Edukacijska robotska ruka

Teči, Tomislav

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Tomislav Teči

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Mladen Crneković, dipl. ing.

Student:

Tomislav Teči

Zagreb, 2024.

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

Zahvaljujem se prvenstveno svome mentoru prof.dr.sc Mladenu Crnekoviću koji je prihvatio moju temu za izradu rada. Navodio me od samoga početka u izradi sa svojim prijedlozima i komentarima, te je svaki moj prijedlog razmotrio i kritički mu pristupio. Hvala asistentu mag.ing. Branimiru Ćaranu koji mi je omogućio svu potrebnu elektroničku, upravljačku te pogonsku opremu za izradu ovoga rada.

Zahvaljujem se također svojoj obitelji što su mi omogućili studiranje kao i njihovoj podršci, te podršci djevojke kroz cijelo vrijeme studiranja.

Tomislav Teči, v.r.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 24 – 06 / 1**	
Ur.broj: **15 – 24 –**	

ZAVRŠNI ZADATAK

Student: **Tomislav Teči** JMBAG: **0035239687**

Naslov rada na hrvatskom jeziku: **Edukacijska robotska ruka**

Naslov rada na engleskom jeziku: **Educational robotic arm**

Opis zadatka:

Za robotičara je uvijek veliki izazov izraditi svoju robotsku ruku. U vrijeme 3D printera, jeftine elektronike i ostalih elemenata od kojih je sastavljena robotska ruka, moguće je za prihvatljiv novčani iznos izraditi vlastitog robota.

Potrebno je istražiti tržište gotovih edukacijskih robota i napraviti njihovu usporedbu, a zatim predložiti svoje rješenje uz pretpostavku da korisnik ima 3D printer i može nabaviti hobističke dijelove koje sam ne izrađuje (motori, senzori, upravljački uređaj itd.).

U radu je potrebno:

- Definirati mehaničku konstrukciju robota s 4 stupnja slobode gibanja i mogućnošću hvatanja predmeta.
- Odabrati Dinamixel motore za pokretanje rotacijskih osi robota i hvataljke.
- Odabrati potrebne senzore i upravljački mikrokontroler.
- Definirati sučelje robota prema korisniku.
- Procijeniti ukupnu cijenu izrade robota.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
24. 4. 2024.	**2. rok (izvanredni):** 11. 7. 2024. **3. rok:** 19. i 20. 9. 2024.	**2. rok (izvanredni):** 15. 7. 2024. **3. rok:** 23. 9. – 27. 9. 2024.

Zadatak zadao: Predsjednik Povjerenstva:

prof. dr. sc. Mladen Crneković

izv. prof. dr. sc. Petar Ćurković

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
q		Vektor unutarnjih koordinata
q1	rad	Kut zakreta motora 1
q2	rad	Kut zakreta motora 2
q3	rad	Kut zakreta motora 3
q4	rad	Kut zakreta motora 4
q5	rad ili m	Općenita varijabla vektora unutarnjih koordinata
q6	rad ili m	Općenita varijabla vektora unutarnjih koordinata
R		Vektor vanjskih koordinata
x	m	X koordinata izvršnog člana
y	m	Y koordinata izvršnog člana
z	m	Z koordinata izvršnog člana
θ	rad	Eulerov kut nagiba izvršnog člana
φ	rad	Eulerov kut skretanja izvršnog člana
ψ	rad	Eulerov kut valjanja izvršnog člana
A_x		Matrica rotacije oko x-osi
A_y		Matrica rotacije oko y-osi
A_z		Matrica rotacije oko z-osi
A_t		Matrica translacije
α	rad	Osnovni kut za prikaz matrica
a_i	m	Udaljenost u smjeru X_i od Z_{i-1} do Z_i
α_i	rad	Kut od Z_{i-1} do Z_i gledajući X_i
d_i	m	Udaljenost uzduž Z_{i-1} do X_{i-1} do X_i
θ_i	rad	Kut zakreta motora gledajući pravilo desne ruke
L_1	m	Udaljenost od ishodišta nultog koordinatnog sustava do motora 2
L_2	m	Duljina drugoga članka robotske ruke
L_3	m	Duljina trećeg članka robotske ruke
L_4	m	Duljina četvrtog članka robotske ruke (hvataljke)
T		Osnovna matrica transformacije prema Denavit-Hartenbergovom standardu
f	m	Oznaka kraka trokuta za izračun inverzne kinematike
r	m	Duljina projekcije robotske ruke do trećega članka na x-y ravninu
s	m	Visina projekcije robotske ruke od trećeg članka do prvog

Oznaka	**Jedinica**	**Opis**
A	m	Pomoćna varijabla u izračunu inverzne kinematike
B	m	Pomoćna varijabla u izračunu inverzne kinematike
D	m	Varijabla za supstituciju
a	m	Pomoćna varijabla u izračunu inverzne kinematike
b	m	Pomoćna varijabla u izračunu inverzne kinematike
J		Jacobian-ova matrica parcijalnih derivacija
K_p	-	Pojačanje P člana regulatora pozicije
K_i	-	Pojačanje I člana regulatora pozicije
K_d	-	Pojačanje D člana regulatora pozicije

SAŽETAK

Ovaj završni rad se bavi sa edukacijskim robotskim rukama, odnosno robotskim rukama koje su predviđene za amatersku upotrebu i edukaciju. Ovakve robotske ruke su idealne za uvođenje studenata ili učenika srednjih strukovnih škola u svijet robotike u svojem osnovnom obliku. U nastavku će se edukacijska robotska ruka smatrati robotskom rukom. Rad započinje s osnovnom motivacijom za izradu i cilj. Nakon uvoda u radu je istraženo tržište već dostupnih robotskih ruku, koje su zatim i uspoređene po svojim specifikacijama, stupnjevima slobode gibanja, cijeni itd. Nadalje, predloženo je vlastito rješenje robotske ruke s četiri stupnja slobode gibanja i mogućnosti hvatanja predmeta, koje je podijeljeno u nekoliko glavnih kategorija. Prvo je opisana osnovna zamisao i struktura robota koja će biti napravljena. Pomoću zadane strukture su odabrani potrebni servo motori, koji će zadovoljiti svojom preciznosti i snagom (momentom). Nakon odabira motora, odabrano je i potrebno napajanje koje će zadovoljiti potrebnu dobavu energije sustavu. Zatim detaljno je objašnjeno povezivanje i postavljanje parametara motora, koji su kasnije neophodni za ispravan rad robotske ruke. Nadalje, rad prolazi kroz kinematsku strukturu robota koja je opisana pomoću Denavit-Hartenberg-ove metode za određivanje direktne i inverzne kinematike. Nakon odrađene kinematike slijedi opis izrade konstrukcije robotske ruke i hvataljke. Za kraj je pokazano i objašnjeno korisničko sučelje (GUI) kao i programski dio robota odrađen pomoću MATLAB-a i App Designer-a.

Ključne riječi: robotska ruka, servo motori, edukacija, App Designer, inverzna kinematika, direktna kinematika

SUMMARY

This thesis deals with educational robotic arms, specifically robotic arms designed for amateur use and education. These robotic arms are ideal for introducing students or high school students to the world of robotics in its basic form. From this point forward, the educational robotic arm will be referred to as the robotic arm. The work begins with the basic motivation for its creation and purpose. Following the introduction, the thesis explores the market for existing robotic arms, comparing them based on their specifications, degrees of freedom, cost, and other factors. Furthermore, a custom solution for a robotic arm with 4 degrees of freedom and gripping capability is proposed, divided into several main categories. First, the basic concept and structure of the robot to be created are described. Based on the defined structure, the necessary servo motors are selected to meet the required precision and power (torque). After selecting the motors, the necessary power supply to meet the energy needs of the system is chosen. Then, the connection and parameter settings of the motors, which are essential for the proper functioning of the robotic arm, are explained in detail. Additionally, the thesis goes into the kinematic structure of the robot, described using the Denavit-Hartenberg method for determining direct and inverse kinematics. Following the kinematics, the construction of the robotic arm and gripper is described. Finally, the user interface (GUI) and the software part of the robot, developed using MATLAB App Designer, are demonstrated and explained.

Keywords: robotic arm, servo motors, education, App Designer, inverse kinematics, direct kinematics

1. UVOD

Prva robotska ruka je ugledala svjetlo dana davne 1961. godine u saveznoj državi New Jersey. Nosila je ime Unimate, te je imala 6 stupnjeva slobode gibanja koji su imitirali rame, lakat i šaku ljudskog bića. Tada je to bila novost i veliki znak napretka u industriji, jer je mogla zamijeniti ljude na radnim mjestima na kojima do tada nisu mogli biti zamijenjeni strojem [1]. U današnje vrijeme pojam robotske ruke nije ništa novo te svatko tko je tehničke struke i čuje ovaj pojam zna o čemu se radi. Povod ovoj temi završnog rada je upravo to što su robotske ruke jako rasprostranjene i koriste se za većinu stvari s kojima imamo doticaja. Potrebno je odvojiti i znati što je to industrijska robotska ruka, a što edukacijska. Industrijska robotska ruka se koristi, kao što i samo ime govori u industriji. Glavna obilježja su joj velika pouzdanost, visoka preciznost, sigurnost u radu, prilagođena je teškim radnim uvjetima i visoke je cijene. S druge strane, edukacijska robotska ruka, koja bi se mogla nazvati i amaterskom, ima sva ista obilježja kao i industrijska samo su joj obilježja na razini ili dvije ispod one industrijske. Glava svrha edukacijskih ruka je da si svatko može doma priuštiti jednu i koristiti je za vlastite svrhe, a time i naučiti sve glavne značajke i principe koji se koriste kod robotske ruke, što uvelike olakšava shvaćanje i razumijevanje tematike za studente i učenike srednjih strukovnih škola [2]. Današnje tržište već nudi nekoliko takvih ruku, te će se napraviti njihova usporedba u sljedećem poglavlju. Izrada robotske ruke od početka do kraja zahtijeva apsolutno korištenje svih znanja koja čine mehatroniku. Od konstrukcije koja je prvenstveno strojarski dio, pa motora i njihovog napajanja i ožičenja koja čine elektronički dio, pa sve do software-a kojeg je potrebno isprogramirati kako bi sve to skupa funkcioniralo. Svaki od ovih postupaka će biti detaljno analiziran i objašnjen, kao i nastali problemi tijekom izrade, kako bi omogućili čitatelju potpuni pogled na cijeli postupak izrade vlastite robotske ruke.

2. USPOREDBA GOTOVIH EDUKACIJSKIH ROBOTA

Tržište je uistinu puno gotovih edukacijskih ruku, od onih koje su amateri napravili pa prodaju po stranicama poput Amazona ili eBay-a, koje koriste jeftine komponente što se nađu na istim tim stranicama, do privatnih tvrtki koje rade malo ozbiljnije robotske ruke. Odabir je pao na pretragu tržišta middle-rage robotskih ruku gdje se cijene kreću do par tisuća eura. Nakon detaljne analize internetskih stranica i webshop-ova nekoliko robotskih ruku se isticalo kao najpoznatije, te su one uzete u daljnju analizu performansi, sposobnosti i cijene.

U tu skupinu upadaju: Dobot magician basic robot arm, Interbotix PincherX_100, Wlkata Mirobot, Niryo Ned2 i Ozobot Robotic Arm.

2.1. Dobot magician basic robot arm

Slika 2.1 Dobot magican basic [3]

Izgrađena da se koristi u edukacijske svrhe na sveučilištima, srednjim strukovnim školama ili na osposobljavanju radnika u određenoj industriji. Sama robotska ruka dolazi s nekoliko nastavaka koji omogućavaju crtanje, pick & place operacije te 3D printanje.

Tablica 2.1 Karakteristike Dobot magician basic-a

Stupnjevi slobode gibanja (SSG)	**4**
Max. opterećenje	**500 g**
Max. doseg	**320 mm**
Težina	**3,4 kg**
Materijal	**Aluminijska legura 6061, ABS**
Ponovljivost	**0,02mm**
Cijena	**1 300 €**

2.2. Interbotix PincherX_100

Slika 2.2 Interbotix PincherX_100 robotska ruka [4]

Dizajniran za edukaciju i istraživanje, ova robotska ruka podržava ROS, Gazebo, MoveIt i MATLAB. XSeries ruke rade s istim središnjim repozitorijem otvorenog koda, što olakšava prijenos koncepata s jedne platforme na drugu. PincherX_100 je idealan za korištenje u učionici i posebno je pogodan za primjene, kao što su preuzimanje i postavljanje temeljeno na viziji, strojno učenje i umjetna inteligencija. Također moguća je izmjena hvataljke s nekom drugom vrstom koja korisniku više odgovara.

Tablica 2.2 Karakteristike Interbotix PincherX_100-a

Stupnjevi slobode gibanja	**4**
Max. opterećenje	**50 g**
Max. doseg	**300 mm**
Težina	**1,9 kg**
Materijal	**P95 akrilna i ABS plastika**
Ponovljivost	**5mm**
Cijena	**890 €**

2.3. Wlkata Mirobot

Slika 2.3 Wlkata Mirobot robotska ruka [5]

Premošćujući jaz između učenja industrijske i kolaborativne robotike, WLKATA-in stolni robot nudi kombinaciju izvanrednih performansi, elegantnog dizajna i neusporedive obrazovne vrijednosti za buduće učenike robotike i profesionalce. Dolazi u više boja, a glavni cilj ove robotske ruke je učenje industrije 4.0.

Tablica 2.3 Karakteristike Wlkata Mirobot-a

Stupnjevi slobode gibanja	**6**
Max. opterećenje	**400 g**
Max. doseg	**315 mm**
Težina	**2 kg**
Materijal	**aluminij i ABS plastika**
Ponovljivost	**0,2 mm**
Cijena	**2 132 €**

2.4. Niryo Ned2

Slika 2.4 Niryo Ned2 robotska ruka [6]

Ova robotska ruka pruža pravo kolaborativno iskustvo između robota i čovjeka, to ostvaruje pomoću svjetlosnih i zvučnih signala pomoću kojih ostvaruje interakciju s korisnikom. Razvijena je u Francuskoj, s ciljem unaprijeđena edukacije radnika s profesionalnim kolaborativnim robotima kojima ova robotska ruka itekako može u određenoj mjeri konkurirati.

Tablica 2.4 Karakteristike Niryo Ned2

Stupnjevi slobode gibanja	**6**
Max. opterećenje	**300 g**
Max. doseg	**440 mm**
Težina	**7 kg**
Materijal	**aluminij i ABS plastika**
Ponovljivost	**0,5 mm**
Cijena	**3 990 €**

2.5. ORA: Ozobot robotic arm

Slika 2.5 ORA: Ozobot robotska ruka [7]

Glavna prednosti ove robotske ruke je njena posvećenost edukaciji. Plug & Play je glavna ideja ove ruke, koju je moguće programirati na vrlo jednostavan način. Ima karakteristike svake profesionalne industrijske robotske ruke, te je zbog toga odlično rješenje za studente koje treba pripremiti za budući rad u struci.

Tablica 2.5 Karakteristike ORA: Ozobot-a

Stupnjevi slobode gibanja	**6**
Max. opterećenje	**600 g**
Max. doseg	**440 mm**
Težina	**7,2 kg**
Materijal	**aluminij i ABS plastika**
Ponovljivost	**0,5 mm**
Cijena	**4 987 €**

2.6. Usporedba karakteristika

Usporedba je provedena na način da će se svaka karakteristika od prethodno spomenutih robotski ruku ocijeniti s ocjenom od 1-5, a zatim će se sve skupa zbrojiti, kako bi dobili konačan rezultat koja je ruka najbolja po ovom kriteriju. Svakako se treba napomenuti da je ovo usporedba striktno po tehničkim karakteristikama i cijeni robota što ne mora uvijek biti uvjet odabira kod kupca. Svatko ima svoje potrebe i preferencije te iskustvo, određeni roboti su više user-friendly, dok su neki napravljeni za iskusnije robotičare. Također treba napomenuti kako u cijenu pojedinih robota spada i pripadajuća aplikacija za programiranje ili dodatni nastavci za hvataljku što naravno povećava cijenu. U svakom slučaju za detaljniju usporedbu bi bilo potrebno isprobati robote ili napraviti određenu anketu među korisnicima spominjanih objekata.

Stupac uz karakteristike označuje ocjenu te karakteristike, te je SSG stupanj slobode gibanja.

Nakon analize se dokazalo kako je Wlkata Mirobot najbolja robotska ruka što se tiče omjera težine, kvalitete i cijene.

