Upravljanje laganom robotskom rukom pomoću PC računala

Trslić, Petar

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Petar Trslić

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Bojan Jerbić

Student:

Petar Trslić

Zagreb, 2015.

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

Ovim putem želim se zahvaliti mentoru, prof.dr.sc. Bojanu Jerbiću, što je prihvatio mentorstvo za ovaj rad te što je svojom pristupačnošću i pružanjem korisnih savjeta pomogao pri izradi istoga. Također želim zahvaliti asistentu, dr.sc. Bojanu Šekoranji koji je sa svojim znanjem, stručnim savjetima i iskustvom pridonio kvaliteti izrade diplomskog rada.

Na kraju želim zahvaliti svojim roditeljima, sestrama, cijeloj obitelji te prijateljima koji su me tokom cijelog studija bodrili i bili mi podrška.

Petar Trslić

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

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

DIPLOMSKI ZADATAK

Student: **Petar Trslić** Mat. br.: 0035178064

Naslov rada na hrvatskom jeziku: **Upravljanje laganom robotskom rukom pomoću PC računala**

Naslov rada na engleskom jeziku: **PC-based controll of lightweight robot arm**

Opis zadatka:

U radu je potrebno proučiti tehničke i programske značajke lagane robotske ruke Kuka LWR4+ sa sedam stupnjeva slobode u cilju razvoja programske podrške koja povezuje robot s računalom. Pri tome koristiti robotsko sučelje FRI (eng. "fast research interface"), koje se temelji na UDP protokolu te omogućava upravljanje robotom i nadziranje njegovog stanja preko vanjskog PC uređaja. Programsku podršku i korisničko sučelje oblikovati pomoću raspoložive C++ programske knjižnice, koje će omogućiti izdavanje naredbi gibanja u kartezijevim koordinatama završnog zgloba robota, kao i upravljanje pojedinim zglobovima.

Razvijenu primjenu provjeriti koristeći opremu dostupnu u Laboratoriju za projektiranje izradbenih i montažnih sustava.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
7. svibnja 2015.	9. srpnja 2015.	15., 16. i 17. srpnja 2015.

Zadatak zadao: Predsjednik Povjerenstva:

Prof. dr. sc. Bojan Jerbić Prof. dr. sc. Franjo Cajner

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
α_i	°	DH parametar, zakret oko osi X
a_i	mm	DH parametar, translacija po osi X
θ_i	°	DH parametar, zakret oko osi Z
d_i	mm	DH parametar, translacija po osi Z
r_{kj}	-	Rotacijski elementi matrice transformacije
p_x, p_y, p_z	mm	Vektor pozicije
R_x	-	Rotacija koordinatnog sustava oko X osi
T_x	-	Translacija koordinatnog sustava oko X osi
R_z	-	Rotacija koordinatnog sustava oko Z osi
T_z	-	Translacija koordinatnog sustava oko Z osi
$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$	°	Iznos kuta zakreta zglobova
$J_1, J_2, J_3, J_4, J_5, J_6$	-	Zglob
a, b	mm	Duljina članka robota
c	mm	Duljina dužine koju zatvara robot
s	mm	Polovina opsega trokuta
r	mm	Polumjer upisane kružnice trokuta
α, β, γ	°	Kutevi koje zatvara robot

SAŽETAK

U radu je nakon uvodnog poglavlja opisan tehnički sustav gdje je obrađen pojam laganog robota, prikazane su sve karakteristike robotske ruke KUKA LWR4+ pomoću koje je izveden ovaj diplomski rad, te je objašnjen UDP protokol i FRI programsko sučelje koje se temelji na njemu. Razvijen je kinematički model robota, određeni su DH parametri robotskog sustava te na temelju njih je izvedena direktna i inverzna kinematika robota. Razvijena je programska podrška i korisničko sučelje pomoću raspoložive C++ knjižnice kao bi se moglo vršiti izdavanje naredbi gibanja robota u kartezijskom koordinatnom sustavu te upravljanje po pojedinim zglobovima.

Ključne riječi: direktna kinematika, inverzna kinematika, DH parametri, KUKA LWR4+,

SUMMARY

Abstract –The system, which is used in a research is described in this paper. The Light-weight robot term is explained and a technical characteristic of the KUKA LWR4+ robot are shown. Fast Research Interface (FRI) which is based on UDP protocol is explained. DH parameters of the KUKA LWR4+ robot have been specified in order to robot kinematics, direct and inverse, can be implemented. The software with the user interface is developed so the robot motion can be controlled in Cartesian or joint frame space.

Key words: direct kinematics, inverse kinematics, DH parameters, KUKA LWR4+

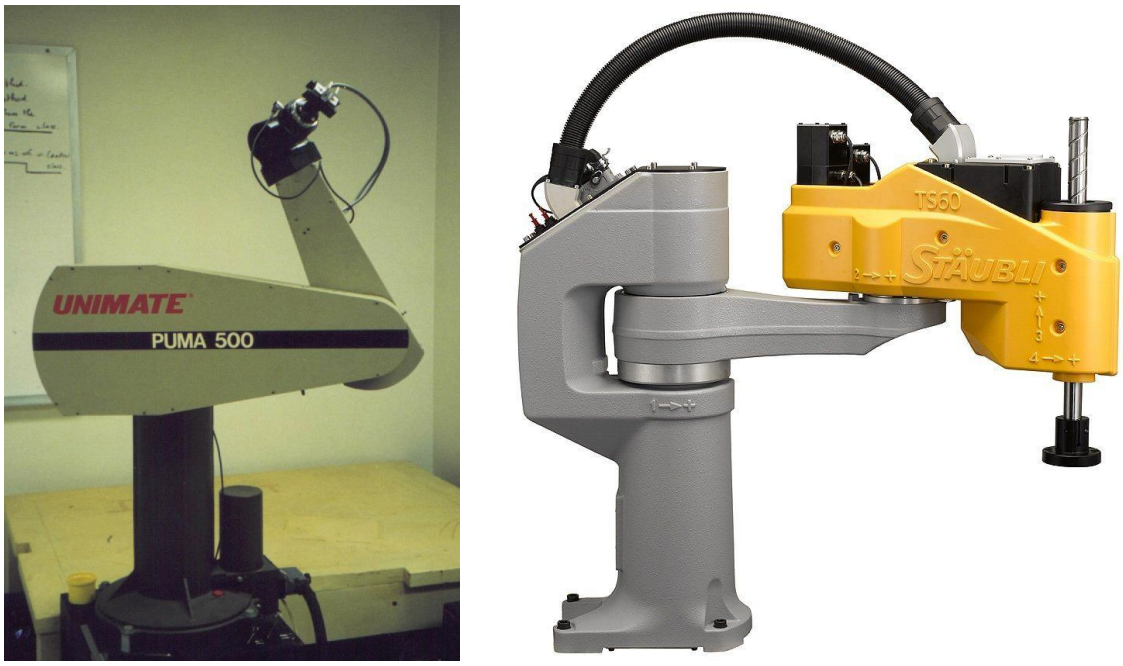
1. UVOD

Povijest robotike je od uvijek bila usko povezana sa povijesnim razvojem tehnologije, znanosti i općenito napretkom društva. Robot je uređaj koji može bez prestanka raditi te nema osobinu čovjeka da na njegovu produktivnost utječe umor. Zbog toga su od najranije povijesti ljudi pokušavali stvoriti uređaje koji će im olakšati rad. Razvoj raznih grana znanosti i tehnologije kao što su elektrotehnika, programiranje, pneumatika, hidraulika te u novije vrijeme psihologija i sociologija, doprinijeli su razvoju te interdisciplinarne grane znanosti. Ako gledamo razvoj robotike na taj način, može se reći da je razvoj robotike počeo oko početka nove ere, u doba Arhiemeda koji, naravno nije izumio robot, ali je pridonio razvoju robotike mnogim matematičkim i mehaničkim principima koji se i danas koriste u robotici. Danas bi industrijske robote mogli definirati kao mehaničke uređaje koje je moguće programirati za izvođenje raznih aplikacija. Također robote možemo nazvati i fizičkom reinkarnacijom računala.

Slika 1. UNIMATE – prvi industrijski robot

Razvoj industrijskog robota, kakvog poznajemo danas odvijao se tokom prošloga stoljeća. Riječ "robot" koja dolazi od riječi "rabota" što znači težak, prisilan rad, je prvi put upotrijebljena u drami češkoga pisca Karela Čapeka pod nazivom "R.U.R", dok je riječ "robotika" prvi put upotrijebio pisac znanstvene fantastike Isaac Asimov. Asimov je također postavio tri zakona robotike te je vrlo dobro predvidio smjer u kojem će se robotika razvijati u budućnosti. Razvojem poluvodičkih komponenti počinje razvoj suvremenog robota. Godine 1954. George Devol je konstruirao prvi, u potpunosti programibilnog robota pod nazivom UNIMATE kojeg prikazuje **Slika 1**.

U slijedećih 30-tak godina nastaju industrijski roboti kakve poznajemo danas. Predstavnici te generacije robota su PUMA robot i SCARA robot (**Slika 2**), a razvojem senzorske tehnologije roboti sve više postaju autonomni i postaju "svjesniji" svoje okoline.



Slika 2. PUMA i SCARA robot

Daljnji razvoj robotike temelji se na razvitku boljih materijala izrade robota, aktuatorskih i senzorskih sustava te boljim algoritmima upravljanja i umjetnom inteligencijom. Upravo zbog razvoja boljih algoritama upravljanja i umjetne inteligencije dolazi do potrebe za razvojem boljih upravljačkih računala. Kako su danas računala relativno velike računalne moći svuda oko nas, te možemo reći da roboti sve više postaju involvirani u rad sa ljudima, dolazimo do motivacije za izradu ovoga rada.

Cilj je napraviti programsku podršku (software) koji će povezati KUKA LWR4+ robot sa računalom te omogućiti upravljanje robotom u kartezijevom koordinatnom sustavu i upravljanje pojedinim zglobovima robota. Izradom takve programske podrške, otvara se

mogućnost daljnjeg razvitka sustava u pogledu razvoja boljih algoritama upravljanja od predefiniranih u upravljačkom računalu robota. Programska podrška je izvedena u programskom jeziku C++, a komunikacija između računala i upravljačkog računala robota izvršena je preko UDP protokola i "Fast Research Interface" sučelja koji su detaljnije objašnjeni u slijedećem poglavlju.

2. OPIS SUSTAVA

U diplomskom radu korištena je robotska ruka KUKA LWR4+ sa robotskim sučeljem FRI koje se temelji na UDP protokolu. U ovom poglavlju dan je opis lagane robotske ruke KUKA LWR4+, UDP protokola i FRI programske knjižnice koja je poslužila za komunikaciju i upravljanje sa robotskom rukom.

2.1. Lagani roboti

Lagani roboti - LWR (eng. Light Weight Robot) su roboti posebno dizajnirani kako bi se mogli koristiti u interakciji sa ljudima ili nepoznatom okolinom. Primjena robota u neposrednoj interakciji sa čovjekom postavlja posebne zahtjeve za robota kao što je visoki omjer mase robota i njegove nosivosti te predodžba o okolini. Modularna struktura laganih robota i velike upravljačke mogućnosti svrstavaju lagane robote u područje robotike koje nazivamo mekana robotika (eng. soft robotics).

Kako su standardni industrijski roboti uglavnom korišteni za određene repetitivne radnje u točno strukturiranoj okolini, njihov dizajn je drugačiji od dizajna laganih robota. Kako bi standardni industrijski roboti mogli postići visoku preciznost i ponovljivost, robotske ruke moraju biti iznimno krute, a prešutno možemo zaključiti i teške. Roboti koji se koriste u interakciji sa ljudima ne smiju biti teški jer velika masa robota pri velikoj brzini može biti smrtonosna za čovjeka. Zbog toga su lagani roboti konstruirani kako bi se mogla osigurati veća nosivost i mobilnost uz manju potrošnju energije i manju masu kako bi se u slučaju kvara robota mogla garantirati sigurnost ljudi. Kako se lagani roboti koriste u nestrukturiranoj okolini, ne mogu se oslanjati samo na točnost pozicioniranja. Takvi roboti moraju imati mogućnost bolje percepcije okoline. Zbog toga je u takvim uvjetima bolje da se robot ponaša popustljivo. Konkretno to znači da kad se na robotskoj ruci ostvaruju sile veće od definiranih, robot će smanjiti silu na način da se giba u smjeru djelovanja sile. Kako bi se takvo ponašanje robota moglo ostvariti, momenti i sile na robotskoj ruci moraju se mjeriti te se informacija o njima šalju u upravljačko računalo robota.

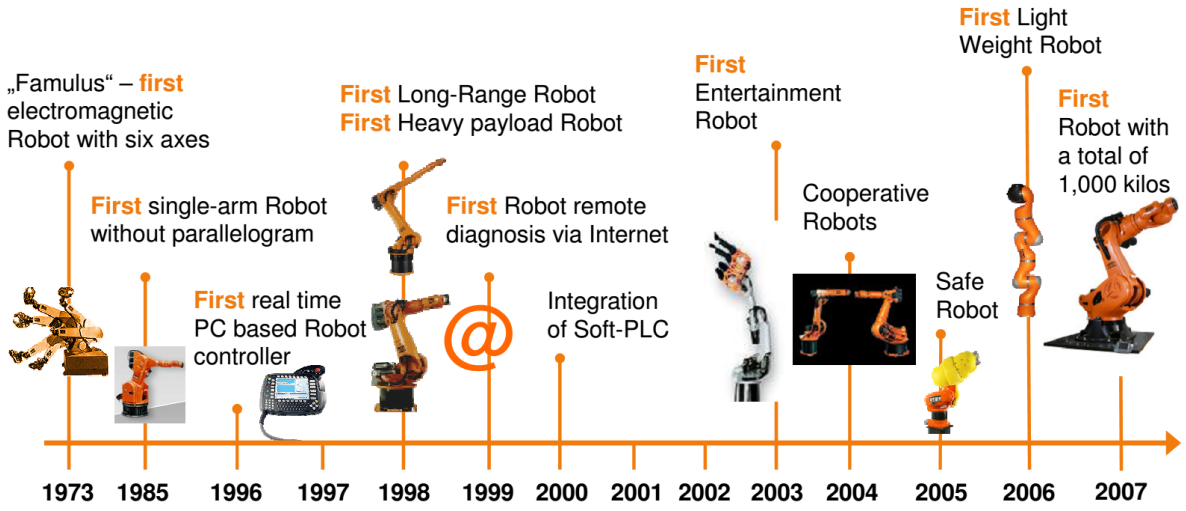
Danas postoje različiti proizvođači laganih robota međutim konstrukcijski zahtjevi su jednaki.

