Analiza točnosti pozicioniranja neurokirurškog robota

Filipović, Ivan

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:210495

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

Sveučilište u Zagrebu

Fakultet strojarstva i brodogradnje

Neurokirurški robot – analiza preciznosti pozicioniranja lučnog sustava

Ivan Filipović

Zagreb, 2019

Sveučilište u Zagrebu

Fakultet strojarstva i brodogradnje

Neurokirurški robot – analiza preciznosti pozicioniranja lučnog sustava

STUDENT:
Ivan Filipović

MENTOR:
Prof. dr. sc. Mladen Crneković

Zagreb, 2019

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru profesoru Mladenu Crnekoviću na iskazanoj potpori tokom faksa još od projekta HEXAPOD s početaka fakulteta. Firmi INETEC i svim članovima NERO tima (Bruno Grgić, Kristijan Pucak, Marija Kurtoić, Danijel Janković, Ante Bakić i Ante Buljac) na pomoći i pruženoj prilici za rad na ovakvom projektu. Zahvalio bi se svojoj curi Ivi Smolković na pomnom ispravljanju gramatičkih pogrešaka i nerazumljivih struktura rečenica. Isto tako velike zahvale prijateljima i kolegama Anti Karagi, na redovno postavljenom pitanju „*Jesi li napisao diplomski?“* i na svim drugim pitanjima, Tomislavu Kosoru, na svim OTS i diplomskim kavama. Zahvala Miji Kovač na društvu svih ovih godina faksa i što je odabrala završit rok kasnije u svrhu omogučavanja međusobnog dolaska na promocije.

I zadnje, ali nipošto manje važno zahvala mojoj obitelji Ivici, Vesni, Agati i Luciji na svemu čime su pridonjeli mom završetku ovog fakulteta.

Ivan Filipović

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **IVAN FILIPOVIĆ** Mat. br.: 0035188244

Naslov rada na hrvatskom jeziku: **Analiza točnosti pozicioniranja nerokirurškog robota**

Naslov rada na engleskom jeziku: **Position accuracy analysis of neurosurgical robot**

Opis zadatka:

Cilj projekta neurokirurškog robota (NERO) je razvoj tzv. robotiziranog stereotaktičkog okvira. Okvir služi kao pomoć kirurgu koji invazivnim sredstvom treba izvesti operaciju, tj. doći što točnije u zadanu točku snimljenu CT uređajem. Kinematika okvira se sastoji od ukupno 5 stupnjeva slobode gibanja, tri translacije i dvije rotacije. Kao tehnički najzahtjevnije, a time i potencijalno žarište pogreške, identificirane su dvije rotacijske osi. Iz toga razloga potrebno je u fazi izrade i dokazivanja tehničkog koncepta ideje, modelirati i testirati rotacijske osi predloženog stereotaktičkog okvira.

U radu je potrebno:
- Objasniti konstrukciju eksperimentalnog postava za mjerenje točnosti pozicioniranja rotacijskih osi sustava.
- Razviti upravljačke algoritme za upravljanje osima.
- Analizirati propagaciju pogreške pozicioniranja kroz pojedine konstrukcijske elemente sustava.
- Analizirati ukupnu pogrešku pozicioniranja sustava (točnost i ponovljivost).
- Donjeti zaključke i prijedloge u cilju smanjenja pogreški pozicioniranja.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
07. ožujka 2019.

Rok predaje rada:
09. svibnja 2019.

Predviđeni datum obrane:
15. svibnja 2019.
16. svibnja 2019.
17. svibnja 2019.

Zadatak zadao:
prof. dr. sc. Mladen Crneković

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

Sažetak:

U ovom radu je provedena analiza preciznosti pozicioniranja eksperimentalnog postava NERO robotskog sustava. Rad pokriva sve radnje potrebne za provedbu analize. Eksperimentalni postav je osmišljen tako da prikaže princip rada rotacijskih osi lučnog sustava pozicioniranja. U radu je provedena izrada upravljačkih aplikacija, mjerenje mjernim sustavom FAROarm te analiza i korekcija pozicioniranja.

Ključne riječi: NERO, medicinska robotika, lučni sustav, preciznost pozicioniranja,

Summary:

This paper analyzes the accuracy of the positioning of the experimental setup of the NERO robot system. The paper covers all the actions needed to conduct the analysis. The experimental setup is designed to show the working principle of the rotating axes of the arc positioning system. This paper deals with the development of control applications, measurements with the FAROarm measuring system and analysis and position correction.

Key words: NERO, medical robotics, arc system, positioning precision

POPIS SLIKA:

Slika 1. Prikaz ulaznih i ciljanih točaka 3
Slika 2. Leksellov stereotaktički okvir. 3
Slika 3. Princip lučnog sustava 4
Slika 4. NeuroMate 5
Slika 5. Pathfinder 6
Slika 6. Renaissance 7
Slika 7. Robocast 8
Slika 8. iSys 9
Slika 9. MARS 10
Slika 10. Stereotaktički okvir robotskog sustava NERO 12
Slika 11. Eksperimentalni postav 13
Slika 12. NERO dizajn 14
Slika 13. Shematski prikaz robota NERO 15
Slika 14. Eksperimentalni postav - ulazna točka 16
Slika 15. Uška 17
Slika 16. Baza luka 18
Slika 17. Radni modul 19
Slika 18. Transparentni pogled na opružni mehanizam 20
Slika 19. Dimenzije motora Maxon 386675 21
Slika 20. Dimenzije motora Maxon 386660 22
Slika 21. Radno područje Maxonovog motora 386660 i 386675 23
Slika 22. Platinum Maestro 25
Slika 23. DC Whistle ServoDrive 26
Slika 24.Optički enkoder RESA+RESOLUTE RA26 27
Slika 25. Magnetski enkoder RLM2IC čitača glava 28
Slika 26. Glavni prozor upravljačkog sučelja 29
Slika 27. Prikaz stanja osi 30
Slika 28. Odabir vrijednosti apsolutnog pozicioniranja radnih osi 30
Slika 29. Format zapisa izmjerenih vrijednosti 31
Slika 30. Prozor za analizu i kompenzaciju pogreške 32
Slika 31. Unos IP adrese uređaja 32
Slika 32. Nicholsov dijagram za regulacijski krug brzine za os R1 34
Slika 33. Nicholsov dijagram za regulacijski krug pozicije za os R1 34
Slika 34. Nicholsov dijagram za regulacijski krug brzine os R2 35
Slika 35. Nicholsov dijagram za regulacijski krug pozicije os R2 35
Slika 36. Prikaz dijelova FaroArm QuantumS-a 37
Slika 37. Umjeravanje luka (Crveno gornja ploha, plavo donja) 38
Slika 38. Unutarnja i vanjska kružnica (plavo i crveno) 39
Slika 39. Orijentacija x osi prema lijevoj uški (crveno) 40
Slika 40. Sferni utor za mjerenje pozicije 40
Slika 41. Odstupanja vrijednosti po osi X 45
Slika 42. Odstupanja vrijednosti po osi Y 45
Slika 43. Odstupanja vrijednosti po osi Z 46
Slika 44. Ukupna vrijednosti odstupanja 46
Slika 45. Kartezijev koordinatni sustav glave pacijenta 47

Slika 46. Skica odstupanja radne igle od ravnine luka 48
Slika 47. Skica izračuna z_k i x_k komponenti vektora rotacije R2 osi 49
Slika 48. Bilinearna interpolacija 51
Slika 49. Ukupno prostorno odstupanje nakon korekcije poziciji 54
Slika 50. Prikaz prostornog odstupanja prije i nakon korekcije pozicioniranja 56

POPIS TABLICA:

Tablica 1. Prikaz grešaka u mjerenju 36
Tablica 2. Mjerenje R1, R2=(0°, 0°) 41
Tablica 3. Mjerenje R1, R2=(0°, 90°) 42
Tablica 4. Mjerenje R1, R2=(0°, 170°) 42
Tablica 5. Mjerenje R1, R2=(25°, 0°) 42
Tablica 6. Mjerenje R1, R2=(25°, 90°) 43
Tablica 7. Mjerenje R1, R2=(25°, 170°) 43
Tablica 8. Mjerenje R1, R2=(60°, 0°) 43
Tablica 9. Mjerenje R1, R2=(60°, 90°) 44
Tablica 10. Mjerenje R1, R2=(60°, 170°) 44
Tablica 11. Prikaz mjerenja prije i nakon kompenzacije 53
Tablica 12. Prikaz mjerenja prije i nakon kompenzacije 54

POPIS OZNAKA

OZNAKA	JEDINICA	OPIS
Euler	mm	prostorno odstupanje mjerene i teorijske vrijednosti točke
E_{UNI}^2	mm	greška mjerenja dvije točke FARO sustava
e_x	mm	odstupanje mjerene od teorijske vrijednosti osi *x*
e_y	mm	odstupanje mjerene od teorijske vrijednosti osi *y*
e_z	mm	odstupanje mjerene od teorijske vrijednosti osi *z*
L_{DIA}^2	mm	greška promjera položaja mjerene sfere FARO sustava
P_{FORM}^2	mm	greška mjerenja oblika FARO sustava
P_{SIZE}^2	mm	greška mjerenja promjera sfere FARO sustava
r	mm	radijus ulazne točke NERO eksperimentalnog sustava
R1	°	kut zakreta prve rotacijske osi, zakret luka
R2	°	kut zakreta druge rotacijske osi, zakret radnog modula
$SPAT^1$	mm	preciznost ispitivanja jednostruke točke FARO sustava
T1	mm	prva translacijska os, z vrijednost ciljne točke
T2	mm	druga translacijska os, y vrijednost ciljne točke
T3	mm	treća translacijska os, z vrijednost ciljne točke
x_k	mm	x koordinata korekcije zakreta α
x_{komp}	mm	x koordinata nakon kompenzacije
x_m	mm	x koordinata mjerenja
x_t	mm	teorijska vrijednost x koordinate
y_k	mm	y koordinata korekcije zakreta α
y_{komp}	mm	y koordinata nakon kompenzacije
y_m	mm	y koordinata mjerenja
y_t	mm	teorijska vrijednost y koordinate
z_k	mm	z koordinata korekcije zakreta α
z_{komp}	mm	z koordinata nakon kompenzacije
z_m	mm	z koordinata mjerenja
z_t	mm	teorijska vrijednost z koordinate
α	°	kut zakreta radnog modula od luka
Δx	mm	odstupanje mjerene od teorijske vrijednosti osi x, primjer
Δy	mm	odstupanje mjerene od teorijske vrijednosti osi y, primjer
Δz	mm	odstupanje mjerene od teorijske vrijednosti osi z, primjer
φ	°	kut zakreta sfernog koordinatnog sustava
θ	°	kut zakreta sfernog koordinatnog sustava

Sadržaj:

Sažetak: ii
Summary: iii
POPIS SLIKA iv
POPIS TABLICA vi
POPIS OZNAKA vii
1. Uvod 1
1.1 Stereotaktička neurokirurgija 2
1.1.1 Leksellov stereotaktički okvir 3
1.2 Robotika u neurokirurgiji 4
1.2.1 NeuroMate 4
1.2.2 Pathfinder 5
1.2.3 Renaissance(MARS) 6
1.2.4 Robocast 8
1.2.5 iSys 8
1.2.6 MARS 9
2. Projekt NERO 11
2.1 Dizajn 14
3. Eksperimentalni postav 15
3.1 Konstrukcija R1 i R2 17
3.1.1 Lijeva i desna uška 17
3.1.2 Luk 18
3.1.3 Radni modul 19
3.1.4 Maxon 386675 21
3.1.5 Maxon 386660 22
3.1.6 P-MAS 24
3.1.7 DC Whistle 26
3.1.8 RESA+RA26 27
3.1.9 RLM2IC 28
3.2 Izvedba Alpha prototipa 29
3.2.1 Korisničko sučelje 29
3.2.2 P-MAS mikrokontroler 33
3.2.3 Regulacija pozicioniranja osi 33
3.3 FARO QUANTUM S 36
4. Analiza pogreške pozicioniranja 38

4.1 Mjerenje 38
4.1.1 Postupak mjerenja 38
4.1.2 Rezultati mjerenja 41
4.1.3 Kompenzacija greške 47
5. Zaključak 55
6. Literatura 57
7. Prilog 58

1. Uvod

U ovom diplomskom radu glavni fokus je na rotacijskim osima NERO robota, koje su označene kao kritične u smislu preciznosti i ponovljivosti pozicioniranja samog sustava. Razlog tome je manja preciznost rotacijskih enkodera te velika vjerojatnosti pojavljivanja zračnosti kod prijenosa potrebnog za generiranje momenta dovoljnog za rotaciju i zaključavanje ovih pozicija.

Kroz rad će biti prikazan dizajn eksperimentalnog postava uređaja na kojem bi se trebala provesti analiza preciznosti pozicioniranja ovakvog lučnog sustava. Prikazat će se sve bitne konstrukcijske komponente, aktuatori, mjerni članovi te upravljačke jedinicae.

Nadalje, u radu će se obraditi:

- regulacijske petlje dvaju osi te način upravljanja, njihova implementacija na servo sustave uređaja
- razvoj aplikacije grafičkog sučelja u JAVAfx programskom jeziku preko kojeg se odvija upravljanje robotskim sustavom
- izrada programa mikrokontrolera kao člana koji izvršava sve radnje potrebne za traženo gibanje
- mrežna povezanost dvaju sustava

Glavni fokus ovog rada je analiza i kompenzacija izmjerenih odstupanja od teorijski točnih pozicija. Analiza i kompenzacija se odvija na dva mjesta. Prva je unutar JAVAfx sučelja, iz više razloga - mogućnost slanja korigirane vrijednosti na robotski sustav, tako da se račun može provjeriti prije slanja na manipulator i pohranjivanje tih vrijednosti u zasebnu „txt“ datoteku, tako da se ove vrijednosti mogu pomoću Matlab-a prikazati u obliku grafova na kojima se jasno vidi propagacija greške po radnom polju uređaja.

Cilj rada je analizirati preciznost pozicioniranja ovakvog sustava te detektirati mjesta i uzroke najvećih odstupanja, zatim na temelju toga napraviti kompenzaciju odstupanja ili predložiti određene radnje u svrhu smanjenja izmjerenih pogreški pozicioniranja.

1.1 Stereotaktička neurokirurgija

Stereotaktika se bazira na mogućnosti mapiranja volumena koristeći pri tome precizna mjerenja unutar nekog koordinatnog sustava. Riječ „stereotaktika“ potječe iz grčkog jezika i znači „tri-dimenzije, uređene“.

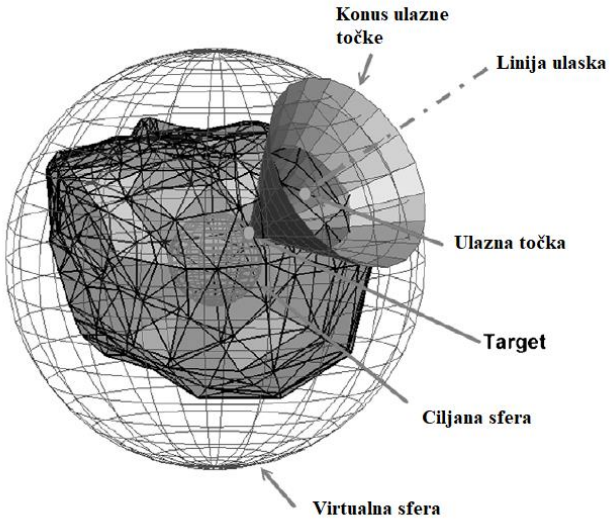
Princip stereotaktike se temelji na određivanju točki u prostoru preko sjecišta koordinata 3 ravnine: lateralne, vertikalne i anteroposteriorne. Na taj način u suradnji sa znanjem iz anatomije i neurologije dolazimo do mogućnosti pronalaska i prilaska ciljanim točkama unutar prostora bez direktne vizualizacije ciljne točke.

Princip prilaska ciljnim točkama te iskorištavanje istoga sa znanjem anatomije doveo je do razvitka novih vrsta operacija. Neki od takvih zahvata su biopsija, SEEG, DBS i mnoge druge, čije su prednosti minimalna invazivnost i preciznije izvršavanje samih zahvata.

Prvi stereotaktički uređaj za neurokirurgiju je godine 1947. objavio Spiegel. Do 1950. u svijetu se razvilo 40 različitih stereotaktičkih uređaja te su se podijelili u kategorije: translacijski sustav, sustav montiranih rupa, lučni sustav, sustav s više lukova koji se presijecaju i bez okvirni sustav.

Svi ovi uređaji imaju istu svrhu, stvaranje krute veze između ciljane i fiksirane točke preko ulazne točke cjelokupnog sustava.

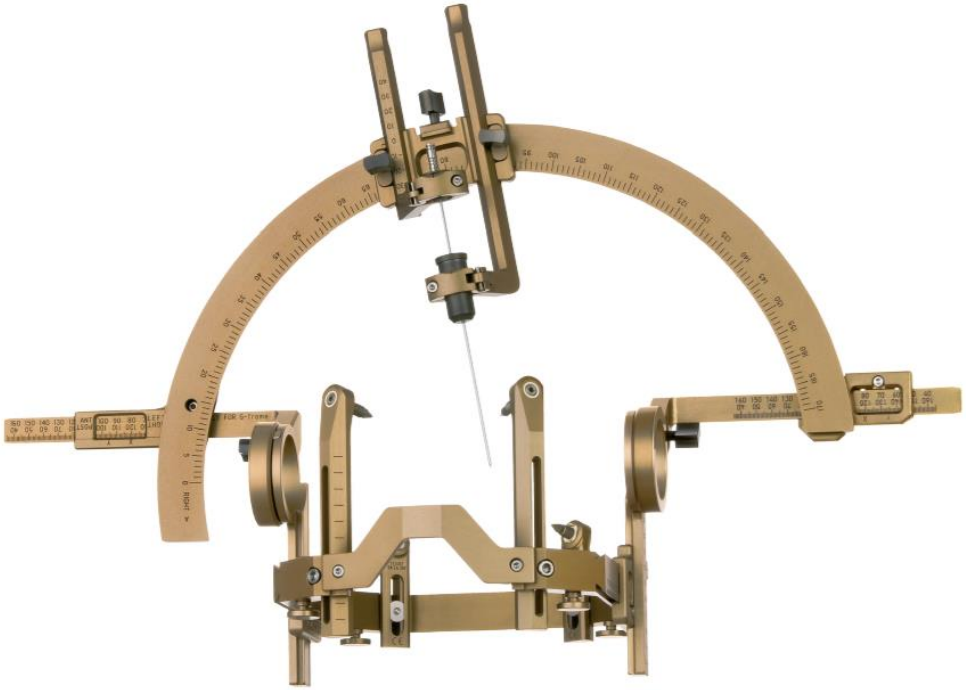
Kako bismo razumjeli stereotaktičke sustave moramo razumjeti pojmove vezane uz sam princip. Prva i najbitnija je „ciljana točka“ - točka unutar mozga u koju želimo dovesti vrh alata. Tu se nalazi područje na kojem se izvodi zahvat (uzima se dio tkiva-biopsija, dovode se elektrode-DBS, stimulira se moždano područje-SEEG). Skup takvih točaka se naziva „ciljana sfera“ i unutar nje se nalaze sve točke u koje želimo imati mogućnost doći pojedinim zahvatima. Zatim postoji „virtualna sfera“, tj. skup točaka oko glave koji se sastoji od skupa „ulaznih točaka“. „Ulazne točke“ su mjesta kroz koja će alat ući u lubanju pacijenta. Svakoj ulaznoj točci se definira i *konus* koji prikazuje sve moguće upadne kutove te „ulazne točke“. Pravac koji povezuje ciljanu i ulaznu točku zove se *linija ulaska*.



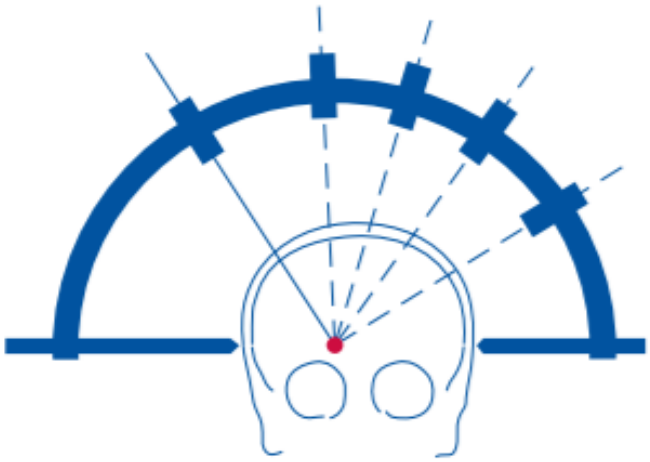
Slika 1. Prikaz ulaznih i ciljanih točaka

1.1.1 Leksellov stereotaktički okvir

Leksellov stereotaktički sustav se bazira na lučnom sustavu, a sastoji se od pravokutnog okvira koji se učvrsti za pacijentovu lubanju. Koordinate se nanose na pravokutni sustav na poprečnoj osovini po kojoj luk klizi lijevo ili desno ovisno o vrijednosti Y. U tom stereotaktičkom okviru ciljna je točka uvijek u sredini luka te je moguće usmjeriti elektrodu ili sondu iz bilo koje ulazne točke na lubanji.



Slika 2. Leksellov stereotaktički okvir.



Slika 3. Princip lučnog sustava

1.2 Robotika u neurokirurgiji

U ovom poglavlju će se napraviti pregled nekih poznatijih postojećih robotskih sustava u neurokirurgiji.

1.2.1 NeuroMate

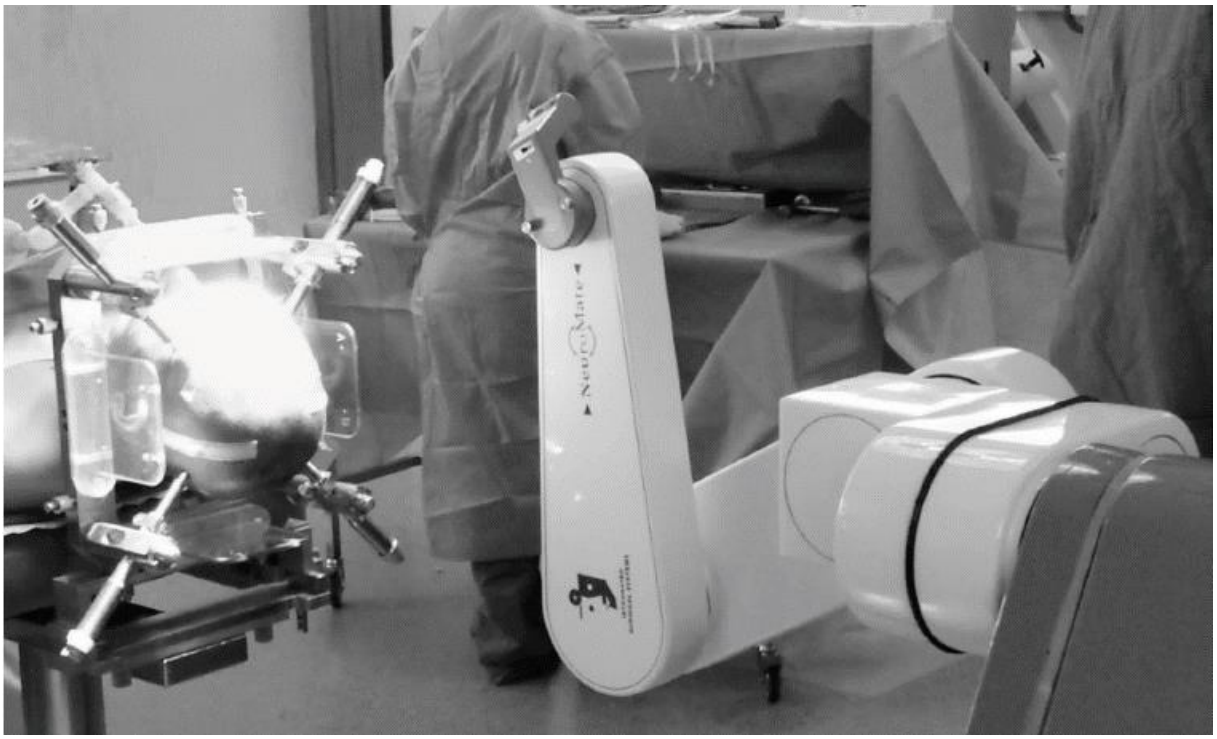
Jedan od prvih robota na području neurokirurgije je NeuroMate. To je robotski sustav koji se sastoji od robotske ruke s pet stupnjeva slobode gibanja (strukture RRRRR) i vizijskog sustava za navođenje. Glavna zadaća mu je pasivno asistiranje u obliku držanja i stabilizacije instrumenta kojima upravlja kirurg. Na taj način postiže se sigurnost i preciznost te posljedično i uspješnost zahvata.

Sustav je 1997. godine dobio EC brand te dozvolu FDA za prakticiranje stereotaktičke neurokirurgije u SAD-u. Dvije godine kasnije sustav dobiva dozvole za rad i bez stereotaktičkog okvira.

Serijski robotski manipulator sa 5 stupnjeva slobode omogućava NeuroMate-u tehničku točnost od 0.7 mm i preciznost sustava od 0.15 mm, pri teretu od 5 kg. Ovisno o potrebama neurokirurškog zahvata preko raznih modula može raditi s baznim okvirom, bez baznog okvira, u raznim okruženjima koristeći razne metode i rješenja potrebna za obavljanje zahvata. Lokalizaciju može obavljati klasičnim lokalizatorima na stereotaktičkom okviru ili bez okvira ultrazvukom.

Istraživanje [1 Faria] je pokazalo da sustav ima nešto manju preciznost od navedene (0.86mm $\pm$ 0.32mm) za uporabu s baznim okvirom i (1.95mm $\pm$ 0.44mm) bez baznog okvira te kao takav nema statistične razlike od klasične primjene stereotaktičkih okvira. Drugo istraživanje je pokazalo kako je NeuroMate u 91 SEEG zahvata imao preciznost pozicioniranja od 0.86mm $\pm$ 0.56mm na ulaznoj točci, a 2.04mm $\pm$ 1.31mm na ciljanoj.

Sve ove greške nalaze se u strukturi robota uz koje se kao negativna strana još navodi i cijena. [1] [2]



Slika 4. NeuroMate

1.2.2 Pathfinder

Pathfinder je sljedeći robotski sustav u neurokirurgiji s kojim ćemo se malo bolje upoznati. Sastoji je od robotske ruke sa šest stupnjeva slobode ugrađene na mobilnu bazu sustava. To omogućava sustavu slobodno kretanje po operacijskoj sali. Kada se robot pozicionira fiksira se za Mayfieldov okvir. Lokalizacija se obavlja pomoću markera pričvršćenih za skalp ili lubanju pacijenta koji se prate vizijskim sustavom u relativnoj poziciji s robotskom rukom. Isto tako, markeri su vidljivi na CT-u te se zbog toga omogućava navigacija sustava uz pomoć CT i MRI snimki.

Registracija bez okvira u ovom sustavu ima milimetarsku preciznost. Sam način rada sustava mu omogućava premještanje sustava u operacijskoj sali i za vrijeme zahvata, zbog mogućnosti praćenja markera. Na taj način je riješena potreba za intraoperativnim skeniranjima zbog mogućnosti greške u odnosu na predoperativna snimanja.

Najveći problemi ovog sustava su mogućnost pomaka kože u vremenu između skeniranja i greške koje bi nastajale unutar vizijskog sustava zbog osvjetljenja. 2009. je projekt Pathfinder ukinut. [1] [3]

Slika 5. Pathfinder

1.2.3 Renaissance(MARS)

Renaissance sustav sastoji se od kontrolera i MARS robota baziranog na Stewartovoj platformi te uz sebe ima laserski 3D skener i računalo kao dio sustava. Programska podrška

sustava sastoji se od softwarea za predoperativno planiranje, za skeniranje površine, registraciju i izvršavanje zahvata.

MARS-ov robot baziran na Stewartovoj platformi je robot paralelne strukture sa šest stupnjeva slobode. Preciznost gibanja mu je 0.1mm. Robot se može montirati na Mayfielda ili direktno na lubanju pacijenta. Nakon provedbe fiksiranja MARS-ova platforma je krutoj vezi iz koje može vršiti bušenje ili vođenje.

Procedura lokalizacije se izvršava bez markera i bez okvira na način da se predoperativno planiranje povezuje sa intraoperativnim preko uspoređivanja površina snimljenih CT-om ili MRI-em i laserskim skeniranjem. U ovoj metodi se traže referentne točke pacijentovog čela ili uha.

Greška nastala registracijom je približna 1mm na ulaznoj točci, dok je u ciljanoj približno 1.7mm.

Sustav je sam po sebi bez okvira i ne koristi markere. Što je prednosti jer uklanja invazivnost koju predstavlja umetanje markera u lubanju. Iako je prvotno ciljana cijena sustava 100,000 USD, procjena je bliže 1,000,000 USD [1]

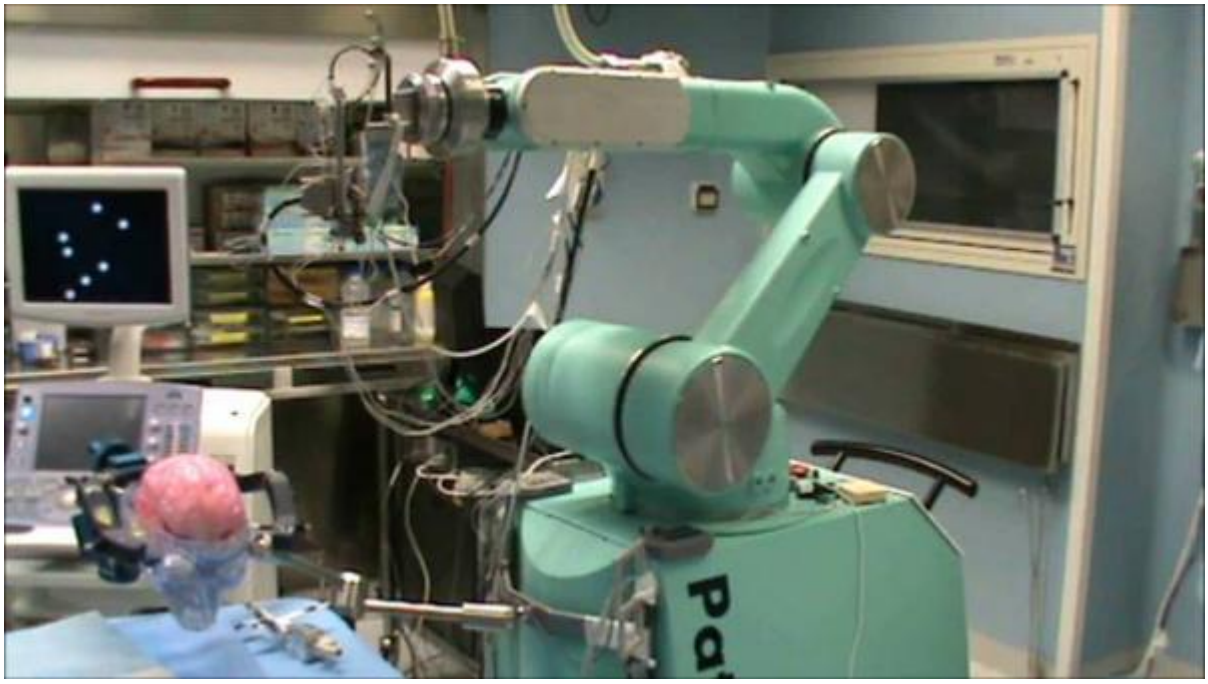
Slika 6. Renaissance

1.2.4 Robocast

Robocast je sustav koji se sastoji od robotske ruke, planera trajektorije, kontrolera i dodatnog robotskog sustava za fino pozicioniranje i dodatne stupnjeve slobode.

Tehnička preciznost ovog sustava je prema istraživanju [1 Faria] 0.6mm za ulazne točke i 0.4mm za ciljane točke. Rotacijska srednja greška $6.5x10^{-3}$ rad, što odgovara zahtjevima kliničke primjene.

Projekt Robocast je završio 2011. te se nastavio pod novim imenom ACTIVE project unutar kojeg se rade istraživanja s dva autonomna kooperativna robotska sustava s 20 stupnjeva slobode. [1]

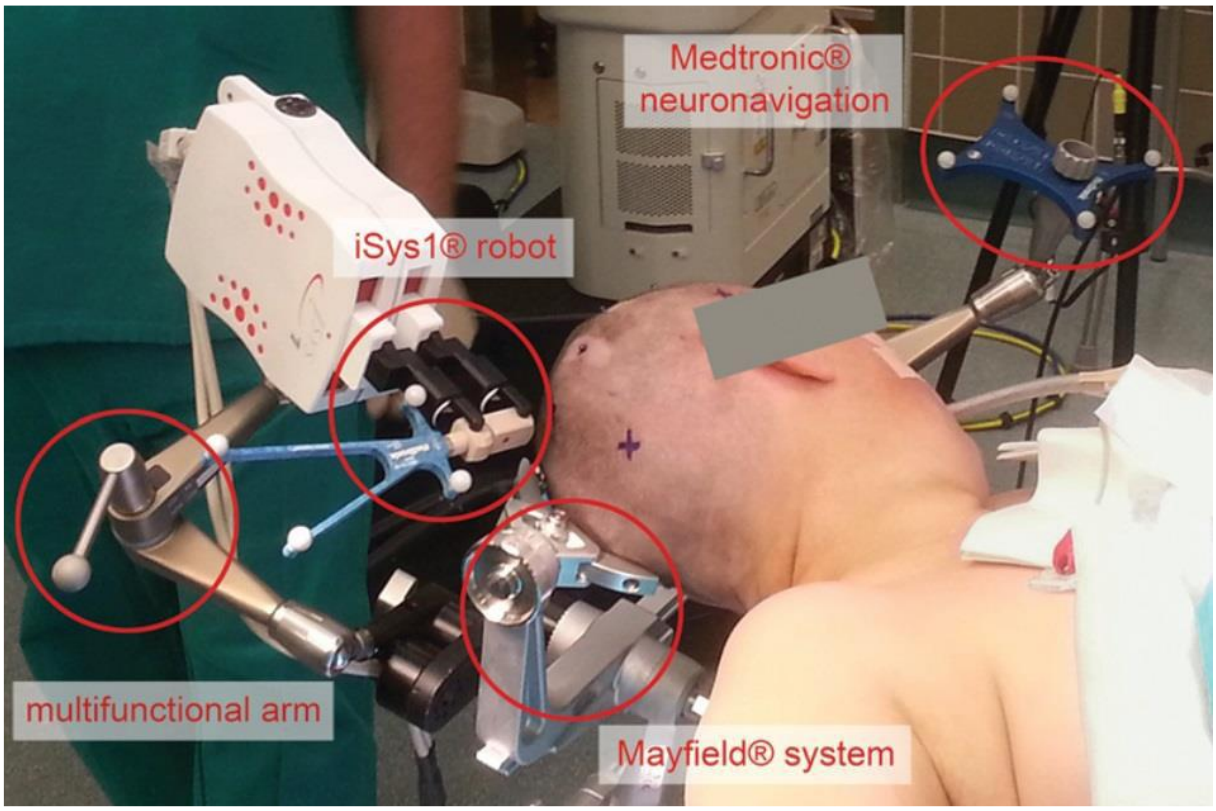


Slika 7. Robocast

1.2.5 iSys

iSys je robotizirani mikro modul koji radi na principu translacije dvije plohe čiji je rezultat pomak alata u željenim smjerovima. Za translaciju se obje translacijske plohe gibaju zajedno, dok se za promjenu kuta upada alata translacijske plohe pomiču paralelno jedna od druge.