Tablica 2.6 Usporedba karakteristika

Ime robota	Dobot magician basic		Interbotix PincherX_100		Wlkata Mirobot		Niryo Ned2		ORA: Ozobot	
SSG	4	**3**	4	**3**	6	**5**	6	**5**	6	**5**
Max. opterećenje	500 g	**5**	50 g	**1**	400 g	**4**	300 g	**3**	600 g	**5**
Max. doseg	320 mm	**3**	300 mm	**3**	315 mm	**3**	440 mm	**5**	440 mm	**5**
Težina	3,4 kg	**4**	1,9 kg	**5**	2 kg	**5**	7 kg	**2**	7,2 kg	**2**
Materijal	Al. Leg.6061,ABS	**5**	P95 akrilna i ABS plastika	**3**	Al. i ABS plastika	**5**	Al. i ABS plastika	**5**	Al. I ABS plastika	**5**
Ponovljivost	0,02 mm	**5**	5 mm	**1**	0,2 mm	**5**	0,5 mm	**4**	0,5 mm	**4**
Cijena	1 300 €	**4**	890 €	**5**	2 132 €	**3**	3 990 €	**2**	4 987 €	**2**
Ukupna ocjena	**4.14**		**3.00**		**4.29**		**3.71**		**4.00**	

3. IZRADA VLASTITE ROBOTSKE RUKE

Izrada se sastoji od nekoliko glavnih cjelina :

- Osnovna ideja robotske ruke
- Odabir motora i popratne elektronike
- Povezivanje upravljačkog sustava, i parametri motora
- Kinematika robotske ruke
- Izrada konstrukcije
- Izrada hvataljke
- Izrada korisničkog sučelja
- Program u MATLAB-u

3.1. Osnovna ideja robotske ruke

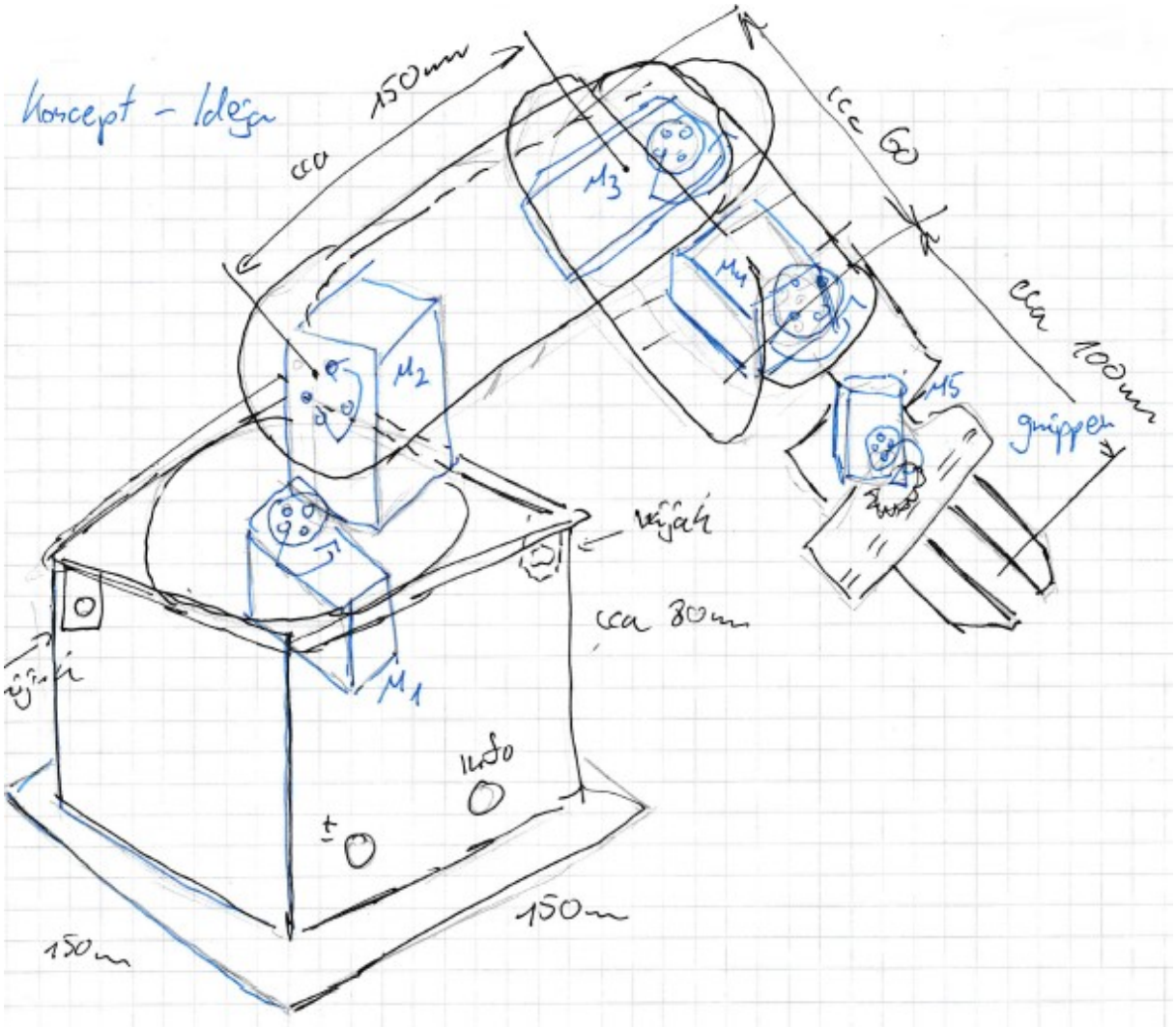
Temeljem pregleda tržišta dostupnih robotskih ruku, odluka je pala da će robot imati 4 stupnja slobode gibanja, pošto će to zahtijevati manja ulaganja u komponente i konstrukcijski će biti jednostavnija izvedba. Količina stupnjeva slobode gibanja (SSG u daljnjem tekstu) određuje fleksibilnost i mogućnosti robotske ruke. Također svaki dodatni SSG doprinosi znatnom kompliciranju matematičkog oblika kinematskog zapisa strukture robota, što bi zahtijevalo velike količine vremena za rješavanje direktnog, a posebice inverznog kinematičkog problema.

3.1.1. Stupnjevi slobode gibanja

Broj stupnjeva slobode gibanja je određen s brojem aktuatora koje robotska ruka posjeduje. Pregledom tržišta je zaključeno kako je neki minimalni broj SSG-a 3, jer nam to omogućuje pozicioniranje izvršnog člana, u slučaju ovog rada hvataljke u točku prostora. Kada bi robotska ruka imala samo 2 SSG-a, pozicioniranje bi bilo moguće ostvariti samo u određenoj ravnini. Svaki dodatni SSG do šestog nam omogućuje dodatno upravljanje orijentacijom izvršnog člana oko svih triju koordinatnih osi. Kada bi se išlo na dodatne SSG-ove oni ne bi više doprinosili poziciji i orijentaciji već načinu, putanji, i volumenu radnog prostora robota [8]. Radni prostor robota je ograničen konstrukcijom robota, te mogućnostima aktuatora. Hvataljka ne spada u SSG, jer njezin aktuator ne utječe na poziciju niti orijentaciju izvršnog člana. Tako da za izradu robota s 4 SSG-a i mogućnosti hvatanja predmeta je potrebno 5 motora. Robotske ruke s više od 6 SSG-a se koriste u zahtjevnoj industriji, te se u edukacijskim robotima takve strukture ne pronalaze.

3.1.2. Idejni koncept robotske ruke

Na početku izrade robotske ruke je bilo važno idejnu misao preslikati na komad papira kako bi se dobio osjećaj izgleda i funkcionalnosti. Važno je napomenuti da se kroz izradu ovaj početni koncept u nekoliko navrata mijenjao, jer nije moguće na početku odmah uzeti sve moguće probleme koji mogu nastati u izradi u obzir. Također je vrlo bitno napraviti strukturu takvu da kada se dođe do matematičkog opisa kinematike robota ona ne bude prekomplicirana. Idealno bi bilo kada bi se translacije vršile samo po jednoj osi. Slika 3.1 prikazuje dimenzije koje su odabrane proizvoljno kako bi doseg robotske ruke odgovarao prosječnom dosegu komercijalno dostupnih robota. S time na umu dimenzije su se kroz izradu promijenile zbog konstrukcije i maksimalnih momenata dostupnih motora koje je bilo moguće nabaviti.



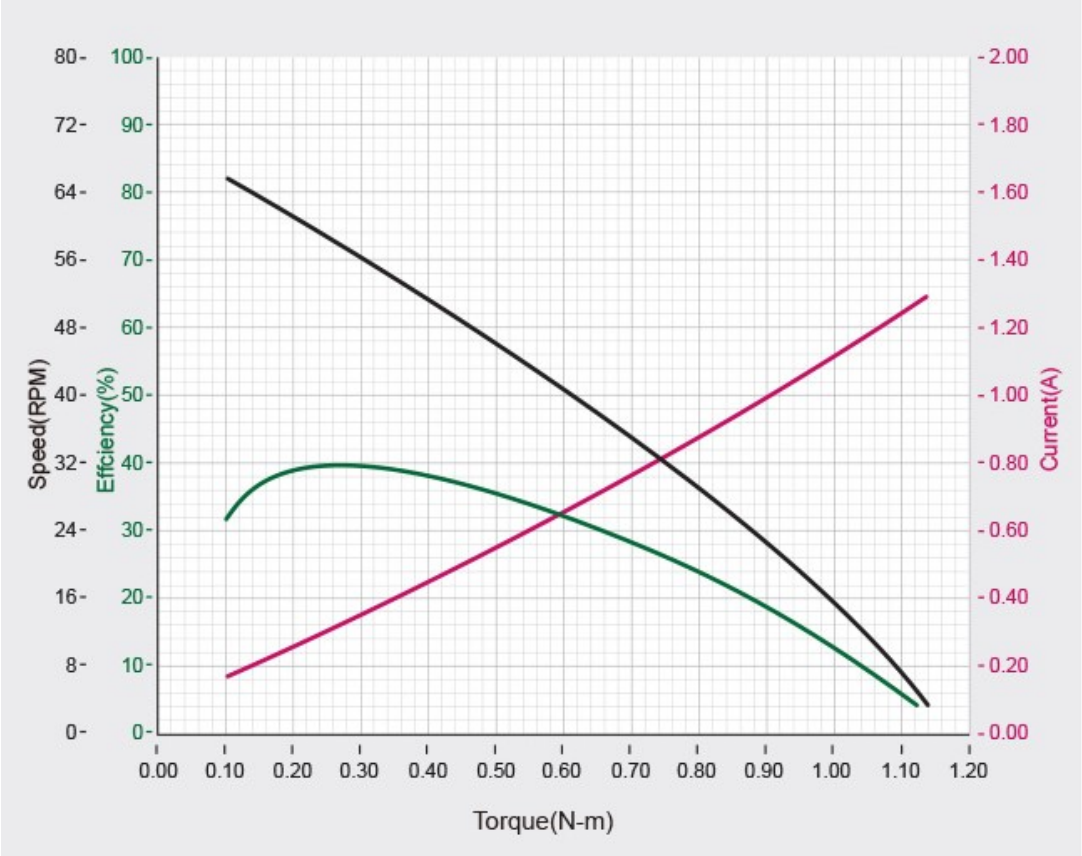
Slika 3.1 Skica koncepta robotske ruke s hvataljkom

3.2. Odabir motora i popratne elektronike

Kod odabira motora za robotsku ruku na samome početku je bilo jasno kako jeftini servo motori koji se viđaju na Amazonu i eBay-u neće zadovoljiti u izradi robotske ruke. Iako dobri za upravljanje RC autima, avionima ili brodovima njihova slaba preciznosti i nedostatak povratnih informacija su dovoljan razlog, da iako momentom zadovoljavaju, ispadaju iz odabira. Odluka je pala na Dynamixel XC430-W240/150-T servo motore, koje je bilo moguće posuditi za izradu robotske ruke. Ova vrsta servo motora se koristi u jeftinijim komercijalnim robotskim rukama te time opravdava korištenje istih za izradu vlastite ruke. Cijela ideja oko ovih motora je da se upravo koriste u svrhe robotike i mehatronike.

3.2.1. Tehničke karakteristike Dynamixel XC430-W240-T

Karakteristike motora su jedna od najvažnijih i prvih stvari koje treba proučiti pri odabiru istog. Najvažnija karakteristika je momentna karakteristika motora koja prikazuje odnos brzine vrtnje s izlaznim momentom kojeg motor daje, što prikazuje slika 3.2. Slika 3.3 prikazuje ostale važne karakteristike.



Slika 3.2 Graf performansi Dynamixel XC430-W240-T motora [9]

MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5601
Motor	Coreless
Baud Rate	9,600 [bps] ~ 4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Operating Modes	Velocity Control Mode Position Control Mode (0 ~ 360 [°]) Extended Position Control Mode (Multi-turn) PWM Control Mode (Voltage Control Mode)
Weight	65 [g]
Dimensions (W x H x D)	28.5 x 46.5 x 34 [mm]
Gear Ratio	245.22 : 1
Stall Torque	1.4 [N.m] (at 9.0 [V], 1.1 [A]) 1.7 [N.m] (at 11.1 [V], 1.3 [A]) **1.9 [N.m] (at 12.0 [V], 1.4 [A])**
No Load Speed	52 [rev/min] (at 9.0 [V]) 65 [rev/min] (at 11.1 [V]) **70 [rev/min] (at 12.0 [V])**
Operating Temperature	-5 ~ +80 [°C]
Input Voltage	6.5 ~ 14.8 [V] (**Recommended : 12.0 [V]**)
Command Signal	Digital Packet
Physical Connection	TTL Level Multidrop Bus Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
ID	253 ID (0 ~ 252)
Feedback	Position, Velocity, Load, Realtime tick, Trajectory, Temperature, Input Voltage, etc
Case Material	Engineering Plastic (Body)
Gear Material	Full Metal Gear
Standby Current	46 [mA]

Slika 3.3 Tehničke karakteristike Dynamixel XC430-W240-T motora [9]

Od svih karakteristika na slici 3.3 važne su sljedeće:

- Brzina prijenosa podataka (Baud Rate)
- Kontrolni algoritam (Control algorithm)
- Operativni modovi (Operating modes)
- Težina (Weight)
- Dimenzije (Dimensions)
- Moment zastoja (Stall torque)
- Ulazni napon (Input Voltage)
- ID
- Povratna informacija (Feedback)

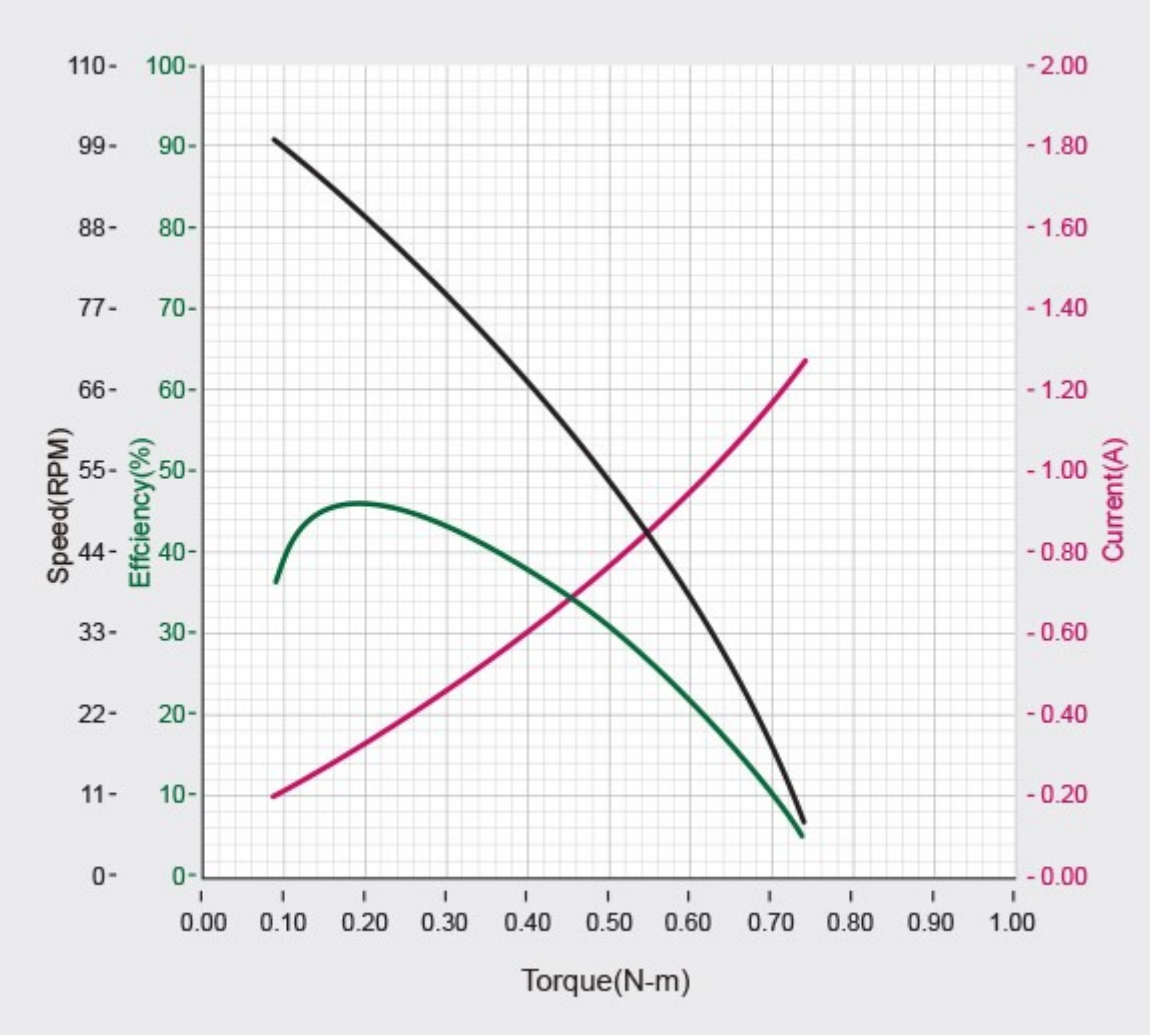
Težina je važna zbog toga što motori predstavljaju jednu od glavnih komponenti koje povećavaju moment potreban za rad robotske ruke. U skladu s njihovom težinom i maksimalnim momentom kojeg mogu ostvariti, kasnije su u konstrukciji određene dužine svakog dijela ruke, kako bi se ostvario dovoljan moment za rad. Dimenzije su bitne za montažu motora na konstrukciju. Kontrolni algoritam koji je ovdje PID , je važan jer je potrebno namjestiti parametre PID regulatora da motori pod određenim opterećenjem uspješno obave zadatak pozicioniranja. Nadalje za pozicioniranje je bitno odabrati operativni režim za kontrolu pozicije. Povratne informacije mogu koristiti u slučajevima kada je potrebno saznati trenutnu poziciju ili opterećenje motora. Brzina prijenosa kao i ID su važni parametri koje treba postaviti da bi svi motori mogli raditi skupa kao cjelina. ID služi kako bi svaki motor imao svoj naziv, moguće je upravljati s 256 motora odjednom u teoriji, u praksi bi to bilo nešto teže jer svaki od motora zahtjeva popriličan iznos struje.

