 5px #000")
579       .attr("fill", "#ffffffff");
580
581     // Altitude
582     // --------------------
583     var altitude_x = width-altitude_width-vertical_speed_width-2*overlay_dx;
584     var altitude_cont = svg.append('rect')
585       .attr('class', 'pfd-altitude')
586       .attr('x', altitude_x)
587       .attr('y', center_height-altitude_height/2)
588       .attr("rx", overlay_radius)
589       .attr('width', altitude_x_size*width)
590       .attr('height', altitude_height)
591       .attr('fill', overlay_color);
592
593     defs.append("clipPath")
594         .attr("id", "pfd-altitude-clip")
595         .append("rect")
596         .attr('x', overlay_dx)
597         .attr('y', -altitude_height/2)
598         .attr("rx", overlay_radius)
599         .attr('width', altitude_width-overlay_dx)
600         .attr('height', altitude_height);
601
602     // Altitude ticks
603     var altitude_ticks_wrap_group = svg.append("g")
```

```
604         .attr('class', 'pfd-wrap-altitude-ticks')
605         .attr("clip-path", "url(#pfd-altitude-clip)")
606         .attr('transform', 'translate('+altitude_x+','+center_height+')');
607       var altitude_ticks_group = altitude_ticks_wrap_group.append("g")
608         .attr('class', 'pfd-altitude-ticks');
609       this.svg_altitude_ticks_group = altitude_ticks_group;
610
611       var altitude_dy = altitude_height/altitude_ticks;
612       for(let i = 0; i < altitude_ticks+2; i++) {
613         altitude_ticks_group.append("line")
614           .attr("x1", 0)
615           .attr("y1", (i-Math.ceil(altitude_ticks/2))*altitude_dy)
616           .attr("x2", altitude_line_width)
617           .attr("y2", (i-Math.ceil(altitude_ticks/2))*altitude_dy)
618           .attr("stroke", "#ffffffff")
619           .style("stroke-width", altitude_line_height)
620           .style("stroke-linecap", "round");
621       }
622
623       // Altitude text container
624       var altitude_text_cont = svg.append("polygon")
625         .attr("points", "0,0 "+altitude_width*0.2+",-"+font_height/2+" "+
              altitude_width+",-"+font_height/2+" "+altitude_width+",0 "+
              altitude_width+","+font_height/2+" "+altitude_width*0.2+","+font_height
              /2)
626         .style("fill", "#000000ff")
627         .attr("transform", "translate("+altitude_x+","+center_height+")");
628
629       // Altitude text
630       var altitude_text = svg.append("text")
631         .text("0")
632         .attr("x", altitude_x+altitude_width/2)
633         .attr("y", center_height+font_size/2-overlay_font_dy)
634         .attr("text-anchor", "middle")
635         .attr("font-family", "Roboto, Arial")
636         .attr("font-size", font_size)
637         .attr("font-weight", "bold")
638         .style("text-shadow", "2px 2px 5px #000")
639         .attr("fill", "#ffffffff");
640       this.svg_altitude_text = altitude_text;
641
642       // Altitude unit text background
643       var altitude_unit_text_cont = svg.append('rect')
644         .attr('x', altitude_x+overlay_dx)
645         .attr('y', center_height+altitude_height/2-unit_font_size-2*font_padding-
              overlay_font_dy)
646         .attr("rx", overlay_radius)
647         .attr('width', altitude_x_size*width-2*overlay_dx)
648         .attr('height', unit_font_size+2*font_padding)
649         .attr('fill', "#00000066");
650
651       // Altitude unit text
652       var altitude_unit_text = svg.append("text")
653         .text("m")
654         .attr("x", altitude_x+altitude_width/2)
655         .attr("y", center_height+altitude_height/2-unit_font_size/2-font_padding)
656         .attr("text-anchor", "middle")
657         .attr("font-family", "Roboto, Arial")
658         .attr("font-size", unit_font_size)
```

```
659         .attr("font-weight", "bold")
660         .style("text-shadow", "2px 2px 5px #000")
661         .attr("fill", "#ffffffff");
662 
663       // Vertical speed
664       // --------------------
665       var vertical_speed_cont = svg.append('rect')
666         .attr('class', 'pfd-vertical-speed')
667         .attr('x', width-vertical_speed_width-overlay_dx)
668         .attr('y', center_height-vertical_speed_height/2)
669         .attr("rx", overlay_radius)
670         .attr('width', vertical_speed_width)
671         .attr('height', vertical_speed_height)
672         .attr('fill', overlay_color);
673 
674       defs.append("clipPath")
675           .attr("id", "pfd-vertical-speed-clip")
676           .append("rect")
677           .attr('x', overlay_dx)
678           .attr('y', -vertical_speed_height/2)
679           .attr("rx", overlay_radius)
680           .attr('width', vertical_speed_width-overlay_dx)
681           .attr('height', vertical_speed_height);
682 
683       // Vertical speed ticks
684       var vertical_speed_ticks_wrap_group = svg.append("g")
685         .attr('class', 'pfd-wrap-vertical-speed-ticks')
686         .attr('transform', 'translate('+(width-vertical_speed_width-overlay_dx)+',
              '+center_height+')');
687       var vertical_speed_ticks_group = vertical_speed_ticks_wrap_group.append("g")
688         .attr('class', 'pfd-vertical-speed-ticks')
689         .attr("clip-path", "url(#pfd-vertical-speed-clip)");
690 
691       // Vertical speed indicator
692       var vertical_speed_indicator = vertical_speed_ticks_group.append("polygon")
693         .attr("points", this.overlay_dx+",0 "+vertical_speed_width+",0 "+this.
              overlay_dx+","+0+"")
694         .style("fill", "#ffffffaa");
695       this.svg_vertical_speed_indicator = vertical_speed_indicator;
696 
697       var vertical_speed_dy = vertical_speed_height/vertical_speed_ticks;
698       for(let i = 0; i < vertical_speed_ticks+2; i++) {
699         vertical_speed_ticks_group.append("line")
700           .attr("x1", 0)
701           .attr("y1", (i-Math.ceil(vertical_speed_ticks/2))*vertical_speed_dy)
702           .attr("x2", vertical_speed_line_width)
703           .attr("y2", (i-Math.ceil(vertical_speed_ticks/2))*vertical_speed_dy)
704           .attr("stroke", "#ffffffff")
705           .style("stroke-width", vertical_speed_line_height)
706           .style("stroke-linecap", "round");
707       }
708 
709       // Vertical speed container
710       var vertical_speed_text_cont = svg.append("polygon")
711         .attr("points", "0,0 0,-"+font_height/2+" "+vertical_speed_width+",-"+
              font_height/2+" "+vertical_speed_width+",0 "+vertical_speed_width+","+
              font_height/2+" 0,"+font_height/2)
712         .style("fill", "#000000ff")
713         .attr("transform", "translate("+(width-vertical_speed_width-overlay_dx)+",
```

```
          "+center_height+")");
714
715       // Vertical speed text
716       var vertical_speed_text = svg.append("text")
717         .text("0")
718         .attr("x", width-vertical_speed_width/2-overlay_dx)
719         .attr("y", center_height+font_size/2-overlay_font_dy)
720         .attr("text-anchor", "middle")
721         .attr("font-family", "Roboto, Arial")
722         .attr("font-size", font_size)
723         .attr("font-weight", "bold")
724         .style("text-shadow", "2px 2px 5px #000")
725         .attr("fill", "#ffffffff");
726       this.svg_vertical_speed_text = vertical_speed_text;
727
728       // Vertical speed unit text background
729       var vertical_speed_unit_text_cont = svg.append('rect')
730         .attr('x', width-vertical_speed_width)
731         .attr('y', center_height+vertical_speed_height/2-unit_font_size-2*
              font_padding-overlay_font_dy)
732         .attr("rx", overlay_radius)
733         .attr('width', vertical_speed_width-2*overlay_dx)
734         .attr('height', unit_font_size+2*font_padding)
735         .attr('fill', "#00000066");
736
737       // Vertical speed unit text
738       var vertical_speed_unt_text = svg.append("text")
739         .text("m/s")
740         .attr("x", width-vertical_speed_width/2-overlay_dx)
741         .attr("y", center_height+vertical_speed_height/2-unit_font_size/2-
              font_padding)
742         .attr("text-anchor", "middle")
743         .attr("font-family", "Roboto, Arial")
744         .attr("font-size", unit_font_size)
745         .attr("font-weight", "bold")
746         .style("text-shadow", "2px 2px 5px #000")
747         .attr("fill", "#ffffffff");
748     }
749
750     /**
751      * Updates the pitch angle displayed.
752      * @param {number} pitch - The pitch angle in degrees.
753      */
754     setPitch(pitch) {
755       this.pitch = pitch;
756       this.showPitch();
757     }
758
759     /**
760      * Updates the roll angle displayed.
761      * @param {number} roll - The roll angle in degrees.
762      */
763     setRoll(roll){
764       this.roll = roll;
765       this.showRoll();
766     }
767
768     /**
769      * Updates the heading angle displayed.
```

```
770     * @param {number} heading - The heading angle in degrees.
771     */
772    setHeading(heading){
773      this.heading = heading;
774      this.showHeading();
775    }
776
777    /**
778     * Updates the airspeed displayed in meters per second.
779     * @param {number} airspeed - The airspeed.
780     */
781    setAirspeed(airspeed){
782      this.airspeed = airspeed;
783      this.showAirspeed();
784    }
785
786    /**
787     * Updates the ground speed displayed in kilometers per hour.
788     * @param {number} groundspeed - The ground speed.
789     */
790    setGroundspeed(groundspeed){
791      this.groundspeed = groundspeed;
792      this.showGroundspeed();
793    }
794
795    /**
796     * Updates the altitude displayed in meters.
797     * @param {number} altitude - The altitude.
798     */
799    setAltitude(altitude){
800      this.altitude = altitude;
801      this.showAltitude();
802    }
803
804    /**
805     * Updates the vertical speed displayed in meters per second.
806     * @param {number} vertical_speed - The vertical speed.
807     */
808    setVerticalSpeed(vertical_speed){
809      this.vertical_speed = vertical_speed;
810      this.showVerticalSpeed();
811    }
812
813    /**
814     * Dynamically updates the displayed pitch angle.
815     */
816    showPitch() {
817      if(this.pitch != this.last_pitch) {
818        var obj = this;
819        this.win.requestAnimationFrame(function() {
820          obj.pitch_y = (obj.pitch_lines_dy/obj.pitch_value_step*obj.pitch);
821          obj.svg_horizon.attr('transform', 'translate(0,'+obj.pitch_y+') rotate('
                 +(-obj.roll)+',0,'+(-obj.pitch_y)+')');
822          obj.svg_roll_disc.attr('transform',' rotate('+(-obj.roll)+')');
823          obj.svg_pitch_text.text(obj.pitch.toFixed(1));
824        });
825        this.last_pitch = this.pitch;
826      }
827    }
```

```
828
829   /**
830    * Dynamically updates the displayed roll angle.
831    */
832   showRoll(){
833     if(this.roll != this.last_roll) {
834       var obj = this;
835       this.win.requestAnimationFrame(function() {
836         obj.svg_horizon.attr('transform', 'translate(0,'+obj.pitch_y+') rotate('
              +(-obj.roll)+',0,'+(-obj.pitch_y)+')');
837         obj.svg_roll_disc.attr('transform', 'rotate('+(-obj.roll)+')');
838         obj.svg_roll_text.text(obj.roll.toFixed(1));
839       });
840       this.last_roll = this.roll;
841     }
842   }
843
844   /**
845    * Dynamically updates the displayed heading.
846    */
847   showHeading(){
848     if(this.heading != this.last_heading) {
849       var obj = this;
850       this.win.requestAnimationFrame(function() {
851         obj.svg_compass_disk.attr('transform', 'rotate('+(-obj.heading)+')');
852         obj.svg_compass_text.text(obj.heading.toFixed(0));
853       });
854       this.last_heading = this.heading;
855     }
856   }
857
858   /**
859    * Dynamically updates the displayed airspeed.
860    */
861   showAirspeed(){
862     if(this.airspeed != this.last_airspeed) {
863       var obj = this;
864       this.win.requestAnimationFrame(function() {
865         var k = (obj.airspeed % obj.airspeed_value_step)/obj.airspeed_value_step
              ;
866         obj.svg_airspeed_ticks_group.attr('transform', 'translate(0,'+(obj.
              airspeed_dy*k)+')');
867         obj.svg_airspeed_text.text(obj.airspeed.toFixed(0));
868       });
869       this.last_airspeed = this.airspeed;
870     }
871   }
872
873   /**
874    * Dynamically updates the displayed ground speed.
875    */
876   showGroundspeed(){
877     if(this.groundspeed != this.last_groundspeed) {
878       var obj = this;
879       this.win.requestAnimationFrame(function() {
880         obj.svg_groundspeed_text.text(obj.groundspeed.toFixed(0));
881       });
882       this.last_groundspeed = this.groundspeed;
883     }
```

```
884     }
885
886     /**
887      * Dynamically updates the displayed altitude.
888      */
889     showAltitude(){
890       if(this.altitude != this.last_altitude) {
891         var obj = this;
892         this.win.requestAnimationFrame(function() {
893           var k = (obj.altitude % obj.altitude_value_step)/obj.altitude_value_step
                ;
894           obj.svg_altitude_ticks_group.attr('transform', 'translate(0,'+(obj.
                altitude_dy*k)+')');
895           obj.svg_altitude_text.text(obj.altitude.toFixed(0));
896         });
897         this.last_altitude = this.altitude;
898       }
899     }
900
901     /**
902      * Dynamically updates the displayed vertical speed.
903      */
904     showVerticalSpeed(){
905       if(this.vertical_speed != this.last_vertical_speed) {
906         var obj = this;
907         this.win.requestAnimationFrame(function() {
908           obj.svg_vertical_speed_indicator
909             .attr("points", obj.overlay_dx+",0 "+obj.vertical_speed_width+",0 "+
                    obj.overlay_dx+","+(-(obj.vertical_speed_dy*obj.vertical_speed/obj.
                    vertical_speed_value_step))+"");
910           obj.svg_vertical_speed_text.text(obj.vertical_speed.toFixed(0));
911         });
912         this.last_vertical_speed = this.vertical_speed;
913       }
914     }
915
916     /**
917      * Resets all flight data to default values.
918      */
919     reset() {
920       this.setPitch(0);
921       this.setRoll(0);
922       this.setHeading(0);
923       this.setAirspeed(0);
924       this.setGroundspeed(0);
925       this.setAltitude(0);
926       this.setVerticalSpeed(0);
927     }
928 }
929
930 exports.PFD = PFD;
```

Listing Z.3 - `PFD.js`

Datoteka `package.json` (listing Z.4), sadrži osnovne informacije o paketu i njegovoj instalaciji, uključujući definisanje drugih potrebnih paketa. U listingu Z.5 je data glavna skripta modula `index.js` koja zapravo samo eksportuje modul `node-mavlink`. Na ovaj način se zapravo omogućava lokalno instaliranje ovog modula i njegovo dalje korišćenje u programu JSLAB.

```
{
  "name": "node-mavlink-wrapper",
  "version": "2.0.7",
  "main": "index.js",
  "dependencies": {
    "node-mavlink": "2.0.7"
  }
}
```

Listing Z.4 - `node-mavlink-wrapper package.json`

```
const mavlink = require('node-mavlink');
module.exports = mavlink;
```

Listing Z.5 - `node-mavlink-wrapper index.js`

AA

Skripta ServerRTSP.jsl

U listingu AA.1 je data skripta `serverRTSP.jsl` za pokretanje `RTSP` servera u programu `JSLAB`. Ova skripta koristi modul `server-rtsp-live555`.

```
/**
 * @file Server RTSP
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var install_module = true;
var capture_type = ['webcam', 'desktop'][0];
var device_id = '5
    ef5e0fd960336959474ed288e44c7b50c8c4c364f38ca6256c6f79b3ef9f29b';
var width = 1280;
var height = 720;
var port = 8553;
var stream_name = 'h264';
var video_fps = 30;
var description = 'Session streamed by "setverRTSP.jsl"';
// --------------------

//getWebcams();
//getDesktopSources();

if(install_module) {
  disp('Installing server-rtsp-live555...');
  installModule(pwd + 'server-rtsp-live555');
}

var { ServerRTSPLive555 } = require(pwd +
  'server-rtsp-live555/build/Release/server_rtsp_live555');
var server = new ServerRTSPLive555();

var first = false;
var ready = false;

function frameCallback(data, width, height) {
  if(!first) {
    first = true;
```

```
40      this.server.start(port, stream_name, description, width, height, video_fps,
41          function(url) {
42        disp('RTSP started at: '+url);
43        ready = true;
44      }, function(err) {
45        disp('RTSP server error: '+err);
46      });
47    }
48    if(ready) {
49      server.setFrame(data);
50    }
51  }
52
53  if(capture_type == 'webcam') {
54    webcamCapture({
55      id: device_id,
56      width: { exact: width },
57      height: { exact: height }
58    }, frameCallback);
59  } else if(capture_type == 'desktop') {
60    desktopCapture({ id: device_id }, frameCallback);
61  }
62
63  addForCleanup(this, function() {
64    server.stop();
65  });
66
67  var win = await openWindowBlank();
68  win.setSize(500, 90);
69  win.setTitle('Timestamp');
70  win.document.body.innerHTML += '<div id="timestamp" style="padding: 10px; font-
      size: 36px; text-align: center;">0</div>';
71  var timestamp_cont = win.document.getElementById('timestamp');
72
73  var t = tic;
74  function frame(timestamp) {
75    var timestamp = stringify(round(tocms(t), 0));
76    timestamp_cont.innerText = timestamp;
77    requestAnimationFrame(frame);
78  }
79  requestAnimationFrame(frame);
```

Listing AA.1 - `serverRTSP.jsl`

AA.1 Modul server-rtsp-live555

Da bi se kompajlirao ovaj modul neophodno je da su dostupne biblioteke `FFmpeg` i `LIVE555`. Biblioteku `FFmpeg` (u dinamičkom obliku) je moguće preuzeti sa:

`https://github.com/BtbN/FFmpeg-Builds/releases`

Datoteka `package.json` (listing AA.2), sadrži osnovne informacije o paketu i njegovoj instalaciji, uključujući definisanje drugih potrebnih paketa.

```
1  {
2    "dependencies": {
3      "node-addon-api": "8.2.1"
4    }
```

```
}
```

Listing AA.2 - `server-rtsp-live555 package.json`

Datoteka `binding.gyp` (listing AA.3) pruža konfiguraciju za `node-gyp`, alat koji omogućava kompajliranje `C++` koda u binarni modul kompatibilan sa `Node.js` okruženjem.

```
{
  "targets": [
    {
      "target_name": "server_rtsp_live555",
      "sources": [
        "server-rtsp-live555.cpp"
      ],
      "include_dirs": [
        "<(module_root_dir)",
        "<!@(node -p \"require('node-addon-api').include\")",
        "<(module_root_dir)/ffmpeg-6.0/include",
        "<(module_root_dir)/live555-2023.07.24/liveMedia/include",
        "<(module_root_dir)/live555-2023.07.24/BasicUsageEnvironment/include",
        "<(module_root_dir)/live555-2023.07.24/groupsock/include",
        "<(module_root_dir)/live555-2023.07.24/UsageEnvironment/include"
      ],
      "libraries": [
        "<(module_root_dir)/ffmpeg-6.0/lib/avcodec.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avdevice.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avformat.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avfilter.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avutil.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/swscale.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/swresample.lib",
        "<(module_root_dir)/live555-2023.07.24/liveMedia/libliveMedia.lib",
        "<(module_root_dir)/live555-2023.07.24/BasicUsageEnvironment/
           libBasicUsageEnvironment.lib",
        "<(module_root_dir)/live555-2023.07.24/groupsock/libgroupsock.lib",
        "<(module_root_dir)/live555-2023.07.24/UsageEnvironment/
           libUsageEnvironment.lib",
        "ws2_32.lib"
      ],
      "cflags!": [
        "-fno-exceptions"
      ],
      "cflags_cc!": [
        "-fno-exceptions"
      ],
      "cflags": [
        "-O3"
      ],
      "cflags_cc": [
        "-O3"
      ],
      "defines": [
        "NAPI_DISABLE_CPP_EXCEPTIONS"
      ],
      "copies": [
        {
          "destination": "<(module_root_dir)/build/Release",
          "files": [
            "<(module_root_dir)/ffmpeg-6.0/bin/avcodec-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avdevice-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avfilter-9.dll",
```

```
53              "<(module_root_dir)/ffmpeg-6.0/bin/avformat-60.dll",
54              "<(module_root_dir)/ffmpeg-6.0/bin/avutil-58.dll",
55              "<(module_root_dir)/ffmpeg-6.0/bin/swresample-4.dll",
56              "<(module_root_dir)/ffmpeg-6.0/bin/swscale-7.dll",
57              "<(module_root_dir)/live555-2023.07.24/liveMedia/libliveMedia.lib",
58              "<(module_root_dir)/live555-2023.07.24/BasicUsageEnvironment/
                    libBasicUsageEnvironment.lib",
59              "<(module_root_dir)/live555-2023.07.24/groupsock/libgroupsock.lib",
60              "<(module_root_dir)/live555-2023.07.24/UsageEnvironment/
                    libUsageEnvironment.lib"
61            ]
62          }
63        ],
64        "msvs_settings": {
65          "VCCLCompilerTool": {
66            "AdditionalOptions": [
67              "-std:c++20"
68            ]
69          }
70        }
71      }
72    ]
73  }
```

Listing AA.3 - `server-rtsp-live555 binding.gyp`

Implementacija modula `server-rtsp-live555` je data u listinzima AA.4 i AA.5.

```
1  // ServerRTSPLive555 - server-rtsp-live555.h
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef SERVER_RTSP_LIVE555_H
8  #define SERVER_RTSP_LIVE555_H
9
10 //#define DEBUG_SERVER_RTSP_LIVE555
11 //#define PROFILE_SERVER_RTSP_LIVE555
12
13 #define DEBUG_SERVER_RTSP_LIVE555_LEVEL 1
14
15 #define NO_OPENSSL 1
16 #define NO_GETIFADDRS 1
17
18 #include <napi.h>
19 #include <chrono>
20 #include <thread>
21 #include <ctime>
22 #include <iomanip>
23 #include <sstream>
24 #include <string>
25 #include <mutex>
26 #include <iostream>
27
28 #include <liveMedia.hh>
29 #include <BasicUsageEnvironment.hh>
30
31 // FFMPEG LGPL librray from: https://github.com/BtbN/FFmpeg-Builds/releases
32 extern "C" {
33   #include "libavformat/avformat.h"
```

```
34    #include "libavcodec/avcodec.h"
35    #include "libswscale/swscale.h"
36    #include "libavdevice/avdevice.h"
37    #include "libavutil/time.h"
38  }
39
40  namespace server_rtsp_live555_ns {
41
42  using namespace std;
43  using namespace std::chrono;
44
45  class ServerRTSPLive555 : public Napi::ObjectWrap<ServerRTSPLive555> {
46   public:
47    static Napi::Object Init(Napi::Env env, Napi::Object exports);
48    ServerRTSPLive555(const Napi::CallbackInfo& info);
49    ~ServerRTSPLive555();
50
51    void serverThreadEntry();
52    void encoderThreadEntry();
53
54    Napi::Function onstart_cb;
55    Napi::Function error_cb;
56
57    Napi::ThreadSafeFunction tsfn_onstart;
58    Napi::ThreadSafeFunction tsfn_error;
59
60    Napi::Value start(const Napi::CallbackInfo& info);
61    void stop(const Napi::CallbackInfo& info);
62    void setFrame(const Napi::CallbackInfo& info);
63    void resetNewFrame();
64    bool checkNewFrame();
65    void setFramePointer(uint8_t*);
66
67    void setError(uint8_t);
68
69    uint8_t* frame_pointer = NULL;
70    bool new_frame = false;
71
72    bool active = false;
73    volatile char cmd_stop_thread = 0;
74    uint8_t error = 0;
75
76    uint32_t port = 554;
77    std::string stream_name = "h264";
78    std::string description = "Session streamed by \"ServerRTSPLive555\"";
79
80    uint32_t frame_width = 1280;
81    uint32_t frame_height = 720;
82    uint32_t video_fps = 30;
83    uint32_t num_bytes;
84
85    std::thread server_thread;
86
87    int32_t debug_level = DEBUG_SERVER_RTSP_LIVE555_LEVEL;
88
89    std::string input;
90  };
91
92  class FfmpegH264LiveServerMediaSubsession:OnDemandServerMediaSubsession {
```

```
93    public:
94      static ServerMediaSubsession* createNew(UsageEnvironment &env,
           ServerRTSPLive555* server_in);
95
96      ServerRTSPLive555* server;
97    protected:
98      FfmpegH264LiveServerMediaSubsession(ServerRTSPLive555* server_in,
99        UsageEnvironment &env, portNumBits initialPortNum = 6970,
100       Boolean multiplexRTCPWithRTP = False);
101     ~FfmpegH264LiveServerMediaSubsession();
102
103   protected:
104     // Override
105     virtual FramedSource* createNewStreamSource(unsigned clientSessionId,
106       unsigned& estBitrate) override;
107
108     virtual RTPSink* createNewRTPSink(Groupsock* rtpGroupsock,
109       unsigned char rtpPayloadTypeIfDynamic, FramedSource* inputSource) override
             ;
110 };
111
112 class Capturer {
113   public:
114     Capturer(ServerRTSPLive555* server_in);
115     ~Capturer();
116
117     ServerRTSPLive555* server;
118     AVFrame* frame;
119     uint32_t num_bytes;
120
121     AVFrame* Capture();
122 };
123
124 class FfmpegEncoder {
125   const AVCodec* encoder;
126   AVCodecContext* encode_context;
127   SwsContext* sws_context = nullptr;
128   ServerRTSPLive555* server;
129   AVFrame* frame_yuv420p;
130
131   public:
132     FfmpegEncoder(ServerRTSPLive555* server_in);
133     int CreateEncoder();
134     int Encode(AVFrame* frame, AVPacket* packet);
135     ~FfmpegEncoder();
136 };
137
138 class FFmpegH264Source : public FramedSource {
139   public:
140     static FFmpegH264Source* createNew(UsageEnvironment& env, ServerRTSPLive555*
          server_in);
141     FFmpegH264Source(UsageEnvironment& env, ServerRTSPLive555* server_in);
142     ~FFmpegH264Source();
143
144     FfmpegEncoder* encoder;
145     Capturer* capturer;
146     ServerRTSPLive555* server;
147   protected:
148     unsigned maxFrameSize() const override;
```

```
149   private:
150
151     virtual void doGetNextFrame();
152     virtual void doStopGettingFrames();
153
154     void FfmpegGetNextFrame();
155
156   private:
157     EventTriggerId m_eventTriggerId;
158
159 };
160
161 }// namespace server_rtsp_live555_ns
162
163 #endif // SERVER_RTSP_LIVE555_H
```

Listing AA.4 - server-rtsp-live555.h

```
1  // ServerRTSPLive555 - server-rtsp-live555.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "server-rtsp-live555.h"
8
9  std::mutex mtx; // mutex for critical section
10
11 namespace server_rtsp_live555_ns {
12
13 // Function to start the timer and return the start time
14 time_point<steady_clock> tic() {
15     return steady_clock::now();
16 }
17
18 // Function to stop the timer and return the elapsed time
19 long toc(const time_point<steady_clock>& startTime) {
20     return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
21 }
22
23 // Function to get current time
24 std::string getCurrentTime() {
25   auto now = system_clock::now();
26   auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
27   auto timer = system_clock::to_time_t(now);
28   std::tm bt = *std::localtime(&timer);
29   std::ostringstream oss;
30   oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
31   oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
32   return oss.str();
33 }
34
35 // Function to console log data
36 int consoleLog(int8_t level, const char* format, ...) {
37 #ifdef DEBUG_SERVER_RTSP_LIVE555_LEVEL
38   if(level <= DEBUG_SERVER_RTSP_LIVE555_LEVEL) {
39     printf("\033[0;33m[%s ServerRTSPLive555]\033[0m ", getCurrentTime().c_str())
       ;
40     va_list vl;
41     va_start(vl, format);
```

```
42      auto ret = vprintf(format, vl);
43      va_end(vl);
44      printf("\n");
45      return ret;
46    }
47  #endif
48    return 0;
49  }
50
51  // ServerRTSPLive555()
52  // Object ServerRTSPLive555
53  // --------------------
54  ServerRTSPLive555::ServerRTSPLive555(const Napi::CallbackInfo& info)
55      : Napi::ObjectWrap<ServerRTSPLive555>(info) {
56  #ifdef DEBUG_SERVER_RTSP_LIVE555
57    consoleLog(0, "Called constructor");
58  #endif
59
60    BOOL success;
61
62    // Get the handle to the current process
63    HANDLE hProcess = GetCurrentProcess();
64
65    // Set the priority class of the process
66    success = SetPriorityClass(hProcess, REALTIME_PRIORITY_CLASS);
67  #ifdef DEBUG_SERVER_RTSP_LIVE555
68    if(!success) {
69      consoleLog(0, "Failed to set process priority.");
70    }
71  #endif
72
73     // Get the handle to the current thread
74    HANDLE hThread = GetCurrentThread();
75
76    // Set the priority class of the process
77    success = SetThreadPriority(hThread, THREAD_PRIORITY_TIME_CRITICAL);
78  #ifdef DEBUG_SERVER_RTSP_LIVE555
79    if(!success) {
80      consoleLog(0, "Failed to set thread priority.");
81    }
82  #endif
83  }
84
85  // ServerRTSPLive555()
86  // Object destructor
87  // --------------------
88  ServerRTSPLive555::~ServerRTSPLive555() {
89
90  }
91
92  // Init() function
93  // --------------------
94  Napi::Object ServerRTSPLive555::Init(Napi::Env env, Napi::Object exports) {
95    Napi::Function func =
96        DefineClass(env,
97                    "ServerRTSPLive555",
98                     {InstanceMethod("start", &ServerRTSPLive555::start),
99                      InstanceMethod("stop", &ServerRTSPLive555::stop),
100                     InstanceMethod("setFrame", &ServerRTSPLive555::setFrame)
```

```
101                   });
102
103   Napi::FunctionReference* constructor = new Napi::FunctionReference();
104   *constructor = Napi::Persistent(func);
105   env.SetInstanceData(constructor);
106
107   exports.Set("ServerRTSPLive555", func);
108   return exports;
109 }
110
111 // start() function
112 // --------------------
113 Napi::Value ServerRTSPLive555::start(const Napi::CallbackInfo& info) {
114 #ifdef DEBUG_SERVER_RTSP_LIVE555
115   consoleLog(0, "Call to start()");
116 #endif
117
118   if(cmd_stop_thread) {
119 #ifdef DEBUG_SERVER_RTSP_LIVE555
120     consoleLog(0, "Previous thread not yet stopped.");
121 #endif
122     return Napi::Boolean::New(info.Env(), false);
123   }
124
125   port = info[0].As<Napi::Number>().Uint32Value();
126   stream_name = info[1].As<Napi::String>();
127   description = info[2].As<Napi::String>();
128   frame_width = info[3].As<Napi::Number>().Uint32Value();
129   frame_height = info[4].As<Napi::Number>().Uint32Value();
130   video_fps = info[5].As<Napi::Number>().Uint32Value();
131   onstart_cb = info[6].As<Napi::Function>();
132   error_cb = info[7].As<Napi::Function>();
133   num_bytes = frame_width * frame_height * 4;
134
135   if(active) {
136     stop(info);
137   }
138
139   tsfn_onstart = Napi::ThreadSafeFunction::New(
140     info.Env(),
141     onstart_cb,
142     "ThreadSafeFunctionOnStart",
143     0,
144     2
145   );
146
147   tsfn_error = Napi::ThreadSafeFunction::New(
148     info.Env(),
149     error_cb,
150     "ThreadSafeFunctionError",
151     0,
152     2
153   );
154
155   mtx.lock();
156   error = 0;
157   active = true;
158   cmd_stop_thread = 0;
159   mtx.unlock();
```

```
160
161 #ifdef DEBUG_SERVER_RTSP_LIVE555
162   consoleLog(0, "Creating new threads");
163 #endif
164   server_thread = std::thread(&ServerRTSPLive555::serverThreadEntry, this);
165
166   return Napi::Boolean::New(info.Env(), true);
167 }
168
169 // stop() function
170 // --------------------
171 void ServerRTSPLive555::stop(const Napi::CallbackInfo& info) {
172   if(active) {
173     mtx.lock();
174     cmd_stop_thread = 1;
175     mtx.unlock();
176     if(server_thread.joinable()) {
177       server_thread.join();  // Wait for the thread to finish
178     }
179     mtx.lock();
180       cmd_stop_thread = 0;
181       active = false;
182     mtx.unlock();
183
184 #ifdef DEBUG_SERVER_RTSP_LIVE555
185     consoleLog(0, "Thread stoped!");
186 #endif
187   }
188 }
189
190 // setError() function
191 // --------------------
192 void ServerRTSPLive555::setError(uint8_t error_in) {
193 #ifdef DEBUG_SERVER_RTSP_LIVE555
194   consoleLog(2, "Call to setError()");
195 #endif
196
197   mtx.lock();
198   error = error_in;
199   mtx.unlock();
200 }
201
202 // setFramePointer() function
203 // --------------------
204 void ServerRTSPLive555::setFramePointer(uint8_t* data) {
205 #ifdef DEBUG_SERVER_RTSP_LIVE555
206   consoleLog(0, "Call to setFramePointer()");
207 #endif
208
209   mtx.lock();
210   frame_pointer = data;
211   mtx.unlock();
212 }
213
214 // setFrame() function
215 // --------------------
216 void ServerRTSPLive555::setFrame(const Napi::CallbackInfo& info) {
217 #ifdef DEBUG_SERVER_RTSP_LIVE555
218   consoleLog(1, "Call to setFrame()");
```