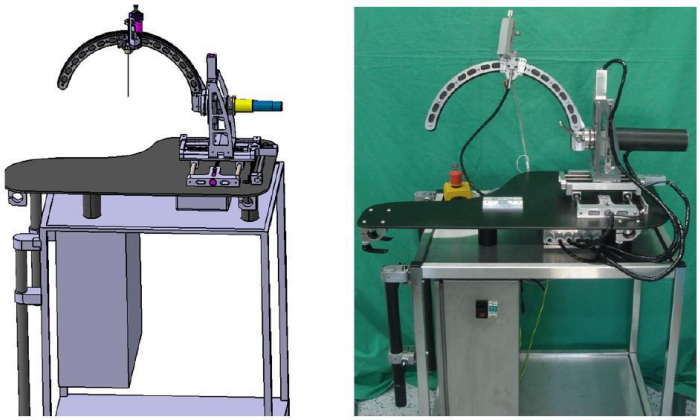
Na ovaj način se dobiva precizan mikro modul no kao i kod MAZOR-a, postoji problem postavljanja sustava u područje oko željenog radnog prostora. Jedna od prednosti naspram MAZOR-a je mogućnost montiranja preko ruke koja je pričvršćena na stol. Na slici 8. se vidi fiksacija za Mayfieldov adapter. Prednost je u tome što neurokirurg ima slobodu namjestiti mikro modul kako želi te si na taj način osloboditi svoje radno područje. Veliki nedostatak je što se prilikom svake promjene regionalnog područja samog modula mora iznova raditi lokalizacija.



Slika 8. iSys

1.2.6 MARS

MARS je neurokirurški sustav baziran na stereotaktičkim okvirima. Njegov princip rada je lučni, što znači da se njegova ciljna točka uvijek nalazi u sredini luka. Posjeduje pet sloboda gibanja (dodatnu translaciju alata) te mu je kinematska struktura TTTRR. Postolje mu omogućava translacije po x, y, z osima dok mu rotacija luka i gibanje po luku omogućavaju još dvije rotacije.



Slika 9. MARS

Ovaj princip robota je NERO projektu najzanimljiviji jer je najbliži pokušaju robotizacije lekselovog baznog okvira. Iako ima neke očite nedostatke, poput pojavljivanja savijanja luka pod svojom i pod težinom radnog modula, očuvanjem lučnog sustava ima i svoje prednosti poput velike kompatibilnosti s načinima rada moderne neurokirurgije. *[4]*

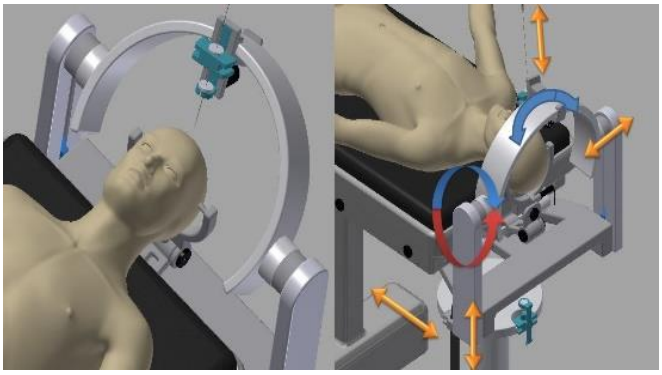
2. Projekt NERO

Projekt NERO-Neurokirurški robot je projekt tvrtke INETEC u suradnji s Fakultetom strojarstva i brodogradnje u Zagrebu i Kliničkom bolnicom Dubrava. Projekt NERO predstavlja svojevrsni slijed istraživačko-razvojnih aktivnosti u segmentu medicinske robotike te se nastavlja na prethodni uspješni projekt robotske neuronavigacije RONNA kojega su vodili FSB i KBD.

Važan aspekti postupaka stereotaktičke neurokirurgije je pouzdana i vremenski učinkovita prostorna navigacija medicinskih instrumenata u interkranijalnom prostoru pacijenta. Trenutno ne postoji dovoljno sofisticirano tehnološko rješenje u ovom području. Trenutni instrumenti korišteni za stereotaktičku neurokirurgiju zbog nedostatka modernih tehnologija čine ovu vrstu neurokirurških zahvata dugotrajnim. Uvođenjem visokih tehnologija (vizijski sustavi i robotika) u proces registracije pacijenta te u praćenje operativnog postupka bi u velikoj mjeri pojednostavilo i podiglo kvalitetu izvedbe stereotaktičkih neurokirurških zahvata. Upotreba postojećih stereotaktičkih okvira podrazumijeva dodatne zahvate na zdravom tkivu pacijenta upotrebom markera i krutom fiksacijom lubanje. Navedeno rezultira psihološkom traumom pacijenata, mogućim operativnim i postoperativnim komplikacijama te dugotrajnim pred operacijskim postupkom tijekom kojeg mora biti prisutan neurokirurg, zbog čega se povećavaju troškovi operacijskih zahvata.

Svrha projekta NERO je kreirati robotizirani stereotaktički neurokirurški sustav koji će biti inovativan svojim sposobnostima i oblikom, a sve to u skladu s preferencijama kliničkih bolničkih centara i neurokirurga kao ciljanih korisnika.

Ciljevi projekta NERO su razvoj tehnološkog rješenja koje će riješiti spomenute nedostatke klasične stereotaktičke neurokirurgije i razvoj cjenovno prihvatljivijeg tehničkog rješenja. NERO će objeđivanjem prednosti trenutnih stereotaktičkih tehnika i robotske tehnologije pokušati riješiti nedostatke i na taj način stvoriti jedinstven, tehnološki napredan robotski sustav. Samim time će pojednostaviti postupke neuronavigacije i skratit vrijeme trajanja predoperativnih i operativnih zahvata u neurokirurgiji.



Slika 10. Stereotaktički okvir robotskog sustava NERO

Ciljevi projekta NERO:

1. *Demonstriran tehnološki koncept u laboratorijskom okruženju*
2. *Prototipiziran robotski sustav NERO spreman za komercijalnu primjenu*
3. *Rezultati istraživanja i razvoja diseminirani i predstavljeni poslovnoj, zdravstvenoj te široj zainteresiranoj javnosti*

Projekt se provodi kroz četiri faze projekta:

E1. Industrijsko istraživanje za potrebe razvoja robotskog sustava NERO
E2. Eksperimentalni razvoj robotskog sustava
E3. Upravljanje projektom i administracija

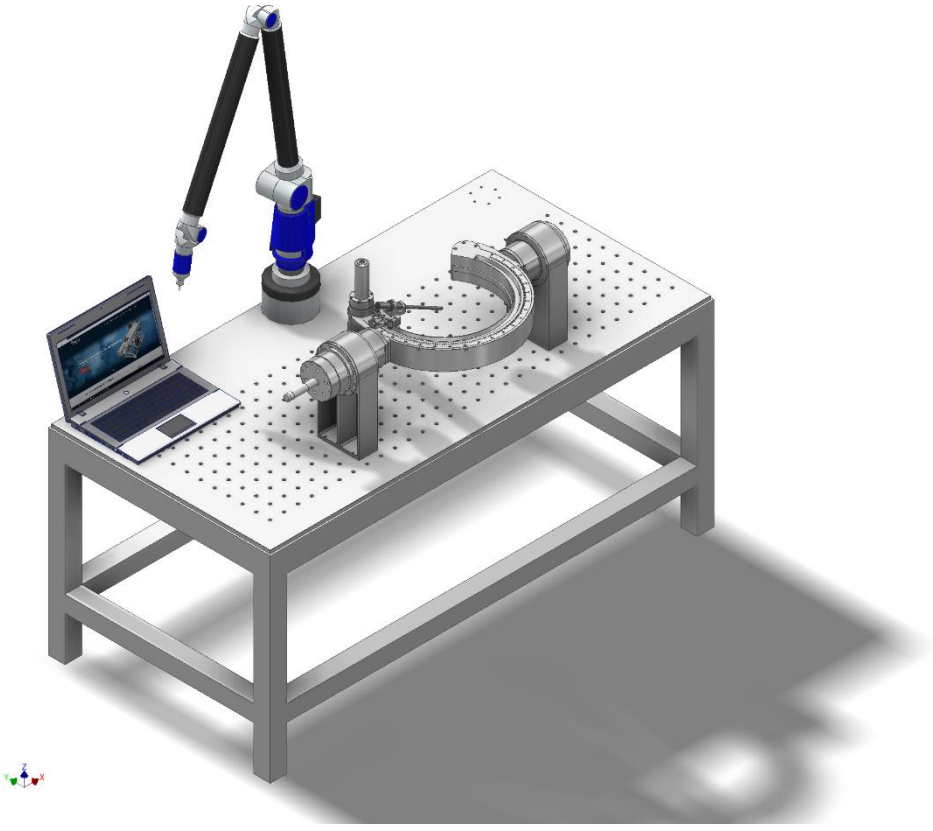
Prva faza se dijeli na tri podfaze :

E1.1. *Formuliranje tehničkog koncepta*
E1.2. *Eksperimentalno dokazivanje i procjena tehnološke izvedivosti koncepta*
E1.3. *Laboratorijska validacija integriranog prototipa*

Projekt je trenutno u fazi E1.2. U ovoj fazi se odvija izrada koncepta i te procjena izvedivosti istoga u smislu preciznosti i ponovljivosti pozicioniranja. Mjerenje preciznosti i ponovljivosti pozicioniranja će biti centralna tema ovog diplomskog rada. Unutar ovog rada će se mjeriti mogućnost preciznog i ponovljivog pozicioniranja dvije rotacijske osi projekta NERO. Rotacija luka će se dalje u radu nazivati R1, dok će rotacija modula po luku biti nazivana R2. Ove dvije radne osi (R1 i R2) su detektirane kao rizične radne osi ovakvog uređaja te su zbog toga fokus mjerenja.

Faza E1.2. u kojoj se trenutno nalazi projekt NERO se sastoji od detaljne konstrukcijske razrade eksperimentalnog postava radnih osi R1 i R2. Glavno ograničenje konstrukcije u ovoj fazi je korištenje postojeće opreme INETEC-a s ciljem ubrzavanja procesa izrade eksperimentalnog postava. Zbog tih ograničenja elektromotori i harmonic *drive* su predimenzionirani što je rezultiralo nereprezentativnim dimenzijama uređaja, gabaritno je veći nego što bi to u idealnom slučaju trebao biti. No kako je ovo faza procjene tehnološke izvedivosti naglasak eksperimentalnog postava projekta je na funkcionalnosti i preciznosti pozicioniranja uređaja s odabranim mehanizmom prijenosa snage. Sva mjerenja potrebna za provjeru preciznosti pozicioniranja će se obavljati mjernom rukom Faro Arm Quantum S.

Poveznica luka i mjernog stola, uške, ne predstavljaju stvarnu vezu luka s operacijskim stolom, već najjednostavniju poveznicu luka i mjernog stola u svrhu ispitivanja preciznosti i ponovljivosti pozicioniranja. Na Slika 11 se nalazi CAD model eksperimentalnog postava s mjernom rukom Faro Arm Quantum S, alpha prototipom NERO uređaja i upravljačkim laptopom na mjernom stolu.

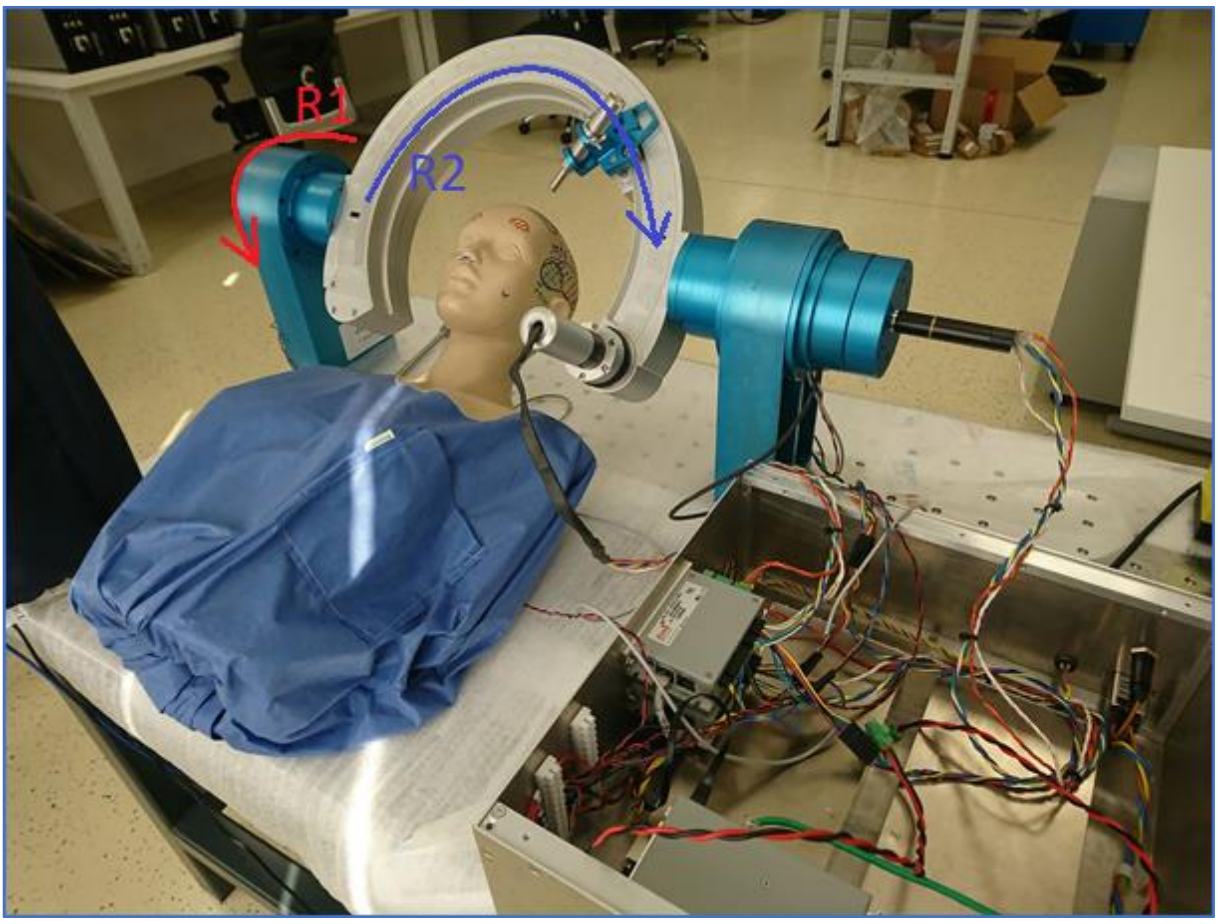


Slika 11. Eksperimentalni postav

2.1 Dizajn

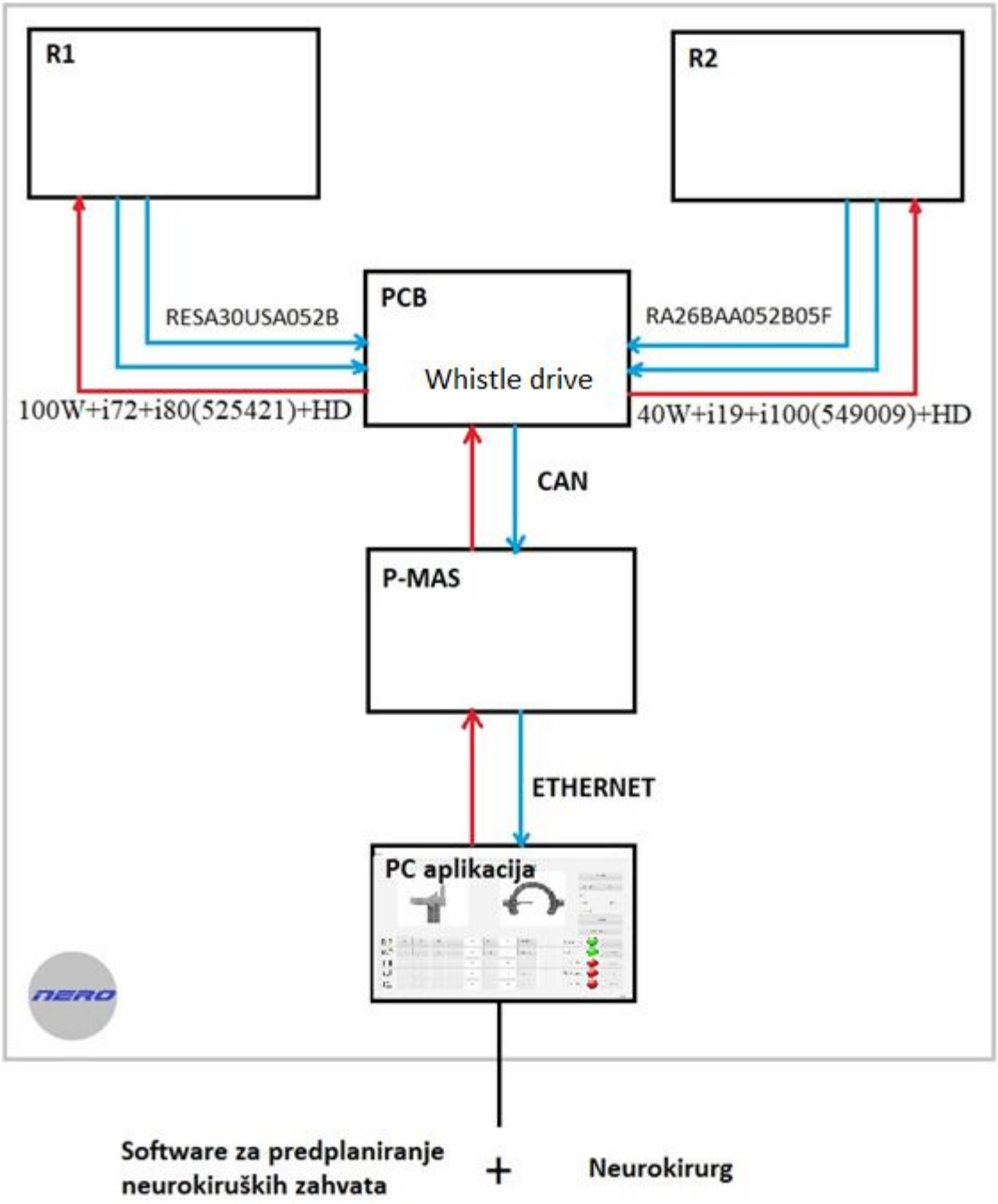
Dizajn NERO robotskog sustava se bazira na dosadašnjim stereotaktičkim okvirima. Dizajn je kao takav funkcionalnošću sličan MARS-u na način da se uzima princip pozicioniranja u lučnom sustavu.

Kao što je spomenuto ranije, eksperimentalni postav sadrži samo dvije od pet inicijalno zamišljenih osi. Te osi su osi R1 i R2 kod kojih radimo provjeru preciznosti pozicioniranja i ponovljivosti ove dvije rotacijske osi. Os R1 se sastoji od dvije uške paralelno montirane. Uške služe kao ležajna mjesta za pogonske dijelove ove osi. Na ovaj način se dobiva rotacija cijelog luka a i time zakret cijelog luka za kut R1. Na luku se nalazi R2 os kao sljedeći član lanca. Os R2 je zamišljena kao radni modul koji se giba po luku uz pomoć remenskog prijenosa. Pogonski član osi R2 se nalazi na luku zajedno s ostatkom komponenti osi R2.



Slika 12. NERO dizajn

3. Eksperimentalni postav



Slika 13. Shematski prikaz robota NERO

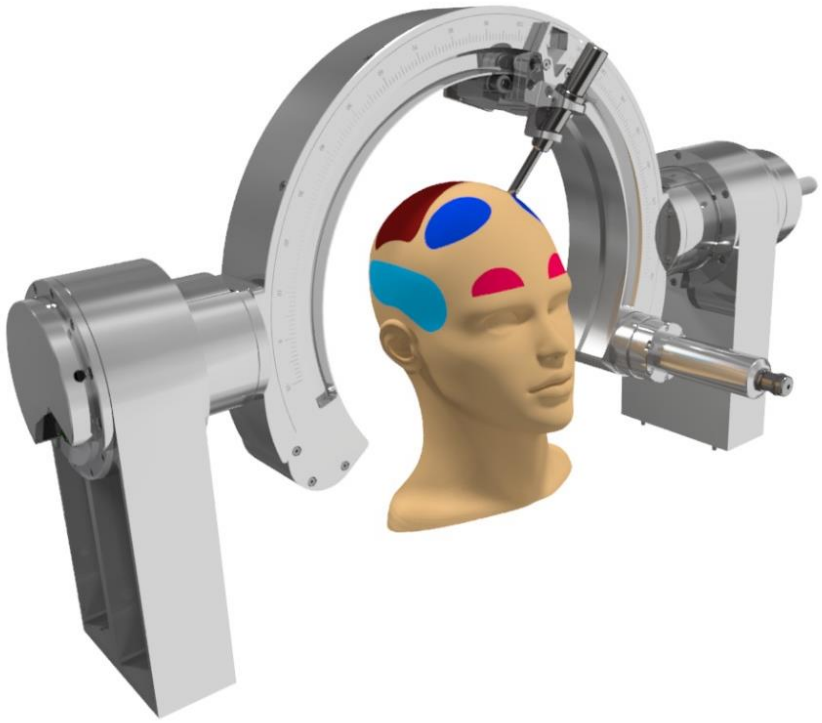
Eksperimentalni postav NERO robota se može rastaviti na 5 manjih cjelina prikazanih na Slika 13. Kao što je već prije spomenuto postoje dvije osi, R1 i R2. Svaka od osi se sastoji od pripadajućih motora i enkodera koji su upravljani Gold Whistle upravljačkim jedinicama smještenim na zajedničkoj tiskanoj pločici.

Radne osi se sastoje od motora, inkrementalnog enkodera na motoru za voženje i apsolutnog enkodera na teretu za pozicioniranje. Komponente koje se koriste su Maxon 386675

motor s ugrađenim inkrementalnim enkoderom 201937 te RA26 apsolutnim enkoderom na teretu za R1 i Maxon386660 motor s enkoderom 201940 i RLM2IC apsolutnim enkoderom na strani tereta za R2.

Sustav se sastoji od šest *Whistle* upravljačkih jedinica, po jedan za upravljanje svakom osi osim *y* osi koja se sastoji od dva paralelna motora.U trenutnom eksperimentalnom postavu koriste se samo dvije osi(R1 i R2). Upravljačke jedinice su spojene preko EtherCat serijske mreže s Platinum Maestro(P-MAS) mikrokontrolerom. Na P-MASu se odvija proračun inverzne kinematike za poziciju dobivenu preko Etherneta. P-MAS je s Ethernetom spojen Java aplikaciju od koje prima koordinate pozicija zahtijevanih od strane neurokirurga ili softwarea za planiranje zahtjeva. P-MAS inverznom kinematikom dobiva iz koordinata vrha alata pozicije pojedinih aktuatora potrebnih da bi se postiglo pozicioniranje.

Mjernom rukom Faro Arm Quantom S se tada vrši validacija preciznosti i ponovljivosti pozicioniranja vrha alata. Eksperimentalni postav će mjeriti preciznost i ponovljivost pozicioniranja ulazne točke (*entry point*) za čiju poziciju su odgovorne osi R1 i R2 kao što je to prikazano na Slika 14.



Slika 14. Eksperimentalni postav - ulazna točka

3.1 Konstrukcija R1 i R2

Konstrukcija samog eksperimentalnog postava se sastoji od dvije uške(lijeva i desna), luka, te radnog modula. Na uškama se nalazi jedan motor(Maxon 386675) koji služi za pokretanje radne osi R1 i pripadajući enkoderi. Radni modul radne osi R2 se nalazi na luku pokretanom drugim motorom(Maxon 386660).

Komponente:

1. Lijeva i desna uška
2. Luk
3. Radni modul
4. Maxon 386675
5. Maxon 386660
6. P-MAS
7. DC Whistle
8. RA26
9. RLM2IC

3.1.1 Lijeva i desna uška

Uške su konstrukcijski element u izvedbi NERO-a koji se sastoje od ležajnog mjesta za R1 te cilindričnog elementa u koje se ugrađuje luk. Unutar ležajnog mjesta luka se nalazi elektromagnet (solenoid) čija je svrha zaključavanje luka u predviđenoj poziciji. Na lijevoj uški se nalazi optički enkoder na strani tereta, HD prijenos te Maxonov servo sustav za R1.

Slika 15. Uška

3.1.2 Luk

Glavni dio konstrukcije eksperimentalnog postava je luk čija je rotacija u eksperimentalnom postavu definiran kao R1. Slika 16. prikazuje bazu samog luka. Cijeli luk se sastoji od baze, unutarnjeg nazubljenog remena sa svim pripadnim komponentama, poklopca i motora za pokretanje radnog modula, R2. Rotacija cijelog luka (R1) se odvija preko *harmonic drive* prijenosa pokretanog od strane motora smještenog u jednoj od uški. Na ovaj način se smanjuje zračnost u prijenosu između motora i samog luka koja je uzrokovana zračnošću motora te njegovog zupčastog prijenosa.

Na luku se nalazi i preko remenskog prijenosa giba radni modul o kojem će više informacija biti u nastavku rada.

Slika 16. Baza luka

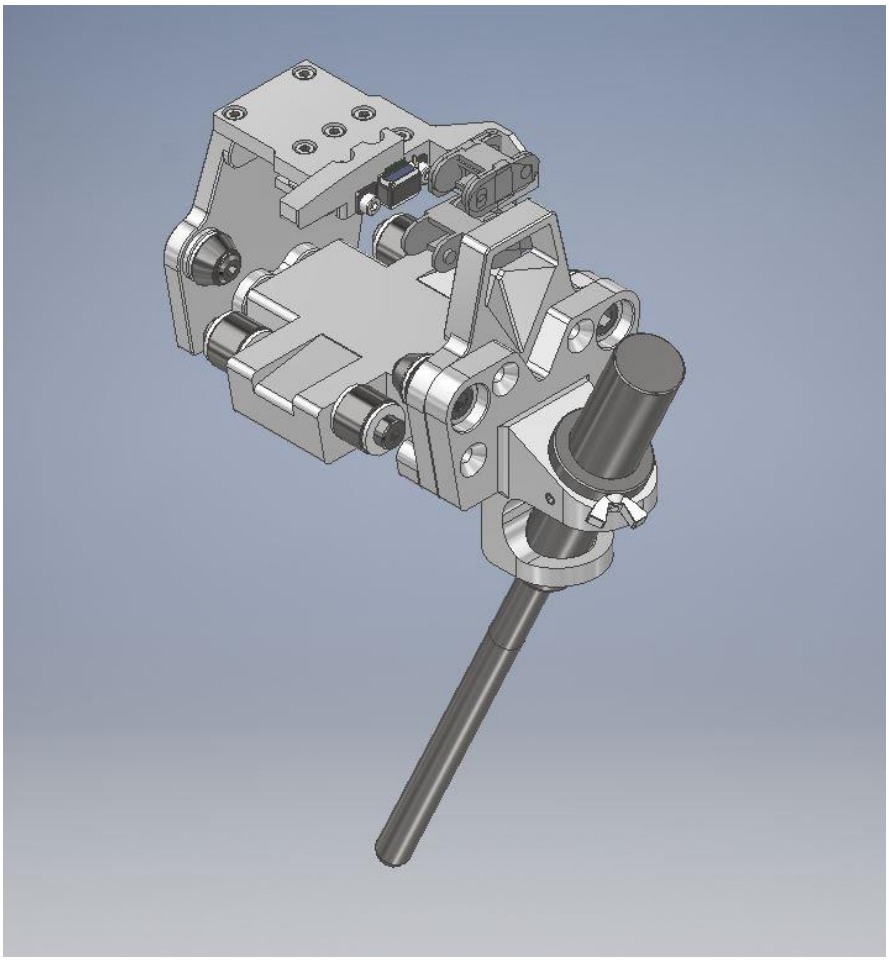
3.1.2.1 Harmonic *drive*

Harmonic *drive* ili prijenos valom naprezanja je specijalni tip mehaničkog prijenosa koji radi na principu iskorištavanja elastične dinamike i fleksibilnosti metala.

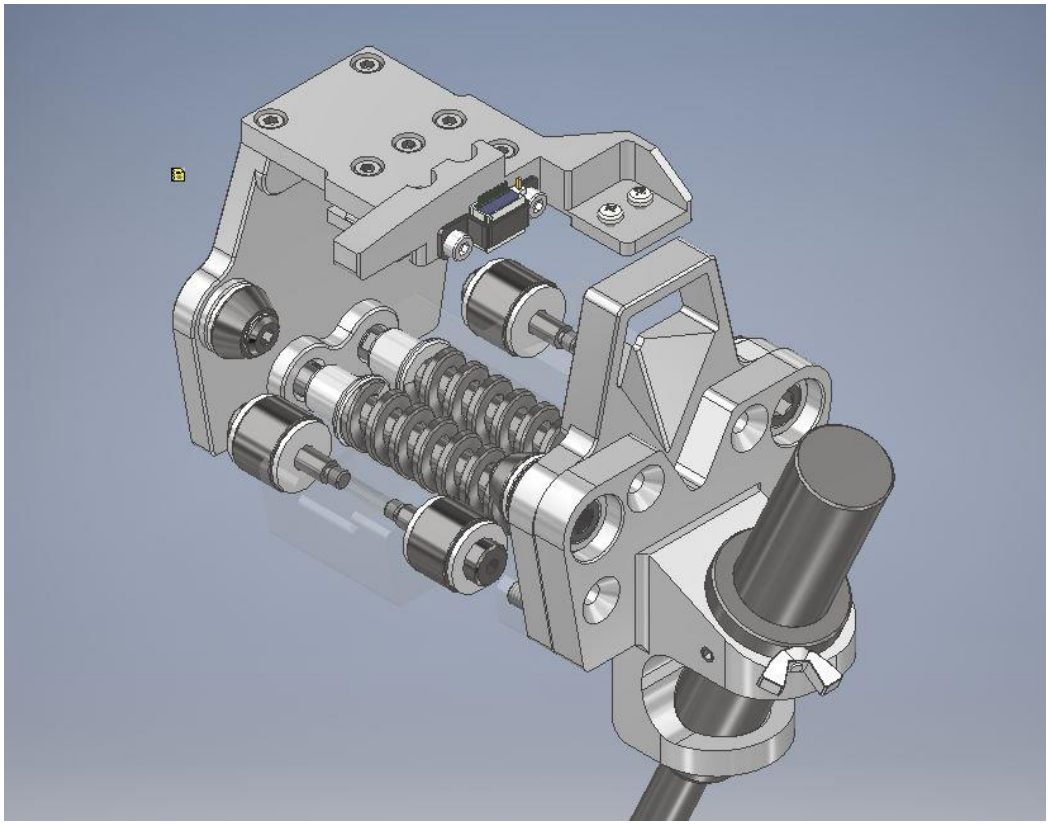
3.1.3 Radni modul

Radni modul je posljednji član kinematskog lanca. Na njemu se nalazi ciljna igla te kao takav usmjerava iglu prema ciljnoj točci.

Na sebi ima magnetni inkrementalni enkoder RLM2IC na strani tereta, induktivni senzor blizine čija svrha je pronalazak nulte pozicije inkrementalnog enkodera. Gibanje se ostvaruje remenom koji se nalazi unutar luka. Radni modul prati putanju luka oblikom. Praćenje putanje je riješeno tako da se radni modul može podešavati preko dva opružna mehanizma koja osiguravaju čvrsto prijanjanje kotačića na za to predviđene plohe luka. Na taj način osigurano je kotrljanje kotačića po luku i čvrsta pozicija radnog modula na luku.



Slika 17. Radni modul



Slika 18. Transparentni pogled na opružni mehanizam

Slika 18. prikazuje pogled na opružni mehanizam koji je dužan osigurati dovoljnu silu za ostvarivanje kotrljanja kotačića radnog modula po luku. Ove dvije opruge se nalaze unutar kučišta prikazanog na Slika 17. Zbog opružnog djelovanja opruga koje bi trebale povezivati dva elementa radnog modula, postoji mogućnost gubitka krutosti radnog modula kao cjeline.

Ovaj konstrukcijski element nosi najveću sumnju za gubitak preciznosti zbog mogućnosti nedostatka krutosti spomenutog elementa.

3.1.4 Maxon 386675

Maxonov servo sustav se sastoji od motora, enkodera za regulaciju okretne brzine te *hallovih* senzora za komutaciju. U sustav je ugrađen i redukcijski prijenos.

Vrijednosti na nominalni napon:

Nominalni napon: 48 V

Nominalna struja: 3.45 A

Nominalni moment: 47.6 mNm

Struja bez tereta: 149 mA

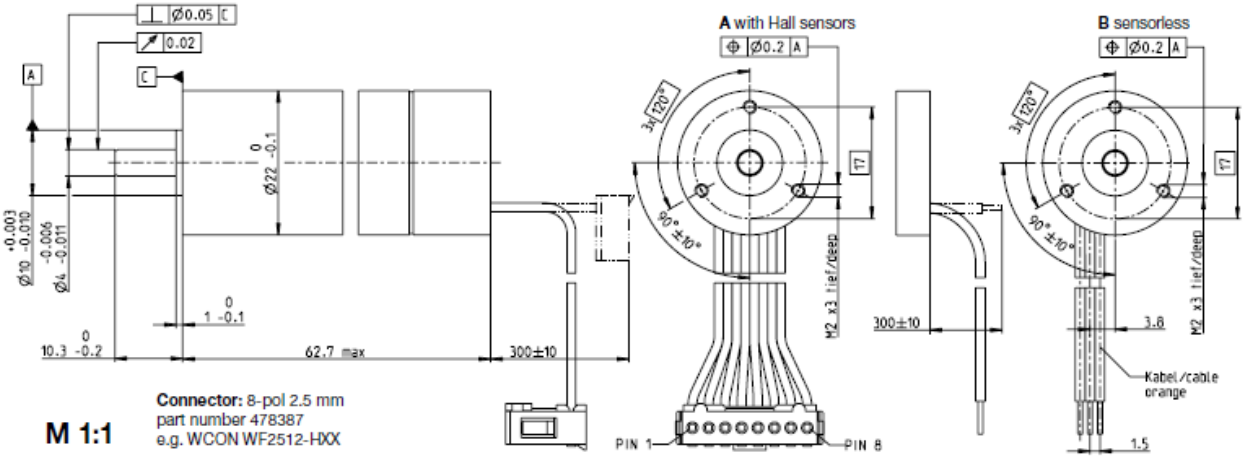
Mehanički podatci:

Tip ležaja: kuglični ležaj

Maksimalno aksijalno opterećenje: 3.5 N

Maksimalno radijalno opterećenje 16 N

Masa: 128 g



Slika 19. Dimenzije motora Maxon 386675 [5]

3.1.5 Maxon 386660

Maxon servo sustav se kao i prethodni sastoji od motora, enkodera za regulaciju okretne brzine te *hallovih* senzora za komutaciju. U sustav je ugrađen i redukcijski prijenos.