Iza slike 3.2 i slike 3.3 se može vidjeti kako maksimalan moment motora nije isti, to je iz razloga što je na slici 3.2 prikazan realni i testirani moment s teretom koji raste, dok je na slici 3.3 prikazan teoretski maksimalan moment određen dizajnom motora, koji se može ostvariti da ne dolazi do pregaranja motora.

3.2.2. Tehničke karakteristike Dynamixel XC430-W150-T

MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5601
Motor	Coreless
Baud Rate	9,600 [bps] ~ 4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Operating Modes	Velocity Control Mode Position Control Mode (0 ~ 360 [°]) Extended Position Control Mode (Multi-turn) PWM Control Mode (Voltage Control Mode)
Weight	65 [g]
Dimensions (W x H x D)	28.5 x 46.5 x 34 [mm]
Gear Ratio	159.59 : 1
Stall Torque	1.2 [N.m] (at 9.0 [V], 1.1 [A]) 1.4 [N.m] (at 11.1 [V], 1.3 [A]) **1.6 [N.m] (at 12.0 [V], 1.4 [A])**
No Load Speed	80 [rev/min] (at 9.0 [V]) 99 [rev/min] (at 11.1 [V]) **106 [rev/min] (at 12.0 [V])**
Operating Temperature	-5 ~ +80 [°C]
Input Voltage	6.5 ~ 14.8 [V] (**Recommended : 12.0 [V]**)
Command Signal	Digital Packet
Physical Connection	TTL Level Multidrop Bus Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
ID	253 ID (0 ~ 252)
Feedback	Position, Velocity, Load, Realtime tick, Trajectory, Temperature, Input Voltage, etc
Case Material	Engineering Plastic (Body)
Gear Material	Full Metal Gear
Standby Current	46 [mA]

Slika 3.4 Tehničke karakteristike Dynamixel XC430-W150-T motora [10]



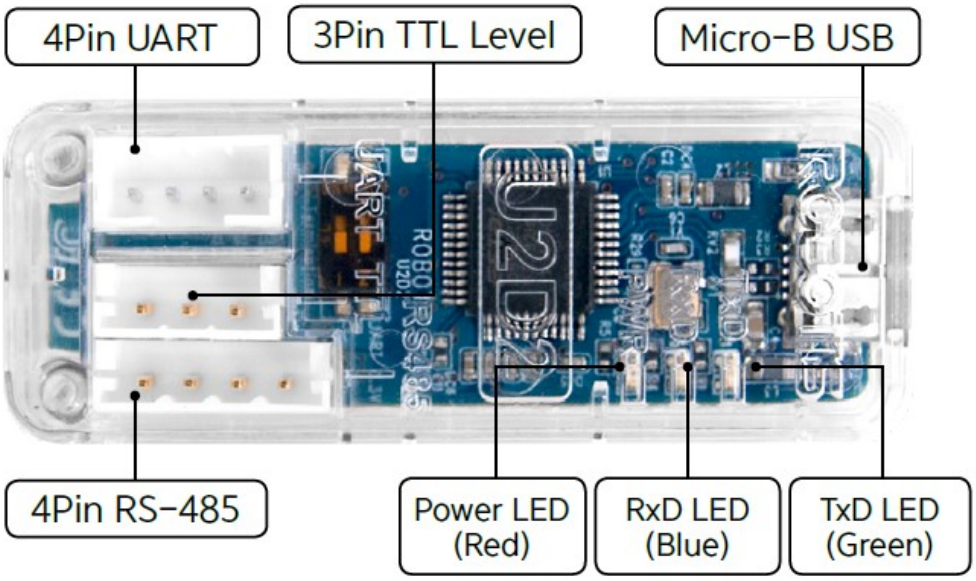
Slika 3.5 Graf performansi Dynamixel XC430-W150-T motora [10]

Iz prethodne dvije slike se vidi kako su karakteristike Dynamixel XC430-W150-T i Dynamixel XC430-W240-T servo motora identične s jedinom razlikom u prijenosnom omjeru. Manji prijenosni omjer kod W150 modela će se koristiti za aktiviranje hvataljke, gdje nije potreban veliki moment već brzina, dok će se motori modela W240 s većim momentom koristiti kao glavni za pokretanje robotske ruke.

3.2.3. Upravljanje s motorima

Dynamixel motori se mogu upravljati na razne načine, no svi koriste određeni komunikacijski konverter pomoću kojeg se preko USB serijske komunikacije ostvaruje veza između računala ili mikroupravljača sa samim motorima.

Za upravljanje je izabran U2D2 komunikacijski konverter, koji je kao takav upravo i namijenjen za rad s motorima koji se koriste.



Slika 3.6 U2D2 komunikacijski konverter

Kao što se može vidjeti na slici 3.6 ovaj konverter koristi Micro-B USB ulaz za spajanje na računalo, a kao izlaz se nudi nekoliko opcija. Može se koristiti četvero žičani UART protokol, zatim slijedi četvero žičani RS-485 protokol i na kraju TTL razina komunikacije koja će se u ovom slučaju koristit za komunikaciju s robotskom rukom. Također postoji i nekoliko led žarulji na samom konverteru, koje indiciraju kada se određene informacije šalju prema motorima, odnosno kada prima informacije s motora. Kada se uređaj spoji na računalo signalizira povezivanje s crvenom led žaruljom koja svijetli cijelo vrijeme dok je ostvarena veza s računalom. Jako je važno saznati na koji je točno serijski port spojen naš uređaj, što se vrlo lako može saznati prateći slijedeće korake: Windows search → Device Manager → Ports (COM & LPT). U padajućem izboru se vidi serijska veza, te na koji port je spojena. Ovo će biti važno kasnije kod početnog postavljanja motora te programiranja istih u MATLAB-u.

3.2.4. Napajanje

Kao uređaj za napajanje je izabran SMPS2Dymaixel što prikazuje slika 3.7 ,koji omogućava međusobno povezivanje i napajanje motora. Pruža četvero žičane odnosno trožičane konektore za bilo koji od prethodno spomenuta 3 komunikacijska protokola. SMP2Dynamixel samo služi kao sudionik u napajanju sustava no glavno napajanje se još mora dovesti na njega. Za glavno napajanje je odabran AC/DC pretvarač firme MEAN-WELL, model LRS-72-12 prikazan na slici 3.8. Kod odabira pretvarača je važno vidjeti da li zadovoljava potrebne zahtjeve. Ako se pogleda na sliku 3.3 možemo vidjet da je preporučeni napon napajanja za odabrane motore 12 V što ovo napajanje pruža(zadnji broj u modelu označava razinu DC napona). Broj 72 u nazivu modela označava prividnu snagu ovog napajanja, iz čega je lako zaključiti da napajanje može pružiti struju od maksimalno 6 A. Ako se ponovo pogleda slika 3.3 vidi se da kod zastojnog momenta struja iznosi 1,4 A. Kako je u slučaju robotske ruke najopterećeniji motor M1 sa slike 3.1, on će najviše vući struje no ni ona ne bude išla prema 1,4 A. Ostali motori su slabije opterećeni te će vući znatno manje od 1 A. S time na umu 6 A je itekako dovoljno da pogoni sve motore u robotskoj ruci.

Slika 3.7 SMPS2dynamixel

Slika 3.8 MEAN-WELL LRS-72-12

3.3. Povezivanje upravljačkog sustava i parametri motora

3.3.1. Povezivanje upravljačkog sustava

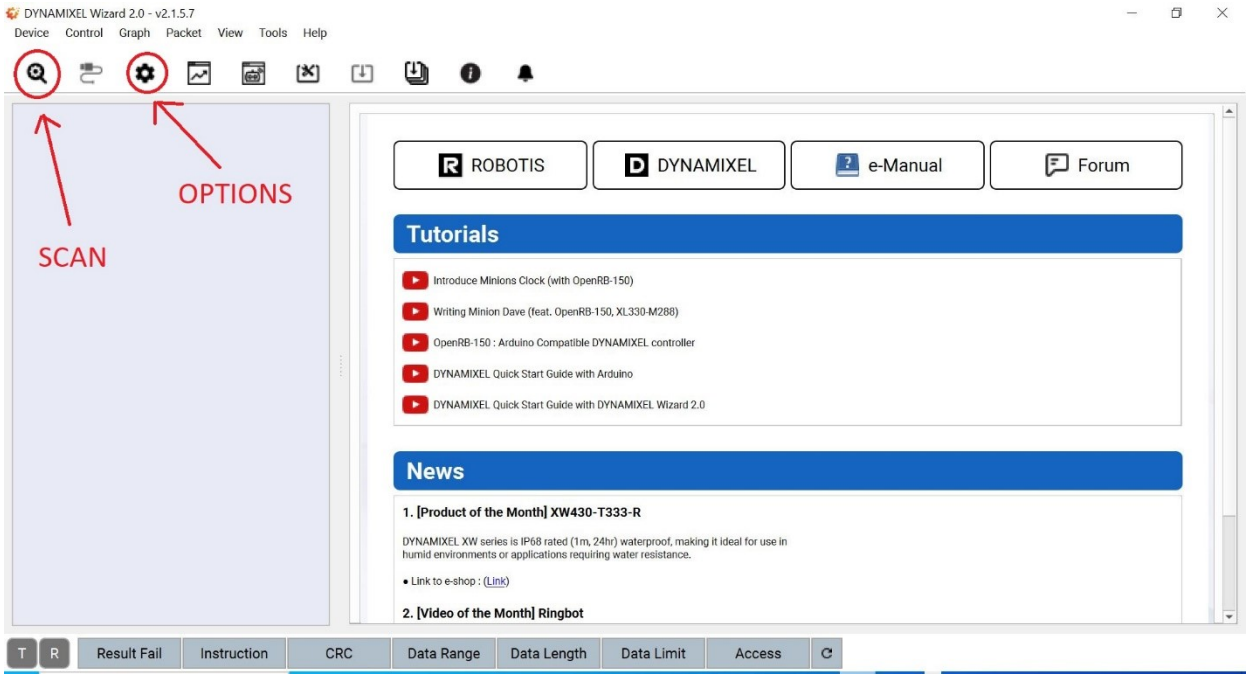
Povezivanje cijelog sustava za upravljanje i napajanje motorima je intuitivno i jednostavno. Za povezivanje se koriste tri jednožilne žice koje se mogu nabaviti s pravilnim konektorom za spajanje koji prikazuje slika 3.9. Svaki od motora ima sa stražnje strane dva utora za konektore, te se međusobno spoje svih 5 motora. Na prvi motor jedna od žica ide na SMPS2dynamixel, također se na SMPS2dynamixel spaja i konektor spojen na TTL utor s U2D2 konvertera.

Slika 3.9 Žica za spajanje

Slika 3.10 Prikaz spojenog sustava

3.3.2. Parametri motora

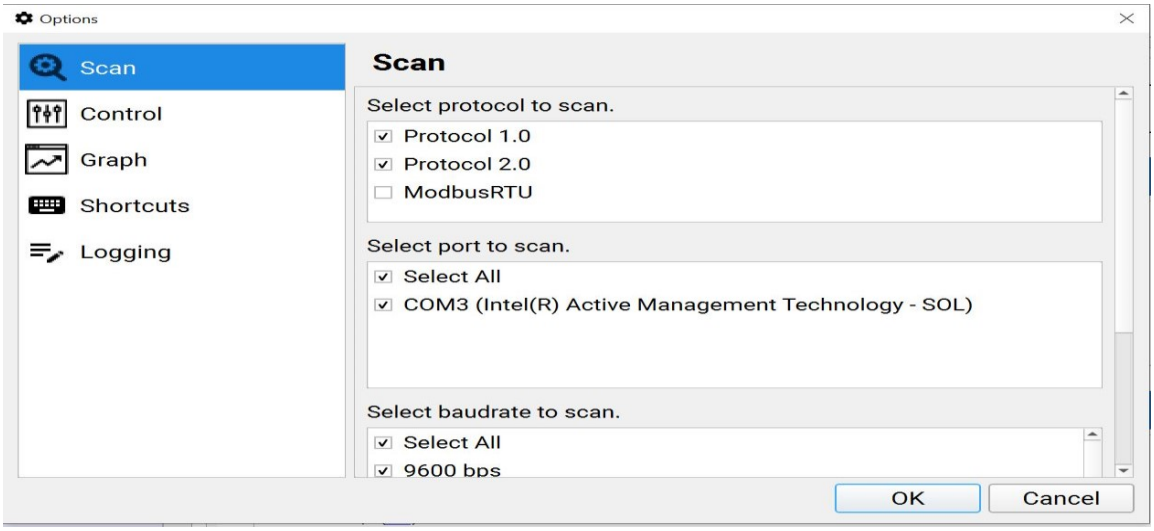
Parametri Dynamixel motora se postavljaju pomoću aplikacije DYNAMIXEL Wizard 2.0. Ova aplikacija pruža sve što je potrebno kako bi se provjerilo, da li svi motori rade, kako funkcionira njihovo pokretanje, u njoj se postavlja ID motora, baudrate, *K*p, *K*i i *K*d pojačanja za PID regulator položaja ili brzine i još mnogo postavki koje se mogu mijenjati. U aplikaciji je dobro vidjeti gdje je koja pozicija motora i u koju se stranu motor vrti ovisno o zadanoj poziciji, ovo će biti jako važno kod danjeg programiranja u MATLAB-u.