1. Lagana konstrukcija robota. Konstrukcija je izrađena od lakih metala ili od kompozitnih materijala. Uz to cijeli sustav (upravljačko računalo, napajanje) je konstruirano kao optimalno po kriteriju mase kako bi se omogućilo korištenje LWR-a u mobilnoj robotici.
2. Niska potrošnja električne energije. Niska potrošnja električne energije bitna je zbog sigurnosnih razloga (novi propisi diktiraju da roboti smiju imati maksimalno 80W mehaničke snage), ali i zbog mogućnosti primjene u mobilnoj robotici. Naime, snažniji motori znače veću potrošnju energije što u konačnici zahtjeva veće baterije ili veće površine za solarne panele itd.
3. Elektronika integrirana unutar samih zglobova robota kako bi se postigla mogućnost modularnosti.
4. Motori sa visokom učinkovitošću. Za razliku od standardnih industrijskih robota kod kojih motori rade na visokim okretajima, motori laganih robota rade pri nižim brzinama, imaju velik moment, energetski su učinkovitiji i imaju brz dinamički odziv.
5. Prijenosnici sa visokim omjerom opterećenje/masa kao što su harmonički prijenosnici gibanja i momenata.
6. Mjerenje momenata i/ili pozicije u svakom zglobu robota.
7. U svrhu sigurnosti redundantna mjerenja primjerice pozicije robota, sile, momenata, struje itd.

Zbog ispunjavanja navedenih zahtjeva lagani roboti ponašaju su elastičnije od klasičnih industrijskih robota. Stoga je dinamika robota kompleksnija te je potrebno implementirati naprednije algoritme upravljanja kako bi preciznije odredili gibanje robota. Zbog kompleksnosti izrade, većeg broja senzora, upotrebe lakših materijala i naprednijih prijenosnika, cijene ovakvih robota su više od cijene standardnih industrijskih robota.
Zadatak ovog diplomskog rada je izveden na laganom robotu KUKA LWR4+ čije su karakteristike opisane u nastavku poglavlja.

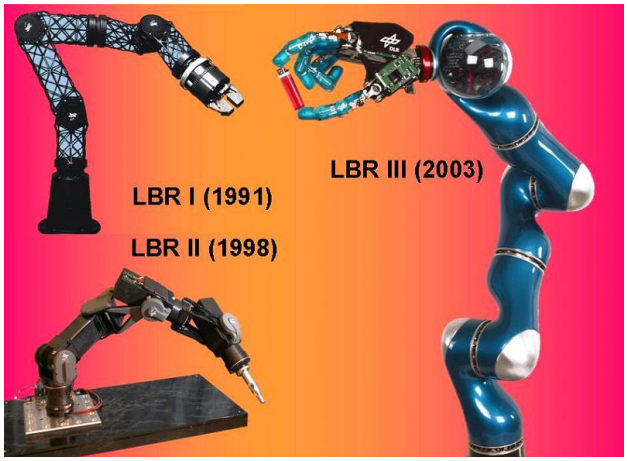
2.2. KUKA LWR4+

KUKA (Keller und Knappich Augsburg) je njemački proizvođač industrijskih robota za različite primjene. Tvrtku su osnovali Johann Josef Keller i Jacob Knappich 1898. godine u Augsburgu u Njemačkoj. Prvi robot pod imenom FAMULUS proizveden je 1973. godine. Tvrtka danas proizvodi različite modele robota nosivosti od 3 do 1300kg koji se koriste u raznim primjenama.



Slika 3. Povijesni razvoj KUKA robota [7]

Proizvodnja serije lakih robota počinje u suradnji sa njemačkim institutom za robotiku i mehatroniku (DLR). Prvotno razvijen robot pod nazivom DLR LWR 1, korišten je 1993. godine u svemirskoj misiji. Njime je dokazano da je moguće upravljati robotskim sustavom u svemiru sa Zemlje. Robot LWR 2 razvijen je 1998 godine. Težio je samo 18 kg a nosivost mu je bila 7 kg pri punoj brzini. Pet godina kasnije razvijen je LWR 3. Robot je imao redundantnu, sedmu os i građen je po uzoru na ljudsku ruku tako da je ostvarena velika pokretljivost. Postignut je omjer nosivost/masa 1:1 jer uz nosivost od 14 kg u cijelom radnom opsegu, masa robota je bila svega 14 kg. To je bio prvi industrijski robot koji je mogao nositi samoga sebe. Generacije KUKA laganih robota su prikazane na slici (**Slika 4**).

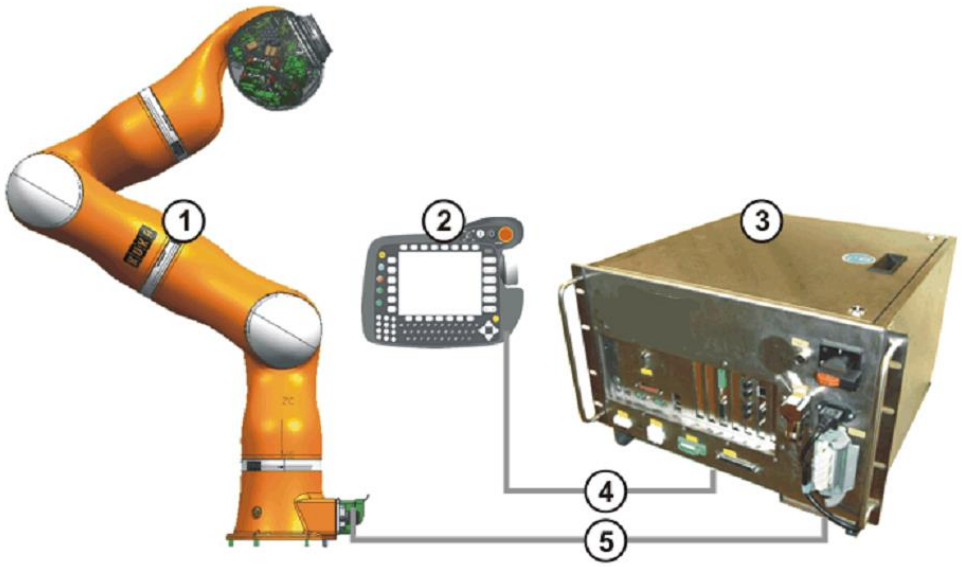


Slika 4. Razvoj laganih robota [7]

Nakon dugogodišnje suradnje KUKA-e i DLR-a razvijen je robot KUKA LWR4+ koji je našao primjenu u znanstvenim krugovima zbog svoje iznimne pokretljivosti te zbog izrazito naprednog upravljačkog i mjernog sustava. Sedam stupnjeva slobode gibanja čine ovaj robot redundantnim te je zbog toga moguće robot postaviti u različite konfiguracije kako bi se

zauzeo određen položaj. Također se zbog redundantnosti izbjegavaju mnoge singularnosti koje se postižu robotom sa šest stupnjeva slobode gibanja. Robot je preuzeo dizajn od prethodnika. Za razliku od prethodnih generacija, četvrta generacija robota razvijana je s ciljem probne primjene LWR-a u industriji. Robot je mase 15 kg dok mu nosivost iznosi 7 kg.

2.2.1. Tehničke specifikacije



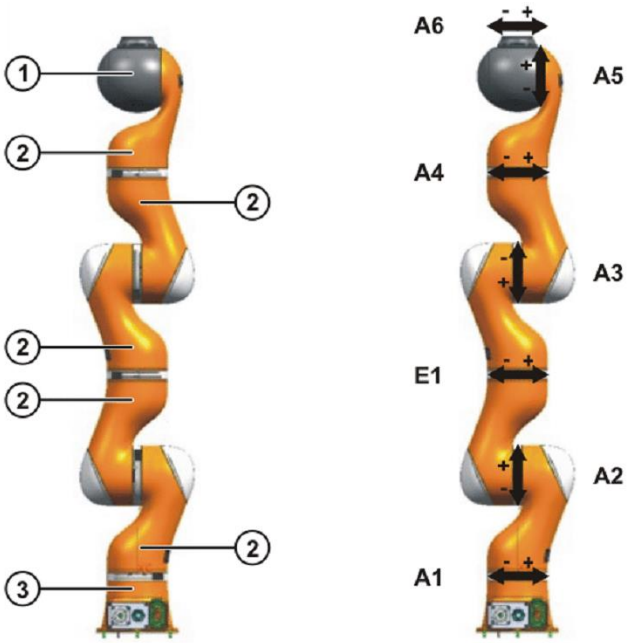
Slika 5. Robot KUKA LWR4+ [4]

Na slici (**Slika 5**) je prikazan robotski sustav koji se sastoji od robotskog manipulatora KUKA LWR4+ (1), privjeska za učenje (2), upravljačkog računala robota (KRC, eng. Kuka Robot Controler)(3) te kablova za povezivanje privjeska za učenje i manipulatora na upravljačko računalo (4,5).

KUKA LWR4+ je robot izveden sa sedam rotacijskih zglobova. U svim zglobovima se nalaze senzori položaja i momenata te je zbog toga moguće upravljati pozicijom, brzinom i momentom robota. Osnovne karakteristike robota su prikazane u tablici (**Tablica 1**)

Tablica 1. Osnovne karakteristike KUKA LWR4+

Nosivost	7 kg (14 kg)
Masa	14 kg
Broj osi	7
Napajanje	48V DC interno 220V AC eksterno
Upravljačko računalo	KRC 2



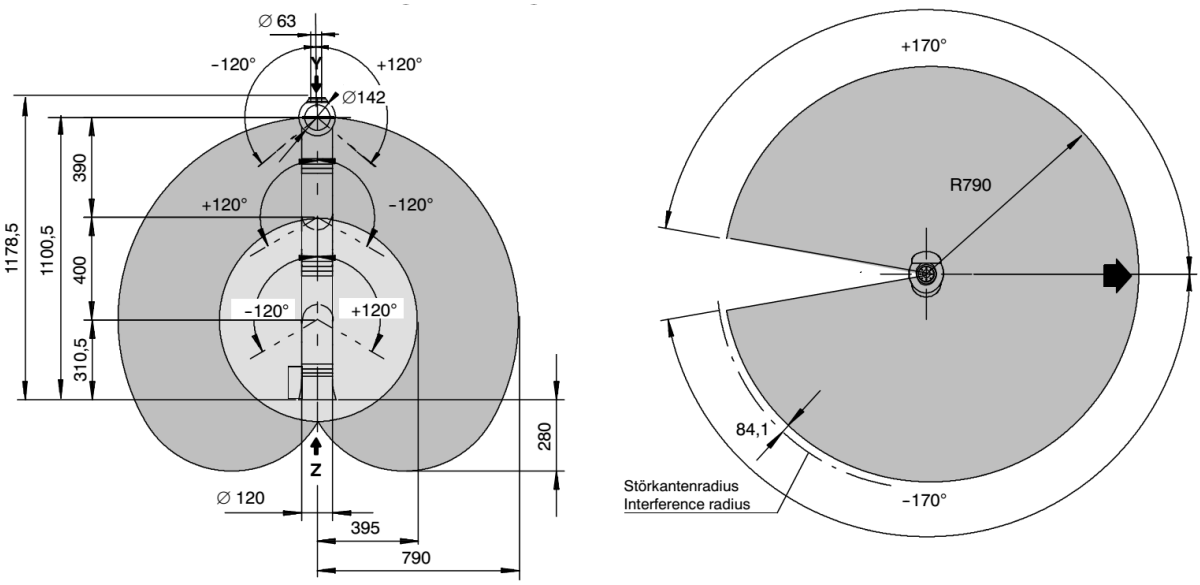
Slika 6. Glavni dijelovi robotske ruke [4]

Na slici (**Slika 6**) su prikazani glavni dijelovi robotske ruke. Robotska ruka se sastoji od dvoosnog zgloba (1), zglobnih modula (2), koji su izrađeni od aluminija i unutar kojih su smještene upravljačke jedinice te od baze robota (3). Potrebno je poznavati ograničenja svakoga zgloba, koja su dana u tablici (**Tablica 2**), kako nebi došlo do oštećenja robotske ruke ili upravljačkog računala prilikom izvođenja napisanog programa.

Tablica 2. Ograničenje zglobova robota

Zglob	Raspon kuta zakreta, softverski ograničen [°]	Brzina bez tereta [°/s]	Maksimalni moment [Nm]
A1 (J1)	+/- 170	112.5	200
A2 (J2)	+/- 120	112.5	200
E1 (J3)	+/- 170	112.5	100
A3 (J4)	+/- 120	112.5	100
A4 (J5)	+/- 170	180.0	100
A5 (J6)	+/- 120	112.5	30
A6 (J7)	+/- 170	112.5	30

Dimenzije robotske ruke i radni prostor prikazan je na slici (**Slika 7**).



Slika 7. Radni prostor i dimenzije robotske ruke [4]

Kako je robot KUKA LWR4+ predviđen za korištenje u različitim okolinama, velik dio pažnje posvećen je njegovoj mobilnosti. Pod mobilnošću, uz malu masu robota, smatramo i mogućnost integriranja robota u sustave više razine. U tu svrhu razvijeno je korisničko sučelje koje se temelji na UDP protokolu i FRI (eng. Fast Reseach Interface) programskom sučelju koji su detaljnije opisani u nastavku poglavlja.

2.3. UDP protokol

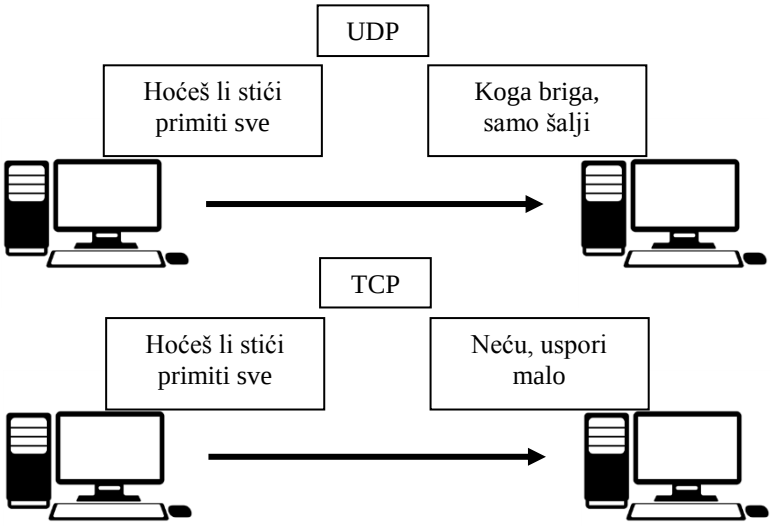
User Datagram Protocol (UDP) je protokol definiran sa RFC 768 te pripada prijenosnoj razini TCP/IP modela mreže. UDP protokol možemo smatrati proširenjem IP protokola , a osnovna zadaća mu je postavljanje neslijedne (nepovezane eng. connectionless) komunikacije između udaljenih računala [6]. Pod neslijednom komunikacijom smatramo da nije potrebno prvo uspostaviti vezu kako bi mogli odaslati podatke putem mreže. Hijerarhija porodice TCP/IP protokola prikazana je u tablici.