Vrijednosti na nominalnom naponu:

Nominalni napon: 48 V

Nominalna struja: 1.56 A

Nominalni moment: 20.1 mNm

Struja bez tereta: 69.3 mA

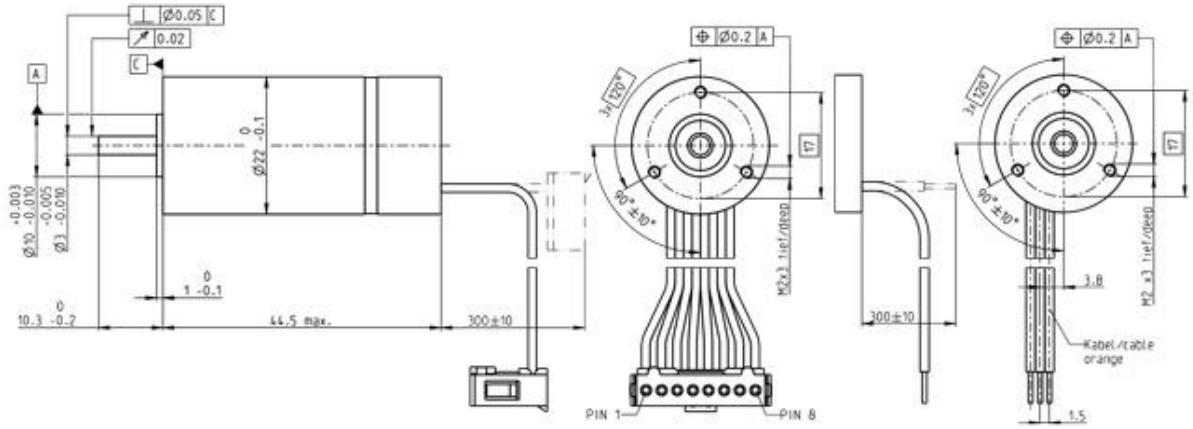
Mehanički podatci:

Tip ležaja: kuglični ležaj

Maksimalno aksijalno opterećenje: 4 N

Maksimalno radijalno opterećenje 16 N

Masa: 85 g



Slika 20. Dimenzije motora Maxon 386660 [5]

Slika 21. Radno područje Maxonovog motora 386660 i 386675 [5]

3.1.6 P-MAS

Platinum Maestro je kontroler gibanja tvrtke Elmo. Rad mu se bazira na mrežnim sustavima u kombinaciji s Elmovim servo upravljačkim jedinicama u svrhu dobivanja višeosne kontrole gibanja sustava.

Platinum Maestro sadrži visoku razinu računalne moći unutar svog dual-core sustava (2x 1.5 GHz) s velikom količinom memorije (RAM, SD kartica i ROM) na raspolaganju korisniku te dodatnu hardware periferiju.

Platinum Maestro omogućava:

- Samodostatnu kontrolu gibanja – bez potrebe konekcije s osobnim računalom
- Vremenski deterministička kontrola gibanja, ulaza/izlaza i ostalih procesa sustava
- Kompatibilnost s velikom većinom mrežnih i komunikacijskih protokola
- Potpunu višeosnu sinkronizaciju gibanja u realnom vremenom

Platinum Maestro nam nudi kontrolu gibanja u realnom vremenu za sustave s više osi. Ima mogućnost programiranja preko raznih platformi, uključujući IEC-61131-3 standardne jezike, C i C++. Time drastično ubrzava korisničku izvedbu programa i zahtjeva.

Niska razina komunikacije s upravljačkim jedinicama i ulazno/izlaznim uređajima preko mreže se odvija preko CAT industrijskog standarda. Ovaj način komunikacije se odvija preko standardne EtherCat mreže. Isto tako, postoji i mogućnost komunikacije preko CAN protokola. No za potrebe ovog projekta odbran je EtherCat zbog redudantnosti, brzine komunikacije i konstantne provjere pristiglih paketa.

Isto tako komunikaciju može ostvarivati preko standardnih industrijskih protokola poput Ethernet TCP/IP.

Specifikacije:

- Dual Core (2x1.5 GHz) procesor
- Flash memorija: 4 GByte
- RAM: 4 GB
- Potrebno napajanje: 12V-31V
- Masa: 410 g
- Radna temperatura: -20 do 85°C

- Upravljanje do 96 osi

Slika 22. Platinum Maestro [6]

3.1.7 DC Whistle

DC Whistle je digitalna servo upravljačka jedinica koji ima kontinuiranu snagu do 1.6kW. Napredni je uređaj koji ima visoku razinu upravljanja, napredni mrežni radi i unaprijed ugrađene mjere sigurnosti. Uz to što služi za kontrolu gibanja, ima mogućnost odrađivanja logičkih zahtjeva do određene razine. Napaja se DC izvorom s 12-95V te kao takav ne mora, ali može imati pomoćno napajanje za logiku (Iako se preporuča imati razdvojene napone za pogon motora i logiku samog uređaja).

Glavne prednosti ovog uređaja su:

- Više opcija komunikacije: EtherCat, Can, CanOpen, USB, Ethernet
- Mogućnost naprednog filtriranja izlaznog signala
- Vektorska kontrola sinusne komutacije

Uz kontrolu motora posjeduje i deset digitalnih ulaza/izlaza te jedan analogni. Što je pogodno za dodavanje raznih perifernih komponenti sustava

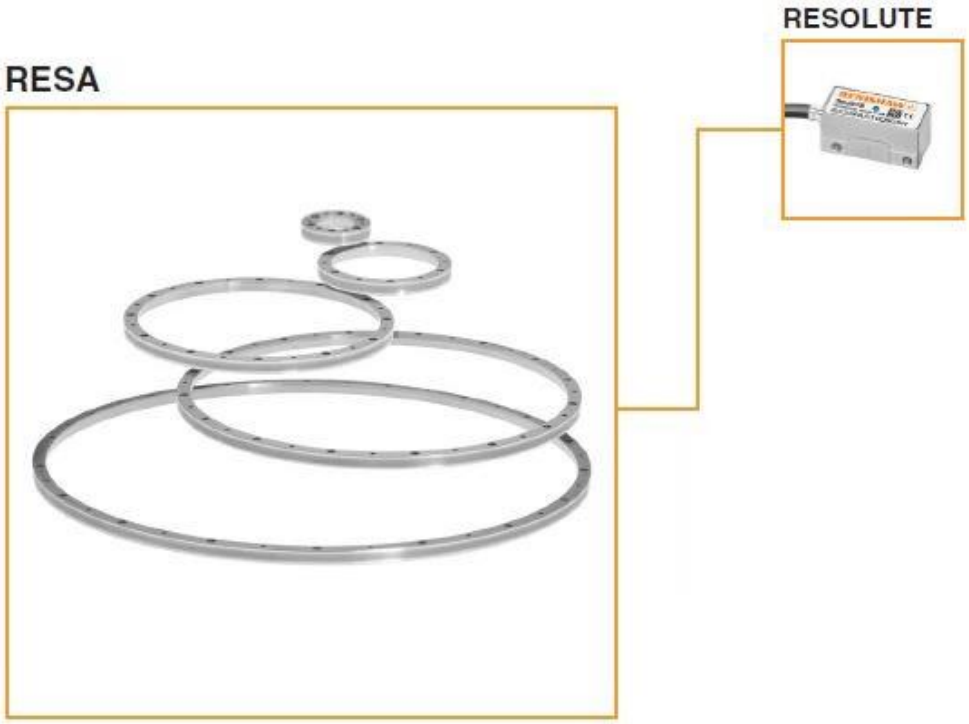
Slika 23. DC Whistle ServoDrive [7]

3.1.8 RESA+RA26

Ovaj optički enkoderski sustav sastoji se od čitače glave i prstena s ugraviranim kodom pozicija. Čitača glava se nalazi na statičnom dijelu uške dok se prsten nalazi na teretu sustava. Enkoder direktno čita kutni pomak osi R1.

Specifikacije enkodera:

- Vanjski promjer prstena: 52mm
- Preciznost sustava: ±5.49 kutnih sekundi
- Rezolucija: 26-bit(67108864 čitanja po krugu)-0.019 kutnih sekundi



Slika 24.Optički enkoder RESA+RESOLUTE RA26 [8]

3.1.9 RLM2IC

RLM2IC je magnetski enkoder koji se sastoji od čitače glave i pripadnog magnetskog prstena ili skale. U eksperimentalnom postavu NERO-a enkoderski sustav je izveden kao kombinacija čitače glave i magnetske skale. Čitača glava se nalazi na radnom modulu, dok je magnetska skala postavljenja na zakrivljenu podlogu luka. Na ovaj način čitača glava može mjeriti duljinu prevaljenu po luku. Ta duljina se konvertira u zakretni kut.

Neke od važnijih specifikacija komponente:

- Rezolucija čitaće glave i magnetske trake: ≈ 3.906 μm
- Broj očitanja(*counts*) na 2 mm: 512
- Preciznost:±40 μm
- Ponovljivost: bolja od mjere rezolucije za jednak smjer gibanja

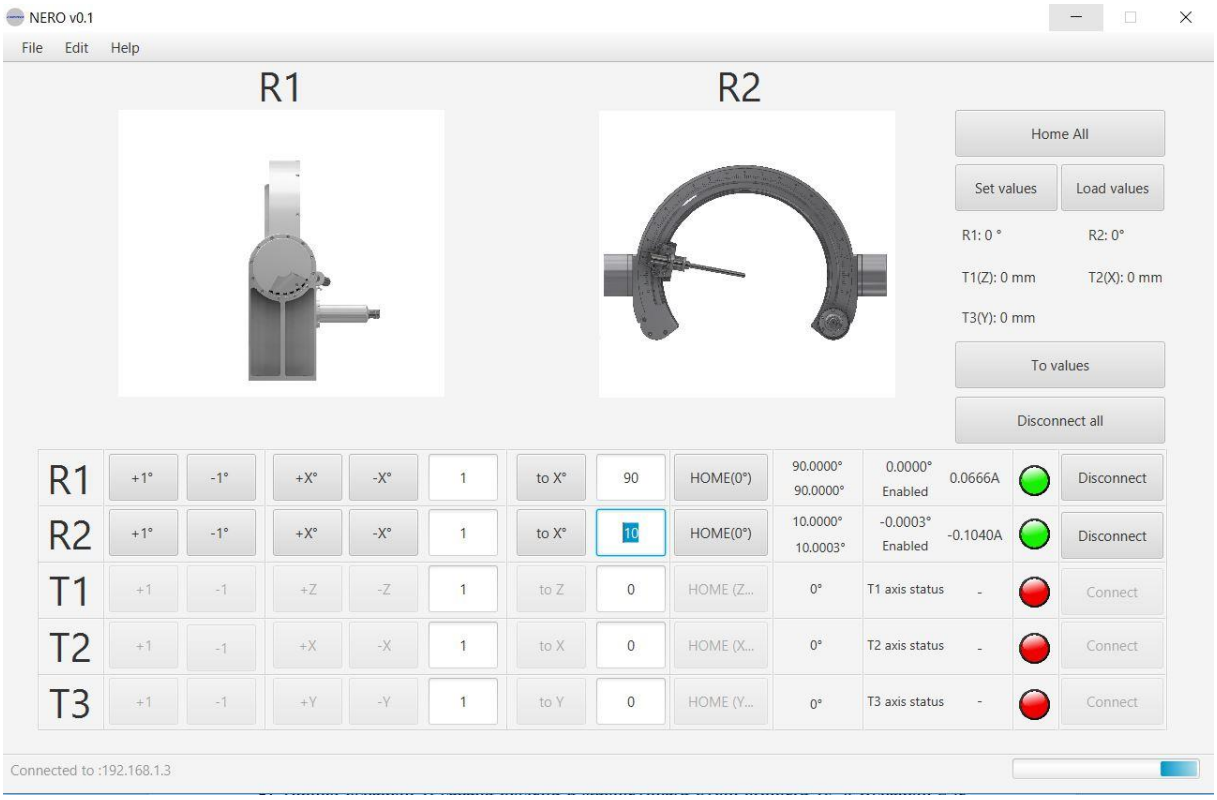
Uz čitaču glavu koristi se magnetska traka oznake MS05BM duljine 800mm koja pokriva sav potreban radni prostor osi R2.

Slika 25. Magnetski enkoder RLM2IC čitača glava [9]

3.2 Izvedba Alpha prototipa

3.2.1 Korisničko sučelje

Eksperimentalni postav je upravljan preko PC-a aplikacijom izrađenom u Java programskom alatu. Sama aplikacija je poveznica između korisnika i robotskog postava. Aplikacija ima razne funkcionalnosti potrebne za upravljanje robotom. Za potrebe ovog rada i zbog trenutnog stanja eksperimentalnog postava, na aplikaciji postoje funkcije za upravljanje radnim osima T1, T2 i T3, ali su onemogućene. Glavne upravljačke funkcije su gibanje osi za 1°, za proizvoljan broj stupnjeva te mogućnost zadavanja apsolutne pozicije ulazne odnosno ciljne točke u stupnjevima.

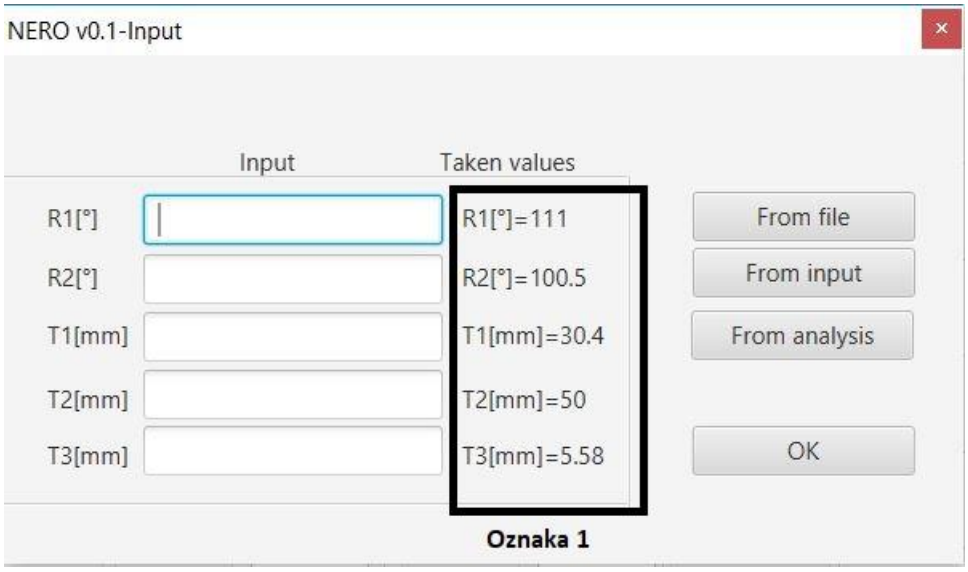


Slika 26. Glavni prozor upravljačkog sučelja

Grafičko sučelje prikazuje i neke korisne povratne informacije sa samog uređaja. Na Slika 27 vidimo, oznakom 1, zadanu poziciju u stupnjevima (u ovom primjeru 15°). Oznakom 2 je prikazan prostor prikaza trenutne stvarne pozicije očitane s optičkog enkodera radne osi R1. Oznaka 3 prikazuje grešku između zadane pozicije i očitanje pozicije enkodera. Oznaka 4 javlja status osi (Enabled, Disabled, Moving). Oznaka 5 prikazuje iznos struje pojedine radne osi.

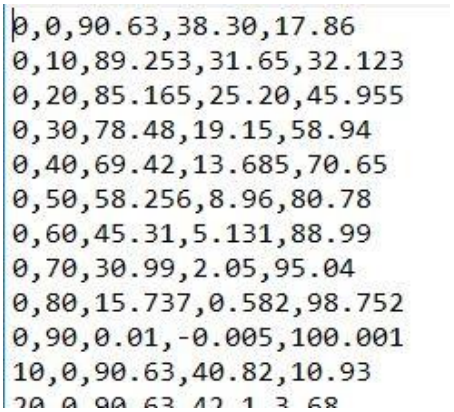
Slika 27. Prikaz stanja osi

S desne strane imamo opcije dovođenja manipulatora u HOME (0°,0°) poziciju, te tri gumba *Set values, Load values* i *To values.* Njihova zadaća je omogućavanje upravljanja svih radnih osi simultano, vođenih po apsolutnoj poziciji. Gumb *Set values* otvara novi prozor za zadavanje apsolutnih vrijednosti pozicije manipulatora. Na prozoru se nalaze i tri opcije mjesta povlačenja vrijednosti upravljanja manipulatora po apsolutnoj poziciji. Prva opcija je povlačenje vrijednosti pozicija radnih osi iz vanjske datoteke u kojoj se nalaze vrijednosti željenih pozicija u točno određenom formatu. Druga opcija je uzimanje vrijednosti pozicija radnih osi unesenih u prostore za unos na lijevoj strani samog prozora. Treća opcija uzima vrijednosti nakon analize mjerenja točnosti pozicioniranja kao rezultat kompenzacije greške. Odabrane vrijednosti su prikazane na mjestu Oznake 1 na Slika 28.



Slika 28. Odabir vrijednosti apsolutnog pozicioniranja radnih osi

Aplikacija može obavljati analizu greške pozicioniranja ovisno o Faro Arm-ovim mjerenjima. U prozoru za analizu imamo tablicu s trenutnim pozicijama radnih osi R1 i R2 u stupnjevima. Vrijednosti tražene ulazne točke u prostoru konvertiranu iz sfernog koordinatnog sustava u kartezijev. Stupac *Measured* prikazuje vrijednosti *x, y* i *z* izmjerene s mjernom rukom. Navedene vrijednosti su skup mjerenih vrijednosti upisanih u posebnu tekstualnu datoteku koju sama aplikacija kod inicijalizacije čita. Format zapisa je u obliku „*R1,R2,x,y,z\n"*. Datoteka u kojoj se nalaze se zove *faro.txt.*



Slika 29. Format zapisa izmjerenih vrijednosti

Shodno tome, u točkama za koje postoje mjerenja se izračunava i kreira datoteka koja sadrži sva odstupanja izmjerenih vrijednosti od teorijskih. Datoteka odstupanja je formata istog kao i datoteka izmjerenih vrijednosti. Na taj način imamo matricu vrijednosti grešaka izbačenih u posebnu datoteku koju možemo koristiti u bilo kojem drugom programu u svrhu provedbe analize grešaka. Ove vrijednosti su spremljene u datoteku *error.txt.*

Mreža izmjerenih točaka točnije grešaka uređaja je baza s konačnim brojem točaka. Kako bi dobili sve točke potrebne za precizno pozicioniranje na bazu točaka je potrebno izvršiti bilinearnu interpolaciju između četiri susjedne točke. Na ovaj način dobivamo vrijednosti greške potrebne za izračun nove korigirane vrijednosti kuta pozicioniranja.

Stupac *Error* prikazuje grešku između teorijskih koordinata i stvarnih izmjerenih. Stupac *New target* prikazuje nove vrijednosti *x, y* i *z* u koje je uračunata greška te predstavlja novu vrijednost u koju moraju doći radne osi da bi postigli željenu točku ulaza. Nova točka je zatim konvertirana iz kartezijevog nazad u sferni koordinatni sustav. Razlika između novih i zadanih kutova za radne osi R1 i R2 je offset potreban za kompenzaciju greške uređaja.

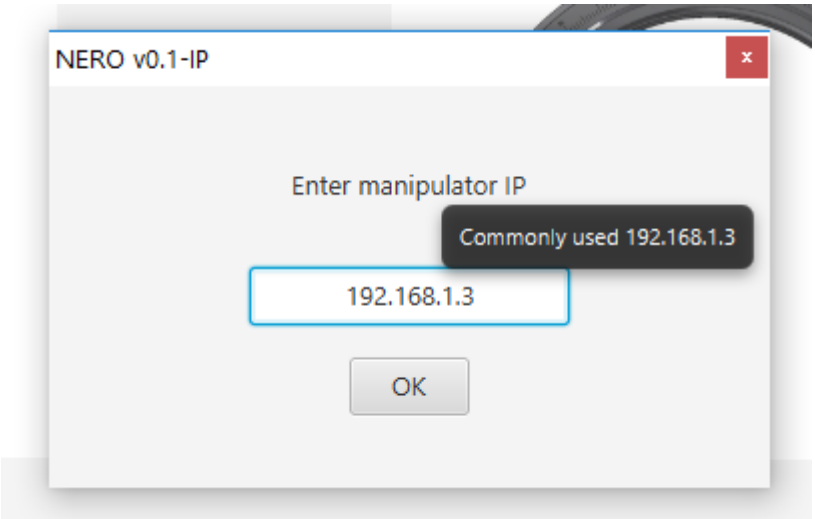
Korištena aplikacija unutar ove analize radi i transformaciju globalnih koordinata sustava u koordinatni sustav samog manipulatora.

Slika 30. Prozor za analizu i kompenzaciju pogreške

Više o samoj analizi i kompenzaciji pogreške pozicioniranja prolazit će se u poglavlju 4 (Analiza).

Sama aplikacija ima i grafički prikaz pozicija radnih osi R1 i R2 preko dva pogleda manipulatora čija je pozicija vidljiva sa Slika 26. Isto tako postoji i 3D prikaz istoga. Animacija prikaza je u inkrementima od 1° po svakoj radnoj osi te ima ograničenja od -10° do 190° za R2 i -25°do 200°za R1. Ovo su i fizička ograničenja uređaja.

Komunikacija sa samim robotskim sustavom tj. PMAS mikrokontrolerom se odvija preko TCP/IP ethernet protokola. Unosom IP adrese uređaja uspostavlja se komunikacijska veza s robotskim sustavom. Status konekcije se može vidjeti na statusnoj traci u donjem lijevom kutu kao što je to prikazano na Slika 31.



Slika 31. Unos IP adrese uređaja

3.2.2 P-MAS mikrokontroler

Mikrokontroler obavlja sve funkcije potrebne robotskom sustavu za pozicioniranje u željenoj točci. S *driveovima* je spojen u serijsku EtherCat mrežu. Program glavnog kontrolera obavlja funkciju komunikacije s PC aplikacijom opisanom u poglavlju ranije. Ovisno o naredbama primljenim preko mreže obavlja zadane naredbe kao što su paljenje/gašenje motora pojedinih osi, gibanje i zaustavljanje istih itd. Paralelno se događa povrat čitanja enkodera i struje na motorima.

Konverzija stupnjeva lokalnog koordinatnog sustava u broj inkrementa enkodera se odvija unutar mikrokontrolera. Isto tako, program radi provjeru stanja osi te o tome obavještava korisnika u slučaju grešaka.

3.2.3 Regulacija pozicioniranja osi

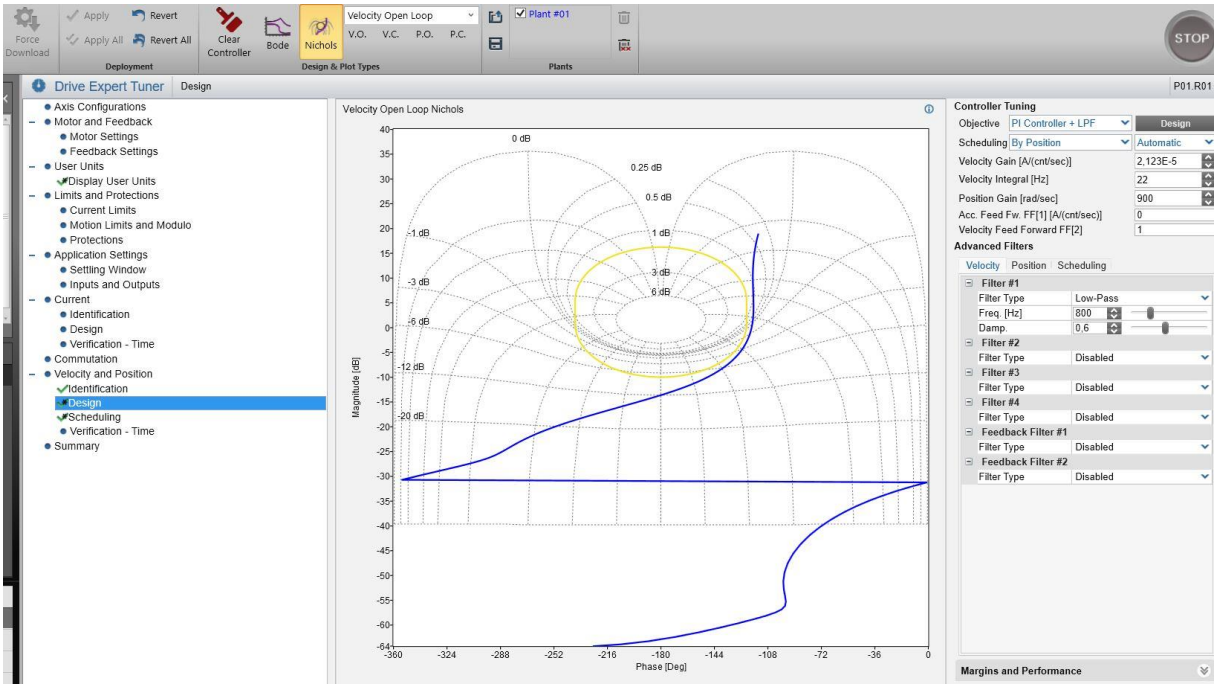
Regulacija osi R1 i R2 se odvija na dva zasebna uređaja, *drivea*. Svaki *drive* odrađuje regulacijski zatvoreni krug sa svojim pripadnim motorom i dva enkodera. Prvi enkoder koji dolazi ugrađen zajedno s motorom služi za regulaciju brzine vrtnje motora, dok se drugi enkoder montira na teret te služi kao povratni član regulacijske petlje pozicije.

Komunikacija P-MAS kontrolera s *driveovima* se odvije preko EtherCAT mreže. *Driveovi* su spojeni serijski s mikrokontrolerom master-slave principom rada. P-MAS je naravno master dok su *driveovi* slaveovi. Sama EtherCat mreža serijskim spajanjem omogućava redudantnost komunikacije (P-MAS->*drive*R1->*drive*R2->P-MAS). To sustavu otvara mogućnost lakog detektiranja pogrešaka ili nepravilnosti rada komunikacije te nastavak rada ostalih osi unutar sustava u slučaju kvara jedne.

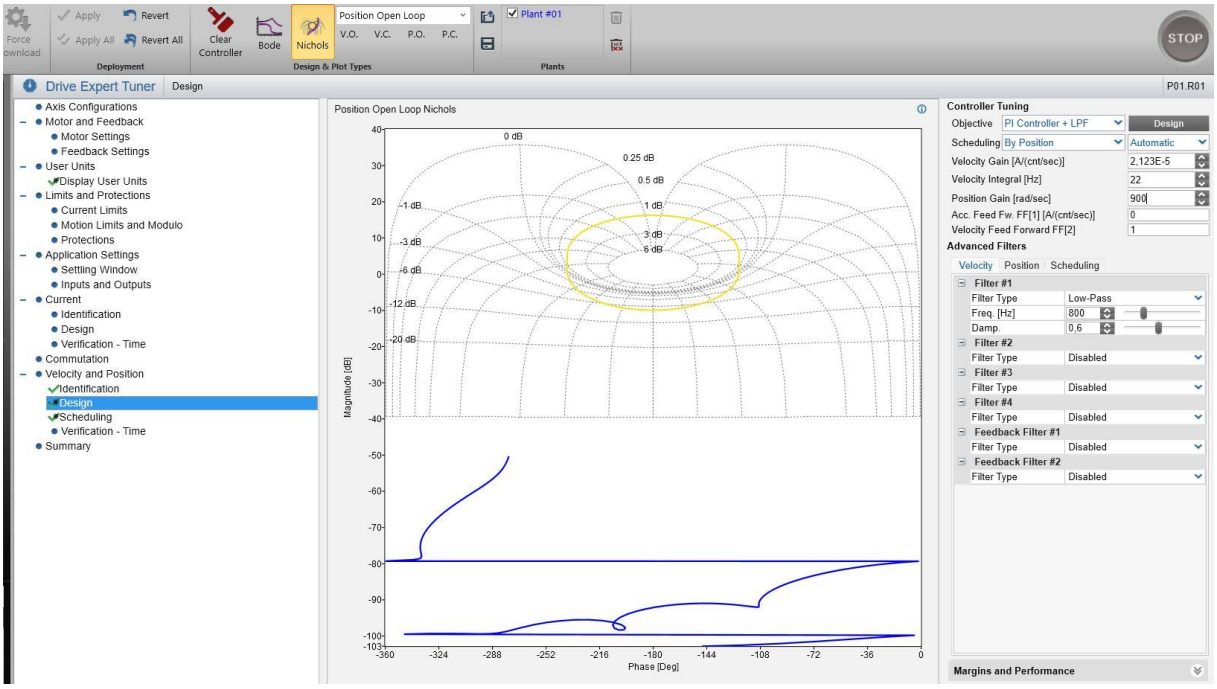
Os R1 se sastoji od Maxon 386675 motora, njegovog pripadnog inkrementalnog enkodera (201937) na osovini i RA26 apsolutnim enkoderom na teretu, tj. na rotacijskom dijelu uške nakon svih prijenosa. Motor je u regulacijskoj petlji brzine s ugrađenim inkrementalnim enkoderom (201937) kao povratnim članom, dok s RA26 apsolutnim enkoderom zatvara regulacijsku petlju pozicije.

Dizajn regulacijskog kruga se radi u ELMO-vom alatnom programu EAS, tako da se prvo odradi identifikacija strujnih karakteristika motora. Nakon toga se obavlja identifikacija članova petlje brzine i petlje pozicije te se po tim parametrima dizajnira regulacijski krug. Mijenjanjem vrijednosti integralnih i proporcionalnih članova regulacijskog kruga dizajniramo

krug tako da u Nicholsovom dijagramu funkcija kruga ne ulazi u nestabilno područje rada niti po brzini niti po poziciji.



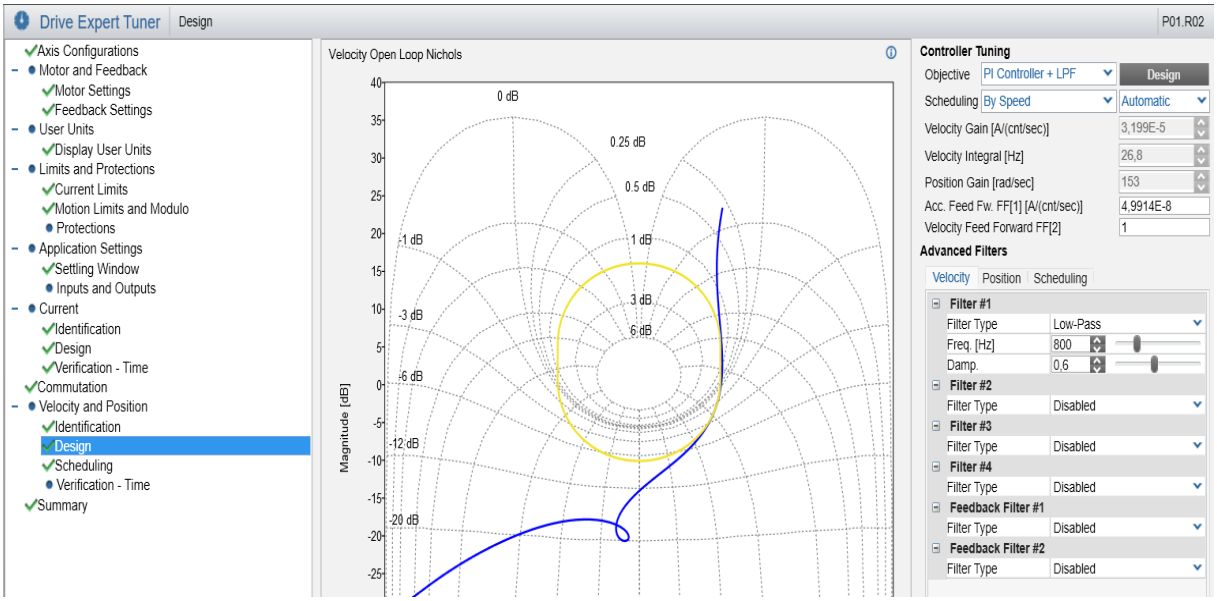
Slika 32. Nicholsov dijagram za regulacijski krug brzine za os R1



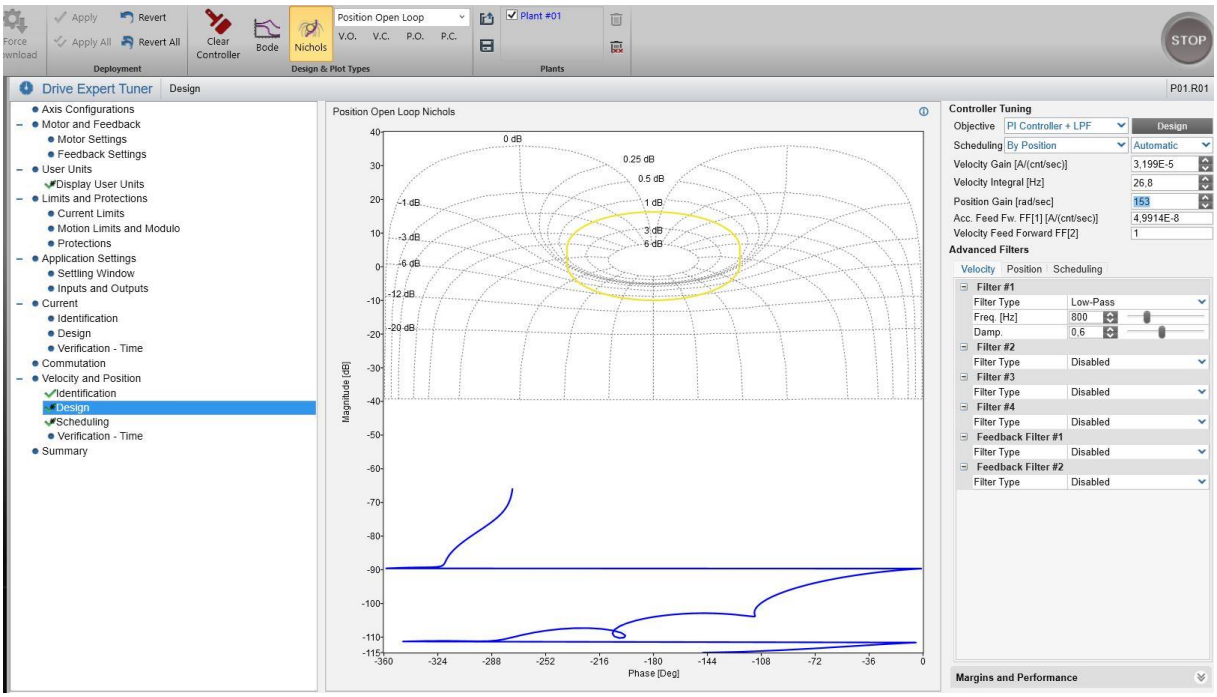
Slika 33. Nicholsov dijagram za regulacijski krug pozicije za os R1

Za regulacijski krug R2 koristi se motor Maxon 386660, njegov pripadni inkrementalni enkoder(201940) na osovini i RLM2IC magnetni apsolutni enkoder na strani tereta. Motor je u

regulacijskoj petlji brzine s inkrementalnim enkoderom na osovini(201940) i regulacijskoj petlji pozicije s magnetnim absolutnim enkoderom RLM2IC.



Slika 34. Nicholsov dijagram za regulacijski krug brzine os R2



Slika 35. Nicholsov dijagram za regulacijski krug pozicije os R2

3.3 FARO QUANTUM S

Mjerni uređaj korišten u ovom radu je FaroArm QuantumS. QuantumS je višeosna mjerna ruka sa sfernim radnim volumenom. Svaki zglob mjerne ruke sadrži optički enkoder pomoću kojih računa položaj ticala. Sustav ima mogućnost dodirnog i laserskog skeniranja. Dolazi u četiri različite izvedbe definirane preko radnog volumena ruke, QuantumS korišten u ovom radu je QuantumS 2.5m. QuantumS je najtočniji FaroArm® ikad proizvoden i ima najvišu razinu izvedbe mjerenja zadovoljavajući i najzahtjevnije tolerancije. Koristi se u zrakoplovnoj, automobilskoj industriji, gdje se koristi za certifikaciju i inspekciju raznih dijelova. Koristi se u svrhu periodičnih inspekcija kod izrade metala te raznih drugih certifikacija i inspekcija. QuantumS je prvi FaroArm koji zadovoljava ISO 10360-12:2016 standard.