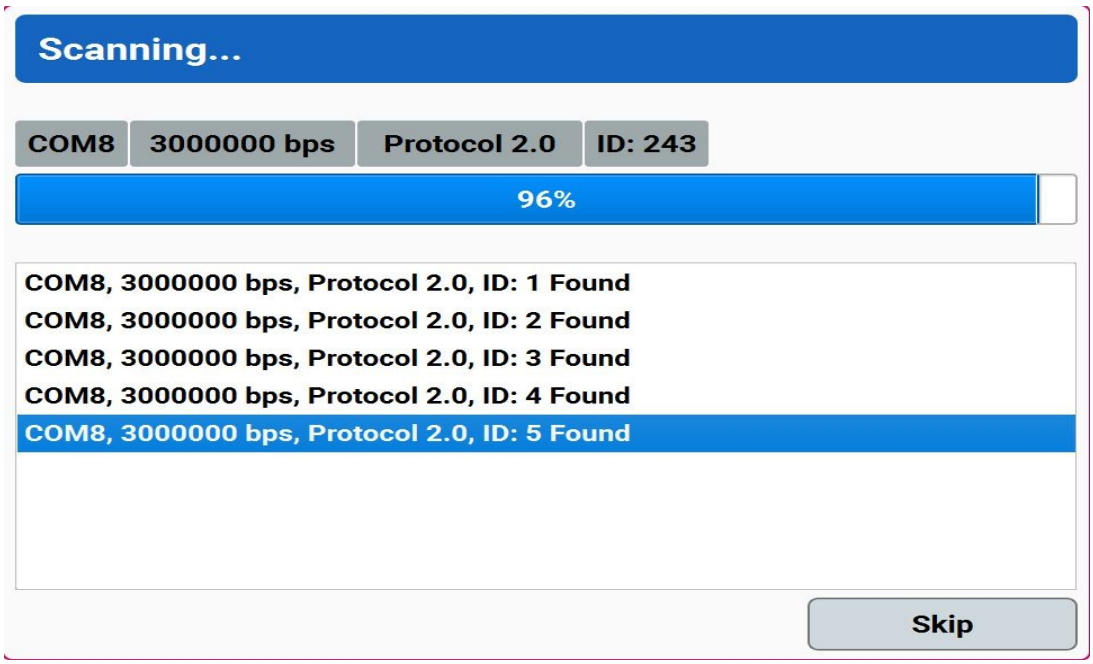
Slika 3.11 Početni prozor DYNAMIXEL wizard-a

Nakon otvaranja aplikacije potrebo je spojiti U2D2 s računalom preko USB priključka i uključiti napajanje u utičnicu. S početnog prozora su bitne trenutno samo dvije stavke SCAN i postavke. Ostale stavke u glavnoj traci služe za povezivanje ili prekid veze s motorima što se ne koristi jer se nakon pretrage motori automatski povežu, te za ažuriranje aplikacije ili ispis grafova. Kreće se od toga da se otvaraju postavke prikazane na slici 3.12, na slici je vidljiv izbornik te je trenutno bitno da se samo postavke za scan promijene. Svaki tvornički kupljeni Dynamixel-ov motor radi na baudrate-u od 57600, te su većinom na protokolu 2.0. Kod portova za skeniranje se izabire port koji je već ranije u radu pronađen pomoću Device Managera. Ako se zna na kojem su baudrate-u motori onda se odabire taj baudrate radi bržeg skeniranja koje u

suprotnome zna potrajati nekoliko minuta. Skeniranje je prikazano na slici 3.13. Svakako se preporučuje za početak da se skeniraju oba protokola i svi baudrate-ovi kako bi bili sigurni da nije ništa promaknulo. Kod skeniranja vlastitih motora je bilo rečeno kako su svi na 3000000 bps, no ispalo je da je jedan od njih bio na 1000000.

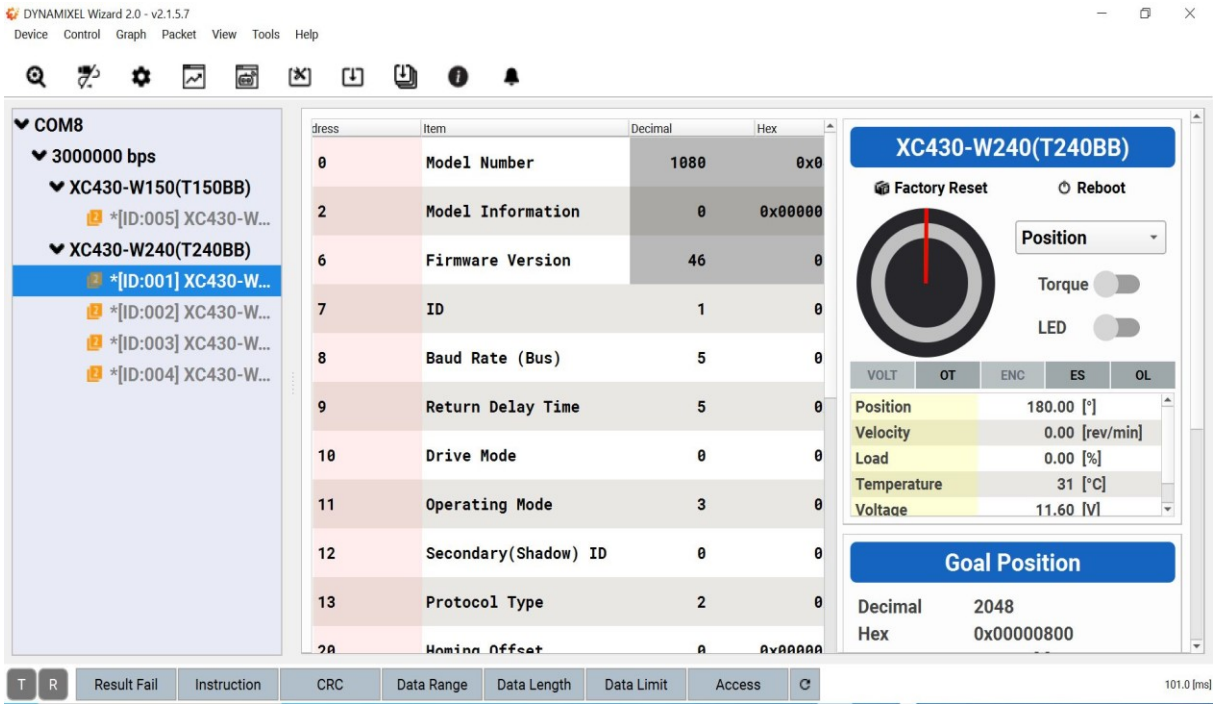


Slika 3.12 Postavke aplikacije



Slika 3.13 Skeniranje motora

Nakon što je skeniranje završeno prozor aplikacije bi trebao izgledati kao što je prikazano na slici 3.14. Sučelje je podijeljeno na 3 glavna dijela: lijevi, središnji i desni. Na lijevome dijelu su prikazani modeli svih motora, na kojem su portu, baudrate-u i koji je ID odnosno ime svakoga od njih. Desni dio pokazuje korisniku trenutnu poziciju, omogućava paljenje ugrađene led žarulje na motoru kako bi vidio o kojem se točno motoru radi i označio ih po potrebi, omogućava također i aktivaciju momenta nakon čega se može pomicati trenutna pozicija servo motora. Središnji dio sučelja je najvažniji jer se u njemu postavljaju i mijenjaju sve postavke i parametri pojedinačnog motora. Ovdje je potrebno postaviti željeni baudrate, ID i protokol te parametre za PID regulator.



Slika 3.14 Korisničko sučelje u DYNAMIXEL Wizard-u

3.4. Kinematika robotske ruke

Matematički zapis strukture robotske ruke, omogućuje nam da znamo točno gdje se izvršni član nalazi u prostoru i kako je orijentiran. Postoje unutarnje i vanjske koordinate robotske ruke, unutarnje su povezane s motorima odnosno aktuatorima koji pokreću robota jed.(3.1) , a vanjske su povezane s položajem i orijentacijom izvršnog člana jed.(3.2).

$$\boldsymbol{q = [q1\ q2\ q3\ q4\ q5\ q6]} \tag{3.1}$$

$$\boldsymbol{R = [x\ y\ z\ \theta\ \varphi\ \psi]} \tag{3.2}$$

Postoje dvije vrste problema koji se pomoću tih koordinata rješavaju, jedan je direktni (engl. forward), a drugi je inverzni (engl. inverse). Direktni problem se rješava pomoću metoda za direktnu kinematiku, dok se inverzni koji je značajno teži može riješiti inverznom kinematikom.

3.4.1. Direktna kinematika

Opisivanje robotske ruke preko direktne kinematike omogućuje određivanje pozicije i orijentacije izvršnog člana. Ima nekoliko načina da se ovo napravi, u ovom radu će se opisati samo dvije. Obje koriste matrice čijim međusobnim množenjem dolazimo do krajnje matrice transformacije. Prvo je objašnjen osnovni princip te način kako bi se izvela direktna kinematika preko osnovnih matrica rotacije i translacije.

Osnovne matrice rotacije prema [11] su:

$$\boldsymbol{Ax = Rot(x,\alpha)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\cos(\alpha)} & \mathbf{-\sin(\alpha)} & \mathbf{0} \\ \mathbf{0} & \mathbf{\sin(\alpha)} & \mathbf{\cos(\alpha)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{Ay = Rot(y,\alpha)} = \begin{bmatrix} \mathbf{\cos(\alpha)} & \mathbf{0} & \mathbf{\sin(\alpha)} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{-\sin(\alpha)} & \mathbf{0} & \mathbf{\cos(\alpha)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.4}$$

$$\boldsymbol{Az = Rot(z,\alpha)} = \begin{bmatrix} \mathbf{\cos(\alpha)} & \mathbf{-\sin(\alpha)} & \mathbf{0} & \mathbf{0} \\ \mathbf{\sin(\alpha)} & \mathbf{\cos(\alpha)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.5}$$

Matrica translacije je :

$$\boldsymbol{At = Tran(a,b,c)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{x} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{y} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{z} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.6}$$

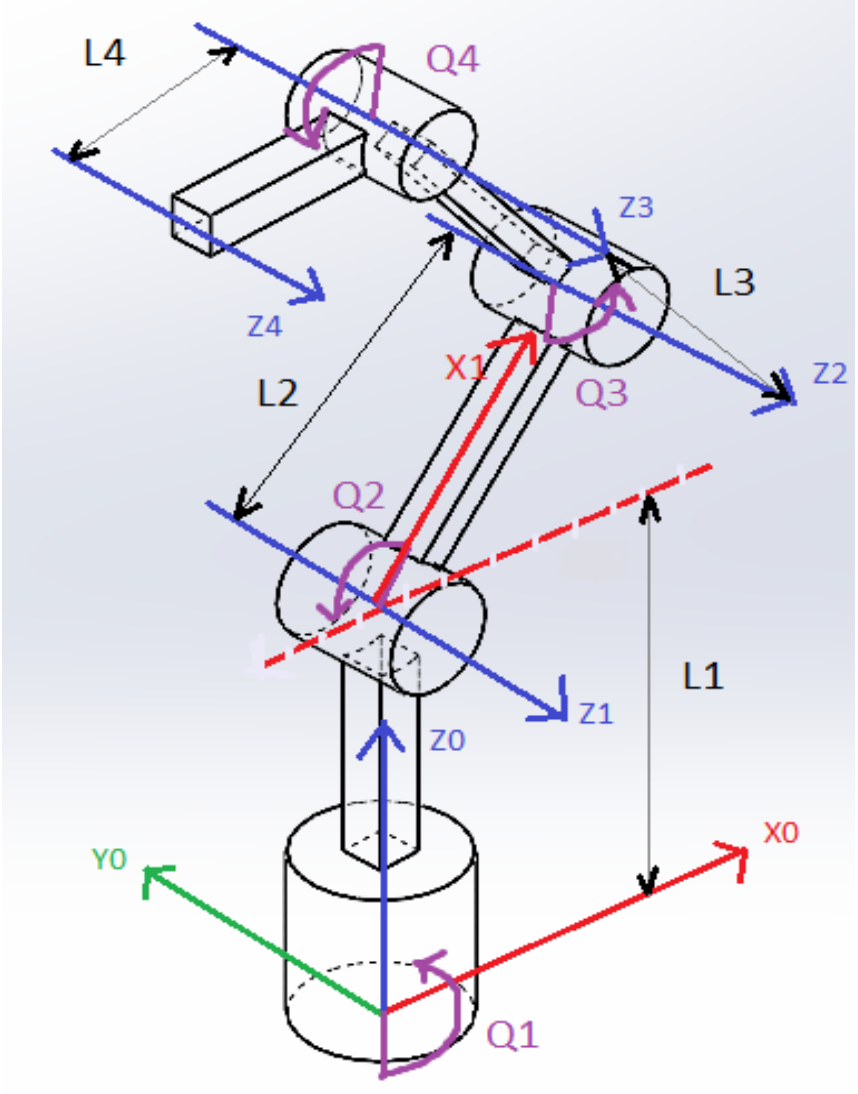
Svaki od prva tri stupca u matricama rotacije predstavljaju orijentacije osi novog koordinatnog sustava. To su zapravo jedinični vektori čijim se zbrojem po komponentama odnosno redcima dobije nova os sustava. Matrica translacije je poprilično jasna, osi koordinatnoga sustava ostaju kakve su bile i prije translacije, to se može vidjeti po tome što su nove osi u istom smjeru kao i stare, ali se položaj promijenio. Koordinata „*x*“ predstavlja pomak po x-osi , „*y*“ po y-osi, a onda time i „*z*“ po z-osi. Kako bi se kinematika opisala potrebno je iz ovih matrica translacije i rotacija dobiti homogene matrice transformacija.

Kratki opis izvoda direktne kinematike preko homogenih transformacija:

Polazi se od nultog koordinatnog sustava koji je većinom smješten u koordinatama (0,0,0) te predstavlja središte robota gdje on dodiruje „tlo“ odnosno površinu na kojoj je pričvršćen. Zatim se on translatira do prvog motora te se rotira oko osi vrtnje motora. Međusobnim množenjem matrice translacije pa zatim rotacije smo dobili homogenu matricu. Ovdje je jako važno da li se prvo stavlja rotacija ili translacija pošto umnožak matrica ovisi o redoslijedu. Ovaj postupak se nastavlja sve do izvršnog člana, do kojega se većinom na kraju dolazi translacijom čije množenje znatno komplicira stvari. Nakon množenja svih homogenih transformacija završna matrica nam govori o koordinatama izvršnoga člana u ovisnosti o kutovima zakreta svakih od motora ili translacije koju oni izvršavaju. Jednadžbe za koordinate *x*, *y* i *z* su smještene u zadnjem stupcu redom. Važno je naglasiti kako se množenje matrica znatno komplicira sa njihovim brojem te izrazi mogu biti izuzetno dugački i komplicirani. Oni se mogu znatno pojednostaviti s pravilima o sinusu i kosinusu zbroja kutova i trigonometrijskim pravilima. U ovom radu će se kinematika izvesti pomoću drugog postupka koji se naziva Denavit-Hartenbergov postupak, odnosno metoda. Detaljniji opis se može naći u literaturi [12] kao i [11] koje daju dublji uvid u prethodno navedeno.

Denavit-Hartenberg metoda

Denavit-Hartenberg (DH) konvencija se često koristi u kinematičkoj analizi robotskog manipulatora . Temelji se na postavljanju koordinatnog sustava na svaki zglob i određivanju četiri parametra poznata kao DH parametri za svaku kariku, te korištenju tih parametara za izradu DH tablice. Na kraju se dobiva transformacijska matrica između različitih koordinatnih sustava. Na samome početku potrebno je na skici robotske ruke označiti nulti koordinatni sustav te ostale osi. Os z_{i-1} koordinatnog sustava (i-1) leži u osi gibanja i-tog stupnja slobode gibanja. Problem se rješavao po uzoru na literaturu [11] i [8].



Slika 3.15 Skica robotske ruke s DH parametrima

Postoje 4 osnovna parametra koja je potrebno odrediti za svaki pojedini članak robotske ruke. Osnovni parametri su prikazani u tablici 3.1, te se ispod iste nalaze opisi svakog od parametara.

Tablica 3.1 DH parametri robotske ruke

Članak (i)	a_i	α_i	d_i	θ_i
1	0	90	$L1$	$q1$
2	$L2$	0	0	$q2$
3	$L3$	0	0	$q3$
4	$L4$	0	0	$q4$

a_i - udaljenost u smjeru X_i od Z_{i-1} do Z_i
α_i - kut od Z_{i-1} do Z_i gledajući X_i
d_i - udaljenost uzduž Z_{i-1} do X_{i-1} do X_i
θ_i – kut zakreta motora gledajući pravilo desne ruke

Transformacija od T_i do T_{i-1} članka je dana standardnom matricom transformacije kod Denavit-Hartenbergove metode iz literature [11] koja glasi :

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{\cos(\theta i)} & \boldsymbol{-\sin(\theta i)\cos(\alpha i)} & \boldsymbol{\sin(\theta i)\sin(\alpha i)} & \boldsymbol{ai * \cos(\theta i)} \\ \boldsymbol{\sin(\theta i)} & \boldsymbol{\cos(\theta i)\cos(\alpha i)} & \boldsymbol{-\cos(\theta i)\sin(\alpha i)} & \boldsymbol{ai * \sin(\theta i)} \\ \boldsymbol{0} & \boldsymbol{\sin(\alpha i)} & \boldsymbol{\cos(\alpha i)} & \boldsymbol{di} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \quad (\mathbf{3.7})$$

Međusobnim množenjem matrica od i=1 do i=4 dobiva se ukupna transformacijska matrica robotske ruke. Pomoću zadnjeg stupca dobivene matrice dolazi se do rješenja direktne kinematike za robota.

$$^0T_4 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4$$

Kako je množenje 4x4 matrica ručno zahtjevan posao i mogućnost za grešku je velika, to nije najbolja opcija kojom možemo riješiti ovaj problem. MATLAB nudi simboličko računanje što je za matrice gdje imamo razne varijable koje se mijenjaju odlično, jer trebamo dobiti opće rješenje matrice u koju zatim samo ubacujemo potrebne DH parametre od kojih se kutovi (θ_i) konstantno mijenjaju.