Tablica 3. Hijerarhija porodice TCP/IP protokola

<table>
<tr><td>TFTP</td><td>Time Service</td><td>Name Res.</td><td>Telnet</td><td>FTP</td><td>SMTP</td></tr>
<tr><td colspan="3">ICMP,UDP</td><td colspan="3">TCP</td></tr>
<tr><td colspan="6">IP</td></tr>
</table>

UDP koristi jednostavni prijenosni neslijedni model kako bi postigao što manju latenciju tj. mogao odaslati podatke što brže. Korištenjem UDP protokola ne postoji garancija da je određeni paket isporučen, da su paketi isporučeni po redoslijedu kojim su poslani ili da isti paket nije više puta poslan. Na Internetu, mreži svih mreža, UDP protokol je dosta zastupljen ali moglo bi se reći za manje važne podatke. Uglavnom se koristi za video i audio streaming zbog svoje brzine. No UDP protokol ne možemo smatrati lošim ili nepouzdanim. Network File System (NFS) protokol je jedan od viših protokola koji pouzdano radi u praksi, a koristi UDP.

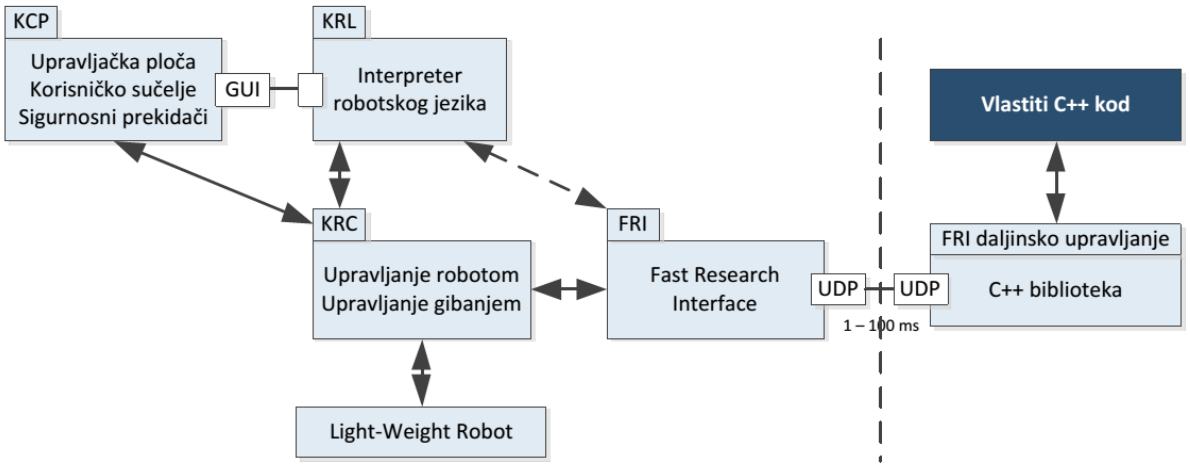
Najveća prednost UDP protokola je što su paketi puno manji od paketa, primjerice TCP protokola, koje možemo zatim slati minimalnim mehanizmom. Za primjenu upravljanja robotom KUKA LWR4 u realnom vremenu, UDP protokol je idealan. Kako se radi o lokalnoj razini mreže, štoviše o direktnoj komunikaciji između računala i upravljačkog računala robota preko mrežnog kabela, mogućnost da se poruka izgubi je svedena na minimum. Također ne može doći do preopterećenja mreže što je jedan od problema UDP protokola. Razina odašiljanja podataka je na razini 1 ms što je u potpunosti prihvatljivo za aplikacije koje se izvode u realnom vremenu. Usporedba TCP i UDP protokola prikazana je na slici (**Slika 8**).



Slika 8. Razlika između UDP i TCP protokola

2.4. Fast Research Interface (FRI)

Fast Research Interface (FRI) je knjižnica koja nam omogućuje jednostavno korisničko sučelje sa robotom KUKA LWR4+. Koristeći FRI ne znamo sve pojedinosti o komunikaciji koja se nalazi iza samog sučelje. Shema FRI sustava prikazana je na slici (**Slika 9**).



Slika 9. Shema FRI sustava upravljanja [5]

FRI je dakako samo sučelje i samo po sebi nije u mogućnosti upravljati robotom. Potrebno je napisati programski kod pomoću kojeg je moguće ostvariti upravljanje.

FRI knjižnica se pokreće na udaljenom računalu koje je povezano na KRC (eng. Kuka Robot Controller) preko ethernet veze. U intervalima od 1 do 100 milisekundi, UDP paketi se periodički šalju između KRC-a i udaljenog računala. Paketi mogu sadržavati razne informacije kao što su položaj zglobova, iznos momenata u zglobovima, temperature određenog zgloba itd. Pregled Fast Research Interface-a dan je u tablici (**Tablica 4**).

Tablica 4. Pregled Fast Research Interface-a

Funkcije	Određivanje vrijeme ciklusa za prijenos podataka (1-100 ms)
	Slanje upravljačkih signala u stvarnom vremenu
	Nadzor kretanja robotske ruke sa udaljenog računala
	Pristup podacima robota sa udaljenog računala
	Prijenos poruka upravljačkog računala na udaljeno računalo
	Primanje i slanje vrijednosti sistemskih i programskih varijabli robota
	Izvršavanje programa sa udaljenog računala
Značajke	Nadgledajući (eng. Monitor mode) način rada
	Upravljački (eng. Command mode) način rada
	Interakcija sa KRL (KUKA Robot language) programima

Prijenos podataka između upravljačkog računala robota i udaljenog računala odvija se kroz dva načina rada:

1. Monitor način rada – cikličan način komunikacije za mogućnosti prijenosa podataka o robotu na udaljeno računalo
2. Upravljački način rada – ciklična komunikacija sa mogućnosti prijenosa podataka (naredbi) sa udaljenog računala na upravljačko računalo robota

Naredbom FRIOPEN aktivira se FRI komunikacija između upravljačkog računala robota i udaljenog računala. Nakon izvršenja naredbe *Monitor* način rada je aktivan, te se određuje i klasificira kvaliteta veze.

Naredbom FRIOPEN aktivira se *Command* način rada ukoliko je kvaliteta veze ocjenjena kao "Dobra" ili "Izvrsna". Izmjena podataka između upravljačkog računala robota i udaljenog računala isključivo ovisi o kvaliteti veze te je stoga potrebno konstantno provjeravati kvalitetu veze. Ukoliko je veza zadovoljavajuća moguće je nadzirati kretanje robotske ruke ili pokretanje određenog programa na upravljačkom računalu sa udaljenog računala.

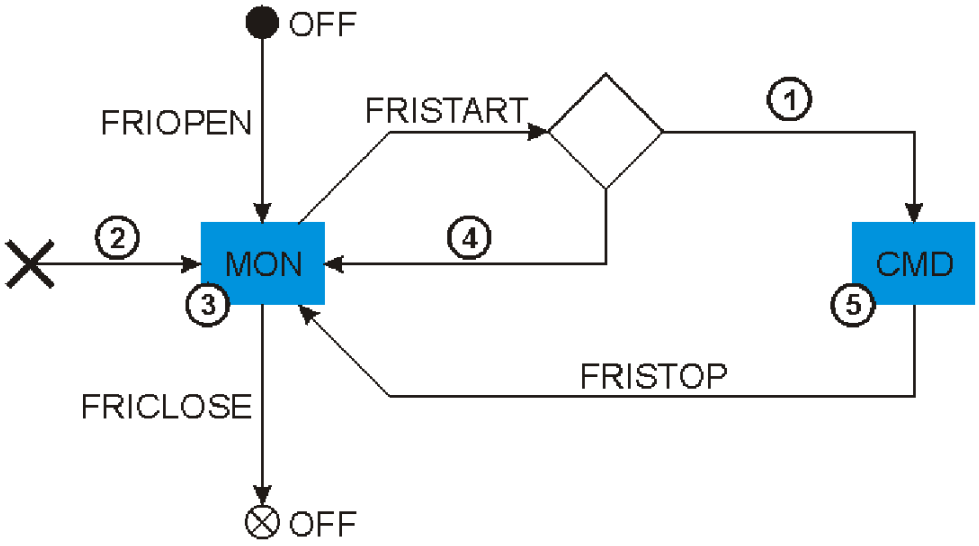
Naredba FRISTOP deaktivira *Command* način rada i sustav se automatski prebacuje u *Monitor* način rada.

Naredbom FRICLOSE u potpunosti se gasi FRI veza.

Ako dođe do ikakve greške dok je sustav u *Command* načinu rada, *Command* način rada se smjesta deaktivira, te se FRI prebacuje u *Monitor* način rada. Do greške može doći radi slijedećih razloga:

1. Kvaliteta veze nije klasificirana kao "Dobra" ili "Odlična"
2. U komunikaciji je izgubljeno više od jednog UDP paketa
3. UDP paketi nisu poslani u određenom nizu
4. Upravljačke vrijednosti u *Command* načinu rada su veće od limitiranih. Npr. Zadana je nedostižna pozicija robota ili brzina veća od maksimalne deklarirane

Algoritam po kojem radi Fast Research Interface prikazan je na slici **(Slika 10**).



Slika 10. Dijagram toka FRI [8]

1. Podatkovni paket poslan u određenom timeframe-u
2. Greška
3. *Monitor* način rada
4. Kvaliteta veze nije dovoljno dobra za prijenos podataka
5. *Command* način rada

Slika prikazuje da se kontinuirano, paralelno sa izvođenjem naredbi, odvija provjera od grešaka (2) poput krivo zadane pozicije robota. Ukoliko grešaka nema, podatkovni paket (1) se može poslat u određenom timeframe-u i *Command* način rada je aktivan (5). Ukoliko kvaliteta veze (4) nije zadovoljavajuća, sustav se prebacuje u *Monitor* način rada.

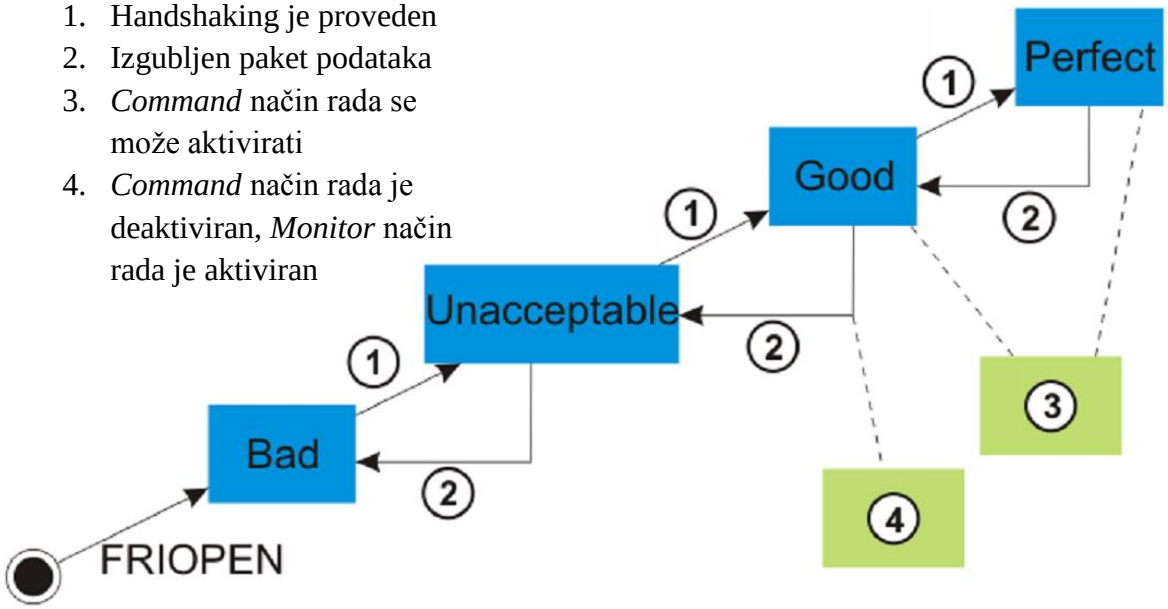
Kao što je već napomenuto, kvaliteta veze je od esencijalne važnosti za rad FRI-a. Kvaliteta veze se klasificira kao "Loša"(eng. Bad), "Neprihvatljiva"(eng. Unacceptable), "Dobra"(eng. Good) i "Izvrsna"(eng. Perfect). Kvalitetu veze možemo provjeriti koristeći sistemsku varijablu $FRIQUALITY. Naredbom FRISHOW() mogu se dobiti detaljniji podaci o kvaliteti veze.

Dakle moguće je aktivirati *Command* način rada sa FRISTART naredbom tek kad sistemska varijabla $FRIQUALITY vrati status "OK". Na slici (**Slika 11**) je prikazan dijagram toka klasifikacije kvalitete veze.

Kada se prvi put uspostavi FRi veza, kvaliteta veze je označena kao "Loša". Sa uspješnim handshake-om kvaliteta veze raste. Ukoliko tokom rada u *Command* načinu rada kvaliteta

prijenosa podataka postane klasificirana kao loša ili neprihvatljiva, *Command* način rada se deaktivira i *Monitor* način rada se aktivira.

1. Handshaking je proveden
2. Izgubljen paket podataka
3. *Command* način rada se može aktivirati
4. *Command* način rada je deaktiviran, *Monitor* način rada je aktiviran



Slika 11. Klasifikacija veze po kriteriju kvalitete [8]

3. KINEMATIČKI MODEL ROBOTA

Kinematika tijela je dio fizike koji se bavi proučavanjem gibanja određenog tijela ne uzimajući u obzir sile ili momente koji utječu na to gibanje. Kinematika robota je pojam koji se odnosi na analitičko proučavanje gibanja robotskog manipulatora. Formuliranje prikladnog kinematičkog modela određenog robotskog mehanizma je nužno kako bi mogli proučavati njegovo ponašanje.

Za modeliranje kinematike manipulatora uglavnom se koristi kartezijski prostor. Transformacija između dva kartezijska koordinatna sustava može se razložiti na translaciju i rotaciju. Rotacija se može prikazati na razne načine kao što su: Eulerovi kutevi, Gibbs-ov vektor, Cayley-Klein parametri, ortogonalne matrice, Hamiltonovi kvaternioni itd. Od navedenih, u robotici se najčešće koristi homogena transformacija koordinatnih sustava temeljena na realnim matricama dimenzija 4x4. Prema Denavit i Hartenberg-u za opću transformaciju između dvaju zglobova potrebna su četiri parametra. Ti parametri, znani kao DH parametri postali su standard pri opisu kinematike robota. Iako kvaternioni opisuju rotacije koordinatnih sustava, mogli bismo reći, elegantnije, više se primjenjuju homogene transformacije zbog svoje jednostavnosti. Primjerice dok orijentaciju određenog tijela u matrici homogene transformacije predstavlja čak devet elemenata matrice, dvostrukim kvaternionom potrebna su samo 4 elementa za opis [1].

Kinematika robota se dijeli na direktnu kinematiku i inverznu kinematiku. Rješavanje problema direktne kinematike je relativno jednostavno i ne zahtjeva izvođenje kompleksnih jednadžbi. Zbog toga uvijek postoji direktno rješenje kinematike manipulatora. Računanje inverzne kinematike je složeniji problem od računanja direktne kinematike. Generalno govoreći rješenje inverzne kinematike je računski zahtjevno te je potrebno "puno" procesnog vremena ukoliko se želi primijeniti za upravljanje robotskim manipulatorom u realnom vremenu. Također postoje mnoge singularnosti i nelinearnosti koje problem inverzne kinematike dodatno otežavaju. Odnos između direktne i inverzne kinematike prikazan je na slici (**Slika 12**).