Tablica 1. Prikaz grešaka u mjerenju

Mjerni domet	SPAT[1]	E_{UNI}[2]	P_{SIZE}[2]	P_{FORM}[2]	L_{DIA}[2]
QuantumS 1.5m	0.012mm	0.023mm	0.008mm	0.015mm	0.027mm
QuantumS 2.5m	0.018mm	0.028mm	0.010mm	0.020mm	0.035mm
QuantumS 3.5m	0.036mm	0.056mm	0.020mm	0.040mm	0.070mm
QuantumS 4.5m	0.045mm	0.068mm	0.024mm	0.045mm	0.086mm

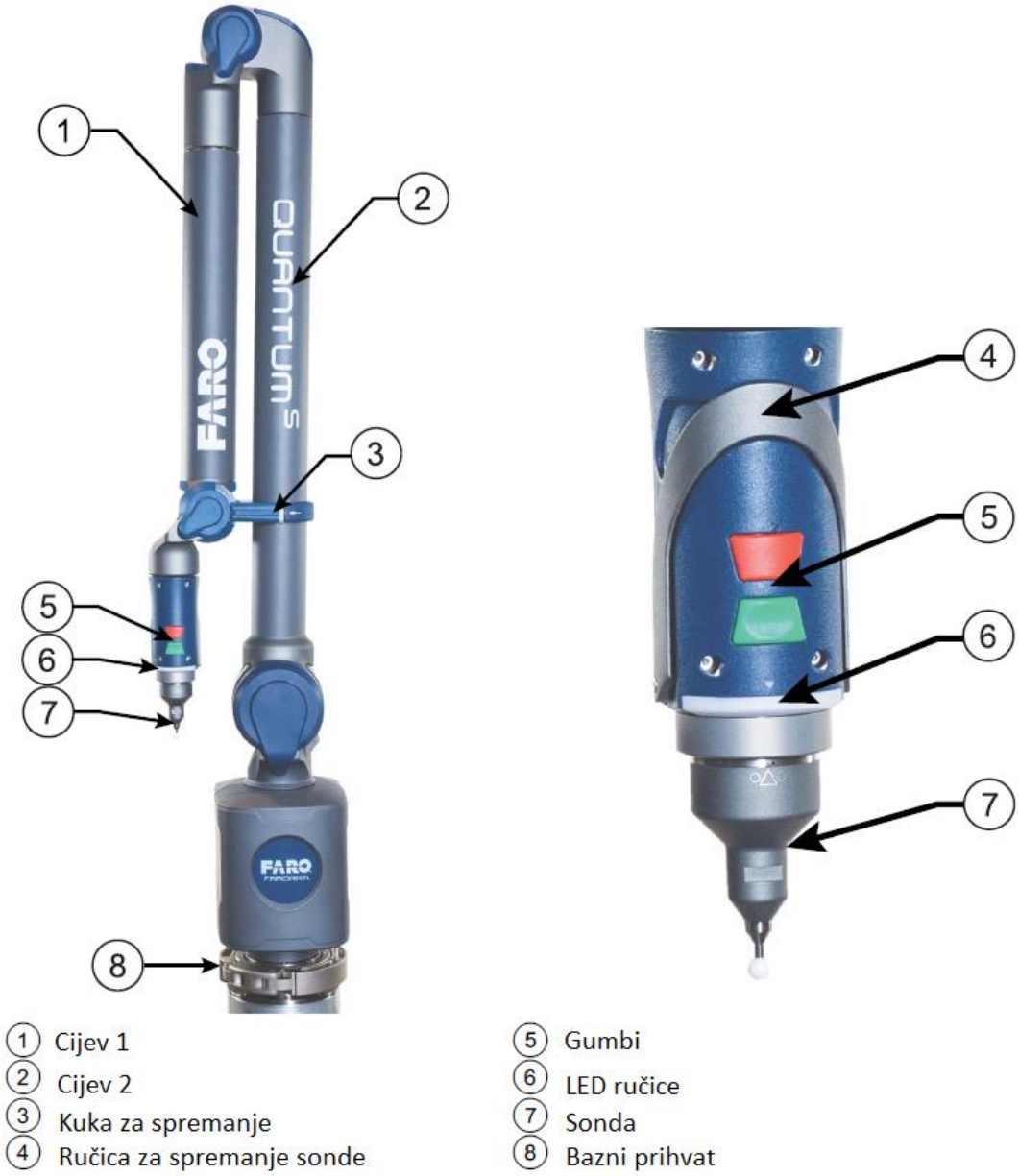
SPAT[1] - Ispitivanje jednostruke točke

E_{UNI}[2] - Greška udaljenosti između dvije točke uspoređujući mjerene i nominalne vrijednosti

P_{SIZE}[2] - Greška mjerenja sfere uspoređujući mjerene i nominalne vrijednosti

P_{FORM}[2] - Greška mjerenja oblika sfere

L_{DIA}[2] - Greška promjera položaja sfere



Slika 36. Prikaz dijelova FaroArm QuantumS-a [10]

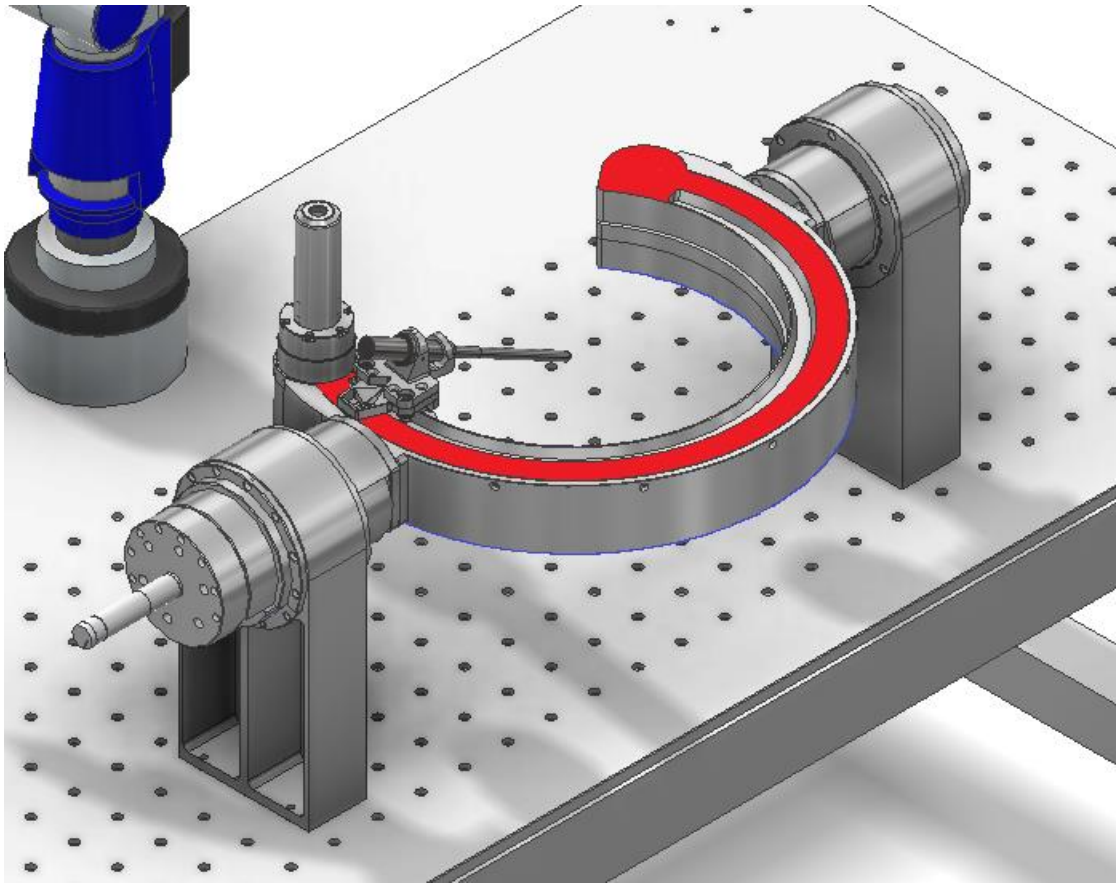
4. Analiza pogreške pozicioniranja

U ovom poglavlju će biti pobliže opisan način mjerenja, analiza pogreške pozicioniranja, izračun kompenzacije i provjera točnosti nakon kompenzacije. Cilj je izmjeriti što točnije mjernu točku, eliminirati greške mjerenja te usporedbom idealne i mjerene točke kreirati kompenzacijski *offset* nakon čega bi se razlika između idealne i mjerene točke trebala smanjiti na minimum.

4.1 Mjerenje

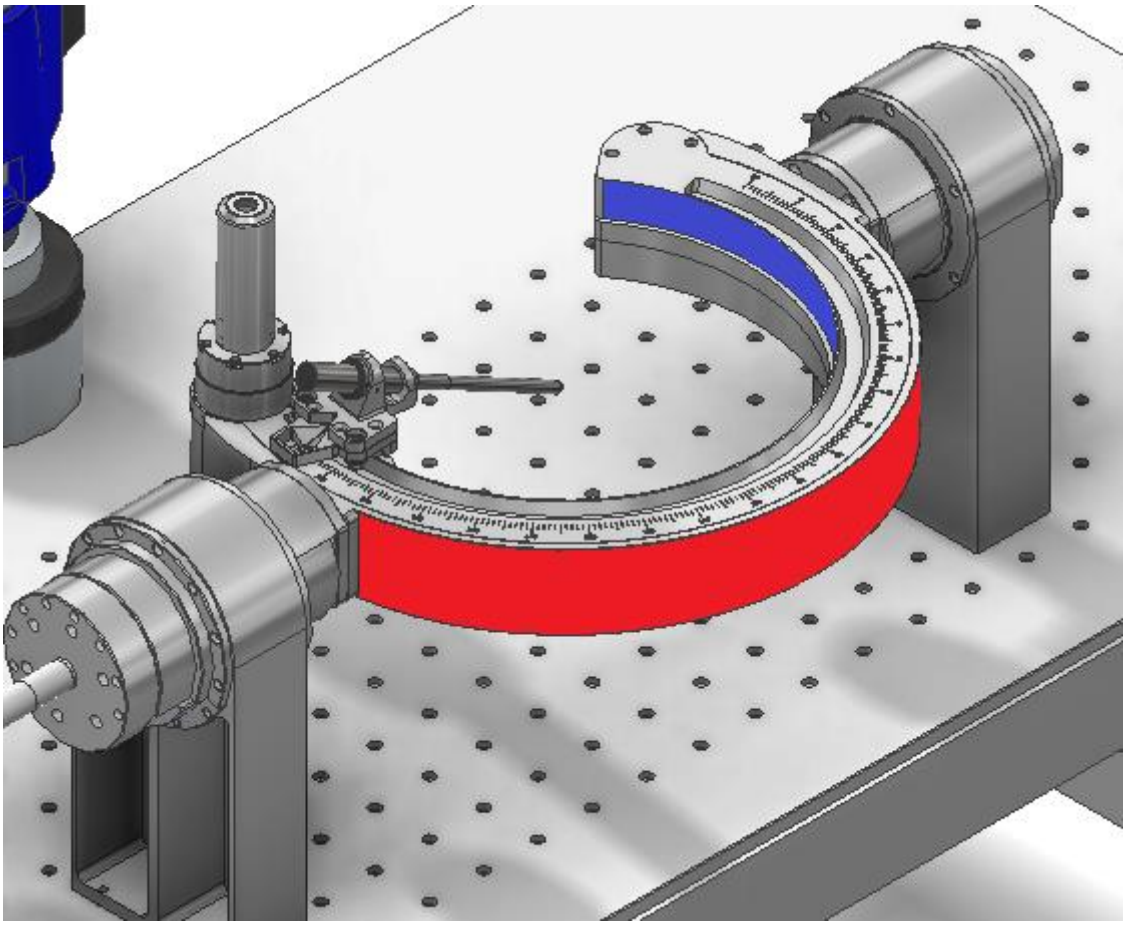
4.1.1 Postupak mjerenja

1) Postavljanje koordinatnog sustava manipulatora. Prvi korak koji radimo je mjerenje plohe stola. Stol je podloga na koju su svi elementi manipulatora vezani s točno određenim položajima. Nakon dobivene informacije o plosi stola, radimo mjerenje gornje i donje plohe luka (Slika 37, crveno i plavo) te konstruiramo plohu na polovini između te dvije plohe. Tu kreiranu plohu zatim umjeravamo u položaj paralelan sa plohom stola.



Slika 37. Umjeravanje luka (Crveno gornja ploha, plavo donja)

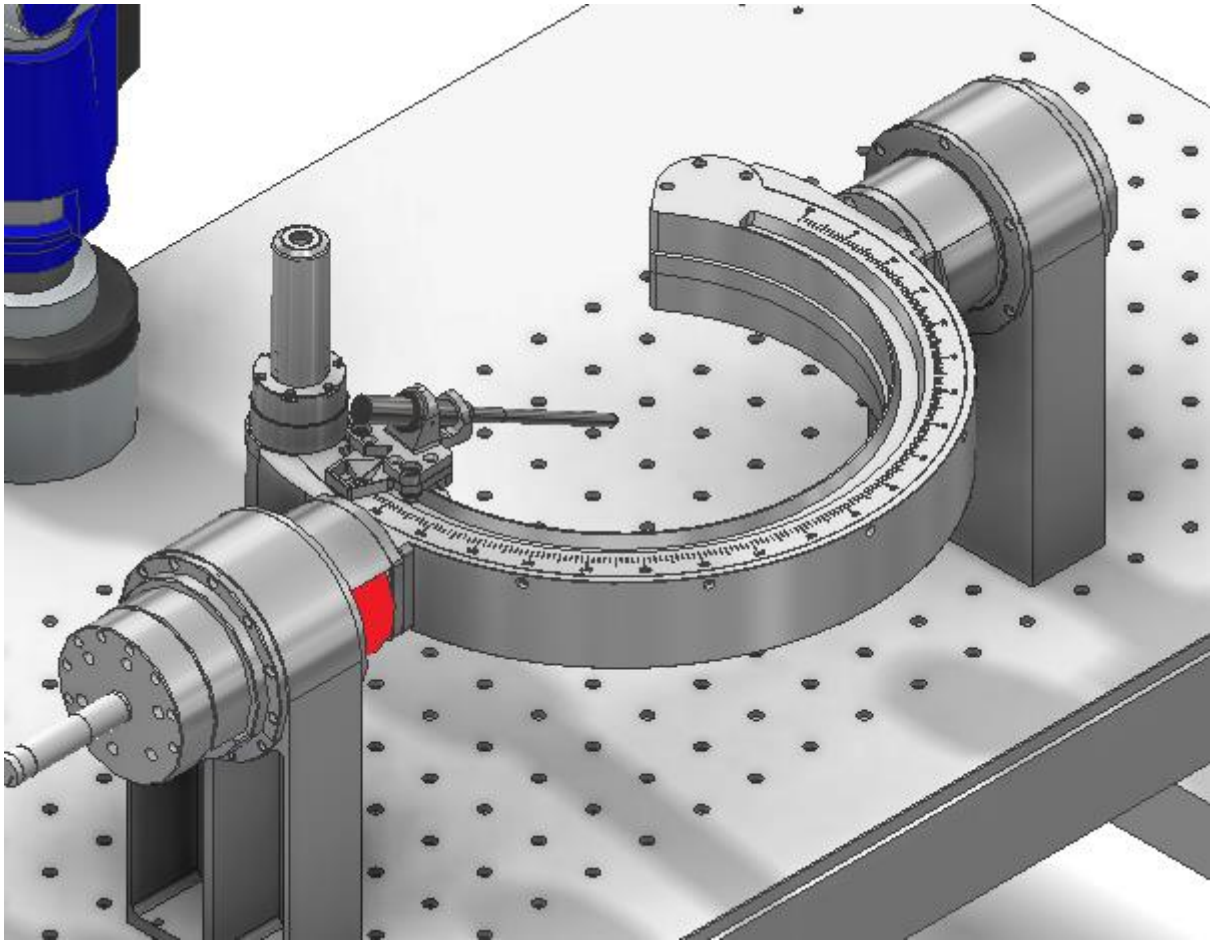
Sljedeći korak je određivanje središta koordinatnog sustava manipulatora u centru luka. Za ovo odabiremo postupak „Ploha-kružnica-kružnica“, gdje za gornju plohu luka odabiremo plohu luka paralelnu s plohom stola. Za kružnice odabiremo unutarnju i vanjsku plohu luka, kao što je prikazano na Slika 38.



Slika 38. Unutarnja i vanjska kružnica (plavo i crveno)

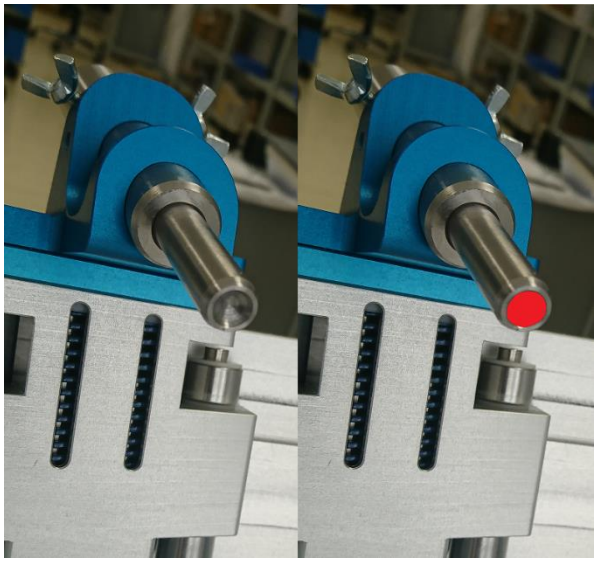
Trenutni centar sustava nam se nalazi na plohi mjerenoj u prethodnom koraku, zato radimo pomak centra pomoću funkcije „Level to“. Odabiremo plohu konstruiranu iz mjerenih ploha, gornje i donje plohe luka, s početka umjeravanja. Ovim korakom je centar koordinatnog sustava postavljen u centar luka, ali orijentacije osi nisu postavljene.

Da bismo postigli pravilnu orijentaciju osi, postavljamo Y os u okomicu na plohu stola. Zatim radimo mjerenje cilindra uške (označeno s crvenom na Slika 39) kroz koji postavljamo orijentaciju osi X. Ovime smo postavili globalni koordinatni sustav u lokalni sustav centara luka.



Slika 39. Orijentacija x osi prema lijevoj uški (crveno)

2) Samo pozicioniranje manipulatora je mjereno tako da se u programskom alatu odabralo mjerenje sfere kao geometrijskog tijela. Zatim je izmjeren sferni utor na vrhu igle (prikaz na Slika 40). Da bi izmjerili sferu potrebno je uzeti pet točaka s mjernom rukom.



Slika 40. Sferni utor za mjerenje pozicije

3) Nakon mjerenja u programskom alatu uzimamo udaljenost centra sfere od središta koordinatnog sustava. Dobivamo vrijednosti po osima x, y i z. Programski alat izrađuje izvještaj s svim potrebnim podatcima.
4) Točku 2 i 3 ponavljamo za sve točke mjerenja potrebne za analizu točnosti pozicioniranja.

4.1.2 Rezultati mjerenja

Samo mjerenje ulazne točke se obavlja Faro Armom prije objašnjenom metodom mjerenja. Svaka mjerna točka je izvedena kao nekoliko uzastopnih ponavljanja mjerenja iste ulazne točke te uzimanje njihove prosječne vrijednosti uz odstupanje kao konačne vrijednosti te mjerne točke. U tablici ispod se nalazi primjer mjerenja točke R1=(0°, 25°, 60°) i R2=(0°, 90°, 170°).

U tablicama su prikazane vrijednosti sve tri osi svakog od četiri mjerenja za određene kutove. Peta vrijednost u redovima *x, y, z* prikazuje teoretske vrijednosti zadane točke. U nastavku tablice se nalaze razlike pojedinog mjerenja od teoretske vrijednosti te najveće odstupanje. Iz tih vrijednosti se u zadnjem retku prikazuje sveukupno prostorno odstupanje. Sve vrijednosti su prikazane u milimterima.

Tablica 2. Mjerenje R1, R2=(0°, 0°)

R1 \ R2		0°				
		1	2	3	4	Idealno
0°	x	90.773	90.794	90.782	90.802	90.6308
	y	38.334	38.304	38.284	38.307	38.3022
	z	18.058	18.048	18.037	18.068	17.8606
	Δx	-0.1422	-0.1632	-0.1512	-0.1712	0.1712
	Δy	-0.0318	-0.0018	0.0182	-0.0048	0.0318
	Δz	-0.1974	-0.1874	-0.1764	-0.2074	0.2074
	euler	0.245355	0.248508	0.233044	0.268974	0.268974

Mjerenje prikazano u *Tablica 2* prikazuje točke kuta R1=0°i R2=0°. Kao što je navedeno prije u tekstu, plavom bojom su označene teorijske vrijednosti točke. Narančastom su označene ćelije s najvećim odstupanjem od teorijske. Žutom bojom su označene ćelije prostornog odstupanja pojedinog mjerenja od teorijske vrijednosti. Crvenom je označeno najveće prostorno odstupanje ovakvog mjerenja. Prostorno odstupanje ovakvog mjerenja je 0.2689 mm.

Tablica 3. Mjerenje R1, R2=(0°, 90°)

R2 / R1		90°				
		1	2	3	4	Idealno
0°	x	0.352	0.314	0.332	0.321	0
	y	0.121	0.17	0.163	0.176	0
	z	100.033	100.117	100.215	100.187	100
	Δx	-0.352	-0.314	-0.332	-0.321	0.352
	Δy	-0.121	-0.17	-0.163	-0.176	0.176
	Δz	-0.033	-0.117	-0.215	-0.187	0.215
	euler	0.373676	0.375746	0.427806	0.411079	0.427806

Tablica 4. Mjerenje R1, R2=(0°, 170°)

R2 / R1		170°				
		1	2	3	4	Idealno
0°	x	-89.404	-89.196	-89.268	-89.407	-89.2539
	y	32.16	32.066	32.191	32.212	31.6511
	z	31.784	31.703	31.685	31.791	32.124
	Δx	0.1501	-0.0579	0.0141	0.1531	0.1531
	Δy	-0.5089	-0.4149	-0.5399	-0.5609	0.5609
	Δz	0.34	0.421	0.439	0.333	0.439
	euler	0.630166	0.593915	0.695997	0.670028	0.695997

Iz tablica mjerenja konstantnog kuta R1=0° je vidljivo da povećanjem kuta R2 raste i sveukupno odstupanje mjerene pozicije od teorijske. Ova pogreška će se u nastavku rada pokušati kompenzirati metodom prikazanom u odlomku Kompenzacija greške.

Tablica 5. Mjerenje R1, R2=(25°, 0°)

R2 / R1		0°				
		1	2	3	4	Idealno
25°	x	90.741	90.72	90.763	90.786	90.6308
	y	42.349	42.34	42.386	42.381	42.2618
	z	0.129	0.139	0.141	0.159	0
	Δx	-0.1102	-0.0892	-0.1322	-0.1552	0.1552
	Δy	-0.0872	-0.0782	-0.1242	-0.1192	0.1242
	Δz	-0.129	-0.139	-0.141	-0.159	0.159
	euler	0.190759	0.182737	0.229747	0.252144	0.252144

Tablica 6. Mjerenje R1, R2=(25°, 90°)

R2 / R1	90°					
		1	2	3	4	Idealno
25°	x	-0.059	-0.007	0.004	-0.028	0
	y	42.382	42.44	42.446	42.244	42.2618
	z	90.683	90.796	90.791	90.358	90.6308
	Δx	0.059	0.007	-0.004	0.028	0.059
	Δy	-0.1202	-0.1782	-0.1842	0.0178	0.1842
	Δz	-0.0522	-0.1652	-0.1602	0.2728	0.2728
	euler	0.143715	0.243095	0.244151	0.27481	0.27481

Tablica 7. Mjerenje R1, R2=(25°, 170°)

R2 / R1	170°					
		1	2	3	4	Idealno
25°	x	-89.326	-89.345	-89.328	-89.38	-89.2539
	y	42.423	42.483	42.432	42.476	42.2618
	z	15.365	15.352	15.402	15.341	15.7379
	Δx	0.0721	0.0911	0.0741	0.1261	0.1261
	Δy	-0.1612	-0.2212	-0.1702	-0.2142	0.2212
	Δz	0.3729	0.3859	0.3359	0.3969	0.3969
	euler	0.412599	0.454035	0.383781	0.468308	0.468308

Kao što je vidljivo iz mjerenja za konstantni R1=25° još jednom se ponavlja porast odstupanja s porastom kuta R2.

Tablica 8. Mjerenje R1, R2=(60°, 0°)

R2 / R1	0°					
		1	2	3	4	Idealno
60°	x	90.643	90.567	90.705	90.629	90.6308
	y	34.823	34.789	34.829	34.823	34.6188
	z	-24.005	-23.972	-24.019	-23.999	-24.2404
	Δx	-0.0122	0.0638	-0.0742	0.0018	0.0742
	Δy	-0.2042	-0.1702	-0.2102	-0.2042	0.2102
	Δz	-0.2354	-0.2684	-0.2214	-0.2414	0.2684
	euler	0.311865	0.324156	0.314178	0.316188	0.324156

Tablica 9. Mjerenje R1, R2=(60°, 90°)

R2 / R1	90°					
60°		1	2	3	4	Idealno
	x	-0.145	-0.172	-0.158	-0.132	0
	y	86.809	86.783	86.716	86.806	86.6025
	z	50.152	50.146	50.073	50.153	50
	Δx	0.145	0.172	0.158	0.132	0.172
	Δy	-0.2065	-0.1805	-0.1135	-0.2035	0.2065
	Δz	-0.152	-0.146	-0.073	-0.153	0.153
	euler	0.29457	0.288929	0.207787	0.286784	0.29457

Tablica 10. Mjerenje R1, R2=(60°, 170°)

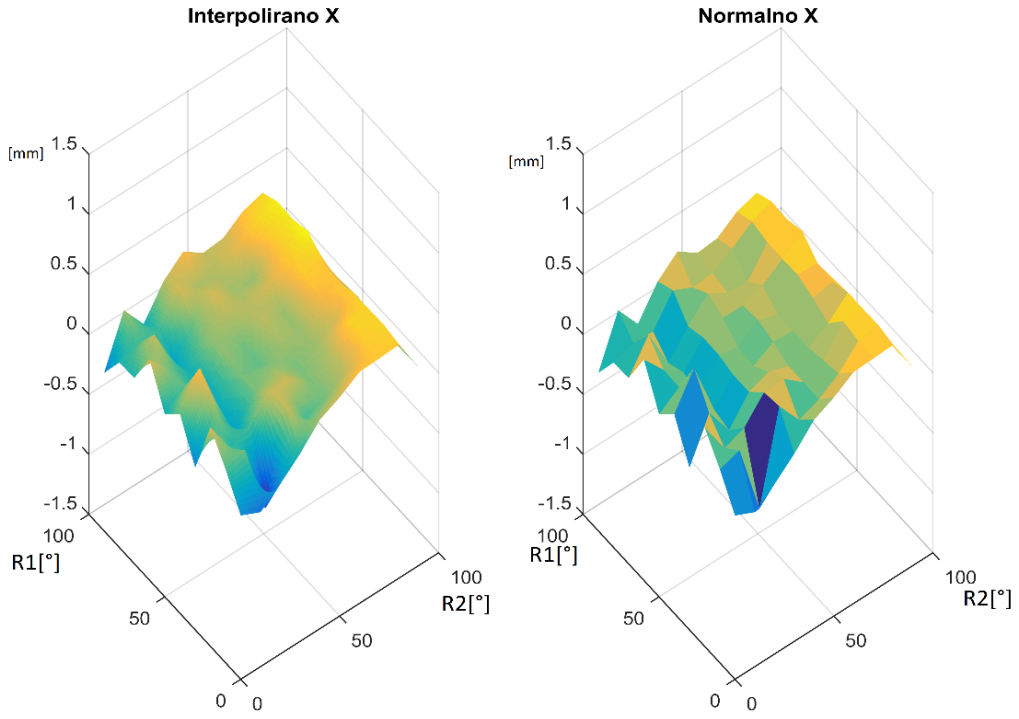
R2 / R1	170°					
60°		1	2	3	4	Idealno
	x	-89.484	-89.506	-89.51	-89.556	-89.2539
	y	43.445	43.447	43.465	43.477	43.2539
	z	-11.567	-11.572	-11.587	-11.571	-11.3487
	Δx	0.2301	0.2521	0.2561	0.3021	0.3021
	Δy	-0.1911	-0.1931	-0.2111	-0.2231	0.2231
	Δz	0.2183	0.2233	0.2383	0.2223	0.2383
	euler	0.370297	0.388207	0.40858	0.436412	0.436412

U nastavku je prikazano odstupanje uređaja po radnom području R1(0°-90°) i R2(0°-90°). Točke su mjerene u inkrementima po 10° svake osi. Prikaz raspodjele po svakoj osi zasebno te raspodjela prostornog odstupanja.

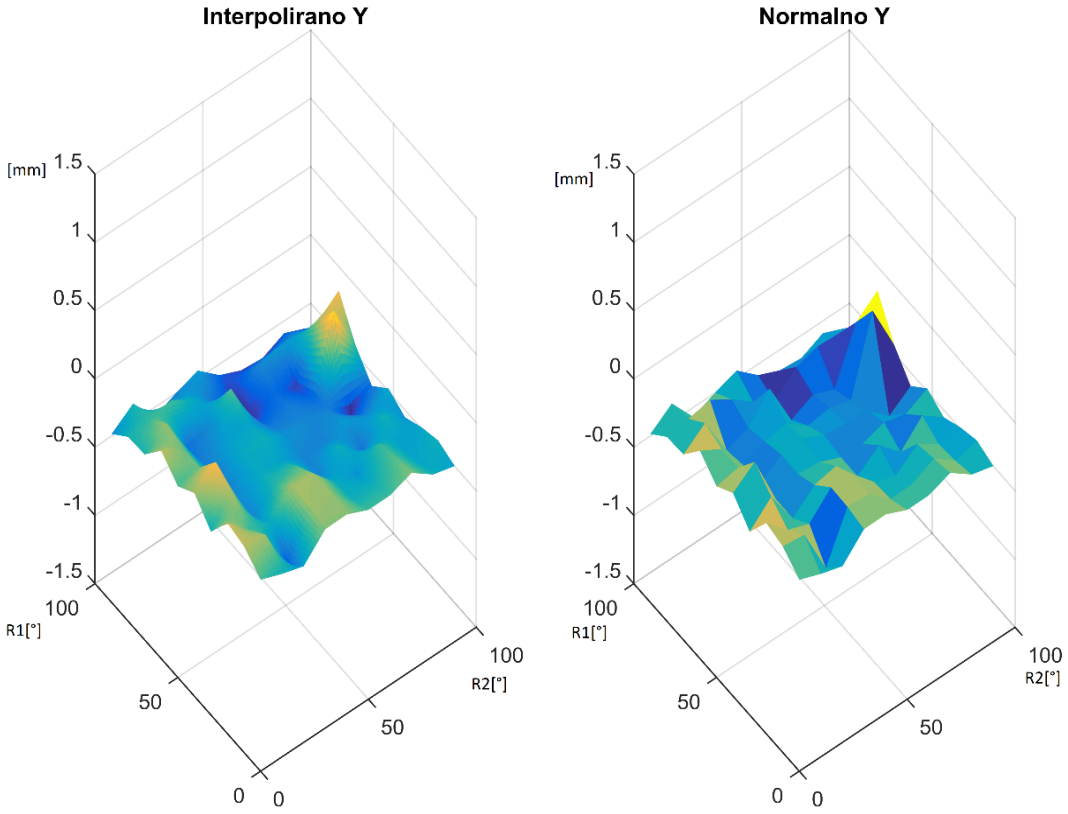
U grafovima radnog područja, vidimo povećanje odstupanja u smjeru osi R2. Odstupanje se najbolje vidi na grafu osi *x*, no prisutno je u svim smjerovima što se vidi iz grafa na Slika 44.

Odstupanje koje se javlja je rezultat više parametara unutar sustava. Uz pojavljivanje odstupanja kod mjerenja zbog raznih faktora, kao što su greške u mjerenju kod uzimanja točaka mjerne sfere, odstupanje se javlja i zbog manjka krutosti radnog modula. Jedan od glavnih mogućih razloga porasta odstupanja s kutom R2 je nakupljanje pogreške kroz gibanje radnog modula po luku.

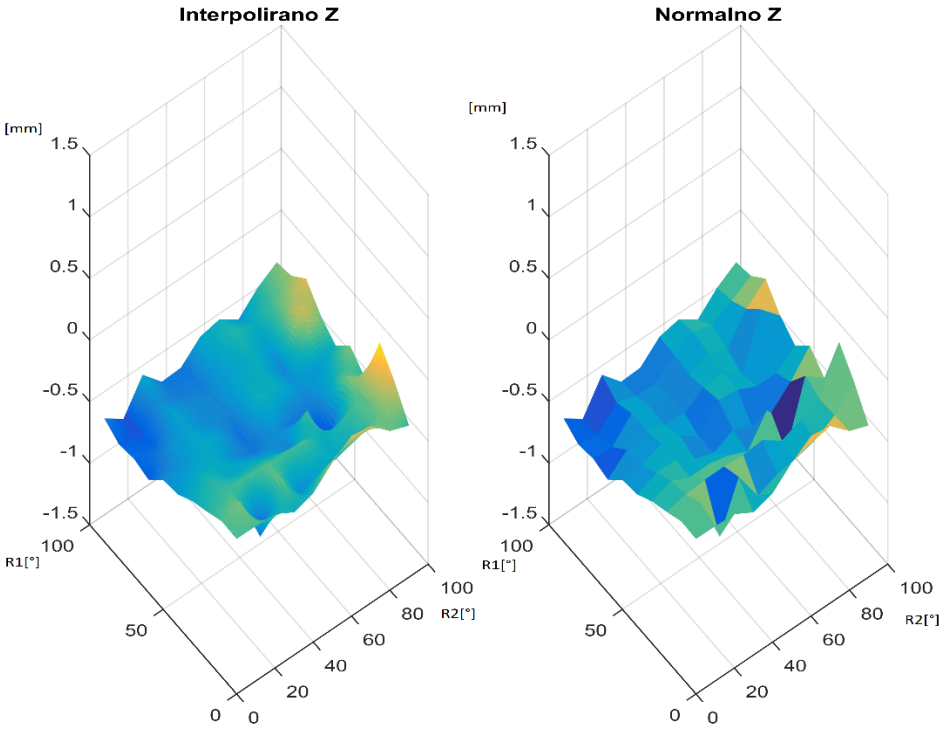
Sva prikazana odstupanja prikazana grafovima na slikamaSlika 41,Slika 422,Slika 433 i Slika 444 su u milimetrima i prikazuju odstupanje mjerene vrijednosti osi točaka od teorijskih.