```
219 #endif
220
221   consoleLog(-1, "New frame");
222   if(!new_frame && frame_pointer != NULL) {
223     mtx.lock();
224
225     Napi::ArrayBuffer frame_data = info[0].As<Napi::ArrayBuffer>();
226     if(frame_data.ByteLength() == num_bytes) {
227       memcpy(frame_pointer, reinterpret_cast<uint8_t*>(frame_data.Data()),
              num_bytes);
228       new_frame = true;
229     }
230     mtx.unlock();
231   }
232 }
233
234 // checkNewFrame() function
235 // --------------------
236 bool ServerRTSPLive555::checkNewFrame() {
237 #ifdef DEBUG_SERVER_RTSP_LIVE555
238   consoleLog(2, "Call to checkNewFrame()");
239 #endif
240
241   return new_frame;
242 }
243
244 // resetNewFrame() function
245 // --------------------
246 void ServerRTSPLive555::resetNewFrame() {
247 #ifdef DEBUG_SERVER_RTSP_LIVE555
248   consoleLog(1, "Call to resetNewFrame()");
249 #endif
250
251   mtx.lock();
252   new_frame = false;
253   mtx.unlock();
254 }
255
256 // serverThreadEntry() function
257 // --------------------
258 void ServerRTSPLive555::serverThreadEntry() {
259 #ifdef DEBUG_SERVER_RTSP_LIVE555
260   consoleLog(0, "Call to serverThreadEntry()");
261 #endif
262
263   // Begin by setting up our usage environment:
264   TaskScheduler* scheduler = BasicTaskScheduler::createNew();
265   UsageEnvironment* env = BasicUsageEnvironment::createNew(*scheduler);
266
267   ServerMediaSession* sms;
268
269   // Create the RTSP server:
270   RTSPServer* rtspServer = RTSPServer::createNew(*env, port, NULL);
271   if(rtspServer == NULL) {
272     setError(1);
273   } else {
274     sms = ServerMediaSession::createNew(*env, stream_name.c_str(), stream_name.
          c_str(), description.c_str());
275     sms->addSubsession(FfmpegH264LiveServerMediaSubsession::createNew(*env, this
```

```
        ));
276     rtspServer->addServerMediaSession(sms);
277
278     // Call onstart callback
279     char* url = rtspServer->rtspURL(sms);
280 #ifdef DEBUG_SERVER_RTSP_LIVE555
281     consoleLog(0, "Stream %s active, play this stream using the URL: %s",
            stream_name, url);
282 #endif
283
284     auto callback = [url](Napi::Env env, Napi::Function jsCallback) {
285       jsCallback.Call({Napi::String::New(env, url)});
286     };
287     tsfn_onstart.BlockingCall(callback);
288
289     env->taskScheduler().doEventLoop(&cmd_stop_thread);
290   }
291
292   setFramePointer(NULL);
293
294   // On error
295   if(error > 0) {
296     std::string message;
297     switch(error) {
298       case 1:
299         message = "Failed to create RTSP server: " + std::string(env->
                getResultMsg());
300         break;
301     }
302
303 #ifdef DEBUG_SERVER_RTSP_LIVE555
304     consoleLog(0, "On error:");
305     consoleLog(0, " -> code = %d", error);
306     consoleLog(0, " -> msg = %s", message.c_str());
307 #endif
308
309     uint8_t code = error;
310     auto callback = [code, message](Napi::Env env, Napi::Function jsCallback) {
311       jsCallback.Call({Napi::Number::New(env, code), Napi::String::New(env,
              message)});
312     };
313     tsfn_error.BlockingCall(callback);
314   }
315
316   // Clean up resources
317 #ifdef DEBUG_SERVER_RTSP_LIVE555
318   consoleLog(0, "Cleaning up resources!");
319 #endif
320
321   // close RTSP client
322   Medium::close(rtspServer);
323
324   // free environment and scheduler
325   env->reclaim();
326   delete scheduler;
327
328   tsfn_onstart.Release();
329   tsfn_error.Release();
330
```

```
331 #ifdef DEBUG_SERVER_RTSP_LIVE555
332   consoleLog(0, "Server thread function end!");
333 #endif
334 }
335
336 FfmpegH264LiveServerMediaSubsession::FfmpegH264LiveServerMediaSubsession(
      ServerRTSPLive555* server_in, UsageEnvironment &env, portNumBits
      initialPortNum,
337      Boolean multiplexRTCPWithRTP):
338        OnDemandServerMediaSubsession(env, true, initialPortNum,
             multiplexRTCPWithRTP),
339        server(server_in)
340 {
341 #ifdef DEBUG_SERVER_RTSP_LIVE555
342   consoleLog(0, "Call to FfmpegH264LiveServerMediaSubsession()");
343 #endif
344 }
345
346 ServerMediaSubsession* FfmpegH264LiveServerMediaSubsession::createNew(
      UsageEnvironment &env, ServerRTSPLive555* server_in) {
347 #ifdef DEBUG_SERVER_RTSP_LIVE555
348   consoleLog(0, "Call to FfmpegH264LiveServerMediaSubsession::createNew()");
349 #endif
350   return new FfmpegH264LiveServerMediaSubsession(server_in, env);
351 }
352
353 FfmpegH264LiveServerMediaSubsession::~FfmpegH264LiveServerMediaSubsession() {
354 #ifdef DEBUG_SERVER_RTSP_LIVE555
355   consoleLog(0, "Call to ~FfmpegH264LiveServerMediaSubsession()");
356 #endif
357 }
358
359 // Override
360 FramedSource* FfmpegH264LiveServerMediaSubsession::createNewStreamSource(
      unsigned clientSessionId, unsigned& estBitrate) {
361 #ifdef DEBUG_SERVER_RTSP_LIVE555
362   consoleLog(0, "Call to FfmpegH264LiveServerMediaSubsession::
        createNewStreamSource()");
363 #endif
364   estBitrate = 1000; // kbps, estimate
365   FFmpegH264Source* ffmpegSource = FFmpegH264Source::createNew(envir(), server);
366
367   // Create a framer for the Video Elementary Stream:
368   return H264VideoStreamFramer::createNew(envir(), ffmpegSource);
369 }
370
371 // "estBitrate" is the stream's estimated bitrate, in kbps
372 RTPSink* FfmpegH264LiveServerMediaSubsession::createNewRTPSink(Groupsock*
      rtpGroupsock,
373      unsigned char rtpPayloadTypeIfDynamic,
374      FramedSource* inputSource) {
375 #ifdef DEBUG_SERVER_RTSP_LIVE555
376   consoleLog(0, "Call to FfmpegH264LiveServerMediaSubsession::createNewRTPSink()
        ");
377 #endif
378   return H264VideoRTPSink::createNew(envir(), rtpGroupsock,
        rtpPayloadTypeIfDynamic);
379 }
380
```

```
381 FFmpegH264Source::FFmpegH264Source(UsageEnvironment& env, ServerRTSPLive555*
        server_in) :
382        FramedSource(env),
383        server(server_in)
384 {
385 #ifdef DEBUG_SERVER_RTSP_LIVE555
386   consoleLog(0, "Call to FfmpegH264Source()");
387 #endif
388   OutPacketBuffer::maxSize = 150000;
389   encoder = new FfmpegEncoder(server_in);
390   capturer = new Capturer(server_in);
391 }
392
393 FFmpegH264Source* FFmpegH264Source::createNew(UsageEnvironment& env,
        ServerRTSPLive555* server_in) {
394 #ifdef DEBUG_SERVER_RTSP_LIVE555
395   consoleLog(0, "Call to FFmpegH264Source::createNew()");
396 #endif
397   return new FFmpegH264Source(env, server_in);
398 }
399
400 FFmpegH264Source::~FFmpegH264Source() {
401 #ifdef DEBUG_SERVER_RTSP_LIVE555
402   consoleLog(0, "Call to ~FfmpegH264Source()");
403 #endif
404 }
405
406 void FFmpegH264Source::doStopGettingFrames() {
407 #ifdef DEBUG_SERVER_RTSP_LIVE555
408   consoleLog(0, "Call to FmpegH264Source::doStopGettingFrames()");
409 #endif
410   FramedSource::doStopGettingFrames();
411 }
412
413 void FFmpegH264Source::doGetNextFrame() {
414 #ifdef DEBUG_SERVER_RTSP_LIVE555
415   consoleLog(1, "Call to FmpegH264Source::doGetNextFrame()");
416 #endif
417   FfmpegGetNextFrame();
418 }
419
420  unsigned FFmpegH264Source::maxFrameSize() const {
421 #ifdef DEBUG_SERVER_RTSP_LIVE555
422   consoleLog(1, "Call to FmpegH264Source::maxFrameSize()");
423 #endif
424   return 150000;
425 }
426
427 void FFmpegH264Source::FfmpegGetNextFrame() {
428 #ifdef DEBUG_SERVER_RTSP_LIVE555
429   consoleLog(1, "Call to FmpegH264Source::FfmpegGetNextFrame()");
430 #endif
431   if(!isCurrentlyAwaitingData()) return; // we're not ready for the data yet
432
433   AVFrame* frame = capturer->Capture();
434   if(frame == NULL) {
435 #ifdef DEBUG_SERVER_RTSP_LIVE555
436     consoleLog(1, "No new frame...");
437 #endif
```

```
438   } else {
439     AVPacket* packet = av_packet_alloc();
440     int ret = encoder->Encode(frame, packet);
441 
442     if(ret < 0) {
443 #ifdef DEBUG_SERVER_RTSP_LIVE555
444       consoleLog(1, "Encoded packet is null!");
445 #endif
446       fFrameSize = 0;
447     } else {
448       u_int8_t* newFrameDataStart = packet->data;
449       unsigned newFrameSize = packet->size;
450 #ifdef DEBUG_SERVER_RTSP_LIVE555
451       consoleLog(1, "Packet length: %d", newFrameSize);
452 #endif
453 
454       // Deliver the data here:
455       if(newFrameSize > fMaxSize) {
456         fFrameSize = fMaxSize;
457         fNumTruncatedBytes = newFrameSize - fMaxSize;
458       } else {
459         fFrameSize = newFrameSize;
460       }
461 
462 #ifdef DEBUG_SERVER_RTSP_LIVE555
463       consoleLog(1, "Sending: %d", fFrameSize);
464 #endif
465       memmove(fTo, newFrameDataStart, fFrameSize);
466 
467       fPresentationTime.tv_sec = av_gettime_relative() / 1000000;
468       fPresentationTime.tv_usec = 1000 * av_gettime_relative();
469     }
470     av_packet_free(&packet);
471   }
472   FramedSource::afterGetting(this);
473 }
474 
475 FfmpegEncoder::FfmpegEncoder(ServerRTSPLive555* server_in) :
476   server(server_in)
477 {
478 #ifdef DEBUG_SERVER_RTSP_LIVE555
479   consoleLog(0, "Call to FfmpegEncoder()");
480 #endif
481   CreateEncoder();
482 }
483 
484 int FfmpegEncoder::CreateEncoder() {
485 #ifdef DEBUG_SERVER_RTSP_LIVE555
486   consoleLog(0, "Call to FfmpegEncoder::CreateEncoder()");
487 #endif
488 
489   int ret = 0;
490   char a[AV_ERROR_MAX_STRING_SIZE] = { 0 };
491 
492   frame_yuv420p = av_frame_alloc();
493   frame_yuv420p->width = server->frame_width;
494   frame_yuv420p->height = server->frame_height;
495   frame_yuv420p->format = AV_PIX_FMT_YUV420P;
496   av_frame_get_buffer(frame_yuv420p, 0);
```

```
497
498   encoder = avcodec_find_encoder(AV_CODEC_ID_H264);
499   if(!encoder) {
500     return -1;
501   }
502
503   encode_context = avcodec_alloc_context3(encoder);
504   if(!encode_context) {
505     return -1;
506   }
507
508   encode_context->width = server->frame_width;
509   encode_context->height = server->frame_height;
510
511   encode_context->time_base.den = server->video_fps;
512   encode_context->time_base.num = 1;
513   encode_context->framerate.num = server->video_fps;
514   encode_context->framerate.den = 1;
515   encode_context->pix_fmt = (enum AVPixelFormat) AV_PIX_FMT_YUV420P;
516   encode_context->gop_size = server->video_fps;
517
518   AVDictionary* options = NULL;
519   av_dict_set(&options, "preset", "ultrafast", 0);
520   av_dict_set(&options, "tune", "zerolatency", 0);
521
522   ret = avcodec_open2(encode_context, encoder, &options);
523   if(ret != 0) {
524 #ifdef DEBUG_SERVER_RTSP_LIVE555
525     consoleLog(0, "Error opening encoder: %s", av_make_error_string(a,
          AV_ERROR_MAX_STRING_SIZE, ret));
526 #endif
527     return -1;
528   }
529   return 0;
530 }
531
532 int FfmpegEncoder::Encode(AVFrame* frame, AVPacket* packet) {
533 #ifdef DEBUG_SERVER_RTSP_LIVE555
534   consoleLog(1, "Call to FfmpegEncoder::Encode()");
535 #endif
536   int ret;
537   int got_packet = 0;
538   char a[AV_ERROR_MAX_STRING_SIZE] = { 0 };
539
540   if(frame->format != AV_PIX_FMT_YUV420P) {
541 #ifdef DEBUG_SERVER_RTSP_LIVE555
542     consoleLog(1, "Changing pix_fmt to YUV420P");
543 #endif
544
545     if(sws_context == nullptr) {
546       sws_context = sws_getContext(frame->width, frame->height, (enum
            AVPixelFormat)frame->format, frame->width, frame->height,
            AV_PIX_FMT_YUV420P, 0, NULL, NULL, NULL);
547     }
548     sws_scale(sws_context, frame->data, frame->linesize, 0, frame->height,
          frame_yuv420p->data, frame_yuv420p->linesize);
549     ret = avcodec_send_frame(encode_context, frame_yuv420p);
550   } else {
551     ret = avcodec_send_frame(encode_context, frame);
```

```
552     }
553
554     if(ret < 0) {
555 #ifdef DEBUG_SERVER_RTSP_LIVE555
556       consoleLog(1, "Error sending frame to encoder: %s", av_make_error_string(a,
              AV_ERROR_MAX_STRING_SIZE, ret));
557 #endif
558     } else {
559       ret = avcodec_receive_packet(encode_context, packet);
560       if(ret == AVERROR(EAGAIN) || ret == AVERROR_EOF) {
561         // EOF
562       } else if(ret < 0) {
563 #ifdef DEBUG_SERVER_RTSP_LIVE555
564         consoleLog(1, "Error receiving packet from encoder: %s",
                av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
565 #endif
566       } else {
567         got_packet = 1; // A packet was encoded successfully
568       }
569     }
570
571     if(!got_packet) {
572       return -1;
573     }
574     return 0;
575 }
576
577 FfmpegEncoder::~FfmpegEncoder() {
578 #ifdef DEBUG_SERVER_RTSP_LIVE555
579     consoleLog(0, "Call to ~FfmpegEncoder()");
580 #endif
581 }
582
583 Capturer::Capturer(ServerRTSPLive555* server_in) :
584     server(server_in)
585 {
586 #ifdef DEBUG_SERVER_RTSP_LIVE555
587     consoleLog(0, "Call to Capturer()");
588 #endif
589     frame = av_frame_alloc();
590     if(!frame) {
591       return;
592     }
593
594     frame->width = server->frame_width;
595     frame->height = server->frame_height;
596     frame->format = AV_PIX_FMT_RGBA; // 4 bytes per pixel: R, G, B, A
597     num_bytes = frame->width * frame->height * 4;
598
599     // Allocate buffer for the frame data
600     int ret = av_frame_get_buffer(frame, 0);
601     if(ret < 0) {
602       // Handle error
603       av_frame_free(&frame);
604       return;
605     }
606
607     server->setFramePointer(frame->data[0]);
608 }
```

```
609
610 AVFrame* Capturer::Capture() {
611 #ifdef DEBUG_SERVER_RTSP_LIVE555
612   consoleLog(1, "Call to Capturer::Capture()");
613 #endif
614   if(!frame) {
615     return NULL;
616   }
617
618   while(!server->checkNewFrame() && server->cmd_stop_thread == 0) {}
619
620   server->resetNewFrame();
621 #ifdef DEBUG_SERVER_RTSP_LIVE555
622   consoleLog(1, "Got new frame...");
623   consoleLog(1, "Frame data copied...");
624 #endif
625   return frame;
626 }
627
628 Capturer::~Capturer() {
629 #ifdef DEBUG_SERVER_RTSP_LIVE555
630   consoleLog(0, "Call to ~Capturer()");
631 #endif
632 }
633
634 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
635   return ServerRTSPLive555::Init(env, exports);
636 }
637
638 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
639
640 }  // namespace server_rtsp_live555_ns
```

Listing AA.5 - `server-rtsp-live555.cpp`

AA.2 Kompajliranje biblioteke LIVE555

U nastavku je objašnjeno kompajliranje biblioteke `LIVE555` na operativnom sistemu *Windows 10*. Izvorni kod biblioteke `LIVE555` je moguće preuzeti sa:

`http://www.live555.com/liveMedia/public/`

- Preuzeti `.tar.gz` fajl i otpakovati u folder `C:\dev\live`.
- Promeniti sadržaj fajla `win32config` sa sledećim sadržajem:

```
1 NODEBUG=1
2 TARGETOS = WINNT
3
4 UI_OPTS =       $(guilflags) $(guilibsdll)
5 CONSOLE_UI_OPTS =   $(conlflags) $(conlibsdll)
6 CPU=amd64
7
8 TOOLS   =       C:\Program Files\Microsoft Visual Studio\2022\Community\VC\
      Tools\MSVC\14.32.31326
9 COMPILE_OPTS =      $(INCLUDES) $(cdebug) $(cflags) $(cvarsdll) -I. /EHsc /
      O2 /MD /GS /D "WIN64" /Oy- /Oi /D "NDEBUG" -DNO_OPENSSL=1 -
      DNO_GETIFADDRS=1
```

```
C = c
C_COMPILER = "$(TOOLS)\bin\Hostx64\x64\cl"
C_FLAGS = $(COMPILE_OPTS)
CPP = cpp
CPLUSPLUS_COMPILER = $(C_COMPILER)
CPLUSPLUS_FLAGS = $(COMPILE_OPTS) /std:c++20
OBJ = obj
LINK = link -out:
LIBRARY_LINK = lib -out:
LINK_OPTS_0 = $(linkdebug) ws2_32.lib /NXCOMPAT
LIBRARY_LINK_OPTS =
LINK_OPTS =     $(LINK_OPTS_0) $(UI_OPTS)
CONSOLE_LINK_OPTS = $(LINK_OPTS_0) $(CONSOLE_UI_OPTS)
SERVICE_LINK_OPTS =     kernel32.lib advapi32.lib shell32.lib ws2_32.lib -
    subsystem:console,$(APPVER)
LIB_SUFFIX =        lib
LIBS_FOR_CONSOLE_APPLICATION =
LIBS_FOR_GUI_APPLICATION =
MULTIMEDIA_LIBS =   winmm.lib
EXE = .exe
PLATFORM = Windows

rc32 = rc.exe
.rc.res:
    $(rc32) $<
```

- Otvoriti `Command Prompt` i pokrenuti:

```
"C:\Program Files\Microsoft Visual Studio\2022\Community\Common7\Tools\
    VsDevCmd.bat"
```

- Nakon toga je moguće preći na kompajliranje sa:

```
cd C:\dev\live
genWindowsMakefiles

cd liveMedia
del *.obj *.lib
nmake /B -f liveMedia.mak
cd ..\groupsock
del *.obj *.lib
nmake /B -f groupsock.mak
cd ..\UsageEnvironment
del *.obj *.lib
nmake /B -f UsageEnvironment.mak
cd ..\BasicUsageEnvironment
del *.obj *.lib
nmake /B -f BasicUsageEnvironment.mak
cd ..
```

AB

Skripta ClientRTSP.jsl

U listingu AB.1 je data skripta `clientRTSP.jsl` za pokretanje `RTSP` klijenta u programu `JSLAB`. Ova skripta koristi module `decoder-ffmpeg`, `stream-decoder-ffmpeg` i `stream-decoder-gst`.

```
/**
 * @file Client RTSP
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
//close('all');
clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var install_module = false;
var demuxer = ["js", "ffmpeg", "gst"][2];
var decoder = ["js", "ffmpeg"][0];
var container = ["canvas", "video"][0];
var rtsp_url = [
  'rtsp://192.168.144.25:8554/main.264',
  'rtsp://192.168.15.100:8554/prdc_h264'
][1];
var stream_protocol = ['udp', 'tcp'][0];
var fps = 30;
var codec = ["avc1.42001f", "hev1.1.0.L150"][0];
var hardware_acceleration = ["no-preference", "prefer-hardware", "prefer-
    software"][0];
var optimize_for_latency = true;
var track = 'track1';
var codec_type = ['H264', 'HEVC'][0];
var pix_fmt = 'YUV420P';
var user_agent = 'ClientRTSP.jsl';
var video_width = 1280;
var video_height = 720;
var ffmpeg_options = {
  max_delay: "50000",
  fflags: "flush_packets+nobuffer",
  flush_packets: "1",
  dec_flags: "low_delay",
  dec_flags2: "fast+showall"
};
var gst_options = {
```

```
39    latency: "500",
40    jitter_latency: "500"
41  };
42  // --------------------
43
44  if(install_module) {
45    if(demuxer == "ffmpeg") {
46      disp('Installing stream-decoder-ffmpeg...');
47      installModule(pwd + 'stream-decoder-ffmpeg');
48    } else if(demuxer == "gst") {
49      disp('Installing stream-decoder-gst...');
50      installModule(pwd + 'stream-decoder-gst');
51    } else if(decoder == "ffmpeg") {
52      disp('Installing decoder-ffmpeg...');
53      installModule(pwd + 'decoder-ffmpeg');
54    }
55  }
56
57  var win = await openWindowBlank();
58  win.setTitle('RTSP stream');
59  await waitMSeconds(1000);
60
61  disp('Container: '+container);
62  if(container == 'canvas') {
63    win.document.body.innerHTML += '<canvas id="canvas"></canvas>';
64    var dom = win.document.getElementById('canvas');
65    var ctx = dom.getContext('2d');
66    dom.width = video_width;
67    dom.height = video_height;
68  } else if(container == 'video') {
69    win.document.body.innerHTML += '<video id="video"></video>';
70    var dom = win.document.getElementById('video');
71
72    var trackGenerator = new MediaStreamTrackGenerator({
73      kind: 'video'
74    });
75    var writer = trackGenerator.writable.getWriter();
76
77    var stream = new MediaStream();
78    stream.addTrack(trackGenerator);
79    dom.srcObject = stream;
80    dom.play();
81  }
82
83  Object.assign(dom.style, {
84    position: 'absolute',
85    top: '50%',
86    left: '50%',
87    transform: 'translate(-50%, -50%)',
88    width: '100%',
89    height: '100%',
90    objectFit: 'contain'
91  });
92
93  async function showFrame(data) {
94    if(container == "video") {
95      if(demuxer == "ffmpeg" || demuxer == "gst") {
96        var new_frame = new VideoFrame(data, {
97          timestamp: 0,
```

```
98          alpha: 'discard',
99          codedWidth: video_width,
100         codedHeight: video_height,
101         format: "RGBA"
102       });
103     } else if(decoder == "js") {
104       var new_frame = data;
105     }
106     await writer.ready;
107     await writer.write(new_frame);
108     await writer.ready;
109     new_frame.close();
110   } else if(container == "canvas") {
111     if(demuxer == "ffmpeg" || demuxer == "gst" || decoder == "ffmpeg") {
112       ctx.putImageData(new ImageData(new Uint8ClampedArray(data), video_width,
              video_height), 0, 0);
113     } else if(decoder == "js") {
114       ctx.drawImage(data, 0, 0, data.displayWidth, data.displayHeight);
115       data.close();
116     }
117   }
118 }
119
120 if(demuxer == "ffmpeg") {
121   disp('Demuxer: FFmpeg');
122   var { StreamDecoderFFmpeg } = require(pwd +
123   'stream-decoder-ffmpeg/build/Release/stream_decoder_ffmpeg');
124   var stream_decoder_ffmpeg = new StreamDecoderFFmpeg();
125
126   var options = ffmpeg_options || {};
127   options.id = 1;
128   options.input = rtsp_url;
129   options.input_format = 'rtsp';
130   options.video_size = video_width+"x"+video_height;
131   options.framerate = String(fps);
132
133   stream_decoder_ffmpeg.start(options,
134       function(data) {
135     showFrame(data);
136   }, function(code, message) {
137     disp('FFMPEG error ['+code+']: ' + message);
138   });
139
140 } else if(demuxer == "gst") {
141   disp('Demuxer: GStreamer');
142   process.env.GST_PLUGIN_PATH = 'C:/gstreamer/1.0/msvc_x86_64/lib/gstreamer-1.0'
      ;
143   var { StreamDecoderGST } = require(pwd +
144   'stream-decoder-gst/build/Release/stream_decoder_gst');
145   var stream_decoder_gst = new StreamDecoderGST();
146
147   var options = gst_options || {};
148   options.id = 1;
149   options.input = rtsp_url;
150   options.video_width = video_width;
151   options.video_height = video_height;
152   options.protocol = stream_protocol;
153   options.codec = codec_type;
154
```

```
155   stream_decoder_gst.start(options,
156       function(data) {
157     showFrame(data);
158   }, function(code, message) {
159     disp('GST error ['+code+']: ' + message);
160   });
161
162 } else if(demuxer == "js") {
163   disp('Demuxer: JavaScript');
164   var { RTSP_DEMUXER } = require(pwd + 'RTSP_DEMUXER.js');
165   var rtsp_demuxer = new RTSP_DEMUXER(user_agent, rtsp_url, track,
166     codec_type, stream_protocol);
167
168   if(decoder == "ffmpeg") {
169     disp('Decoder: FFmpeg');
170     var { DecoderFFmpeg } = require(pwd +
171     'decoder-ffmpeg/build/Release/decoder_ffmpeg');
172     var decoder_ffmpeg = new DecoderFFmpeg();
173     decoder_ffmpeg.configure({
174       frame_width: video_width,
175       frame_height: video_height,
176       framerate: fps,
177       pix_fmt_id: pix_fmt,
178       codec_id: codec_type
179     });
180   } else if(decoder == "js") {
181     disp('Decoder: JavaScript');
182     var { DECODER_JS } = require(pwd + 'DECODER_JS.js');
183     var decoder_js = new DECODER_JS({
184       codec: codec,
185       video_width: video_width,
186       video_height: video_height,
187       hardware_acceleration: hardware_acceleration,
188       optimize_for_latency: optimize_for_latency
189     }, function(data) {
190       showFrame(data);
191     });
192   }
193
194   rtsp_demuxer.start(function(data) {
195     if(decoder == "ffmpeg") {
196       var decoded_data = decoder_ffmpeg.decode(new Uint8Array(data).buffer);
197       if(hasKey(decoded_data, 'buffer')){
198         showFrame(decoded_data.buffer);
199       } else if(hasKey(decoded_data, 'error')){
200         disp('Decode error ['+decoded_data.code+']: ' + decoded_data.error);
201       }
202     } else if(decoder == "js") {
203       try {
204         decoder_js.decode(data);
205       } catch(err) {
206         disp('Decode error: ' + err);
207       }
208     }
209   });
210 }
211
212 addForCleanup(undefined, function() {
213   if(demuxer == 'ffmpeg') {
```

```
    stream_decoder_ffmpeg.stop();
  } else if(demuxer == 'gst') {
    stream_decoder_gst.stop();
  } else if(demuxer == 'js') {
    rtsp_demuxer.requestTeardown();
    if(decoder == "js") {
      decoder_js.close();
    }
  }
});

var win = await openWindowBlank();
win.setSize(500, 90);
win.setTitle('Timestamp');
win.document.body.innerHTML += '<div id="timestamp" style="padding: 10px; font-
    size: 36px; text-align: center;">0</div>';
var timestamp_cont = win.document.getElementById('timestamp');

var t = tic;
function frame(timestamp) {
  var timestamp = stringify(round(tocms(t), 0));
  timestamp_cont.innerText = timestamp;
  requestAnimationFrame(frame);
}
requestAnimationFrame(frame);
```

Listing AB.1 - `clientRTSP.jsl`

U listingu AB.2 je dat kod klase `RTSP_DEMUXER` za povezivanje na *RTSP* server i izdvajanje enkodiranih `H264` i `H265` video paketa iz `RTP` paketa.

```
/**
 * @file RTSP demuxer lib
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */

const net = require('net');
const dgram = require('dgram');

/**
 * Class for Flight control RTSP demuxer.
 */
class RTSP_DEMUXER {

  /**
   * Creates an instance of the RTSP demuxer with the provided configuration.
   * @param {string} user_agent - The user agent string to identify the client.
   * @param {string} rtsp_url - The RTSP URL of the stream.
   * @param {string} track - The track identifier for the stream.
   * @param {string} codec_type - The codec type used in the stream (e.g., H
       .264, H.265).
   * @param {string} stream_protocol - The protocol used for streaming (e.g.,
       TCP, UDP).
   */
  constructor(user_agent, rtsp_url, track, codec_type, stream_protocol) {
    // Variables
    var obj = this;
    this.rtsp_url = rtsp_url;
```

```
29      this.track = track;
30      this.codec_type = codec_type;
31      this.protocol = stream_protocol;
32
33      var regex = /^rtsp:\/\/([^/:]+):(\d+)/;
34      var match = rtsp_url.match(regex);
35      this.ip = match[1];
36      this.stream_port = match[2];
37
38      this.RTSP_DISCARD_PACKETS = true;
39      this.RTP_PACKET_SKIPPED_PRINT_MAX = 5;
40      this.RTSP_REORDER_QUEUE_SIZE = 30;
41      this.RTSP_MIN_PORT = 5000;
42      this.RTSP_MAX_PORT = 65000;
43
44      this.RTSP_MAX_DELAY = 150; // [ms]
45      this.RTSP_KEEPALIVE_PERIOD = 40000; // [ms]
46
47      this.UDP_BUFFER_SIZE = 100000000;
48
49      this.DEBUG = true;
50      this.DEBUG_INFO = false;
51      this.DEBUG_RTP_PACKETS_INFO = false;
52      this.DEBUG_RTP_PACKETS_MISSING = false;
53      this.DEBUG_TIMESTAMPS = false;
54      this.DEBUG_MARKER = false;
55
56      this.data_callbacks = {
57        1: this.requestOptions.bind(this),
58        2: this.requestDescribe.bind(this),
59        3: this.requestSetup.bind(this),
60        4: this.requestPlay.bind(this),
61      };
62
63      this.last_rtsp_id = undefined;
64      this.current_rtsp_id = 0;
65
66      this.rtsp_cseq = 0;
67
68      this.user_agent = user_agent;
69
70      this.nalu_start_code = [0, 0, 0, 1];
71      this.interleaved_header = 0x24;
72      this.max_packet_length = 8192;
73
74      this.rtp_packets_counter = 0;
75      this.nalu_packets_counter = {};
76      this.rtp_packets = {};
77      this.rtp_packet_ids = [];
78      this.rtp_packet_ids_max_size = 100;
79      this.rtp_packet_ids_i = -1;
80      this.rtp_packet_last_sequence_number = -1;
81      this.rtp_packet_last_decoded_id = undefined;
82      this.rtp_packet_decoding_id = undefined;
83      this.rtp_packets_buffered = 0;
84      this.rtp_packets_skipped = false;
85      this.rtp_packet_sequence_number_rollovers = 0;
86      this.rollover_diff = 1000;
87      this.got_header = false;
```

```
88      this.rtp_header;
89
90      this.rtp_packets_skipped_counter = 0;
91      this.rtp_packets_missed_counter = 0;
92
93      this.tcp_buffer = Buffer.alloc(0);
94
95      this.nal_unit = {
96        'payload': [],
97        'rtp_packets_id': []
98      };
99      this.nal_unit_num = 0;
100     this.last_used_port = this.RTSP_MIN_PORT-1;
101   }
102
103   /**
104    * Determines available RTP and RTCP ports for UDP communication.
105    * @param {Function} callback - A callback function to be executed once the
          ports are determined.
106    */
107   async getPorts() {
108     this.client_port_rtp = await findFirstUnusedPort(this.last_used_port+1, this
          .RTSP_MIN_PORT, this.RTSP_MAX_PORT);
109     this.last_used_port = this.client_port_rtp;
110     this.client_port_rtcp = await findFirstUnusedPort(this.last_used_port+1,
          this.RTSP_MIN_PORT, this.RTSP_MAX_PORT);
111   }
112
113   /**
114    * Starts the RTSP session and initializes RTP/RTCP servers or client as per
          configuration.
115    * @param {Function} packet_callback - A callback function to handle extracted
          NAL units from the RTP stream.
116    */
117   async start(packet_callback) {
118     this.packet_callback = packet_callback;
119     if(this.protocol == 'udp') {
120       await this.getPorts();
121       this.createServerUdpRTCP();
122       this.createServerUdpRTP();
123       this.createClientRTSP();
124     } else if(this.protocol == 'tcp') {
125       this.createClientRTSP();
126     }
127   }
128
129   /**
130    * Handles incoming data from the RTSP server, executing appropriate callbacks
          based on the current RTSP command ID.
131    * @param {Buffer} data - The raw data received from the RTSP server.
132    */
133   onData(data) {
134     if(data) {
135       this.dispIf(this.DEBUG, 'Response from RTSP server:\n'+data.toString());
136     }
137     this.current_rtsp_id += 1;
138     if(hasKey(this.data_callbacks, this.current_rtsp_id)) {
139       this.data_callbacks[this.current_rtsp_id](data);
140     }
```