Postoje dvije glavne metode kojima se može riješiti problem inverzne kinematike. To su analitička i numerička metoda. U ovom diplomskom radu primijenjena je analitička metoda stoga numerička neće biti razmatrana.

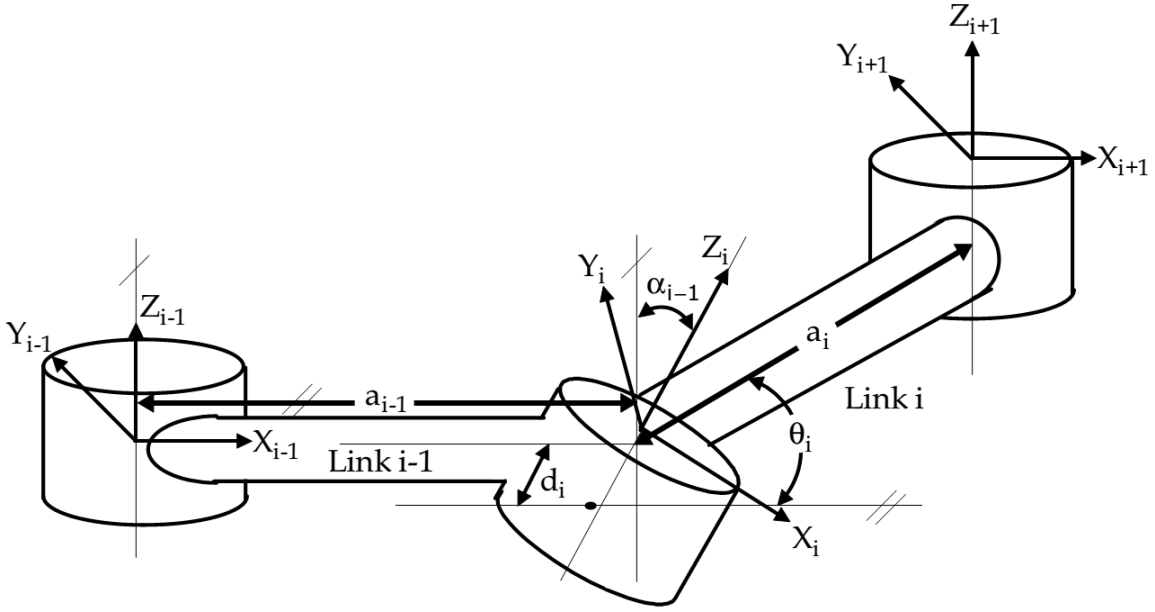
Slika 12. Blokovski prikaz direktne i inverzne kinematike

U analitičkoj metodi postoje dva pristupa: geometrijski i algebarski. Ovdje je primijenjen geometrijski pristup. Geometrijski pristup je odabran prvenstveno zbog toga što uzima najmanje procesnog vremena, koje je veoma bitno kod aplikacija koje se izvode u realnom vremenu. U nastavku ovog poglavlja bit će detaljnije opisana direktna i inverzna kinematika uz primjenu na robotskom manipulatoru KUKA LWR4+.

3.1. Direktna kinematika

Svaki manipulator je sastavljen od članaka koji su pričvršćeni međusobno revolutnim ili prizmatičnim zglobom, od baze manipulatora do članka koji služi za prihvat alata. Upravo se računanje pozicije i orijentacije članka za prihvat alata, uz zadane kuteve zakreta svih zglobova, naziva direktna kinematika. Kako bi stvorili kinematički model robota koristi se Denavit-Hartenbeg-ova metoda koja koristi četiri parametra za opis kinematike robota. To su: kut zakreta zgloba oko Z osi (θ_i), odmak članka po Z osi (d_i), zakret članka oko X osi (α_i) i odmak članka po X osi (a_i). Koordinatni sustav je postavljen na svaki zglob kako bismo mogli odrediti DH parametre. Z_i os koordinatnog sustava pokazuje smjer rotacije ili translacije određenog zgloba. Prema DH zapisu će se manipulator od N zglobova, koji su numerirani od *1* do *N*, sastojati od *N+1* članaka numeriranih od *0* do *N*. Članak *0* je baza manipulatora a članak *N* služi za prihvat alata. Na slici je prikazan postupak određivanja DH parametara. Članak *i* te članak *i-1*, povezani su zglobom *i*. Može se reći da zglob *i* zakreće članak *i* oko osi Z_i. Članak je kruto tijelo koje određuje prostorni odnos između dva susjedna

zgloba. Definira se preko pomaka članka d_i i kuta zakreta zgloba θ_i. Zakret članka α_i i duljina članka a_i su uvijek konstantni dok su pomak članka d_i i kut zakreta zgloba θ_i promjenjivi. Na slici možemo vidjeti DH parametre.



Slika 13. Denavit - Hartenberg parametri [1]

Kako bi kinematički model robota bio točan, potrebno je, između ostalog, poznavati točne dimenzije robota kako bi mogli točno odrediti DH parametre. Od izuzetne je važnosti da su dimenzije robota točno definirane u kinematičkom modelu jer u suprotnom se stvara pogreška. Odmak članka d_i iščitava se iz dokumentacije samoga robota. Vrijednosti d_i su ujedno i udaljenost između zglobova po *Z* osi. Kako, zbog geometrije KUKA LWR4+ robota centri rotacije svih zglobova stoje na istom pravcu, možemo zaključiti da su svi parametri a_i jednaki nuli. Pregledavajući literaturu više autora zaključio sam da postoje dva redoslijeda kojim je moguće pravilno definirati transformaciju iz jednog koordinatnog sustava u drugi po četiri DH parametra. Prvi redoslijed je $R_x T_x R_z T_z$, koji je korišten u radu, dok je drugi $R_z T_z T_x R_x$. Gdje *R* označava rotaciju, a *T* translaciju koordinatnog sustava po *X* ili *Z* osi.

Dakle transformacija *i*-tog zgloba jednaka je:

$$ {}^{i-1}_{i}T = R_x(\alpha_{i-1}) * T_x(a_{i-1}) * R_z(\theta_i) * T_z(d_i) $$

$$ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$= \begin{matrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d\_i \\ 0 & 0 & 0 & 1 \end{matrix}$$

Konkretno za robotski manipulator KUKA LWR4+ matricu transformacije dobijemo kao umnožak transformacije koordinatnog sustava od baze do zadnjeg članka za prihvat alata:

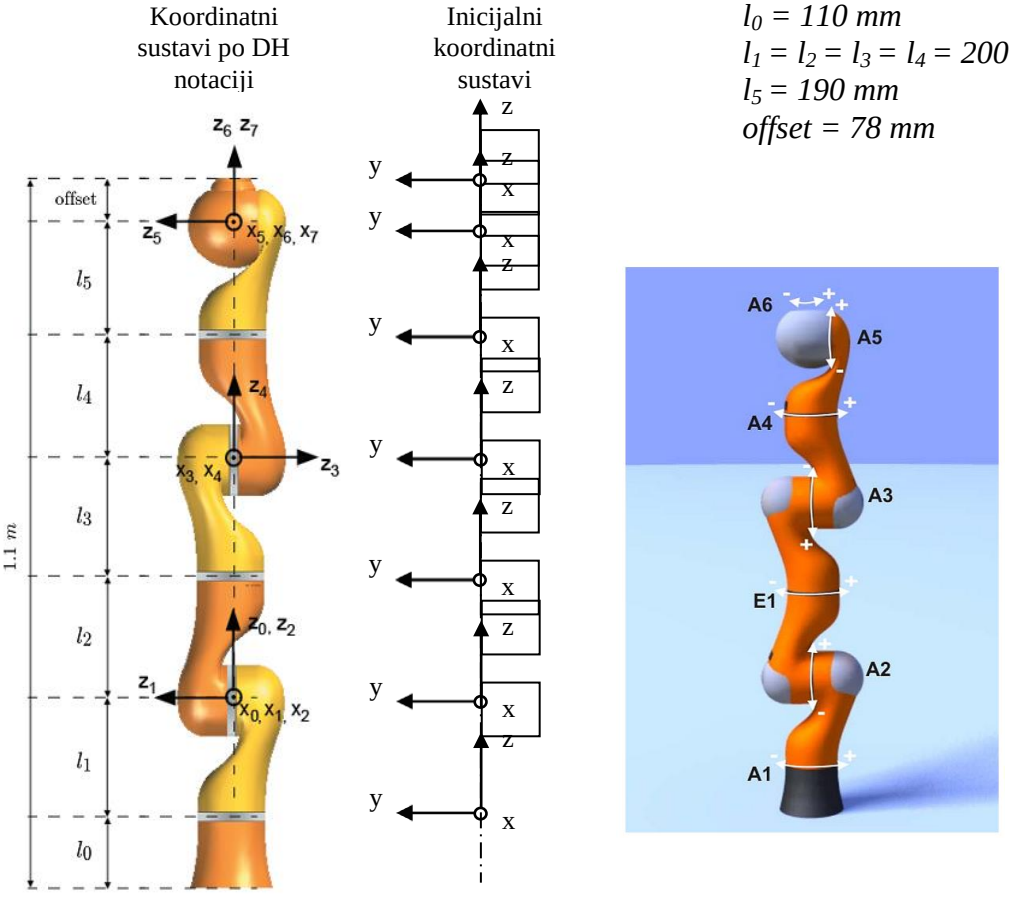
$$^{7}_{0}T = {}^{1}_{0}T^{2}_{1}T^{3}_{2}T^{4}_{3}T^{5}_{4}T^{6}_{5}T^{7}_{6}T.$$

Alternativno matrica transformacije $^{7}_{0}T$ može se zapisati kao:

$$^{7}_{0}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Gdje r_{kj} predstavljaju rotacijske elemente matrice transformacije, a $p_x, p_y \, i \, p_z$ predstavljaju vektor pozicije.

Na slici (**Slika 14**) možemo vidjeti inicijalne koordinatne sustave svakoga zgloba i koordinatne sustave nakon određivanja DH parametara. Koordinatni sustav prvog zgloba je pomaknut u koordinatni sustav drugog zgloba jer je lakše izvesti transformaciju. Isto je izvedeno i sa ostalim zglobovima. Određeni DH parametri dani su u tablici (**Tablica 5**).



Slika 14. Određivanje DH parametara

Tablica 5. Iznos DH parametara za robotsku ruku KUKA LWR4+

oznaka zgloba	α_i [°]	a_i [mm]	θ_i [°]	d_i [mm]
A1	0	0	0	310
A2	90	0	-90	0
A3	-90	0	0	400
A4	-90	0	0	0
A5	90	0	0	390
A6	90	0	0	0
E1	-90	0	0	78

Matematički model direktne kinematike izveden je u programskom sučelju MATLAB zbog jednostavnosti matričnog računa. Za potrebe programske podrške kasnije je programski kod izveden u C++ programskom sučelju te se on nalazi u prilogu.

```
j=[0 0 0 0 0 0 0];%ulazni parametri - zakret zglobova
j(3)=j(3)+90;%offset
fi=j*pi/180;%pretvorba u radijane
alfa=[0 pi/2 -pi pi/2 pi/2 -pi/2];
a=[0 0 400 0 0 0];
d=[310 0 0 390 0 78];
%matrica rotacije T=Rx*Tx*Rz*Tz
for i=1:6
    T{i}=[cos(fi(i)) -sin(fi(i)) 0 a(i);
          cos(alfa(i))*sin(fi(i))cos(alfa(i))*cos(fi(i))      -
          sin(alfa(i)) -d(i)*sin(alfa(i));
          sin(alfa(i))*sin(fi(i))cos(fi(i))*sin(alfa(i))
          cos(alfa(i)) d(i)*cos(alfa(i));
          0 0 0 1];
end
polozaj=T{1}*T{2}*T{3}*T{4}*T{5}*T{6};
```

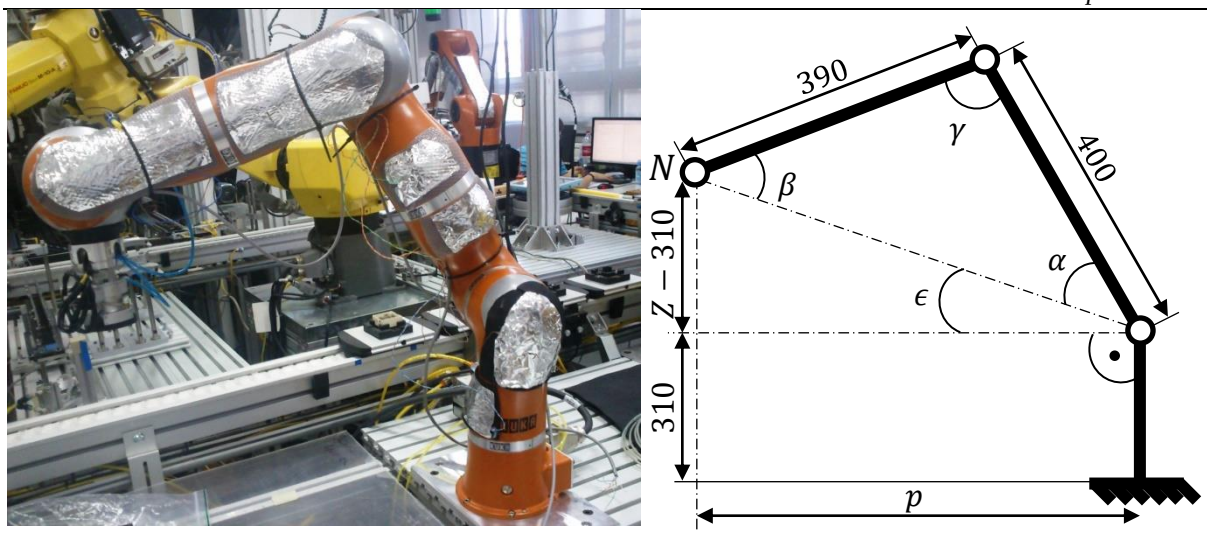
3.2. Inverzna kinematika

Kako bi riješili problem inverzne kinematike potrebno je za danu poziciju i orijentaciju vrha alata tj. TCP-a (Tool Center Point) odrediti kut zakreta svih zglobova. U slučaju da se na kraju kinematičkog lanca ne nalazi nikakav alat potrebno je izvesti još jednu transformaciju koja će dovesti do stvarnog TCP-a. Kako robotom u suštini možemo upravljati samo po kutevima zakreta zglobova od iznimne je važnosti odrediti ih, kako bi mogli položaj TCP-a, zadan u kartezijskom koordinatnom sustavu prevesti u odgovarajuće zakrete zglobova. Kod direktne kinematike za svaku kombinaciju kuteva zakreta zglobova postoji jednoznačno rješenje dok je računanje inverzne kinematike daleko kompleksnije zbog slijedećih razloga:

1. Jednadžbe koje se trebaju riješiti su uglavnom nelinearne te stoga nije uvijek moguće dobiti konačno rješenje
2. Moguća su višestruka rješenja
3. Beskonačan broj rješenja je moguć kod kinematički redundantnog manipulatora
4. Moguća su neprihvatljiva rješenja zbog kinematičke strukture manipulatora

Problem višestrukih rješenja ovisi poglavito o broju stupnjeva slobode gibanja i o geometriji samog robotskog manipulatora. Za manipulator sa 6 SSG-a koji teoretski nema mehaničkih ograničenja, postoji 16 prihvatljivih rješenja. Zbog toga se moraju odrediti kriteriji po kojima će se odabrati najbolje rješenje. Kako postoje mehanička ograničenja u zglobovima manipulatora smanjuje se broj mogućih višestrukih rješenja za realni manipulator. Za manipulator sa sedam SSG-a, kao što je KUKA LWR4+, postoji mnogo višestrukih rješenja i postaje jako kompleksno izračunati inverznu kinematiku. Zbog toga je u ovom radu zanemaren zglob E1 na način da je on softverski "zaključan" na kut zakreta 0°. Na taj način taj zglob se može zanemariti i u direktnoj kinematici manipulatora.