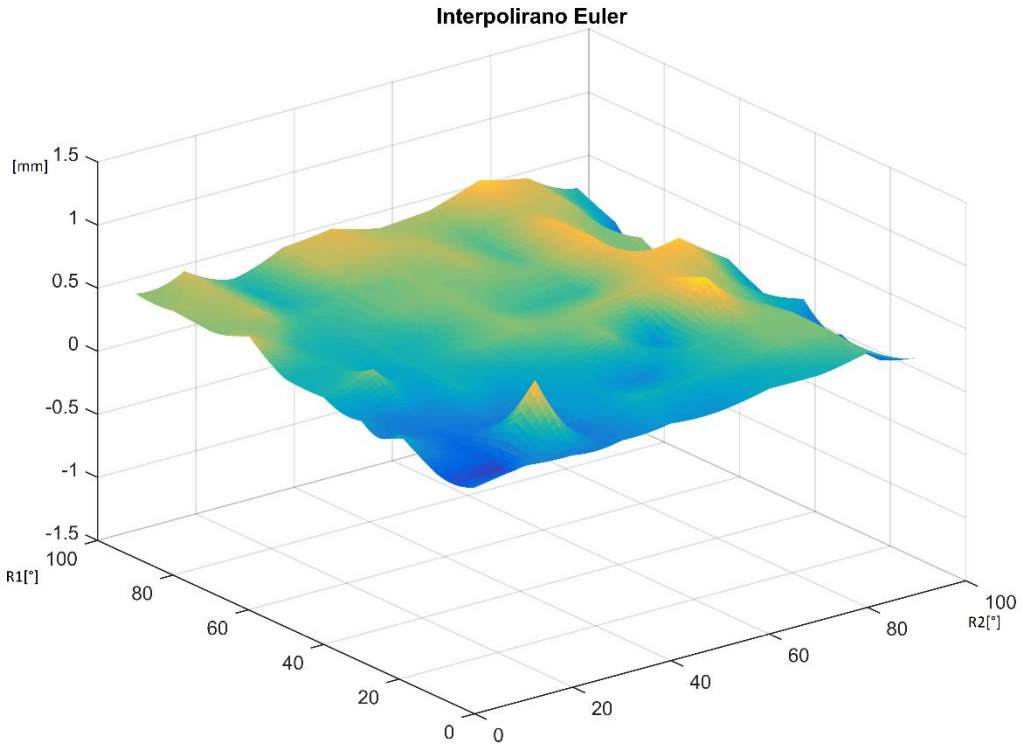
Slika 41. Odstupanja vrijednosti po osi X



Slika 42. Odstupanja vrijednosti po osi Y



Slika 43. Odstupanja vrijednosti po osi Z



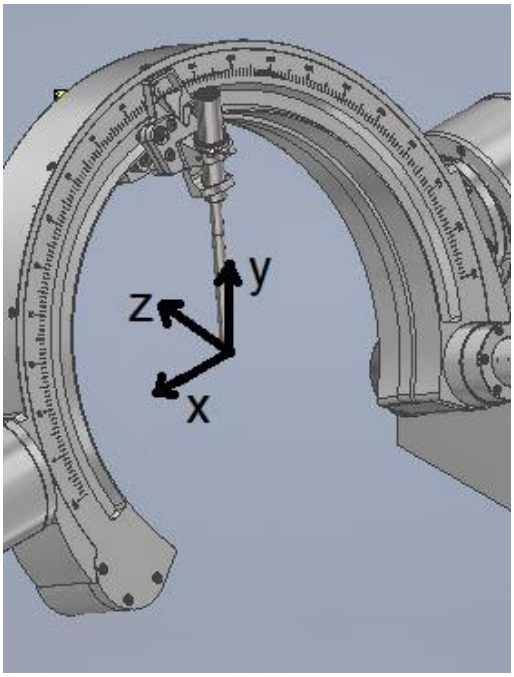
Slika 44. Ukupna vrijednosti odstupanja

4.1.3 Kompenzacija greške

Pogreške u pozicioniranju nastale zbog nesavršenosti konstrukcijske izvedbe se mogu kompenzirati. U ovom odlomku će biti objašnjeno po kojem principu će se sama kompenzacija vršiti. Prvi korak je transformacija koordinatnog sustava glave pacijenta u koordinatni sustav robota te preračunavanje ulazne točke iz sfernog koordinatnog sustava (*r*, R1, R2) u kartezijev (*x*, *y*, *z*).

Ulazni parametri uređaja potrebni za ovaj izračun su:

- *r*, radijus vrha testne igle za mjerenje ulazne točke
- Kut α, kut odmaka testne igle od R1 kuta luka
- R1 i R2, proizvoljne kutove pripadnih osi.



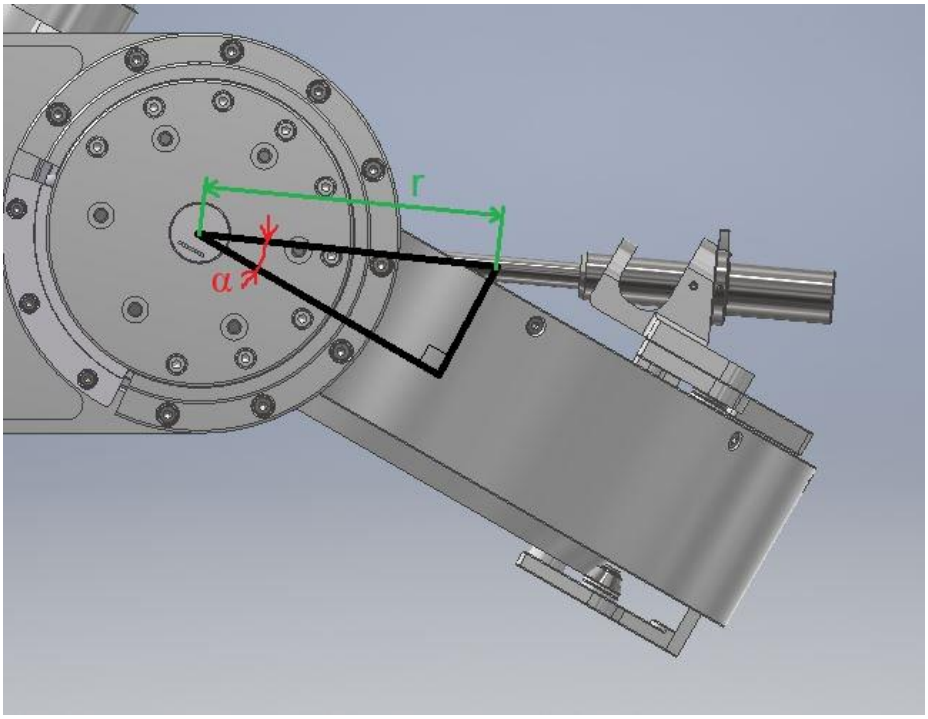
Slika 45. Kartezijev koordinatni sustav glave pacijenta

Da bi smo dobili vrijednosti x_t, y_t i z_t ovakvog sustava koristimo sljedeći izraz:

$$rot(x)\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \quad (1)$$

Drugi član izraza je korekcija kuta α i utjecaj rotacije R2. Iznos kuta α je 25° što je definirano samom konstrukcijom luka. To je vektor unutar kojeg se nalaze među vrijednosti

točaka u kartezijevom koordinatnom sustavu. Vrijednosti na koje je još potrebno izvesti operaciju rotacije oko osi *x*. Ovaj vektor je ujedino i rezultat konverzije ulaznih sfernih koordinatnih veličina u ulazne veličine prikazane u kartezijevom koordinatnom sustavu.



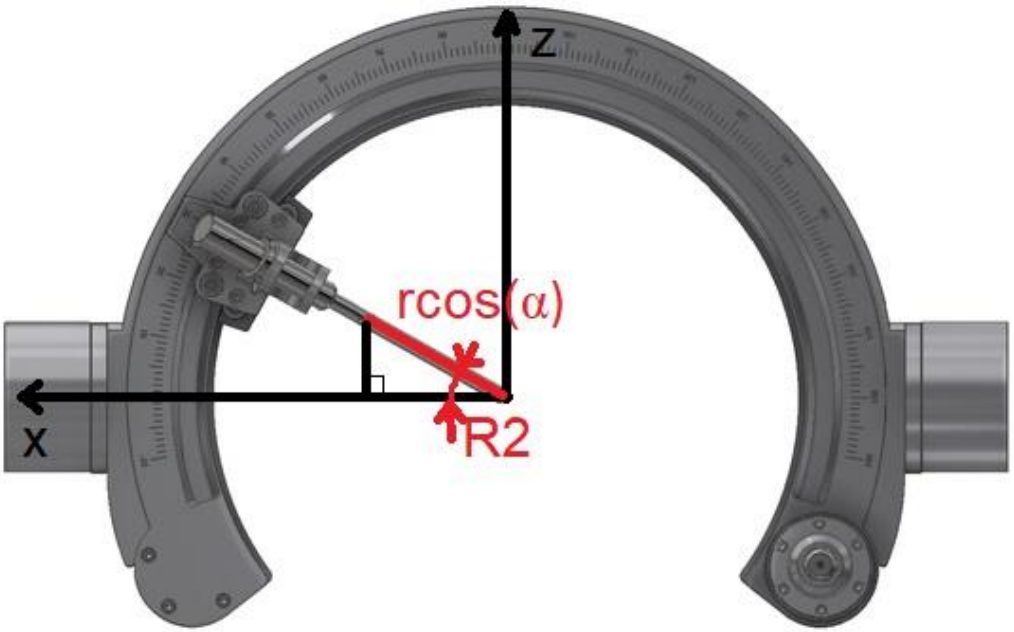
Slika 46. Skica odstupanja radne igle od ravnine luka

Iz gore prikazane skice možemo izračunati dvije vrijednosti potrebne za kreiranje vektora korekcije koordinatnih točaka x_k, y_k i z_k. Prvi izraz je izračun duljine koja je projekcija radijusa *r* na ravninu samog luka:

$$r_p = r \cdot cos(\alpha) \tag{2}$$

A drugi je visina ulazne točke od plohe luka, koji je kao takav i rješenje za drugi član vektora rotacije R2:

$$y_k = r \cdot sin(\alpha) \tag{3}$$



Slika 47. Skica izračuna z_k i x_k komponenti vektora rotacije R2 osi

Kada u pogled s Slika 47 uvrstimo izraz (2) dobivamo sljedeća rješenja:

$$x_k = r \cdot cos(\alpha) \cdot \cos(\mathrm{R2}) \tag{4}$$

$$z_k = r \cdot cos(\alpha) \cdot \sin(\mathrm{R2}) \tag{5}$$

Uvrštavanjem jednadžbi (3), (4) i (5) u drugi član izraza (1) dobivamo:

$$\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} r \cdot cos(\alpha) \cdot \cos(\mathrm{R2}) \\ r \cdot sin(\alpha) \\ r \cdot cos(\alpha) \cdot \sin(\mathrm{R2}) \end{bmatrix} \tag{6}$$

Matrica rotacije oko osi x ima modificirane predznake ispred sinusnih članova zbog suprotnog smjera pozitivnih vrijednosti osi *x* u definirano koordinatnom sustavu. Iz istog razloga se i rotacija oko *x*, koristi kao negativna vrijednost. Rotaciju oko *x* prikazujemo kao izraz:

$$rot(x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-R1) & -\sin(-R1) \\ 0 & \sin(-R1) & \cos(-R1) \end{bmatrix} \tag{7}$$

Uvrštavanjem ova dva rješenja, (6) i (7), u početnu jednadžbu (1) dobivamo krajnji izraz za korekciju i transformaciju naših kutova iz zadanih globalnih, u lokalne uređaja

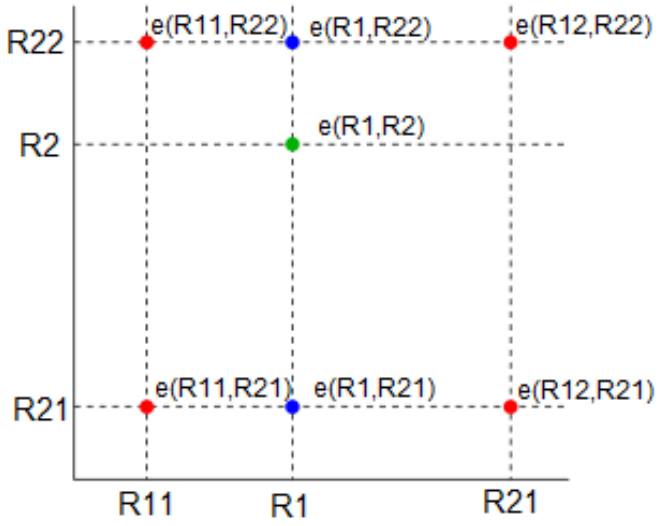
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-R1) & -\sin(-R1) \\ 0 & \sin(-R1) & \cos(-R1) \end{bmatrix} \begin{bmatrix} r \cdot cos(\alpha) \cdot \cos(\mathrm{R2}) \\ r \cdot sin(\alpha) \\ r \cdot cos(\alpha) \cdot \sin(\mathrm{R2}) \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}. \quad (8)$$

Sljedeći korak je mjerenje stvarnih vrijednosti $\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$ za pojedine kutove te kreiranje baze mjerenih vrijednosti. Te usporedba s teoretskim vrijednostima za pojedine kutove. Mjerenje je odrađeno kao skup točaka u razmaku od 10°, tako kreiranu matricu onda koristimo za izračun odstupanja realne pozicije od teorijske.

Da bi to mogli potrebno je izračunati teorijske vrijednosti kutova za koje imamo mjerene vrijednosti, a to je učinjeno preko izraza za konverziju sfernog koordinatnog sustava u kartezijev (7). Tada od tih vrijednosti oduzimamo mjerene vrijednosti i dobivamo matricu odstupanja za mjereni uređaj.

$$\begin{bmatrix} x_t(R1_i, R2_i) \\ y_t(R1_i, R2_i) \\ z_t(R1_i, R2_i) \end{bmatrix} - \begin{bmatrix} x_m(R1_i, R2_i) \\ y_m(R1_i, R2_i) \\ z_m(R1_i, R2_i) \end{bmatrix} = \begin{bmatrix} e_x(R1_i, R2_i) \\ e_y(R1_i, R2_i) \\ e_z(R1_i, R2_i) \end{bmatrix} \quad (9)$$

Dobivena matrica *e* u sebi sadrži sva odstupanja od teoretskih vrijednosti za koje postoji mjerenje. No, ograničenje ovog modela je to što postoje informacije o odstupanju samo za konačan broj traženih pozicija tj. onoliko koliko je mjerenja napravljeno. Zbog ovog je potrebno na matricu odstupanja E izvršiti bilinearnu interpolaciju vrijednosti između četiri susjedne točke od traženog kuta.



Slika 48. Bilinearna interpolacija

Bilinearna interpolacije je dvodimenzionalna interpolacija koja se sastoji od dva koraka. Prvi korak je interpolacija jedne dimenzije u prvoj vrijednosti druge (11) i interpolacija te iste dimenzije, ali u drugoj vrijednosti druge dimenzije (12). Drugi korak je interpolacija ta dva rješenja po drugoj dimenziji (10). Grafički prikaz ove vrste interpolacije je na Slika 48. Interpolaciju vršimo za svaku os zasebno te je izraz matrično prikazan finalnom jednadžbom:

$$\begin{bmatrix} e_x(R1,R2) \\ e_y(R1,R2) \\ e_z(R1,R2) \end{bmatrix} = \begin{bmatrix} \frac{R22-R2}{R22-R21} e_x(R1,R21) + \frac{R2-R21}{R22-R21} e_x(R1,R22) \\ \frac{R22-R2}{R22-R21} e_y(R1,R21) + \frac{R2-R21}{R22-R21} e_y(R1,R22) \\ \frac{R22-R2}{R22-R21} e_z(R1,R21) + \frac{R2-R21}{R22-R21} e_z(R1,R22) \end{bmatrix} \quad (10)$$

Među vrijednosti *e*(R1,R21) i *e*(R1,R22) su zasebne interpolacije po jednoj dimenziji ovakve dvodimenzionalne interpolacije te se računaju prema sljedećim izrazima:

$$\begin{bmatrix} e_x(R1,R21) \\ e_y(R1,R21) \\ e_z(R1,R21) \end{bmatrix} = \begin{bmatrix} \frac{R12-R1}{R12-R11} e_x(R11,R21) + \frac{R1-R11}{R12-R11} e_x(R12,R21) \\ \frac{R12-R1}{R12-R11} e_y(R11,R21) + \frac{R1-R11}{R12-R11} e_y(R12,R21) \\ \frac{R12-R1}{R12-R11} e_z(R11,R21) + \frac{R1-R11}{R12-R11} e_z(R12,R21) \end{bmatrix} \quad (11)$$

$$\begin{bmatrix} e_x(R1,R22) \\ e_y(R1,R22) \\ e_z(R1,R22) \end{bmatrix} = \begin{bmatrix} \frac{R12-R1}{R12-R11} e_x(R11,R22) + \frac{R1-R11}{R12-R11} e_x(R12,R22) \\ \frac{R12-R1}{R12-R11} e_y(R11,R22) + \frac{R1-R11}{R12-R11} e_y(R12,R22) \\ \frac{R12-R1}{R12-R11} e_z(R11,R22) + \frac{R1-R11}{R12-R11} e_z(R12,R22) \end{bmatrix} \quad (12)$$

U ovom trenutku znamo vrijednost odstupanja mjerene, stvarne, od teorijske te pomoću tog odstupanja možemo kompenzirati nastalu grešku. Kompenzacija se izvršava tako da za zadane kutove R1 i R2 izračunamo teorijsku vrijednost u kartezijevim koordinatama te njima dodajemo vrijednost odstupanja te tako dobivamo nove vrijednost u koje manipulator mora doći da bi se u realnosti našao što bliže teorijski izračunatoj vrijednosti.

$$\begin{bmatrix} x_{komp} \\ y_{komp} \\ z_{komp} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_x(R1,R2) \\ e_y(R1,R2) \\ e_z(R1,R2) \end{bmatrix} \quad (13)$$

Kompenzirane vrijednosti su sada nove vrijednosti u kartezijevom koordinatnom sustavu u koje manipulator mora doći da bi se u stvarnosti pozicionirao u teorijsku točku za tražene vrijednosti R1 i R2. Iz razloga što su upravljive vrijednosti manipulatora kutovi R1 i R2 u stupnjevima potrebno je izvršiti transformaciju nazad u sferni koordinatni sustav. Ovaj korak se izvršava preko formula za transformaciju:

$$R1 = atan2\left(-z_{kompenzirano}, y_{kompenzirano}\right)$$
$$= \begin{cases} \arctan\left(\frac{-z_{kompenzirano}}{y_{kompenzirano}}\right), & za\ y > 0, \\ \arctan\left(\frac{-z_{kompenzirano}}{y_{kompenzirano}}\right) + \pi\,, & za\ y < 0\ i - z \geq 0, \\ \arctan\left(\frac{-z_{kompenzirano}}{y_{kompenzirano}}\right) - \pi, & za\ y < 0\ i - z < 0, \\ +\frac{\pi}{2}, & za\ y = 0\ i - z > 0, \\ -\frac{\pi}{2}, & za\ y = 0\ i - z < 0, \\ nedefinirano, & za\ y = 0\ i - z = 0. \end{cases} \quad (14)$$

$$R2 = \cos^{-1}\left(\frac{x_{kompenzirano}}{r \cdot cos(\alpha)}\right) \quad (15)$$

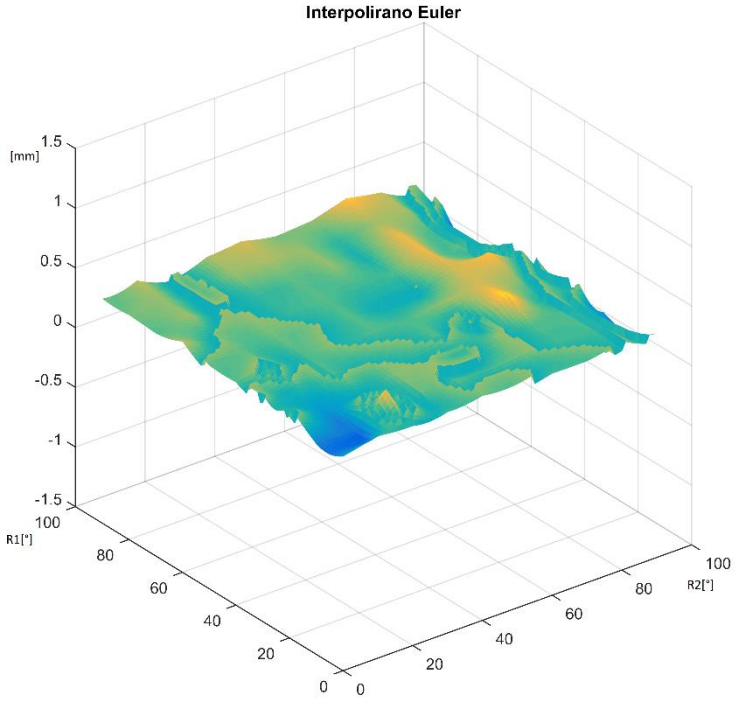
Posljednji korak je validacija i to na način da se ponovno rade mjerenja točaka te usporedba istih prije i nakon kompenzacije. [11]

Tablica 11. Prikaz mjerenja prije i nakon kompenzacije

R1 \ R2	170					
0		1	2	3	4	Idealno
	x	-89.404	-89.196	-89.268	-89.407	-89.2539
	y	32.16	32.066	32.191	32.212	31.6511
	z	31.784	31.703	31.685	31.791	32.124
	Δx	0.1501	-0.0579	0.0141	0.1531	0.1531
	Δy	-0.5089	-0.4149	-0.5399	-0.5609	0.5609
	Δz	0.34	0.421	0.439	0.333	0.439
	euler	0.630166	0.593915	0.695997	0.670028	0.695997
		1	2	3	4	Idealno
	x	-89.18	-88.992	-89.325	-89.262	-89.2539
	y	31.889	31.813	31.926	31.927	31.6511
	z	32.119	32.022	32.147	32.114	32.124
	Δx	-0.0739	-0.2619	0.0711	0.0081	0.2619
	Δy	-0.2379	-0.1619	-0.2749	-0.2759	0.2759
	Δz	0.005	0.102	-0.023	0.01	0.102
	euler	0.249164	0.324357	0.284876	0.2762	0.324357

Tablica 12. Prikaz mjerenja prije i nakon kompenzacije

R2 / R1	170					
25°		1	2	3	4	Idealno
	x	-89.326	-89.345	-89.328	-89.38	-89.2539
	y	42.423	42.483	42.432	42.476	42.2618
	z	15.365	15.352	15.402	15.341	15.7379
	Δx	0.0721	0.0911	0.0741	0.1261	0.1261
	Δy	-0.1612	-0.2212	-0.1702	-0.2142	0.2212
	Δz	0.3729	0.3859	0.3359	0.3969	0.3969
	euler	0.412599	0.454035	0.383781	0.468308	0.468308
		1	2	3	4	Idealno
	x	-89.391	-89.498	-89.561	-89.472	-89.2539
	y	42.36	42.456	42.483	42.465	42.2618
	z	15.658	15.66	15.696	15.675	15.7379
	Δx	0.1371	0.2441	0.3071	0.2181	0.3071
	Δy	-0.0982	-0.1942	-0.2212	-0.2032	0.2212
	Δz	0.0799	0.0779	0.0419	0.0629	0.0799
	euler	0.186611	0.321507	0.380783	0.304654	0.380783

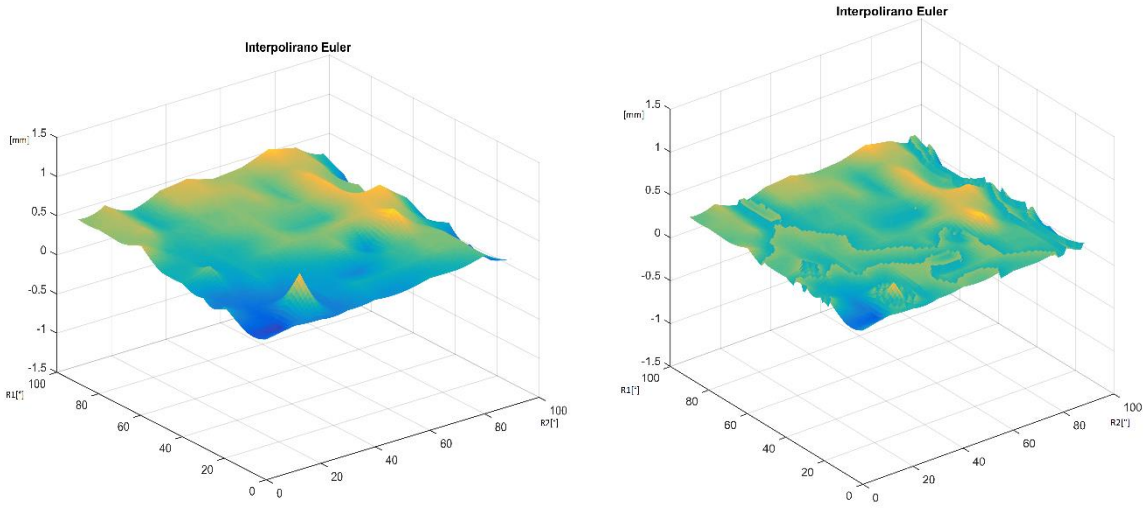


Slika 49. Ukupno prostorno odstupanje nakon korekcije poziciji

5. Zaključak

Nakon primjene kompenzacije greška pozicioniranja se smanjila, ali nije u potpunosti nestala, kao što je vidljivo iz tablica u prethodnom poglavlju. Poboljšanja preciznosti su vidljiva iz crvenih polja. Za kut R1=0°, R2=170° odstupanje od teorijeske vrijednosti je bilo 0.696 mm dok je nakon repozicioniranja s korekcijom izmjerena točka odstupala za 0.324 mm. Korekcija je uspjela eliminirati odstupanje nastalo povećanjem kuta R2 koje smo primjetili u inicijalnim mjerenjima. No, određena količina odstupanja je ipak prisutna. To je vidljivo iz drugog mjerenja, gdje je inicijalno odstupanje bilo nešto manje, 0.468 mm. Korekcija je smanjila odstupanje na 0.381 mm.

Ako se pogleda preostala mjerenja može se vidjeti da se prostorno odstupanje nije smanjivalo ispod ±~0.38 mm što je odstupanje preostalo nakon eliminacije skupljanja pogreške gibanjem radnog modula po luku, odnosno R2 osi. Isto tako, korigirana je i ishodišna točka, tako da je početno odstupanje svedeno na minimum. Preostala odstupanja nastaju dijelom iz ograničenja mjerne ruke, a dijelom iz nepravilnosti izrade sferne površine preko koje je vršeno mjerenje. Zadnji i najveći problem je u prijenosu sile unutar radnog modula. Sila gibanja koja dolazi s remena bi trebala pomicati radni modul i na taj način krutom vezom pomicati vrh modula. To se ne događa u potpunosti jer se na opružnom elementu za stezanje modula javlja elastično ponašanje opruge te se gubi krutost. Iz ovog razloga imamo odstupanje koje nastaje uslijed različitog naprezanja ove dvije opruge. Naprezanje opruga, a samim time i odstupanje, ovisi o inicijalnom smjeru gibanja radnog modula. Na taj način se javlja „prozor“ pozicija unutar kojih se sama ciljna točka može nalaziti ovisno o smjeru u kojem se giba radni modul. Ovakvo odstupanje je loše iz razloga što tokom rada uređaj mora moći neometano mjenjati smjerove gibanja radne osi R2.



Slika 50. Prikaz prostornog odstupanja prije i nakon korekcije pozicioniranja

Uklanjanje ovog odstupanja bi se moglo izvesti poboljšanjem ili promjenom mehanizma osiguravanja stezne sile radnog modula na način da se omogući kruto zaključavanje ovog mehanizma. Jednostavniji prijedlog bi bio mogućnost dodavanja vijaka koji bi bili u mogućnosti zaključati međusobne pozicije dvaju dijelova radnog modula te na taj način ukrutiti vezu između ta dva dijela. Alternativno rješenje također može biti potpuni redizajn radnog modula u izvedbu iz jednog komada. Čime bi se u potpunosti ukrutila veza između igle, ulazne točke, i pričvrsnog mjesta remena i enkodera.

6. Literatura

[1] F. e. al., Review of Robotic Technology for Stereotactic Neur, 2015.

[2] Haidegger, Improving the Accuracy and Safety of a Robotic Systems, 2008.

[3] H. H. M. P. R. S. a. T. Deacon, The Pathfinder image-guided surgical robot.

[4] H. e. al., Calibration of the motor-assisted robotic stereotaxy system:MARS, 2012.

[5] »Maxon Motors catalog,« Maxon, [Mrežno]. Available: http://maxon.blaetterkatalog.ch/b9990/catalog/index.html?data=b9990/b999045&lang=e#6.

[6] ELMO, P MAESTRO IG.

[7] ELMO, ELMO_MAN-G-TWIIG.

[8] »Renishaw enkoderi,« [Mrežno]. Available: https://www.renishaw.com/en/resolute-encoder-series--37823.

[9] »RLS enkoderi,« [Mrežno]. Available: https://www.rls.si/en/products/rolin-rotary-incremental-magnetic-encoder-system.

[10] faro, FARO Quantum S Tech Sheet.

[11] J. e. al., Renaissance robotic system for keyhole cranial neurosurgery, 2011.

[12] L. e. al., The Impact of the Reference Imaging Modality, 2014.

[13] S. a. Ernst, Medical Robotics, 2015.

7. Prilog

1 JavaFx GUI 58
1.1 Main.java 58
1.2 Controler_prozor1.java 62
1.3 Xform.java 94
1.4 Unos_IP.java 97
1.5 FxmlFiles.java 98
1.6 Komunikacija 99
1.7 Texts.java 100
1.8 Controler_unos.java 104
1.9 Menu.java 106
1.10 Slike.java 108
2 PMAS 109
2.1 Main.cpp 109
2.2 Main.h 117
2.3 CallBack.cpp 118
2.4 CallBack.h 121
3 Euler_interpol.m 122

1 JavaFx GUI

1.1 Main.java

```
package nerov0.pkg1;

import java.io.File;
import java.io.IOException;
import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Application;
import javafx.application.Platform;
import static javafx.application.Platform.exit;
import javafx.beans.binding.Bindings;
import javafx.beans.property.DoubleProperty;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.media.Media;
import javafx.scene.media.MediaView;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

/**
*
* @author Filipovic novi
* Klasa MAIN
*/
public class Main_neroV1 extends Application{
    public static Stage  load_stage;
    final Xform world = new Xform();
    private AnchorPane rootPrviProzor, loadPrviProzor;
    public FXMLLoader prviProzor=null, loadProzor=null;
    final menu menuIPTest= new menu();
    public static Controler_prozor1 mainProzor;
    Alert aPotvrda= new Alert(AlertType.CONFIRMATION);
    Stage error=(Stage) aPotvrda.getDialogPane().getScene().getWindow();



    @Override
    public void start(Stage nultiStage) throws IOException, InterruptedException{
        Main_neroV1.load_stage=nultiStage;
        loading();
    }

    private void handleKeyboardProzor(Scene scene, final Node root) {
        scene.setOnKeyPressed((KeyEvent event) -> {
            switch (event.getCode()) {
                case C:
                    error.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco16)));
                    aPotvrda.setTitle(Texts.sConTitle);
                    ImageView
iIkona=new ImageView(new Image(getClass().getResourceAsStream(Slike.sIco64_1)));
                    aPotvrda.setGraphic(iIkona);
                    aPotvrda.setHeaderText(Texts.sConHead);
                    aPotvrda.setContentText(Texts.sConCont);
                    aPotvrda.showAndWait().ifPresent((ButtonType rs)->{
                        if(rs==ButtonType.OK){
                            try {
```

```
                    Controler_prozor1.bServer=false;
                    Controler_prozor1.sockPrimanje.close();
                    exit();
                } catch (IOException ex) {
                    Logger.getLogger(Main_neroV1.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            /*if(rs==ButtonType.CANCEL){

            }*/
        });

            break;
        case I:
            menuIPTest.menuIP2();
            break;
        case H:
            menuIPTest.menuAbout();
            break;
        case L:
            Texts.jezik();
            break;
        case A:
            Controler_prozor1.vSwap();
            break;
        case E:
            Controler_prozor1.analiza();
            break;
      }
    });
 }

 private void menuIP(){
      Parent root;
   try{
      FXMLLoader treciProzor = new FXMLLoader();
      treciProzor.setLocation(Controler_prozor1.class.getResource(FxmlFiles.sUnIP));
      AnchorPane rootTreciProzor = (AnchorPane) treciProzor.load();
      Scene scena3 = new Scene(rootTreciProzor);
      Stage unos_stage3 = new Stage();
      unos_stage3.setScene(scena3);
      unos_stage3.setTitle(Texts.sMainTitle+Texts.sAddTitleIP);
      unos_stage3.setResizable(false);
      unos_stage3.setScene(scena3);
      unos_stage3.initStyle(StageStyle.UTILITY);
      unos_stage3.show();
    }catch(IOException e){
    }
 }
 public void loading() throws IOException, InterruptedException{
    File filmic= new File("src\\nerov0\\pkg1\\NERO.mp4");
```

```
        Media med=new Media(filmic.toURI().toURL().toString());
        javafx.scene.media.MediaPlayer player = new   javafx.scene.media.MediaPlayer(med);
        MediaView viewer=new MediaView(player);
        DoubleProperty width = viewer.fitWidthProperty();
        DoubleProperty height = viewer.fitHeightProperty();
        width.bind(Bindings.selectDouble(viewer.sceneProperty(), "width"));
        height.bind(Bindings.selectDouble(viewer.sceneProperty(), "height"));
        viewer.setPreserveRatio(true);

        loadProzor=new FXMLLoader();
        loadProzor.setLocation(Main_neroV1.class.getResource(FxmlFiles.sLoad));
        loadPrviProzor=(AnchorPane) loadProzor.load();
        loadPrviProzor.getChildren().add(viewer);
        Scene scenaLoad=new Scene(loadPrviProzor,1067,600);
        load_stage.setScene(scenaLoad);
        load_stage.initStyle(StageStyle.UNDECORATED);
        load_stage.show();
        player.play();
        player.setOnEndOfMedia(new Runnable(){
            @Override
            public void run() {
                glavni_prozor();
                load_stage.hide();
            }
        });
    }
    public void glavni_prozor(){
        Parent root;
        try{
            prviProzor=new FXMLLoader();
            prviProzor.setLocation(Main_neroV1.class.getResource(FxmlFiles.sMain));
            rootPrviProzor=(AnchorPane) prviProzor.load();
            Scene scena1=new Scene(rootPrviProzor);
            handleKeyboardProzor(scena1,world);
            Stage init_stage = new Stage();
            init_stage.setTitle(Texts.sMainTitle);
            init_stage.setResizable(false);
            init_stage.setScene(scena1);
            init_stage.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco16)));
            init_stage.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco32)));
            init_stage.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco64)));
            init_stage.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco128)));
            init_stage.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco256)));
            init_stage.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco512)));
            //init_stage.initStyle(StageStyle.UTILITY);
            init_stage.show();
        }
        catch(IOException e){
        }
    }
    public static void main(String[] args) {
```

```
        launch(args);
    }

}
```