Pomoću prefiksa syms MATLAB zna da će slijedeće varijable smatrati kao simbole. Da bi ovo radilo potrebno je imati instaliran Add-on koji se naziva Symbolic Toolbox. Zatim se sve standardne matrice po DH metodi (vidjeti jed. (3.7)), upišu i pomnože kako bi se dobila ukupna matrica transformacije za robotsku ruku. Iz dobivene matrice su važna samo prva tri retka u zadnjem stupcu pošto oni govore o poziciji novog koordinatnog sustava u odnosu na nulti (koordinate izvršnog člana). Na slici 3.16 se može vidjeti kod za spomenuto dobivanje konačne transformacijske matrice. Kako se na slici ne vide dobivena rješenja ona će naknadno biti napisana.

```
syms l1 l2 l3 l4 q1 q2 q3 q4;
A1=[cos(q1) 0 sin(q1) 0; sin(q1) 0 -cos(q1) 0; 0 1 0 l1; 0 0 0 1 ];
A2=[cos(q2) -sin(q2) 0 l2*cos(q2); sin(q2) cos(q2) 0 l2*sin(q2); 0 0 1 0; 0 0 0 1 ];
A3=[cos(q3) -sin(q3) 0 l3*cos(q3); sin(q3) cos(q3) 0 l3*sin(q3); 0 0 1 0; 0 0 0 1 ];
A4=[cos(q4) -sin(q4) 0 l4*cos(q4); sin(q4) cos(q4) 0 l4*sin(q4); 0 0 1 0; 0 0 0 1 ];

A=A1*A2*A3*A4;
A(1,4)
A(2,4)
A(3,4)
```

Slika 3.16 Kod u MATLAB-u za direktnu kinematiku

Nakon pokretanja programa dobivaju se jednadžbe za dobivanje *x*, *y* i *z* koordinata izvršnog člana.

$$.x = l2 * cos(q1) * cos(q2) + l4 * cos(q4) * (cos(q1) * cos(q2) * cos(q3) - cos(q1) * sin(q2) * sin(q3)) - l4 * sin(q4) * (cos(q1) * cos(q2) * sin(q3) + cos(q1) * cos(q3) * sin(q2)) + l3 * cos(q1) * cos(q2) * cos(q3) - l3 * cos(q1) * sin(q2) * sin(q3)$$

(3.8)

$$.y = l2 * cos(q2) * sin(q1) - l4 * sin(q4) * (cos(q2) * sin(q1) * sin(q3) + cos(q3) * sin(q1) * sin(q2)) - l4 * cos(q4) * (sin(q1) * sin(q2) * sin(q3) - cos(q2) * cos(q3) * sin(q1)) + l3 * cos(q2) * cos(q3) * sin(q1) - l3 * sin(q1) * sin(q2) * sin(q3)$$

(3.9)

$$.z = l1 + l2 * sin(q2) + l4 * cos(q4) * (cos(q2) * sin(q3) + cos(q3) * sin(q2)) + l4 * sin(q4) * (cos(q2) * cos(q3) - sin(q2) * sin(q3)) + l3 * cos(q2) * sin(q3) + l3 * cos(q3) * sin(q2)$$

(3.10)

Ovako dobivene jednadžbe izgledaju komplicirano te ih kao takve nije praktično koristiti u programu robota zbog nepreglednosti i kasnije za dobivanje inverzne kinematike, je ove jednadžbe potrebno pojednostaviti. Sve tri jednadžbe se mogu poprilično jednostavno pojednostaviti, ako se uoče određena ponavljanja, te upotrijebe jednadžbe i pravila za sinus i kosinus zbroja kutova. U nastavku je raspisan cijeli postupak pojednostavljenja prethodno dobivenih jednadžbi iz MATLAB-a.

Pojednostavljene jednadžbe (3.8):

U daljnjem raspisivanju će se koristiti sljedeća pojednostavljenja:

cos(*q*i) = ci , sin(*q*i) = si , sin(*q*1+*q*2+*q*3) = s123 i slično

U jed. (3.8) se može primijetiti kako se C1 pojavljuje u svim članovima te ga je moguće izlučiti ispred svih ostalih izraza, a njih staviti pod zagrade.

Time se dobiva sljedeći izraz:

x = c1*(*L*2*c2 + *L*4*c4*(c2*c3-s2*s3) – *L*4*s4*(c2*s3 + c3*s2) + *L*3*(c2*c3 – s2*s3))

Moguće je uočiti kako se u članovima s *L*4 i *L*3 pojavljuju izrazi za sinuse i kosinuse zbrojeva kutova te se nakon uvođenja pravila dobiva sljedeći izraz:

x = c1*(*L*2*c2 + *L*4*(c4*c23 – s4*s23) + *L*3*c23)

Nadalje, uz član *L*4 se ponovo pojavljuje kosinus zbroja kutova te je konačan izraz:

$$x = \cos(q1)*[L2*\cos(q2) + L3*\cos(q2+q3) + L4*\cos(q2+q3+q4)] \quad \textbf{(3.11)}$$

Može se primijetiti kako *L*1 ne utječe na *x* koordinatu robotske ruke, što je i intuitivno jer je to samo visina robota do motora M2. Svakako treba kritički razmatrati svako dobiveno rješenje i ne uzimati sve zdravo za gotovo. Da se ovdje pojavio *L*1 to bi značilo da su krivo napisane matrice transformacija.

Pojednostavljenje jednadžbe (3.9):

I dalje važe ranije spomenuta i objašnjena pojednostavljenja u raspisivanju jednadžbe.

U jed.(3.9) je moguće primijetiti da je vrlo slična onoj za dobivanje x koordinate. U ovom slučaju je moguće izlučiti s1, te nastaviti s raspisivanjem.

y = s1*($L2$*c2 – $L4$*s4*(c2*s3 + c3*s2) – $L4$*c4*(s2*s3 – c2*c3) + $L3$*(c2*c3 – s2*s3))

Moguće je uočiti kako se u članovima s L4 i L3 pojavljuju izrazi za sinuse i kosinuse zbrojeva kutova te se nakon uvođenja pravila dobiva sljedeći izraz:

y = s1*($L2$*c2 – $L4$*(s4*s23 – c4*c23) + $L3$*c23)

Nadalje, uz član L4 se ponovo pojavljuje kosinus zbroja kutova te je konačan izraz:

$y = \sin(q1)*[\ L2*\cos(q2) + L3*\cos(q2+q3) + L4*\cos(q2+q3+q4)\ ]$

Kao i kod izraza za x, $L1$ nam ne utječe na y koordinatu što potvrđuje da je izraz dobro dobiven. Također se vidi kako su izrazi za x i y koordinate identični osim toga što je kod y koordinate sinus ispred zagrade, a kod x koordinate je kosinus.

Pojednostavljenje jednadžbe (3.10):

Pogledom na jed.(3.10), može se uočiti da nije moguće izlučiti nijedan član, no ponovo se pojavljuju sinusi i kosinusi zbroja kutova te to koristimo kako bi pojednostavili izraz.

$z = L1 + L2*s2 + L4*c4*s23 + L4*s4*c23 + L3*s23$

Konačni izraz glasi:

$z = L1 + L2*\sin(q2) + L3*\sin(q2+q3) + L4*\sin(q2+q3+q4)$

Može se vidjeti da koordinata z jedina sadrži $L1$ što ukazuje na dobro riješenu jednadžbu. Ovakvo pojednostavljenje jednadžbi pomoću zbroja kutova je vrlo često kod izvođenja direktne kinematike, bilo da se radi pomoću homogenih matrica transformacija ili pomoću Denavit-Hartenberg-ove metode. Ova rješenja su vrlo laka i za predočiti te se moglo doći do istih pomoću geometrije i trigonometrije.

Nakon dobivenih koordinata položaja hvataljke još preostaje dobiti i njenu orijentaciju, jer robot s 4 stupnja slobode gibanja omogućuju i kontrolu jedne od 3 rotacija oko koordinatnih osi hvataljke. Moguće je zamisliti kako radi robot, kada se robotska ruka okreće oko svoje z osi ($q1$) vidi se kako hvataljka ne mijenja svoju orijentaciju u prostoru već samo položaj. Kada se aktivira bilo koji od preostalih motora (M2,M3 ili M4) odnosno zakrenemo robotsku ruku za ($q2$, $q3$ ili $q4$), orijentacija hvataljke se mijenja u ovisnosti od sva tri prethodno spomenuta kuta. S time na umu, 4 stupanj slobode gibanja je rotacija hvataljke oko svoje z osi, kao što je nacrtano na slici 3.15.

Ukupan kut nagiba hvataljke je određen sljedećom jednadžbom:

$$\boldsymbol{\theta = q2 + q3 + q4} \qquad (\mathbf{3.12})$$

Time je direktna kinematika robota s 4 stupnja slobode gibanja u potpunosti određena. Potrebo je napomenuti problem do kojeg može doći u posebnim slučajevima, a to je singularnost.

Singularnost robotske ruke

Pojam singularnosti je poznat kod robotičara i govori o singularnom položaju robota. Identificiranje singularnosti je važno zbog mnogo razloga:

1. Singularnosti predstavljaju konfiguracije iz kojih određeni smjerovi gibanja mogu biti nedostižni.
2. Kod singularnosti, ograničene brzine izvršnog člana mogu odgovarati neograničenim brzinama zglobova.
3. Kod singularnosti, ograničene sile i momenti na izvršnome članu mogu odgovarati neograničenim momentima u zglobovima.
4. **Singularnosti obično (ali ne uvijek) odgovaraju točkama na granici radnog prostora manipulatora, odnosno točkama maksimalnog dosega manipulatora.**
5. Singularnosti odgovaraju točkama u radnom prostoru manipulatora koje mogu biti nedostižne pod malim perturbacijama parametara veze, poput duljine, pomaka, itd.
6. **U blizini singularnosti ne postoji jedinstveno rješenje problema inverzne kinematike. U takvim slučajevima možda neće postojati rješenje ili će postojati beskonačno mnogo rješenja.** [11]

Kako bi opisali singularnost postoji mnogo načina no svaki od njih zahtjeva poznavanje Jacobian-ove matrice [8].

Jacobian-ova matrica daje odnos između brzina izvršnog člana i brzina vrtnje motora, a predočena je sljedećim izrazom i jednadžbom:

$$J = \frac{\partial R}{\partial q}$$

Vektori **R** i **q** su dani u jed. (3.2) i jed.(3.1). Potpuno raspisana Jacobian-ova matrica je dana sljedećim izrazom (3.13).

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial x}{\partial q1} & \frac{\partial x}{\partial q2} & \frac{\partial x}{\partial q3} & \frac{\partial x}{\partial q4} & \frac{\partial x}{\partial q5} & \frac{\partial x}{\partial q6} \\ \frac{\partial y}{\partial q1} & \frac{\partial y}{\partial q2} & \frac{\partial y}{\partial q3} & \frac{\partial y}{\partial q4} & \frac{\partial y}{\partial q5} & \frac{\partial y}{\partial q6} \\ \frac{\partial z}{\partial q1} & \frac{\partial z}{\partial q2} & \frac{\partial z}{\partial q3} & \frac{\partial z}{\partial q4} & \frac{\partial z}{\partial q5} & \frac{\partial z}{\partial q6} \\ \frac{\partial \theta}{\partial q1} & \frac{\partial \theta}{\partial q2} & \frac{\partial \theta}{\partial q3} & \frac{\partial \theta}{\partial q4} & \frac{\partial \theta}{\partial q5} & \frac{\partial \theta}{\partial q6} \\ \frac{\partial \varphi}{\partial q1} & \frac{\partial \varphi}{\partial q2} & \frac{\partial \varphi}{\partial q3} & \frac{\partial \varphi}{\partial q4} & \frac{\partial \varphi}{\partial q5} & \frac{\partial \varphi}{\partial q6} \\ \frac{\partial \psi}{\partial q1} & \frac{\partial \psi}{\partial q2} & \frac{\partial \psi}{\partial q3} & \frac{\partial \psi}{\partial q4} & \frac{\partial \psi}{\partial q5} & \frac{\partial \psi}{\partial q6} \end{bmatrix} \tag{3.13}$$

Položaj robota je singularan kada je determinanta Jacobian-ove matrice jednaka nuli:

$$\det(\mathrm{J}(q)) = 0$$

Kao što je ranije navedeno ova matrica se koristi kao poveznica između brzine unutarnjih i vanjskih koordinata robotske ruke. Sljedeći izrazi prikazuju upravo taj odnos, a koristit će se u nastavku za objašnjenje jednog od načina računanja inverzne kinematike.

$$\dot{\boldsymbol{R}} = \boldsymbol{J}(\boldsymbol{q}) * \dot{\boldsymbol{q}} \tag{3.14}$$

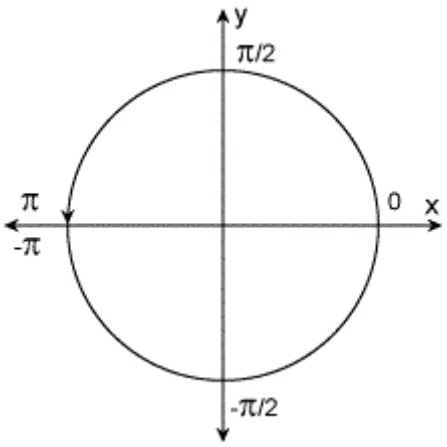
$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q}) * \dot{\boldsymbol{R}} \tag{3.15}$$

3.4.2. Inverzna kinematika

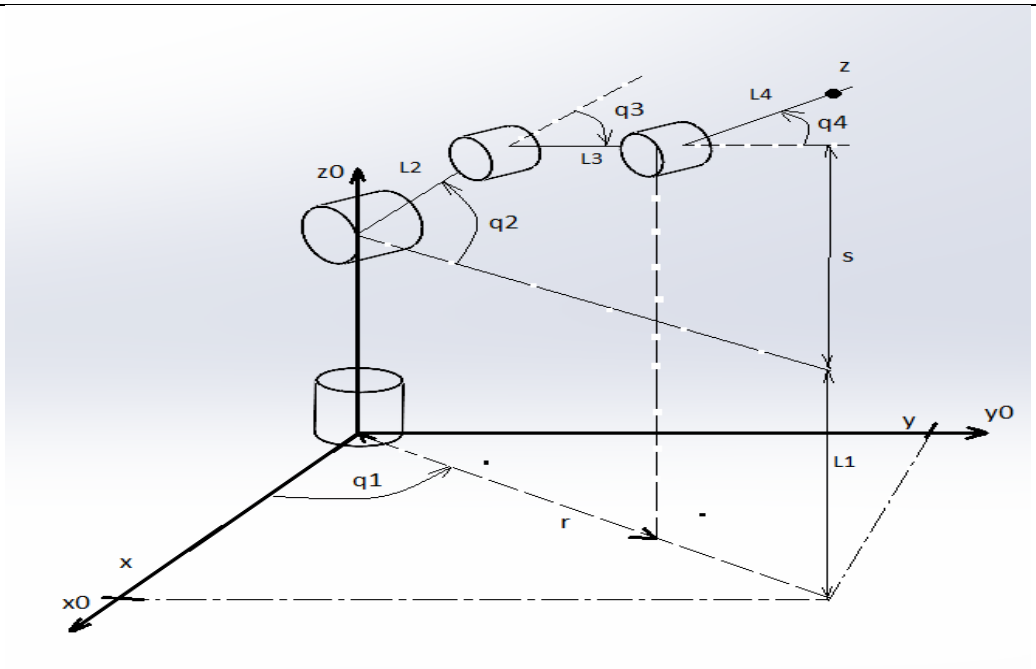
Inverzna kinematika, kao što joj i samo ime govori, služi za rješavanje suprotnog problema od direktne kinematike. Kod rješavanja inverzne kinematike robota cilj je pomoću poznatih koordinata izvršnog člana doći do unutarnjih koordinata (kutova) servo motora. Kao takva mnogo je zahtjevnija od direktne kinematike i ne mora uvijek biti moguće doći do nje pomoću neke od tipičnih metoda. Postoji mnogo načina za riješiti inverznu kinematiku, no u ovome dijelu će biti ukratko objašnjena samo tri od kojih će se jedan koristiti za ovaj rad.
Prvi i najintuitivniji način za riješiti inverznu kinematiku je pomoću geometrije, zatim slijedi analitičko rješavanje, te kao treći način će biti spomenut numerički postupak koji koristi Jacobian-ovu matricu.

Geometrijski postupak

Geometrijski je postupak u većini slučajeva za jednostavnije strukture robotskih manipulatora najjednostavnija i najintuitivnija metoda za rješavanje inverznog problema. Lako se može predočiti i objasniti, te daje potpuno razumijevanje u rješenje problema.
Opća ideja geometrijskog pristupa je riješiti za zglobnu varijablu q_i, projiciranjem manipulatora na ravninu $x_{i-1} - y_{i-1}$, te rješavanjem jednostavnog trigonometrijskog problema [11].
Pri rješavanju trigonometrije će se koristiti atan2, što je u svojoj biti obični inverzan tangens, ali u intervalu $[-\pi, \pi]$ umjesto intervala $[-\pi/2, \pi/2]$, što ga čini četvero kvadratnim. Ovo je važno zbog toga što postoji više rješenja ovisno o tome u kojemu se kvadrantu nalazi izvršni član robotske ruke. Kao pomoć pri rješavanju je korištena literatura [11] i [13].