```
141   }
142
143   /**
144    * Processes and handles RTP packets received via UDP.
145    * @param {Buffer} data - The raw RTP packet received.
146    */
147   depayDataUDP(data) {
148     if(data.length > this.max_packet_length) {
149       console.log('UDP packet length ' + data.length);
150     }
151     this.onNewPacketRTP(data);
152   }
153
154   /**
155    * Processes and handles RTP packets received via TCP.
156    * @param {Buffer} data - The raw RTP packet received.
157    */
158   depayDataTCP(data) {
159     this.tcp_buffer = Buffer.concat([this.tcp_buffer, data]);
160
161     // We get RTP, RTCP and RTSP packets together
162     // Interleaved RTP and RTCP packets are starting with [36, 0 or 1, p1, p2]
163     // where p1 and p2 form data length without this 4 header bytes
164     while(this.tcp_buffer.length > 0) {
165       if(this.tcp_buffer[0] == this.interleaved_header && [0, 1].includes(this.
            tcp_buffer[1])) {
166         // Interleaved header
167         var channel = this.tcp_buffer[1];
168         var packet_length = (this.tcp_buffer[2] << 8) | this.tcp_buffer[3];
169         if(this.tcp_buffer.length >= packet_length+4) {
170           var packet_data = this.tcp_buffer.slice(4, packet_length+4);
171           this.tcp_buffer = this.tcp_buffer.slice(packet_length+4);
172           if(channel == 0) {
173             // RTP
174             this.onNewPacketRTP(packet_data);
175           } else {
176             // RTCP
177           }
178         } else {
179           // No enough data
180           break;
181         }
182       } else if(this.tcp_buffer.slice(0, 8).toString() == 'RTSP/1.0') {
183         // RTSP data
184         // Check if there is interleaved header
185         var id = this.findInreleavedHeader(this.tcp_buffer);
186         var rtsp_data = this.tcp_buffer.slice(0, id);
187         this.tcp_buffer = this.tcp_buffer.slice(id);
188         this.onData(rtsp_data);
189       } else {
190         if(this.tcp_buffer.length >= 2) {
191           // Check if there is interleaved header and skip data
192           var id = this.findInreleavedHeader(this.tcp_buffer);
193           var rtsp_data = this.tcp_buffer.slice(0, id);
194           this.tcp_buffer = this.tcp_buffer.slice(id);
195           this.dispIf(this.DEBUG, 'Data from RTSP server skipped:\n'+rtsp_data.
                toString());
196         } else {
197           // No enough data
```

```
198             break;
199           }
200         }
201       }
202     }
203
204     /**
205      * Finds the position of the next interleaved header within a buffer.
206      * @param {Buffer} buffer - The buffer to search for an interleaved header.
207      * @returns {number} The position of the interleaved header or the buffer
           length if not found.
208      */
209     findInreleavedHeader(buffer) {
210       for(let i = 0, len = buffer.length - 1; i < len; ++i) {
211         if(buffer[i] === this.interleaved_header && (buffer[i + 1] === 0 || buffer
              [i + 1] === 1)) {
212           return i;
213         }
214       }
215       return buffer.length;
216     }
217
218     /**
219      * Initiates a keep-alive mechanism for the RTSP session to prevent timeouts.
220      */
221     startKeepAlive() {
222       var obj = this;
223       this.keep_alive_interval = setInterval(function() {
224         obj.rtsp_cseq += 1;
225         var request = 'GET_PARAMETER '+obj.rtsp_url+' RTSP/1.0\r\nCSeq: '+obj.
              rtsp_cseq+'\r\nUser-Agent: '+obj.user_agent+'\r\nSession: '+obj.
              session_id+'\r\n\r\n';
226
227         obj.dispIf(obj.DEBUG, 'Sent RTSP request: '+request);
228         obj.rtsp_client.write(request);
229       }, this.RTSP_KEEPALIVE_PERIOD);
230     }
231
232     /**
233      * Sends an RTSP OPTIONS request to the server to discover supported commands.
234      * @param {Buffer} data - The data received in response to the previous RTSP
           command.
235      */
236     requestOptions(data) {
237       this.rtsp_cseq += 1;
238       this.last_rtsp_id = 1;
239       var request = 'OPTIONS '+this.rtsp_url+' RTSP/1.0\r\nCSeq: '+this.rtsp_cseq+
            '\r\nUser-Agent: '+this.user_agent+'\r\n\r\n';
240
241       this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
242       this.rtsp_client.write(request);
243     }
244
245     /**
246      * Sends an RTSP DESCRIBE request to the server to retrieve session
           description.
247      * @param {Buffer} data - The data received in response to the previous RTSP
           command.
248      */
```

```
249   requestDescribe(data) {
250     this.rtsp_cseq += 1;
251     this.last_rtsp_id = 2;
252     var request = 'DESCRIBE '+this.rtsp_url+' RTSP/1.0\r\nCSeq: '+this.rtsp_cseq
          +'\r\nUser-Agent: '+this.user_agent+'\r\nAccept: application/sdp\r\n\r\n'
          ;
253
254     this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
255     this.rtsp_client.write(request);
256   }
257
258   /**
259    * Sends an RTSP SETUP request to the server to establish a session for an
          individual stream.
260    * @param {Buffer} data - The data received in response to the previous RTSP
          command.
261    */
262   requestSetup(data) {
263     var request;
264     this.rtsp_cseq += 1;
265     this.last_rtsp_id = 3;
266
267     if(this.protocol == 'udp') {
268       request = 'SETUP '+this.rtsp_url+'/'+this.track+' RTSP/1.0\r\nCSeq: '+this
            .rtsp_cseq+'\r\nUser-Agent: '+this.user_agent+'\r\nTransport: RTP/AVP/
            UDP;unicast;client_port='+this.client_port_rtp+'-'+this.
            client_port_rtcp+'\r\n\r\n';
269     } else if(this.protocol == 'tcp') {
270       request = 'SETUP '+this.rtsp_url+'/'+this.track+' RTSP/1.0\r\nCSeq: '+this
            .rtsp_cseq+'\r\nUser-Agent: '+this.user_agent+'\r\nTransport: RTP/AVP/
            TCP;unicast;interleaved=0-1\r\n\r\n';
271     }
272
273     this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
274     this.rtsp_client.write(request);
275   }
276
277   /**
278    * Sends an RTSP PLAY request to the server to start streaming data.
279    * @param {Buffer} data - The data received in response to the previous RTSP
          command.
280    */
281   requestPlay(data) {
282     this.rtsp_cseq += 1;
283     this.last_rtsp_id = 4;
284     var input_string;
285
286     input_string = data.toString();
287
288     // Extract server_port values
289     var server_port_match = input_string.match(/server_port=(\d+)-(\d+)/);
290
291     // Extract session value
292     var session_match = input_string.match(/Session: (\w+)(;|$)/);
293
294     // Check if matches are found
295     if(this.protocol == 'udp') {
296       if(server_port_match && server_port_match.length > 2) {
297         this.client_port_rtp = server_port_match[1];
```

```
298         this.client_port_rtcp = server_port_match[2];
299
300         this.dispIf(this.DEBUG, "Server Port RTP: "+this.client_port_rtp+",
               Server Port RTCP:"+this.client_port_rtcp);
301       } else {
302         this.dispIf(this.DEBUG, "Server Ports not found!");
303       }
304     }
305
306     if(session_match && session_match.length > 1) {
307       this.session_id = session_match[1];
308       this.dispIf(this.DEBUG, "Session: " + this.session_id);
309     } else {
310       this.dispIf(this.DEBUG, "Session value not found!");
311     }
312
313     var request = 'PLAY '+this.rtsp_url+' RTSP/1.0\r\nCSeq: '+this.rtsp_cseq+'\r
          \nUser-Agent: '+this.user_agent+'\r\nSession: '+this.session_id+'\r\
          nRange: npt=now-\r\n\r\n';
314
315     this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
316     this.rtsp_client.write(request);
317
318     this.startKeepAlive();
319   }
320
321   /**
322    * Sends an RTSP PAUSE request to the server to temporarily halt streaming
          data.
323    */
324   requestPause() {
325     this.rtsp_cseq += 1;
326     this.last_rtsp_id = 5;
327     var request = 'PAUSE '+this.rtsp_url+' RTSP/1.0\r\nCSeq: '+this.rtsp_cseq+'\
          r\nUser-Agent: '+this.user_agent+'\r\nSession: '+this.session_id+'\r\n\r\
          n';
328
329     this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
330     this.rtsp_client.write(request);
331   }
332
333   /**
334    * Sends an RTSP PLAY request to the server to resume streaming data after a
          pause.
335    */
336   requestResume() {
337     this.rtsp_cseq += 1;
338     this.last_rtsp_id = 6;
339     var request = 'PLAY '+this.rtsp_url+' RTSP/1.0\r\nCSeq: '+this.rtsp_cseq+'\r
          \nUser-Agent: '+this.user_agent+'\r\nSession: '+this.session_id+'\r\n\r\n
          ';
340
341     this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
342     this.rtsp_client.write(request);
343   }
344
345   /**
346    * Sends an RTSP TEARDOWN request to the server to terminate the session and
          close the connection.
```

```
347     */
348   requestTeardown() {
349     if(this.rtsp_client) {
350       this.rtsp_cseq += 1;
351       this.last_rtsp_id = 7;
352       var request = 'TEARDOWN '+this.rtsp_url+' RTSP/1.0\r\nCSeq: '+this.
              rtsp_cseq+'\r\nUser-Agent: '+this.user_agent+'\r\nSession: '+this.
              session_id+'\r\n\r\n';
353
354       this.dispIf(this.DEBUG, 'Sent RTSP request: '+request);
355       this.rtsp_client.write(request);
356       clearInterval(this.keep_alive_interval);
357     }
358
359     try {
360       this.rtsp_client.destroy();
361       delete this.rtsp_client;
362     } catch(err) {
363       this.dispIf(this.DEBUG, 'RTSP server close failed '+stringify(err));
364     }
365
366     if(this.protocol == 'udp') {
367       // Free ports
368       try {
369         this.rtp_server.close();
370         delete this.rtp_server;
371       } catch(err) {
372         this.dispIf(this.DEBUG, 'RTP server close failed '+stringify(err));
373       }
374
375       try {
376         this.rtcp_server.close();
377         delete this.rtcp_server;
378       } catch(err) {
379         this.dispIf(this.DEBUG, 'RTCP server close failed '+stringify(err));
380       }
381     }
382   }
383
384   /**
385    * Initializes the RTSP client for communication with the RTSP server.
386    */
387   createClientRTSP() {
388     var obj = this;
389
390     this.rtsp_client = net.createConnection({
391       host: this.ip,
392       port: this.stream_port,
393       keepAlive: true,
394       noDelay: false
395     }, function() {
396       obj.dispIf(obj.DEBUG, 'Connected to RTSP server.');
397
398       obj.onData();
399     });
400
401     this.rtsp_client.on('data', function(data) {
402       if(obj.protocol == 'udp') {
403         obj.onData(data);
```

```
404       } else if(obj.protocol == 'tcp') {
405         obj.depayDataTCP(data);
406       }
407     });
408
409     this.rtsp_client.on('end', function() {
410       obj.dispIf(obj.DEBUG, 'Disconnected from RTSP server.');
411     });
412
413     this.rtsp_client.on('error', function(err) {
414       obj.dispIf(obj.DEBUG, 'RTSP Error:' + stringify(err));
415       obj.rtsp_client.destroy();
416     });
417
418     this.rtsp_client.on('close', function() {
419       obj.dispIf(obj.DEBUG, 'RTSP connection closed.');
420     });
421
422   }
423
424   /**
425    * Initializes the UDP server for receiving RTP packets.
426    */
427   createServerUdpRTP() {
428     var obj = this;
429
430     this.rtp_server = dgram.createSocket('udp4');
431
432     // Listen for RTP packets
433     this.rtp_server.on('message', function(msg, rinfo) {
434       obj.dispIf(obj.DEBUG_INFO, 'Received RTP UDP packet from ' + rinfo.address
            + ' port ' + rinfo.port);
435
436       obj.depayDataUDP(msg);
437     });
438
439     this.rtp_server.on("listening", function() {
440       obj.rtp_server.setRecvBufferSize(obj.UDP_BUFFER_SIZE);
441       obj.rtp_server.setSendBufferSize(obj.UDP_BUFFER_SIZE);
442
443       obj.dispIf(obj.DEBUG, 'RTP UDP server listening. Buffers size: ['+obj.
            rtp_server.getRecvBufferSize()+', '+obj.rtp_server.getSendBufferSize()+
            ']');
444     });
445
446     // Handle errors on the RTP server
447     this.rtp_server.on('error', function(err) {
448       obj.dispIf(obj.DEBUG, 'RTP UDP Server Error:' + stringify(err));
449     });
450
451     // Handle closure of the RTP server
452     this.rtp_server.on('close', function() {
453       obj.dispIf(obj.DEBUG, 'RTP UDP Server closed.');
454     });
455
456     // Bind the UDP socket to the specified port
457     this.rtp_server.bind(this.client_port_rtp, function() {
458       obj.dispIf(obj.DEBUG, 'RTP UDP Server bind complete.');
459     });
```

```
460   }
461
462   /**
463    * Initializes the UDP server for receiving RTCP packets.
464    */
465   createServerUdpRTCP() {
466     var obj = this;
467     this.rtcp_server = dgram.createSocket('udp4');
468
469     // Listen for RTCP packets
470     this.rtcp_server.on('message', function(msg, rinfo) {
471       obj.dispIf(obj.DEBUG_INFO, 'Received UDP RTCP packet from ' + rinfo.
              address + ' port ' + rinfo.port);
472     });
473
474     this.rtcp_server.on("listening", function() {
475       obj.rtcp_server.setRecvBufferSize(obj.UDP_BUFFER_SIZE);
476       obj.rtcp_server.setSendBufferSize(obj.UDP_BUFFER_SIZE);
477
478       obj.dispIf(obj.DEBUG, 'RTCP UDP server listening. Buffers size: ['+obj.
              rtcp_server.getRecvBufferSize()+', '+obj.rtcp_server.getSendBufferSize
              ()+']');
479     });
480
481     // Handle errors on the RTCP server
482     this.rtcp_server.on('error', function(err) {
483       obj.dispIf(obj.DEBUG, 'RTCP Server Error:' + stringify(err));
484     });
485
486     // Handle closure of the RTCP server
487     this.rtcp_server.on('close', function() {
488       obj.dispIf(obj.DEBUG, 'RTCP UDP Server closed.');
489     });
490
491     // Bind the UDP socket to the specified port
492     this.rtcp_server.bind(this.client_port_rtcp, function() {
493       obj.dispIf(obj.DEBUG, 'RTCP UDP Server bind complete.');
494     });
495   }
496
497   /**
498    * Handles new RTP packets, decoding headers and preparing for NAL unit
          extraction.
499    * @param {Buffer} rtp_packet - The raw RTP packet to process.
500    */
501   onNewPacketRTP(rtp_packet) {
502     var id = this.decodeHeaderRTP(rtp_packet);
503     if(!isUndefined(id)) {
504       this.dispIf(this.DEBUG_INFO, 'RTP '+id);
505     }
506     this.decodeNextPayloadRTP(id);
507   }
508
509   /**
510    * Decodes the payload of the next RTP packet, assembling NAL units for video
          frames.
511    * @param {number} id - The ID of the RTP packet to decode.
512    */
513   decodeNextPayloadRTP(id) {
```

```
514     var obj = this;
515
516     if(isUndefined(this.rtp_packet_decoding_id) && !isUndefined(id)) {
517       this.rtp_packet_decoding_id = id-1;
518     } else if(id != (this.rtp_packet_decoding_id+1)) {
519       if(isUndefined(id)) {
520         this.dispIf(this.DEBUG_RTP_PACKETS_INFO, 'Invalid RTP instead of '+(this
              .rtp_packet_decoding_id+1));
521       } else {
522         this.dispIf(this.DEBUG_RTP_PACKETS_INFO, 'Got RTP '+id+' instead of '+(
              this.rtp_packet_decoding_id+1));
523         if(id == (this.rtp_packet_decoding_id+2)) {
524           this.dispIf(this.DEBUG_RTP_PACKETS_INFO, 'Last used RTP '+this.
                rtp_packet_decoding_id);
525         }
526       }
527
528       // Missing some packet
529       if(this.protocol == 'udp') {
530         this.rtp_packets_buffered += 1;
531         if(this.rtp_packets_buffered >= this.RTSP_REORDER_QUEUE_SIZE) {
532           this.dispIf(this.DEBUG_RTP_PACKETS_MISSING, 'UDP max packets buffered!
                ');
533           this.skipPacketsRTP();
534         } else if(!this.skip_packets_timeout) {
535           this.skip_packets_timeout = setTimeout(function() {
536             obj.dispIf(obj.DEBUG_RTP_PACKETS_MISSING, 'UDP max packet delay!');
537             obj.skipPacketsRTP();
538           }, this.RTSP_MAX_DELAY);
539         }
540       } else if(this.protocol == 'tcp') {
541         this.dispIf(this.DEBUG_RTP_PACKETS_MISSING, 'TCP missing packet!');
542         this.skipPacketsRTP();
543       }
544     }
545
546     // Decode consecutive packets
547     while(Object.keys(this.rtp_packets).includes(''+(this.rtp_packet_decoding_id
          +1))) {
548       // Decode packet
549       this.decodePayloadRTP(this.rtp_packet_decoding_id+1);
550     }
551   }
552
553   /**
554    * Skips RTP packets until the next valid packet is found or a threshold is
          reached.
555    */
556   skipPacketsRTP() {
557     var obj = this;
558     var next_id;
559
560     this.rtp_packets_buffered = 0;
561     this.skip_packets_timeout = clearTimeoutIf(this.skip_packets_timeout);
562
563     if(Object.keys(this.rtp_packets).length) {
564       if(this.RTSP_DISCARD_PACKETS) {
565         next_id = Math.max(...Object.keys(this.rtp_packets));
566       } else {
```

```
567         next_id = Object.keys(this.rtp_packets).find(function(id) {
568           return id > obj.rtp_packet_decoding_id;
569         });
570       }
571     }
572
573     if(!isUndefined(next_id)) {
574       this.rtp_packet_decoding_id = next_id-1; // Get id before next id
575     } else {
576       this.rtp_packet_decoding_id += 1;
577     }
578
579     // Remove skipped packets
580     Object.keys(this.rtp_packets).forEach(function(id) {
581       if(id < obj.rtp_packet_decoding_id) {
582         delete obj.rtp_packets[id];
583       }
584     });
585     this.rtp_packets_skipped = true;
586
587     this.rtp_packets_skipped_counter += (this.rtp_packet_decoding_id - this.
          rtp_packet_last_decoded_id);
588     this.rtp_packets_missed_counter += 1;
589     if((this.rtp_packet_decoding_id - (this.rtp_packet_last_decoded_id+1)) <
          this.RTP_PACKET_SKIPPED_PRINT_MAX) {
590       for(var id = this.rtp_packet_last_decoded_id+1; id <= this.
            rtp_packet_decoding_id; id++) {
591         this.dispIf(this.DEBUG_RTP_PACKETS_MISSING, 'Skipped RTP packet ' + id);
592       }
593     } else {
594       this.dispIf(this.DEBUG_RTP_PACKETS_MISSING, 'Skipped RTP packets ' + (this
            .rtp_packet_last_decoded_id+1) + '-'+this.rtp_packet_decoding_id);
595     }
596     this.rtp_packet_last_decoded_id = this.rtp_packet_decoding_id;
597     this.decodeNextPayloadRTP(this.rtp_packet_decoding_id+1); // We need to
          process next available packet
598   }
599
600   /**
601    * Decodes the RTP packet header to extract sequencing information and payload
          type.
602    * @param {Buffer} rtp_packet_data - The raw RTP packet data.
603    * @returns {number|undefined} The ID of the RTP packet if successfully
          decoded, undefined otherwise.
604    */
605   decodeHeaderRTP(rtp_packet_data) {
606     // Check if the packet is at least 12 bytes (minimum RTP header size)
607     if(rtp_packet_data.length < 12) {
608       this.dispIf(this.DEBUG_RTP_PACKETS_INFO, "Invalid RTP packet. It must be
            at least 12 bytes.");
609       return false;
610     }
611
612     this.rtp_packets_counter += 1;
613
614     // Extract RTP header fields
615     var rtp_packet = {};
616     rtp_packet.version = (rtp_packet_data[0] >> 6) & 0x03;
617     rtp_packet.padding = (rtp_packet_data[0] >> 5) & 0x01;
```

```
618     rtp_packet.extension = (rtp_packet_data[0] >> 4) & 0x01;
619     rtp_packet.csrc_count = rtp_packet_data[0] & 0x0F;
620     rtp_packet.marker = (rtp_packet_data[1] >> 7) & 0x01;
621     rtp_packet.packet_payload_type = rtp_packet_data[1] & 0x7F;
622     rtp_packet.sequence_number = (rtp_packet_data[2] << 8) | rtp_packet_data[3];
623     rtp_packet.timestamp = (rtp_packet_data[4] << 24) | (rtp_packet_data[5] <<
            16) | (rtp_packet_data[6] << 8) | rtp_packet_data[7];
624     rtp_packet.ssrc = (rtp_packet_data[8] << 24) | (rtp_packet_data[9] << 16) |
            (rtp_packet_data[10] << 8) | rtp_packet_data[11];
625     this.rtp_header = rtp_packet_data[0];
626
627     // Extract CSRC identifiers
628     rtp_packet.csrc_identifiers = [];
629     for(let i = 0; i < rtp_packet.csrc_count; i++) {
630       var csrc_start_index = 12 + i * 4;
631       var csrc = (rtp_packet_data[csrc_start_index] << 24) | (rtp_packet_data[
              csrc_start_index + 1] << 16) | (rtp_packet_data[csrc_start_index + 2]
              << 8) | rtp_packet_data[csrc_start_index + 3];
632       rtp_packet.csrc_identifiers.push(csrc);
633     }
634
635     // Check for extension header
636     rtp_packet.extension_header = [];
637     if(rtp_packet.extension) {
638       var extension_start_index = 12 + rtp_packet.csrc_count * 4;
639       var extension_length = ((rtp_packet_data[extension_start_index + 2] << 8)
              | rtp_packet_data[extension_start_index + 3]) * 4;
640       rtp_packet.extension_header = rtp_packet_data.subarray(
              extension_start_index, extension_start_index + extension_length);
641     }
642
643     // Calculate payload size
644     rtp_packet.header_size = 12 + rtp_packet.csrc_count * 4 + (rtp_packet.
            extension ? rtp_packet.extension_header.length : 0);
645     rtp_packet.payload_size = rtp_packet_data.length - rtp_packet.header_size;
646     rtp_packet.packet_size = rtp_packet_data.length;
647
648     rtp_packet.header = rtp_packet_data.slice(0, rtp_packet.header_size);
649     rtp_packet.payload = rtp_packet_data.slice(rtp_packet.header_size);
650
651     rtp_packet.sequence_number += this.rtp_packet_sequence_number_rollovers
            *65536;
652
653     // Detect sequence number rollover
654     if(this.rtp_packet_last_sequence_number > -1 && Math.abs(rtp_packet.
            sequence_number-this.rtp_packet_last_sequence_number) > this.
            rollover_diff) {
655       this.rtp_packet_sequence_number_rollovers += 1;
656       rtp_packet.sequence_number += 65536;
657     }
658     this.rtp_packet_last_sequence_number = rtp_packet.sequence_number;
659
660     this.dispIf(this.DEBUG_TIMESTAMPS, "RTP packet: "+rtp_packet.sequence_number
            +" with timestamp: "+rtp_packet.timestamp+" at "+getTimestamp()+".");
661
662     if(rtp_packet.marker) {
663       this.dispIf(this.DEBUG_MARKER, "RTP packet: "+rtp_packet.sequence_number+"
              with marker and timestamp: "+rtp_packet.timestamp+" at "+getTimestamp
              ()+".");
```

```
664     }
665
666     if(this.rtp_packet_ids.includes(rtp_packet.sequence_number)) {
667       this.dispIf(this.DEBUG_RTP_PACKETS_INFO, 'Duplicate RTP '+rtp_packet.
          sequence_number+' packet!');
668       return false;
669     }
670
671     // NAL units counter
672     var nal_unit_type;
673     if(this.codec_type == "H264") {
674       nal_unit_type = rtp_packet.payload[0] & 0x1F;
675     } else if(this.codec_type == "HEVC") {
676       nal_unit_type = (rtp_packet.payload[0] & 0x7E) >>> 1;
677     }
678     if(!hasKey(this.nalu_packets_counter, nal_unit_type)) {
679       this.nalu_packets_counter[nal_unit_type] = 0;
680     }
681     this.nalu_packets_counter[nal_unit_type] += 1;
682
683     var id = rtp_packet.sequence_number;
684     this.rtp_packets[id] = rtp_packet;
685     this.rtp_packet_ids_i += 1;
686     if(this.rtp_packet_ids_i >= this.rtp_packet_ids_max_size) {
687       this.rtp_packet_ids_i = 0;
688     }
689     this.rtp_packet_ids[this.rtp_packet_ids_i] = id;
690
691     return id;
692   }
693
694   /**
695    * Decodes the payload of a single RTP packet, extracting and assembling NAL
          units.
696    * @param {number} id - The ID of the RTP packet whose payload is to be
          decoded.
697    */
698   decodePayloadRTP(id) {
699     this.rtp_packets_buffered = 0;
700     this.skip_packets_timeout = clearTimeoutIf(this.skip_packets_timeout);
701
702     this.nal_unit.rtp_packets_id.push(id);
703     this.rtp_packet_last_decoded_id = id;
704     this.rtp_packet_decoding_id = id;
705
706     if(this.codec_type == "H264") {
707       // https://github.com/ireader/media-server/blob/master/librtp/payload/rtp-
            h264-unpack.c
708       this.nal_unit.type = this.rtp_packets[id].payload[0] & 0x1F;
709
710       this.dispIf(this.DEBUG_INFO, 'NALU '+this.nal_unit.type);
711       if(this.nal_unit.type == 28 || this.nal_unit.type == 29) {
712         // FU-A (Fragmentation Units)
713         this.nal_unit.fu_header = {} ;
714         this.nal_unit.fu_header.S = (this.rtp_packets[id].payload[1] & 0x80) >>>
              7;
715         this.nal_unit.fu_header.E = (this.rtp_packets[id].payload[1] & 0x40) >>>
              6;
716         this.nal_unit.fu_header.fu_type = this.rtp_packets[id].payload[1] & 0x1F
```

```
        ;
717       var n = 2;
718       if(this.nal_unit.type == 29) {
719         n = 4; // skip DON
720       }
721       if(this.nal_unit.fu_header.S) {
722         // Recreate FU header
723         this.rtp_packets_skipped = false;
724         this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
              addNalUniteStartCode([(this.rtp_packets[id].payload[0] & 0xE0) | (
              this.rtp_packets[id].payload[1] & 0x1F)]));
725       }
726
727       this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
            rtp_packets[id].payload.slice(n));
728       if(this.nal_unit.fu_header.E && !this.rtp_packets_skipped) {
729         if(this.rtp_packets[id].marker) {
730           this.sendNalUnit();
731         }
732       }
733     } else if([6, 7, 8].includes(this.nal_unit.type)) {
734       // 7 SPS, 8 PPS, 6 SEI
735       this.got_header = true;
736       this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
            addNalUniteStartCode(this.rtp_packets[id].payload));
737     } else {
738       // 1, 5 IDR, ...
739       this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
            addNalUniteStartCode(this.rtp_packets[id].payload));
740       if(this.rtp_packets[id].marker) {
741         this.sendNalUnit();
742       }
743     }
744   } else if(this.codec_type == "HEVC") {
745     // https://github.com/ireader/media-server/blob/master/librtp/payload/rtp-
          h265-unpack.c
746
747     this.nal_unit.header = {};
748     this.nal_unit.header.forbidden_zero_bit = (this.rtp_packets[id].payload[0]
          & 0x80) >>> 7;
749     this.nal_unit.header.nal_unit_type = (this.rtp_packets[id].payload[0] & 0
          x7E) >>> 1;
750     this.nal_unit.header.nuh_layer_id = ((this.rtp_packets[id].payload[0] & 0
          x01) << 5) | ((this.rtp_packets[id].payload[1] & 0xF8) >>> 3);
751     this.nal_unit.header.nuh_temporal_id_plus = this.rtp_packets[id].payload
          [1] & 0x07;
752
753     this.dispIf(this.DEBUG_INFO, 'NALU '+this.nal_unit.header.nal_unit_type);
754     if(this.nal_unit.header.nal_unit_type == 49) {
755       // 49 FU (Fragmentation Units)
756       this.nal_unit.fu_header = {} ;
757       this.nal_unit.fu_header.S = (this.rtp_packets[id].payload[2] & 0x80) >>>
            7;
758       this.nal_unit.fu_header.E = (this.rtp_packets[id].payload[2] & 0x40) >>>
            6;
759       this.nal_unit.fu_header.fu_type = this.rtp_packets[id].payload[2] & 0x1F
          ;
760
761       var n = 3;
```

```
762         if(this.nal_unit.fu_header.S) {
763           // Recreate FU header
764           this.rtp_packets_skipped = false;
765           this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
                addNalUniteStartCode([((this.rtp_packets[id].payload[2] & 0x1F) <<
                1) | (this.rtp_packets[id].payload[0] & 0x81), this.rtp_packets[id
                ].payload[1]]));
766         }
767
768         this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
              rtp_packets[id].payload.slice(n));
769
770         if(this.nal_unit.fu_header.E && !this.rtp_packets_skipped) {
771           if(this.rtp_packets[id].marker) {
772             this.sendNalUnit();
773           }
774         }
775       } else if([32, 33, 34, 21].includes(this.nal_unit.header.nal_unit_type)) {
776         // 32 VPS, 33 SPS, 34 PPS, 21
777         this.got_header = true;
778         this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
              addNalUniteStartCode(this.rtp_packets[id].payload));
779       } else {
780         // 1 TRAIL_R, 19 IDR_W_RADL, ...
781         this.nal_unit.payload = this.mergeData(this.nal_unit.payload, this.
              addNalUniteStartCode(this.rtp_packets[id].payload));
782         if(this.rtp_packets[id].marker) {
783           this.sendNalUnit();
784         }
785       }
786     }
787   }
788
789   /**
790    * Adds the NAL unit start code to the given data array.
791    * @param {Array} data - The data array to prepend with the NAL unit start
          code.
792    * @returns {Uint8Array} A new array with the NAL unit start code prepended.
793    */
794   addNalUniteStartCode(data) {
795     return this.mergeData(this.nalu_start_code, data);
796   }
797
798   /**
799    * Merges two data arrays into a single Uint8Array.
800    * @param {Array} data1 - The first data array.
801    * @param {Array} data2 - The second data array.
802    * @returns {Uint8Array} The merged data array.
803    */
804   mergeData(data1, data2) {
805     var new_data = new Uint8Array(data1.length+data2.length);
806     new_data.set(data1, 0);
807     new_data.set(data2, data1.length);
808     return new_data;
809   }
810
811   /**
812    * Sends the assembled NAL unit to the packet callback for processing.
813    */
```

```
814   sendNalUnit() {
815     var obj = this;
816
817     // Delete packets
818     var max_id = Math.max(...this.nal_unit.rtp_packets_id);
819     Object.keys(this.rtp_packets).forEach(function(id) {
820       if(id <= max_id) {
821         delete obj.rtp_packets[id];
822       }
823     });
824
825     this.nal_unit.rtp_packets_id = [];
826
827     if(this.got_header) {
828       this.nal_unit_num += 1;
829       this.packet_callback(this.nal_unit.payload, this.nal_unit_num);
830     }
831
832     this.nal_unit.payload = [];
833     this.nal_unit.started = false;
834   }
835
836   /**
837    * Conditionally displays a message if the specified condition is true.
838    * @param {boolean} condition - The condition to evaluate.
839    * @param {string} msg - The message to display if the condition is true.
840    */
841   dispIf(condition, msg) {
842     if(condition) {
843       disp(msg);
844     }
845   }
846 }
847
848 exports.RTSP_DEMUXER = RTSP_DEMUXER;
```

Listing AB.2 - RTSP_DEMUXER.js

U listingu AB.3 je dat kod klase DECODER_JS za dekodiranje enkodiranih video paketa u JavaScript-u.

```
1  /**
2   * @file Video decoder JavaScript
3   * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
4   * Doktorska disertacija
5   * Mašinski fakultet Univerziteta u Beogradu
6   * @version 0.0.1
7   */
8
9  /**
10  * Class for video decoder.
11  */
12 class DECODER_JS {
13
14   /**
15    * Creates a new VideoDecoder instance with the specified options and frame
          callback.
16    * @param {Object} options - Configuration options for the decoder.
17    * @param {string} options.codec - The codec to use for decoding (e.g., "vp8",
          "h264").
18    * @param {number} options.video_height - The height of the video in pixels.
```

```
19      * @param {number} options.video_width - The width of the video in pixels.
20      * @param {string} options.hardware_acceleration - The hardware acceleration
            setting (e.g., "preferred", "required").
21      * @param {boolean} options.optimize_for_latency - Whether to optimize the
            decoder for low latency.
22      * @param {Function} onFrame - Callback function invoked with each decoded
            frame.
23      */
24    constructor(options, onFrame) {
25      var obj = this;
26      this.options = options;
27      this.onFrame = onFrame;
28
29      this.decoder = new VideoDecoder({
30        output: function(data) {
31          obj.onFrame(data);
32        },
33        error: function(err) {
34          disp('Decoder error: '+err);
35        }
36      });
37
38      this.decoder_config = {
39        codec: this.options.codec,
40        codedHeight: this.options.video_height,
41        codedWidth: this.options.video_width,
42        hardwareAcceleration: this.options.hardware_acceleration,
43        optimizeForLatency: this.options.optimize_for_latency
44      };
45      this.decoder.configure(this.decoder_config);
46    }
47
48    /**
49     * Decodes a packet of video data
50     * @param {ArrayBuffer} packet - The video packet data to decode.
51     */
52    decode(packet) {
53      if(this.decoder && this.decoder.state == "configured") {
54        try {
55          const chunk = new EncodedVideoChunk({
56            data: packet.buffer,
57            timestamp: 0,
58            type: "key"
59          });
60          this.decoder.decode(chunk);
61        } catch(err) {
62          disp('Decoder error: '+err);
63        }
64      }
65    }
66
67    /**
68     * Closes the decoder and releases any associated resources.
69     */
70    close() {
71      this.decoder.close();
72      this.decoder = null;
73    }
74  }
```

```
75
76 exports.DECODER_JS = DECODER_JS;
```

Listing AB.3 - `DECODER_JS.js`

AB.1 Modul decoder-ffmpeg

Da bi se komapjlirao ovaj modul neophodno je da je dostupna biblioteka `FFmpeg` (kao i za modul `server-rtsp-live555`).

Datoteka `package.json` (listing AB.4), sadrži osnovne informacije o paketu i njegovoj instalaciji, uključujući definisanje drugih potrebnih paketa.