1.2 Controler_prozor1.java

```
package nerov0.pkg1;

import com.mathworks.engine.EngineException;
import com.mathworks.engine.MatlabEngine;
import com.mathworks.engine.MatlabSyntaxException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import static java.lang.Math.acos;
import static java.lang.Math.cos;
import static java.lang.Math.sin;
import java.net.ServerSocket;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.Locale;
import static javafx.application.Platform.exit;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.MenuItem;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import java.net.Socket;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import java.util.concurrent.CancellationException;
import java.util.concurrent.ExecutionException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.animation.AnimationTimer;
```

```
import javafx.application.Platform;
import javafx.beans.property.LongProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.fxml.Initializable;
import javafx.scene.DepthTest;
import javafx.scene.Group;
import javafx.scene.Node;
import javafx.scene.PerspectiveCamera;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Menu;
import javafx.scene.control.ProgressBar;
import javafx.scene.control.Separator;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.shape.Box;
import javafx.scene.shape.Cylinder;
import javafx.scene.shape.Sphere;
import javafx.scene.transform.Rotate;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;


public class Controler_prozor1 implements Initializable {
    double errx,erry,errz;
    static String[] linesMj=new String[256];
    static String[][] linesMj2=new String[256][5];
    static String[][] linesError2=new String[256][5];
    static String[]  linesMj1=new String[256];
    static String[][] linesMj3=new String[256][5];
    double dInter[]=new double[3];
    public int iCounter=0, jCounter=0;
    static double x_R1, x_R2;
    static double x_1, x_2;
    static float i;
    static double progress;
    public String x_s, x_ms, y_ms, z_ms, x_sdox, x_sdox2, r1_ntar, r2_ntar, sPoruka, sPorukaOs
= null, sPorukaVrijednost, sPorukaVrijednost2, sPorukaVrijednost3,
sPorukaVrijednost4,sPorukaVrijednost5,sPorukaVrijednost6;
    String kod;
    public String sErr1, sErr2;
    String slika1_21, slika1_2, slika2_22, slika2_21;
    String x_R1_string, x_R2_string;
    public double x_tar, y_tar, z_tar, iz, ntar, er, n_tar_x, n_tar_y, n_tar_z, pi = 3.141592653589793,
dMedjuPrimanje1, dMedjuPrimanje2, dMedjuPrimanje3, dMedjuPrimanje4;
    public static double r_1novi, r_2novi;
    public double x_meas = 0, y_meas = 0, z_meas = 0,x_meas1 = 0, y_meas1 = 0, z_meas1 = 0,
R1_meas1=0, R2_meas1=0, R1_meas=0, R2_meas=0;
```

```
    public double dErr1 = 0, dErr2 = 0;
    int iBroj = 0;
    public static ServerSocket sockPrimanje = null;
    Socket s1 = null;
    Socket s = null;
    byte[] stringPrimljenUBytovima = null;
    byte[] stringuBitovima = null;
    InputStreamReader inXSR = null;
    BufferedReader disPrimanje = null;
    static boolean anal1=true, anal=true, proracun=true,proracun1=true;
    JSONObject jsonPoruka=new JSONObject();

    //BufferedReader disPrimanje=null;
    /**
     * broj imena slika za animaciju R1, R2
     */
    public int slika1, slika2;
    /**
     * markeri 0 ili 1 za enable disable buttona
     */
    public boolean btn_disable = false, btn2_disable = false, btn_skrivanje = true;
    public static boolean bServer = true;
    /**
     * format decimalnog zapisa broja
     */
    public DecimalFormat df
= new DecimalFormat("0.0000", DecimalFormatSymbols.getInstance(Locale.ENGLISH)), df1
= new DecimalFormat("0", DecimalFormatSymbols.getInstance(Locale.ENGLISH));
    /**
     * slika crveni gumb(dis) i zeleni gumb(ena)
     */
    public Image dis = new Image(getClass().getResourceAsStream(Slike.sRed)), ena
= new Image(getClass().getResourceAsStream(Slike.sGrn));
    /**
     * deklaracija za animaciju r1 i r2
     */
    public Image img = null, img2 = null;
    /**
     * alociranje memorije za loader
     */
    public FXMLLoader drugiProzor = null;
    /**
     * alociranje prostora za novu scenu
     */
    public Scene scena2 = null, scena3 = null, scena4 = null;
    private Stage unos_stage, unos_stage2, unos_stage3, unos_stage4, stage5_3d;
    Stage stageTitle = null;
    private AnchorPane rootDrugiProzor, rootTreciProzor, rootCetvrtiProzor;
    /**
     * alociranje memorije za loader treceg prozora
     */
```

```
public FXMLLoader treciProzor = null;
public FXMLLoader cetvrtiProzor = null;
/**
 * alociranje memorije za trecu scenu
 */
public static String unos_IP;

final menu menuIPG = new menu();
final Group root1 = new Group();
final Xform axisGroup = new Xform();
final Xform world = new Xform();
final Xform Group3D = new Xform();
final PerspectiveCamera camera = new PerspectiveCamera(true);
final Xform cameraXform = new Xform();
final Xform cameraXform2 = new Xform();
final Xform cameraXform3 = new Xform();
private static final double CAMERA_INITIAL_DISTANCE = -450;
private static final double CAMERA_INITIAL_X_ANGLE = 20;
private static final double CAMERA_INITIAL_Y_ANGLE = 200;
private static final double CAMERA_INITIAL_Z_ANGLE = -90;
private static final double CAMERA_NEAR_CLIP = 0.1;
private static final double CAMERA_FAR_CLIP = 10000.0;
private static final double AXIS_LENGTH = 250.0;
private static final double HYDROGEN_ANGLE = 104.5;
private static final double CONTROL_MULTIPLIER = 0.1;
private static final double SHIFT_MULTIPLIER = 10.0;
private static final double MOUSE_SPEED = 0.5;
private static final double ROTATION_SPEED = 2.0;
private static final double TRACK_SPEED = 0.3;
private static final double MODEL_SCALE_FACTOR = 400;
private static final double MODEL_X_OFFSET = 0; // standard
private static final double MODEL_Y_OFFSET = 0; // standard
private static final int VIEWPORT_SIZE = 800;
private static final Color lightColor = Color.rgb(244, 255, 250);
private static final Color jewelColor = Color.rgb(0, 190, 222);
public PhongMaterial redMaterial = new PhongMaterial();
public PhongMaterial whiteMaterial = new PhongMaterial();
public PhongMaterial greyMaterial = new PhongMaterial();
public Xform XForm3D = new Xform();
public Xform hydrogen3SideXform = new Xform();
public Xform hydrogen3Xform = new Xform();
Sphere hydrogen3Sphere = new Sphere(30.0);
Cylinder bond3Cylinder = new Cylinder(100, 10);

double mousePosX;
double mousePosY;
double mouseOldX;
double mouseOldY;
double mouseDeltaX;
double mouseDeltaY;
private int iTest;
```

```
    String sTest;

    @FXML
    private MenuItem close_btn, menuIP, item_anal, menuAbout, menu_Lang,
menu_3d;//menu_file,menu_edit,menu_help;
    @FXML
    private Menu menu_file, menu_edit, menu_help;
    @FXML
    private Label lbl_R1, lbl_R1_pos, lbl_R1_stat, lbl_count_R1, lbl_err_R1, lbl_curR1;
    @FXML
    public  Button btn_R1_p, btn_R1_m, btn_R1_px, btn_R1_mx, btn_R1_dox, btn_R1_home,
btn_R1_con/*, btn_res_R1*/;

    @FXML
    public static Button btn_res_R1;
    @FXML
    private TextField txt_fld_R1_x, txt_fld_R1_x2;
    @FXML
    private ImageView img_R1_stat, img_wat;
    @FXML
    private Label lbl_R2, lbl_R2_pos, lbl_R2_stat, lbl_count_R2, lbl_err_R2, lbl_curR2;
    @FXML
    private Button btn_R2_p, btn_R2_m, btn_R2_px, btn_R2_mx, btn_R2_dox, btn_R2_home,
btn_R2_con, btn_res_R2;
    @FXML
    private TextField txt_fld_R2_x, txt_fld_R2_x2;
    @FXML
    private ImageView img_R2_stat, img_R1, img_R2;
    @FXML
    private Button btn_home_all, btn_load_v, btn_to_v, btn_disable_all, btn_load_v1;
    @FXML
    public Label lbl_R1_load, lbl_R2_load, lbl_T1_load, lbl_T2_load, lbl_T3_load;
    @FXML
    private ProgressBar proBar;
    @FXML
    private Label lbl_R11, lbl_R21;
    @FXML
    private Button btn_imp;
    @FXML
    private Label lbl_R1_anal, lbl_R2_anal;
    @FXML
    private Label lbl_Xkoord_anal, lbl_Ykoord_anal, lbl_Zkoord_anal;
    @FXML
    private Label lbl_target, lbl_measured, lbl_error, lbl_ntarget;
    @FXML
    private Label lbl_Xkoord_meas, lbl_Ykoord_meas, lbl_Zkoord_meas;
    @FXML
    private Label lbl_Xkoord_err, lbl_Ykoord_err, lbl_Zkoord_err;
    @FXML
    private Label lbl_Xkoord_ntar, lbl_Ykoord_ntar, lbl_Zkoord_ntar;
    @FXML
```

```
    private Label lbl_R1_ntar, lbl_R2_ntar;
    @FXML
    private TextField txt_fieldX, txt_fieldY, txt_fieldZ;
    @FXML
    private Separator sep1, sep2, sep3, sep4, sep5, sep6, sep7, sep8, sep9, sep10, sep11, sep12, sep13,
sep14;

    Alert aError=new Alert(AlertType.ERROR);
    ImageView
iIkona=new ImageView(new Image(getClass().getResourceAsStream(Slike.sIco64_1)));
    Stage error=(Stage) aError.getDialogPane().getScene().getWindow();

    Alert aPoruka=new Alert(AlertType.INFORMATION);
    Stage poruka1=(Stage) aPoruka.getDialogPane().getScene().getWindow();

    private void vPorukaMsg(String poruka,int i,int j){
        poruka1.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco16)));
        aPoruka.setTitle(Texts.sPower);
        switch(i){
            case 1:
                aPoruka.setHeaderText(Texts.sHeadPowerOff);
                switch(j){
                    case 1:
                        aPoruka.setContentText(Texts.sPowerRecOff+"R1");
                        break;
                    case 2:
                        aPoruka.setContentText(Texts.sPowerRecOff+"R2");
                        break;
                }
                break;
            case 2:
                aPoruka.setHeaderText(Texts.sHeadPowerOn);
                switch(j){
                    case 1:
                        aPoruka.setContentText(Texts.sPowerRecOn+"R1");
                        break;
                    case 2:
                        aPoruka.setContentText(Texts.sPowerRecOn+"R2");
                        break;
                }
                break;
        }
        aPoruka.showAndWait().ifPresent((ButtonType rs)->{
            if(rs==ButtonType.OK){

            }
        });
    }

    private void vErrorMsg(String poruka){
        error.getIcons().add(new Image(getClass().getResourceAsStream(Slike.sIco16)));
```

```
        aError.setTitle(Texts.sError);
        aError.setHeaderText(Texts.sHeadError);
        aError.setContentText(poruka);
        aError.showAndWait().ifPresent((ButtonType rs)->{
            if(rs==ButtonType.OK){

            }
        });
    }

    ArrayList<Label> listaLbl = new ArrayList<>();
    ArrayList<TextField> listaTxt = new ArrayList<>();
    ArrayList<Separator> listaSep = new ArrayList<>();
    ArrayList<Image> listaImgR1 = new ArrayList();
    ArrayList<Image> listaImgR2 = new ArrayList();

    @FXML
    public Label lbl_status;

    public Controler_prozor1() throws IOException {
        Controler_prozor1.sockPrimanje = new ServerSocket(5004);
    }

    @FXML
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        jsonPoruka.putIfAbsent("RefPos1", x_R1);
        jsonPoruka.putIfAbsent("Nare1", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("ReaPos1", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("Current1", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("Error1", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("RefPos2", x_R2);
        jsonPoruka.putIfAbsent("Nare2", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("ReaPos2", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("Current2", Komunikacija.sZero);
        jsonPoruka.putIfAbsent("Error2",Komunikacija.sZero);
        ArraySep();
        ArrayTxt();
        ArrayLbl();
        ArrayImgR1();
        ArrayImgR2();
        refresh();
        loadMjerenja();
        try {
            loadError();
        } catch (IOException ex) {
            vErrorMsg(Texts.sFileError);
        }
        timer.start();
        new Thread(() -> {
            while (bServer) {
```

```
            try {
                sockPrimanje = new ServerSocket(5004);
                stringPrimanje();
            } catch (IOException ex) {
                //Logger.getLogger(Controler_prozor1.class.getName()).log(Level.SEVERE, null, ex);
            } finally {
                try {
                    sockPrimanje.close();
                } catch (IOException ex) {
                    //Logger.getLogger(Controler_prozor1.class.getName()).log(Level.SEVERE, null, ex);
                }
            }

        }
    }).start();
}

private void buildCamera() {
    root1.getChildren().add(cameraXform);
    cameraXform.getChildren().add(cameraXform2);
    cameraXform2.getChildren().add(cameraXform3);
    cameraXform3.getChildren().add(camera);
    cameraXform3.setRotateZ(180.0);
    camera.setNearClip(CAMERA_NEAR_CLIP);
    camera.setFarClip(CAMERA_FAR_CLIP);
    camera.setTranslateZ(CAMERA_INITIAL_DISTANCE);
    cameraXform.ry.setAngle(CAMERA_INITIAL_Y_ANGLE);
    cameraXform.rx.setAngle(CAMERA_INITIAL_X_ANGLE);
    cameraXform.rz.setAngle(CAMERA_INITIAL_Z_ANGLE);
}

private void buildAxes() {
    final PhongMaterial redMaterial = new PhongMaterial();
    redMaterial.setDiffuseColor(Color.DARKRED);
    redMaterial.setSpecularColor(Color.RED);
    final PhongMaterial greenMaterial = new PhongMaterial();
    greenMaterial.setDiffuseColor(Color.DARKGREEN);
    greenMaterial.setSpecularColor(Color.GREEN);
    final PhongMaterial blueMaterial = new PhongMaterial();
    blueMaterial.setDiffuseColor(Color.DARKBLUE);
    blueMaterial.setSpecularColor(Color.BLUE);
    final Box xAxis = new Box(AXIS_LENGTH, 1, 1);
    final Box yAxis = new Box(1, AXIS_LENGTH, 1);
    final Box zAxis = new Box(1, 1, AXIS_LENGTH);
    xAxis.setMaterial(redMaterial);
    yAxis.setMaterial(greenMaterial);
    zAxis.setMaterial(blueMaterial);
    axisGroup.getChildren().addAll(xAxis, yAxis, zAxis);
    axisGroup.setVisible(true);
    world.getChildren().addAll(axisGroup);
}
```

```
    private void handleMouse(Scene scene, final Node root) {
        scene.setOnMousePressed((MouseEvent me) -> {
            mousePosX = me.getSceneX();
            mousePosY = me.getSceneY();
            mouseOldX = me.getSceneX();
            mouseOldY = me.getSceneY();
        });
        scene.setOnMouseDragged((MouseEvent me) -> {
            mouseOldX = mousePosX;
            mouseOldY = mousePosY;
            mousePosX = me.getSceneX();
            mousePosY = me.getSceneY();
            mouseDeltaX = (mousePosX - mouseOldX);
            mouseDeltaY = (mousePosY - mouseOldY);
            double modifier = 1.0;
            if (me.isControlDown()) {
                modifier = CONTROL_MULTIPLIER;
            }
            if (me.isShiftDown()) {
                modifier = SHIFT_MULTIPLIER;
            }
            if (me.isPrimaryButtonDown()) {
                cameraXform.ry.setAngle(cameraXform.ry.getAngle() - mouseDeltaX * MOUSE_SPEED *
modifier * ROTATION_SPEED);
                cameraXform.rx.setAngle(cameraXform.rx.getAngle() + mouseDeltaY * MOUSE_SPEED
* modifier * ROTATION_SPEED);
            } else if (me.isSecondaryButtonDown()) {
                double z = camera.getTranslateZ();
                double newZ = z + mouseDeltaX * MOUSE_SPEED * modifier;
                camera.setTranslateZ(newZ);
            } else if (me.isMiddleButtonDown()) {
                cameraXform2.t.setX(cameraXform2.t.getX() + mouseDeltaX * MOUSE_SPEED *
modifier * TRACK_SPEED);
                cameraXform2.t.setY(cameraXform2.t.getY() + mouseDeltaY * MOUSE_SPEED *
modifier * TRACK_SPEED);
            }
        });
    }

    private void handleKeyboard(Scene scene, final Node root) {
        scene.setOnKeyPressed((KeyEvent event) -> {
            switch (event.getCode()) {
                case Z:
                    cameraXform2.t.setX(0.0);
                    cameraXform2.t.setY(0.0);
                    camera.setTranslateZ(CAMERA_INITIAL_DISTANCE);
                    cameraXform.ry.setAngle(CAMERA_INITIAL_Y_ANGLE);
                    cameraXform.rx.setAngle(CAMERA_INITIAL_X_ANGLE);
                    break;
                case X:
```

```
                axisGroup.setVisible(!axisGroup.isVisible());
                break;
            case V:
                Group3D.setVisible(!Group3D.isVisible());
                break;
            case C:
                Controler_prozor1.bServer=false;
                try {
                    Controler_prozor1.sockPrimanje.close();
                } catch (IOException ex) {
                    //Logger.getLogger(Controler_prozor1.class.getName()).log(Level.SEVERE, null, ex);
                }
                exit();
                break;
            default:
        }
    });
}

public void refresh3DY(double kut) {
    hydrogen3SideXform.setRotateX(kut);
}

public void refresh3DZ(double kut) {
    hydrogen3SideXform.setRotateY(kut);
}

private void build3D() {
    redMaterial.setDiffuseColor(Color.DARKRED);
    redMaterial.setSpecularColor(Color.RED);
    whiteMaterial.setDiffuseColor(Color.WHITE);
    whiteMaterial.setSpecularColor(Color.LIGHTBLUE);
    greyMaterial.setDiffuseColor(Color.DARKGREY);
    greyMaterial.setSpecularColor(Color.GREY);
    bond3Cylinder.setMaterial(greyMaterial);
    bond3Cylinder.setTranslateX(0);
    bond3Cylinder.setRotationAxis(Rotate.Z_AXIS);
    bond3Cylinder.setRotate(90);
    XForm3D.getChildren().add(hydrogen3SideXform);
    hydrogen3SideXform.getChildren().add(hydrogen3Xform);
    hydrogen3Xform.getChildren().add(hydrogen3Sphere);
    hydrogen3SideXform.getChildren().add(bond3Cylinder);
    hydrogen3Xform.setTy(100.0);
    hydrogen3SideXform.setRotateX(0);
    Group3D.getChildren().add(XForm3D);
    world.getChildren().addAll(Group3D);
}


public void refresh() {
    btn_R1_dox.setText(Texts.sToX);
```

```
        btn_R2_dox.setText(Texts.sToX);
        btn_R1_home.setText(Texts.sHOME);
        btn_R2_home.setText(Texts.sHOME);
        btn_R1_con.setText(Texts.sDiss);
        btn_R2_con.setText(Texts.sDiss);
        btn_home_all.setText(Texts.sHomeAll);
        btn_load_v.setText(Texts.sSetV);
        btn_load_v1.setText(Texts.sLoaV);
        btn_to_v.setText(Texts.sToV);
        btn_disable_all.setText(Texts.sDissAll);
        item_anal.setText(Texts.sAna);
        menuIP.setText(Texts.sIP);
        close_btn.setText(Texts.sCls);
        menuAbout.setText(Texts.sAb);
        menu_Lang.setText(Texts.sLang);
        menu_file.setText(Texts.sFile);
        menu_edit.setText(Texts.sEdit);
        menu_help.setText(Texts.sHelp);
        lbl_status.setText(Texts.sNoCon);
        btn_imp.setText(Texts.sImport);
    }

    private void ArrayLbl() {
        listaLbl.add(0, lbl_R1_anal);
        listaLbl.add(1, lbl_R2_anal);
        listaLbl.add(2, lbl_Xkoord_anal);
        listaLbl.add(3, lbl_Ykoord_anal);
        listaLbl.add(4, lbl_Zkoord_anal);
        listaLbl.add(5, lbl_target);
        listaLbl.add(6, lbl_measured);
        listaLbl.add(7, lbl_ntarget);
        listaLbl.add(8, lbl_Xkoord_meas);
        listaLbl.add(9, lbl_Ykoord_meas);
        listaLbl.add(10, lbl_Zkoord_meas);
        listaLbl.add(11, lbl_Xkoord_err);
        listaLbl.add(12, lbl_Ykoord_err);
        listaLbl.add(13, lbl_Zkoord_err);
        listaLbl.add(14, lbl_Xkoord_ntar);
        listaLbl.add(15, lbl_Ykoord_ntar);
        listaLbl.add(16, lbl_Zkoord_ntar);
        listaLbl.add(17, lbl_R1_ntar);
        listaLbl.add(18, lbl_R2_ntar);
        listaLbl.add(19, lbl_error);
    }

    private void ArrayTxt() {
        listaTxt.add(0, txt_fieldX);
        listaTxt.add(1, txt_fieldY);
        listaTxt.add(2, txt_fieldZ);
    }
```

```
private void ArraySep() {
    listaSep.add(0, sep1);
    listaSep.add(1, sep2);
    listaSep.add(2, sep3);
    listaSep.add(3, sep4);
    listaSep.add(4, sep5);
    listaSep.add(5, sep6);
    listaSep.add(6, sep7);
    listaSep.add(7, sep8);
    listaSep.add(8, sep9);
    listaSep.add(9, sep10);
    listaSep.add(10, sep11);
    listaSep.add(11, sep12);
    listaSep.add(12, sep13);
    listaSep.add(13, sep14);
}

private void ArrayImgR1() {
    int iX;
    String sX;
    for (iX = 0; iX < 190; iX++) {
        sX = df1.format(iX);
        img = new Image(getClass().getResourceAsStream(sX + ".png"));
        listaImgR1.add(iX, img);
    }
}

private void ArrayImgR2() {
    int iX;
    String sX;
    for (iX = 0; iX < 188; iX++) {
        sX = df1.format(iX);
        img2 = new Image(getClass().getResourceAsStream("R2_" + sX + ".png"));
        listaImgR2.add(iX, img2);
    }
}

/**
 * Refresh status bar
 */
public void statusLabel() {
    this.lbl_status.setText(Texts.sConMsg + unos_IP);
}

/**
 * Ispis poruke No connection na status baru Controler_prozor1
 */
public void statusLabel_nemaKonekcije() {
    this.lbl_status.setText(Texts.sNoCon);
}
```

```
    private void slikaR1(double x_R1_S) {
        slika1 = (int) Math.round(x_R1_S);
        img_R1.setImage(listaImgR1.get(slika1 + 10));
        img = null;
    }

    private void slikaR2(double x_R2_S) {
        slika2 = (int) Math.round(x_R2_S);
        img_R2.setImage(listaImgR2.get(slika2 + 8));
        img2 = null;
    }

    private String povecanje_kuta(double x, int iOs) {
        df.setMaximumFractionDigits(4); //340 = DecimalFormat.DOUBLE_FRACTION_DIGITS
        df1.setMaximumFractionDigits(0); //340 = DecimalFormat.DOUBLE_FRACTION_DIGITS
        switch (iOs){
            case 1:
                x_R1 = x_R1 + x;
                if (x_R1 < -10) {
                    x_R1 = -10;
                }
                if (x_R1 > 190) {
                    x_R1 = 190;
                }
                x_R1_string = df.format(x_R1);
                Platform.runLater(() -> lbl_R1_pos.setText(x_R1_string + "°"));
                Platform.runLater(() -> lbl_R1_anal.setText(Texts.sRot1 + x_R1_string + "°"));
                return (x_R1_string);
            case 2:
                x_R2 = x_R2 + x;
                if (x_R2 < -8) {
                    x_R2 = -8;
                }
                if (x_R2 > 188) {
                    x_R2 = 188;
                }
                x_R2_string = df.format(x_R2);
                Platform.runLater(() -> lbl_R2_pos.setText(x_R2_string + "°"));
                Platform.runLater(() -> lbl_R2_anal.setText(Texts.sRot2 + x_R2_string + "°"));
                return (x_R2_string);
            case 3:
                return("0");
            case 4:
                return("0");
            case 5:
                return("0");
            default :
                return("0");
        }
```

```
    }

    private void refresh3D() {
        refresh3DY(x_R2);
        refresh3DZ(x_R1);
    }

    private String sTrim(String sUlaz) {
        df.setMaximumFractionDigits(4);
        dMedjuPrimanje1 = Double.parseDouble(sUlaz.trim());
       dMedjuPrimanje1=dMedjuPrimanje1/186413.61111111;
//        dMedjuPrimanje1 = dMedjuPrimanje1 / 27.777777;
        dErr1 = x_R1 - dMedjuPrimanje1;
        sErr1 = df.format(dErr1);
        return (df.format(dMedjuPrimanje1));
    }

    private String sTrim2(String sUlaz) {
        df.setMaximumFractionDigits(4);
        dMedjuPrimanje2 = Double.parseDouble(sUlaz.trim());
        dMedjuPrimanje2 = dMedjuPrimanje2 / 955.267;
        dErr2 = x_R2 - dMedjuPrimanje2;
        sErr2 = df.format(dErr2);
        return (df.format(dMedjuPrimanje2));
    }

    private String sTrim3(String sUlaz) {
        df.setMaximumFractionDigits(4);
        dMedjuPrimanje3 = Double.parseDouble(sUlaz.trim());
        return (df.format(dMedjuPrimanje3));
    }

    private String sTrim4(String sUlaz) {
        df.setMaximumFractionDigits(4);
        dMedjuPrimanje4 = Double.parseDouble(sUlaz.trim());
        return (df.format(dMedjuPrimanje4));
    }

    final long minUpdateInterval = 500; // nanoseconds. Set to higher number to slow output.
    final LongProperty lastUpdate = new SimpleLongProperty();
    AnimationTimer timer = new AnimationTimer() {
        @Override
        public void handle(long now) {
            df.setMaximumFractionDigits(4);
            if (now - lastUpdate.get() > minUpdateInterval) {
                if (sPoruka != null) {
                    sPorukaOs = sPoruka.substring(0, 2);
                    if (sPorukaOs.equals(Komunikacija.sR1)) {
                        sPorukaVrijednost = sPoruka.substring(2, sPoruka.length());
                        lbl_count_R1.setText(sTrim(sPorukaVrijednost) + "°");
                        slikaR1(dMedjuPrimanje1);
```

```
            lbl_err_R1.setText(sErr1 + "°");
            jsonPoruka.replace("RelPos1", dMedjuPrimanje1);
            refresh3DZ(dMedjuPrimanje1);
        }
        if (sPorukaOs.equals(Komunikacija.sR2)) {
            sPorukaVrijednost2 = sPoruka.substring(2, sPoruka.length());
            lbl_count_R2.setText(sTrim2(sPorukaVrijednost2) + "°");
            slikaR2(dMedjuPrimanje2);
            lbl_err_R2.setText(sErr2 + "°");
            jsonPoruka.replace("RelPos2", dMedjuPrimanje2);
            refresh3DY(dMedjuPrimanje2);
        }
        if (sPorukaOs.equals(Komunikacija.sCurrent1)) {
            sPorukaVrijednost3 = sPoruka.substring(2, sPoruka.length());
            lbl_curR1.setText(sTrim3(sPorukaVrijednost3) + "A");
            jsonPoruka.replace("Current1", sPorukaVrijednost3);
        }
        if (sPorukaOs.equals(Komunikacija.sCurrent2)) {
            sPorukaVrijednost4 = sPoruka.substring(2, sPoruka.length());
            lbl_curR2.setText(sTrim4(sPorukaVrijednost4) + "A");
            jsonPoruka.replace("Current2", sPorukaVrijednost4);
        }
        if (sPorukaOs.equals(Komunikacija.sError1)) {
            sPorukaVrijednost5=sPoruka.substring(2, sPoruka.length());
            lbl_status.setText(sPorukaVrijednost5);
            btn_res_R1.setDisable(false);
            jsonPoruka.replace("Error1", sPorukaVrijednost5);

            dissR1(btn_disable);
        }
        if (sPorukaOs.equals(Komunikacija.sError2)) {
            sPorukaVrijednost6=sPoruka.substring(2, sPoruka.length());
            lbl_status.setText(sPorukaVrijednost6);
            btn_res_R2.setDisable(false);
            jsonPoruka.replace("Error2",sPorukaVrijednost6);

            dissR2(btn2_disable);
        }
    }
    motStat1();
    motStat2();
    refresh();
    if (anal!=anal1){
        skrivanje();
        anal1=anal;
    }
    if (proracun!=proracun1){
        analizaRacun();
        proracun1=proracun;
    }
    lastUpdate.set(now);
```

```
            }
        }
    };


    private void stringPrimanje() throws IOException {
        df.setMaximumFractionDigits(4);
        stringPrimljenUBytovima = null;
        try {
            s1 = sockPrimanje.accept();

            inXSR = new InputStreamReader(s1.getInputStream());
            disPrimanje = new BufferedReader(inXSR);
            sPoruka = null;
            sPoruka = disPrimanje.readLine();
            disPrimanje.close();
            inXSR.close();
            s1.close();
        } catch (IOException e) {
            //vErrorMsg(Texts.sErrorRec);
        }
    }

    private void stringSlanje(String os, String naredba, String pozicija) throws
IOException, InterruptedException {
        df.setMaximumFractionDigits(4);
        kod = os + naredba + pozicija;
        stringuBitovima = kod.getBytes();
        if(os.equals("1")){
            jsonPoruka.replace("Nare1", naredba);
            jsonPoruka.replace("RefPos1",pozicija);
        }
        if(os.equals("2")){
            jsonPoruka.replace("Nare2", naredba);
            jsonPoruka.replace("RefPos2",pozicija);
        }
        System.out.println(jsonPoruka);
        try {
            s = new Socket(unos_IP, 5005);
            s.getOutputStream().write(stringuBitovima);
            new Thread(() -> {
                for (int i_progress = 0; i_progress <= 10; i_progress++) {
                    try {
                        Thread.sleep(50);
                    } catch (InterruptedException e) {
                        this.statusLabel_nemaKonekcije();
                    }
                    if (i_progress < 10) {
                        progress = i_progress * 0.1;
                        Platform.runLater(() -> proBar.setProgress(progress));
                    } else {
```

```
                    Platform.runLater(() ->
proBar.setProgress(ProgressBar.INDETERMINATE_PROGRESS));
                }
            }
        }).start();
        Thread.currentThread().interrupt();
        Platform.runLater(() -> statusLabel());
        Platform.runLater(() -> analizaRacun());
    } catch (IOException e) {
        Platform.runLater(() -> analizaRacun());
        Platform.runLater(() -> vErrorMsg(e.getLocalizedMessage()));
        Platform.runLater(() -> this.statusLabel_nemaKonekcije());
    } finally {
        s.close();
    }
}

@FXML
private void handleLang(ActionEvent event) throws IOException {
    Texts.jezik();
    refresh();
}

@FXML
private void handebtnR1pAction(ActionEvent event) throws IOException, InterruptedException {
    new Thread(() -> {
        try {
            stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(1, 1));
        } catch (IOException | InterruptedException ex) {
            statusLabel_nemaKonekcije();
            Logger.getLogger(Controler_prozor1.class.getName()).log(Level.SEVERE, null, ex);
        }
    }).start();
}

@FXML
private void handebtnR1mAction(ActionEvent event) throws IOException, InterruptedException {
    new Thread(() -> {
        try {
            stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(-1, 1));
        } catch (IOException | InterruptedException ex) {
            statusLabel_nemaKonekcije();
            Logger.getLogger(Controler_prozor1.class.getName()).log(Level.SEVERE, null, ex);
        }
    }).start();
}

@FXML
private void handebtnR1pxAction(ActionEvent event) throws IOException, InterruptedException {
    x_s = txt_fld_R1_x.getText();
    x_1 = Double.parseDouble(x_s.trim());
```

```
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(x_1, 1));
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR1mxAction(ActionEvent event) throws IOException, InterruptedException {
        x_s = txt_fld_R1_x.getText();
        x_1 = Double.parseDouble(x_s.trim());
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(-x_1, 1));
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR1doxAction(ActionEvent event) throws
IOException, InterruptedException {
        x_sdox = txt_fld_R1_x2.getText();
        x_2 = Double.parseDouble(x_sdox.trim());
        x_R1 = x_2;
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(0, 1));
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR1homeAction(ActionEvent event) throws
IOException, InterruptedException {
        x_R1 = 0;
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(0, 1));
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
```

```
    private void handebtnR2pAction(ActionEvent event) throws IOException, InterruptedException {
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(1, 2));//stringSlanjer2
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR2mAction(ActionEvent event) throws IOException, InterruptedException {
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(-
1, 2));//stringSlanjer2
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR2pxAction(ActionEvent event) throws IOException, InterruptedException {
        x_s = txt_fld_R2_x.getText();
        x_2 = Double.parseDouble(x_s.trim());
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove,
povecanje_kuta(x_2, 2));//stringSlanjer2
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR2mxAction(ActionEvent event) throws IOException, InterruptedException {
        x_s = txt_fld_R2_x.getText();
        x_2 = Double.parseDouble(x_s.trim());
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(-
x_2, 2));//stringSlanjer2
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
```

```
    private void handebtnR2doxAction(ActionEvent event) throws
IOException, InterruptedException {
        x_sdox2 = txt_fld_R2_x2.getText();
        x_2 = Double.parseDouble(x_sdox2.trim());
        x_R2 = x_2;
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(0, 2));//stringSlanjer2
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    @FXML
    private void handebtnR2homeAction(ActionEvent event) throws
IOException, InterruptedException {
        x_R2 = 0;
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(0, 2));//stringSlanjer2
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    private void motStat1() {
        if (btn_disable == false) {
            if ((dErr1 > 0.005) || (dErr1 < -0.005)) {
                btn_R1_p.setDisable(true);
                btn_R1_m.setDisable(true);
                btn_R1_px.setDisable(true);
                btn_R1_mx.setDisable(true);
                btn_R1_dox.setDisable(true);
                btn_R1_home.setDisable(true);
                img_R1.setDisable(true);
                lbl_R1_stat.setText(Texts.sMov);
                btn_R1_con.setText(Texts.sMov);
            } else {
                btn_R1_p.setDisable(false);
                btn_R1_m.setDisable(false);
                btn_R1_px.setDisable(false);
                btn_R1_mx.setDisable(false);
                btn_R1_dox.setDisable(false);
                btn_R1_home.setDisable(false);
                img_R1.setDisable(false);
                lbl_R1_stat.setText(Texts.sEna);
                btn_R1_con.setText(Texts.sDiss);
            }
        }
```

```
    }

    private void motStat2() {
        if (btn2_disable == false) {
            if ((dErr2 > 0.2) || (dErr2 < -0.2)) {
                btn_R2_p.setDisable(true);
                btn_R2_m.setDisable(true);
                btn_R2_px.setDisable(true);
                btn_R2_mx.setDisable(true);
                btn_R2_dox.setDisable(true);
                btn_R2_home.setDisable(true);
                img_R2.setDisable(true);
                lbl_R2_stat.setText(Texts.sMov);
                btn_R2_con.setText(Texts.sMov);
            } else {
                btn_R2_p.setDisable(false);
                btn_R2_m.setDisable(false);
                btn_R2_px.setDisable(false);
                btn_R2_mx.setDisable(false);
                btn_R2_dox.setDisable(false);
                btn_R2_home.setDisable(false);
                img_R2.setDisable(false);
                lbl_R2_stat.setText(Texts.sEna);
                btn_R2_con.setText(Texts.sDiss);
            }
        }
    }

    private void dissR1(boolean bool) {
        btn_R1_p.setDisable(!bool);
        btn_R1_m.setDisable(!bool);
        btn_R1_px.setDisable(!bool);
        btn_R1_mx.setDisable(!bool);
        btn_R1_dox.setDisable(!bool);
        btn_R1_home.setDisable(!bool);
        btn_disable = !bool;
    }

    private void dissR2(boolean bool) {
        btn_R2_p.setDisable(!bool);
        btn_R2_m.setDisable(!bool);
        btn_R2_px.setDisable(!bool);
        btn_R2_mx.setDisable(!bool);
        btn_R2_dox.setDisable(!bool);
        btn_R2_home.setDisable(!bool);
        btn2_disable = !bool;
    }

    @FXML
    private void handebtnR1dissAction(ActionEvent event) throws
```

```
IOException, InterruptedException {
        if (false == btn_disable) {
            //dissR1(btn_disable);
            vPorukaMsg("",1,1);
            btn_R1_p.setDisable(true);
            btn_R1_m.setDisable(true);
            btn_R1_px.setDisable(true);
            btn_R1_mx.setDisable(true);
            btn_R1_dox.setDisable(true);
            btn_R1_home.setDisable(true);
            img_R1.setDisable(true);
            lbl_R1_stat.setText(Texts.sDisa);
            btn_disable = true;
            btn_R1_con.setText(Texts.sConn);
            img_R1_stat.setImage(dis);
            new Thread(() -> {
                try {
                    stringSlanje(Komunikacija.R1, Komunikacija.sDiss, Komunikacija.sZero);
                } catch (IOException | InterruptedException ex) {
                    statusLabel_nemaKonekcije();
                }
            }).start();
        } else {
            vPorukaMsg("",2,1);
            //dissR1(btn_disable);
            btn_R1_p.setDisable(false);
            btn_R1_m.setDisable(false);
            btn_R1_px.setDisable(false);
            btn_R1_mx.setDisable(false);
            btn_R1_dox.setDisable(false);
            btn_R1_home.setDisable(false);
            img_R1.setDisable(false);
            lbl_R1_stat.setText(Texts.sEna);
            btn_disable = false;
            btn_R1_con.setText(Texts.sDiss);
            img_R1_stat.setImage(ena);
            new Thread(() -> {
                try {
                    stringSlanje(Komunikacija.R1, Komunikacija.sEnab, Komunikacija.sZero);
                } catch (IOException | InterruptedException ex) {
                    statusLabel_nemaKonekcije();
                }
            }).start();
        }
        if ((true == btn_disable) && (btn2_disable == true)) {
            btn_disable_all.setText(Texts.sConAll);
        }
        if ((false == btn_disable) && (btn2_disable == false)) {
            btn_disable_all.setText(Texts.sDissAll);
        }
    }
```

```
@FXML
private void handebtnR2dissAction(ActionEvent event) {
    if (false == btn2_disable) {
        vPorukaMsg("",1,2);
        btn_R2_p.setDisable(true);
        btn_R2_m.setDisable(true);
        btn_R2_px.setDisable(true);
        btn_R2_mx.setDisable(true);
        btn_R2_dox.setDisable(true);
        btn_R2_home.setDisable(true);
        //dissR2(btn2_disable);
        lbl_R2_stat.setText(Texts.sDisa);
        btn2_disable = true;
        btn_R2_con.setText(Texts.sConn);
        img_R2_stat.setImage(dis);
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sDiss, Komunikacija.sZero);
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    } else {
        vPorukaMsg("",2,2);
        btn_R2_p.setDisable(false);
        btn_R2_m.setDisable(false);
        btn_R2_px.setDisable(false);
        btn_R2_mx.setDisable(false);
        btn_R2_dox.setDisable(false);
        btn_R2_home.setDisable(false);
        //dissR2(btn2_disable);
        lbl_R2_stat.setText(Texts.sEna);
        btn2_disable = false;
        btn_R2_con.setText(Texts.sDiss);
        img_R2_stat.setImage(ena);
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sEnab, Komunikacija.sZero);
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }
    if ((true == btn_disable) && (btn2_disable == true)) {
        btn_disable_all.setText(Texts.sConAll);
    }
    if ((false == btn_disable) && (btn2_disable == false)) {
        btn_disable_all.setText(Texts.sDissAll);
    }
}
```

```
@FXML
private void homeAll(ActionEvent event) throws IOException, InterruptedException {
    x_R1 = 0;
    x_R2 = 0;
    new Thread(() -> {
        try {
            stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(0, 1));
        } catch (IOException | InterruptedException ex) {
            statusLabel_nemaKonekcije();
        }
    }).start();
    new Thread(() -> {
        try {
            stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(0, 2));//stringSlanjer2
        } catch (IOException | InterruptedException ex) {
            statusLabel_nemaKonekcije();
        }
    }).start();
}

@FXML
private void handebtnDissAll(ActionEvent event) throws IOException, InterruptedException {
    if ((true == btn_disable) && (btn2_disable == true)) {
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.D9, Komunikacija.sEnab, Komunikacija.sZero);
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
        btn_R1_p.setDisable(!btn_disable);
        btn_R1_m.setDisable(!btn_disable);
        btn_R1_px.setDisable(!btn_disable);
        btn_R1_mx.setDisable(!btn_disable);
        btn_R1_dox.setDisable(!btn_disable);
        btn_R1_home.setDisable(!btn_disable);
        img_R1.setDisable(!btn_disable);
        lbl_R1_stat.setText(Texts.sEna);
        btn_disable = !btn_disable;
        btn_R1_con.setText(Texts.sDiss);
        btn_disable_all.setText(Texts.sDissAll);
        img_R1_stat.setImage(ena);

        btn_R2_p.setDisable(!btn2_disable);
        btn_R2_m.setDisable(!btn2_disable);
        btn_R2_px.setDisable(!btn2_disable);
        btn_R2_mx.setDisable(!btn2_disable);
        btn_R2_dox.setDisable(!btn2_disable);
        btn_R2_home.setDisable(!btn2_disable);
        lbl_R2_stat.setText(Texts.sEna);
```

```
            btn2_disable = !btn2_disable;
            btn_R2_con.setText(Texts.sDiss);
            btn_R2_con.setText(Texts.sDissAll);
            img_R2_stat.setImage(ena);
        } else {
            new Thread(() -> {
                try {
                    stringSlanje(Komunikacija.D9, Komunikacija.sDiss, Komunikacija.sZero);
                } catch (IOException | InterruptedException ex) {
                    statusLabel_nemaKonekcije();
                }
            }).start();
            if (btn_disable == false) {
                btn_R1_p.setDisable(!btn_disable);
                btn_R1_m.setDisable(!btn_disable);
                btn_R1_px.setDisable(!btn_disable);
                btn_R1_mx.setDisable(!btn_disable);
                btn_R1_dox.setDisable(!btn_disable);
                btn_R1_home.setDisable(!btn_disable);
                img_R1.setDisable(true);
                lbl_R1_stat.setText(Texts.sDisa);
                btn_disable = !btn_disable;
                btn_R1_con.setText(Texts.sConn);
                btn_disable_all.setText(Texts.sConAll);
                img_R1_stat.setImage(dis);
            }
            if (btn2_disable == false) {
                btn_R2_p.setDisable(!btn2_disable);
                btn_R2_m.setDisable(!btn2_disable);
                btn_R2_px.setDisable(!btn2_disable);
                btn_R2_mx.setDisable(!btn2_disable);
                btn_R2_dox.setDisable(!btn2_disable);
                btn_R2_home.setDisable(!btn2_disable);
                lbl_R2_stat.setText(Texts.sDisa);
                btn2_disable = !btn2_disable;
                btn_R2_con.setText(Texts.sConn);
                btn_disable_all.setText(Texts.sConAll);
                img_R2_stat.setImage(dis);
            }
        }
    }

    @FXML
    public void closeAll(ActionEvent event) throws IOException {
        bServer = false;
        Controler_prozor1.sockPrimanje.close();
        exit();
    }

    @FXML
    private void handleUnosBtn(ActionEvent event) {
```

```
    Parent root;
    try {
        drugiProzor = new FXMLLoader();
        drugiProzor.setLocation(Controler_prozor1.class.getResource(FxmlFiles.sUnos));
        rootDrugiProzor = (AnchorPane) drugiProzor.load();
        scena2 = new Scene(rootDrugiProzor);
        unos_stage2 = new Stage();
        unos_stage2.setScene(scena2);
        unos_stage2.setTitle(Texts.sMainTitle + Texts.sAddTitleIN);
        unos_stage2.setResizable(false);
        unos_stage2.setScene(scena2);
        unos_stage2.initStyle(StageStyle.UTILITY);
        unos_stage2.show();
    } catch (IOException e) {
    }
}

@FXML
private void handleMenuIP(ActionEvent event) {
    menuIPG.menuIP2();
}

@FXML
private void handleMenuAbout(ActionEvent event) {
    menuIPG.menuAbout();
}

@FXML
private void handle3d(ActionEvent event) throws IOException {
    root1.getChildren().add(world);
    root1.setDepthTest(DepthTest.ENABLE);
    buildCamera();
    buildAxes();
    build3D();
    Scene scene5 = new Scene(root1, 1024, 768, true);
    scene5.setFill(Color.GREY);
    handleKeyboard(scene5, world);
    handleMouse(scene5, world);
    stage5_3d = new Stage();
    stage5_3d.setScene(scene5);
    stage5_3d.setTitle(Texts.sNero3D);
    stage5_3d.show();
    scene5.setCamera(camera);
}

@FXML
private void handleLoadValue(ActionEvent event) {
    lbl_R1_load.setText("R1:" + Controler_unos.unos_R1 + " °");
    lbl_R2_load.setText("R2:" + Controler_unos.unos_R2 + " °");
    lbl_T1_load.setText("T1(Z):" + Controler_unos.unos_T1 + Texts.smm);
    lbl_T2_load.setText("T2(X):" + Controler_unos.unos_T2 + Texts.smm);
```

```
        lbl_T3_load.setText("T3(Y):" + Controler_unos.unos_T3 + Texts.smm);
    }

    @FXML
    private void handleToVbtn(ActionEvent event) throws IOException, InterruptedException {
        x_R1 = Double.parseDouble(Controler_unos.unos_R1);
        x_R2 = Double.parseDouble(Controler_unos.unos_R2);
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R1, Komunikacija.sMove, povecanje_kuta(0, 1));
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
        new Thread(() -> {
            try {
                stringSlanje(Komunikacija.R2, Komunikacija.sMove, povecanje_kuta(0, 2));
            } catch (IOException | InterruptedException ex) {
                statusLabel_nemaKonekcije();
            }
        }).start();
    }

    public static void vSwap(){
        anal=!anal;
    }
    public static void analiza(){
        proracun=!proracun;
    }

    public void skrivanje() {
        if (btn_skrivanje == true) {
            stageTitle = (Stage) btn_R1_con.getScene().getWindow();
            stageTitle.setTitle(Texts.sMainTitle + Texts.sAddTitleAN);
            img_R1.setVisible(!btn_skrivanje);
            img_R2.setVisible(!btn_skrivanje);
            lbl_R11.setVisible(!btn_skrivanje);
            lbl_R21.setVisible(!btn_skrivanje);
            img_wat.setVisible(!btn_skrivanje);
            item_anal.setText(Texts.sPos);
            btn_imp.setVisible(btn_skrivanje);
            for (iBroj = 0; iBroj < 20; iBroj++) {
                listaLbl.get(iBroj).setVisible(btn_skrivanje);
            }
            for (iBroj = 0; iBroj < 3; iBroj++) {
                listaTxt.get(iBroj).setVisible(btn_skrivanje);
            }
            for (iBroj = 0; iBroj < 14; iBroj++) {
                listaSep.get(iBroj).setVisible(btn_skrivanje);
            }
            btn_skrivanje = !btn_skrivanje;
```

```
        } else {
            stageTitle = (Stage) btn_R1_con.getScene().getWindow();
            stageTitle.setTitle(Texts.sMainTitle);
            img_R1.setVisible(!btn_skrivanje);
            img_R2.setVisible(!btn_skrivanje);
            lbl_R11.setVisible(!btn_skrivanje);
            lbl_R21.setVisible(!btn_skrivanje);
            img_wat.setVisible(!btn_skrivanje);
            item_anal.setText(Texts.sAna);
            btn_imp.setVisible(btn_skrivanje);
            for (iBroj = 0; iBroj < 20; iBroj++) {
                listaLbl.get(iBroj).setVisible(btn_skrivanje);
            }
            for (iBroj = 0; iBroj < 3; iBroj++) {
                listaTxt.get(iBroj).setVisible(btn_skrivanje);
            }
            for (iBroj = 0; iBroj < 14; iBroj++) {
                listaSep.get(iBroj).setVisible(btn_skrivanje);
            }
            btn_skrivanje = !btn_skrivanje;
        }
    }

    private String racunX(double r1, double r2) {
        df.setMaximumFractionDigits(6);
//        r1=Math.toRadians(r1-25);
        r2=Math.toRadians(r2);
        x_tar= 100* cos(25*Math.PI/180)*cos(r2);
        String x_tars = df.format(x_tar);
        return (x_tars);
    }


    private String racunY(double r1, double r2) {
        df.setMaximumFractionDigits(6);
        r1=Math.toRadians(r1-25);
        r2=Math.toRadians(r2);
        y_tar=(cos(-r1)*100*sin(25*Math.PI/180))+(-sin(-r1)*100*cos(25*Math.PI/180)*sin(r2));
        String y_tars = df.format(y_tar);
        return (y_tars);
    }

    private String racunZ(double r1, double r2) {
        df.setMaximumFractionDigits(6);
        r1=Math.toRadians(r1-25);
        r2=Math.toRadians(r2);
        z_tar=(sin(-r1)*100*sin(25*Math.PI/180))+(cos(-r1)*100*cos(25*Math.PI/180)*sin(r2));
        String z_tars = df.format(z_tar);
        return (z_tars);
    }
```

```
    private String racunErr(double t, double m) {
        df.setMaximumFractionDigits(6);
        iz = t - m;
        String x_izl = df.format(iz);
        return (x_izl);
    }

    private String racunNtar(double t, double m) {
        df.setMaximumFractionDigits(6);
        er = Double.parseDouble(racunErr(t, m).trim());
        ntar = t + er;
        String x_izl = df.format(ntar);
        return (x_izl);
    }

    @FXML
    private void handleAnal(ActionEvent event) {
        skrivanje();
    }

    private void loadMjerenja(){
        iCounter=0;
        BufferedReader brM;
        FileReader frM;
        txt_fieldX.setEditable(false);
        txt_fieldY.setEditable(false);
        txt_fieldZ.setEditable(false);
        try{
            frM=new FileReader(FxmlFiles.sFilePodatci);
            brM=new BufferedReader(frM);
            String sTrenutnaLinija;
            while ((sTrenutnaLinija=brM.readLine())!=null){
                linesMj[iCounter]= sTrenutnaLinija;
                String [] lol=linesMj[iCounter].split(",");
                for (jCounter=0;jCounter<5;jCounter++){
                    linesMj2[iCounter][jCounter]=lol[jCounter];
                }
                iCounter=iCounter+1;
            }
        }catch(IOException e){
        }
    }

    private void loadError() throws IOException{
        BufferedWriter bwM;
        FileWriter fwM;
        double medju;
        try{
            fwM=new FileWriter(FxmlFiles.sFileError);
            bwM=new BufferedWriter(fwM);
            for (int iW=0;iW<iCounter;iW++){
```

```
            errx=(100*cos(Math.toRadians(25))*cos(Math.toRadians(Double.parseDouble(linesMj2[iW
][1]))));
            erry=(cos(Math.toRadians(-(Double.parseDouble(linesMj2[iW][0])-
25)))*100*sin(Math.toRadians(25)))+(-sin(Math.toRadians(-(Double.parseDouble(linesMj2[iW][0])-
25)))*100*cos(Math.toRadians(25))*sin(Math.toRadians(Double.parseDouble(linesMj2[iW][1]))));
            errz=(sin(Math.toRadians(-(Double.parseDouble(linesMj2[iW][0])-
25)))*100*sin(Math.toRadians(25)))+(cos(Math.toRadians(-(Double.parseDouble(linesMj2[iW][0])-
25)))*100*cos(Math.toRadians(25))*sin(Math.toRadians(Double.parseDouble(linesMj2[iW][1]))));


            for(int iL=0;iL<5;iL++){
               switch(iL){
                  case 0:
                     bwM.write(linesMj2[iW][iL]);
                     bwM.write(",");
                     break;
                  case 1:
                     bwM.write(linesMj2[iW][iL]);
                     bwM.write(",");
                     break;
                  case 2:
                     medju=errx-Double.parseDouble(linesMj2[iW][iL]);
                     bwM.write(df.format(medju));
                     bwM.write(",");
                     break;
                  case 3:
                     medju=erry-Double.parseDouble(linesMj2[iW][iL]);
                     bwM.write(df.format(medju));
                     bwM.write(",");
                     break;
                  case 4:
                     medju=errz-Double.parseDouble(linesMj2[iW][iL]);
                     bwM.write(df.format(medju));
                     bwM.newLine();
                     break;
                  default:
                     break;
               }
            }
         }
         bwM.close();
      }catch(IOException e){
      }
   }

   private void ErrInter(){
      iCounter=0;
      BufferedReader brM;
      FileReader frM;
      try{
         frM=new FileReader(FxmlFiles.sFileError);
```

```
            brM=new BufferedReader(frM);
            String sTrenutnaLinija;
            while ((sTrenutnaLinija=brM.readLine())!=null){
               linesMj1[iCounter]= sTrenutnaLinija;
               String [] lol=linesMj1[iCounter].split(",");
               for (jCounter=0;jCounter<5;jCounter++){
                  linesMj3[iCounter][jCounter]=lol[jCounter];
               }
               iCounter=iCounter+1;
            }
         }catch(IOException e){
         }
      }

      private void analizaRacun(){
         df1.setMaximumFractionDigits(5);
         df.setMaximumFractionDigits(4);
         dInter=interpol(x_R1,x_R2);
         x_meas=dInter[0];
         y_meas=dInter[1];
         z_meas=dInter[2];
         /*x_meas=Double.parseDouble(txt_fieldX.getText());
         y_meas=Double.parseDouble(txt_fieldY.getText());
         z_meas=Double.parseDouble(txt_fieldZ.getText());*/
         txt_fieldX.setText(df.format(x_meas).trim());
         txt_fieldY.setText(df.format(y_meas).trim());
         txt_fieldZ.setText(df.format(z_meas).trim());
         povecanje_kuta(0, 1);
         povecanje_kuta(0, 2);
         lbl_Xkoord_anal.setText("X:" + racunX(x_R1, x_R2) + Texts.smm);
         lbl_Ykoord_anal.setText("Y:" + racunY(x_R1, x_R2) + Texts.smm);
         lbl_Zkoord_anal.setText("Z:" + racunZ(x_R1, x_R2) + Texts.smm);
         lbl_Xkoord_err.setText("X:" + racunErr(x_tar, x_meas) + Texts.smm);
         lbl_Ykoord_err.setText("Y:" + racunErr(y_tar, y_meas) + Texts.smm);
         lbl_Zkoord_err.setText("Z:" + racunErr(z_tar, z_meas) + Texts.smm);
         lbl_Xkoord_ntar.setText("X:" + racunNtar(x_tar, x_meas) + Texts.smm);
         lbl_Ykoord_ntar.setText("Y:" + racunNtar(y_tar, y_meas) + Texts.smm);
         lbl_Zkoord_ntar.setText("Z:" + racunNtar(z_tar, z_meas) + Texts.smm);
         n_tar_z = Double.parseDouble(racunNtar(z_tar, z_meas));
         n_tar_x = Double.parseDouble(racunNtar(x_tar, x_meas));
         n_tar_y = Double.parseDouble(racunNtar(y_tar, y_meas));
         r_2novi=(n_tar_x/(100*cos(Math.toRadians(25))));
         r_2novi=Math.toDegrees(acos((r_2novi)));
         r2_ntar = df.format(r_2novi);

         r_1novi=Math.toDegrees(Math.atan2(-n_tar_z,n_tar_y));

         r1_ntar = df.format(r_1novi);
         lbl_R1_ntar.setText("R1:" + r1_ntar + "°");
         lbl_R2_ntar.setText("R2:" + r2_ntar + "°");
      }
```

```
    private double[] interpol(double dxR1, double dxR2){
        double x_int,y_int,z_int;
        double x_11=0,y_11=0,z_11=0;
        double x_12=0,y_12=0,z_12=0;
        double x_21=0,y_21=0,z_21=0;
        double x_22=0,y_22=0,z_22=0;
        double xy1, xy2;

        double R11=0,R12=0,R21=0,R22=0;
        for(int iInt=0;iInt<iCounter;iInt++){
            R1_meas1=Double.parseDouble(linesMj2[iInt][0].trim());
            R2_meas1=Double.parseDouble(linesMj2[iInt][1].trim());
            if (((R1_meas1-dxR1)>-5)&&((R1_meas1-dxR1)<=5)){
                if(R1_meas1>dxR1){
                    R12=R1_meas1;
                    R11=R12-10;
                }
                if(R1_meas1<=dxR1){
                    R11=R1_meas1;
                    R12=R11+10;
                }
            }
            if (((R2_meas1-dxR2)>-5)&&((R2_meas1-dxR2)<=5)){
                if(R2_meas1>=dxR2){
                    R22=R2_meas1;
                    R21=R22-10;
                }
                if(R2_meas1<=dxR2){
                    R21=R2_meas1;
                    R22=R21+10;
                }
            }
        }
        for(int iInt1=0;iInt1<iCounter;iInt1++){
            if((R11==Double.parseDouble(linesMj2[iInt1][0].trim()))&&(R21==Double.parseDouble(line
sMj2[iInt1][1].trim()))){
                x_11=Double.parseDouble(linesMj2[iInt1][2]);
                y_11=Double.parseDouble(linesMj2[iInt1][3]);
                z_11=Double.parseDouble(linesMj2[iInt1][4]);
            }
            if((R12==Double.parseDouble(linesMj2[iInt1][0].trim()))&&(R21==Double.parseDouble(line
sMj2[iInt1][1].trim()))){
                x_12=Double.parseDouble(linesMj2[iInt1][2]);
                y_12=Double.parseDouble(linesMj2[iInt1][3]);
                z_12=Double.parseDouble(linesMj2[iInt1][4]);
            }
            if((R11==Double.parseDouble(linesMj2[iInt1][0].trim()))&&(R22==Double.parseDouble(line
sMj2[iInt1][1].trim()))){
                x_21=Double.parseDouble(linesMj2[iInt1][2]);
                y_21=Double.parseDouble(linesMj2[iInt1][3]);
```

```
                z_21=Double.parseDouble(linesMj2[iInt1][4]);
            }
           if((R12==Double.parseDouble(linesMj2[iInt1][0].trim()))&&(R22==Double.parseDouble(line
sMj2[iInt1][1].trim()))){
                x_22=Double.parseDouble(linesMj2[iInt1][2]);
                y_22=Double.parseDouble(linesMj2[iInt1][3]);
                z_22=Double.parseDouble(linesMj2[iInt1][4]);
            }
        }
        xy1=(((R12-dxR1)/(R12-R11))*x_11)+(((dxR1-R11)/(R12-R11))*x_12);
        xy2=(((R12-dxR1)/(R12-R11))*x_21)+(((dxR1-R11)/(R12-R11))*x_22);
        x_int=(((R22-dxR2)/(R22-R21))*xy1)+(((dxR2-R21)/(R22-R21))*xy2);
        xy1=(((R12-dxR1)/(R12-R11))*y_11)+(((dxR1-R11)/(R12-R11))*y_12);
        xy2=(((R12-dxR1)/(R12-R11))*y_21)+(((dxR1-R11)/(R12-R11))*y_22);
        y_int=(((R22-dxR2)/(R22-R21))*xy1)+(((dxR2-R21)/(R22-R21))*xy2);
        xy1=(((R12-dxR1)/(R12-R11))*z_11)+(((dxR1-R11)/(R12-R11))*z_12);
        xy2=(((R12-dxR1)/(R12-R11))*z_21)+(((dxR1-R11)/(R12-R11))*z_22);
        z_int=(((R22-dxR2)/(R22-R21))*xy1)+(((dxR2-R21)/(R22-R21))*xy2);
        return(new double[]{x_int,y_int,z_int});
    }



    @FXML
    public void handleImport(ActionEvent event) {
        analizaRacun();
    }

    @FXML
    private void handleResR1(ActionEvent event) throws IOException, InterruptedException {
        stringSlanje(Komunikacija.R1, Komunikacija.sRes1, "");
        dissR1(btn_disable);
    }

    @FXML
    private void handleResR2(ActionEvent event) throws IOException, InterruptedException {
        stringSlanje(Komunikacija.R1, Komunikacija.sRes2, "");
        dissR2(btn2_disable);
    }
}
```