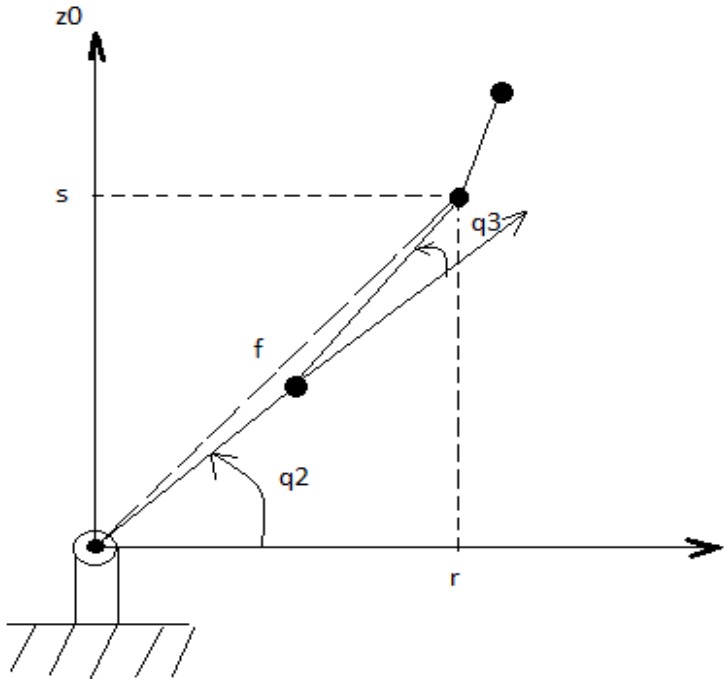
Slika 3.17 Kvadranti atan2 operacije



Slika 3.18 Skica geometrije robotske ruke

Pogledom na sliku 3.18 moguće je odmah uočiti kako će se do $q1$ doći vrlo lako. Pomoću $\boldsymbol{x}$ i $\boldsymbol{y}$ koordinata moguće je doći do kuta q1 preko operacije tangensa.

$$\boldsymbol{q1 = atan2(y, x)} \tag{3.16}$$



Slika 3.19 Skica za računanje kutova $q3$ i $q2$

Nakon određivanja $q1$, sljedeći kut koji se može odrediti je $q3$. Pogledom na sliku 3.19 mogu se vidjeti krakovi $L2$, $L3$ i $L4$, koordinatni sustav je već odignut za $L1$. Kako su dužine $\boldsymbol{s}$ i $\boldsymbol{r}$ označene do trećeg motora, duljina $L4$ će se oduzimati od koordinata izvršnoga člana za izračun spomenutih dužina. Krakovi $L2$, $L3$ i $\boldsymbol{f}$ čine trokut nad kojim će se primijeniti kosinusni poučak.

$$\boldsymbol{f^2 = L2^2 + L3^2 - 2 * L2 * L3 * \cos(180 - q3)} \quad (3.17)$$

Nakon raspisivanja $f^2 = r^2 + s^2$ i uvođenja [cos(180-$q3$) = -cos($q3$)] dobiva se sljedeći izraz za $q3$:

$$\boldsymbol{q3 = \cos^{-1}\left(\frac{r^2 + s^2 - L2^2 - L3^2}{2 * L2 * L3}\right)} \quad (3.18)$$

Za izračunavanje varijabli $\boldsymbol{r^2}$ i $\boldsymbol{s^2}$ se koriste sljedeći izrazi:

$$\boldsymbol{r^2 = A^2 + B^2} \quad (3.19)$$

$$\boldsymbol{s = z - L1 - L4 * \sin(\theta)} \quad (3.20)$$

$$\boldsymbol{A = x - L4 * \cos(q1) * \cos(\theta)} \quad (3.21)$$

$$\boldsymbol{B = y - L4 * \sin(q1) * \cos(\theta)} \quad (3.22)$$

Inverzan kosinus nije dobar za upotrebu u računanju inverzne kinematike, zbog toga što ne razlikuje u kojem se kvadrantu nalazi izvršni član te može dati kriva rješenja. Zbog toga je potrebno izraz pod (3.18) preoblikovati da se koristi atan2 operacija. Izraz unutar kosinusa zamijenimo s varijablom D, zatim kako bi dobili sinus iskoristimo odnos:

$$\sin(q3) = +/-\sqrt{1 - D^2}$$

Krajnji izraz za $q3$ glasi :

$$\boldsymbol{q3 = atan2\left(-\sqrt{1 - D^2}, D\right)} \quad (3.23)$$

Kako bi se odredilo da li se uzima + ili – u izrazu potrebno je napraviti test između direktne i inverzne kinematike u ovom slučaju negativan predznak daje točne rezultate svaki put, dok pozitivni zna dati pomaknute rezultate za određeni kut.

Nakon određenoga $q3$, pomoću njega se sada može odrediti kut $q2$ gledajući sliku 3.19. Gledajući *r* i *s* moguće je doći do kuta kojeg te dvije mjere rade, kako bi se dobio $q2$ potrebno je još od toga kuta oduzeti mali kut kojeg radi trokut s krakovima $L2 + L3*\cos(q3)$ i $L3*\sin(q3)$. Nakon uočenih kutova izraz za $q2$ je lako izraziti i glasi:

$$\boldsymbol{q2 = atan2(s,r) - atan2(a,b)} \qquad (\mathbf{3.24})$$
$$a = L3 * \sin(q3)$$
$$b = L2 + \mathrm{L3} * \cos(\mathrm{q3})$$

Do zadnjega kuta q4 je vrlo lako doći nakon prethodno određena 3 kuta. Za određivanje q4 se koristi jednadžba (3.12) iz koje se q4 samo izrazi u ovisnostima od ostalih varijabli i dobiva se posljednji izraz za koji dovršava inverznu kinematiku:

$$\boldsymbol{q4 = \theta - q2 - q3} \qquad (\mathbf{3.25})$$

U nastavku inverzne kinematike su kratko objašnjeni analitički i numerički postupci.
Kod ozbiljnih i profesionalnih robotskih ruku se sva 3 postupka uz još naprednije koji nisu u ovome radu obrađivani, skupa koriste za najtočnije određivanje i računanje inverznog problema.

Analitički postupak

Kako ovaj postupak nije bio korišten u rješavanju inverznog problema, on će biti samo ukratko opisan da se stekne neka osnovna ideja o rješavanju. Postoji mnogobrojna literatura u kojoj je postupak objašnjen.
Za ovaj postupak je potrebno imati poznate sve matrice transformacija koje opisuju robotsku ruku. Postupak se sastoji od toga da se izjednači umnožak inverza prve matrice transformacije s ukupnom matricom transformacija i matrica transformacije koja se dobije umnoškom svih preostalih matrica transformacija izuzet prve. Zatim se izjednačuju članovi povezani s pozicijom izvršnog člana i pokušava se riješiti skup jednadžbi. Ukoliko nije dovoljno jednadžbi iz prvog postupka , potrebno je dodatno s lijeva pomnožiti inverz s dodatnim inverzom sljedeće matrice transformacije, a na drugoj strani postaviti umnožak preostalih matrica, postupak se nastavlja dokle god se ne dobije algebarsko rješenje. Ovaj postupak zna biti izuzetno naporan i moguće je doći do zaključka kako skup jednadžbi nije rješiv jer se postupak zasniva na samom tipu robotske ruke.

Prednosti analitičke metode uključuju:

- Izračunava sva rješenja inverzne kinematike (IK) i utvrđuje kada rješenje ne postoji.
- Nakon što se jednadžbe izvedu, rješenja se mogu vrlo brzo izračunati.
- Nema potrebe za definiranjem parametara rješenja ili početnih pretpostavki q_0.

Nedostaci analitičke metode uključuju:

- Često je teško ili zamorno izvesti jednadžbe.
- Moraju se neovisno izvoditi za robote s različitim kinematičkim strukturama.
- Primjenjivo je samo na neredundantne robote (broj stupnjeva slobode = broj dimenzija radnog prostora). [11]

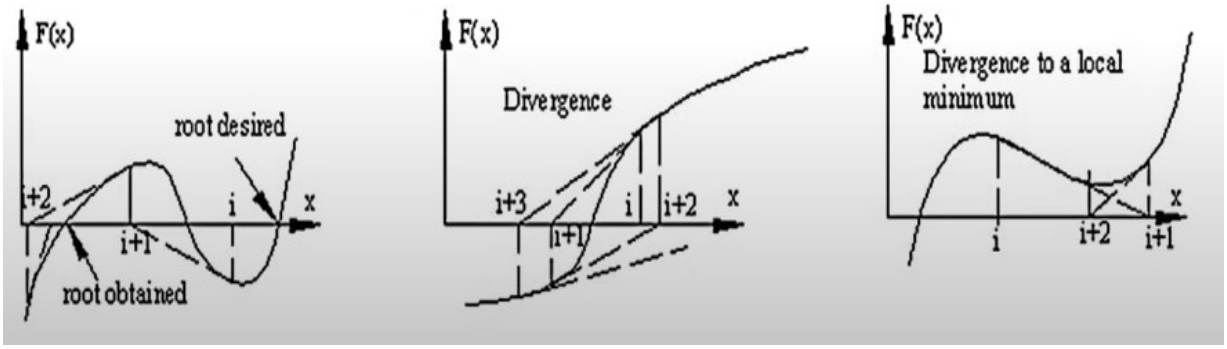
Numerički postupak

Kao što mu i samo ime govori za ovaj postupak rješavanja je potrebno računalo, pošto se radi o iterativnom postupku za razliku od prethodno spominjanog analitičkog. Postupak se zasniva na jed. (3.15), te poznavanju rješenja direktnog kinematičkog problema.
Prvo je potrebno pretpostaviti unutarnje koordinate (pogledati jed.(3.1)) , nakon početnih pretpostavljenih koordinata se izračunavaju vanjske koordinate robotske ruke preko jednadžbi dobivenih prilikom rješavanja direktne kinematike. Zatim se od vektora trenutnih vanjskih koordinata oduzima vektor izračunatih koordinata kako bi se dobila razlika između njih označena s ΔR. Upravo je ta razlika vektora vanjskih koordinata derivacija vektora **R**. Zatim kako bi dobili mali pomak unutarnjih koordinata odnosno Δq koristimo izraz (3.15) . Sada kada se dobije Δq, nova pretpostavka kutova jest:

$$q_n = q_{n-1} + \Delta q_n \qquad \textbf{(3.26)}$$

Ovaj postupak se odvija sve do uvjeta: $\Delta R < \varepsilon$, gdje je ε dopustiva greška ili ako se prijeđe određeni zadani broj iteracija, što govori o lošoj početnoj pretpostavci kutova. Ukoliko je početna pretpostavka bila loša potrebno je unesti novu i ponoviti postupak, ovo može biti poprilično zahtjevno na procesor računala i na vrijeme obavljanja radnje što je kod robotske ruke itekako bitno.

Na slici 3.20 su prikazane ovisnosti vektora ε **(**F[x] na slici) i vektora **q** (x na slici). Slika prikazuje razne mogućnosti problema do kojih može doći kod pogađanja početnih unutarnjih koordinata, što je ujedno i najveći nedostatak numeričke metode. Prvi graf pokazuje kako je moguće doći do rješenja, ali koje je krivo odnosno nije se tražilo. Drugi graf pokazuje divergenciju, odnosno iteracije će ići u beskonačno jer svaki sljedeći pokušaj će biti sve dalje od rješenja. Na kraju treći graf prikazuje da je moguće da rješenje ode u lokalni minimum i tu ostane te se ne pronađe rješenje. Također izuzet ovih grafova, ako graf ima ravnu karakteristiku potrebno je isprogramirati na koju stranu će ići sljedeća iteracija.



Slika 3.20 Grafovi ovisnosti vektora ε u zavisnosti od vektora q [14]

Jedna od glavnih prednosti numeričke metode jest to da ju je moguće primijeniti na redundantne robote (roboti koji imaju više SSG-a od fizičkih značajki pozicije, odnosno roboti s više od 6 SSG-a). Također još jedna važna prednost je što je primjenjiva na svim strukturama robota što nije slučaj kod analitičke metode.

Najveći problem je kao što je već ranije spomenuto, to što je izuzetno zahtjevna na računalo, zbog složenih matričnih operacija i konstantnog računanja inverza. Analitičko rješenje je po ovome pitanju u ogromnoj prednosti, zbog toga što se izuzetno brzo izračuna i nije zahtjevno na računalu. Još veći problem od spomenutoga je dobro pogađanje početnih unutarnjih koordinata robota. Kod loše odabranih početnih unutarnjih koordinata numeričko rješenje nije moguće naći.

3.5. Izrada konstrukcije

3.5.1. Izbor materijala

Na temelju istrage dostupnih edukacijskih robotskih ruku, lako je zaključiti kako je glavni izbor za velik dio konstrukcije pao na plastiku dostupnu za 3D print, te na aluminijske komponente. Pri izradi vlastite robotske ruke odluka je pala na čeličnu konstrukciju zbog toga što je čvršći i žilaviji od aluminija, te se može zavariti bez upotrebe TIG uređaja. Uz to se lakše obrađuje od aluminija primjenom ručnog i amaterskog alata. Za debljinu samoga čeličnog lima je odabrana debljina 0.8 mm, što je dovoljno tanko kako bi sama težina konstrukcije bila što manja, a opet dovoljno debelo za zavarivanje i snažnu konstrukciju. Također je korišten i čelik od 3 mm za određene dijelove gdje je krutost to zahtijevala. Kod izrade vlastite ruke moguće je samu konstrukciju izraditi i od obične 3D printane plastike, no takav materijal je s razlogom napravljen kako bi se mogao printati, što znači da nema najbolja mehanička svojstva te ona naglo slabe s vremenom i temperaturom.

3.5.2. Spojevi

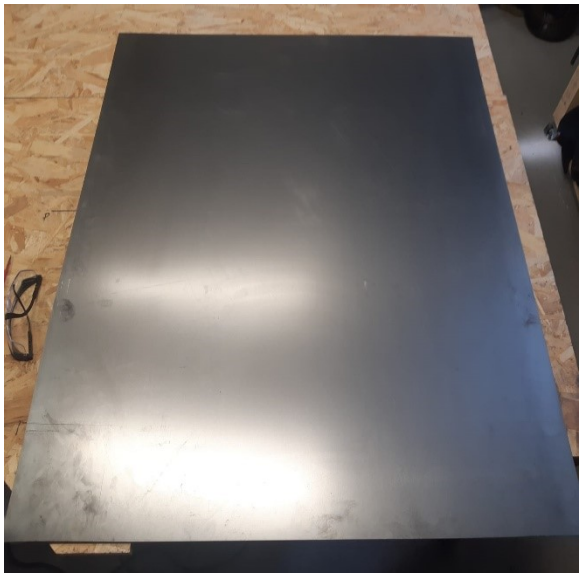
Nerastavljivi spojevi su ostvareni pomoću zavarivanja punjenom žicom. Samo zavarivanje je postupak za koji treba imati određeni uređaj te mnogo prakse, posebice kod zavarivanja tankog lima poput ovoga koji se koristi. Rastavljivi spojevi su ostvareni pomoću vijaka od M2 do M2,5.

3.5.3. Izrada postolja/kućišta

Kod izrade postolja, cilj je bio da bude što jednostavnije, te da ga je moguće pričvrstiti uza stol ili određeno radno mjesto pomoću stega, odnosno moguće ga je ušarafiti uz radnu površinu ukoliko je to potrebno.

Postupci koji su se koristili prilikom izrade su savijanje, rezanje, bušenje, brušenje i zavarivanje. Ovi postupci su se koristili i prilikom izrade ostalih dijelova.

U kućištu se nalaze dva otvora kroz koja se spaja napajanje i komunikacijski kabel s računalom za rad s robotskom rukom. Također unutar kućišta se nalaze posebno oblikovana kućišta za U2D2 konverter i SMPS2dynamixel napajanje. Glavno napajanje se nalazi izvan robota zbog svoje veličine. Kada bi napajanje stajalo unutar robota to bi zahtijevalo nepotrebno proširenje kućišta i dodatno kompliciralo njegovu izradu što svakako nije dobro pri izradi vlastite robotske ruke. Kućište je prikazano na slici 3.25. Na sljedećoj stani je prikazan čelični lim, te postupak zavarivanja i bušenja.