```
1 {
2   "dependencies": {
3     "node-addon-api": "8.2.1"
4   }
5 }
```

Listing AB.4 - `decoder-ffmpeg package.json`

Datoteka `binding.gyp` (listing AB.5) pruža konfiguraciju za `node-gyp`, alat koji omogućava kompajliranje `C++` koda u binarni modul kompatibilan sa `Node.js` okruženjem.

```
1 {
2   "targets": [
3     {
4       "target_name": "decoder_ffmpeg",
5       "sources": [
6         "decoder-ffmpeg.cpp"
7       ],
8       "include_dirs": [
9         "<(module_root_dir)",
10        "<!@(node -p \"require('node-addon-api').include\")",
11        "<(module_root_dir)/ffmpeg-6.0/include"
12      ],
13      "libraries": [
14        "<(module_root_dir)/ffmpeg-6.0/lib/avcodec.lib",
15        "<(module_root_dir)/ffmpeg-6.0/lib/avdevice.lib",
16        "<(module_root_dir)/ffmpeg-6.0/lib/avformat.lib",
17        "<(module_root_dir)/ffmpeg-6.0/lib/avfilter.lib",
18        "<(module_root_dir)/ffmpeg-6.0/lib/avutil.lib",
19        "<(module_root_dir)/ffmpeg-6.0/lib/swscale.lib",
20        "<(module_root_dir)/ffmpeg-6.0/lib/swresample.lib"
21      ],
22      "cflags!": [
23        "-fno-exceptions"
24      ],
25      "cflags_cc!": [
26        "-fno-exceptions"
27      ],
28      "cflags": [
29        "-O3"
30      ],
31      "cflags_cc": [
32        "-O3"
33      ],
34      "defines": [
35        "NAPI_DISABLE_CPP_EXCEPTIONS"
```

```
      ],
      "copies": [
        {
          "destination": "<(module_root_dir)/build/Release",
          "files": [
            "<(module_root_dir)/ffmpeg-6.0/bin/avcodec-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avdevice-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avfilter-9.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avformat-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avutil-58.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/swresample-4.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/swscale-7.dll"
          ]
        }
      ],
      "msvs_settings": {
        "VCCLCompilerTool": {
          "AdditionalOptions": [
            "-std:c++17"
          ]
        }
      }
    }
  ]
}
```

Listing AB.5 - `decoder-ffmpeg binding.gyp`

Implementacija modula `decoder-ffmpeg` je data u listinzima AB.6 i AB.7.

```
// DecoderFFmpeg - decoder_ffmpeg.h
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#ifndef DECODER_FFMPEG_H
#define DECODER_FFMPEG_H

//#define DEBUG_DECODER_FFMPEG
//#define PROFILE_DECODER_FFMPEG
//#define DEBUG_FFMPEG
//#define DEBUG_FFMPEG_CUSTOM_CALLBACK

#define DEBUG_DECODER_FFMPEG_LEVEL 1

#include <napi.h>
#include <chrono>
#include <Windows.h>
#include <ctime>
#include <iomanip>
#include <sstream>
#include <string>

namespace decoder_ffmpeg_ns {

using namespace std;
using namespace std::chrono;

// FFMPEG LGPL librray from: https://github.com/BtbN/FFmpeg-Builds/releases
extern "C" {
```

```
32    #include "libavformat/avformat.h"
33    #include "libavcodec/avcodec.h"
34    #include "libswscale/swscale.h"
35    #include "libavdevice/avdevice.h"
36  }
37
38  class DecoderFFmpeg : public Napi::ObjectWrap<DecoderFFmpeg> {
39   public:
40    static Napi::Object Init(Napi::Env env, Napi::Object exports);
41    DecoderFFmpeg(const Napi::CallbackInfo& info);
42    ~DecoderFFmpeg();
43
44    int consoleLog(int32_t, const char*, ...);
45
46    Napi::Value configure(const Napi::CallbackInfo& info);
47    Napi::Value decode(const Napi::CallbackInfo& info);
48
49    uint32_t frame_number;
50  #ifdef PROFILE_STREAM_DECODER_FFMPEG
51    time_point<steady_clock> t_packet;
52  #endif
53
54    uint32_t id = 0;
55    int32_t debug_level = DEBUG_DECODER_FFMPEG_LEVEL;
56
57    std::string codec_id;
58    uint32_t frame_width;
59    uint32_t frame_height;
60    uint32_t framerate;
61    std::string pix_fmt_id;
62
63    Napi::ArrayBuffer packet_data;
64
65    AVCodecContext* decode_context = NULL;
66    const AVCodec* decoder = NULL;
67    AVFrame* frame = NULL;
68    AVPacket* packet = NULL;
69    AVFrame* rgba_frame = NULL;
70    SwsContext* rgba_sws_context = NULL;
71    AVPixelFormat pix_fmt;
72  };
73
74  }// namespace decoder_ffmpeg_ns
75
76  #endif // DECODER_FFMPEG_H
```

Listing AB.6 - `decoder-ffmpeg.h`

```
1   // DecoderFFmpeg - decoder_ffmpeg.cpp
2   // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Mašinski fakultet Univerziteta u Beogradu
5   // --------------------
6
7   #include "decoder-ffmpeg.h"
8
9   namespace decoder_ffmpeg_ns {
10
11  // Function to start the timer and return the start time
12  time_point<steady_clock> tic() {
```

```
13      return steady_clock::now();
14  }
15
16  // Function to stop the timer and return the elapsed time
17  long toc(const time_point<steady_clock>& startTime) {
18      return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
19  }
20
21  // Function to get current time
22  std::string getCurrentTime() {
23    auto now = system_clock::now();
24    auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
25    auto timer = system_clock::to_time_t(now);
26    std::tm bt = *std::localtime(&timer);
27    std::ostringstream oss;
28    oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
29    oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
30    return oss.str();
31  }
32
33  #ifdef DEBUG_FFMPEG_CUSTOM_CALLBACK
34  // FFmpeg logging callback
35  // --------------------
36  void ffmpegLogCallback(void* avcl, int level, const char* format, va_list args)
        {
37    const int bufferSize = 1024;
38    char buffer[bufferSize];
39    vsnprintf(buffer, bufferSize, format, args);
40    printf("[DecoderFFmpeg LOGGING]: %s\n", buffer);
41  }
42  #endif
43
44  // consoleLog() function
45  // Function to console log data
46  // --------------------
47  int DecoderFFmpeg::consoleLog(int32_t level, const char* format, ...) {
48    if(level < 0 || level <= debug_level) {
49      if(id > 0) {
50        printf("\033[0;33m[%s DecoderFFmpeg %d]\033[0m ", getCurrentTime().c_str()
              , id);
51      } else {
52        printf("\033[0;33m[%s DecoderFFmpeg]\033[0m ", getCurrentTime().c_str());
53      }
54
55      va_list vl;
56      va_start(vl, format);
57      auto ret = vprintf(format, vl);
58      va_end(vl);
59      printf("\n");
60      return ret;
61    }
62    return 0;
63  }
64
65  // ElectronVideoRTSP()
66  // Object DecoderFFmpeg
67  // --------------------
68  DecoderFFmpeg::DecoderFFmpeg(const Napi::CallbackInfo& info)
69      : Napi::ObjectWrap<DecoderFFmpeg>(info) {
```

```
70
71    // Get the handle to the current process
72    HANDLE hProcess = GetCurrentProcess();
73
74    // Set the priority class of the process
75    BOOL success = SetPriorityClass(hProcess, REALTIME_PRIORITY_CLASS);
76    if(!success) {
77      consoleLog(0, "Failed to set process priority.");
78    }
79
80     // Get the handle to the current thread
81    HANDLE hThread = GetCurrentThread();
82
83    // Set the priority class of the process
84    BOOL success2 = SetThreadPriority(hThread, THREAD_PRIORITY_TIME_CRITICAL);
85    if(!success2) {
86      consoleLog(0, "Failed to set thread priority.");
87    }
88
89    // Set FFmpeg log level and callback
90  #ifdef DEBUG_FFMPEG
91    av_log_set_level(AV_LOG_DEBUG);
92  #endif
93  #ifdef DEBUG_FFMPEG_CUSTOM_CALLBACK
94    av_log_set_callback(ffmpegLogCallback);
95  #endif
96  }
97
98  // DecoderFFmpeg()
99  // Object destructor
100 // --------------------
101 DecoderFFmpeg::~DecoderFFmpeg() {
102   // Cleanup
103   avcodec_free_context(&decode_context);
104 }
105
106 // Init() function
107 // --------------------
108 Napi::Object DecoderFFmpeg::Init(Napi::Env env, Napi::Object exports) {
109   Napi::Function func =
110       DefineClass(env,
111                   "DecoderFFmpeg",
112                    {InstanceMethod("configure", &DecoderFFmpeg::configure),
113                     InstanceMethod("decode", &DecoderFFmpeg::decode),
114                   });
115
116   Napi::FunctionReference* constructor = new Napi::FunctionReference();
117   *constructor = Napi::Persistent(func);
118   env.SetInstanceData(constructor);
119
120   exports.Set("DecoderFFmpeg", func);
121   return exports;
122 }
123
124 // configure() function
125 // --------------------
126 Napi::Value DecoderFFmpeg::configure(const Napi::CallbackInfo& info) {
127 #ifdef DEBUG_DECODER_FFMPEG
128   consoleLog(0, "Call to configure()");
```

```
129 #endif
130   Napi::Object return_data = Napi::Object::New(info.Env());
131   int ret;
132   char a[AV_ERROR_MAX_STRING_SIZE] = { 0 };
133 
134   if(!info[0].IsObject()) {
135 #ifdef DEBUG_DECODER_FFMPEG
136     consoleLog(0, "Wrong arguments.");
137 #endif
138     return_data.Set("error", "Wrong arguments.");
139     return_data.Set("code", 19);
140     return return_data;
141   }
142 
143   Napi::Object options = info[0].As<Napi::Object>();
144 
145   // Handle options
146   if(options.Has("id")) {
147     id = options.Get("id").As<Napi::Number>().Uint32Value();
148 #ifdef DEBUG_DECODER_FFMPEG
149     consoleLog(0, "ID set to %d.", id);
150 #endif
151   } else {
152     id = 0;
153   }
154 
155   if(options.Has("debug_level")) {
156     debug_level = options.Get("debug_level").As<Napi::Number>().Int32Value();
157 #ifdef DEBUG_DECODER_FFMPEG
158     consoleLog(0, "Debug level set to %d.", debug_level);
159 #endif
160   }
161 
162   if(options.Has("frame_width")) {
163     frame_width = options.Get("frame_width").As<Napi::Number>().Uint32Value();
164 #ifdef DEBUG_DECODER_FFMPEG
165     consoleLog(0, "Frame width set to %d.", frame_width);
166 #endif
167   } else {
168 #ifdef DEBUG_DECODER_FFMPEG
169     consoleLog(0, "Unknown frame width.");
170 #endif
171     return_data.Set("error", "Unknown frame width.");
172     return_data.Set("code", 20);
173     return return_data;
174   }
175 
176   if(options.Has("frame_height")) {
177     frame_height = options.Get("frame_height").As<Napi::Number>().Uint32Value();
178 #ifdef DEBUG_DECODER_FFMPEG
179     consoleLog(0, "Frame height set to %d.", frame_height);
180 #endif
181   } else {
182 #ifdef DEBUG_DECODER_FFMPEG
183       consoleLog(0, "Unknown frame height.");
184 #endif
185       return_data.Set("error", "Unknown frame height.");
186       return_data.Set("code", 21);
187       return return_data;
```

```
188   }
189
190   if(options.Has("framerate")) {
191     framerate = options.Get("framerate").As<Napi::Number>().Uint32Value();
192 #ifdef DEBUG_DECODER_FFMPEG
193     consoleLog(0, "Framerate set to %d.", framerate);
194 #endif
195   } else {
196 #ifdef DEBUG_DECODER_FFMPEG
197     consoleLog(0, "Unknown framerate.");
198 #endif
199     return_data.Set("error", "Unknown framerate.");
200     return_data.Set("code", 22);
201     return return_data;
202   }
203
204   if(options.Has("pix_fmt_id")) {
205     pix_fmt_id = options.Get("pix_fmt_id").As<Napi::String>();
206 #ifdef DEBUG_DECODER_FFMPEG
207     consoleLog(0, "Pixel format set to %s.", pix_fmt_id);
208 #endif
209     if(pix_fmt_id == "YUVJ420P") {
210       pix_fmt = AV_PIX_FMT_YUVJ420P;
211     } else if(pix_fmt_id == "YUV420P") {
212       pix_fmt = AV_PIX_FMT_YUV420P;
213     } else {
214 #ifdef DEBUG_DECODER_FFMPEG
215       consoleLog(0, "Unknown pixel format.");
216 #endif
217     }
218   } else {
219 #ifdef DEBUG_DECODER_FFMPEG
220     consoleLog(0, "Unknown pixel format.");
221 #endif
222     return_data.Set("error", "Unknown pixel format.");
223     return_data.Set("code", 23);
224     return return_data;
225   }
226
227   if(options.Has("codec_id")) {
228     codec_id = options.Get("codec_id").As<Napi::String>();
229 #ifdef DEBUG_DECODER_FFMPEG
230     consoleLog(0, "Codec ID set to %s.", codec_id);
231 #endif
232     if(codec_id == "HEVC") {
233       decoder = avcodec_find_decoder(AV_CODEC_ID_HEVC);
234     } else if(codec_id == "H264") {
235       decoder = avcodec_find_decoder(AV_CODEC_ID_H264);
236     } else {
237 #ifdef DEBUG_DECODER_FFMPEG
238       consoleLog(0, "Unknown decoder.");
239 #endif
240       return_data.Set("error", "Unknown decoder.");
241       return_data.Set("code", 24);
242       return return_data;
243     }
244   } else {
245 #ifdef DEBUG_DECODER_FFMPEG
246     consoleLog(0, "Unknown decoder.");
```

```
247 #endif
248     return_data.Set("error", "Unknown decoder.");
249     return_data.Set("code", 24);
250     return return_data;
251   }
252
253   if(!decoder) {
254 #ifdef DEBUG_DECODER_FFMPEG
255       consoleLog(0, "Failed to find decoder.");
256 #endif
257     return_data.Set("error", "Failed to find decoder.");
258     return_data.Set("code", 6);
259     return return_data;
260   }
261
262   // Allocate a codec context
263   decode_context = avcodec_alloc_context3(decoder);
264   if(decode_context == NULL) {
265     return_data.Set("error", "Could not allocate codec context.");
266     return_data.Set("code", 7);
267     return return_data;
268   }
269 #ifdef DEBUG_DECODER_FFMPEG
270     else {
271     consoleLog(0, "Video decoder context created.");
272   }
273 #endif
274
275   AVRational video_framerate;
276   video_framerate.num = framerate;
277   video_framerate.den = 1;
278
279   decode_context->width = frame_width;  // Set video width
280   decode_context->height = frame_height; // Set video height
281   decode_context->framerate = video_framerate;  // Set video framerate
282   decode_context->pix_fmt = pix_fmt;
283
284   // Open the codec
285   ret = avcodec_open2(decode_context, decoder, nullptr);
286   if(ret != 0) {
287 #ifdef DEBUG_DECODER_FFMPEG
288     consoleLog(0, "Failed to open input codec: %s",
289       av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
290 #endif
291     return_data.Set("error", "Failed to open input codec: " +
292       std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret)));
293     return_data.Set("code", 9);
294     return return_data;
295   }
296 #ifdef DEBUG_DECODER_FFMPEG
297     else {
298     consoleLog(0, "Video codec opened.");
299   }
300 #endif
301
302   // Allocate video packet
303   packet = av_packet_alloc();
304   if(!packet) {
305 #ifdef DEBUG_DECODER_FFMPEG
```

```
306     consoleLog(0, "Failed to allocate packet.");
307 #endif
308     return_data.Set("error", "Failed to allocate packet.");
309     return_data.Set("code", 10);
310     return return_data;
311   }
312 #ifdef DEBUG_DECODER_FFMPEG
313     else {
314     consoleLog(0, "Packet allocated.");
315   }
316 #endif
317
318   // Allocate video frame
319   frame = av_frame_alloc();
320   if(!frame) {
321 #ifdef DEBUG_DECODER_FFMPEG
322     consoleLog(0, "Failed to allocate input video frame.");
323 #endif
324     return_data.Set("error", "Failed to allocate input video frame.");
325     return_data.Set("code", 11);
326     return return_data;
327   }
328 #ifdef DEBUG_DECODER_FFMPEG
329     else {
330     consoleLog(0, "Frame allocated.");
331   }
332 #endif
333
334   // Allocate RGBA video frame
335   rgba_frame = av_frame_alloc();
336   if(!rgba_frame) {
337 #ifdef DEBUG_DECODER_FFMPEG
338     consoleLog(0, "Failed to allocate output frame.");
339 #endif
340     return_data.Set("error", "Failed to allocate output frame.");
341     return_data.Set("code", 25);
342     return return_data;
343   }
344 #ifdef DEBUG_DECODER_FFMPEG
345     else {
346     consoleLog(0, "Frame allocated.");
347   }
348 #endif
349
350   rgba_frame->format = AV_PIX_FMT_RGBA;
351   rgba_frame->width = decode_context->width;
352   rgba_frame->height = decode_context->height;
353   av_frame_get_buffer(rgba_frame, 0);
354
355   // Pixel change context
356   rgba_sws_context = sws_getContext(decode_context->width,
357     decode_context->height, decode_context->pix_fmt,
358     decode_context->width, decode_context->height,
359     AV_PIX_FMT_RGBA, SWS_BICUBIC, NULL, NULL, NULL);
360
361   frame_number = 1;
362 #ifdef PROFILE_DECODER_FFMPEG
363   t_packet = tic();
364 #endif
```

```
365
366   return Napi::Boolean::New(info.Env(), true);
367 }
368
369 // decode() function
370 // --------------------
371 Napi::Value DecoderFFmpeg::decode(const Napi::CallbackInfo& info) {
372 #ifdef DEBUG_DECODER_FFMPEG
373   consoleLog(2, "Call to decode()");
374 #endif
375   Napi::Object return_data = Napi::Object::New(info.Env());
376
377   int ret;
378   char a[AV_ERROR_MAX_STRING_SIZE] = { 0 };
379
380   packet_data = info[0].As<Napi::ArrayBuffer>();
381
382   // Set packet data and size
383   packet->data = reinterpret_cast<uint8_t*>(packet_data.Data());
384   packet->size = packet_data.ByteLength();
385
386 #ifdef PROFILE_STREAM_DECODER_FFMPEG
387   auto t_decode = tic();
388   consoleLog(1, "--- DecoderFFmpeg profiler ---");
389   consoleLog(1, "New packet %d, %d ms", frame_number, toc(t_packet));
390   t_packet = tic();
391 #endif
392
393   ret = avcodec_send_packet(decode_context, packet);
394   if(ret < 0) {
395 #ifdef DEBUG_DECODER_FFMPEG
396     consoleLog(0, "Failed to send packet to the decoder: %s",
397       av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
398 #endif
399     return_data.Set("error", "Failed to send packet to the decoder: " +
400       std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret)));
401     return_data.Set("code", 26);
402     return return_data;
403   }
404
405   // Receive and process the decoded frames
406   while(ret >= 0) {
407     ret = avcodec_receive_frame(decode_context, frame);
408     if(ret == AVERROR(EAGAIN) || ret == AVERROR_EOF) {
409       break;
410     } else if(ret < 0) {
411 #ifdef DEBUG_DECODER_FFMPEG
412       consoleLog(0, "Failed to decode packet: %s",
413         av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
414 #endif
415       return_data.Set("error", "Failed to decode packet: " +
416         std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret)));
417       return_data.Set("code", 27);
418       return return_data;
419     }
420
421 #ifdef PROFILE_DECODER_FFMPEG
422     consoleLog(1, "Decoded frame %d", frame_number);
423 #endif
```

```
424
425     sws_scale(rgba_sws_context, frame->data, frame->linesize, 0,
426       frame->height, rgba_frame->data, rgba_frame->linesize);
427
428     uint8_t* frame_data = rgba_frame->data[0];
429     int frame_size = rgba_frame->width * rgba_frame->height * 4;
430     uint32_t packet_size = packet->size * 8;
431
432     return_data.Set("buffer", Napi::Buffer<uint8_t>::Copy(info.Env(), frame_data
            , frame_size));
433     return_data.Set("packet_size", packet_size);
434     return_data.Set("frame_number", frame_number);
435
436     frame_number++;
437     return return_data;
438   }
439
440 #ifdef PROFILE_DECODER_FFMPEG
441   consoleLog(1, "Frame decode time %d ms", toc(t_decode));
442   fflush(stdout);
443 #endif
444
445   return Napi::Boolean::New(info.Env(), false);
446 }
447
448 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
449   return DecoderFFmpeg::Init(env, exports);
450 }
451
452 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
453
454 }  // namespace decoder_ffmpeg_ns
```

Listing AB.7 - `decoder-ffmpeg.cpp`

AB.2 Modul stream-decoder-ffmpeg

Da bi se komapjlirao ovaj modul neophodno je da je dostupna biblioteka `FFmpeg` (kao i za modul `server-rtsp-live555`).

Datoteka `package.json` (listing AB.8), sadrži osnovne informacije o paketu i njegovoj instalaciji, uključujući definisanje drugih potrebnih paketa.

```
1 {
2   "dependencies": {
3     "node-addon-api": "8.2.1"
4   }
5 }
```

Listing AB.8 - `stream-decoder-ffmpeg package.json`

Datoteka `binding.gyp` (listing AB.9) pruža konfiguraciju za `node-gyp`, alat koji omogućava kompajliranje `C++` koda u binarni modul kompatibilan sa `Node.js` okruženjem.

```
1 {
2   "targets": [
3     {
4       "target_name": "stream_decoder_ffmpeg",
5       "sources": [
```

```
        "stream-decoder-ffmpeg.cpp"
      ],
      "include_dirs": [
        "<(module_root_dir)",
        "<!@(node -p \"require('node-addon-api').include\")",
        "<(module_root_dir)/ffmpeg-6.0/include"
      ],
      "libraries": [
        "<(module_root_dir)/ffmpeg-6.0/lib/avcodec.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avdevice.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avformat.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avfilter.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/avutil.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/swscale.lib",
        "<(module_root_dir)/ffmpeg-6.0/lib/swresample.lib"
      ],
      "cflags!": [
        "-fno-exceptions"
      ],
      "cflags_cc!": [
        "-fno-exceptions"
      ],
      "cflags": [
        "-O3"
      ],
      "cflags_cc": [
        "-O3"
      ],
      "defines": [
        "NAPI_DISABLE_CPP_EXCEPTIONS"
      ],
      "copies": [
        {
          "destination": "<(module_root_dir)/build/Release",
          "files": [
            "<(module_root_dir)/ffmpeg-6.0/bin/avcodec-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avdevice-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avfilter-9.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avformat-60.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/avutil-58.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/swresample-4.dll",
            "<(module_root_dir)/ffmpeg-6.0/bin/swscale-7.dll"
          ]
        }
      ],
      "msvs_settings": {
        "VCCLCompilerTool": {
          "AdditionalOptions": [
            "-std:c++17"
          ]
        }
      }
    }
  ]
}
```

Listing AB.9 - `stream-decoder-ffmpeg binding.gyp`

Implementacija modula `stream-decoder-ffmpeg` je data u listinzima AB.10 i AB.11.

```
// StreamDecoderFFmpeg - stream-decoder-ffmpeg.h
```

```
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #ifndef STREAM_DECODER_FFMPEG_H
8  #define STREAM_DECODER_FFMPEG_H
9
10 //#define DEBUG_STREAM_DECODER_FFMPEG
11 //#define LIST_STREAM_DECODER_FFMPEG_OPTIONS
12 //#define PROFILE_STREAM_DECODER_FFMPEG
13 //#define DEBUG_FFMPEG
14 //#define DEBUG_FFMPEG_CUSTOM_CALLBACK
15
16 #define DEBUG_STREAM_DECODER_FFMPEG_LEVEL 1
17 #define FRAME_READ_TIMEOUT 1000
18 #define OPEN_INPUT_TIMEOUT 1000
19 #define CLOSE_INPUT_FORMAT
20
21 #include <napi.h>
22 #include <chrono>
23 #include <thread>
24 #include <Windows.h>
25 #include <ctime>
26 #include <iomanip>
27 #include <sstream>
28 #include <string>
29 #include <mutex>
30
31 int interruptCallBack(void* ptr);
32
33 namespace stream_decoder_ffmpeg_ns {
34
35 using namespace std;
36 using namespace std::chrono;
37
38 // FFMPEG LGPL librray from: https://github.com/BtbN/FFmpeg-Builds/releases
39 extern "C" {
40   #include "libavformat/avformat.h"
41   #include "libavcodec/avcodec.h"
42   #include "libswscale/swscale.h"
43   #include "libavdevice/avdevice.h"
44 }
45
46 class StreamDecoderFFmpeg : public Napi::ObjectWrap<StreamDecoderFFmpeg> {
47  public:
48   static Napi::Object Init(Napi::Env env, Napi::Object exports);
49   StreamDecoderFFmpeg(const Napi::CallbackInfo& info);
50   ~StreamDecoderFFmpeg();
51
52   void threadEntry();
53   Napi::Value start(const Napi::CallbackInfo& info);
54   void stop(const Napi::CallbackInfo& info);
55   void pause(const Napi::CallbackInfo& info);
56   void resume(const Napi::CallbackInfo& info);
57
58   Napi::Function frame_cb;
59   Napi::Function error_cb;
60
```

```
61    Napi::ThreadSafeFunction tsfn_frame;
62    Napi::ThreadSafeFunction tsfn_error;
63
64    int consoleLog(int32_t, const char*, ...);
65    void listObjOptions(void*, int);
66    void listObjOptions(const AVClass*, int);
67
68    void setError(uint8_t);
69    bool isStoped();
70
71    std::thread thread;
72
73    uint32_t id = 0;
74    int32_t debug_level = DEBUG_STREAM_DECODER_FFMPEG_LEVEL;
75
76    bool active = false;
77    uint8_t error = 0;
78    bool opening_input = false;
79    bool reading_packets = false;
80
81    bool cmd_stop_thread = false;
82    bool cmd_pause = false;
83    bool cmd_resume = false;
84
85    time_point<steady_clock> t_open_input;
86    time_point<steady_clock> t_read_frame;
87
88    std::string input_address;
89    std::string input_format_type;
90
91    std::string video_size;
92    std::string framerate;
93
94    bool flag_nonblock;
95    AVIOInterruptCB int_cb = { interruptCallBack, this};
96
97    std::string rtbufsize;
98    std::string max_delay;
99    std::string avioflags;
100   std::string probesize;
101   std::string fpsprobesize;
102   std::string formatprobesize;
103   std::string max_probe_packets;
104   std::string analyzeduration;
105   std::string skip_estimate_duration_from_pts;
106   std::string max_ts_probe;
107   std::string max_interleave_delta;
108   std::string fflags;
109   std::string flush_packets;
110
111   std::string buffer_size;
112   std::string rtsp_transport;
113   std::string rtsp_flags;
114
115   uint32_t dec_thread_count;
116   std::string dec_flags;
117   std::string dec_flags2;
118
119   AVDictionary* input_format_options = NULL;
```

```
120 };
121
122 }// namespace stream_decoder_ffmpeg_ns
123
124 #endif // STREAM_DECODER_FFMPEG_H
```

Listing AB.10 - `stream-decoder-ffmpeg.h`

```
1  // StreamDecoderFFmpeg - stream-decoder-ffmpeg.cpp
2  // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3  // Doktorska disertacija
4  // Mašinski fakultet Univerziteta u Beogradu
5  // --------------------
6
7  #include "stream-decoder-ffmpeg.h"
8
9  std::mutex mtx; // mutex for critical section
10
11 namespace stream_decoder_ffmpeg_ns {
12
13 // Function to start the timer and return the start time
14 time_point<steady_clock> tic() {
15     return steady_clock::now();
16 }
17
18 // Function to stop the timer and return the elapsed time
19 long toc(const time_point<steady_clock>& startTime) {
20     return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
21 }
22
23 // Function to get current time
24 std::string getCurrentTime() {
25   auto now = system_clock::now();
26   auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
27   auto timer = system_clock::to_time_t(now);
28   std::tm bt = *std::localtime(&timer);
29   std::ostringstream oss;
30   oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
31   oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
32   return oss.str();
33 }
34
35 #ifdef DEBUG_FFMPEG_CUSTOM_CALLBACK
36 // FFmpeg logging callback
37 // --------------------
38 void ffmpegLogCallback(void* avcl, int level, const char* format, va_list args)
       {
39   const int bufferSize = 1024;
40   char buffer[bufferSize];
41   vsnprintf(buffer, bufferSize, format, args);
42   printf("[StreamDecoderFFmpeg LOGGING]: %s\n", buffer);
43 }
44 #endif
45
46 // interruptCallBack() function
47 // --------------------
48 int interruptCallBack(void *ptr) {
49   StreamDecoderFFmpeg* stream_decoder = static_cast<StreamDecoderFFmpeg*>(ptr);
50
51   if(stream_decoder->isStoped()) {
```

```
52      return 1; // exit
53    } else if(stream_decoder->reading_packets && toc(stream_decoder->t_read_frame)
        > FRAME_READ_TIMEOUT) {
54      stream_decoder->setError(1);
55      return 1; // exit
56    } else if(stream_decoder->opening_input && toc(stream_decoder->t_open_input) >
        OPEN_INPUT_TIMEOUT) {
57      return 1; // exit
58    }
59    return 0; // continue
60  }
61
62  // consoleLog() function
63  // Function to console log data
64  // --------------------
65  int StreamDecoderFFmpeg::consoleLog(int32_t level, const char* format, ...) {
66    if(level < 0 || level <= debug_level) {
67      if(id > 0) {
68        printf("\033[0;33m[%s StreamDecoderFFmpeg %d]\033[0m ", getCurrentTime().
             c_str(), id);
69      } else {
70        printf("\033[0;33m[%s StreamDecoderFFmpeg]\033[0m ", getCurrentTime().
             c_str());
71      }
72
73      va_list vl;
74      va_start(vl, format);
75      auto ret = vprintf(format, vl);
76      va_end(vl);
77      printf("\n");
78      return ret;
79    }
80    return 0;
81  }
82
83  // listObjOptions() function
84  // Function to list object options
85  // --------------------
86  void StreamDecoderFFmpeg::listObjOptions(void* obj, int req_flags) {
87    consoleLog(-1, "--------------------");
88          consoleLog(-1, "Options for: %s",(*(AVClass**) obj)->class_name);
89          av_opt_show2(obj, stdout, req_flags, NULL);
90          consoleLog(-1, "--------------------");
91  }
92  void StreamDecoderFFmpeg::listObjOptions(const AVClass* obj, int req_flags) {
93    consoleLog(-1, "--------------------");
94          consoleLog(-1, "Options for: %s", obj->class_name);
95          av_opt_show2(&obj, stdout, req_flags, NULL);
96          consoleLog(-1, "--------------------");
97  }
98
99  // StreamDecoderFFmpeg()
100 // Object StreamDecoderFFmpeg
101 // --------------------
102 StreamDecoderFFmpeg::StreamDecoderFFmpeg(const Napi::CallbackInfo& info)
103     : Napi::ObjectWrap<StreamDecoderFFmpeg>(info) {
104 #ifdef DEBUG_STREAM_DECODER_FFMPEG
105   consoleLog(0, "Called constructor");
106 #endif
```