1.3 Xform.java

```
package nerov0.pkg1;
import javafx.scene.Group;
import javafx.scene.transform.Rotate;
import javafx.scene.transform.Scale;
import javafx.scene.transform.Translate;
```

```
public class Xform extends Group {

    public enum RotateOrder {
        XYZ, XZY, YXZ, YZX, ZXY, ZYX
    }

    public Translate t  = new Translate();
    public Translate p = new Translate();
    public Translate ip = new Translate();
    public Rotate rx = new Rotate();
    { rx.setAxis(Rotate.X_AXIS); }
    public Rotate ry = new Rotate();
    { ry.setAxis(Rotate.Y_AXIS); }
    public Rotate rz = new Rotate();
    { rz.setAxis(Rotate.Z_AXIS); }
    public Scale s = new Scale();

    public Xform() {
        super();
        getTransforms().addAll(t, rz, ry, rx, s);
    }

    public Xform(RotateOrder rotateOrder) {
        super();
        switch (rotateOrder) {
        case XYZ:
            getTransforms().addAll(t, p, rz, ry, rx, s, ip);
            break;
        case YXZ:
            getTransforms().addAll(t, p, rz, rx, ry, s, ip);
            break;
        case YZX:
            getTransforms().addAll(t, p, rx, rz, ry, s, ip);
            break;
        case ZXY:
            getTransforms().addAll(t, p, ry, rx, rz, s, ip);
            break;
        case ZYX:
            getTransforms().addAll(t, p, rx, ry, rz, s, ip);
            break;
        }
    }

    public void setTranslate(double x, double y, double z) {
        t.setX(x);
        t.setY(y);
        t.setZ(z);
    }

    public void setTranslate(double x, double y) {
        t.setX(x);
```

```
        t.setY(y);
    }

    public void setTx(double x) { t.setX(x); }
    public void setTy(double y) { t.setY(y); }
    public void setTz(double z) { t.setZ(z); }

    public void setRotate(double x, double y, double z) {
        rx.setAngle(x);
        ry.setAngle(y);
        rz.setAngle(z);
    }

    public void setRotateX(double x) { rx.setAngle(x); }
    public void setRotateY(double y) { ry.setAngle(y); }
    public void setRotateZ(double z) { rz.setAngle(z); }
    public void setRy(double y) { ry.setAngle(y); }
    public void setRz(double z) { rz.setAngle(z); }

    public void setScale(double scaleFactor) {
        s.setX(scaleFactor);
        s.setY(scaleFactor);
        s.setZ(scaleFactor);
    }

    public void setSx(double x) { s.setX(x); }
    public void setSy(double y) { s.setY(y); }
    public void setSz(double z) { s.setZ(z); }

    public void setPivot(double x, double y, double z) {
        p.setX(x);
        p.setY(y);
        p.setZ(z);
        ip.setX(-x);
        ip.setY(-y);
        ip.setZ(-z);
    }

    public void reset() {
        t.setX(0.0);
        t.setY(0.0);
        t.setZ(0.0);
        rx.setAngle(0.0);
        ry.setAngle(0.0);
        rz.setAngle(0.0);
        s.setX(1.0);
        s.setY(1.0);
        s.setZ(1.0);
        p.setX(0.0);
        p.setY(0.0);
        p.setZ(0.0);
```

```
        ip.setX(0.0);
        ip.setY(0.0);
        ip.setZ(0.0);
    }

    public void resetTSP() {
        t.setX(0.0);
        t.setY(0.0);
        t.setZ(0.0);
        s.setX(1.0);
        s.setY(1.0);
        s.setZ(1.0);
        p.setX(0.0);
        p.setY(0.0);
        p.setZ(0.0);
        ip.setX(0.0);
        ip.setY(0.0);
        ip.setZ(0.0);
    }

    public void debug() {
        System.out.println("t = (" +
                           t.getX() + ", " +
                           t.getY() + ", " +
                           t.getZ() + ")  " +
                           "r = (" +
                           rx.getAngle() + ", " +
                           ry.getAngle() + ", " +
                           rz.getAngle() + ")  " +
                           "s = (" +
                           s.getX() + ", " +
                           s.getY() + ", " +
                           s.getZ() + ")  " +
                           "p = (" +
                           p.getX() + ", " +
                           p.getY() + ", " +
                           p.getZ() + ")  " +
                           "ip = (" +
                           ip.getX() + ", " +
                           ip.getY() + ", " +
                           ip.getZ() + ")");
    }
}
```

1.4 Unos_IP.java

```
package nerov0.pkg1;

import javafx.event.ActionEvent;
```

```
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.Initializable;
import javafx.stage.Stage;

/**
*
* @author Filipovic novi
* Klasa za unos IP adrese mikrokontrolera na koju se spaja aplikacija
*/
public class unos_IP implements Initializable {
    static String unos_Ip="192.168.1.3";
    Stage stageProzor=null;
    //Controler_prozor1 Cont_proz=new Controler_prozor1();

    @FXML
    private TextField txt_Field_IP;
    @FXML
    private Button btn_OK;
    @FXML
    private Label lbl_IP=new Label(unos_Ip);

    @FXML
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        this.lbl_IP.setText(Texts.sUnosIP);
    }

    @FXML
    private void handleBtnOk(ActionEvent event) throws IOException{
      unos_Ip=txt_Field_IP.getText();
      this.lbl_IP.setText(unos_Ip);
      Controler_prozor1.unos_IP=unos_Ip;
      stageProzor=(Stage) btn_OK.getScene().getWindow();
      stageProzor.hide();
    }

}
```

1.5 FxmlFiles.java

```
package nerov0.pkg1;

public class FxmlFiles {
```

```
    static String sMain="prozor1.fxml";
    static String sUnos="prozor_unos.fxml";
    static String sUnIP="prozor_IP.fxml";
    static String sAbo ="about.fxml";

    static String sIkona="3_mST_1_16.png";

    static String sFile="in.txt";
    static String sFilePodatci="faro.txt";
    static String sFileError="error.txt";

    static String sLoad="loading.fxml";
}
```

1.6 Komunikacija

```
package nerov0.pkg1;

/* Klasa koja sadrži kodove osi i naredbi za slanje na mikrokontroler
*/
public class Komunikacija {
    static String R1="1";
    static String R2="2";
    static String T1="3";
    static String T2="4";
    static String T3="5";

    static String sR1="R1";
    static String sR2="R2";
    static String sCurrent1="C1";
    static String sCurrent2="C2";
    static String sError1="E1";
    static String sError2="E2";

    static String D9="9";
    static String sZero="000";
    static String sMove="111";
    static String sDiss="211";
    static String sEnab="212";

    static String sRes1="213";
    static String sRes2="214";


}
```

1.7 Texts.java

```
package nerov0.pkg1;

/**
*
* @author Filipovic novi
* Klasa strignova korištenih u programu eksperimentalnoj verziji NERO-a
*/
public class Texts {
	static String Language="HRV";
	static String sMainTitle="NERO v0.1";
	static String sAddTitleIN="-Input";
	static String sAddTitleIP="-IP";
	static String sAddTitleAB="-About";
	static String sAddTitleAN="-Analysis";
	static String sToX="to X°";
	static String sHOME="HOME(0°)";
	static String sHomeAll="Home All";
	static String sSetV="Set values";
	static String sLoaV="Load values";
	static String sToV="To values";
	static String sIP="IP";
	static String sCls="Close";
	static String sAb="About";
	static String sLang="Language: ENG";
	static String sConMsg="Connected to :";
	static String sNoCon="No connection to "+unos_IP.unos_Ip;
	static String sConn="Connect";
	static String sDiss="Disconnect";
	static String sMov="Moving...";
	static String sConAll="Connect all";
	static String sDissAll="Disconnect all";
	static String sEna="Enabled";
	static String sDisa="Disabled";
	static String sPos="Position display";
	static String sAna="Error analysis";
	static String sRot1="R1: ";
	static String sRot2="R2: ";
	static String sR1="R1[°]=";
	static String sR2="R2[°]=";
	static String sT1="T1[mm]=";
	static String sT2="T2[mm]=";
	static String sT3="T3[mm]=";
	static String smm="mm";
	static String sFile="File";
	static String sEdit="Edit";
	static String sHelp="Help";
	static String sUnosIP="Enter manipulator IP";
	static String sImport="Import R1 & R2";
```

```
    static String sLblIn="Input";
    static String sLblVa="Taken values";
    static String sBtnSave="From input";
    static String sBtnSave1="From analysis";
    static String sBtnFile="From file";
    static String sNero3D="Nero 3D view";
    static String text="Controler application for experimental setup of NERO - neurosurgical "
                + "robot. Application is used for axes control build in this version "
                + "of NERO. Current axes are Rotation 1 (R1) and Rotation 2 (R2), commonly "
                + "marked as \u03B8 and \u03C1 in spherical coordinate system notation. "
                + "\n"
                + "Application has ability to move asxes, analyse precision and correction"
                + " of position error using methods from MARSes PhD thesis. One of options is"
                + "to use premade values for axes loaded from .txt file. "
                + "\n"
                + "It is connected with PMAS controler via EtherNet. Commonly used IP address"
                + "for PMAS is 192.168.1.3 and used port is 5005. Drives are connected\n"
                + "via EtherCat.\n"
                + "This is version: "+sMainTitle;
    static String sConTitle="Exit?";
    static String sConCont="Are you sure you want to exit?";
    static String sConHead="Exit confirmation";
    static String sError="Error";
    static String sHeadError="Something went wrong!!";
    static String sFileError="Error ocured while loading from File";
    static String sErrorRec="Error while reciving from "+unos_IP.unos_Ip;
    static String sPower="Power";
    static String sHeadPowerOff="Power turned off";
    static String sHeadPowerOn="Power turned on";
    static String sPowerRecOff="Power turned off: ";
    static String sPowerRecOn="Power turned on: ";


  public static void jezik(){
    if ("ENG".equals(Language)){
      System.out.println(Language);
      sMainTitle="NERO v0.1";
      sAddTitleIN="-Input";
      sAddTitleIP="-IP";
      sAddTitleAB="-About";
      sAddTitleAN="-Analysis";
      sToX="to X°";
      sHOME="HOME(0°)";
      sHomeAll="Home All";
      sSetV="Set values";
      sLoaV="Load values";
      sToV="To values";
      sIP="IP";
      sCls="Close";
      sAb="About";
      sLang="Language: ENG";
```

```
sConMsg="Connected to :";
sNoCon="No connection "+unos_IP.unos_Ip;
sConn="Connect";
sDiss="Disconnect";
sMov="Moving...";
sConAll="Connect all";
sDissAll="Disconnect all";
sEna="Enabled";
sDisa="Disabled";
sPos="Position display";
sAna="Error analysis";
sRot1="R1: ";
sRot2="R2: ";
sR1="R1[°]=";
sR2="R2[°]=";
sT1="T1[mm]=";
sT2="T2[mm]=";
sT3="T3[mm]=";
smm="mm";
sFile="File";
sEdit="Edit";
sHelp="Help";
sUnosIP="Enter manipulator IP";
sImport="Import R1 & R2";
sLblIn="Input";
sLblVa="Taken values";
sBtnSave="From input";
sBtnSave1="From analysis";
sBtnFile="From file";
sNero3D="Nero 3D view";
text="Controler application for experimental setup of NERO - neurosurgical "
  + "robot. Application is used for control of axes build in this version "
  + "of NERO. Current axes are Rotation 1 (R1) and Rotation 2 (R2), commonly "
  + "marked as \u03B8 and \u03C1 in spherical coordinate system notation. "
  + "\n"
  + "Application has ability to move asxes, analyse precision and correction"
  + "of position error using methods from MARS PhD thesis. One of options is"
  + "to use premade values for axes loaded from .txt file. "
  + "\n"
  + "It is connected with PMAS controler via EtherNet. Commonly used IP address"
  + "for PMAS is 192.168.1.3 and used port is 5005. Drives are connected\n"
  + "via EtherCat.\n"
  + "This is version: "+sMainTitle;
sConTitle="Exit?";
sConCont="Are you sure you want to exit?";
sConHead="Exit confirmation";
sError="Error";
sHeadError="Something went wrong!!";
sFileError="Error ocured while loading from File";
sErrorRec="Error while reciving from "+unos_IP.unos_Ip;
Language="HRV";
```

```
}else if("HRV".equals(Language)){
   System.out.println(Language);
   sMainTitle="NERO v0.1";
   sAddTitleIN="-Unos";
   sAddTitleIP="-IP";
   sAddTitleAB="-O programu";
   sAddTitleAN="-Analiza";
   sToX="do X°";
   sHOME="Početna(0°)";
   sHomeAll="Sve u početnu";
   sSetV="Postavi";
   sLoaV="Učitaj";
   sToV="Vozi u";
   sIP="IP";
   sCls="Zatvori";
   sAb="O programu";
   sLang="Jezik: HRV";
   sConMsg="Spojeno na :";
   sNoCon="Nema veze s "+unos_IP.unos_Ip;
   sConn="Spoji";
   sDiss="Prekini";
   sMov="Kretanje...";
   sConAll="Spoji sve";
   sDissAll="Prekini sve";
   sEna="Omogućeno";
   sDisa="Onemogućeno";
   sPos="Prikaz pozicije";
   sAna="Analiza greške";
   sRot1="R1: ";
   sRot2="R2: ";
   sR1="R1[°]=";
   sR2="R2[°]=";
   sT1="T1[mm]=";
   sT2="T2[mm]=";
   sT3="T3[mm]=";
   smm="mm";
   sFile="Datoteka";
   sEdit="Izmjeni";
   sHelp="Pomoć";
   sUnosIP="Unesi IP manipulatora";
   sImport="Uvezi R1 & R2";
   sLblIn="Unos";
   sLblVa="Unešene vrijednosti";
   sBtnSave="Iz unosa";
   sBtnSave1="Iz analize";
   sBtnFile="Iz datoteke";
   sNero3D="Nero 3D pogled";
   text="Aplikacija za kontrolu eksperimentalnog postava NERO - Neurokirurški "
     + "robot. Aplikacija se koristi za kontrolu osi ugrađenih u ovu verziju "
     + "NERO-a. Trenutne osi su u Rotacija 1 (R1) i Rotacija 2 (R2), obično"
     + "označeni sa \u03B8 i \u03C1 u nazivlju korištenom za sferni koordinatni"
```

```
            + " sustav."
            + "\n"
            + "Aplikacija ima opcije micanja osi, analize preciznosti i korekciju "
            + "greške pozicioniranja koristeč metode uzete iz MARS-ovog doktorata. "
            + "Još jedna od opcija je korištenje predefiniranih vrijednosti za osi "
            + "povučene iz .txt filea."
            + "\n"
            + "Spajanje na PMAS se odvija preko EtherNeta. Često korištena IP adresa "
            + "za PMAS je 192.168.1.3 a korišteni port je 5005. Driveovi su spojeni "
            + "preko EtherCata.\n"
            + "Trenutna verzija: "+sMainTitle;
        sConTitle="Zatvori?";
        sConCont="Želiš li sigurno ugasiti program?";
        sConHead="Potvrda gašenja";
        sError="Error";
        sHeadError="Nešto nije u redu!!";
        sFileError="Greška pri čitanju datoteke";
        sErrorRec="Greška u komunikaciji s "+unos_IP.unos_Ip;
        Language="ENG";
    }
  }
}
```