Kako ne postoji mogućnost *low-level* upravljanja robotom u kartezijskom koordinatnom sustavu već samo upravljanje po kutevima zakreta zglobova, potrebno je odrediti iz homogene matrice transformacije prirubnice robota kuteve zakreta zglobova. Prvo je potrebno odrediti prva tri stupnja slobode gibanja.



Slika 15. Proračun inverzne kinematike robota

Na slici (**Slika 15**) je prikazana geometrija robota. Kutevi prva tri zgloba jednoznačno određuju poziciju točke N. Primijetite da ova konfiguracija robota za točku N nije unikatna. Robot je trenutno u elbow-up poziciji no on se mogao postaviti npr. u elbow-down poziciju. Ova konfiguracija je izabrana jer je najčešća tokom korištenja robota u laboratoriju zbog većih manevarskih sposobnosti. Kako bi dobili poziciju točke N moramo je transformirati po Z osi koordinatnog sustava prirubnice za iznos *d*.

Kad izračunamo poziciju točke N možemo izračunati vrijednosti iznosa prva tri kuta zakreta zglobova $\theta_1, \theta_2, \theta_3$. Najlakše je odrediti iznos kuta θ_1. Računamo ga prema formuli:

$$\theta_1 = atan2(N_y, N_x).$$

Matematička funkcija koju koristimo za izračun eulerovih kuteva je inverzna tangens funkcija *atan2*. Kako bi predznak mogao biti određen, a vrijednost izračunatog kuta spremljena za sva četiri kvadranta nužno je korištenje ove funkcije. Također za razliku od klasične inverzne tangens funkcije, sa *atan2* je moguće dobiti rješenje u slučaju $N_y = 0$. Kuteve zakreta θ_2 i θ_3 je malo kompliciranije odrediti pa je potrebno detaljno razraditi geometriju robota (Slika 4). Prvo je potrebno odrediti kut ϵ i udaljenost točke N od ishodišta *world* koordinatnog sustava.

$$p = \sqrt{N_x^2 + N_y^2},$$

$$\epsilon = atan2(Nz - 310, p).$$

Kako bi izračunali α i γ potrebno je prvo izračunati stranicu *c* trokuta. Stranice *a* i *b* su nam već poznate i one iznose 390 mm i 400 mm tako da možemo izračunati stranicu *c:*

$$c = \sqrt{(N_z - 300)^2 + p^2}.$$

Dodatno je izračunat poluopseg trokuta s i radijus upisane kružnice trokuta r izveden iz Heronove formule za trokut.

$$s = \frac{a+b+c}{2};$$

$$r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}};$$

Sada se mogu odrediti željeni kutevi α i γ danoga trokuta:

$$\alpha = 2atan2\big(r, (s-a)\big);$$

$$\gamma = 2atan2\big(r, (s-c)\big);$$

Kako bi dobili konačnu vrijednost kuta θ_2 i kuta θ_3 potrebno je određene parametre zbrojiti:

$$\theta_2 = \alpha + \epsilon,$$

$$\theta_3 = \pi - \gamma.$$

Sada je potrebno odrediti preostala tri kuta zakreta zglobova θ_4, θ_5 i θ_6 koji u biti određuju orijentaciju zadane točke što nas navodi da su to u stvari Eulerove rotacije *A*, *B* i *C*. Kako bismo Eulerove kuteve mogli izračunati potrebno je prvo izračunati direktnu kinematiku robota za prva tri kuta zakreta.

$${}^3_0T = {}^1_0T^2_1T^3_2T.$$

Budući da je sada poznata točna pozicija zadane točke ${}^6_0T$ i izračunali smo poziciju i orijentaciju trećeg zgloba, moguće je dobiti matricu transformacije iz koje se mogu izračunati Eulerovi kutevi tj. kutevi zakreta θ_4, θ_5 i θ_6.

$${}^6_3T = inv\ {}^3_0T\ {}^6_0T$$

Računamo kuteve zakreta:

$$\theta_4 = atan2\big({}^6_3T(11),\ {}^6_3T(9)\big),$$

$$\theta_5 = atan2\left(\sqrt{{}^6_3T^2(6) + {}^6_3T^2(2)},\ {}^6_3T(10)\right),$$

$$\theta_6 = atan2\big({}^6_3T(6), -{}^6_3T(2)\big).$$

3.3. Usporedba rješenja

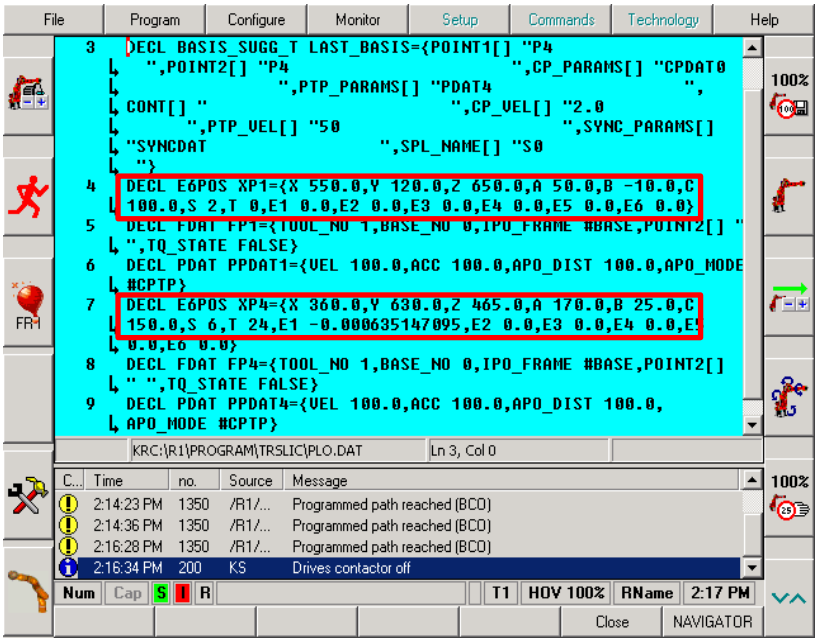
U ovom poglavlju je dana usporedba postizanja zadane pozicije robota definirane, u prvom slučaju na upravljačkom računalu robota, a u drugom slučaju na korisničkom računalu. Potrebno je dakle dokazati da je inverzni kinematički model robota, koji je implementiran na korisničkom računalu, ispravan. Kod inverznog rješenja u cijelosti je dan u prilogu. Definirani

su položaj i orijentacija dviju točaka u koje robotski manipulator mora doći (**Tablica 6**). Nakon pokretanja programa robot mora otići u zadanu poziciju i zauzeti zadani položaj.

Tablica 6. Zadane pozicije položaja robota

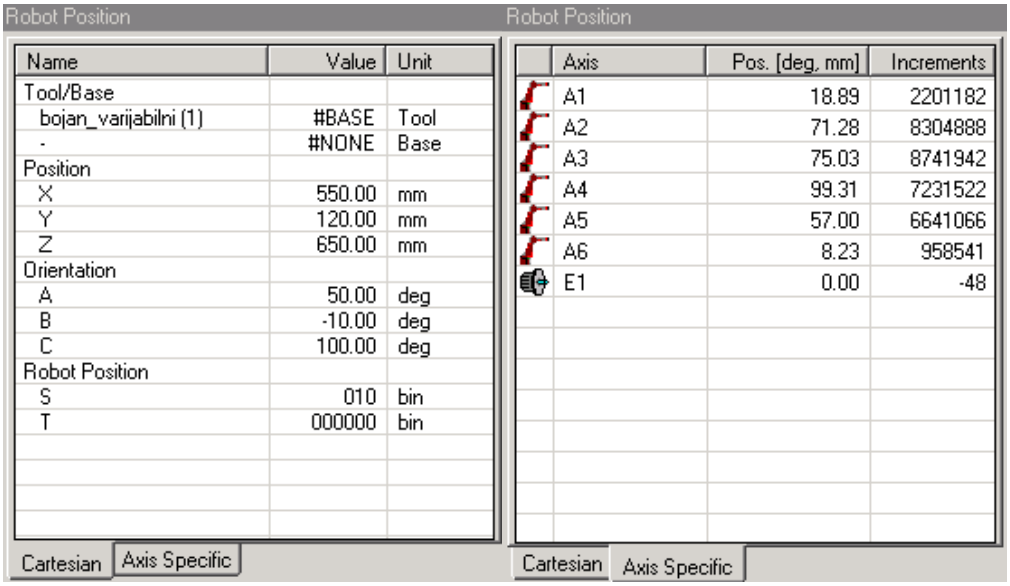
Zadane pozicije robota			
Pozicija 1		Pozicija 2	
X	550 mm	X	360 mm
Y	120 mm	Y	630 mm
Z	650 mm	Z	465 mm
A	50 deg	A	175 deg
B	-10 deg	B	25 deg
C	100 deg	C	150 deg

Nakon što se robotski manipulator zaustavi, očitat ćemo rezultate sa privjeska za učenje i usporediti sa onima upisanim u korisničko računalo.



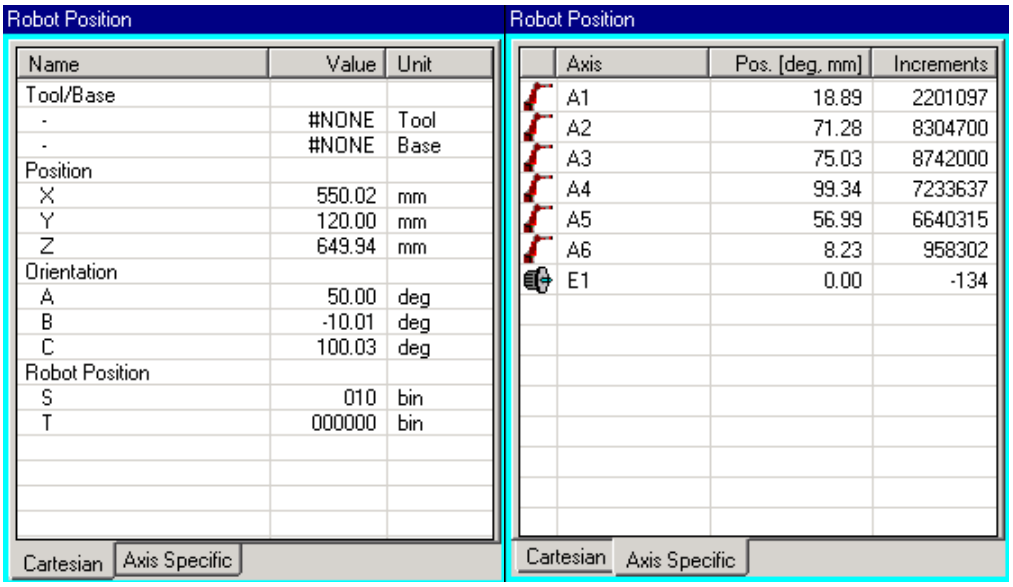
Slika 16. Pozicije definirane u programskom sučelju robota

Na upravljačkom računalu robota, u KRL-u (Kuka Robot Language), robot je programiran da dođe u dvije pozicije definirane u *world* koordinatnom sustavu što je prikazano na slici (**Slika 16**). Na slici (**Slika 17**) prikazano je da robot bez odstupanja dolazi u poziciju 1 zadanu na računalu robota.



Slika 17. Robot u prvoj poziciji definiranoj na KRC-u

Kako bi provjerili ispravnost inverznog i direktnog matematičkog modela robota, koji je implementiran u programskoj podršci, potrebno je robot, u korisničkom sučelju, poslati u iste točke kao i u prethodnom slučaju. Snimljeni rezultati za prvu poziciju prikazani su na slici (**Slika 18**)



Slika 18. Robot u prvoj poziciji definiranoj na korisničkom računalu

Može se primijetiti da postoji određeno odstupanje između zadane i pozicije robota upravljanje korisničkim računalom. No navedena odstupanja su veoma mala pa možemo smatrati da je matematički model prihvatljiv. U tablici (**Tablica 7**) su dani usporedni rezultati postignutih pozicija.

Tablica 7. Usporedba rješenja

Zadana pozicija 1		Zadana pozicija 2		Postignuta pozicija 1		Postignuta pozicija 2	
X	550 mm	X	360 mm	X	550.02 mm	X	360.04 mm
Y	120 mm	Y	630 mm	Y	120.00 mm	Y	630.03 mm
Z	650 mm	Z	465 mm	Z	649.94 mm	Z	464.77 mm
A	50 deg	A	175 deg	A	50.00 deg	A	175.01 deg
B	-10 deg	B	25 deg	B	-10.01 deg	B	25.01 deg
C	100 deg	C	150 deg	C	100.03 deg	C	150.03 deg

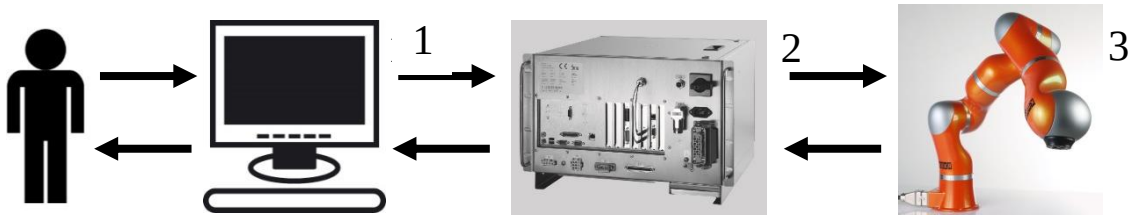
4. PROGRAMSKA PODRŠKA

U ovom poglavlju je objašnjen tok izrade programske podrške. Nakon početnog proučavanja dokumentacije robota, UDP protokola, upoznavanja sa C++ programskim jezikom i Fast Research Interface knjižnicom, koja nam omogućava programsko sučelje sa robotom, pristupilo se izradi programske podrške.