```
107
108   // Get the handle to the current process
109   HANDLE hProcess = GetCurrentProcess();
110
111   // Set the priority class of the process
112   BOOL success = SetPriorityClass(hProcess, REALTIME_PRIORITY_CLASS);
113 #ifdef DEBUG_STREAM_DECODER_FFMPEG
114   if(!success) {
115     consoleLog(0, "Failed to set process priority.");
116   }
117 #endif
118 }
119
120 // StreamDecoderFFmpeg()
121 // Object destructor
122 // --------------------
123 StreamDecoderFFmpeg::~StreamDecoderFFmpeg() {
124
125 }
126
127 // Init() function
128 // --------------------
129 Napi::Object StreamDecoderFFmpeg::Init(Napi::Env env, Napi::Object exports) {
130   Napi::Function func =
131       DefineClass(env,
132                   "StreamDecoderFFmpeg",
133                    {InstanceMethod("start", &StreamDecoderFFmpeg::start),
134                     InstanceMethod("stop", &StreamDecoderFFmpeg::stop),
135                     InstanceMethod("pause", &StreamDecoderFFmpeg::pause),
136                     InstanceMethod("resume", &StreamDecoderFFmpeg::resume),
137                   });
138
139   Napi::FunctionReference* constructor = new Napi::FunctionReference();
140   *constructor = Napi::Persistent(func);
141   env.SetInstanceData(constructor);
142
143   exports.Set("StreamDecoderFFmpeg", func);
144   return exports;
145 }
146
147 // start() function
148 // --------------------
149 Napi::Value StreamDecoderFFmpeg::start(const Napi::CallbackInfo& info) {
150 #ifdef DEBUG_STREAM_DECODER_FFMPEG
151   consoleLog(0, "Call to start()");
152 #endif
153
154   if(cmd_stop_thread) {
155 #ifdef DEBUG_STREAM_DECODER_FFMPEG
156     consoleLog(0, "Previous thread not yet stopped.");
157 #endif
158     return Napi::Boolean::New(info.Env(), false);
159   }
160
161   if(!info[0].IsObject() || !info[1].IsFunction() || !info[2].IsFunction()) {
162 #ifdef DEBUG_STREAM_DECODER_FFMPEG
163     consoleLog(0, "Wrong arguments.");
164 #endif
165     Napi::TypeError::New(info.Env(), "Wrong arguments")
```

```
166         .ThrowAsJavaScriptException();
167       return Napi::Boolean::New(info.Env(), false);
168     }
169
170     Napi::Object options = info[0].As<Napi::Object>();
171     frame_cb = info[1].As<Napi::Function>();
172     error_cb = info[2].As<Napi::Function>();
173
174     if(active) {
175       stop(info);
176     }
177
178     // Handle options
179     if(options.Has("id")) {
180       id = options.Get("id").As<Napi::Number>().Uint32Value();
181 #ifdef DEBUG_STREAM_DECODER_FFMPEG
182       consoleLog(0, "ID set to %d.", id);
183 #endif
184     } else {
185       id = 0;
186     }
187     if(options.Has("debug_level")) {
188       debug_level = options.Get("debug_level").As<Napi::Number>().Int32Value();
189 #ifdef DEBUG_STREAM_DECODER_FFMPEG
190       consoleLog(0, "Debug level set to %d.", debug_level);
191 #endif
192     }
193
194     if(options.Has("input")) {
195       input_address = options.Get("input").As<Napi::String>();
196 #ifdef DEBUG_STREAM_DECODER_FFMPEG
197       consoleLog(0, "Input set to %s.", input_address.c_str());
198 #endif
199     } else {
200 #ifdef DEBUG_STREAM_DECODER_FFMPEG
201       consoleLog(0, "Input must be specified");
202 #endif
203       Napi::TypeError::New(info.Env(), "Input must be specified")
204         .ThrowAsJavaScriptException();
205       return Napi::Boolean::New(info.Env(), false);
206     }
207     if(options.Has("input_format")) {
208       input_format_type = options.Get("input_format").As<Napi::String>();
209 #ifdef DEBUG_STREAM_DECODER_FFMPEG
210       consoleLog(0, "Input format set to %s.", input_format_type.c_str());
211 #endif
212     } else {
213 #ifdef DEBUG_STREAM_DECODER_FFMPEG
214       consoleLog(0, "Input format must be specified");
215 #endif
216       Napi::TypeError::New(info.Env(), "Input format must be specified")
217         .ThrowAsJavaScriptException();
218       return Napi::Boolean::New(info.Env(), false);
219     }
220      if(options.Has("flag_nonblock")) {
221       flag_nonblock = options.Get("flag_nonblock").As<Napi::Boolean>();
222 #ifdef DEBUG_STREAM_DECODER_FFMPEG
223       consoleLog(0, "Flag NONBLOCK set to true.");
224 #endif
```

```
225   } else {
226     flag_nonblock = false;
227   }
228
229   // AVFormatContext options
230   if(options.Has("video_size")) {
231     video_size = options.Get("video_size").As<Napi::String>();
232     av_dict_set(&input_format_options, "video_size", video_size.c_str(), 0);
233 #ifdef DEBUG_STREAM_DECODER_FFMPEG
234     consoleLog(0, "video_size set to %s.", video_size.c_str());
235 #endif
236   }
237   if(options.Has("framerate")) {
238     framerate = options.Get("framerate").As<Napi::String>();
239     av_dict_set(&input_format_options, "framerate", framerate.c_str(), 0);
240 #ifdef DEBUG_STREAM_DECODER_FFMPEG
241     consoleLog(0, "framerate set to %s.", framerate.c_str());
242 #endif
243   }
244   if(options.Has("rtbufsize")) {
245     rtbufsize = options.Get("rtbufsize").As<Napi::String>();
246     av_dict_set(&input_format_options, "rtbufsize", rtbufsize.c_str(), 0);
247 #ifdef DEBUG_STREAM_DECODER_FFMPEG
248     consoleLog(0, "rtbufsize set to %s.", rtbufsize.c_str());
249 #endif
250   }
251   if(options.Has("max_delay")) {
252     max_delay = options.Get("max_delay").As<Napi::String>();
253     av_dict_set(&input_format_options, "max_delay", max_delay.c_str(), 0);
254 #ifdef DEBUG_STREAM_DECODER_FFMPEG
255     consoleLog(0, "max_delay set to %s.", max_delay.c_str());
256 #endif
257   }
258   if(options.Has("avioflags")) {
259     avioflags = options.Get("avioflags").As<Napi::String>();
260     av_dict_set(&input_format_options, "avioflags", avioflags.c_str(), 0);
261 #ifdef DEBUG_STREAM_DECODER_FFMPEG
262     consoleLog(0, "avioflags set to %s.", avioflags.c_str());
263 #endif
264   }
265   if(options.Has("probesize")) {
266     probesize = options.Get("probesize").As<Napi::String>();
267     av_dict_set(&input_format_options, "probesize", probesize.c_str(), 0);
268 #ifdef DEBUG_STREAM_DECODER_FFMPEG
269     consoleLog(0, "probesize set to %s.", probesize.c_str());
270 #endif
271   }
272   if(options.Has("fpsprobesize")) {
273     fpsprobesize = options.Get("fpsprobesize").As<Napi::String>();
274     av_dict_set(&input_format_options, "fpsprobesize", fpsprobesize.c_str(), 0);
275 #ifdef DEBUG_STREAM_DECODER_FFMPEG
276     consoleLog(0, "fpsprobesize set to %s.", fpsprobesize.c_str());
277 #endif
278   }
279   if(options.Has("formatprobesize")) {
280     formatprobesize = options.Get("formatprobesize").As<Napi::String>();
281     av_dict_set(&input_format_options, "formatprobesize", formatprobesize.c_str
          (), 0);
282 #ifdef DEBUG_STREAM_DECODER_FFMPEG
```

```
283     consoleLog(0, "formatprobesize set to %s.", formatprobesize.c_str());
284 #endif
285   }
286   if(options.Has("max_probe_packets")) {
287     max_probe_packets = options.Get("max_probe_packets").As<Napi::String>();
288     av_dict_set(&input_format_options, "max_probe_packets", max_probe_packets.
          c_str(), 0);
289 #ifdef DEBUG_STREAM_DECODER_FFMPEG
290     consoleLog(0, "max_probe_packets set to %s.", max_probe_packets.c_str());
291 #endif
292   }
293   if(options.Has("analyzeduration")) {
294     analyzeduration = options.Get("analyzeduration").As<Napi::String>();
295     av_dict_set(&input_format_options, "analyzeduration", analyzeduration.c_str
          (), 0);
296 #ifdef DEBUG_STREAM_DECODER_FFMPEG
297     consoleLog(0, "analyzeduration set to %s.", analyzeduration.c_str());
298 #endif
299   }
300   if(options.Has("skip_estimate_duration_from_pts")) {
301     skip_estimate_duration_from_pts = options.Get("
          skip_estimate_duration_from_pts").As<Napi::String>();
302     av_dict_set(&input_format_options, "skip_estimate_duration_from_pts",
          skip_estimate_duration_from_pts.c_str(), 0);
303 #ifdef DEBUG_STREAM_DECODER_FFMPEG
304     consoleLog(0, "skip_estimate_duration_from_pts set to %s.",
          skip_estimate_duration_from_pts.c_str());
305 #endif
306   }
307   if(options.Has("max_ts_probe")) {
308     max_ts_probe = options.Get("max_ts_probe").As<Napi::String>();
309     av_dict_set(&input_format_options, "max_ts_probe", max_ts_probe.c_str(), 0);
310 #ifdef DEBUG_STREAM_DECODER_FFMPEG
311     consoleLog(0, "max_ts_probe set to %s.", max_ts_probe.c_str());
312 #endif
313   }
314   if(options.Has("max_interleave_delta")) {
315     max_interleave_delta = options.Get("max_interleave_delta").As<Napi::String
          >();
316     av_dict_set(&input_format_options, "max_interleave_delta",
          max_interleave_delta.c_str(), 0);
317 #ifdef DEBUG_STREAM_DECODER_FFMPEG
318     consoleLog(0, "max_interleave_delta set to %s.", max_interleave_delta.c_str
          ());
319 #endif
320   }
321   if(options.Has("fflags")) {
322     fflags = options.Get("fflags").As<Napi::String>();
323     av_dict_set(&input_format_options, "fflags", fflags.c_str(), 0);
324 #ifdef DEBUG_STREAM_DECODER_FFMPEG
325     consoleLog(0, "fflags set to %s.", fflags.c_str());
326 #endif
327   }
328   if(options.Has("flush_packets")) {
329     flush_packets = options.Get("flush_packets").As<Napi::String>();
330     av_dict_set(&input_format_options, "flush_packets", flush_packets.c_str(),
          0);
331 #ifdef DEBUG_STREAM_DECODER_FFMPEG
332     consoleLog(0, "flush_packets set to %s.", flush_packets.c_str());
```

```
333 #endif
334   }
335
336   // RTSP muxer options
337   if(options.Has("buffer_size")) {
338     buffer_size = options.Get("buffer_size").As<Napi::String>();
339     av_dict_set(&input_format_options, "buffer_size", buffer_size.c_str(), 0);
340 #ifdef DEBUG_STREAM_DECODER_FFMPEG
341     consoleLog(0, "buffer_size set to %s.", buffer_size.c_str());
342 #endif
343   }
344   if(options.Has("rtsp_transport")) {
345     rtsp_transport = options.Get("rtsp_transport").As<Napi::String>();
346     av_dict_set(&input_format_options, "rtsp_transport", rtsp_transport.c_str(),
          0);
347 #ifdef DEBUG_STREAM_DECODER_FFMPEG
348     consoleLog(0, "rtsp_transport set to %s.", rtsp_transport.c_str());
349 #endif
350   }
351   if(options.Has("rtsp_flags ")) {
352     rtsp_flags = options.Get("rtsp_flags").As<Napi::String>();
353     av_dict_set(&input_format_options, "rtsp_flags", rtsp_flags.c_str(), 0);
354 #ifdef DEBUG_STREAM_DECODER_FFMPEG
355     consoleLog(0, "rtsp_flags set to %s.", rtsp_flags.c_str());
356 #endif
357   }
358
359   // Decoder options
360   if(options.Has("dec_thread_count")) {
361     dec_thread_count = options.Get("dec_thread_count").As<Napi::Number>().
         Uint32Value();
362 #ifdef DEBUG_STREAM_DECODER_FFMPEG
363     consoleLog(0, "dec_thread_count set to %d.", dec_thread_count);
364 #endif
365   } else {
366     dec_thread_count = 0;
367   }
368   if(options.Has("dec_flags")) {
369     dec_flags = options.Get("dec_flags").As<Napi::String>();
370 #ifdef DEBUG_STREAM_DECODER_FFMPEG
371     consoleLog(0, "dec_flags set to %s.", dec_flags.c_str());
372 #endif
373   } else {
374     dec_flags.clear();
375   }
376   if(options.Has("dec_flags2")) {
377     dec_flags2 = options.Get("dec_flags2").As<Napi::String>();
378 #ifdef DEBUG_STREAM_DECODER_FFMPEG
379     consoleLog(0, "dec_flags2 set to %s.", dec_flags2.c_str());
380 #endif
381   } else {
382     dec_flags2.clear();
383   }
384
385   // Create a ThreadSafeFunction for the JavaScript callbacks
386   tsfn_frame = Napi::ThreadSafeFunction::New(
387     info.Env(),
388     frame_cb,
389     "ThreadSafeFunctionFrame",
```

```
390       0,
391       2
392     );
393     tsfn_error = Napi::ThreadSafeFunction::New(
394       info.Env(),
395       error_cb,
396       "ThreadSafeFunctionError",
397       0,
398       2
399     );
400
401 #ifdef DEBUG_STREAM_DECODER_FFMPEG
402     consoleLog(0, "Creating new thread");
403 #endif
404     thread = std::thread(&StreamDecoderFFmpeg::threadEntry, this);
405
406     return Napi::Boolean::New(info.Env(), true);
407 }
408
409 // stop() function
410 // --------------------
411 void StreamDecoderFFmpeg::stop(const Napi::CallbackInfo& info) {
412 #ifdef DEBUG_STREAM_DECODER_FFMPEG
413     consoleLog(0, "Call to stop()");
414 #endif
415
416     if(active) {
417       mtx.lock();
418       cmd_stop_thread = true;
419       mtx.unlock();
420       if(thread.joinable()) {
421         thread.join();  // Wait for the thread to finish
422         mtx.lock();
423         cmd_stop_thread = false;
424         mtx.unlock();
425       }
426       mtx.lock();
427       active = false;
428       mtx.unlock();
429
430 #ifdef DEBUG_STREAM_DECODER_FFMPEG
431       consoleLog(0, "Thread stoped!");
432 #endif
433     }
434 }
435
436 // pause() function
437 // --------------------
438 void StreamDecoderFFmpeg::pause(const Napi::CallbackInfo& info) {
439 #ifdef DEBUG_STREAM_DECODER_FFMPEG
440     consoleLog(0, "Call to pause()");
441 #endif
442     if(active) {
443       mtx.lock();
444       cmd_resume = false;
445       cmd_pause = true;
446       mtx.unlock();
447     }
448 #ifdef DEBUG_STREAM_DECODER_FFMPEG
```

```
449   consoleLog(0, "FFmpeg stream decoder paused!");
450 #endif
451 }
452
453 // resume() function
454 // --------------------
455 void StreamDecoderFFmpeg::resume(const Napi::CallbackInfo& info) {
456 #ifdef DEBUG_STREAM_DECODER_FFMPEG
457   consoleLog(0, "Call to resume()");
458 #endif
459   if(active) {
460     mtx.lock();
461     cmd_pause = false;
462     cmd_resume = true;
463     mtx.unlock();
464   }
465 #ifdef DEBUG_STREAM_DECODER_FFMPEG
466   consoleLog(0, "FFmpeg stream decoder resumed!");
467 #endif
468 }
469
470 // setError() function
471 // --------------------
472 void StreamDecoderFFmpeg::setError(uint8_t error_in) {
473 #ifdef DEBUG_USB2530
474   consoleLog(2, "Call to setError()");
475 #endif
476
477   mtx.lock();
478   error = error_in;
479   mtx.unlock();
480 }
481
482 // isStoped() function
483 // --------------------
484 bool StreamDecoderFFmpeg::isStoped() {
485 #ifdef DEBUG_STREAM_DECODER_FFMPEG
486   consoleLog(2, "Call to isStoped()");
487 #endif
488
489   return cmd_stop_thread || error > 0;
490 }
491
492 // threadEntry() function
493 // --------------------
494 void StreamDecoderFFmpeg::threadEntry() {
495 #ifdef DEBUG_STREAM_DECODER_FFMPEG
496   consoleLog(0, "Call to threadEntry()");
497 #endif
498
499   mtx.lock();
500   error = 0;
501   active = true;
502   cmd_stop_thread = false;
503   cmd_pause = false;
504   cmd_resume = false;
505   mtx.unlock();
506
507    // Get the handle to the current thread
```

```
508   HANDLE hThread = GetCurrentThread();
509
510   // Set the priority class of the process
511   BOOL success = SetThreadPriority(hThread, THREAD_PRIORITY_TIME_CRITICAL);
512 #ifdef DEBUG_STREAM_DECODER_FFMPEG
513   if(!success) {
514     consoleLog(0, "Failed to set thread priority.");
515   }
516 #endif
517
518   // Transcoding
519   time_point<steady_clock> t_open;
520
521   AVCodecContext* decode_context = NULL;
522   const AVCodec* decoder = NULL;
523   AVFrame* frame = NULL;
524   AVFrame* rgba_frame = NULL;
525   AVPacket* packet = NULL;
526   SwsContext* rgba_sws_context = NULL;
527   int stream_index = -1;
528   AVCodecParameters* decode_parameters = NULL;
529   AVFormatContext* input_format_context = NULL;
530   const AVInputFormat* input_format = NULL;
531
532   int ret;
533   char a[AV_ERROR_MAX_STRING_SIZE] = { 0 };
534   bool paused = false;
535
536   // Register all available input/output devices
537   avdevice_register_all();
538
539   // Set FFmpeg log level and callback
540 #ifdef DEBUG_FFMPEG
541   av_log_set_level(AV_LOG_DEBUG);
542 #endif
543 #ifdef DEBUG_FFMPEG_CUSTOM_CALLBACK
544   av_log_set_callback(ffmpegLogCallback);
545 #endif
546
547   input_format = av_find_input_format(input_format_type.c_str());
548   if(!input_format) {
549     setError(2);
550   }
551 #ifdef DEBUG_STREAM_DECODER_FFMPEG
552     else {
553     consoleLog(0, "Input format found.");
554   }
555 #endif
556
557   if(!isStoped()) {
558   // Open the input video
559 #ifdef PROFILE_STREAM_DECODER_FFMPEG
560     t_open = tic();
561     consoleLog(1, "Open the input video...");
562 #endif
563
564     mtx.lock();
565     t_open_input = tic();
566     opening_input = true;
```

```
567     mtx.unlock();
568     input_format_context = avformat_alloc_context();
569     input_format_context->interrupt_callback = int_cb;
570     if(flag_nonblock) {
571       input_format_context->flags |= AVFMT_FLAG_NONBLOCK;
572     }
573 
574     ret = avformat_open_input(&input_format_context, input_address.c_str(),
575       input_format, &input_format_options);
576     mtx.lock();
577     opening_input = false;
578     mtx.unlock();
579     if(ret < 0) {
580       setError(3);
581     }
582 #ifdef DEBUG_STREAM_DECODER_FFMPEG
583       else {
584       consoleLog(0, "Input format opened.");
585     }
586 #endif
587 
588 #ifdef LIST_STREAM_DECODER_FFMPEG_OPTIONS
589     listObjOptions(input_format_context, AV_OPT_FLAG_DECODING_PARAM);
590     listObjOptions(input_format->priv_class, AV_OPT_FLAG_DECODING_PARAM);
591 #endif
592   }
593 
594   if(!isStoped()) {
595     // Retrieve stream information
596     ret = avformat_find_stream_info(input_format_context, NULL);
597     if(ret < 0) {
598       setError(4);
599     }
600 #ifdef DEBUG_STREAM_DECODER_FFMPEG
601     else {
602     consoleLog(0, "Stream info retrieved.");
603   }
604 #endif
605   }
606 
607   if(!isStoped()) {
608     for(unsigned int i = 0; i < input_format_context->nb_streams; ++i) {
609       if(input_format_context->streams[i]->codecpar->codec_type ==
           AVMEDIA_TYPE_VIDEO) {
610         stream_index = i;
611         break;
612       }
613     }
614 
615     if(stream_index < 0) {
616       setError(5);
617     }
618 #ifdef DEBUG_STREAM_DECODER_FFMPEG
619       else {
620       consoleLog(0, "Video stream found.");
621     }
622 #endif
623   }
624 
```

```
625   if(!isStoped()) {
626 #ifdef LIST_STREAM_DECODER_FFMPEG_OPTIONS
627     // Print stream information
628     consoleLog(0, "Input format: %s", input_format_context->iformat->name);
629     av_dump_format(input_format_context, stream_index, input_address.c_str(), 0)
          ;
630 #endif
631
632     AVStream* videoStream = input_format_context->streams[stream_index];
633
634     // Get the codec parameters for the video stream
635     decode_parameters = videoStream->codecpar;
636
637     // Find decoder
638     decoder = avcodec_find_decoder(decode_parameters->codec_id);
639     if(decoder == NULL) {
640       setError(6);
641     }
642 #ifdef DEBUG_STREAM_DECODER_FFMPEG
643       else {
644       consoleLog(0, "Video decoder found.");
645     }
646 #endif
647   }
648
649   if(!isStoped()) {
650     // Create a codec context
651     decode_context = avcodec_alloc_context3(decoder);
652     if(decode_context == NULL) {
653       setError(7);
654     }
655 #ifdef DEBUG_STREAM_DECODER_FFMPEG
656       else {
657       consoleLog(0, "Video decoder context created.");
658     }
659 #endif
660   }
661
662   if(!isStoped()) {
663     if(dec_thread_count > 0) {
664       decode_context->thread_count = dec_thread_count;
665     }
666
667     // Copy codec parameters to codec context
668     ret = avcodec_parameters_to_context(decode_context, decode_parameters);
669     if(ret != 0) {
670       setError(8);
671     }
672 #ifdef DEBUG_STREAM_DECODER_FFMPEG
673       else {
674       consoleLog(0, "Codec parameters copied to codec context.");
675     }
676 #endif
677   }
678
679   if(!isStoped()) {
680     AVDictionary* dec_options = NULL;
681     // AVCodecContext options
682     if(!dec_flags.empty()) {
```

```
683       av_dict_set(&dec_options, "flags", dec_flags.c_str(), 0);
684     }
685     if(!dec_flags2.empty()) {
686       av_dict_set(&dec_options, "flags2", dec_flags2.c_str(), 0);
687     }
688 
689     // Open the codec
690     ret = avcodec_open2(decode_context, decoder, &dec_options);
691     if(ret != 0) {
692       setError(9);
693     }
694 #ifdef DEBUG_STREAM_DECODER_FFMPEG
695       else {
696       consoleLog(0, "Video codec opened.");
697     }
698 #endif
699 
700 #ifdef LIST_STREAM_DECODER_FFMPEG_OPTIONS
701     listObjOptions(decode_context, AV_OPT_FLAG_DECODING_PARAM);
702     listObjOptions(decode_context->priv_data, AV_OPT_FLAG_DECODING_PARAM);
703 #endif
704   }
705 
706   if(!isStoped()) {
707 #ifdef PROFILE_STREAM_DECODER_FFMPEG
708     // Time took for open
709     consoleLog(1, "Video opened in %d ms...", toc(t_open));
710 #endif
711 
712     // Allocate video packet
713     packet = av_packet_alloc();
714     if(!packet) {
715       setError(10);
716     }
717 #ifdef DEBUG_STREAM_DECODER_FFMPEG
718       else {
719       consoleLog(0, "Packet allocated.");
720     }
721 #endif
722   }
723 
724   if(!isStoped()) {
725     // Allocate video frame
726     frame = av_frame_alloc();
727     if(!frame) {
728       setError(11);
729     }
730 #ifdef DEBUG_STREAM_DECODER_FFMPEG
731       else {
732       consoleLog(0, "Frame allocated.");
733     }
734 #endif
735   }
736 
737   if(!isStoped()) {
738     // RGBA Scale context
739     rgba_sws_context = sws_getContext(decode_context->width,
740       decode_context->height, decode_context->pix_fmt,
741       decode_context->width, decode_context->height,
```

```
742       AV_PIX_FMT_RGBA, SWS_BICUBIC, NULL, NULL, NULL);
743
744     // Allocate RGBA frame
745     rgba_frame = av_frame_alloc();
746     rgba_frame->format = AV_PIX_FMT_RGBA;
747     rgba_frame->width = decode_context->width;
748     rgba_frame->height = decode_context->height;
749
750     av_frame_get_buffer(rgba_frame, 0);
751
752     avcodec_flush_buffers(decode_context);
753
754     // Loop over frames and write them to the output file
755     uint32_t frame_number = 1;
756 #ifdef PROFILE_STREAM_DECODER_FFMPEG
757     auto t_packet = tic();
758 #endif
759
760 #ifdef DEBUG_STREAM_DECODER_FFMPEG
761     consoleLog(0, "Start reading frames!");
762 #endif
763     mtx.lock();
764     reading_packets = true;
765     mtx.unlock();
766     while(true) {
767 #ifdef DEBUG_STREAM_DECODER_FFMPEG
768       consoleLog(2, "Read loop");
769 #endif
770
771       // Check for stop
772       if(isStoped()) {
773         break;
774       }
775
776       // Check for pause or resume
777       if(cmd_pause) {
778         if(!paused) {
779           paused = true;
780           mtx.lock();
781           reading_packets = false;
782           mtx.unlock();
783 #ifdef CLOSE_INPUT_FORMAT
784           avformat_close_input(&input_format_context);
785 #else
786           av_read_pause(input_format_context);
787 #endif
788         }
789         continue;
790       } else if(paused) {
791         if(cmd_resume) {
792           paused = false;
793 #ifdef CLOSE_INPUT_FORMAT
794           input_format_context = avformat_alloc_context();
795           input_format_context->interrupt_callback = int_cb;
796           if(flag_nonblock) {
797             input_format_context->flags |= AVFMT_FLAG_NONBLOCK;
798           }
799
800           ret = avformat_open_input(&input_format_context, input_address.c_str()
```

```
                ,
801               input_format, &input_format_options);
802             mtx.lock();
803             opening_input = false;
804             mtx.unlock();
805             if(ret < 0) {
806               setError(3);
807             }
808 #else
809             av_read_play(input_format_context);
810 #endif
811           } else {
812             continue;
813           }
814         }
815 
816         // Capture a frame
817         mtx.lock();
818         reading_packets = true;
819         t_read_frame = tic();
820         mtx.unlock();
821         ret = av_read_frame(input_format_context, packet);
822         if(isStoped()) {
823           break;
824         }
825 
826         if(ret < 0) {
827 #ifdef DEBUG_STREAM_DECODER_FFMPEG
828           consoleLog(0, "Failed to read frame: %s",
829             av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
830 #endif
831           if(isStoped()) {
832             break;
833           } else {
834             continue;
835           }
836         }
837 
838         if(packet->stream_index == stream_index) {
839 #ifdef PROFILE_STREAM_DECODER_FFMPEG
840           auto t_decode = tic();
841           consoleLog(1, "--- StreamDecoderFFmpeg profiler ---");
842           consoleLog(1, "New packet %d, %d ms", frame_number, toc(t_packet));
843           t_packet = tic();
844 #endif
845           ret = avcodec_send_packet(decode_context, packet);
846           if(ret < 0) {
847             consoleLog(0, "Packet sending error: %s",
848               av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
849             break;
850           }
851 
852           while(ret >= 0) {
853             ret = avcodec_receive_frame(decode_context, frame);
854             if(ret == AVERROR(EAGAIN) || ret == AVERROR_EOF) {
855               break;
856             } else if(ret < 0) {
857              consoleLog(0, "Failed to receive frame: %s",
858                 av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
```

```
859             break;
860           }
861
862 #ifdef PROFILE_STREAM_DECODER_FFMPEG
863           consoleLog(1, "Decoded frame %d", frame_number);
864 #endif
865           sws_scale(rgba_sws_context, frame->data, frame->linesize, 0,
866             frame->height,rgba_frame->data, rgba_frame->linesize);
867
868           uint8_t* data = rgba_frame->data[0];
869           int size = rgba_frame->width * rgba_frame->height * 4;
870           uint32_t packet_size = packet->size * 8;
871
872 #ifdef DEBUG_STREAM_DECODER_FFMPEG
873           consoleLog(2, "Sending frame...");
874 #endif
875           auto callback = [data, size, frame_number, packet_size](Napi::Env env,
                Napi::Function jsCallback) {
876             jsCallback.Call({Napi::Buffer<uint8_t>::Copy(env, data, size),
877               Napi::Number::New(env, frame_number), Napi::Number::New(env,
                  packet_size)});
878           };
879           if(!isStoped()) {
880             tsfn_frame.BlockingCall(callback);
881           }
882         }
883 #ifdef PROFILE_STREAM_DECODER_FFMPEG
884         consoleLog(1, "Frame decode time %d ms", toc(t_decode));
885         fflush(stdout);
886 #endif
887       }
888
889       frame_number++;
890
891       av_packet_unref(packet);
892
893       if(isStoped()) {
894         break;
895       }
896     }
897   }
898   mtx.lock();
899   reading_packets = false;
900   mtx.unlock();
901
902   // Clean up resources
903 #ifdef DEBUG_STREAM_DECODER_FFMPEG
904   consoleLog(0, "Cleaning up resources!");
905 #endif
906   av_frame_free(&frame);
907   av_packet_free(&packet);
908
909   sws_freeContext(rgba_sws_context);
910
911   avcodec_free_context(&decode_context);
912   avformat_close_input(&input_format_context);
913
914   if(error > 0) {
915     std::string message;
```

```
916       switch(error) {
917         case 1:
918           message = "Frame read timeout";
919           break;
920         case 2:
921           message = "Failed to find input format";
922           break;
923         case 3:
924           message = "Failed to open input video: " +
925             std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
926           break;
927         case 4:
928           message = "Failed to retrieve stream information: " +
929             std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
930           break;
931         case 5:
932           message = "Failed to find video stream";
933           break;
934         case 6:
935           message = "Failed to find decoder";
936           break;
937         case 7:
938           message = "Failed to create input codec context";
939           break;
940         case 8:
941           message = "Could not copy input codec parameters: " +
942             std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
943           break;
944         case 9:
945           message = "Failed to open input codec: " +
946             std::string(av_make_error_string(a, AV_ERROR_MAX_STRING_SIZE, ret));
947           break;
948         case 10:
949           message = "Could not allocate input packet";
950           break;
951         case 11:
952           message = "Could not allocate video frame";
953           break;
954       }
955
956   #ifdef DEBUG_STREAM_DECODER_FFMPEG
957       consoleLog(0, "On error:");
958       consoleLog(0, " -> code = %d", error);
959       consoleLog(0, " -> msg = %s", message.c_str());
960   #endif
961
962       uint8_t code = error;
963       auto callback = [code, message](Napi::Env env, Napi::Function jsCallback) {
964         jsCallback.Call({Napi::Number::New(env, code), Napi::String::New(env,
              message)});
965       };
966       tsfn_error.BlockingCall(callback);
967     }
968
969     tsfn_frame.Release();
970     tsfn_error.Release();
971
972   #ifdef DEBUG_STREAM_DECODER_FFMPEG
973     consoleLog(0, "Thread function end!");
```

```
974 #endif
975 }
976
977 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
978   return StreamDecoderFFmpeg::Init(env, exports);
979 }
980
981 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
982
983 }  // namespace stream_decoder_ffmpeg_ns
```

Listing AB.11 - `stream-decoder-ffmpeg.cpp`

AB.3 Modul stream-decoder-gst

Da bi se komapjlirao ovaj modul neophodno je da je dostupna biblioteka `GStreamer`.

Datoteka `package.json` (listing AB.12), sadrži osnovne informacije o paketu i njegovoj instalaciji, uključujući definisanje drugih potrebnih paketa.

```
1 {
2   "dependencies": {
3     "node-addon-api": "8.2.1"
4   }
5 }
```

Listing AB.12 - `stream-decoder-gst package.json`

Datoteka `binding.gyp` (listing AB.13) pruža konfiguraciju za `node-gyp`, alat koji omogućava kompajliranje `C++` koda u binarni modul kompatibilan sa `Node.js` okruženjem.

```
1 {
2   "targets": [
3     {
4       "target_name": "stream_decoder_gst",
5       "sources": [
6         "stream-decoder-gst.cpp"
7       ],
8       "include_dirs": [
9         "<(module_root_dir)",
10        "<!@(node -p \"require('node-addon-api').include\")",
11        "<(module_root_dir)/gstreamer-1.20.4/include",
12        "<(module_root_dir)/gstreamer-1.20.4/include/gstreamer-1.0",
13        "<(module_root_dir)/gstreamer-1.20.4/include/glib-2.0",
14        "<(module_root_dir)/gstreamer-1.20.4/lib/glib-2.0/include"
15      ],
16      "libraries": [
17        "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0.lib",
18        "<(module_root_dir)/gstreamer-1.20.4/lib/gobject-2.0.lib",
19        "<(module_root_dir)/gstreamer-1.20.4/lib/glib-2.0.lib",
20        "<(module_root_dir)/gstreamer-1.20.4/lib/gstapp-1.0.lib"
21      ],
22      "cflags!": [ "-fno-exceptions" ],
23      "cflags_cc!": [ "-fno-exceptions" ],
24      "defines": ["NAPI_DISABLE_CPP_EXCEPTIONS"],
25      "copies":[{
26        "destination":"<(module_root_dir)/build/Release",
27        "files":[
28            "<(module_root_dir)/gstreamer-1.20.4/bin/avcodec-58.dll",
```

```
            "<(module_root_dir)/gstreamer-1.20.4/bin/avfilter-7.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/avformat-58.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/avutil-56.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/swresample-3.dll",

            "<(module_root_dir)/gstreamer-1.20.4/bin/bz2.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/ffi-7.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gio-2.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/glib-2.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gmodule-2.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gobject-2.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstapp-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstaudio-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstbase-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstnet-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstpbutils-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstreamer-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstrtp-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstrtsp-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstsdp-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gsttag-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/gstvideo-1.0-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/intl-8.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/orc-0.4-0.dll",
            "<(module_root_dir)/gstreamer-1.20.4/bin/z-1.dll",

            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/gstapp.dll",
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/gstrtp.dll",
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/
               gstcoreelements.dll",
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/gstlibav.dll"
               ,
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/gstrtsp.dll",
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/gstudp.dll",
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/gsttcp.dll",
            "<(module_root_dir)/gstreamer-1.20.4/lib/gstreamer-1.0/
               gstvideoconvert.dll"
          ]
        }],
        "msvs_settings": {
          "VCCLCompilerTool": {
            "AdditionalOptions": [ "-std:c++17"]
          }
        }
      }
    ]
}
```

Listing AB.13 - `stream-decoder-gst binding.gyp`

Implementacija modula `stream-decoder-gst` je data u listinzima AB.14 i AB.15.