1.8 Controler_unos.java

```
package nerov0.pkg1;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.net.URL;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.Arrays;
import java.util.Locale;
import java.util.ResourceBundle;
import java.util.Scanner;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
/**
```

```
*
* @author Filipovic novi
* Klasa prozora za unos vrijednosti pozicija osi
*/
public class Controler_unos implements Initializable{
    static String unos_R1,unos_R2,unos_T1,unos_T2,unos_T3;
    static String[] lines=new String[5];
    int i;
    Stage stageProzor=null;
    //Controler_prozor1 prijenos=new Controler_prozor1();
    public DecimalFormat df
= new DecimalFormat("0.####", DecimalFormatSymbols.getInstance(Locale.ENGLISH));

    @FXML
    private Button btn_save,btn_save1, btn_ok, btn_from_file;
    @FXML
    private TextField lbl_R1,lbl_R2,lbl_T1,lbl_T2,lbl_T3;
    @FXML
    private Label lbl_R1_2,lbl_R2_2,lbl_T1_2,lbl_T2_2,lbl_T3_2,lbl_input,lbl_value;

    @FXML
    private void handleBtnOk(ActionEvent event){
        stageProzor=(Stage) btn_ok.getScene().getWindow();
        stageProzor.close();
    }

    @FXML
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        lbl_input.setText(Texts.sLblIn);
        lbl_value.setText(Texts.sLblVa);
        btn_save.setText(Texts.sBtnSave);
        btn_save1.setText(Texts.sBtnSave1);
        btn_from_file.setText(Texts.sBtnFile);
    }

    @FXML
    private void handleBtnSave(ActionEvent event){
        unos_R1=lbl_R1.getText();
        unos_R2=lbl_R2.getText();
        unos_T1=lbl_T1.getText();
        unos_T2=lbl_T2.getText();
        unos_T3=lbl_T3.getText();
        lbl_R1_2.setText(Texts.sR1+unos_R1);
        lbl_R2_2.setText(Texts.sR2+unos_R2);
        lbl_T1_2.setText(Texts.sT1+unos_T1);
        lbl_T2_2.setText(Texts.sT2+unos_T2);
        lbl_T3_2.setText(Texts.sT3+unos_T3);
    }

    @FXML
```

```
    private void hadnelFromAnal(ActionEvent event){
        unos_R1=df.format(Controler_prozor1.r_1novi);
        unos_R2=df.format(Controler_prozor1.r_2novi);
        lbl_R1_2.setText(Texts.sR1+unos_R1);
        lbl_R2_2.setText(Texts.sR2+unos_R2);
        lbl_T1_2.setText(Texts.sT1+unos_T1);
        lbl_T2_2.setText(Texts.sT2+unos_T2);
        lbl_T3_2.setText(Texts.sT3+unos_T3);
    }

    @FXML
    private void handleFromFile(ActionEvent event) throws FileNotFoundException, IOException{
        i=0;
        BufferedReader br=null;
        FileReader fr=null;
        try{
            fr=new FileReader(FxmlFiles.sFile);
            br=new BufferedReader(fr);
            String sTrenutnaLinija;
            while ((sTrenutnaLinija=br.readLine())!=null){
                lines[i]= sTrenutnaLinija;
                i=i+1;
            }
        }catch(IOException e){
        }
        unos_R1=lines[0];
        unos_R2=lines[1];
        unos_T1=lines[2];
        unos_T2=lines[3];
        unos_T3=lines[4];
        lbl_R1_2.setText(Texts.sR1+unos_R1);
        lbl_R2_2.setText(Texts.sR2+unos_R2);
        lbl_T1_2.setText(Texts.sT1+unos_T1);
        lbl_T2_2.setText(Texts.sT2+unos_T2);
        lbl_T3_2.setText(Texts.sT3+unos_T3);
    }
}
```

1.9 Menu.java

```
package nerov0.pkg1;

import java.io.IOException;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
```

```
/**
*
* @author Filipovic novi
*/
public class menu {

    FXMLLoader cetvrtiProzor=null;
    AnchorPane rootCetvrtiProzor=null;
    Scene scena4;
    Stage unos_stage4;
    Scene scena3;
    Stage unos_stage3;


    public void menuIP2(){
          Parent root;
      try{
          FXMLLoader treciProzor = new FXMLLoader();
          treciProzor.setLocation(Controler_prozor1.class.getResource(FxmlFiles.sUnIP));
          AnchorPane rootTreciProzor = (AnchorPane) treciProzor.load();
          scena3 = new Scene(rootTreciProzor);
          unos_stage3 = new Stage();
          unos_stage3.setScene(scena3);
          unos_stage3.setTitle(Texts.sMainTitle+Texts.sAddTitleIP);
          unos_stage3.setResizable(false);
          unos_stage3.setScene(scena3);
          unos_stage3.initStyle(StageStyle.UTILITY);
          unos_stage3.show();
       }catch(IOException e){
       }
    }

    public void menuAbout(){
            Parent root;
      try{
          cetvrtiProzor=new FXMLLoader();
          cetvrtiProzor.setLocation(Controler_prozor1.class.getResource(FxmlFiles.sAbo));
          rootCetvrtiProzor=(AnchorPane) cetvrtiProzor.load();
          scena4=new Scene(rootCetvrtiProzor);
          unos_stage4=new Stage();
          unos_stage4.setScene(scena4);
          unos_stage4.setTitle(Texts.sMainTitle+Texts.sAddTitleAB);
          unos_stage4.setResizable(false);
          unos_stage4.setScene(scena4);
          unos_stage4.initStyle(StageStyle.UTILITY);
          unos_stage4.show();
      }
       catch(IOException e){
       }
```

```
    }
}
```

1.10 Slike.java

```
package nerov0.pkg1;

public class Slike {
    static String sIco16="ikona_16.png";
    static String sIco32="ikona_32.png";
    static String sIco64="ikona_64.png";
    static String sIco128="ikona_128.png";
    static String sIco256="ikona_256.png";
    static String sIco512="ikona_512.png";
    static String sIco514="ikona_514.png";
    static String sIco64_1="ikona_64_1.png";

    static String sRed="2000px-Red_Light_Icon.png";
    static String sGrn="2000px-Green_Light_Icon.png";
}
```

2 PMAS

2.1 Main.cpp

```
#include <iostream>
#include "mmcpplib.h"
#include <string.h>
#include "Main.h"
#include <sstream>
#include "mmc_definitions.h"
#include <signal.h>
#include "CallBack.h"
#include <thread>


#define SSTR( x ) static_cast< std::ostringstream & >( \
      ( std::ostringstream() << std::dec << x ) ).str()


using namespace std;


#define DEMO_ADDR "192.168.1.3"
#define DEMO_PORT 5005

typedef enum {eSOCKET_CREATE_STATE=0, eSOCKET_ACCEPT_STATE,
            eSOCKET_READ_STATE}EDEMO_SOCK_STATE;
typedef enum {eSOCKET_CONNECT_STATE,
                        eSOCKET_WRITABLE_STATE, eSOCKET_SEND_STATE
}EDEMO2_SOCK_STATE;
typedef struct _sockdemo_msg_t{
            char _iID;
            char _cCom[3];
            char _szUserData[128];
}dsock_msg_t;

typedef struct _sockclient_msg_t{
            //char _iID2;
            char _sUserData2[256];

}dsock2_msg_t;

static int iSock2Out;
static CMMCTCP tcpClient;
bool EmgRec=false;


void thread1(){
            MainInit();
            iCon1=true;
            iCon2=true;
            while(true){
                        if(iCon==true){
                                    ReadAllInputData();
                        }
```

```
		TCPDemoServer();
		sleep(0.005);
		}
}
void thread2(){
		while(true){
			vPrimanje('R','1', giPos1);
			sleep(0.1);

			vPrimanje('R','2', giPos2);
			sleep(0.1);

			vPrimanje('C','1', giCur1);
			sleep(0.1);

			vPrimanje('C','2', giCur2);
			sleep(0.1);
		}
}

int main(){
		std::thread first(thread1);
		std::thread second(thread2);
		while(1){

		}
		return 0;
}

int TCPDemoClient(char sOs, char sOs1,double sPoruka){
		static EDEMO2_SOCK_STATE eState2=eSOCKET_CONNECT_STATE;
		int rc2=-1;
		bool bWait;
		static dsock2_msg_t msg2;
		static int iSock2;

		switch(eState2){
		case eSOCKET_CONNECT_STATE:
			rc2=tcpClient.Connect("192.168.1.7",g_usPort2,iSock2,bWait);
			cout<<__func__<<rc2<<"\n";
			//sleep(0.1);
			if (rc2>=0){
				eState2=eSOCKET_WRITABLE_STATE;
			}
			break;
		case eSOCKET_WRITABLE_STATE:
			if (tcpClient.IsWritable(iSock2)){
				eState2=eSOCKET_SEND_STATE;
			}
			break;

		case eSOCKET_SEND_STATE:
			memset(&msg2,0,sizeof(dsock2_msg_t));
			sprintf(msg2._sUserData2,"%c%c%f\n",sOs,sOs1,sPoruka);
			rc2=tcpClient.Send(iSock2,sizeof(dsock2_msg_t),(void *)&msg2);
```

```
                        //sleep(0.1);
                        if(rc2>0){
                                bSlanje=false;
                                eState2=eSOCKET_CONNECT_STATE;
                                return(iSock2);
                        }
                        eState2=eSOCKET_CONNECT_STATE;
                        break;
                }
                return(0);
}

void TCPDemoServer(){
                static EDEMO_SOCK_STATE eState= eSOCKET_CREATE_STATE;
                static int rc=-1;
                static int iSock;
                static dsock_msg_t msg;
                static CMMCTCP tcpServer;

                switch(eState){
                case eSOCKET_CREATE_STATE:
                        rc=tcpServer.Create(g_usPort);
                        eState= eSOCKET_ACCEPT_STATE;
                        break;
                case eSOCKET_ACCEPT_STATE:{
                        if ((rc=tcpServer.Accept())>0){
                                iSock=rc;
                                break;
                        }
                        eState=eSOCKET_READ_STATE;
                        //break;
                }
                case eSOCKET_READ_STATE:{
                        memset(&msg, 0 , sizeof(dsock_msg_t));
                        rc=tcpServer.Receive(iSock, sizeof(dsock_msg_t),(void*)&msg);
                        std::string s2;
                        s2.append(reinterpret_cast<const char *>(msg._cCom), 3);
                        if(rc>0){
                                std::string s3 = SSTR( msg._iID );
                                std::string s1 = SSTR( msg._cCom );
                                std::string s = SSTR( msg._szUserData );
                                fID=atof(s3.c_str());
                                fCom=atof(s2.c_str());
                                fPoruka=atof(s.c_str());
                                dPozicija=fPoruka;
                                //giState1=eIDLE;
                                for(int i=0;i<3;i++){
                                        MachineSequences(fID);
                                }
                                break;
                        }
                        eState=eSOCKET_ACCEPT_STATE;
                        break;
```

```
                    }
            }
}
void MainInit(){
            gConnHndl = cConn.ConnectIPCEx(0x7fffffff,NULL) ;
            GetGMASOperationMode (gConnHndl);
            SetGMASOperationMode (gConnHndl, 0);
            iCon=true;
            cAxes1.InitAxisData("R01",gConnHndl);
            cAxes2.InitAxisData("R02",gConnHndl);
            // Register the callback function for Modbus and Emergency:
            cConn.RegisterEventCallback(MMCPP_EMCY,(void*)Emergency_Received) ;
            cConn.RegisterEventCallback(MMCPP_MOTIONENDED,(void*)Motion_Ended);
            //MMCPP_MOTIONENDED
            cConn.RegisterEventCallback(MMCPP_NODE_ERROR,(void*)Emergency_Received) ;
            //MMCPP_NODE_ERROR
            cAxes1.EnableMotionEndedEvent();
            cAxes2.EnableMotionEndedEvent();
            MMC_INSNOTIFICATIONFB_IN pInParam;
            MMC_INSNOTIFICATIONFB_OUT pOutParam;
            pInParam.iEventCode = 1;
            //cAxes1.reset();
            //cAxes2.reset();
            aAxes1=0;
            aAxes2=1;
            MMC_InsertNotificationFb(gConnHndl,aAxes1,&pInParam,&pOutParam);
            MMC_InsertNotificationFb(gConnHndl,aAxes2,&pInParam,&pOutParam);


            /* Register Run Time Error Callback function    */
            CMMCPPGlobal::Instance()->RegisterRTE(OnRunTimeError);

            //cGroup.InitAxisData("v01",gConnHndl);

/*          cGroup.
*/
            stSingleDefault.fEndVelocity  = 0 ;
            stSingleDefault.dbDistance              = 100000 ;
            stSingleDefault.dbPosition              = 0 ;
            stSingleDefault.fVelocity               = 1000000;
            stSingleDefault.fAcceleration  = 5000000;
            stSingleDefault.fDeceleration  = 5000000 ;
            stSingleDefault.fJerk                           = 10000000 ;
            stSingleDefault.eDirection              = MC_POSITIVE_DIRECTION ;
            stSingleDefault.eBufferMode  = MC_BUFFERED_MODE ;
            stSingleDefault.ucExecute               = 1 ;

            stSingleDefault2.fEndVelocity = 0 ;
            stSingleDefault2.dbDistance             = 100000 ;
            stSingleDefault2.dbPosition             = 0 ;
            stSingleDefault2.fVelocity              = 10000 ;
            stSingleDefault2.fAcceleration          = 100000 ;
            stSingleDefault2.fDeceleration          = 100000 ;
            stSingleDefault2.fJerk                          = 20000000 ;
            stSingleDefault2.eDirection     = MC_POSITIVE_DIRECTION ;
```

```
		stSingleDefault2.eBufferMode		= MC_BUFFERED_MODE ;
		stSingleDefault2.ucExecute		= 1 ;



		cAxes1.SetDefaultParams(stSingleDefault);
		cAxes2.SetDefaultParams(stSingleDefault2);


		stGroupDefault.fVelocity =	5000.0;
		stGroupDefault.fAcceleration =	8000000.0;
		stGroupDefault.fDeceleration =	8000000.0;
		stGroupDefault.fJerk =	100000000.0;
		stGroupDefault.eCoordSystem = MC_MCS_COORD;
		stGroupDefault.eBufferMode = MC_BUFFERED_MODE;
//	stGroupDefault.eTransitionMode = MC_TM_CORNER_DEVIATION_MODE_PLN6;
		stGroupDefault.eTransitionMode = MC_TM_NONE_MODE;
		stGroupDefault.fTransitionParameter[0] = 2000.0;
		stGroupDefault.ucExecute = 1;
}

void MachineSequences(float fOs){

		if ((giStatus1 & NC_AXIS_ERROR_STOP_MASK)||(giStatus2 &
NC_AXIS_ERROR_STOP_MASK)){
			giState1= eIDLE;
		}
		switch (giState1){
			case eIDLE:{
				if ((!(giStatus1 & NC_AXIS_ERROR_STOP_MASK))||(!(giStatus2 &
NC_AXIS_ERROR_STOP_MASK))){
					giState1=eSM1;
				}
				if(EmgRec==true){
					giState1=eSM1;
					EmgRec=false;
				}
				break;
			}
			case eSM1:{
				if (giStatus1 & NC_AXIS_DISABLED_MASK){
					cAxes1.PowerOn();
				}
				if (giStatus2 & NC_AXIS_DISABLED_MASK){
					cAxes2.PowerOn();
				}
				if ((giStatus1 & NC_AXIS_STAND_STILL_MASK)||(giStatus2 &
NC_AXIS_STAND_STILL_MASK)){
					giState1=eSM2;
				}
				break;
			}
			case eSM2:{
				if (fID==1){
					if( giStatus1 & NC_AXIS_DISABLED_MASK){
```

```
                                        if((fCom==212)&&(iCon1==false)){
                                                cAxes1.PowerOn();
                                                iCon1=true;
                                        }
                                }
                                if (giStatus1 & NC_AXIS_STAND_STILL_MASK){
                                        if((fCom==211)&&(iCon1==true)){
                                                cAxes1.PowerOff();
                                                iCon1=false;
                                        }

        if((iCon1==1)&&(fCom==111)){//&&(dInput==1)
                                                cout<<__func__<<dPozicija<<"
Kretanje osi 1 "<<"Status: eSM2 INPUT: "<< dInput<<"\n";

        cAxes1.MoveAbsolute(dPozicija*dConverzijaR1 ,1000000 ,MC_ABORTING_MODE) ;
                                                giState1=eIDLE;
                                        }else{
                                        }
                                }
                                break;
                        }
                        if (fID==2){

                                if( giStatus2 & NC_AXIS_DISABLED_MASK){
                                        if((fCom==212)&&(iCon2==false)){
                                                cAxes2.PowerOn();
                                                iCon2=true;
                                                cout<<__func__<<" PowerOn2"<<
"\n";
                                                //sleep(0.001);
                                                }
                                }
                                if (giStatus2 & NC_AXIS_STAND_STILL_MASK){
                                                if((fCom==211)&&(iCon2==true)){
                                                        cAxes2.PowerOff();
                                                        iCon2=false;
                                                        cout<<__func__<<"
PowerOn2"<< "\n";
                                                        //sleep(0.001);
                                                }
                                                if((iCon2==true)&&(fCom==111)){

        cout<<__func__<<dPozicija<<" kretanje2"<<"eSM2"<<"\n";

        cAxes2.MoveAbsolute(dPozicija*dConverzijaR2 ,100000 ,MC_ABORTING_MODE) ;
                                                        giState1=eIDLE;
                                                }

                                                }
                                break;
                        }
                        if (fID==9){
                                if ((iCon1==false)&&(iCon2==false)){
                                        iCon=false;
```

```
                                        MMC_CloseConnection(gConnHndl);
                                }
                                if((fCom==211)&&(iCon==true)){
                                        cAxes1.PowerOff();
                                        cAxes2.PowerOff();
                                        MMC_CloseConnection(gConnHndl);
                                        iCon=false;
                                        iCon1=false;
                                        iCon2=false;
                                }else if((fCom==212)&&(iCon==false)){
                                        MainInit();
                                        /*cAxes1.Reset();
                                        cAxes2.Reset();*/
                                        cAxes1.PowerOn();
                                        cAxes2.PowerOn();
                                        iCon1=true;
                                        iCon2=true;
                                }
                                break;
                        }
                }

                case eSM3:{
                        if ((giStatus1 & NC_AXIS_STAND_STILL_MASK)||(giStatus2 &
NC_AXIS_STAND_STILL_MASK)){
                                        //cout<<__func__<<dPozicija<<" eSM3"<<"\n";
                                        giState1=eIDLE;
                        }
                        break;
                }
                case eSM7:{
                        if((fCom==213)&&(fID==1)){
                                //cAxes1.Reset();
                                resetAxis(gConnHndl,aAxes1);
                                cout<<__func__<<" eSM7"<<"\n";
                                giState1=eIDLE;
                                break;
                        }
                        if((fCom==214)&&(fID==2)){
                                        //cAxes2.Reset();
                                        resetAxis(gConnHndl,aAxes2);
                                        giState1=eIDLE;
                                        break;
                                }
                        }
                }
        }
void vPrimanje(char sOs, char sOs2,double dSend){
        while(bSlanje){
                        iSock2Out=TCPDemoClient(sOs,sOs2,dSend);
                        //sleep(0.1);
        }
        bSlanje=true;
        usleep(10000);
        //cout<<__func__<<iSock2Out<<endl;
```

```
		tcpClient.Close(iSock2Out);
		usleep(10000);
}

void resetAxis(MMC_CONNECT_HNDL hConn, MMC_AXIS_REF_HNDL iAxisRef){
		int rc;
		stReset_in.ucExecute=1;
		rc=MMC_Reset(hConn, iAxisRef, &stReset_in, &stReset_out);
		if(rc!=0){
			cout<<"Axis: "<<iAxisRef<<endl;
		}
}


void GetDigitalInput(MMC_CONNECT_HNDL hConn, MMC_AXIS_REF_HNDL iAxisRef){
		int rc;
		MMC_READDIGITALINPUT_IN	stReadDigInput_in;
		MMC_READDIGITALINPUT_OUT	stReadDigInput_out;

		stReadDigInput_in.iInputNumber	= 16;	//Selects the input depending on the drive
		stReadDigInput_in.ucEnable	= 1;	//Enabled

		rc = MMC_ReadDigitalInputCmd (hConn, iAxisRef, &stReadDigInput_in,
&stReadDigInput_out);
		dInput=stReadDigInput_out.ucValue;
}

void ReadAllInputData(){
		giStatus1  = cAxes1.ReadStatus() ;
		giPos1		= (int)cAxes1.GetActualPosition() ;
		giCur1		= cAxes1.GetActualTorque();
//		giErr1	= cAxes1.GetAxisError(gConnHndl);
		giStatus2	= giStatus1;
		giStatus2  = cAxes2.ReadStatus() ;
		giPos2		= (int)cAxes2.GetActualPosition() ;
		giCur2		= cAxes2.GetActualTorque();
}

void GetGMASOperationMode (MMC_CONNECT_HNDL hConn){
		int rc;
		MMC_GET_GMASOP_MODE_IN	stGetGMASOpMode_in;
		MMC_GET_GMASOP_MODE_OUT	stGetGMASOpMode_out;
		stGetGMASOpMode_in.ucDummy	= 1;	// Dummy input
		//

		rc = MMC_GetGMASOperationMode (hConn, &stGetGMASOpMode_out);

		if (rc != 0){
			//printf("MMC_GetGMASOperationMode Error\n");
		}
}

void SetGMASOperationMode (MMC_CONNECT_HNDL hConn, int mode){
		MMC_SET_GMAS_OP_OUT		pOutParamOP;
		MMC_SET_GMAS_PREOP_OUT		pOutParamPREOP;
```

```
		switch (mode){
			case 0:{
				MMC_ChangeToOperationMode(hConn, &pOutParamOP) ;
				break;
			}
			case 1:{
				MMC_ChangeToPreOPMode(hConn, &pOutParamPREOP) ;
				break;
			}
		}
}
```

2.2 Main.h

```
MMC_CONNECT_HNDL gConnHndl ;
CMMCConnection cConn ;
CMMCSingleAxis a1 ;
CMMCSingleAxis a2 ;
CMMCECATIO cECATIO ;
MMC_MOTIONPARAMS_SINGLE	stSingleDefault ;	// Single axis default data
MMC_MOTIONPARAMS_SINGLE	stSingleDefault2 ;	// Single axis default data
CMMCSingleAxis	cAxes1 ;
CMMCSingleAxis	cAxes2 ;
CMMCGroupAxis cGroup;
MMC_MOTIONPARAMS_GROUP	stGroupDefault;
MMC_SETKINTRANSFORM_IN	stKinDefault;
MMC_AXIS_REF_HNDL aAxes1;
MMC_AXIS_REF_HNDL aAxes2;
MMC_RESET_IN stReset_in;
MMC_RESET_OUT stReset_out;

void TCPDemoServer();
int TCPDemoClient(char sOs, double sPoruka);
void vPrimanje(char sOs, char sOs1, double dSend);
void thread1();
void thread2();
void MainInit();
void MachineSequences(float fOs);
void ReadAllInputData();
void GetDigitalInput(MMC_CONNECT_HNDL hConn, MMC_AXIS_REF_HNDL iAxisRef);
void GetGMASOperationMode (MMC_CONNECT_HNDL hConn);
void SetGMASOperationMode (MMC_CONNECT_HNDL hConn, int mode);
void resetAxis(MMC_CONNECT_HNDL hConn, MMC_AXIS_REF_HNDL iAxisRef);
char rxbuff[256]={0};
char txbuff[256]={0};
int r;
CMMCUDP		cUDP_X1 ;

int		giState1 ;			// Holds the current state of the 1st main state machine
int		giStatus1;
int		giPos1	;
```

```
double  giCur1   ;
int     giErr1   ;

int          giState2 ;                    // Holds the current state of the 2st main state machine
int          giStatus2;
int          giPos2   ;
double  giCur2   ;
int     giErr2   ;

int giGroupStatus;
static unsigned short g_usPort=5005,g_usPort2=5004;
double dPozicija=0,x=0, fID=0,fCom=0, fPoruka=0, dInput;
double dConverzijaR1=186413.61111111,dConverzijaR1demo=27.7777777 ,
dConverzijaR2=955.267; // konverzija kutevi u countovi na testnom postavu
string s;
int iPow=1;
bool iCon=false,iCon1=false, iCon2=false, bSlanje=true,bOs=true;//Brojac konekcija, brojac
disable/enable osi 1, brojac dis/abl osi 2
extern bool EmgRec;
int xTest=0;

#define     FIRST_SUB_STATE                         1

enum eMainStateMachines                                                 // TODO: Change
names of state machines to reflect dedicated project
{
            eIDLE               =           0,
            eSM1                =           1,
            eSM2                =           2,
            eSM3                =           3,
            eSM4                =           4,
            eSM5                =           5,
            eSM6                =           6,
            eSM7                =           7,
} ;
```

2.3 CallBack.cpp

```
#include <iostream>
#include "mmcpplib.h"
#include "CallBack.h"
#include <string.h>
#include <sstream>

using namespace std;

#define SSTR( x ) static_cast< std::ostringstream & >( \
      ( std::ostringstream() << std::dec << x ) ).str()*/

typedef enum {eSOCKET_CREATE_STATE=0, eSOCKET_ACCEPT_STATE,
            eSOCKET_READ_STATE}EDEMO_SOCK_STATE;
typedef enum {eSOCKET_CONNECT_STATE,
```

```
                    eSOCKET_WRITABLE_STATE, eSOCKET_SEND_STATE
}EDEMO2_SOCK_STATE;
typedef struct _sockdemo_msg_t{
        char _iID;
        char _cCom[3];
        char _szUserData[128];
}dsock_msg_t;

typedef struct _sockclient_msg_t{
        //char _iID2;
        char _sUserData2[256];

}dsock2_msg_t;

bool bSlanjeC=true;
static int iSock2Out;
static CMMCTCP tcpClient1;

int TCPDemoClientC(char sOs, char sOs1,double sPoruka){
        static EDEMO2_SOCK_STATE eState2=eSOCKET_CONNECT_STATE;
        int rc2=-1;
        bool bWait;
        static dsock2_msg_t msg2;
        static int iSock2;

        switch(eState2){
        case eSOCKET_CONNECT_STATE:
                rc2=tcpClient1.Connect("192.168.1.7",5004,iSock2,bWait);
                //sleep(0.1);
                if (rc2>=0){
                        eState2=eSOCKET_WRITABLE_STATE;
                }
                break;
        case eSOCKET_WRITABLE_STATE:
                if (tcpClient1.IsWritable(iSock2)){
                        eState2=eSOCKET_SEND_STATE;
                }
                break;

        case eSOCKET_SEND_STATE:
                memset(&msg2,0,sizeof(dsock2_msg_t));
                sprintf(msg2._sUserData2,"%c%c%f\n",sOs,sOs1,sPoruka);
                rc2=tcpClient1.Send(iSock2,sizeof(dsock2_msg_t),(void *)&msg2);
                //sleep(0.1);
                if(rc2>0){
                        bSlanjeC=false;
                        eState2=eSOCKET_CONNECT_STATE;
                        return(iSock2);
                }
                eState2=eSOCKET_CONNECT_STATE;
                break;
        }
        return(0);
}
void vPrimanjeC(char sOs, char sOs2,double dSend){
```

```
		while(bSlanjeC){
					iSock2Out=TCPDemoClientC(sOs,sOs2,dSend);
					//sleep(0.1);
		}
		bSlanjeC=true;
		usleep(10000);
		//cout<<__func__<<iSock2Out<<endl;
		tcpClient1.Close(iSock2Out);
		usleep(10000);

}


// Callback Function once an Emergency is received.
void Emergency_Received(unsigned short usAxisRef, short sEmcyCode){
		bool EmgRec=true;
		printf("Emergency Message Received on Axis %d. Code: %x\n",usAxisRef,sEmcyCode);
		switch(usAxisRef){
		case 0:
			vPrimanjeC('E','1',sEmcyCode);
			break;
		case 1:
			vPrimanjeC('E','2',sEmcyCode);
			break;
		}
}
void Motion_Ended(unsigned short usAxisRef, short sEmcyCode){
		printf("Motion ended on axis %d. Code: %x\n",usAxisRef,sEmcyCode);
		switch(usAxisRef){
		case 0:
			vPrimanjeC('E','1',sEmcyCode);
			break;
		case 1:
			vPrimanjeC('E','2',sEmcyCode);
			break;
		}
}


void EndMotionEventCB(unsigned short usAxisRef){
// ===========================================
 printf("\n\t\t %s: Received usAxisRef=%d ", __func__, (int)usAxisRef);
 //printf("\n\t\t %s \n", EndMotionEventCB_MESSAGE);
 fflush(stdout); fflush(stderr);
}
int    OnRunTimeError(const char *msg,  unsigned int uiConnHndl, unsigned short usAxisRef, short
sErrorID, unsigned short usStatus){
//
================================================================================
============
 printf("\n APP: MMCPPExitClbk: Run time Error in function %s, axis ref=%d, err=%d, status=%d,
bye\n",
    msg, usAxisRef, sErrorID, usStatus);

		vPrimanjeC('E','1',sErrorID);
```

```
 fflush(stdout); fflush(stderr);
 MMC_CloseConnection(uiConnHndl);
 exit(0);

}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
int CallbackFunc(unsigned char* recvBuffer, short recvBufferSize,void* lpsock){
        // Whcih function ID was received ...
        switch(recvBuffer[1]){
        case ASYNC_REPLY_EVT:
                printf("ASYNC event Reply\r\n") ;
                break ;
        case EMCY_EVT:
                // Please note - The emergency event was registered.
                // printf("Emergency Event received\r\n") ;
                break ;
        case MOTIONENDED_EVT:
                printf("Motion Ended Event received\r\n") ;
                break ;
        case HBEAT_EVT:
                printf("H Beat Fail Event received\r\n") ;
                break ;
        case PDORCV_EVT:
                printf("PDO Received Event received - Updating Inputs\r\n") ;
                break ;
        case DRVERROR_EVT:
                printf("Drive Error Received Event received\r\n") ;
                break ;
        case HOME_ENDED_EVT:
                printf("Home Ended Event received\r\n") ;
                break ;
        case SYSTEMERROR_EVT:
                printf("System Error Event received\r\n") ;
                break ;
        }
        //
        return 1 ;
}
```

2.4 CallBack.h

```
void Emergency_Received(unsigned short usAxisRef, short sEmcyCode) ;
void Motion_Ended(unsigned short usAxisRef, short sEmcyCode) ;
int  CallbackFunc(unsigned char* recvBuffer, short recvBufferSize,void* lpsock);
int  OnRunTimeError(const char *msg,  unsigned int uiConnHndl, unsigned short usAxisRef, short
sErrorID, unsigned short usStatus);
void EndMotionEventCB(unsigned short usAxisRef);
//bool EmgRec=false;
extern bool bSlanjeC;
```

3 Euler_interpol.m

```
filename='error.txt';
delimiterIn=',';
AerX(10,10)=0;
AerY(10,10)=0;
AerZ(10,10)=0;
InterEul(91,91)=0;
A=importdata(filename,delimiterIn);
[Xer,Yer]=meshgrid(0:10:90,0:10:90);
[xer,yer]=meshgrid(0:90,0:90);
for i=0:1:9
   for j=1:1:10
      AerX(i+1,j)= A((i*10)+j,3);
   end
end

for i=0:1:9
   for j=1:1:10
      AerY(i+1,j)= A((i*10)+j,4);
   end
end
%
for i=0:1:9
   for j=1:1:10
      AerZ(i+1,j)= A((i*10)+j,5);
   end
end

interAX=interp2(Xer,Yer,AerX,xer,yer);
interAY=interp2(Xer,Yer,AerY,xer,yer);
interAZ=interp2(Xer,Yer,AerZ,xer,yer);

for i=1:1:91
   for j=1:1:91
      InterEul(i,j)=sqrt(interAX(i,j)^2+interAY(i,j)^2+interAZ(i,j)^2);
   end
end

figE=figure('NumberTitle', 'off', 'Name', 'INTERPOLACIJA UKUPNE GREŠKE');
subplot(1,1,1)
sE=surf(xer,yer,InterEul);
sE.EdgeColor = 'none';
% sX.FaceColor='interp';
title('Interpolirano Euler');
zlim([-1.5,1.5])

figX=figure('NumberTitle', 'off', 'Name', 'INTERPOLACIJA GREŠKE X');
subplot(1,2,1)
sX=surf(xer,yer,interAX);
```

```
sX.EdgeColor = 'none';
% sX.FaceColor='interp';
title('Interpolirano X');
zlim([-1.5,1.5])

subplot(1,2,2)
s1X=surf(Xer,Yer,AerX);
s1X.EdgeColor = 'none';
title('Normalno X');
zlim([-1.5,1.5])

figY=figure('NumberTitle', 'off', 'Name', 'INTERPOLACIJA GREŠKE X');
subplot(1,2,1)
sY=surf(xer,yer,interAY);
sY.EdgeColor = 'none';
title('Interpolirano Y');
zlim([-1.5,1.5])

subplot(1,2,2)
s1Y=surf(Xer,Yer,AerY);
s1Y.EdgeColor = 'none';
title('Normalno Y');
zlim([-1.5,1.5])

figZ=figure('NumberTitle', 'off', 'Name', 'INTERPOLACIJA GREŠKE Z');
subplot(1,2,1)
sZ=surf(xer,yer,interAZ);
sZ.EdgeColor = 'none';
title('Interpolirano Z');
zlim([-1.5,1.5])

subplot(1,2,2)
s1Z=surf(Xer,Yer,AerZ);
s1Z.EdgeColor = 'none';
title('Normalno Z');
zlim([-1.5,1.5])
%colorbar;
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [7 7 7 7];
print(figE,'Efinal','-dpng','-r600')
print(figX,'Xfinal','-dpng','-r600')
print(figY,'Yfinal','-dpng','-r600')
print(figZ,'Zfinal','-dpng','-r600')
```