Slika 3.21 Čelični lim

Slika 3.22 Postupak bušenja

Slika 3.23 Priprema za zavarivanje

Slika 3.24 Postupak zavarivanja

Slika 3.25 Postolje/kućište robotske ruke

Kao dio kućišta pripada i poklopac koji se spaja s kućištem pomoću vijaka te omogućava montiranje servo motora (M1). U poklopcu je napravljen otvor koji omogućuje prolaz i pomicanje žice koja spaja motor M1 s motorom M2. U suprotnom da je samo napravljen otvor žica bi se povlačila i ne bi imalo slobodnu rotaciju, te bi postojala opasnost od pucanja ili izvlačenja žice iz konektora u servo motorima.

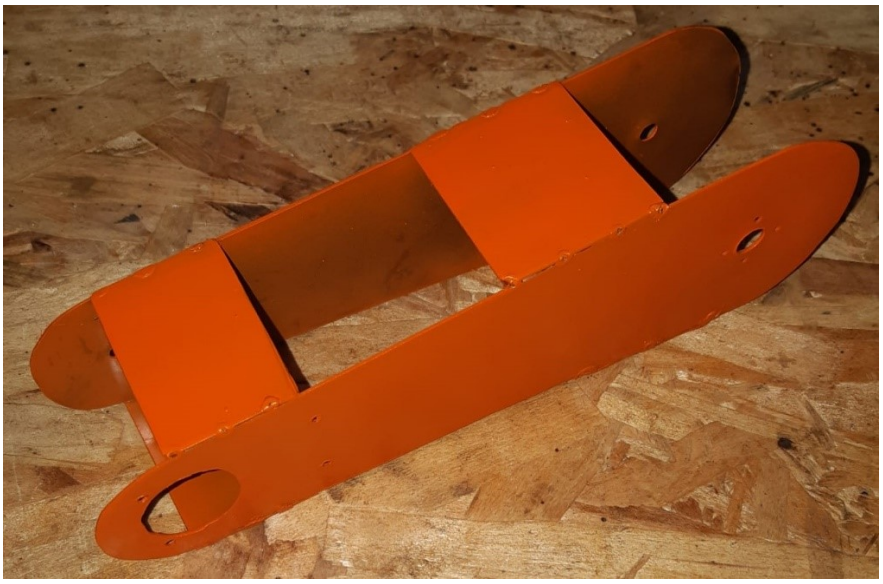
Slika 3.26 Poklopac za kućište

3.5.4. Izrada članaka ruke

Slika 3.27 Prvi članak robotske ruke

Dio prikazan na slici 3.27 služi kao prvi pokretni dio robotske ruke preko kojega se ostvaruje prvi stupanj slobode gibanja, te se montira na motor M1 pomoću 4 M2x10 vijka. Također omogućava montažu drugog motora, te klizni ležaj za rotaciju drugog člana robotske ruke. Imajući na umu kako je ova ruka edukativnog i prezentacijskog tipa , zbog jednostavnije izrade i montaže izbjegnut je kuglični ležaj koji se koristi kod profesionalnih robotskih ruku. Kao i vertikalni kuglični ležaj prvi član robotske ruke. Rupa na dijelu omogućuje prolaz žice za međusobno povezivanje motora. Cijeli dio je izrađen od 3 mm debeloga čelika zbog otpornosti na savijanje koje bi god lima od 0.8 mm bilo prisutno i ne bi bilo moguće izraditi čvrsti i kruti dio. Težina ovoga dijela ne utječe značajno na opterećenost motora pošto se rotacija vrši oko same osi motora pa je njegovo opterećenje minimalno.

Što se tiče ostala dva članka oni su izrađeni od 0.8 mm lima kako bi se što više smanjila njihova težina, iz razloga što ovdje težina utječe značajno na potrebni moment motora. Također je svaki član prema svome kraju sve uži kako bi se smanjila masa, a spojni dijelovi su postavljeni na početak i sredinu samih članaka kao i na kraju, ali znatno uži. Sve navedeno je vidljivo na slici 3.28, te na slici 3.29.



Slika 3.28 Drugi članak

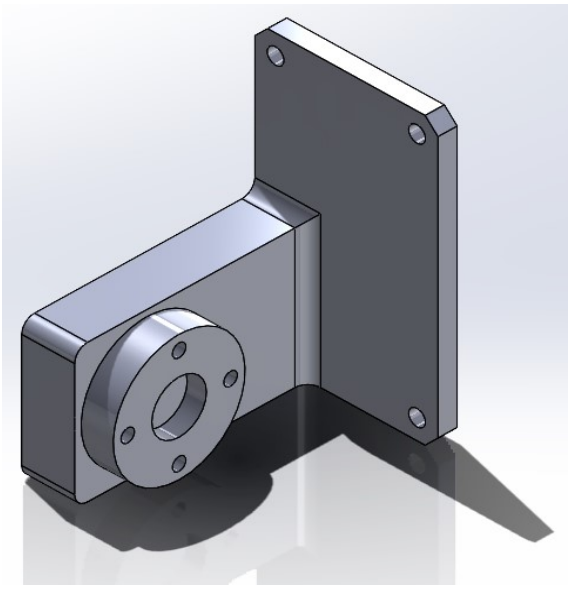
Slika 3.29 Treći članak

Svi dijelovi si bili obojani pomoću sprej boje koja sprječava koroziju čelika.

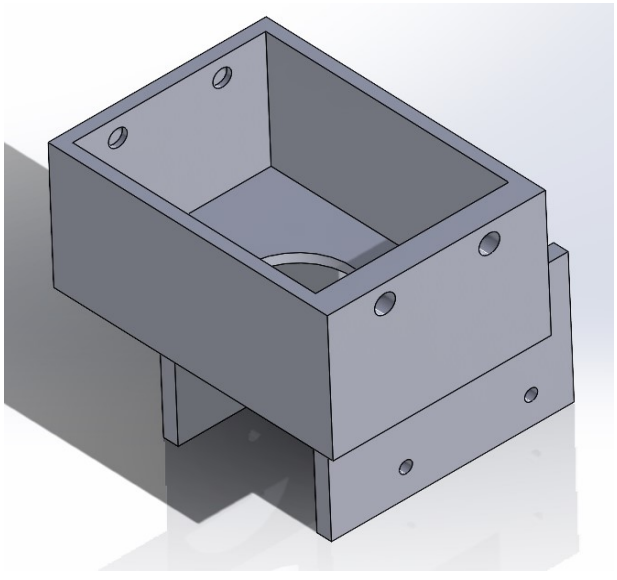
3.5.5. Izrada hvataljke

Za razliku od ostale konstrukcije hvataljka je zbog svoje kompleksnosti i same pozicije (bitno je koristiti što lakši materijal jer se nalazi na najvećemu kraku robotske ruke) izrađena pomoću 3D print-a. Svaki od dijelova hvataljke je bio konstruiran pomoću SOLIDWORKS programa. Nakon proučavanja literature [15] ,došlo se do zaključka da je PLA plastika najbolja za izradu hvataljke, pošto je ona dovoljno kruta za zupčani spoj, a i njena biorazgradivost joj daje veliku prednost što uz korištenje čelika kao materijala za ostatak konstrukcije ovu robotsku ruku čini 100% biorazgradivom.

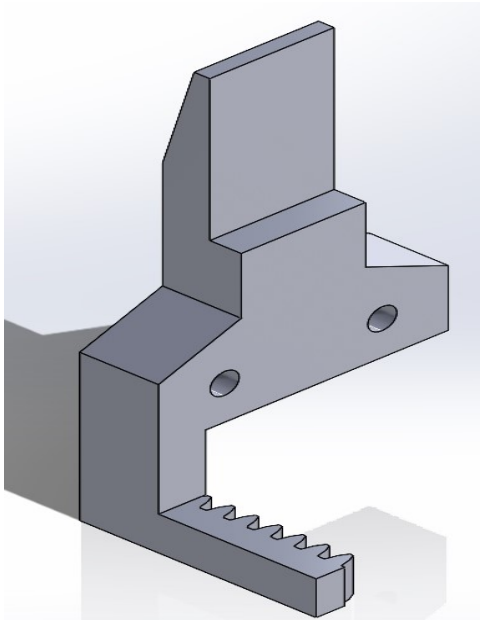
Sam mehanizam hvataljke je već napravljen odavno, te je osnova ideja uzeta razgledavanjem interneta, a zatim je bila konstruirana inženjerskim razmišljanjem i dostupnim dijelovima. U dijelu hvataljke prikazanim na slici 3.32 je ostavljen prostor, gdje je kasnije bila dvokomponentnim epoksi ljepilom zalijepljena guma kako bi se povećalo trenje hvataljke i mekši prihvat. Hvataljka se sastoji od šest dijelova, glavni dio je kućište u kojem se nalazi cijeli mehanizam i na koje se montira motor za ostvarenje gibanja. Zatim su tu dvije prihvatnice sa zubima, zupčanik, ruka za spoj s prethodnim člankom i vodilice izrađene od ispoliranih čavala. Sve navedeno je prikazano na sljedećim slikama. Oko svih pokretnih dijelova je dodana mast za podmazivanje ležaja kako bi se smanjilo trenje.



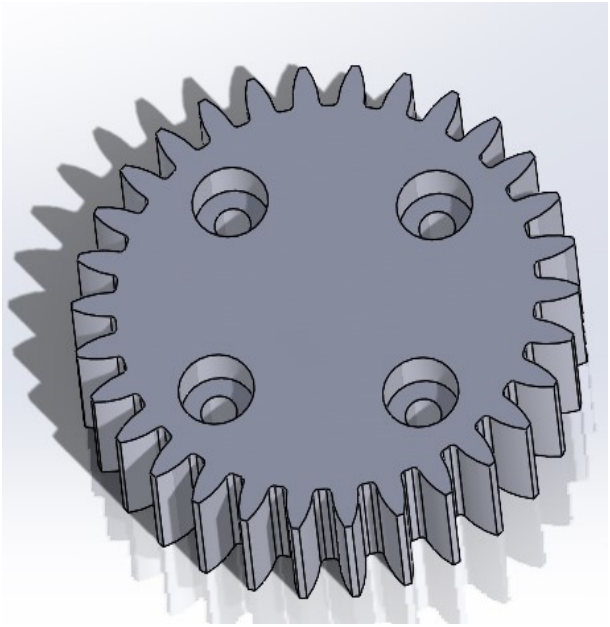
Slika 3.30 Ruka hvataljke



Slika 3.31 Kućište hvataljke



Slika 3.32 Prihvatnica



Slika 3.33 Zupčanik

Slika 3.34 Hvataljka robotske ruke

Naredne dvije slike prikazuju sklopljenu robotsku ruku spremnu za testiranje i podešavanja parametara motora kako bi kretnja robotske ruke bila sigurna i precizna.

Slika 3.35 Robotska ruka

Slika 3.36 Sklopljena robotska ruka

3.6. Postavke motora

Nakon izrade konstrukcije potrebno je podesiti PID parametre svakoga od motora, zbog toga što dodatni moment tereta na svakom od motora unosi određenu grešku u poziciji. Također potrebno je i doraditi brzinu kojom se motori smiju okretati zbog velikih inercijskih sila koje mogu utjecati na stabilnost konstrukcije i dodatno nepotrebno opterećenja na motore, te iz sigurnosnih razloga je bolje da je brzina manja i bez trzaja.

3.6.1. Postavljanje brzine vrtnje

Brzina vrtnje se može podešavati preko adresa motora dostupnih u Dynamixel Wizard 2 programu. Slika 3.37 prikazuje adresu pomoću koje je spomenuto moguće napraviti. Ovdje se radi o metodi pokušaja te se određuje brzina koja se čini najoptimalnija. Važno je napomenuti kako se krajnji članci motora mogu brže kretati pošto oni ne utječu značajno na negativne posljedice velikih brzina, zbog malih masa koje pokreću. Brzina M5 je postavljena na 20 , M4 na 30, M3 na 20, M2 na 10, te M1 na 15.

112	Profile Velocity	30	0x0000001E	6.87 [rev/min]

Slika 3.37 Adresa za profil brzine

3.6.2. Podešavanje P, I i D pojačanja za regulaciju pozicije motora

Na slici 3.38 su prikazane tri adrese u kojima se moraju mijenjati parametri kako bi regulacija pozicije bila točna. Rezultati su prikazani grafovima u Dynamixel Wizard-u , na kojima su prikazana zadana pozicija i trenutna pozicija svakoga od motora. U analizi nisu prikazani rezultati za hvataljku pošto se kod nje očekuje kako zadana i trenutna pozicija neće biti iste zbog samog predmeta koji se bude hvatao pomoću nje. Također kako je M1 (odnosno motor koji omogućava $q1$ zakret) jako slabo opterećen, tvorničke postavke će zadovoljavati.

80	Position D Gain	0	0x0000
82	Position I Gain	0	0x0000
84	Position P Gain	700	0x02BC

Slika 3.38 Adrese za D, I i P pojačanja

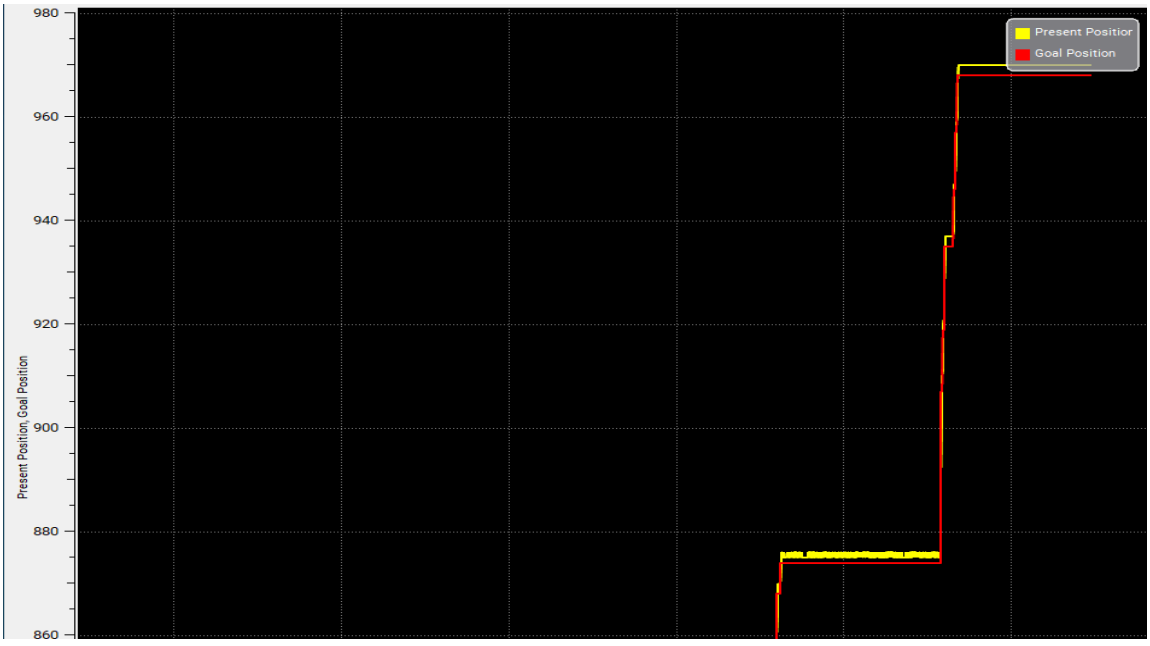
Tablica 3.2 Efekti samostalnog povećanja parametra

Parametar	Vrijeme porasta	Nadvišenje	Vrijeme smirivanja	Stacionarna greška	Stabilnost
K_p	Smanjuje	Povećava	Mala promjena	Smanjuje	Pogoršava
K_i	Smanjuje	Povećava	Povećava	Uklanja	Pogoršava
K_d	Mala promjena	Smanjuje	Smanjuje	U teoriji nema utjecaja	Poboljšava ako je Kp mal

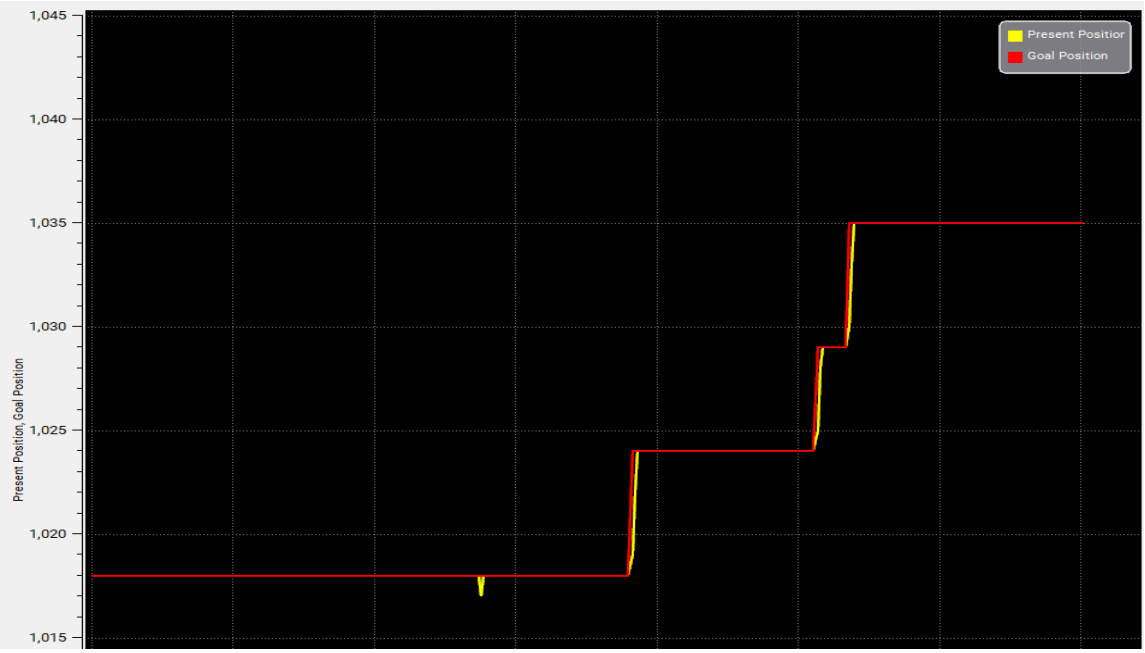
Podešavanje parametara motora 4

Žuta boja predstavlja trenutnu poziciju, a crvena zadanu poziciju motora. Iz priloženih slika se može vidjeti kako je bilo preskakanja referentne vrijednosti pri tvorničkim pojačanjima koja su K_d=0, K_i=0 i K_p=700. Imajući na umu tablicu 3.2 nakon određenog borja iteracija konačni parametri koji zadovoljavaju su: K_d=900, K_i =200, te K_p =700.