4.1. Struktura sustava

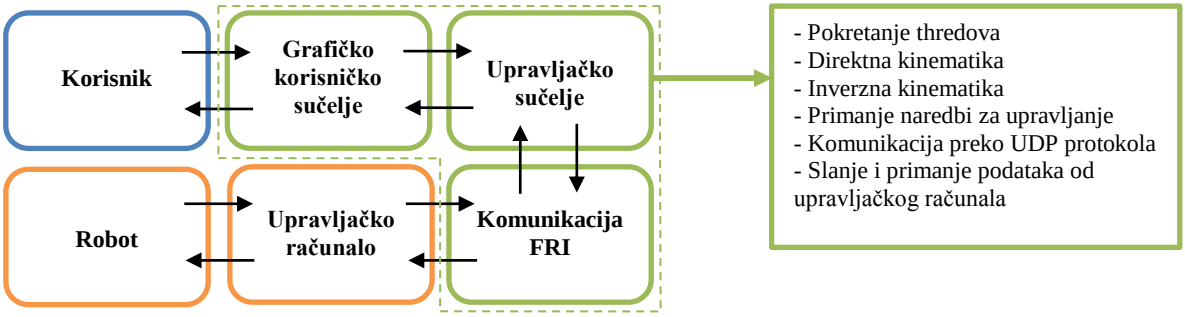
Kako bi bilo olakšano praćenje, na slici (**Slika 19**) možemo vidjeti strukturu sustava. Glavni dijelovi sustava su:

1. Računalo koje korisniku prikazuje podatke o sustavu i koje od korisnika prima naredbe
2. Upravljačko računalo koje podatke prikupljene sa različitih senzora robotskog manipulatora obrađuje i šalje u korisničko računalo
3. Robotski manipulator KUKA LWR4+



Slika 19. Struktura sustava

Računalo možemo podijeliti na tri podsustava prikazana na slici (**Slika 20**), a to su: korisničko sučelje, upravljački dio i komunikacijski dio koji se sastoje od više funkcija koje služe za izračun direktne i inverzne kinematike, za primanje naredbi, za upravljanje te za komunikaciju sa korisnikom i KUKA LWR4+ robotom.



Slika 20. Shematski prikaz sustava

4.2. Komunikacija između računala i upravljačkog računala robota.

Komunikacija između računala i upravljačkog računala robota ostvarena je u *fri.cpp* putem UDP protokola koji je detaljnije objašnjen u drugom poglavlju. Uspostavljanje veze je postignuto FRI programskom knjižnicom koja je dobivena od strane proizvođača.

Prvi korak je postavljanje statične IP adrese korisničkog računala što se jednostavno postiže isključivanjem automatskog postavljanja adrese te definiranjem adrese, u mome slučaju 192.168.1.60. Nakon što se postavila statična IP adresa korisničkog računala, potrebno je upravljačko računalo postaviti u "Monitor mode" način rada.

```
if ( friInst.getQuality() >= FRI_QUALITY_OK)
      {
            // send a second marker
            friInst.setToKRLInt(0,10);
      }

if (friInst.doReceiveData()>=0)
      {
            kretanje();
            friInst.doSendData();
       }
 else
       {
            cout << "greska!";
       }
```

Slika 21. Otvaranje veze preko UDP protokola

Na slici (**Slika 21**) možemo vidjeti dio koda koji služi za uspostavljanje veze preko UDP protokola. Prva naredba *friInst.getQuality() >= FRI_QUALITY_OK* služi kako bi se ostvario *handshake* sa KRL-om (Kuka Robot Language) koji se nalazi na upravljačkom računalu robota. Naredbom *friInst.setToKRLInt(0,10);* poslan je marker tj, set podataka te ukoliko se naredbom *friInst.doReceiveData()>=0* primi integer veći ili jednak nuli, može se krenuti sa izvođenjem funkcije *kretanje()*. U suprotnom korisnik će biti obaviješten da je došlo do pogreške u izvođenju koda. Funkcija *kretanje()* (u prilogu) izvršava naredbu *friInst.doPositionControl(cmdJntPos,false);* koja šalje iznose zadanih kuteva zakreta robota upravljačkom računalu robota.

4.2.1. Upravljački dio sučelja

Upravljački dio koda se izvršava u *command.cpp* te on sadrži esencijalni dio programske podrške. To su funkcije *forward(), inverzi()* i *command().*

4.2.1.1. Funkcija forward()

Funkcija *forward()* sadrži direktan kinematički model robota KUKA LWR4+ te je korištena kako bi se mogli izračunati eulerovi kutevi nultog koordinatnog sustava (eng. *null frame*) ili koordinatnog sustava alata (eng.*tool frame*). Na slici (**Slika 22**) je prikazan dio funkcije *forward()*. Na početku su definirani Denavit-Hartenberg parametri koji su određeni i detaljno objašnjeni u prethodnom poglavlju. Zbog lakšeg i preglednijeg računanja korišten je matrični račun. Kako bi bilo kompliciranо i zahtjevno programirati funkciju koja računa sa dvodimenzionalnim poljima (matricama), izračun je izveden pomoću programske knjižnice *eigen* koja je napravljena upravo za tu svrhu.

```
float * forward(float jnt_pos[6])
{
	float alfa[6]={0, PI/2, -PI+(0.000679*PI/180), PI/2, PI/2, -PI/2};
	float a[6]={0, 0, 400, 0, 0, 0};
	float d[6]={310, 0, 0, 390, 0, 78};

	Matrix4f T0(4,4);
	T0 << cos(fi[0]), -sin(fi[0]), 0, a[0],
	 cos(alfa[0])*sin(fi[0]), cos(alfa[0])*cos(fi[0]), -sin(alfa[0]), -
	d[0]*sin(alfa[0]),
	sin(alfa[0])*sin(fi[0]), cos(fi[0])*sin(alfa[0]), cos(alfa[0]),
	d[0]*cos(alfa[0]),
	0, 0, 0, 1;
	.
	.
	.
	.
	T=T0*T1*T2*T3*T4*T5;

	static float eulerkut[3];
	 eulerkut[0]=atan2(T(1,0),T(0,0));
	 eulerkut[1]=-atan2(T(2,0),sqrt(pow(T(2,1),2)+pow(T(2,2),2)));
	 eulerkut[2]=atan2(T(2,1),T(2,2));

	return eulerkut;
}
```

Slika 22. Funkcija forward()

Izračunate su matrice transformacija za zglobove *T0* do *T5*, te je na kraju izračunata ukupna matrica transformacije *T* koja je umnožak matrica transformacija svih zglobova robotskog manipulatora. Nakon toga izračunati su Eulerovi kutevi kako bi dobili točnu orijentaciju posljednjeg članka manipulatora. Varijabla *eulerkut* koju vraćamo je definirana kao *static* kako bi zadržala svoju vrijednost do idućeg poziva funkcije. Matematička funkcija koju koristimo za izračun Eulerovih kuteva je inverzna tangens funkcija *atan2*, kako bi

predznak mogao biti određen, a vrijednost izračunatog kuta spremljena za sva četiri kvadranta. Dakle funkcija *atan2* daje rezultate u intervalu $[-\pi,\pi]$.

4.2.1.2. Funkcija inverzi()

U funkciji *inverzi()* je sadržan inverzni kinematički model robota KUKA LWR4+ koji je izveden u prethodnom poglavlju. Inverzni kinematički model je neophodan kako bi upravljanje robota u kartezijevom koordinatnom sustavu moglo biti izvršeno. Dio funkcije *inverzi()* prikazan je na slici (**Slika 23**). Prvo je potrebno definirati matricu transformacije alata koji se nalazi na robotu. Ukoliko nema priključenog alata, vrijednosti x_1, y_1, z_1, w_1, p_1, r_1 su postavljene u nulu. Kako je priključeni alat, u konkretnom slučaju, orijentiran jednako kao i prethodni članak robota, onda su vrijednosti eulerovih kuteva w_1, p_1, r_1 jednaki nuli te su samo učitane koordinate vrha alata.

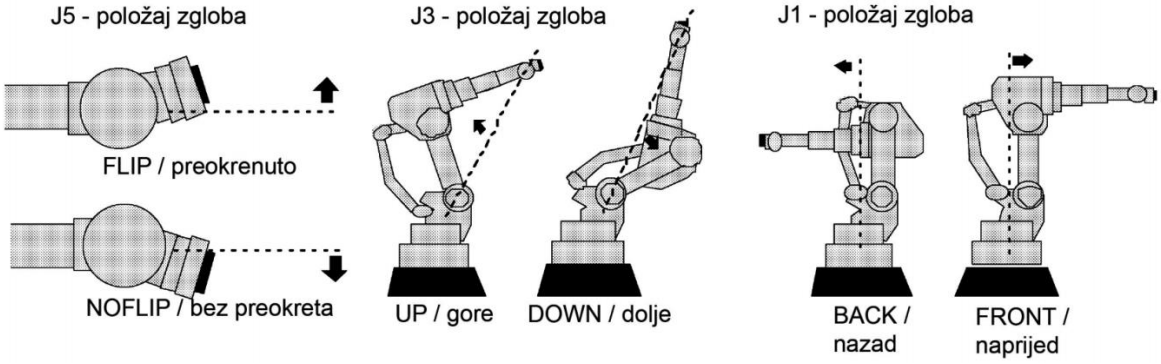
```
float * inverzi(float pos_r[6]){
	//Tool  na robotu
	float w1=0, p1=0, r1=0, x1=-1.31, y1=1.26, z1=191.2;
	Tool1=Trans01*Rtz1*Rty1*Rtx1;

	//User frame UF na robotu
	float w=0, p=0, r=0, x0=0, y0=0, z0=0;
	UF=Trans0*Rtz*Rty*Rtx;

	//POZICIJA ROBOTA
	x0=pos_r[0]; y0=pos_r[1]; z0=pos_r[2];
	w=pos_r[5]-PI; p=pos_r[4]; r=pos_r[3];
	T0=UF*Trans0*Rz*Ry*Rx;

	//definiramo matricu za zadnji zglob pomaknut 78mm po Z osi
	Matrix4f zglob(4,4);
	zglob << 1, 0, 0, 0,
		 0, 1, 0, 0,
		 0, 0, 1, 78,
		 0, 0, 0, 1;
	Matrix4f T(4,4);

	T=T0*zglob*Tool1.inverse();

	float Nx=T(0,3);
	float Ny=T(1,3);
	float Nz=T(2,3);
```

Slika 23. Funkcija *inverzi()*

Definiran je koordinatni sustav korisnika (eng. *user frame*) koji je u konkretnom slučaju postavljen u ishodište *world* koordinatnog sustava, dok je varijabla *pos_r[i]*, argument funkcije *inverzi()* te je u njoj pohranjena pozicija robota učitana od strane korisnika. Nakon što su definirani korisnički koordinatni sustav, koordinatni sustav alata i zadnjeg članka

manipulatora, može se izvesti matrica transformacije *T* točke *N*, prikazane na slici (**Slika 15**) u prethodnom poglavlju. Iz matrice transformacije *T* direktno se mogu dobiti X,Y i Z koordinate točke *N*. Nakon toga slijedi proračun kuteva zakreta zglobova robota čiji pregled je također dan u prethodnom poglavlju. Dok je problem rješavanja direktne kinematike robota jednoznačan, problem inverzne kinematike robota je puno zahtjevniji za izračunati jer ne postoji jedno, univerzalno rješenje

Robot sa šest stupnjeva slobode gibanja se teoretski u prostoru za bilo koju točku može postaviti u 2^3 tj. u 8 konfiguracija kao što je prikazano na slici. Proračunom je dobivena samo jedna konfiguracija robota, ali iz nje se mogu izvesti preostalih sedam.



Slika 24. Različite konfiguracije robota [referenca IMS vjezbe2]

Na slici je prikazan dio koda za određivanje druge konfiguracije robota. U varijablu *Inverse1* je spremljena izračunata konfiguracija zglobova. Druga konfiguracija, prikazana na slici (**Slika 25**), za istu orijentaciju koordinatnog sustava alata dobivena je na način da su zglobovi A4 i A6 zarotirani za 180° dok je zglobu A5 promijenjen predznak. Na sličan način određene su i preostale konfiguracije robota.

U funkciji *inverzi()* izveden je i algoritam odlučivanja koja je konfiguracija robota najbolja. Algoritam funkcionira na način da se prvo provjeri da li je moguće da robot za iznose svih zglobova može zauzeti zadani položaj. To je provjereno na način da se uspoređuju vrijednosti dobivene konfiguracije sa minimalnom i maksimalnom vrijednosti kuta koju neki zglob može zauzeti. Ukoliko je izračunata vrijednost zgloba veća ili manja od dozvoljene, koja je određena od strane proizvođača, onda se ta vrijednost "kažnjava" na način da joj se dodjeljuje vrijednost 10000. Na taj je način u slijedećem koraku omogućena usporedba rješenja i može se donijeti odluka koja je konfiguracija robota najbolja.