```
// StreamDecoderGST - stream-decoder-gst.h
// @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
// Doktorska disertacija
// Mašinski fakultet Univerziteta u Beogradu
// --------------------

#ifndef STREAM_DECODER_GST_H
#define STREAM_DECODER_GST_H

```

```
10 //#define DEBUG_STREAM_DECODER_GST
11 #define DEBUG_STREAM_DECODER_GST_LEVEL 0
12 //#define PROFILE_STREAM_DECODER_GST
13
14 #include <napi.h>
15 #include <chrono>
16 #include <thread>
17 #include <Windows.h>
18 #include <ctime>
19 #include <iomanip>
20 #include <sstream>
21 #include <string>
22 #include <mutex>
23
24 #include <gst/gst.h>
25 #include <gst/app/gstappsink.h>
26
27 namespace stream_decoder_gst_ns {
28
29 using namespace std;
30 using namespace std::chrono;
31
32 class StreamDecoderGST : public Napi::ObjectWrap<StreamDecoderGST> {
33  public:
34   static Napi::Object Init(Napi::Env env, Napi::Object exports);
35   StreamDecoderGST(const Napi::CallbackInfo& info);
36   ~StreamDecoderGST();
37
38   Napi::Value start(const Napi::CallbackInfo& info);
39   void stop(const Napi::CallbackInfo& info);
40   void pause(const Napi::CallbackInfo& info);
41   void resume(const Napi::CallbackInfo& info);
42
43   Napi::Function frame_cb;
44   Napi::Function error_cb;
45
46   Napi::ThreadSafeFunction tsfn_frame;
47   Napi::ThreadSafeFunction tsfn_error;
48
49   int consoleLog(int32_t, const char*, ...);
50
51   void setError(uint8_t);
52   bool isStopped();
53
54   bool busProcessMsg(GstElement*, GstMessage*);
55
56   void threadEntry();
57   void threadBusEntry();
58
59   std::thread thread;
60   std::thread thread_bus;
61
62   uint32_t id = 0;
63   int32_t debug_level = DEBUG_STREAM_DECODER_GST_LEVEL;
64   uint32_t packet_size = 0;
65
66   void handleHandoff(GstElement*, GstBuffer*);
67   static void handoffCallback(GstElement*, GstBuffer*, GstPad*, gpointer);
68
```

```
69    GstElement *pipeline = nullptr;
70    GstElement *sink_video = nullptr;
71
72    bool started = false;
73    bool active = false;
74    uint8_t error = 0;
75    bool opening_input = false;
76    bool reading_packets = false;
77
78    bool cmd_stop_thread = false;
79    bool cmd_pause = false;
80    bool cmd_resume = false;
81
82    std::string input_address;
83    std::string protocol;
84    std::string codec;
85    std::string latency;
86    std::string jitter_latency;
87    uint32_t video_width;
88    uint32_t video_height;
89  };
90
91  }// namespace stream_decoder_gst_ns
92
93  #endif // STREAM_DECODER_GST_H
```

Listing AB.14 - `stream-decoder-gst.h`

```
1   // StreamDecoderGST - stream-decoder-gst.cpp
2   // @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
3   // Doktorska disertacija
4   // Mašinski fakultet Univerziteta u Beogradu
5   // --------------------
6
7   #include "stream-decoder-gst.h"
8
9   std::mutex mtx; // mutex for critical section
10
11  namespace stream_decoder_gst_ns {
12
13  // Function to start the timer and return the start time
14  time_point<steady_clock> tic() {
15      return steady_clock::now();
16  }
17
18  // Function to stop the timer and return the elapsed time
19  long toc(const time_point<steady_clock>& startTime) {
20      return duration_cast<milliseconds>(steady_clock::now() - startTime).count();
21  }
22
23  // Function to get current time
24  std::string getCurrentTime() {
25    auto now = system_clock::now();
26    auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
27    auto timer = system_clock::to_time_t(now);
28    std::tm bt = *std::localtime(&timer);
29    std::ostringstream oss;
30    oss << std::put_time(&bt, "%H:%M:%S"); // HH:MM:SS
31    oss << '.' << std::setfill('0') << std::setw(3) << ms.count();
32    return oss.str();
```

```
33 }
34
35 // consoleLog() function
36 // Function to console log data
37 // --------------------
38 int StreamDecoderGST::consoleLog(int32_t level, const char* format, ...) {
39   if(level < 0 || level <= debug_level) {
40     if(id > 0) {
41       printf("\033[0;33m[%s StreamDecoderGST %d]\033[0m ", getCurrentTime().
             c_str(), id);
42     } else {
43       printf("\033[0;33m[%s StreamDecoderGST]\033[0m ", getCurrentTime().c_str()
             );
44     }
45
46     va_list vl;
47     va_start(vl, format);
48     auto ret = vprintf(format, vl);
49     va_end(vl);
50     printf("\n");
51     return ret;
52   }
53   return 0;
54 }
55
56 // StreamDecoderGST()
57 // Object constructor
58 // --------------------
59 StreamDecoderGST::StreamDecoderGST(const Napi::CallbackInfo& info)
60     : Napi::ObjectWrap<StreamDecoderGST>(info) {
61
62   // Get the handle to the current process
63   HANDLE hProcess = GetCurrentProcess();
64
65   // Set the priority class of the process
66   BOOL success = SetPriorityClass(hProcess, HIGH_PRIORITY_CLASS);
67 #ifdef DEBUG_STREAM_DECODER_GST
68   if(!success) {
69     consoleLog(0, "Failed to set process priority.");
70   }
71 #endif
72
73   // Get the current directory.
74   char currentDir[FILENAME_MAX];
75   GetCurrentDirectory(FILENAME_MAX, currentDir);
76
77   // Get the current PATH environment variable.
78   char* pathEnv;
79   size_t len;
80   _dupenv_s(&pathEnv, &len, "PATH");
81
82   // Append the current directory to the PATH.
83   std::string newPath = std::string(pathEnv) + ";" + std::string(currentDir);
84
85   // Set the new PATH for the current process.
86   SetEnvironmentVariable("PATH", newPath.c_str());
87 }
88
89 // StreamDecoderGST()
```

```
90  // Object destructor
91  // --------------------
92  StreamDecoderGST::~StreamDecoderGST() {
93
94  }
95
96  // Init() function
97  // --------------------
98  Napi::Object StreamDecoderGST::Init(Napi::Env env, Napi::Object exports) {
99    Napi::Function func =
100       DefineClass(env,
101                   "StreamDecoderGST",
102                    {InstanceMethod("start", &StreamDecoderGST::start),
103                     InstanceMethod("stop", &StreamDecoderGST::stop),
104                     InstanceMethod("pause", &StreamDecoderGST::pause),
105                     InstanceMethod("resume", &StreamDecoderGST::resume),
106                   });
107
108   Napi::FunctionReference* constructor = new Napi::FunctionReference();
109   *constructor = Napi::Persistent(func);
110   env.SetInstanceData(constructor);
111
112   exports.Set("StreamDecoderGST", func);
113   return exports;
114 }
115
116 // start() function
117 // --------------------
118 Napi::Value StreamDecoderGST::start(const Napi::CallbackInfo& info) {
119 #ifdef DEBUG_STREAM_DECODER_GST
120   consoleLog(0, "Call to start()");
121 #endif
122
123   if(cmd_stop_thread) {
124 #ifdef DEBUG_STREAM_DECODER_GST
125     consoleLog(0, "Previous thread not yet stopped.");
126 #endif
127     return Napi::Boolean::New(info.Env(), false);
128   }
129
130   if(!info[0].IsObject() ||!info[1].IsFunction() || !info[2].IsFunction()) {
131 #ifdef DEBUG_STREAM_DECODER_GST
132     consoleLog(0, "Wrong arguments.");
133 #endif
134     Napi::TypeError::New(info.Env(), "Wrong arguments")
135       .ThrowAsJavaScriptException();
136     return Napi::Boolean::New(info.Env(), false);
137   }
138
139   Napi::Object options = info[0].As<Napi::Object>();
140   frame_cb = info[1].As<Napi::Function>();
141   error_cb = info[2].As<Napi::Function>();
142
143   if(active) {
144     stop(info);
145   }
146
147   // Handle options
148   if(options.Has("id")) {
```

```
149     id = options.Get("id").As<Napi::Number>().Uint32Value();
150 #ifdef DEBUG_STREAM_DECODER_GST
151     consoleLog(0, "ID set to %d.", id);
152 #endif
153   } else {
154     id = 0;
155   }
156   if(options.Has("debug_level")) {
157     debug_level = options.Get("debug_level").As<Napi::Number>().Int32Value();
158 #ifdef DEBUG_STREAM_DECODER_GST
159     consoleLog(0, "Debug level set to %d.", debug_level);
160 #endif
161   }
162 
163   if(options.Has("input")) {
164     input_address = options.Get("input").As<Napi::String>();
165 #ifdef DEBUG_STREAM_DECODER_GST
166     consoleLog(0, "Input set to %s.", input_address.c_str());
167 #endif
168   } else {
169 #ifdef DEBUG_STREAM_DECODER_GST
170     consoleLog(0, "Input must be specified");
171 #endif
172     Napi::TypeError::New(info.Env(), "Input must be specified")
173       .ThrowAsJavaScriptException();
174     return Napi::Boolean::New(info.Env(), false);
175   }
176 
177   if(options.Has("protocol")) {
178     protocol = options.Get("protocol").As<Napi::String>();
179 #ifdef DEBUG_STREAM_DECODER_GST
180     consoleLog(0, "Protocol set to %s.", protocol.c_str());
181 #endif
182   } else {
183 #ifdef DEBUG_STREAM_DECODER_GST
184     consoleLog(0, "Protocol must be specified");
185 #endif
186     Napi::TypeError::New(info.Env(), "Protocol must be specified")
187       .ThrowAsJavaScriptException();
188     return Napi::Boolean::New(info.Env(), false);
189   }
190   if(options.Has("codec")) {
191     codec = options.Get("codec").As<Napi::String>();
192 #ifdef DEBUG_STREAM_DECODER_GST
193     consoleLog(0, "Codec set to %s.", codec.c_str());
194 #endif
195     if(codec != "H264" && codec != "H265" && codec != "HEVC") {
196 #ifdef DEBUG_STREAM_DECODER_GST
197       consoleLog(0, "Codec must be H264, H265 or HEVC!");
198 #endif
199       Napi::TypeError::New(info.Env(), "Codec must be H264, H265 or HEVC!")
200         .ThrowAsJavaScriptException();
201       return Napi::Boolean::New(info.Env(), false);
202     }
203   } else {
204 #ifdef DEBUG_STREAM_DECODER_GST
205     consoleLog(0, "Codec must be specified");
206 #endif
207     Napi::TypeError::New(info.Env(), "Codec must be specified")
```

```
208         .ThrowAsJavaScriptException();
209       return Napi::Boolean::New(info.Env(), false);
210     }
211
212     if(options.Has("video_width")) {
213       video_width = options.Get("video_width").As<Napi::Number>().Uint32Value();
214 #ifdef DEBUG_STREAM_DECODER_GST
215       consoleLog(0, "Video width set to %d.", video_width);
216 #endif
217     } else {
218       video_width = 1280;
219     }
220     if(options.Has("video_height")) {
221       video_height = options.Get("video_height").As<Napi::Number>().Uint32Value();
222 #ifdef DEBUG_STREAM_DECODER_GST
223       consoleLog(0, "Video height set to %d.", video_height);
224 #endif
225     } else {
226       video_height = 720;
227     }
228     if(options.Has("latency")) {
229       latency = options.Get("latency").As<Napi::String>();
230 #ifdef DEBUG_STREAM_DECODER_GST
231       consoleLog(0, "Latency set to %s.", latency);
232 #endif
233     } else {
234       latency = "0";
235     }
236     if(options.Has("jitter_latency")) {
237       jitter_latency = options.Get("jitter_latency").As<Napi::String>();
238 #ifdef DEBUG_STREAM_DECODER_GST
239       consoleLog(0, "Jitter latency set to %s.", jitter_latency);
240 #endif
241     } else {
242       jitter_latency = "50";
243     }
244
245     // Init gstreamer
246     gst_init(nullptr, nullptr);
247
248     string codec_params;
249     if(codec == "H265" || codec == "HEVC") {
250       codec_params = "rtph265depay ! h265parse ! avdec_h265";
251     } else if(codec == "H264") {
252       codec_params = "rtph264depay ! h264parse ! avdec_h264";
253     }
254
255     // Set up the pipeline
256     string pipeStr = "rtspsrc protocols="+protocol+" location="+input_address+"
          latency="+latency+" ! queue leaky=downstream max-size-buffers=1 !
          rtpjitterbuffer latency="+jitter_latency+" drop-on-latency=true ! identity
          signal-handoffs=true name=myidentity ! "+codec_params+" ! videoconvert !
          appsink name=mysink max-buffers=1 sync=false caps=""video/x-raw,format=RGBA
          """;
257 #ifdef DEBUG_STREAM_DECODER_GST
258     consoleLog(0, "Pipeline: %s.", pipeStr.c_str());
259 #endif
260
261     GError *err = nullptr;
```

```
262   pipeline = gst_parse_launch(pipeStr.c_str(), &err);
263   if(err) {
264 #ifdef DEBUG_STREAM_DECODER_GST
265     consoleLog(0, "Error: %s", err->message);
266 #endif
267     return Napi::Boolean::New(info.Env(), false);
268   }
269
270   // Find our appsink by name
271   sink_video = gst_bin_get_by_name(GST_BIN(pipeline), "mysink");
272
273   // Play the pipeline
274   gst_element_set_state(pipeline, GST_STATE_PLAYING);
275
276   // Find the identity element in the pipeline
277   GstElement *identity = gst_bin_get_by_name(GST_BIN(pipeline), "myidentity");
278
279   // Connect to the "handoff" signal of the identity element
280   g_signal_connect(identity, "handoff", G_CALLBACK(StreamDecoderGST::
        handoffCallback), this);
281
282   // Create a ThreadSafeFunction for the JavaScript callbacks
283   tsfn_frame = Napi::ThreadSafeFunction::New(
284     info.Env(),
285     frame_cb,
286     "ThreadSafeFunctionFrame",
287     0,
288     2
289   );
290   tsfn_error = Napi::ThreadSafeFunction::New(
291     info.Env(),
292     error_cb,
293     "ThreadSafeFunctionError",
294     0,
295     2
296   );
297
298 #ifdef DEBUG_STREAM_DECODER_GST
299   consoleLog(0, "Creating new threads");
300 #endif
301   thread = std::thread(&StreamDecoderGST::threadEntry, this);
302   thread_bus = std::thread(&StreamDecoderGST::threadBusEntry, this);
303
304   started = true;
305   return Napi::Boolean::New(info.Env(), true);
306 }
307
308 // handoffCallback() function
309 // Static method to serve as a trampoline
310 // --------------------
311 void StreamDecoderGST::handoffCallback(GstElement* identity, GstBuffer* buffer,
        GstPad* pad, gpointer user_data) {
312   StreamDecoderGST* self = static_cast<StreamDecoderGST*>(user_data);
313   self->handleHandoff(identity, buffer);
314 }
315
316 // handleHandoff() function
317 // --------------------
318 void StreamDecoderGST::handleHandoff(GstElement* identity, GstBuffer* buffer) {
```

```
319   GstMapInfo map;
320   if (gst_buffer_map(buffer, &map, GST_MAP_READ)) {
321     packet_size += map.size;
322     gst_buffer_unmap(buffer, &map);
323   }
324 }
325
326 // stop() function
327 // --------------------
328 void StreamDecoderGST::stop(const Napi::CallbackInfo& info) {
329 #ifdef DEBUG_STREAM_DECODER_GST
330   consoleLog(0, "Call to stop()");
331 #endif
332
333   if(started) {
334     if(active) {
335       mtx.lock();
336       cmd_stop_thread = true;
337       mtx.unlock();
338       if(thread.joinable()) {
339         thread.join();
340         thread_bus.join();
341         mtx.lock();
342         cmd_stop_thread = false;
343         mtx.unlock();
344       }
345       mtx.lock();
346       active = false;
347       mtx.unlock();
348
349 #ifdef DEBUG_STREAM_DECODER_GST
350       consoleLog(0, "Thread stoped!");
351 #endif
352     }
353
354     // Destroy the pipeline
355     gst_element_set_state(pipeline, GST_STATE_NULL);
356     gst_object_unref(pipeline);
357   }
358 }
359
360 // pause() function
361 // --------------------
362 void StreamDecoderGST::pause(const Napi::CallbackInfo& info) {
363 #ifdef DEBUG_STREAM_DECODER_GST
364   consoleLog(0, "Call to pause()");
365 #endif
366   if(active) {
367     mtx.lock();
368     cmd_resume = false;
369     cmd_pause = true;
370     gst_element_set_state(pipeline, GST_STATE_PAUSED);
371     mtx.unlock();
372   }
373 #ifdef DEBUG_STREAM_DECODER_GST
374   consoleLog(0, "GStreamer stream decoder paused!");
375 #endif
376 }
377
```

```
378 // resume() function
379 // --------------------
380 void StreamDecoderGST::resume(const Napi::CallbackInfo& info) {
381 #ifdef DEBUG_STREAM_DECODER_GST
382   consoleLog(0, "Call to resume()");
383 #endif
384   if(active) {
385     mtx.lock();
386     cmd_pause = false;
387     cmd_resume = true;
388     gst_element_set_state(pipeline, GST_STATE_PLAYING);
389     mtx.unlock();
390   }
391 #ifdef DEBUG_STREAM_DECODER_GST
392   consoleLog(0, "GStreamer stream decoder resumed!");
393 #endif
394 }
395
396 // setError() function
397 // --------------------
398 void StreamDecoderGST::setError(uint8_t error_in) {
399 #ifdef DEBUG_USB2530
400   consoleLog(2, "Call to setError()");
401 #endif
402
403   mtx.lock();
404   error = error_in;
405   mtx.unlock();
406 }
407
408 // isStopped() function
409 // --------------------
410 bool StreamDecoderGST::isStopped() {
411 #ifdef DEBUG_STREAM_DECODER_GST
412   consoleLog(2, "Call to isStopped()");
413 #endif
414
415   return cmd_stop_thread || error > 0;
416 }
417
418 // busProcessMsg() function
419 // Process a single bus message, log messages, exit on error, return false on
      eof
420 // --------------------
421 bool StreamDecoderGST::busProcessMsg(GstElement *pipeline, GstMessage *msg) {
422   GstMessageType mType = GST_MESSAGE_TYPE(msg);
423
424 #ifdef DEBUG_STREAM_DECODER_GST
425   consoleLog(0, "mType: %i", mType);
426   switch (mType) {
427     case (GST_MESSAGE_ERROR):
428       // Parse error and exit program, hard exit
429       GError *err;
430       gchar *dbg;
431       gst_message_parse_error(msg, &err, &dbg);
432       consoleLog(0, "Error: %s from %s", err->message, GST_OBJECT_NAME(msg->src)
          );
433       consoleLog(0, "Debug: %s", dbg);
434       g_clear_error(&err);
```

```
435       g_free(dbg);
436     case (GST_MESSAGE_EOS) :
437       // Soft exit on EOS
438       consoleLog(0, "End of stream!");
439     case (GST_MESSAGE_STATE_CHANGED):
440       // Parse state change, print extra info for pipeline only
441       consoleLog(0, "State changed!");
442       if(GST_MESSAGE_SRC(msg) == GST_OBJECT(pipeline)) {
443         GstState sOld, sNew, sPenging;
444         gst_message_parse_state_changed(msg, &sOld, &sNew, &sPenging);
445         consoleLog(0, "Pipeline changed from %s to %s",
              gst_element_state_get_name(sOld), gst_element_state_get_name(sNew));
446       }
447       break;
448     case (GST_MESSAGE_STEP_START):
449       consoleLog(0, "Step start!");
450       break;
451     case (GST_MESSAGE_STREAM_STATUS):
452       consoleLog(0, "Stream status!");
453       break;
454     case (GST_MESSAGE_ELEMENT):
455       consoleLog(0, "Message element!");
456       break;
457
458       // You can add more stuff here if you want
459   }
460 #endif
461   return true;
462 }
463
464 // threadBusEntry() function
465 // --------------------
466 void StreamDecoderGST::threadBusEntry() {
467 #ifdef DEBUG_STREAM_DECODER_GST
468   consoleLog(0, "Call to threadBusEntry()");
469 #endif
470
471   GstBus *bus = gst_element_get_bus(pipeline);
472
473   int res;
474   while (true) {
475
476     if (isStopped()) {
477       break;
478     }
479
480     GstMessage *msg = gst_bus_timed_pop(bus, 100 * GST_MSECOND);
481
482     if (msg != NULL) {
483       res = busProcessMsg(pipeline, msg);
484       gst_message_unref(msg);
485
486       if (!res)
487         break;
488     }
489   }
490
491   gst_object_unref(bus);
492 #ifdef DEBUG_STREAM_DECODER_GST
```

```
493   consoleLog(0, "Bus thread function end!");
494 #endif
495 }
496
497 // threadEntry() function
498 // --------------------
499 void StreamDecoderGST::threadEntry() {
500 #ifdef DEBUG_STREAM_DECODER_GST
501   consoleLog(0, "Call to threadEntry()");
502 #endif
503
504   mtx.lock();
505   error = 0;
506   active = true;
507   cmd_stop_thread = false;
508   cmd_pause = false;
509   cmd_resume = false;
510   mtx.unlock();
511
512    // Get the handle to the current thread
513   HANDLE hThread = GetCurrentThread();
514
515   // Set the priority class of the process
516   BOOL success = SetThreadPriority(hThread, THREAD_PRIORITY_HIGHEST);
517 #ifdef DEBUG_STREAM_DECODER_GST
518   if(!success) {
519     consoleLog(0, "Failed to set thread priority.");
520   }
521 #endif
522
523   uint32_t frame_number = 1;
524
525 #ifdef DEBUG_STREAM_DECODER_FFMPEG
526   consoleLog(0, "Start reading frames!");
527 #endif
528   mtx.lock();
529   reading_packets = true;
530   mtx.unlock();
531   while(true) {
532 #ifdef DEBUG_STREAM_DECODER_FFMPEG
533     consoleLog(2, "Read loop");
534 #endif
535
536     // Check for stop
537     if(isStopped()) {
538       break;
539     }
540
541     if(!cmd_pause) {
542       // Exit on EOS
543       if(gst_app_sink_is_eos(GST_APP_SINK(sink_video))) {
544 #ifdef DEBUG_STREAM_DECODER_FFMPEG
545         consoleLog(0, "End of stream!");
546 #endif
547         break;
548       }
549
550       // Capture a frame
551       mtx.lock();
```

```
552       reading_packets = true;
553       mtx.unlock();
554       GstSample *sample = gst_app_sink_pull_sample(GST_APP_SINK(sink_video));
555       if(sample == nullptr) {
556 #ifdef DEBUG_STREAM_DECODER_FFMPEG
557         consoleLog(0, "No frame!");
558 #endif
559         break;
560       }
561 
562       GstBuffer *buffer = gst_sample_get_buffer(sample);
563       GstMapInfo m;
564       gst_buffer_map(buffer, &m, GST_MAP_READ);
565 
566       uint8_t* data = m.data;
567       int size = video_width*video_height*4;
568       uint32_t packet_size_last = packet_size;
569       packet_size = 0;
570 
571 #ifdef DEBUG_STREAM_DECODER_FFMPEG
572       consoleLog(2, "Sending frame...");
573 #endif
574       auto callback = [data, size, frame_number, packet_size_last](Napi::Env env
              , Napi::Function jsCallback) {
575         jsCallback.Call({Napi::Buffer<uint8_t>::Copy(env, data, size),
576           Napi::Number::New(env, frame_number), Napi::Number::New(env,
                 packet_size_last)});
577       };
578       if(!isStopped()) {
579         tsfn_frame.BlockingCall(callback);
580       }
581 
582       frame_number++;
583 
584       // Don't forget to unmap the buffer and unref the sample
585       gst_buffer_unmap(buffer, &m);
586       gst_sample_unref(sample);
587     } else {
588       reading_packets = false;
589     }
590   }
591   mtx.lock();
592   reading_packets = false;
593   mtx.unlock();
594 
595   if(error > 0) {
596     std::string message;
597 #ifdef DEBUG_STREAM_DECODER_FFMPEG
598     consoleLog(0, "On error:");
599     consoleLog(0, " -> code = %d", error);
600     consoleLog(0, " -> msg = %s", message.c_str());
601 #endif
602 
603     uint8_t code = error;
604     auto callback = [code, message](Napi::Env env, Napi::Function jsCallback) {
605       jsCallback.Call({Napi::Number::New(env, code), Napi::String::New(env,
             message)});
606     };
607     tsfn_error.BlockingCall(callback);
```

```
608     }
609
610     tsfn_frame.Release();
611     tsfn_error.Release();
612
613 #ifdef DEBUG_STREAM_DECODER_GST
614     consoleLog(0, "Thread function end!");
615 #endif
616 }
617
618 Napi::Object InitAll(Napi::Env env, Napi::Object exports) {
619     return StreamDecoderGST::Init(env, exports);
620 }
621
622 NODE_API_MODULE(NODE_GYP_MODULE_NAME, InitAll)
623
624 }  // namespace transcoder_gst_ns
```

Listing AB.15 - `stream-decoder-gst.cpp`

AC

Skripta serveOculus.jsl

Skriptom `serveOculus.jsl`, koja je u celosti data u listingu AC.1, se pokreće `HTTP` server i klijentima servira fajl `index.html` iz listinga AC.2 za potrebe testiranja upotrebe uređaja za virtuelnu realnost.

```
/**
 * @file Serve Oculus
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var hostname = 'localhost';
var port = 8000;
var device_id = "
    d0049e1ff308815f53b8dc50682d86a99009ce8aeb918f154820641b39f8cb90";
var data_type = ["default", "jpg"][1];
// --------------------

// android-platform-tools i oculus-go-adb-driver-2.0
// adb devices
// adb reverse tcp:8000 tcp:8000
//getWebcams();

var win = await openCanvas();
win.setTitle('Capture test');
var ctx = win.canvas.getContext('2d');
var first = true;
var client;

function broadcast(data) {
  if(client) {
    client.write(`data: ${data}\n\n`);
  }
}

function frameCallback(data, width, height, frame) {
  if(first) {
    first = false;
    win.canvas.width = width;
    win.canvas.height = height;
```

```
  }

  ctx.drawImage(frame, 0, 0, width, height);
  if(data_type == 'default') {
    var data = win.canvas.toDataURL();
  } else {
    var data = win.canvas.toDataURL('image/jpeg', 0.7);
  }
  broadcast(data);
}

webcamCapture({ id:  device_id}, frameCallback);

const http = require('http');

const server = http.createServer((req, res) => {
  if(req.url == '/data') {
    res.writeHead(200, {
      'Content-Type': 'text/event-stream',
      'Cache-Control': 'no-cache',
      'Connection': 'keep-alive',
      'Access-Control-Allow-Origin': '*',
    });
    res.write('\n');
    client = res;
    console.log(`Client connected`);
  } else {
    res.statusCode = 200;
    var file_path = pwd + req.url.substring(1);
    if(req.url === '/') {
      file_path = pwd + 'index.html';
    }
    res.setHeader('Content-Type', getContentType(file_path));
    var contents = readFile(file_path);
    res.end(contents);
  }
});

server.listen(port, '0.0.0.0', () => {
  disp(`Server running at http://${hostname}:${port}/`);
});

addForCleanup(this, function() {
  server.close();
});
```

Listing AC.1 - `serveOculus.jsl`

```
<html>
  <head>
    <script src="./aframe.min.js"></script>
  </head>
  <body>
    <a-scene id="my-scene">
      <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
      <a-sphere position="2 5 -5" radius="1.25" color="#EF2D5E"></a-sphere>
      <a-cylinder
        position="1 0.75 -3"
        radius="0.5"
        height="1.5"
```

```
13          color="#FFC65D"
14        ></a-cylinder>
15        <a-entity
16          position="-2 1.25 -5"
17          scale="5 5 5"
18          rotation="0 0 0"
19          gltf-model="odonatOptim.glb"
20          animation-mixer
21        ></a-entity>
22        <a-plane
23          position="0 0 -4"
24          rotation="-90 0 0"
25          width="4"
26          height="4"
27          color="#7BC8A4"
28        ></a-plane>
29
30        <a-image id="dynamic-image" src="" position="0 1.6 -5" width="4" height="3
              "></a-image>
31
32        <a-sky color="#ECECEC"></a-sky>
33
34        <a-entity id="player" position="0 0 0">
35          <a-camera id="player-camera" track-orientation camera="" position=""
                rotation="" look-controls="" wasd-controls=""><a-entity geometry=""
                material="" visible=""></a-entity>
36          <a-text
37              id="rotation-text"
38              value="TEST"
39              position="0 0 -2"
40              align="center"
41              color="#000"
42              width="2">
43            </a-text>
44          </a-camera>
45
46          <a-entity id="left-controller" oculus-touch-controls="hand: left"></a-
                entity>
47          <a-entity id="right-controller" oculus-touch-controls="hand: right"></a-
                entity>
48          <a-entity laser-controls="hand: left"></a-entity>
49          <a-entity laser-controls="hand: right"></a-entity>
50        </a-entity>
51      </a-scene>
52
53    <script>
54      var scene_el = document.getElementById('my-scene');
55      var rotation_text_el = document.getElementById('rotation-text');
56      var right_controller_el = document.getElementById('right-controller');
57      var dynamic_image = document.getElementById('dynamic-image');
58
59      AFRAME.registerComponent('track-orientation', {
60        tick: function() {
61          var rotation = this.el.object3D.rotation;
62          rotation_text_el.setAttribute('value', JSON.stringify(rotation));
63        }
64      });
65
66      right_controller_el.addEventListener('bbuttondown', () => {
```

```
67        scene_el.exitVR();
68      });
69
70      var event_source = new EventSource('./data');
71
72      event_source.onmessage = function(event) {
73        dynamic_image.setAttribute('src', event.data);
74      };
75    </script>
76
77    </body>
78  </html>
```

Listing AC.2 - `serveOculus index.html`

AD

Skripta showModel3D.jsl za vizualizaciju simulatora leta pomoću Three.js

U listingu AD.1 je data skripta `showModel3D.jsl` kojom se vrši vizualizacija simulatora leta pomoću `Three.js` biblioteke.