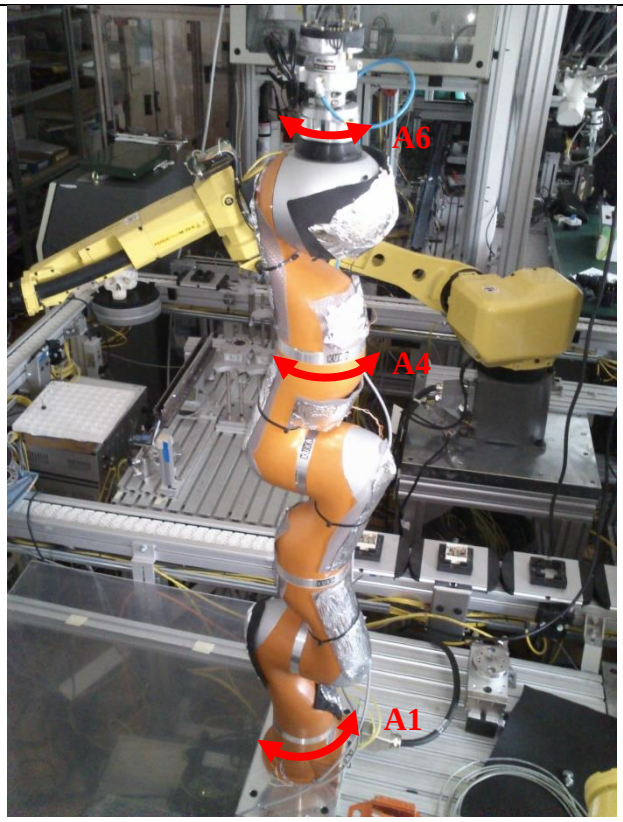
U pomoćnu varijablu *po1* spremljen je zbroj apsolutne vrijednosti razlike trenutne i proračunom dobivene vrijednosti konfiguracije robota. Na kraju se usporede sve pomoćne

varijable i odabere se konfiguracija koja ima najmanju pomoćnu varijablu. Kao dodatni kriterij mogu se uvesti određene "težine" na svaki zglob, gdje bi primjerice zglob A1 trebao imati najveću težinu zbog više kriterija. Ako se za kriterij upravljanja uzme ciljna funkcija minimuma energije, a zna se da je motor prvog zgloba najsnažniji, onda je logično da njegova promjena ima najveću "težinu" za sustav. Također se zakretom prvoga zgloba manipulatora cijela konstrukcija robota zakreće, što u većini slučajeva nije praktično zbog relativno skučenih i nestrukturiranih okolina u kojima se robot nalazi.

```
float inverse1 [6]={J1, J2, J3, J4, J5, J6};//PRVI POLOZAJ
float J11, J22, J33, J44, J55, J66; //pomocne varijable
//%%poz 2 elbow up, flip J4,J5,J6
if (J4>=0)
       J44=J4-180;
if (J4<0)
       J44=J4+180;
J55=-1*J5;
if (J6>=0)
       J66=J6-180;
if (J6<0)
       J66=J6+180;
float inverse2 [6]={J1, J2, J3, J44, J55, J66};//DRUGI POLOZAJ
.
.
.
float ogranicenja[6]={170, 120, 120, 170, 130, 170};
for (i=0; i<6; i++){
       if(abs(inverse1[i])>ogranicenja[i])
              inverse1[i]=10000;
       pom1[i]=trenutno[i]-inverse1[i];
       pom1[i]=abs(pom1[i]);
       po1=po1+pom1[i];
```

Slika 25. Određivanje druge konfiguracije robota

Za određene položaje robota postoji beskonačno mnogo rješenja što dovodi do singularnosti u proračunu. Na slici (**Slika 26**) možemo vidjeti jedan takav položaj gdje postoji beskonačno rješenja za zglobove A1, A4 i A6. Na primjer ukoliko je ruka robotskog manipulatora ispružena kao na slici i ako želimo da prihvatnica robota ne rotira tj. da ostane u istom položaju, moguće je zarotirati zglob A1 za 10°, a zglob A4 za -10° ili pak zglob A6 možemo zarotirati za 50°, a zglobove A4 i A1 za -30°i -20°. Dakle postoji nebrojeno kombinacija i dokle god će, za dani slučaj, zbroj relativnog zakreta zgloba A1, A4 i A6 biti jednak nuli, prihvatnica robota neće promijeniti orijentaciju.



Slika 26. Robot u računski singularnom položaju

4.2.1.3. Funkcija command()

Funkcija *command()* komunicira i razmjenjuje podatke sa grafičkim korisničkim sučeljem i sa dijelom programa zaduženim za komunikaciju sa upravljačkim računalom KUKE. Ova funkcija se vrti u beskonačnoj *while* petlji kako bi u svakom trenutku mogla prihvatiti i odaslati podatke. U funkciji su korištene predefinirane funkcije Fast Research Interface-a kao što su . *friInst.getMsrCmdJntPosition()[i];* koja služi za dohvaćanje trenutne pozicije robota zapisane u kutevima zakreta zglobova.

```
while (pomocni2[0]!=mojjoint[0] || pomocni2[1]!=mojjoint[1]+PI/2...)
	{
		if( pomocni2[0]-mojjoint[0]<0.0005 && pomocni2[0]-mojjoint[0]>-0.0005
		{
		break;
		Sleep(1);
		}
		if(pomocni2[0]>mojjoint[0]){
		if(abs(mojjoint[0]-pomocni2[0])<brzina2)
		mojjoint[0]=mojjoint[0]+abs(mojjoint[0]-pomocni2[0]);
		else
		mojjoint[0]=mojjoint[0]+brzina2;}
	}
```

Slika 27. Regulacija položaja

Funkcijom *friInst.getMsrCartPosition()[i];* može se dohvatiti matrica transformacije *world* koordinatnog sustava za *null frame*. Također je u funkciji izvedena i regulacija položaja prikazana na slici (**Slika 27**). U varijabla *pomocni2[i]* sprema se trenutna pozicija robota dok je u varijabli *mojjoint[i]*spremljena zadana pozicija robota. Dok je zadana pozicija različita od trenutne, izvršava se kod u *while* petlji. Prvi uvjet će prekinuti izvršavanje koda ukoliko je razlika između zadane i trenutne pozicije zgloba robota manja od 0.0005 radijana. Ostali uvjeti služe kako bi se robot usporilo kad dolazi do zadane pozicije.

4.2.2. Grafičko korisničko sučelje (GUI)

Grafičko sučelje je izvedeno u programskom paketu Visual Studio 2010 te je korištena Windows Application Forms knjižnica za izradu grafičkog sučelja koja je dio Microsoft .NET frameworka. Sučelje se u svojoj suštini temelji na C++ kodu. Na slici (**Slika 28**) je prikazano grafičko sučelje.

Slika 28. Grafičko korisničko sučelje

U crvenom dijelu prozora moguće je u svakom trenutku očitati položaj u kojem se robot nalazi. Položaj se ispisuje u kutevima zakreta svakog zgloba robotskog manipulatora te u kartezijskom koordinatnom sustavu robota.

U *Frame control* dijelu prozora moguće je upravljati u kartezijskom koordinatnom sustavu robotom na način u prazan polja upišemo koordinate željene točke, odaberemo na koji se to koordinatni sustav odnosi (*world, tool1, tool2*), odredimo brzinu te pritisnemo tipku *Go* kako bi pokrenuli robota. Ako se pritisne tab *Direct* onda je moguće upravljati u koordinatnom sustavu robota po X,Y i Z osi direktno pritiskom na gumb + ili – s tim da je prethodno potrebno definirati korak i brzinu kojom će se robot kretati.

U *Joint control* dijelu prozora moguće je upravljati robotom po pojedinom zglobu robotskog manipulatora i to na način da upišemo kuteve zakreta svih zglobova te pritisnemo tipku *Go*. Ukoliko se odabere tab *Direct* onda je moguće direktno upravljanje zglobovima pritiskom na tipku + i -.

Grafičko sučelje je važno jer omogućava veću produktivnost uz manju primjenu kognitivnih sposobnosti te ono omogućava razvojnim inženjerima da prikažu informacije o sustavu na korisnicima prikladniji način. Iako je uvijek moguće unaprijediti korisničko sučelje, vođen navedenim smjernicama, smatram da je izvedeno sučelje prikazano na slici (**Slika 28**) dovoljno jasno i jednostavno za primjenu.

5. Zaključak

U ovom radu prikazan je tijek izrade programske podrške potrebne za upravljanje sa robotskom rukom KUKA LWR4+. Putem UDP protokola potrebno je bilo povezati korisničko računalo sa upravljačkim računalom robota, što je ostvareno putem Fast Research Interface knjižnice koja nam omogućava korištenje korisničkog sučelja robota. Povezivanje robota sa vanjskim računalima ima velike prednosti jer se otvara potpuno nova platforma na kojoj se mogu vršiti zahtjevniji algoritmi upravljanja i općenito, ponašanja robota.

Na samom početku pristupilo se izradi direktnog kinematičkog modela robota, koji je prvotno izveden u MATLAB programskom paketu. Nakon toga pristupio sam implementaciji upravljanja pojedinim zglobovima robota što je uspješno izvršeno. No najviše problema je bilo oko pokušaja upravljanja robotom u kartezijskom koordinatnom sustavu. Naime, KUKA LWR4+ ne dopušta upravljanje direktno putem kartezijskih koordinata koristeći inverzni kinematički model definiran u upravljačkom računalu robota. Zbog toga je bilo potrebno izraditi vlastiti inverzni kinematički model određen analitičkom metodom, konkretno geometrijskim pristupom. Kako je KUKA LWR4+ robot sa sedam stupnjeva slobode gibanja, proračun inverzne kinematike postaje kompleksan zbog velikog broja mogućih rješenja te je stoga u ovom radu jedan stupanj slobode gibanja "zanemaren" tako što ga se postavilo u nulu i zaključalo u toj poziciji. Dokazano je da je pozicioniranje robota koristeći direktan i inverzni matematički model kojeg sam izradio uspješno izvedeno.

Programska podrška sastoji se od tri glavna dijela koji se istovremeno izvršavaju. To je potprogram zadužen za komunikaciju sa robotom, potprogram zadužen za proračun inverznog i direktnog kinematičkog modela te potprogram za korisničko sučelje koji služi za komunikaciju sa korisnikom. Programska podrška izvedena je koristeći programski jezik C++.

U izradi ovoga rada korišteno je znanje stečeno tokom studija iz područja mehanike, računalnih mreža, robotike i automatike što je i suština smjera kojeg završavam te mi je zadovoljstvo što sam to znanje uspio objediniti u diplomskom radu.

LITERATURA

[1] Kucuk, S., Bingul, Z.: Robot Kinematics: Forward and Inverse Kinematics, *Industrial Robotics: Theory, Modelling and Control,* pIV pro literatur Verlag Robert Mayer-Scholz, Mammendorf, Germany, 2007.

[2] Meyer, H., Nagy, I., Knoll, A.: Inverse Kinematics of a Manipulator for Minimally Invasive Surgery, Technische Universitat Munchen, 2004.

[3] Siciliano B., Sciavicco L., Villani L., Oriolo G.: Robotics: Modelling, Planning and control, Springer, 2009.

[4] Lightweight Robot 4+: Operating instructions, KUKA roboter Gmbh, Augsburg, 2010

[5] Drobilo, L.: Unaprijeđivanje apsolutne točnosti robotske ruke, Sveučilište u Zagrebu, 2012

[6] Jerbić, B., Stipančić, T., Šuligoj, F.: Računalne mreže, predavanje

[7] Bischoff, R.: From research to products: The development of KUKA light-weight robot, presentation at 40th International Symposium on Robotics, Barcelona, 2009

[8] KUKA Fast Research Interface: For KUKA System Software 5.6 lr, KUKA roboter Gmbh, Augsburg, 2011

6. PRILOG

1. Funkcija *forward()*
2. Funkcija *inverz()*
3. Funkcija *kretanje()*
4. Funkcija *command()*

Prilog 1.

Funkcija *forward()*

(C++ kod)

```
float * forward(float jnt_pos[6]){
	for (i=0; i<6; i++)
		{//cout<<"jnt u funkc: "<<jnt_pos[i]<<endl;
	if(i ==2)
		fi[i]=(jnt_pos[i]+90)*PI/180;
	else
	fi[i]=jnt_pos[i]*PI/180;
	}

	for (i=0; i<6; i++)
		{//cout<<"fi u funkc: "<<fi[i]<<endl;

	}
	float alfa[6]={0, PI/2, -PI, PI/2, PI/2, -PI/2};
	float a[6]={0, 0, 400, 0, 0, 0};
	float d[6]={310, 0, 0, 390, 0, 78};

	Matrix4f T0(4,4);
	T0 << cos(fi[0]), -sin(fi[0]), 0, a[0],
	 cos(alfa[0])*sin(fi[0]), cos(alfa[0])*cos(fi[0]), -sin(alfa[0]), -
d[0]*sin(alfa[0]),
	    sin(alfa[0])*sin(fi[0]), cos(fi[0])*sin(alfa[0]), cos(alfa[0]),
d[0]*cos(alfa[0]),
	0, 0, 0, 1;

	Matrix4f T1(4,4);
	T1 << cos(fi[1]), -sin(fi[1]), 0, a[1],
	 cos(alfa[1])*sin(fi[1]), cos(alfa[1])*cos(fi[1]), -sin(alfa[1]), -
d[1]*sin(alfa[1]),
	    sin(alfa[1])*sin(fi[1]), cos(fi[1])*sin(alfa[1]), cos(alfa[1]),
d[1]*cos(alfa[1]),
	0, 0, 0, 1;

	Matrix4f T2(4,4);
	T2 << cos(fi[2]), -sin(fi[2]), 0, a[2],
	 cos(alfa[2])*sin(fi[2]), cos(alfa[2])*cos(fi[2]), -sin(alfa[2]), -
d[2]*sin(alfa[2]),
	    sin(alfa[2])*sin(fi[2]), cos(fi[2])*sin(alfa[2]), cos(alfa[2]),
d[2]*cos(alfa[2]),
	0, 0, 0, 1;
	Matrix4f T3(4,4);
	T3 << cos(fi[3]), -sin(fi[3]), 0, a[3],
	 cos(alfa[3])*sin(fi[3]), cos(alfa[3])*cos(fi[3]), -sin(alfa[3]), -
d[3]*sin(alfa[3]),
	    sin(alfa[3])*sin(fi[3]), cos(fi[3])*sin(alfa[3]), cos(alfa[3]),
d[3]*cos(alfa[3]),
	0, 0, 0, 1;
	Matrix4f T4(4,4);
    T4 << cos(fi[4]), -sin(fi[4]), 0, a[4],
	 cos(alfa[4])*sin(fi[4]), cos(alfa[4])*cos(fi[4]), -sin(alfa[4]), -
d[4]*sin(alfa[4]),
	    sin(alfa[4])*sin(fi[4]), cos(fi[4])*sin(alfa[4]), cos(alfa[4]),
d[4]*cos(alfa[4]),
	0, 0, 0, 1;
	Matrix4f T5(4,4);
	T5 << cos(fi[5]), -sin(fi[5]), 0, a[5],
	 cos(alfa[5])*sin(fi[5]), cos(alfa[5])*cos(fi[5]), -sin(alfa[5]), -
d[5]*sin(alfa[5]),
	    sin(alfa[5])*sin(fi[5]), cos(fi[5])*sin(alfa[5]), cos(alfa[5]),
d[5]*cos(alfa[5]),
	0, 0, 0, 1;
  Matrix4f T(4,4);
```

```
  T=T0*T1*T2*T3*T4*T5;
  //std::cout << T << std::endl;
  static float eulerkut[3];
        eulerkut[0]=atan2(T(1,0),T(0,0));
        eulerkut[1]=-atan2(T(2,0),sqrt(pow(T(2,1),2)+pow(T(2,2),2)));
        eulerkut[2]=atan2(T(2,1),T(2,2));
       return eulerkut;

}
```

Prilog 2.

Funkcija *inverzi()*

(C++ kod)

```
float * inverzi(float pos_r[6]){
//User frame UF na robotu
	float w=0, p=0, r=0, x0=0, y0=0, z0=0; // x0, y0, z0 su koordinate u flanði
	Matrix4f Rtx(4,4);
	Rtx << 1, 0, 0, 0,
		0, cos(w), -sin(w), 0,
		0, sin(w), cos(w), 0,
		0, 0, 0, 1;
	Matrix4f Rty(4,4);
	Rty << cos(p), 0, sin(p), 0,
		0, 1, 0, 0,
		-sin(p), 0, cos(p), 0,
		0, 0, 0, 1;
	Matrix4f Rtz(4,4);
	Rtz << cos(r), -sin(r), 0, 0,
		sin(r), cos(r), 0, 0,
		0, 0, 1, 0,
		0, 0, 0, 1;
	Matrix4f Trans0(4,4);
	Trans0 << 1, 0, 0, x0,
		0, 1, 0, y0,
		0, 0, 1, z0,
		0, 0, 0, 1;
	Matrix4f UF(4,4);
	UF=Trans0*Rtz*Rty*Rtx;


	//POZICIJA ROBOTA
	for (i=3; i<6; i++)
	pos_r[i]=pos_r[i]*PI/180;
	x0=pos_r[0]; y0=pos_r[1]; z0=pos_r[2]; w=pos_r[5]-PI; p=pos_r[4]; r=pos_r[3];
	Matrix4f Rx(4,4);
	Rx << 1, 0, 0, 0,
		0, cos(w), -sin(w), 0,
		0, sin(w), cos(w), 0,
		0, 0, 0, 1;
	Matrix4f Ry(4,4);
	Ry << cos(p), 0, sin(p), 0,
		0, 1, 0, 0,
		-sin(p), 0, cos(p), 0,
		0, 0, 0, 1;
	Matrix4f Rz(4,4);
	Rz << cos(r), -sin(r), 0, 0,
		sin(r), cos(r), 0, 0,
		0, 0, 1, 0,
		0, 0, 0, 1;
	Trans0 << 1, 0, 0, x0,
		0, 1, 0, y0,
		0, 0, 1, z0,
		0, 0, 0, 1;
	Matrix4f T0(4,4);
	T0=UF*Trans0*Rz*Ry*Rx;

	//kinematika-zelimo vratiti 6. zglob u presjecište zgloba 4 i 5
	//definiramo matricu sa kojom cemo pomaknuti zadnji zglob za 78mm po Z osi
	Matrix4f tool(4,4);
	tool << 1, 0, 0, 0,
		0, 1, 0, 0,
		0, 0, 1, 78,
		0, 0, 0, 1;
	Matrix4f T(4,4);
	T=T0*tool; //pomaknuli smo zeljenu poziciju za 78mm po Z osi
```

```
	//cout << "Here is the matrix T:\n" << T << endl;

	//%racunamo inverz za j1 j2 i j3
	float Nx=T(0,3); //%U Nx spremamo X koordinatu zeljene tocke
	float Ny=T(1,3);
	float Nz=T(2,3);
	if (Nx<0.0001 && Nx>-0.0001)
		Nx=0;
	if (Ny<=0.0001 && Ny>=-0.0001)
		Ny=0;
	if (Nz<=0.0001 && Nz>=-0.0001)
		Nz=0;
	float J1=atan2(Ny,Nx)*180/PI;//J1

	float pp=sqrt(pow(Nx,2)+pow(Ny,2));
	float e=atan2(Nz-310,pp);
	int aa=390;
	int bb=400;
	float cc=sqrt(pow((Nz-310),2)+pow(pp,2));
	float s=(aa+bb+cc)/2;
	float k=s-cc;
	if (k<=0.0000001 && k>=-0.0000001)
		k=0;
	float rr=sqrt((s-aa)*(s-bb)*(k)/s);
	float alfaa=2*atan2(rr,(s-aa));
	float gama=2*atan2(rr,(s-cc));
	float J2=(alfaa+e)*(180)/PI;
	float J3=180-(gama*180/PI);

	//forward za prva 3 zgloba
	float J333=J3+90;
	float fi1=J1*PI/180, fi2=J2*PI/180, fi3=J333*PI/180;
	float alfa[6]={0, PI/2, -PI, PI/2, PI/2, -PI/2};
	float a[6]={0, 0, 400, 0, 0, 0};
	float d[6]={310, 0, 0, 390, 0, 78};

	Matrix4f T01(4,4);
	T01 << cos(fi1), -sin(fi1), 0, a[0],
	cos(alfa[0])*sin(fi1), cos(alfa[0])*cos(fi1), -sin(alfa[0]), -
d[0]*sin(alfa[0]),
	   sin(alfa[0])*sin(fi1), cos(fi1)*sin(alfa[0]), cos(alfa[0]),
d[0]*cos(alfa[0]),
	0, 0, 0, 1;

	Matrix4f T12(4,4);
	T12 << cos(fi2), -sin(fi2), 0, a[1],
	cos(alfa[1])*sin(fi2), cos(alfa[1])*cos(fi2), -sin(alfa[1]), -
d[1]*sin(alfa[1]),
	   sin(alfa[1])*sin(fi2), cos(fi2)*sin(alfa[1]), cos(alfa[1]),
d[1]*cos(alfa[1]),
	0, 0, 0, 1;

	Matrix4f T23(4,4);
	T23 << cos(fi3), -sin(fi3), 0, a[2],
	cos(alfa[2])*sin(fi3), cos(alfa[2])*cos(fi3), -sin(alfa[2]), -
d[2]*sin(alfa[2]),
	   sin(alfa[2])*sin(fi3), cos(fi3)*sin(alfa[2]), cos(alfa[2]),
d[2]*cos(alfa[2]),
	0, 0, 0, 1;

	Matrix4f polozaj(4,4);
	polozaj=T01*T12*T23;
```

```
	Matrix4f T2(4,4);
	T2=polozaj.inverse()*T0;

	float J4=atan2(T2(2,2),T2(0,2))*180/PI;
	float J5=atan2(sqrt(pow(T2(1,1),2)+pow(T2(1,0),2)),T2(1,2))*180/PI;
	float J6=-atan2(T2(1,1),-T2(1,0))*180/PI;
	//izvedene su 4 konfiguracije robota
	float inverse1 [6]={J1, J2, J3, J4, J5, J6};//PRVI POLOZAJ
	float J11, J22, J33, J44, J55, J66; //pomocne varijable
	//%%poz 2 elbow up, flip J4,J5,J6
	if (J4>=0)
		J44=J4-180;
	if (J4<0)
		J44=J4+180;
	J55=-1*J5;
	if (J6>=0)
		J66=J6-180;
	if (J6<0)
		J66=J6+180;
	float inverse2 [6]={J1, J2, J3, J44, J55, J66};//DRUGI POLOZAJ
	//%%poz3 J1 za 180, elbow up
	if (J1>=0)
		J11=J1-180;
	if (J1<0)
		J11=J1+180;
	J22=180-J2;
	J33=-1*J3;
	float inverse3 [6]={J11, J22, J33, J44, J5, J6};//TRECI POLOZAJ
	//%%poz4 J1 za 180, elbow up, flipan
	J55=-1*J5;
	float inverse4 [6]={J11, J22, J33, J4, J55, J66};//CETVRTI POLOZAJ

	//cout<<"\n\n Konfiguracije "<<endl<<endl;
	//for(i=0; i<6; i++)
	//cout
<<left<<setw(10)<<inverse1[i]<<"\t"<<left<<setw(10)<<inverse2[i]<<"\t"<<left<<setw(10)
<<inverse3[i]<<"\t"<<left<<setw(10)<<inverse4[i] << endl;

	///Usporedba rjesenja. Prvo odredimo pomocnu var pom1 koja je razlika trenutnog
polozaja i dobivenih polozaja inverznnom kinematikom
	//zatim napravimo absolutnu vrijednost vektora razlike te zbrojimo sve clanove
tog vektora. na kraju di će zbroj biti najmanji taj cemo vektor odabrati
	float pom1[6], po1=0;
	float pom2[6], po2=0;
	float pom3[6], po3=0;
	float pom4[6], po4=0;
	float ogranicenja[6]={170, 120, 120, 170, 130, 170};
	for (i=0; i<6; i++){
		if(abs(inverse1[i])>ogranicenja[i])
			inverse1[i]=10000;
		pom1[i]=trenutno[i]-inverse1[i];
		pom1[i]=abs(pom1[i]);
		po1=po1+pom1[i];

		if(abs(inverse2[i])>ogranicenja[i])
			inverse2[i]=10000;
		pom2[i]=trenutno[i]-inverse2[i];
		pom2[i]=abs(pom2[i]);
		po2=po2+pom2[i];

		if(abs(inverse3[i])>ogranicenja[i])
			inverse3[i]=10000;
```

```
                pom3[i]=trenutno[i]-inverse3[i];
                pom3[i]=abs(pom3[i]);
                po3=po3+pom3[i];

                if(abs(inverse4[i])>ogranicenja[i])
                        inverse4[i]=10000;
                pom4[i]=trenutno[i]-inverse4[i];
                pom4[i]=abs(pom4[i]);
                po4=po4+pom4[i];
         }

        //usporedba gdje je najmanji zbroj zakret osi
        float min=po1;
        if (po2<min)
                min=po2;
        if (po3<min)
                min=po3;
        if(po4<min)
                min=po4;
        //odluka koju cemo konfiguraciju uzeti
        static float konfig[6];
        for(i=0; i<6; i++){

        if(min==po1)
                konfig[i]=inverse1[i];
        else if(min==po2)
                konfig[i]=inverse2[i];
        else if(min==po3)
                konfig[i]=inverse3[i];
        else
                konfig[i]=inverse4[i];
        }

        return konfig;

}
```

Prilog 1.

Funkcija *kretanje()*

(C++ kod)

```
void kretanje()
	{
		float cmdJntPos[LBR_MNJ];
		for ( int i= 0; i < LBR_MNJ; i++)
			{
			cmdJntPos[i]= friInst.getMsrCmdJntPosition()[i] +
			friInst.getMsrCmdJntPositionOffset()[i];
			}

		if(input=='p')
			cmdJntPos[0]=mojjoint[0];
			cmdJntPos[1]=mojjoint[1];
			cmdJntPos[3]=mojjoint[3];
			cmdJntPos[4]=mojjoint[4];
			cmdJntPos[5]=mojjoint[5];
			cmdJntPos[6]=mojjoint[6];
		if(input=='1' || input=='2')
			cmdJntPos[0]=mojjoint[0];
		if(input=='3' || input=='4')
			cmdJntPos[1]=mojjoint[1];
		if(input=='e' || input=='r')
			cmdJntPos[2]=mojjoint[2];
		if(input=='5' || input=='6')
			cmdJntPos[3]=mojjoint[3];
		if(input=='7' || input=='8')
			cmdJntPos[4]=mojjoint[4];
		if(input=='9' || input=='0')
			cmdJntPos[5]=mojjoint[5];
		if(input=='q' || input=='w')
			cmdJntPos[6]=mojjoint[6];

		friInst.doPositionControl(cmdJntPos,false);

	}
```

Prilog 1.

Funkcija *kretanje()*

(C++ kod)

```
while (pomocni2[0]!=mojjoint[0] || pomocni2[1]!=mojjoint[1]+PI/2 ||
pomocni2[3]!=mojjoint[3]+PI/2 ||
              pomocni2[4]!=mojjoint[4] || pomocni2[5]!=mojjoint[5]+PI/2 ||
pomocni2[6]!=mojjoint[6]+PI/2 )
       {
              if( pomocni2[0]-mojjoint[0]<0.0005 && pomocni2[0]-mojjoint[0]>-0.0005 &&
pomocni2[1]-(mojjoint[1]+PI/2)<0.0005 && pomocni2[1]-(mojjoint[1]+PI/2)>-0.0005 &&
              pomocni2[3]-mojjoint[3]<0.0005 && pomocni2[3]-mojjoint[3]>-0.0005&&
pomocni2[4]-mojjoint[4]<0.0005 && pomocni2[4]-mojjoint[4]>-0.0005 &&
               pomocni2[5]-mojjoint[5]<0.0005 && pomocni2[5]-mojjoint[5]>-0.0005 &&
pomocni2[6]-mojjoint[6]<0.0005 && pomocni2[6]-mojjoint[6]>-0.0005)
              {break;
       Sleep(1);
              }
       if(pomocni2[0]>mojjoint[0]){
       if(abs(mojjoint[0]-pomocni2[0])<brzina2)
              mojjoint[0]=mojjoint[0]+abs(mojjoint[0]-pomocni2[0]);
       else
              mojjoint[0]=mojjoint[0]+brzina2;
       }
       if(pomocni2[0]<mojjoint[0]){
              if(abs(mojjoint[0]-pomocni2[0])<brzina2)
                     mojjoint[0]=mojjoint[0]-abs(mojjoint[0]-pomocni2[0]);
              else
              mojjoint[0]=mojjoint[0]-brzina2;
       }
       if(pomocni2[1]>mojjoint[1]+PI/2){
              if(abs(mojjoint[1]-pomocni2[1])<brzina2)
                     mojjoint[1]=mojjoint[1]+abs(mojjoint[1]-pomocni2[1]);
              else
              mojjoint[1]=mojjoint[1]+brzina2;
       }

       if(pomocni2[1]<mojjoint[1]+PI/2){
              if(abs(mojjoint[1]-pomocni2[1])<brzina2)
                     mojjoint[1]=mojjoint[1]-abs(mojjoint[1]-pomocni2[1]);
              else
                     mojjoint[1]=mojjoint[1]-brzina2;
       }
       if(pomocni2[3]>mojjoint[3]){
              if(abs(mojjoint[3]-pomocni2[3])<2*brzina2)
                     mojjoint[3]=mojjoint[3]+abs(mojjoint[3]-pomocni2[3]);
              else
              mojjoint[3]=mojjoint[3]+2*brzina2;
       }
       if(pomocni2[3]<mojjoint[3]){
              if(abs(mojjoint[3]-pomocni2[3])<2*brzina2)
                     mojjoint[3]=mojjoint[3]-abs(mojjoint[3]-pomocni2[3]);
              else
              mojjoint[3]=mojjoint[3]-2*brzina2;
       }
       if(pomocni2[4]>mojjoint[4]){
              if(abs(mojjoint[4]-pomocni2[4])<brzina2)
                     mojjoint[4]=mojjoint[4]+abs(mojjoint[4]-pomocni2[4]);
              else
              mojjoint[4]=mojjoint[4]+brzina2;
       }
       if(pomocni2[4]<mojjoint[4]){
              if(abs(mojjoint[4]-pomocni2[4])<brzina2)
                     mojjoint[4]=mojjoint[4]-abs(mojjoint[4]-pomocni2[4]);
```

```
		else
		mojjoint[4]=mojjoint[4]-brzina2;
	}
	if(pomocni2[5]>mojjoint[5]){
		if(abs(mojjoint[5]-pomocni2[5])<2*brzina2)
			mojjoint[5]=mojjoint[5]+abs(mojjoint[5]-pomocni2[5]);
		else
		mojjoint[5]=mojjoint[5]+2*brzina2;
	}
	if(pomocni2[5]<mojjoint[5]){
		if(abs(mojjoint[5]-pomocni2[5])<2*brzina2)
			mojjoint[5]=mojjoint[5]-abs(mojjoint[5]-pomocni2[5]);
		else
		mojjoint[5]=mojjoint[5]-2*brzina2;
	}
	if(pomocni2[6]>mojjoint[6]){
		if(abs(mojjoint[6]-pomocni2[6])<2*brzina2)
			mojjoint[6]=mojjoint[6]+abs(mojjoint[6]-pomocni2[6]);
		else
		mojjoint[6]=mojjoint[6]+2*brzina2;
	}
	if(pomocni2[6]<mojjoint[6]){
		if(abs(mojjoint[6]-pomocni2[6])<2*brzina2)
			mojjoint[6]=mojjoint[6]-abs(mojjoint[6]-pomocni2[6]);
		else
		mojjoint[6]=mojjoint[6]-2*brzina2;
	}

	Sleep(1);
```