```
/**
 * @file Create 3D model
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var id = 'MP_6DOF_XXX'; // ID mehanizma
var show_cockpit = true;
var cockpit_scale = 10;
var x = [ // translacija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  0 // [deg] psi
];
var d1 = 30;
var d2 = 20;
var D = 50;
// --------------------

var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
var MP = new MP_6RSS_MECHANISM();

var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
var solver = new MP_6RSS_SOLVER(MP);

// Odredjivanje geometrijskih karakteristika
disp(' Odredjivanje geometrijskih karakteristika...');
solver.getGeometry();

var P0 = MP.P;

```

```
// Odredjivanje potrebnog ugla alpha za date podatke
disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
solver.setParameters(x);

// Provera uglova u zglobovim
disp(' Provera uglova u zglobovima...');
solver.checkJoints();

if(any(MP.e)) {
  disp(' Neuspešan proračun!');
  dispRowVector('x', MP.x, MP.m);
  dispRowVector('e', MP.e, MP.m);
  onEnd();
} else {
  disp(' Uspešan proračun!');
  dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);

  disp(' Modeliranje geometrije mehanizma leta...');

  // Variables translation for initial node and line calculations
  // Nodes and diameters array
  var nodes = [
    ...MP.A,
    ...MP.B,
    ...MP.Pb,
  ];
  nodes = reshape(transpose(nodes, nodes.length / 3, 3) , nodes.length / 3, 3);
  var Di = Array(nodes.length).fill(D);
  var C = Array(nodes.length).fill(0xFF9900);

  // Lines definitions
  var lines1 = concatCol(2*6,
    concatRow(6, MP.A, MP.B),
    concatRow(6, MP.A, MP.Pb),
  );
  var d1i = Array(lines1.length / 6).fill(d1);
  var c1 = Array(lines1.length / 6).fill(0x595959);

  // Additional lines definitions
  var lines2 = concatCol(2*6,
    concatRow(6, MP.B, movelr(MP.B, 3)),
    concatRow(6, MP.Pb, movelr(MP.Pb, 3))
  );
  var d2i = Array(lines2.length / 6).fill(d2);
  var c2 = Array(lines2.length / 6).fill(0x0099FF);
  var L = [MP.a*1.1, D*1.5];
  var T = [-MP.a*1.1, -D*1.5/2];

  // Combine lines and diameters
  var lines = [...lines1, ...lines2];
  lines = reshape(transpose(lines, lines.length / 6, 6) , lines.length / 6, 6);
  var di = [...d1i, ...d2i];
  var c = [...c1, ...c2];

  // Open window for 3D model and show it
  (async function() {
    var win = await openWindow3D([
      {import: 'OrbitControls', from: 'three/addons/controls/OrbitControls.js'},
      {import: 'GLTFLoader ', from: 'three/addons/loaders/GLTFLoader.js'}
```

```
98      ]);
99      win.document.title = id + " - JSLAB | PR-DC";
100     var THREE = win.THREE;
101
102     var width = win.innerWidth;
103     var height = win.innerHeight;
104
105     const frustumSize = 2000;
106     const aspect = width /height;
107
108     const camera = new THREE.OrthographicCamera(
109       -frustumSize * aspect / 2,
110       frustumSize * aspect / 2,
111       frustumSize / 2,
112       -frustumSize / 2,
113       -8000,
114       8000
115     );
116     camera.up.set(0, 0, 1);
117     camera.position.set(0, -1000, 1000);
118     camera.lookAt(new THREE.Vector3(0, 0, 500));
119
120     const scene = new THREE.Scene();
121     scene.background = new THREE.Color(0xffffff);
122
123     const ambientLight = new THREE.AmbientLight(0xffffff, 1.0);
124     scene.add(ambientLight);
125
126     const directionalLight = new THREE.DirectionalLight(0xffffff, 1.0);
127     directionalLight.position.set(1000, -1000, 1000); // Position the light
128     directionalLight.target.position.set(0, 0, 500); // Point towards the center
129     scene.add(directionalLight);
130     scene.add(directionalLight.target);
131
132     // Enable shadow casting for the light
133     directionalLight.castShadow = true;
134
135     // Set shadow properties
136     directionalLight.shadow.mapSize.width = 2048; // Increase shadow quality
137     directionalLight.shadow.mapSize.height = 2048;
138     directionalLight.shadow.camera.near = 0.5;
139     directionalLight.shadow.camera.far = 5000;
140
141     directionalLight.shadow.camera.left = -frustumSize * aspect / 2;
142     directionalLight.shadow.camera.right = frustumSize * aspect / 2;
143     directionalLight.shadow.camera.top = frustumSize / 2;
144     directionalLight.shadow.camera.bottom = -frustumSize / 2;
145
146     nodes.forEach((node, i) => {
147       const geometry = new THREE.SphereGeometry(Di[i] / 2, 32, 32);
148       const material = new THREE.MeshPhongMaterial({ color: C[i] });
149       const sphere = new THREE.Mesh(geometry, material);
150       sphere.position.set(node[0], node[1], node[2]);
151       scene.add(sphere);
152     });
153
154     const renderer = new THREE.WebGLRenderer( { antialias: true } );
155     renderer.setSize( width, height );
156     renderer.setAnimationLoop( animate );
```

```
157     win.document.body.appendChild( renderer.domElement );
158
159     lines.forEach((line, i) => {
160       const startPoint = new THREE.Vector3(line[0], line[1], line[2]);
161       const endPoint = new THREE.Vector3(line[3], line[4], line[5]);
162       const direction = new THREE.Vector3().subVectors(endPoint, startPoint);
163       const length = direction.length();
164
165       const axis = new THREE.Vector3(0, 1, 0);
166       const quaternion = new THREE.Quaternion().setFromUnitVectors(axis,
            direction.clone().normalize());
167
168       const geometry = new THREE.CylinderGeometry(di[i] / 2, di[i] / 2, length,
            8, 1, false);
169       const material = new THREE.MeshPhongMaterial({ color: c[i] });
170       const cylinder = new THREE.Mesh(geometry, material);
171       const midpoint = new THREE.Vector3().addVectors(startPoint, endPoint).
            multiplyScalar(0.5);
172       cylinder.position.copy(midpoint);
173       cylinder.applyQuaternion(quaternion);
174       scene.add(cylinder);
175     });
176
177     var material = new THREE.MeshPhongMaterial({
178       color: 0x0099FF,
179       opacity: 0.7,
180       transparent: true
181     });
182     var points = [...MP.B, ...P0];
183     for(var i = 0; i < 2; i++) {
184       var shape = new THREE.Shape();
185       shape.moveTo(points[3 * 6 * i], points[3 * 6 * i + 1]);
186       for(var j = 0; j < 6; j++) {
187         shape.lineTo(points[3 * 6 * i + j * 3], points[3 * 6 * i + j * 3 + 1]);
188       }
189       shape.lineTo(points[3 * 6 * i], points[3 * 6 * i + 1]);
190       const extrudeSettings = {
191         steps: 1,
192         depth: L[i],
193         bevelEnabled: false,
194       };
195       const geometry = new THREE.ExtrudeGeometry(shape, extrudeSettings);
196       geometry.translate(0, 0, points[3 * 6 * i + 2] + T[i]);
197       var mesh = new THREE.Mesh(geometry, material);
198       scene.add(mesh);
199     }
200     mesh.rotation.x = x[3]*d2r;
201     mesh.rotation.y = x[4]*d2r;
202     mesh.rotation.z = x[5]*d2r;
203     mesh.position.x = MP.POb[3];
204     mesh.position.y = MP.POb[4];
205     mesh.position.z = MP.POb[5];
206
207     const controls = new win.OrbitControls(camera, renderer.domElement);
208     controls.target.set(0, 0, 500);
209     controls.update();
210
211     // Create a grid helper
212     const size = 5000; // Size of the grid
```

```
213     const divisions = 10; // Number of divisions
214     const gridColor1 = 0xcccccc; // Gray lines
215     const gridColor2 = 0xcccccc; // White lines
216 
217     const gridHelper = new THREE.GridHelper(size, divisions, gridColor1,
            gridColor2);
218     gridHelper.rotation.x = pi/2;
219     gridHelper.position.z = -L[0];
220     scene.add(gridHelper);
221 
222     if(show_cockpit) {
223       const loader = new win.GLTFLoader();
224       loader.load(pwd + 'resources/cockpit.glb', function(gltf) {
225         const model = gltf.scene;
226         model.rotation.x = pi/2+x[3]*d2r;
227         model.rotation.y = pi+x[4]*d2r;
228         model.rotation.z = x[5]*d2r;
229         model.scale.x = cockpit_scale;
230         model.scale.y = cockpit_scale;
231         model.scale.z = cockpit_scale;
232         model.position.x = MP.POb[3];
233         model.position.y = MP.POb[4];
234         model.position.z = MP.POb[5]+L[1]/2;
235         scene.add(model);
236       });
237     }
238 
239     renderer.shadowMap.enabled = true;
240     directionalLight.castShadow = true;
241 
242     win.addEventListener('resize', onWindowResize, false);
243 
244     function onWindowResize() {
245       var aspect = win.innerWidth / win.innerHeight;
246 
247       camera.left = -frustumSize * aspect / 2;
248       camera.right = frustumSize * aspect / 2;
249       camera.top = frustumSize / 2;
250       camera.bottom = -frustumSize / 2;
251       camera.updateProjectionMatrix();
252       controls.update();
253       renderer.setSize(win.innerWidth, win.innerHeight);
254     }
255 
256     function animate( time ) {
257       renderer.render( scene, camera );
258     }
259 
260     onEnd();
261   })();
262 }
263 
264 function onEnd() {
265   // - Kraj proračuna
266   disp(' Program je uspešno izvršen... ');
267   disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
268   disp(' -------------------- ');
269 }
```

Listing AD.1 - showModel3D.jsl

AE

Skripte solveInverseDynamicsOpenModelica.jsl za analizu mehanizma pomoću OpenModelica programa

U listingu AE.1 je data skripta `solveInverseDynamicsOpenModelica.jsl` kojom se vrši uporedna analiza rezultata proračuna iz programskog paketa `JSLAB`, na osnovu već prikazanih metoda, i programskog paketa `OpenModelica`. U listingu AE.2 je dat šablon `OpenModelica` modela koji se koristi za analizu nakon unosa odgovarajućih vrednosti parametara.

Da bi se omogućila ovakva analiza u JSLAB-u je napisana klasa `PRDC_JSLAB_OPENMODELICA_LINK` koja je integrisana u sam program. Ova klasa implementira sve neophodne metode za pokretanje `OpenModelica` programa i komunikaciju sa njim u cilju pokretanja proračuna i dobijanja rezultata.

```
/**
 * @file Solve Inverse Dynamics
 * @author Milos Petrasinovic <mpetrasinovic@mas.bg.ac.rs>
 * Doktorska disertacija
 * Mašinski fakultet Univerziteta u Beogradu
 * @version 0.0.1
 */
close('all'); clear; clc; tic;
disp(' --- ' + jsl_file_name + ' ---');
// - Ulazni podaci
var recalculate = true;
var save_figs = 1; // [-] Selektor za čuvanje grafika
var figure_size = [560, 420]; // [px] Veličina grafika
var id = 'MP_6DOF_XXX'; // ID mehanizma
var x = [ // translacija i rotacija
  0, // [mm] x
  0, // [mm] y
  0, // [mm] z
  0, // [deg] phi
  0, // [deg] theta
  0 // [deg] psi
];
// Karakteristike reduktora
var x_R = 100; // [mm] polozaj napadne tacke sile duz vratila
var a_R = 162; // [mm] a konstanta reduktora
```

```
26 var b_R = 122; // [mm] b konstanta reduktora
27 var Fr_R = 5000; // [N] maksimalna dozvoljena radijalna sila
28 // OpenModelica podesavanja
29 var exe = "C:/Program Files/OpenModelica1.14.1-64bit/bin/omc.exe";
30 var modelName = 'modelDinamikaParametarski';  // naziv OpenModelica modela
31 var model = pwd + '/resources/' + modelName + '.mo'; // putanja do modela
32 var stopTime = 1; // vreme simulacije
33 // OpenModelica parametri
34 var p = {};
35 p.table1 = '[0, 0; 0.1, from_deg(0); 0.3, from_deg(0); 0.7, from_deg(60); 0.7,
       from_deg(60)]';
36 p.table2 = '[0, 0; 0.1, from_deg(60); 0.2, from_deg(60); 0.8, from_deg(0); 0.8,
       from_deg(0)]';
37 p.table3 = '[0, 0; 0.1, from_deg(60); 0.2, from_deg(60); 0.8, from_deg(0); 0.8,
       from_deg(0)]';
38 p.table4 = '[0, 0; 0.1, from_deg(0); 0.2, from_deg(0); 0.8, from_deg(60); 0.8,
       from_deg(60)]';
39 p.table5 = '[0, 0; 0.1, from_deg(60); 0.2, from_deg(60); 0.8, from_deg(0); 0.8,
       from_deg(0)]';
40 p.table6 = '[0, 0; 0.1, from_deg(0); 0.3, from_deg(0); 0.7, from_deg(60); 0.7,
       from_deg(60)]';
41 p.D_M = 60; // [mm]
42 p.L_M = 150; // [mm]
43 p.M_M = 0.0001; // [kg]
44 p.D_B = 80; // [mm]
45 p.L_B = 80; // [mm]
46 p.D_BA = 60; // [mm]
47 p.D_BAs = 80; // [mm]
48 p.M_BA = 0.0001; // [kg]
49 p.D_A = 80; // [mm]
50 p.D_AP = 60; // [mm]
51 p.D_APs = 80; // [mm]
52 p.M_AP = 0.0001; // [kg]
53 p.D_P = 80; // [mm]
54 p.H_P = 60; // [mm]
55 p.D_Op = 150; // [mm]
56 // --------------------
57
58 var jsl_file_basename = removeExtension(jsl_file_name);
59
60 var { MP_6RSS_MECHANISM } = require('./configs/'+id+'.jsl');
61 var MP = new MP_6RSS_MECHANISM();
62 var { MP_6RSS_SOLVER } = require('./classes/SOLVER_MP_6RSS.jsl');
63 var solver = new MP_6RSS_SOLVER(MP);
64 var { MP_6RSS_PLOTTER } = require('./classes/PLOTTER_MP_6RSS.jsl');
65 var plotter = new _MP_6RSS_PLOTTER(MP);
66
67 // Odredjivanje geometrijskih karakteristika
68 disp(' Odredjivanje geometrijskih karakteristika...');
69 solver.getGeometry();
70
71 // Odredjivanje potrebnog ugla alpha za date podatke
72 disp(' Odredjivanje potrebnih uglova otklona donjih poluga...');
73 solver.setParameters(x);
74 solver.getJacobian();
75 solver.setAlpha(MP.alpha);
76
77 // Provera uglova u zglobovim
78 disp(' Provera uglova u zglobovima...');
```

```
79  solver.checkJoints();
80
81  // Odredjivanje dozvoljenih sila za reduktor
82  disp(' Odredjivanje dozvoljenih sila za reduktor...');
83  var [Frd_R, Fad_R] = solver.getActuatorAllowedLoad(Fr_R, a_R, b_R, x_R);
84  disp(' Dozvoljena radijalna sila reduktora: ' + num2str(Frd_R, 2) + ' N');
85  disp(' Dozvoljena aksijalna sila reduktora: ' + num2str(Fad_R, 2) + ' N');
86
87  // Podešavanje OpenModelica modela
88  disp(' Podešavanje OpenModelica modela...');
89  var Op0 = [0, 0, MP.z0];
90  var mstr = readFile(pwd + '/resources/' + modelName + '_template.mo', 'utf8');
91  var m36s = (x) => sprintf('[%.16f,%.16f,%.16f,%.16f,%.16f,%.16f;%.16f,%.16f,%.16
        f,%.16f,%.16f,%.16f;%.16f,%.16f,%.16f,%.16f,%.16f,%.16f]', ...transpose(x, 6,
         3));
92  mstr = strrep(mstr, '$Op0$', sprintf('[%.16f;%.16f;%.16f]', ...Op0));
93  mstr = strrep(mstr, '$Ab$', m36s(MP.A));
94  mstr = strrep(mstr, '$B$', m36s(MP.B));
95  mstr = strrep(mstr, '$Pb$', m36s(MP.Pb));
96  mstr = strrep(mstr, '$i_m$', m36s(MP.i_m));
97  p.m = MP.M;
98  p.g = norm(MP.g);
99  p.M_I_11 = MP.I[0];
100 p.M_I_21 = MP.I[1];
101 p.M_I_22 = MP.I[4];
102 p.M_I_31 = MP.I[2];
103 p.M_I_32 = MP.I[5];
104 p.M_I_33 = MP.I[8];
105 p.M_I_33 = MP.I[8];
106 p.cg = '{'+(MP.cg[0]/1000)+', '+(MP.cg[1]/1000)+', '+(MP.cg[2]/1000)+'}';
107 var fn = fieldnames(p);
108 for(var i = 0; i < fn.length; i++) {
109   mstr = strrep(mstr, '$' + fn[i] + '$', p[fn[i]]);
110 }
111 writeFile(model, mstr);
112
113 if(any(MP.e)) {
114   disp(' Neuspešan proračun!');
115   dispRowVector('x', MP.x, MP.m);
116   dispRowVector('e', MP.e, MP.m);
117 } else {
118   disp(' Uspešan proračun!');
119   dispRowVector('alpha', multiplyEl(MP.alpha, r2d), MP.m);
120
121   disp(' Prikaz mehanizma...');
122   plotter.show();
123
124   if(recalculate) {
125     // Run OpenModelicaLink
126     await om_link.start(exe); // Start OpenModelica programa
127
128     // Otvaranje i kompajliranje OpenModelica modela
129     disp(' Otvaranje i kompajliranje OpenModelica modela...');
130     await om_link.ModelicaSystem(model, modelName);
131
132     // Podesavanje OpenModelica parametara
133     disp(' Podesavanje OpenModelica parametara...');
134     var opts = ['stopTime=' + num2str(stopTime, '%.16f')];
135     om_link.setSimulationOptions(opts);
```

```
136     await om_link.simulate();
137     var res = await om_link.getSolutions([
138       'time',
139       'B1.phi', 'B2.phi', 'B3.phi', 'B4.phi', 'B5.phi', 'B6.phi',
140       'B1.w', 'B2.w', 'B3.w', 'B4.w', 'B5.w', 'B6.w',
141       'B1.a', 'B2.a', 'B3.a', 'B4.a', 'B5.a', 'B6.a',
142       'B1.axis.tau', 'B2.axis.tau', 'B3.axis.tau', 'B4.axis.tau', 'B5.axis.tau',
              'B6.axis.tau',
143       'M.v_0[1]', 'M.v_0[2]', 'M.v_0[3]',
144       'M.a_0[1]', 'M.a_0[2]', 'M.a_0[3]',
145       'M.z_a[1]', 'M.z_a[2]', 'M.z_a[3]',
146       'M.w_a[1]', 'M.w_a[2]', 'M.w_a[3]',
147     ]);
148     var res = res.map((r) => r.map((e) => Number(e)));
149     await om_link.close();
150     save(pwd+'results/'+jsl_file_basename+'.json', 'res');
151   } else {
152     load(pwd+'results/'+jsl_file_basename+'.json');
153   }
154
155   disp(' Proracun JSLAB...');
156   var v_pi = [];
157   var a_pi = [];
158   var epsilon_pi = [];
159   var M_mi = [];
160   for(var i = 0; i < res[0].length; i++) {
161
162     var alpha = [res[1][i], res[2][i], res[3][i],
163                  res[4][i], res[5][i], res[6][i]];
164     solver.setAlpha(scale(alpha, r2d));
165
166     var omega = [res[7][i], res[8][i], res[9][i],
167                  res[10][i], res[11][i], res[12][i]];
168     var epsilon = [res[13][i], res[14][i], res[15][i],
169                    res[16][i], res[17][i], res[18][i]];
170
171     var BA = divideEl(minus(MP.A, MP.B), 1000);
172     var P = divideEl(MP.P, 1000);
173     var cg = divideEl(multiply(MP.R, MP.cg, 3, 3, 1), 1000);
174     var s = MP.s/1000;
175     var AP = divideEl(minus(MP.Pb, MP.A), 1000);
176     var OpP = divideEl(minus(MP.Pb,
177       repCol(getSub(MP.POb, range(3, 5)), MP.m)), 1000);
178     var J_a = divideEl(MP.J_a, 1000);
179     var J_x = cloneArray(MP.J_x);
180     setSub(J_x, range(18, 35), divideEl(getSub(J_x, range(18, 35)), 1000));
181     var invJ_x = inv(J_x, MP.m);
182     var J = multiply(invJ_x, J_a, MP.m, MP.m, MP.m);
183
184     var v_A = multiplyEl(cross3D(MP.i_m, BA, MP.m), repRow(omega, 3));
185
186     var dot_x = multiply(J, omega, 6, 6, 1);
187     var v_p = getSub(dot_x, range(0, 2));
188     var omega_p = getSub(dot_x, range(3, 5));
189     v_pi.push(v_p);
190
191     var v_P = plus(repCol(v_p, MP.m), cross3D(repCol(omega_p, MP.m), OpP, MP.m))
          ;
192     var omega_ul = scale(cross3D(minus(v_P, v_A), AP, MP.m),
```

```
193       -1/pow(s, 2));
194
195     var dot_J_x = transpose(concatRow(6, cross3D(omega_ul, MP.i_ul, MP.m),
196       plus(cross3D(cross3D(repCol(omega_p, MP.m), OpP, MP.m), MP.i_ul, MP.m),
197       cross3D(OpP, cross3D(omega_ul, MP.i_ul, MP.m), MP.m))), MP.m, MP.m);
198
199     var dot_J_ai = plus(dot(cross3D(omega_ul, MP.i_ul, MP.m),
200       cross3D(MP.i_m, BA, MP.m), 6), dot(MP.i_ul, cross3D(MP.i_m,
201       cross3D(multiplyEl(repRow(omega, 3), MP.i_m), BA, MP.m), MP.m), 6));
202     var dot_J_a = diag(dot_J_ai, MP.m);
203
204     var dot2_x = multiply(invJ_x, plus(
205         multiply(dot_J_a, omega, MP.m, MP.m, 1),
206         multiply(J_a, epsilon, MP.m, MP.m, 1),
207         scale(multiply(dot_J_x, dot_x, MP.m, MP.m, 1), -1)
208       ), MP.m, MP.m, 1);
209
210     var a_p = getSub(dot2_x, range(0, 2));
211     var epsilon_p = getSub(dot2_x, range(3, 5));
212     a_pi.push(a_p);
213     epsilon_pi.push(epsilon_p);
214
215     var a_CG = plus(a_p, cross3D(epsilon_p, cg, 1),
216       cross3D(omega_p, cross3D(omega_p, cg, 1)));
217
218     var F = scale(minus(MP.g, a_CG), MP.M);
219     var T = minus(cross3D(cg, F, 1),
220       plus(multiply(MP.I, epsilon_p, 3, 3, 1),
221       cross3D(omega_p, multiply(MP.I, omega_p, 3, 3, 1), 1)));
222     var F_x = concat(F, T);
223     var M_m = scale(multiply(transpose(J, 6, 6), F_x, 6, 6, 1), -1);
224     M_mi.push(M_m);
225   }
226
227   // - Prikaz dijagrama
228   disp(' Prikaz dijagrama...');
229   var fj = 0;
230
231   for(var i = 0; i < 6; i++) {
232     fj = fj+1;
233     figure(fj);
234     plot([
235       {x: res[0], y: res[19+i], name: 'OpenModelica '},
236       {x: res[0], y: column(M_mi, i), name: 'JSLAB'}
237     ], {'font': {family: 'LatinModern', size: 18}});
238     xlabel('t [s]');
239     ylabel('M<sub>'+(i+1)+'</sub> [Nm]');
240   }
241
242   for(var i = 0; i < 6; i++) {
243     fj = fj+1;
244     figure(fj);
245     plot([
246       {x: res[0], y: res[19+i], name: 'OpenModelica '},
247       {x: res[0], y: column(M_mi, i), name: 'JSLAB'}
248     ], {'font': {family: 'LatinModern', size: 18}});
249     xlabel('t [s]');
250     ylabel('M<sub>'+(i+1)+'</sub> [Nm]');
251     ylim([-500, 500]);
```

```
252   }
253 }
254
255 // - Čuvanje dijagrama
256 if(save_figs && fj) {
257   disp(' Čuvanje dijagrama... ');
258   mkdir(pwd+'figures');
259   for(var i = 1; i <= fj; i++) {
260     figure(i);
261     printFigure(pwd+'figures/'+jsl_file_basename+'-'+num2str(i, 0), {type: 'png'
          , size: figure_size});
262     printFigure(pwd+'figures/'+jsl_file_basename+'-'+num2str(i, 0), {type: 'pdf'
          , size: figure_size});
263   }
264 }
265
266 // - Kraj proračuna
267 disp(' Program je uspešno izvršen... ');
268 disp(' Potrebno vreme: ' + num2str(toc(), 2) + ' sekundi');
269 disp(' -------------------- ');
```

Listing AE.1 - `solveInverseDynamicsOpenModelica.jsl`

```
1  model modelDinamikaParametarski "Inverzna dinamika mehanizma simulatora leta"
2    import SI = Modelica.SIunits;
3    import Modelica.SIunits.Conversions.*;
4
5    parameter SI.Mass m(min = 0) = $m$ "Masa tereta i platforme";
6    parameter SI.Acceleration g = $g$ "[m/s^2] gravitaciono ubrzanje";
7    parameter SI.Position cg[3] = $cg$ "Koordinate tezista tereta ";
8    parameter Real M_I_11 = $M_I_11$;
9    parameter Real M_I_21 = $M_I_21$;
10   parameter Real M_I_22 = $M_I_22$;
11   parameter Real M_I_31 = $M_I_31$;
12   parameter Real M_I_32 = $M_I_32$;
13   parameter Real M_I_33 = $M_I_33$;
14   parameter Real table1[:, 2] = $table1$;
15   parameter Real table2[:, 2] = $table2$;
16   parameter Real table3[:, 2] = $table3$;
17   parameter Real table4[:, 2] = $table4$;
18   parameter Real table5[:, 2] = $table5$;
19   parameter Real table6[:, 2] = $table6$;
20   parameter SI.Position Op0[3, 1] = $Op0$ / 1000 "Pocetni polozaj platforme";
21   parameter SI.Position Ab[3, 6] = $Ab$ / 1000 "Polozaj tacaka Ai";
22   parameter SI.Position B[3, 6] = $B$ / 1000 "Polozaj tacaka Bi";
23   parameter SI.Position Pb[3, 6] = $Pb$ / 1000 "Polozaj tacaka Pi";
24   parameter Real i_m[3, 6] = $i_m$ "Jedinicni vektor ose aktuatora";
25   parameter SI.Position BA[3, 6] = Ab - B;
26   parameter SI.Position AP[3, 6] = Pb - Ab;
27   parameter SI.Position POp[3, 6] = Op0 * ones(1, 6) - Pb;
28   parameter SI.Position Pij[3, 6] = [Pb[:,2:6]-Pb[:,1:5],Pb[:,1]-Pb[:,6]];
29   parameter SI.Diameter D_M(min = 0) = $D_M$ / 1000;
30   parameter SI.Length L_M(min = 0) = $L_M$ / 1000;
31   parameter SI.Mass M_M(min = 0) = $M_M$;
32   parameter SI.Diameter D_B(min = 0) = $D_B$ / 1000;
33   parameter SI.Length L_B(min = 0) = $L_B$ / 1000;
34   parameter SI.Diameter D_BA(min = 0) = $D_BA$ / 1000;
35   parameter SI.Diameter D_BAs(min = 0) = $D_BAs$ / 1000;
36   parameter SI.Mass M_BA(min = 0) = $M_BA$;
37   parameter SI.Diameter D_A(min = 0) = $D_A$ / 1000;
```

```
38  parameter SI.Diameter D_AP(min = 0) = $D_AP$ / 1000;
39  parameter SI.Diameter D_APs(min = 0) = $D_APs$ / 1000;
40  parameter SI.Mass M_AP(min = 0) = $M_AP$;
41  parameter SI.Diameter D_P(min = 0) = $D_P$ / 1000;
42  parameter SI.Length H_P(min = 0) = $H_P$ / 1000;
43  parameter SI.Diameter D_Op(min = 0) = $D_Op$ / 1000;
44  parameter SI.Position r_Mb[3, 6] = -i_m * L_M / 2;
45
46  inner Modelica.Mechanics.MultiBody.World world(g = g, n = {0, 0, -1})
        annotation(
47    Placement(visible = true, transformation(origin = {-116, 82}, extent =
        {{-10, -10}, {10, 10}}, rotation = 0)));
48  Modelica.Mechanics.MultiBody.Parts.Fixed F1(animation = false, r = B[:, 1])
        annotation(
49    Placement(visible = true, transformation(origin = {-74, 56}, extent = {{-4,
        -4}, {4, 4}}, rotation = -90)));
50  Modelica.Mechanics.MultiBody.Parts.Fixed F2(animation = false, r = B[:, 2])
        annotation(
51    Placement(visible = true, transformation(origin = {-74, 32}, extent = {{-4,
        -4}, {4, 4}}, rotation = -90)));
52  Modelica.Mechanics.MultiBody.Parts.Fixed F3(animation = false, r = B[:, 3])
        annotation(
53    Placement(visible = true, transformation(origin = {-74, 8}, extent = {{-4,
        -4}, {4, 4}}, rotation = -90)));
54  Modelica.Mechanics.MultiBody.Parts.Fixed F4(animation = false, r = B[:, 4])
        annotation(
55    Placement(visible = true, transformation(origin = {-74, -16}, extent = {{-4,
        -4}, {4, 4}}, rotation = -90)));
56  Modelica.Mechanics.MultiBody.Parts.Fixed F5(animation = false, r = B[:, 5])
        annotation(
57    Placement(visible = true, transformation(origin = {-74, -40}, extent = {{-4,
        -4}, {4, 4}}, rotation = -90)));
58  Modelica.Mechanics.MultiBody.Parts.Fixed F6(animation = false, r = B[:, 6])
        annotation(
59    Placement(visible = true, transformation(origin = {-74, -64}, extent = {{-4,
        -4}, {4, 4}}, rotation = -90)));
60  Modelica.Mechanics.MultiBody.Joints.Revolute B1(cylinderColor = {255, 153,
      51}, cylinderDiameter = D_B, cylinderLength = L_B, n = i_m[:, 1], phi(start
       = 0), stateSelect = StateSelect.always, useAxisFlange = true) annotation(
61    Placement(visible = true, transformation(origin = {-54, 54}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
62  Modelica.Mechanics.MultiBody.Joints.Revolute B2(cylinderColor = {255, 153,
      51}, cylinderDiameter = D_B, cylinderLength = L_B, n = i_m[:, 2], phi(start
       = 0), stateSelect = StateSelect.always, useAxisFlange = true) annotation(
63    Placement(visible = true, transformation(origin = {-56, 28}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
64  Modelica.Mechanics.MultiBody.Joints.Revolute B3(cylinderColor = {255, 153,
      51}, cylinderDiameter = D_B, cylinderLength = L_B, n = i_m[:, 3], phi(start
       = 0), stateSelect = StateSelect.always, useAxisFlange = true) annotation(
65    Placement(visible = true, transformation(origin = {-56, 4}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
66  Modelica.Mechanics.MultiBody.Joints.Revolute B4(cylinderColor = {255, 153,
      51}, cylinderDiameter = D_B, cylinderLength = L_B, n = i_m[:, 4], phi(start
       = 0), stateSelect = StateSelect.always, useAxisFlange = true) annotation(
67    Placement(visible = true, transformation(origin = {-56, -20}, extent =
        {{-10, -10}, {10, 10}}, rotation = 0)));
68  Modelica.Mechanics.MultiBody.Joints.Revolute B5(cylinderColor = {255, 153,
      51}, cylinderDiameter = D_B, cylinderLength = L_B, n = i_m[:, 5], phi(start
       = 0), stateSelect = StateSelect.always, useAxisFlange = true) annotation(
```

```
69      Placement(visible = true, transformation(origin = {-56, -44}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
70    Modelica.Mechanics.MultiBody.Joints.Revolute B6(cylinderColor = {255, 153,
        51}, cylinderDiameter = D_B, cylinderLength = L_B, n = i_m[:, 6], phi(start
        = 0), stateSelect = StateSelect.always, useAxisFlange = true) annotation(
71      Placement(visible = true, transformation(origin = {-56, -68}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
72    Modelica.Mechanics.MultiBody.Parts.BodyShape B1A1(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_BA,
        r = BA[:, 1], r_0(each fixed = false, start = Ab[:, 1]), r_CM = BA[:, 1] /
        2, shapeType = "cylinder", sphereDiameter = D_BAs, useQuaternions = false,
        width = D_BA) annotation(
73      Placement(visible = true, transformation(origin = {-32, 52}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
74    Modelica.Mechanics.MultiBody.Parts.BodyShape B2A2(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_BA,
        r = BA[:, 2], r_0(each fixed = false, start = Ab[:, 2]), r_CM = BA[:, 2] /
        2, shapeType = "cylinder", sphereDiameter = D_BAs, useQuaternions = false,
        width = D_BA) annotation(
75      Placement(visible = true, transformation(origin = {-32, 28}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
76    Modelica.Mechanics.MultiBody.Parts.BodyShape B3A3(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_BA,
        r = BA[:, 3], r_0(each fixed = false, start = Ab[:, 3]), r_CM = BA[:, 3] /
        2, shapeType = "cylinder", sphereDiameter = D_BAs, useQuaternions = false,
        width = D_BA) annotation(
77      Placement(visible = true, transformation(origin = {-32, 4}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
78    Modelica.Mechanics.MultiBody.Parts.BodyShape B4A4(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_BA,
        r = BA[:, 4], r_0(each fixed = false, start = Ab[:, 4]), r_CM = BA[:, 4] /
        2, shapeType = "cylinder", sphereDiameter = D_BAs, useQuaternions = false,
        width = D_BA) annotation(
79      Placement(visible = true, transformation(origin = {-32, -20}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
80    Modelica.Mechanics.MultiBody.Parts.BodyShape B5A5(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_BA,
        r = BA[:, 5], r_0(each fixed = false, start = Ab[:, 5]), r_CM = BA[:, 5] /
        2, shapeType = "cylinder", sphereDiameter = D_BAs, useQuaternions = false,
        width = D_BA) annotation(
81      Placement(visible = true, transformation(origin = {-32, -44}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
82    Modelica.Mechanics.MultiBody.Parts.BodyShape B6A6(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_BA,
        r = BA[:, 6], r_0(each fixed = false, start = Ab[:, 6]), r_CM = BA[:, 6] /
        2, shapeType = "cylinder", sphereDiameter = D_BAs, useQuaternions = false,
        width = D_BA) annotation(
83      Placement(visible = true, transformation(origin = {-32, -68}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
84    Modelica.Mechanics.MultiBody.Joints.Spherical A1(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_A) annotation(
85      Placement(visible = true, transformation(origin = {-8, 52}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
86    Modelica.Mechanics.MultiBody.Joints.Spherical A2(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_A) annotation(
87      Placement(visible = true, transformation(origin = {-8, 28}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
88    Modelica.Mechanics.MultiBody.Joints.Spherical A3(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_A) annotation(
```

```
89      Placement(visible = true, transformation(origin = {-8, 4}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
90    Modelica.Mechanics.MultiBody.Joints.Spherical A4(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_A) annotation(
91      Placement(visible = true, transformation(origin = {-8, -20}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
92    Modelica.Mechanics.MultiBody.Joints.Spherical A5(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_A) annotation(
93      Placement(visible = true, transformation(origin = {-8, -44}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
94    Modelica.Mechanics.MultiBody.Joints.Spherical A6(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_A) annotation(
95      Placement(visible = true, transformation(origin = {-8, -68}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
96    Modelica.Mechanics.MultiBody.Parts.BodyShape A1P1(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_AP,
         r = AP[:, 1], r_0(each fixed = false, start = Pb[:, 1]), r_CM = AP[:, 1] /
         2, shapeType = "cylinder", sphereDiameter = D_APs, useQuaternions = false,
         width = D_AP) annotation(
97      Placement(visible = true, transformation(origin = {16, 52}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
98    Modelica.Mechanics.MultiBody.Parts.BodyShape A2P2(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_AP,
         r = AP[:, 2], r_0(each fixed = false, start = Pb[:, 2]), r_CM = AP[:, 2] /
         2, shapeType = "cylinder", sphereDiameter = D_APs, useQuaternions = false,
         width = D_AP) annotation(
99      Placement(visible = true, transformation(origin = {16, 28}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
100   Modelica.Mechanics.MultiBody.Parts.BodyShape A3P3(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_AP,
         r = AP[:, 3], r_0(each fixed = false, start = Pb[:, 3]), r_CM = AP[:, 3] /
         2, shapeType = "cylinder", sphereDiameter = D_APs, useQuaternions = false,
         width = D_AP) annotation(
101     Placement(visible = true, transformation(origin = {16, 4}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
102   Modelica.Mechanics.MultiBody.Parts.BodyShape A4P4(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_AP,
         r = AP[:, 4], r_0(each fixed = false, start = Pb[:, 4]), r_CM = AP[:, 4] /
         2, shapeType = "cylinder", sphereDiameter = D_APs, useQuaternions = false,
         width = D_AP) annotation(
103     Placement(visible = true, transformation(origin = {16, -20}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
104   Modelica.Mechanics.MultiBody.Parts.BodyShape A5P5(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_AP,
         r = AP[:, 5], r_0(each fixed = false, start = Pb[:, 5]), r_CM = AP[:, 5] /
         2, shapeType = "cylinder", sphereDiameter = D_APs, useQuaternions = false,
         width = D_AP) annotation(
105     Placement(visible = true, transformation(origin = {16, -44}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
106   Modelica.Mechanics.MultiBody.Parts.BodyShape A6P6(animateSphere = false,
        animation = true, color = {153, 153, 153}, enforceStates = false, m = M_AP,
         r = AP[:, 6], r_0(each fixed = false, start = Pb[:, 6]), r_CM = AP[:, 6] /
         2, shapeType = "cylinder", sphereDiameter = D_APs, useQuaternions = false,
         width = D_AP) annotation(
107     Placement(visible = true, transformation(origin = {16, -68}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
108   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P1Op(color = {51, 51, 51},
        r = POp[:, 1], width = H_P) annotation(
109     Placement(visible = true, transformation(origin = {66, 52}, extent = {{-10,
```

```
      -10}, {10, 10}}, rotation = 0)));
110   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P2Op(color = {51, 51, 51},
        r = POp[:, 2], width = H_P) annotation(
111     Placement(visible = true, transformation(origin = {66, 28}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
112   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P3Op(color = {51, 51, 51},
        r = POp[:, 3], width = H_P) annotation(
113     Placement(visible = true, transformation(origin = {66, 4}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
114   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P4Op(color = {51, 51, 51},
        r = POp[:, 4], width = H_P) annotation(
115     Placement(visible = true, transformation(origin = {66, -20}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
116   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P5Op(color = {51, 51, 51},
        r = POp[:, 5], width = H_P) annotation(
117     Placement(visible = true, transformation(origin = {66, -44}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
118   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P6Op(color = {51, 51, 51},
        r = POp[:, 6], width = H_P) annotation(
119     Placement(visible = true, transformation(origin = {66, -68}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
120   Modelica.Mechanics.MultiBody.Joints.Spherical P1(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_P) annotation(
121     Placement(visible = true, transformation(origin = {40, 52}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
122   Modelica.Mechanics.MultiBody.Joints.Spherical P2(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_P) annotation(
123     Placement(visible = true, transformation(origin = {40, 28}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
124   Modelica.Mechanics.MultiBody.Joints.Spherical P3(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_P) annotation(
125     Placement(visible = true, transformation(origin = {40, 4}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
126   Modelica.Mechanics.MultiBody.Joints.Spherical P4(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_P) annotation(
127     Placement(visible = true, transformation(origin = {40, -20}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
128   Modelica.Mechanics.MultiBody.Joints.Spherical P5(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_P) annotation(
129     Placement(visible = true, transformation(origin = {40, -44}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
130   Modelica.Mechanics.MultiBody.Joints.Spherical P6(enforceStates = false,
        sphereColor = {255, 153, 51}, sphereDiameter = D_P) annotation(
131     Placement(visible = true, transformation(origin = {40, -68}, extent = {{-10,
        -10}, {10, 10}}, rotation = 0)));
132   Modelica.Mechanics.MultiBody.Parts.Body M(I_11 = M_I_11, I_21 = M_I_21, I_22 =
        M_I_22, I_31 = M_I_31, I_32 = M_I_32, I_33 = M_I_33, enforceStates = false
        , m = m, r_CM = cg, sphereColor = {51, 51, 51}, sphereDiameter = D_Op,
        useQuaternions = false) annotation(
133     Placement(visible = true, transformation(origin = {89, -9}, extent = {{-7,
        -7}, {7, 7}}, rotation = 0)));
134   Modelica.Mechanics.MultiBody.Parts.FixedTranslation Mb1(color = {51, 51, 51},
        length = L_M, r = i_m[:, 1], r_shape = r_Mb[:, 1], width = D_M) annotation(
135     Placement(visible = true, transformation(origin = {-74, 46}, extent = {{-6,
        -6}, {6, 6}}, rotation = 0)));
136   Modelica.Mechanics.MultiBody.Parts.FixedTranslation Mb2(color = {51, 51, 51},
        length = L_M, r = i_m[:, 2], r_shape = r_Mb[:, 2], width = D_M) annotation(
137     Placement(visible = true, transformation(origin = {-74, 22}, extent = {{-6,
        -6}, {6, 6}}, rotation = 0)));
```

```
138   Modelica.Mechanics.MultiBody.Parts.FixedTranslation Mb3(color = {51, 51, 51},
          length = L_M, r = i_m[:, 3], r_shape = r_Mb[:, 3], width = D_M) annotation(
139     Placement(visible = true, transformation(origin = {-74, -2}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
140   Modelica.Mechanics.MultiBody.Parts.FixedTranslation Mb4(color = {51, 51, 51},
          length = L_M, r = i_m[:, 4], r_shape = r_Mb[:, 4], width = D_M) annotation(
141     Placement(visible = true, transformation(origin = {-74, -26}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
142   Modelica.Mechanics.MultiBody.Parts.FixedTranslation Mb5(color = {51, 51, 51},
          length = L_M, r = i_m[:, 5], r_shape = r_Mb[:, 5], width = D_M) annotation(
143     Placement(visible = true, transformation(origin = {-74, -50}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
144   Modelica.Mechanics.MultiBody.Parts.FixedTranslation Mb6(color = {51, 51, 51},
          length = L_M, r = i_m[:, 6], r_shape = r_Mb[:, 6], width = D_M) annotation(
145     Placement(visible = true, transformation(origin = {-74, -74}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
146   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P12(color = {51, 51, 51},
          r = Pij[:, 1], width = H_P) annotation(
147     Placement(visible = true, transformation(origin = {31, -87}, extent = {{-7,
          -7}, {7, 7}}, rotation = 180)));
148   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P23(color = {51, 51, 51},
          r = Pij[:, 2], width = H_P) annotation(
149     Placement(visible = true, transformation(origin = {13, -87}, extent = {{-7,
          -7}, {7, 7}}, rotation = 180)));
150   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P34(color = {51, 51, 51},
          r = Pij[:, 3], width = H_P) annotation(
151     Placement(visible = true, transformation(origin = {-5, -87}, extent = {{-7,
          -7}, {7, 7}}, rotation = 180)));
152   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P45(color = {51, 51, 51},
          r = Pij[:, 4], width = H_P) annotation(
153     Placement(visible = true, transformation(origin = {-23, -87}, extent = {{-7,
          -7}, {7, 7}}, rotation = 180)));
154   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P56(color = {51, 51, 51},
          r = Pij[:, 5], width = H_P) annotation(
155     Placement(visible = true, transformation(origin = {-41, -87}, extent = {{-7,
          -7}, {7, 7}}, rotation = 180)));
156   Modelica.Mechanics.MultiBody.Parts.FixedTranslation P61(color = {51, 51, 51},
          r = Pij[:, 6], width = H_P) annotation(
157     Placement(visible = true, transformation(origin = {49, -87}, extent = {{-7,
          -7}, {7, 7}}, rotation = 180)));
158   Modelica.Mechanics.Rotational.Sources.Position M1(w(each fixed = true))
          annotation(
159     Placement(visible = true, transformation(origin = {-102, 56}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
160   Modelica.Mechanics.Rotational.Sources.Position M2(w(each fixed = true))
          annotation(
161     Placement(visible = true, transformation(origin = {-102, 30}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
162   Modelica.Mechanics.Rotational.Sources.Position M3(w(each fixed = true))
          annotation(
163     Placement(visible = true, transformation(origin = {-102, 6}, extent = {{-10,
          -10}, {10, 10}}, rotation = 0)));
164   Modelica.Mechanics.Rotational.Sources.Position M4(w(each fixed = true))
          annotation(
165     Placement(visible = true, transformation(origin = {-102, -16}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
166   Modelica.Mechanics.Rotational.Sources.Position M5(w(each fixed = true))
          annotation(
167     Placement(visible = true, transformation(origin = {-102, -40}, extent =
```

```
        {{-10, -10}, {10, 10}}, rotation = 0)));
168   Modelica.Mechanics.Rotational.Sources.Position M6(w(each fixed = true))
        annotation(
169     Placement(visible = true, transformation(origin = {-102, -64}, extent =
          {{-10, -10}, {10, 10}}, rotation = 0)));
170   Modelica.Blocks.Sources.TimeTable timeTable1(table = table1)  annotation(
171     Placement(visible = true, transformation(origin = {-126, 56}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
172   Modelica.Blocks.Sources.TimeTable timeTable2(table = table2) annotation(
173     Placement(visible = true, transformation(origin = {-126, 30}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
174   Modelica.Blocks.Sources.TimeTable timeTable3(table = table3) annotation(
175     Placement(visible = true, transformation(origin = {-126, 6}, extent = {{-6,
          -6}, {6, 6}}, rotation = 0)));
176   Modelica.Blocks.Sources.TimeTable timeTable4(table = table4) annotation(
177     Placement(visible = true, transformation(origin = {-126, -16}, extent =
          {{-6, -6}, {6, 6}}, rotation = 0)));
178   Modelica.Blocks.Sources.TimeTable timeTable5(table = table5) annotation(
179     Placement(visible = true, transformation(origin = {-126, -40}, extent =
          {{-6, -6}, {6, 6}}, rotation = 0)));
180   Modelica.Blocks.Sources.TimeTable timeTable6(table = table6) annotation(
181     Placement(visible = true, transformation(origin = {-126, -64}, extent =
          {{-6, -6}, {6, 6}}, rotation = 0)));
182 equation
183   connect(F1.frame_b, B1.frame_a) annotation(
184     Line(points = {{-74, 52}, {-64, 52}, {-64, 54}}));
185   connect(F2.frame_b, B2.frame_a) annotation(
186     Line(points = {{-74, 28}, {-66, 28}}));
187   connect(F3.frame_b, B3.frame_a) annotation(
188     Line(points = {{-74, 4}, {-66, 4}}));
189   connect(F4.frame_b, B4.frame_a) annotation(
190     Line(points = {{-74, -20}, {-66, -20}}));
191   connect(F5.frame_b, B5.frame_a) annotation(
192     Line(points = {{-74, -44}, {-66, -44}}));
193   connect(F6.frame_b, B6.frame_a) annotation(
194     Line(points = {{-74, -68}, {-66, -68}}));
195   connect(B1.frame_b, B1A1.frame_a) annotation(
196     Line(points = {{-44, 54}, {-44, 53}, {-42, 53}, {-42, 52}}));
197   connect(B2.frame_b, B2A2.frame_a) annotation(
198     Line(points = {{-46, 28}, {-42, 28}}));
199   connect(B3.frame_b, B3A3.frame_a) annotation(
200     Line(points = {{-46, 4}, {-42, 4}}));
201   connect(B4.frame_b, B4A4.frame_a) annotation(
202     Line(points = {{-46, -20}, {-42, -20}}));
203   connect(B5.frame_b, B5A5.frame_a) annotation(
204     Line(points = {{-46, -44}, {-42, -44}}));
205   connect(B6.frame_b, B6A6.frame_a) annotation(
206     Line(points = {{-46, -68}, {-42, -68}}));
207   connect(B1A1.frame_b, A1.frame_a) annotation(
208     Line(points = {{-22, 52}, {-18, 52}}));
209   connect(B2A2.frame_b, A2.frame_a) annotation(
210     Line(points = {{-22, 28}, {-18, 28}}));
211   connect(B3A3.frame_b, A3.frame_a) annotation(
212     Line(points = {{-22, 4}, {-18, 4}}));
213   connect(B4A4.frame_b, A4.frame_a) annotation(
214     Line(points = {{-22, -20}, {-18, -20}}));
215   connect(B5A5.frame_b, A5.frame_a) annotation(
216     Line(points = {{-22, -44}, {-18, -44}}));
217   connect(B6A6.frame_b, A6.frame_a) annotation(
```

```
    Line(points = {{-22, -68}, {-18, -68}}));
  connect(A1.frame_b, A1P1.frame_a) annotation(
    Line(points = {{2, 52}, {6, 52}}));
  connect(A2.frame_b, A2P2.frame_a) annotation(
    Line(points = {{2, 28}, {6, 28}}));
  connect(A3.frame_b, A3P3.frame_a) annotation(
    Line(points = {{2, 4}, {6, 4}}));
  connect(A4.frame_b, A4P4.frame_a) annotation(
    Line(points = {{2, -20}, {6, -20}}));
  connect(A5.frame_b, A5P5.frame_a) annotation(
    Line(points = {{2, -44}, {6, -44}}));
  connect(A6.frame_b, A6P6.frame_a) annotation(
    Line(points = {{2, -68}, {6, -68}}));
  connect(A1P1.frame_b, P1.frame_a) annotation(
    Line(points = {{26, 52}, {30, 52}}));
  connect(A2P2.frame_b, P2.frame_a) annotation(
    Line(points = {{26, 28}, {30, 28}}));
  connect(A3P3.frame_b, P3.frame_a) annotation(
    Line(points = {{26, 4}, {30, 4}}));
  connect(A4P4.frame_b, P4.frame_a) annotation(
    Line(points = {{26, -20}, {30, -20}}));
  connect(A5P5.frame_b, P5.frame_a) annotation(
    Line(points = {{26, -44}, {30, -44}}));
  connect(A6P6.frame_b, P6.frame_a) annotation(
    Line(points = {{26, -68}, {30, -68}}));
  connect(P1.frame_b, P1Op.frame_a) annotation(
    Line(points = {{50, 52}, {56, 52}}));
  connect(P2.frame_b, P2Op.frame_a) annotation(
    Line(points = {{50, 28}, {56, 28}}));
  connect(P3.frame_b, P3Op.frame_a) annotation(
    Line(points = {{50, 4}, {56, 4}}));
  connect(P4.frame_b, P4Op.frame_a) annotation(
    Line(points = {{50, -20}, {56, -20}}));
  connect(P5.frame_b, P5Op.frame_a) annotation(
    Line(points = {{50, -44}, {56, -44}}));
  connect(P6.frame_b, P6Op.frame_a) annotation(
    Line(points = {{50, -68}, {56, -68}}));
  connect(P1Op.frame_b, M.frame_a) annotation(
    Line(points = {{76, 52}, {76, 52.5}, {82, 52.5}, {82, -9}}));
  connect(P2Op.frame_b, M.frame_a) annotation(
    Line(points = {{76, 28}, {76, 29}, {82, 29}, {82, -9}}));
  connect(P3Op.frame_b, M.frame_a) annotation(
    Line(points = {{76, 4}, {76, 5}, {82, 5}, {82, -9}}));
  connect(P4Op.frame_b, M.frame_a) annotation(
    Line(points = {{76, -20}, {76, -19}, {82, -19}, {82, -9}}));
  connect(P5Op.frame_b, M.frame_a) annotation(
    Line(points = {{76, -44}, {76, -43.5}, {82, -43.5}, {82, -9}}));
  connect(P6Op.frame_b, M.frame_a) annotation(
    Line(points = {{76, -68}, {76, -67.75}, {82, -67.75}, {82, -9}}));
  connect(Mb1.frame_a, F1.frame_b) annotation(
    Line(points = {{-80, 46}, {-80, 52}, {-74, 52}}));
  connect(Mb2.frame_a, F2.frame_b) annotation(
    Line(points = {{-80, 22}, {-80, 28}, {-74, 28}}));
  connect(Mb3.frame_a, F3.frame_b) annotation(
    Line(points = {{-80, -2}, {-80, 4}, {-74, 4}}));
  connect(Mb4.frame_a, F4.frame_b) annotation(
    Line(points = {{-80, -26}, {-80, -20}, {-74, -20}}));
  connect(Mb5.frame_a, F5.frame_b) annotation(
    Line(points = {{-80, -50}, {-80, -44}, {-74, -44}}));
```

```
277   connect(Mb6.frame_a, F6.frame_b) annotation(
278     Line(points = {{-80, -74}, {-80.5, -74}, {-80.5, -68}, {-74, -68}}));
279   connect(P61.frame_a, P6Op.frame_a) annotation(
280     Line(points = {{56, -87}, {56, -68}}));
281   connect(P61.frame_b, P12.frame_a) annotation(
282     Line(points = {{42, -86}, {38, -86}, {38, -86}, {38, -86}}));
283   connect(P12.frame_b, P23.frame_a) annotation(
284     Line(points = {{24, -86}, {20, -86}, {20, -86}, {20, -86}}));
285   connect(P23.frame_b, P34.frame_a) annotation(
286     Line(points = {{6, -86}, {2, -86}, {2, -86}, {2, -86}}));
287   connect(P34.frame_b, P45.frame_a) annotation(
288     Line(points = {{-12, -86}, {-16, -86}, {-16, -86}, {-16, -86}}));
289   connect(P45.frame_b, P56.frame_a) annotation(
290     Line(points = {{-30, -86}, {-34, -86}, {-34, -86}, {-34, -86}}));
291   connect(M2.flange, B2.axis) annotation(
292     Line(points = {{-92, 30}, {-86, 30}, {-86, 40}, {-56, 40}, {-56, 38}, {-56,
          38}}));
293   connect(M3.flange, B3.axis) annotation(
294     Line(points = {{-92, 6}, {-86, 6}, {-86, 16}, {-56, 16}, {-56, 14}, {-56,
          14}}));
295   connect(M4.flange, B4.axis) annotation(
296     Line(points = {{-92, -16}, {-86, -16}, {-86, -8}, {-56, -8}, {-56, -10},
          {-56, -10}}));
297   connect(M5.flange, B5.axis) annotation(
298     Line(points = {{-92, -40}, {-86, -40}, {-86, -32}, {-56, -32}, {-56, -34},
          {-56, -34}}));
299   connect(M6.flange, B6.axis) annotation(
300     Line(points = {{-92, -64}, {-86, -64}, {-86, -56}, {-56, -56}, {-56, -58}}))
          ;
301   connect(M1.flange, B1.axis) annotation(
302     Line(points = {{-92, 56}, {-69, 56}, {-69, 64}, {-54, 64}}));
303   connect(timeTable1.y, M1.phi_ref) annotation(
304     Line(points = {{-119, 56}, {-114, 56}}, color = {0, 0, 127}));
305   connect(timeTable2.y, M2.phi_ref) annotation(
306     Line(points = {{-120, 30}, {-114, 30}, {-114, 30}, {-114, 30}}, color = {0,
          0, 127}));
307   connect(timeTable3.y, M3.phi_ref) annotation(
308     Line(points = {{-120, 6}, {-114, 6}, {-114, 6}, {-114, 6}}, color = {0, 0,
          127}));
309   connect(timeTable4.y, M4.phi_ref) annotation(
310     Line(points = {{-120, -16}, {-114, -16}, {-114, -16}, {-114, -16}}, color =
          {0, 0, 127}));
311   connect(timeTable5.y, M5.phi_ref) annotation(
312     Line(points = {{-120, -40}, {-116, -40}, {-116, -40}, {-114, -40}}, color =
          {0, 0, 127}));
313   connect(timeTable6.y, M6.phi_ref) annotation(
314     Line(points = {{-120, -64}, {-114, -64}, {-114, -64}, {-114, -64}}, color =
          {0, 0, 127}));
315   annotation(
316     uses(Modelica(version = "3.2.3")),
317     Diagram(coordinateSystem(extent = {{-150, -100}, {150, 100}})),
318     Icon(coordinateSystem(extent = {{-150, -100}, {150, 100}})),
319     version = "",
320     experiment(StartTime = 0, StopTime = 1, Tolerance = 1e-09, Interval = 0.001)
          );
321 end modelDinamikaParametarski;
```

Listing AE.2 - `modelDinamikaParametarski_template.mo`

Indeks

4N35, 182

Aerial Torpedo, 7
Aeroklub Franjo Kluz|hyperpage, 11
Aeroput, 11
AirSim, 281
AK8963, 245
Aleksandar Veg, 176
Antoinette simulator, 9
AQAP 2110, 39
Arduino Uno R3, 169
ATmega328, 169
ATmega328P, 244

bifilarno klatno, 249
BMP280, 252
Boeing 787, 11

Chiaravalli, 184
ChibiOS, 262
CLIK-Mate, 191
CMSIS-DSP, 218
CRC-16/MCRF4XX, 271
CS-LUAS, 40
CS-LURS, 40
Cube Orange, 259
Cygwin, 264

Direktorat civilnog vazduhoplovstva Republike Srbije, 23
DJI, 29
Dubai Airshow, 16
Dà-Jiāng Innovations, 29

Edwin A. Link, 10
ELVEM, 184
Emax 12A BLHELI, 250
EMAX MT1806, 250
Embention, 262
EN 9100:2018, 39
EN600, 221
EUROSATORY Pariz, 16

Farnborough International Airshow, 16
Flight Dynamics Model, 279
FlightGear, 279–281
FLYSKY FS-i6, 243

Gemfan 5030, 250
GY91, 252

Hewitt-Sperry Automatic Airplane, 24
Holybro, 259

Icarex, 246
IDEX Abu Dabi, 16
IEEE 802.11, 58
IEEE 802.3, 58
ILA Berlin, 16
INAV, 262
IPv4, 57
IPv6, 57

John Carmack, 167
JSBSim, 279

Keil MDK, 218
Kettering Bug, 24

LibrePilot, 243
Link Flight Trainer AN-2550-1, 11

M79 Osa, 29
MAX485, 182
Mašinski fakultet, 176
MCP4725, 182
Mehatronika, 176
MEMS, 245
Mickey Mantle, 19
Microchip, 169
MicroPilot, 262
Microsoft Flight Simulator, 279

Microsoft HoloLens, 68
MIL-STD-461G, 39
MIL-STD-810H, 39
Milipol Pariz, 16
Molex, 191
MPU-6500, 245
MPU-9250, 245
MQ-1 Predator, 24
Muzej vazduhoplovstva u Beogradu, 11, 25

NASA, 25
NATO AEP-80, 39
NATO AEP-83, 39
NATO AEP-89, 39
NEO-M8N, 252
Nikola Tesla, 1, 8
nRF24L01+, 244, 252

Oculus Quest 2, 68, 291

Pravilnik o bespilotnim vazduhoplovima, 38
PX4, 262

RATEL, 38
RFC 3550, 301
RFC 7826, 301
RTCA DO-178C, 39
RTCA DO-254, 39
RTCM 3, 305

S-8E/P, 245
SAKAC 30, 203
SORS 5706, 39
SORS 9000:2021, 39
SORS 9423:2018, 39
SRPS EN 9100:2018, 39
SRPS ISO 9001:2015, 39
STANAG 4671, 39
STANAG 4702, 39
STANAG 4703, 40
STANAG 4738, 40
STM32 F103, 182, 227
STM32 G051, 191
STM32 H7, 183
STM32 H743, 193
STM32CubeIDE, 218
STMicroelectronics, 218
Syma X6, 238

Talon GT Rebel, 259
Tehnički opitni centar, 39
Tim Sweeney, 309
TLE5012B, 191

UAV Navigation, 262
UAVOS Inc, 262
UBLOX, 252
UMEX Abu Dabi, 16
Univerzitet u Beogradu, 176

Vojna kontrola kvaliteta, 39

waf, 262
WebXR, 68

X-Plane, 279

Zakon o elektronskim komunikacijama, 39
Zakon o proizvodnji naoružanja i vojne opreme, 38
Zakon o vazdušnom saobraćaju, 38
ZOHD, 259

...ja samo to moram raditi, to je moja priroda, moje obrazovanje i moj život.

Arsen Dedić

Biografija autora

Miloš D. Petrašinović, master inženjer mašinstva, rođen je 8. oktobra 1994. godine u Beogradu. Osnovnu školu „Borislav Pekić" u Beogradu završio je 2009. godine i nosilac je diplome „Vuk Karadžić". Srednju školu „Petu beogradsku gimnaziju", prirodno-matematičkog smera, završio je 2013. godine u Beogradu i odbranio je maturski rad iz predmeta „Matematika" na temu „Elipsa".

Osnovne akademske studije na Mašinskom fakultetu Univerziteta u Beogradu upisao je 2013. godine. Osnovne akademske studije završio je 2016. godine, sa prosečnom ocenom 10 (deset), odbranivši (B.Sc.) rad iz završnog predmeta „Konstrukcija i tehnologija proizvodnje letelica" na temu „Projektovanje, proračun i parametarsko modeliranje stajnog trapa aviona", sa ocenom 10 (deset). Master akademske studije, modul Vazduhoplovstvo Katedre za vazduhoplovstvo, na Mašinskom fakultetu Univerziteta u Beogradu upisao je 2016. godine. Master akademske studije završio je 19. februara 2018. godine sa ukupnom prosečnom ocenom 10 (deset), odbranivši Diplomski (M.Sc.) rad na temu „Projektovanje simuliranog upravljanja letelicom" iz predmeta „Projektovanje letelica". Doktorske akademske studije na Mašinskom fakultetu Univerziteta u Beogradu upisao je 2018. godine i kao potencijalnog mentora predložio je prof. dr Aleksandra Grbovića. Trenutno je student treće godine Doktorskih akademskih studija. Dana 17.11.2022. godine mu je odobrena izrada doktorske disertacije „SISTEM ZA DALjINSKO PILOTIRANjE LETELICOM SA PARAMETRIMA KRETANjA LETELICE U POVRATNOJ SPREZI" („A SYSTEM FOR REMOTE PILOTING OF AN AIRCRAFT WITH PARAMETERS IN THE FEEDBACK LOOP"), za mentora doktorske disertacije imenovan je prof. dr Aleksandar Grbović. Tokom četvrte godine studija, 2017. godine, je zajedno sa koautorima napisao univerzitetski udžbenik pod naslovom „Stajni trap letelica" koji je izdao Mašinski fakultet Univerziteta u Beogradu.

Od decembra 2016. godine je registrovan kao preduzetnik za inženjerske delatnosti i tehničko savetovanje. Od juna 2020. godine je zaposlen na Mašinskom fakultetu Univerziteta u Beogradu kao saradnik u zvanju asistenta sa punim radnim vremenom za užu naučnu oblast Vazduhoplovstvo. Od jula 2020. godine je osnivač i suvlasnik kompanije Pink Research & Development Center (PR-DC), kao vazduhoplovni inženjer je zadužen za istraživanje i razvoj proizvoda. Kompanija je pod njegovim vođstvom ispunila sve zakonske uslove i dobila dozvolu za proizvodnju naoružanja i vojne opreme u oblasti istraživanja, razvoja i izrade bespilotnih letelica i njenih komponenti, posebno konstruisanih za vojnu namenu i izrade tehničke dokumentacije za navedena sredstva.

Dobitnik je Povelje za izuzetan uspeh tokom studiranja kao najbolji student generacije Mašinskog fakulteta koji je diplomirao u školskoj 2015/2016. godini povodom Dana Univerziteta u Beogradu. Dobitnik je Pohvala za odličan uspeh povodom Dana fakulteta svake školske godine tokom Osnovnih i Master akademskih studija. Takođe, povodom Dana fakulteta dobitnik je Pohvala za najboljeg studenta na osnovnim akademskim studijama iz generacije upisane na studije školske 2013/2014. godine, za najboljeg studenta na master akademskim studijama iz generacije upisane na studije školske 2016/2017 i Student generacije

najbolji student iz generacije upisane na fakultet školske 2013/2014 godine. Tokom studiranja je bio stipendista Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije kao i dobitnik stipendije „Dositija" za najbolje studente u Republici Srbiji Fonda za mlade talente Ministarstva omladine i sporta Republike Srbije, tokom završnih godina Osnovnih i Master akademskih studija.

U svakodnevnom radu koristi sledeće programske jezike i pakete: FORTRAN, C/C++, GNU Octave, Python, PHP (uz HTML i CSS), JavaScript (NodeJS, Electron, Cordova), LaTeX, Arduino, Arm CMSIS, Arduino_Core_STM32, FreeCAD, OpenModelica, CGAL, FFMpeg, GStreamer, Live555, OpenCV, Meshlab, Nastran, Code_Aster, KiCad, Blender, MakeHuman, Inkscape, Gimp i LibreOffice.

Učestvovao je na mnogim naučnim skupovima, skupovima studenata, predavanjima i konferencijama kako iz oblasti vezanih za mašinstvo, tako i za druge sfere kulturnog i društvenog života. Konkretne rezultate istraživanja za potrebe doktorske disertacije objavio je kao rad u tematskom zborniku od međunarodnog značaja Computational and Experimental Approaches in Materials Science and Engineering izdatoj od strane Springer-a 2020. godine i u časopisu Applied Sciences 2022. godine, dok je iste godine prezentovao deo istraživanja na konferenciji GAMM 2022 u Ahenu, Nemačkoj.

Njegov cilj, kome je potpuno posvećen, da Srbiju vrati na svetsku mapu u industriji dronova potiče iz duge vazduhoplovne tradicije u njegovoj porodici i činjenice da je njegova zemlja nekada imala važnu ulogu u ovoj industriji.

Pored maternjeg srpskog jezika, govori, čita i piše na engleskom jeziku. Ima vozačku dozvolu B kategorije. Oženjen je i ima jedno dete.

Izjava o autorstvu

Ime i prezime autora: **Miloš Petrašinović**
Broj indeksa: **d07/2018**

Izjavljujem

da je doktorska disertacija pod naslovom

Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi

- rezultat sopstvenog istraživačkog rada;
- da predložena disertacija u celini ni u delovima nije bila predložena za sticanje druge diplome prema studijskim programima drugih visokoškolskih ustanova;
- da su rezultati korektno navedeni i
- da nisam kršio autorska prava i koristio intelektualnu svojinu drugih lica.

Potpis doktoranda

U Beogradu, _____

_____

Izjava o istovetnosti štampane i elektronske verzije doktorskog rada

Ime i prezime autora: **Miloš Petrašinović**

Broj indeksa: **d07/2018**

Studijski program: **Doktorske akademske studije**

Naslov rada: **Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi**

Mentor: **prof. dr Aleksandar Grbović**

Izjavljujem da je štampana verzija mog doktorskog rada istovetna elektronskoj verziji koju sam predao za objavljivanje na portalu **Digitalnog repozitorijuma Univerziteta u Beogradu**.

Dozvoljavam da se objave moji lični podaci vezani za dobijanje akademskog zvanja doktora nauka, kao što su ime i prezime, godina i mesto rođenja i datum odbrane rada.

Ovi lični podaci mogu se objaviti na mrežnim stranicama digitalne biblioteke, u elektronskom katalogu i u publikacijama Univerziteta u Beogradu.

Potpis doktoranda

U Beogradu, _____

_____

Izjava o korišćenju

Ovlašćujem Univerzitetsku biblioteku „Svetozar Marković“ da u Digitalni repozitorijum Univerziteta u Beogradu unese moju doktorsku disertaciju pod naslovom:

Sistem za daljinsko pilotiranje letelicom sa parametrima kretanja letelice u povratnoj sprezi

koja je moje autorsko delo.

Disertaciju sa svim prilozima predao sam u elektronskom formatu pogodnom za trajno arhiviranje.

Moju doktorsku disertaciju pohranjenu u Digitalnom repozitorijum Univerziteta u Beogradu i dostupnu u otvorenom pristupu mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative Commons) za koju sam se odlučio.

1. Autorstvo (CC BY)
2. Autorstvo - nekomercijalno (CC BY-NC)
3. Autorstvo - nekomercijalno - bez prerade (CC BY-NC-ND)
4. Autorstvo - nekomercijalno - deliti pod istim uslovima (CC BY-NC-SA)
5. Autorstvo - bez prerade (CC BY-ND)
6. Autorstvo - deliti pod istim uslovima (CC BY-SA)

(Molimo da zaokružite samo jednu od šest ponuđenih licenci.
Kratak opis licenci je sastavni deo ove izjave).

Potpis doktoranda

U Beogradu, _____

_____

1. **Autorstvo**. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence, čak i u komercijalne svrhe. Ovo je najslobodnija od svih licenci.

2. **Autorstvo - nekomercijalno**. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca ne dozvoljava komercijalnu upotrebu dela.

3. **Autorstvo - nekomercijalno - bez prerade**. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, bez promena, preoblikovanja ili upotrebe dela u svom delu, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca ne dozvoljava komercijalnu upotrebu dela. U odnosu na sve ostale licence, ovom licencom se ograničava najveći obim prava korišćenja dela.

4. **Autorstvo - nekomercijalno - deliti pod istim uslovima**. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence i ako se prerada distribuira pod istom ili sličnom licencom. Ova licenca ne dozvoljava komercijalnu upotrebu dela i prerada.

5. **Autorstvo - bez prerade**. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, bez promena, preoblikovanja ili upotrebe dela u svom delu, ako se navede ime autora na način određen od strane autora ili davaoca licence. Ova licenca dozvoljava komercijalnu upotrebu dela.

6. **Autorstvo - deliti pod istim uslovima**. Dozvoljavate umnožavanje, distribuciju i javno saopštavanje dela, i prerade, ako se navede ime autora na način određen od strane autora ili davaoca licence i ako se prerada distribuira pod istom ili sličnom licencom. Ova licenca dozvoljava komercijalnu upotrebu dela i prerada. Slična je softverskim licencama, odnosno licencama otvorenog koda.