Za ostale motore će samo biti napisana konačni parametri i prikazani prije/nakon karakteristike. Također važno je provjeriti odzive i kod spuštanja reference kao i dizanja jer moment potpomaže kod spuštanja što zna znatno utjecati na odzive.



Slika 3.39 Karakteristika M4 -prije podešavanja

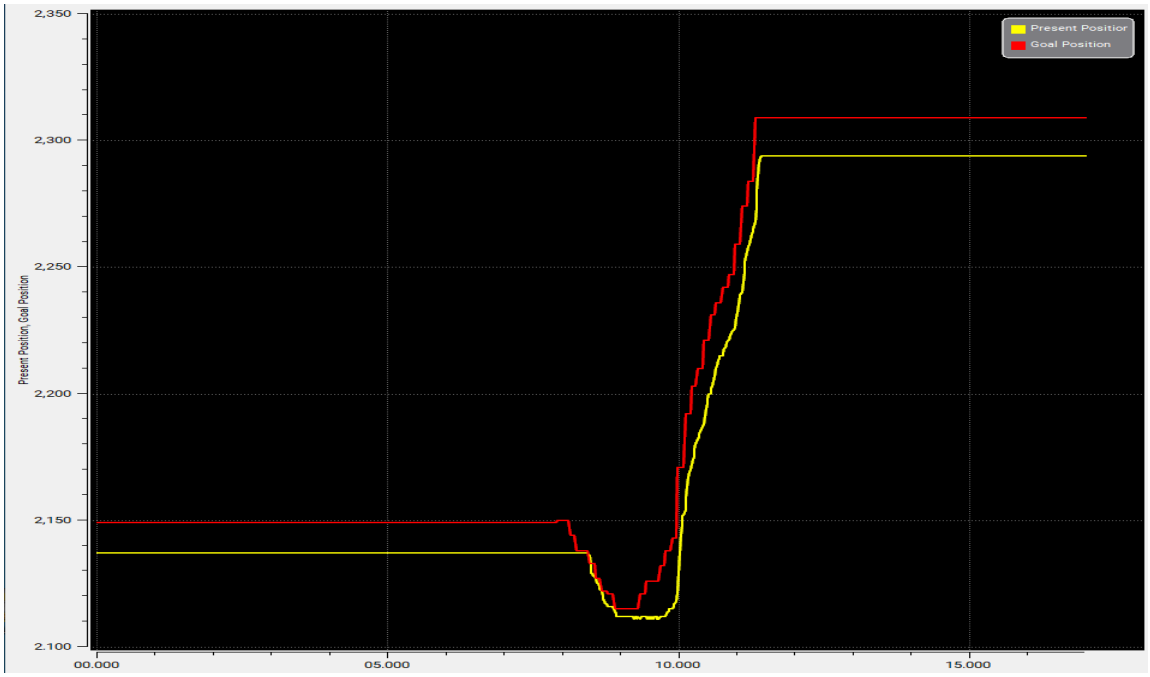


Slika 3.40 Karakteristika M4 -nakon podešavanja

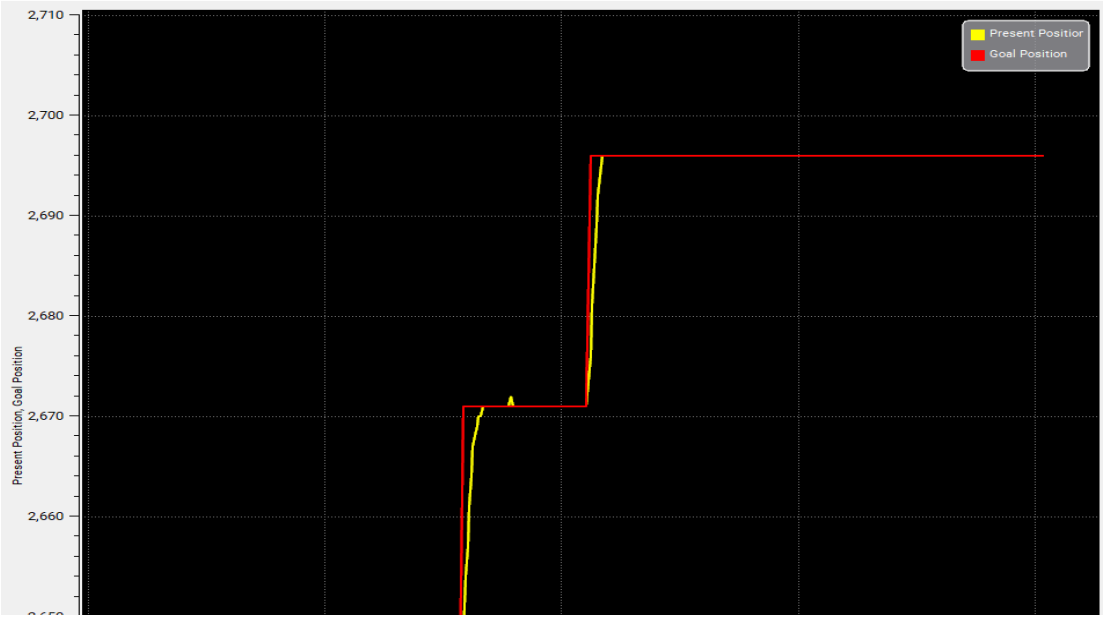
Parametri motora 3 i 2

Pojačanja za M3 su: K_d =1000, K_i =0, te K_p =600.

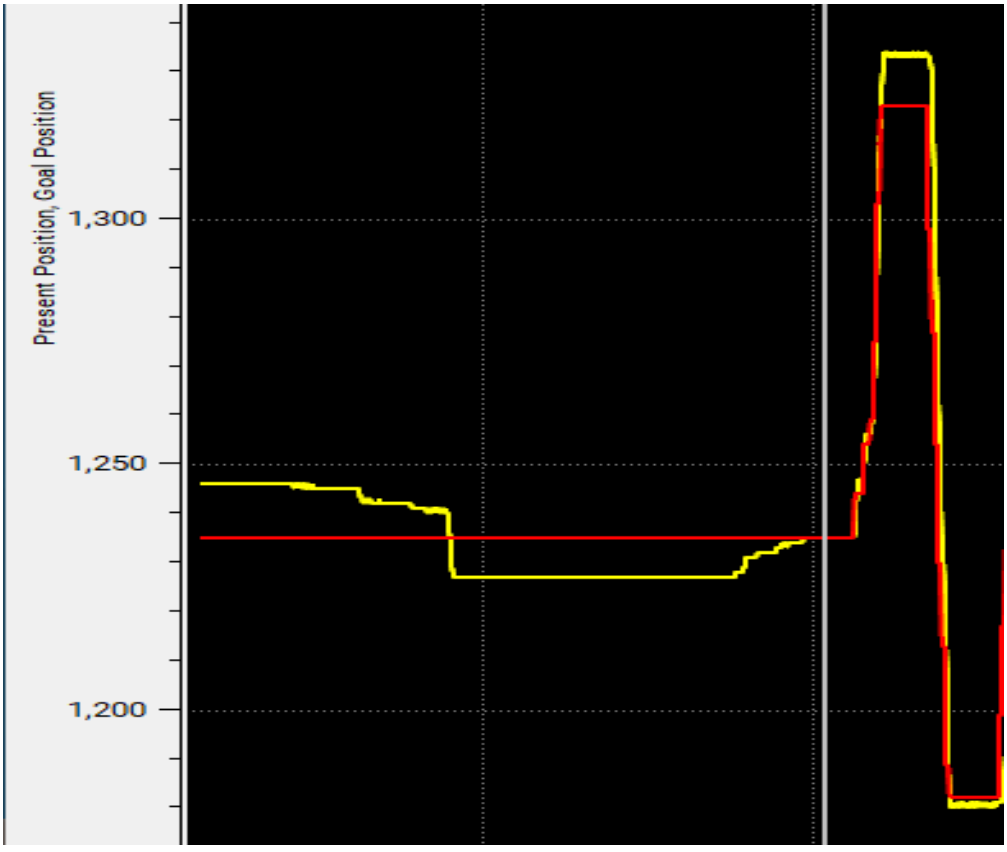
Pojačanja za M2 su: K_d =0, K_i =200, te K_p =700



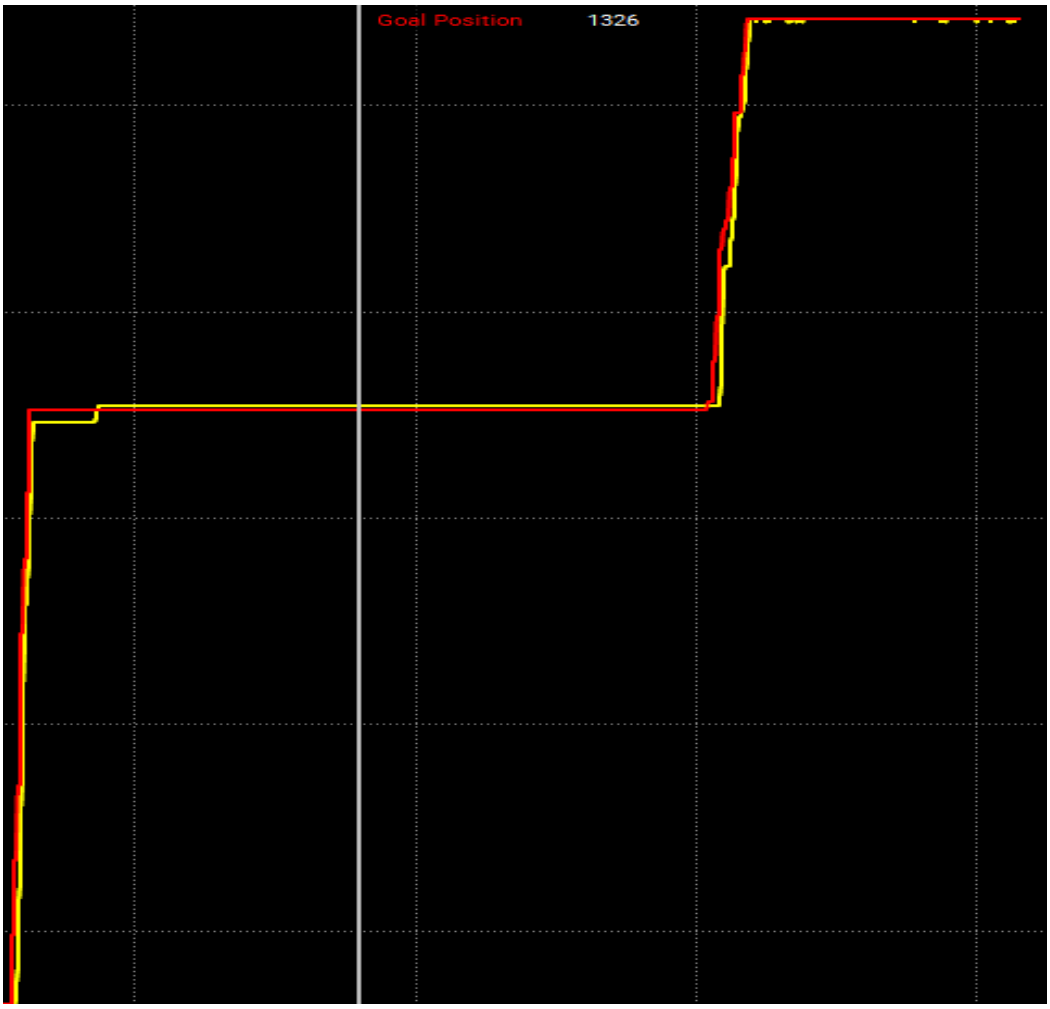
Slika 3.41 Karakteristika M3 -prije podešavanja



Slika 3.42 Karakteristika M3 -nakon podešavanja



Slika 3.43 Karakteristika M2 -prije podešavanja

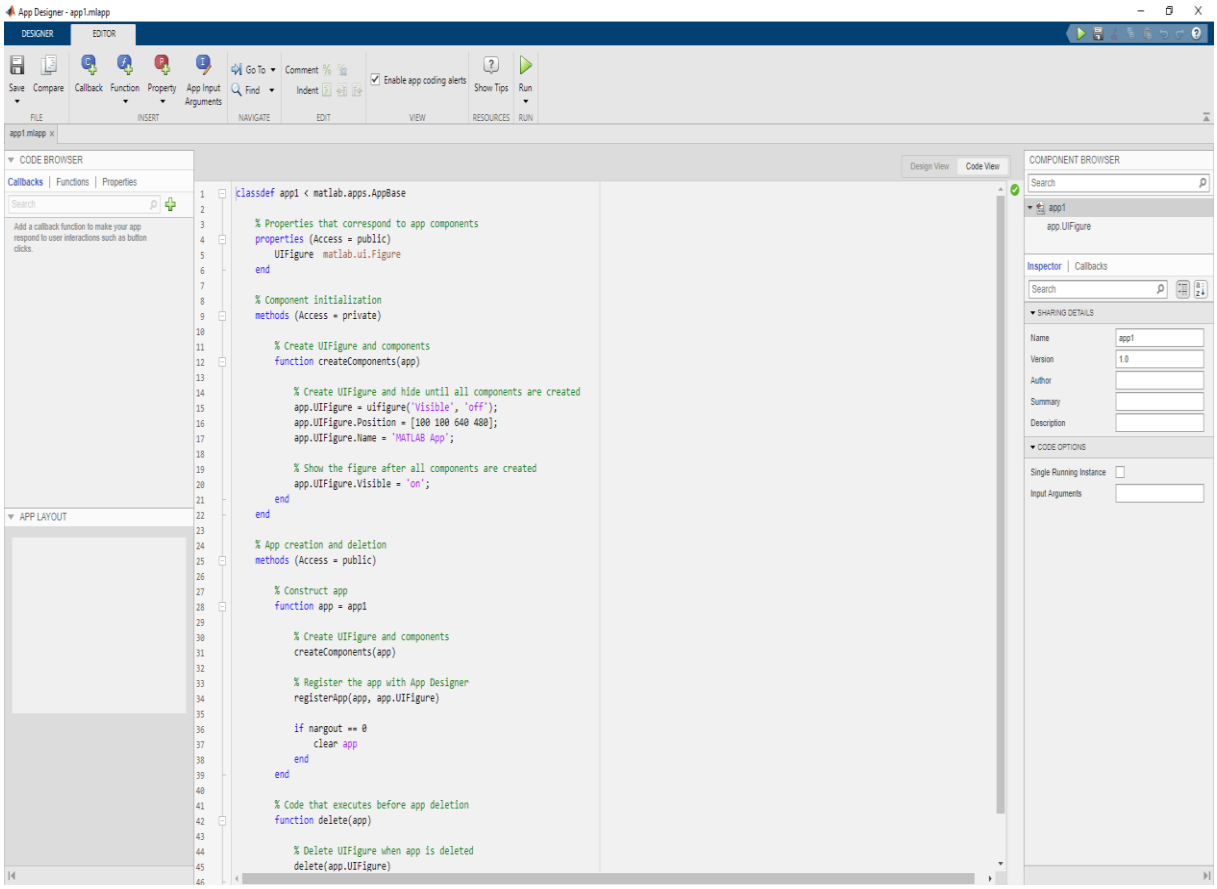


Slika 3.44 Karakteristika M2 -nakon podešavanja

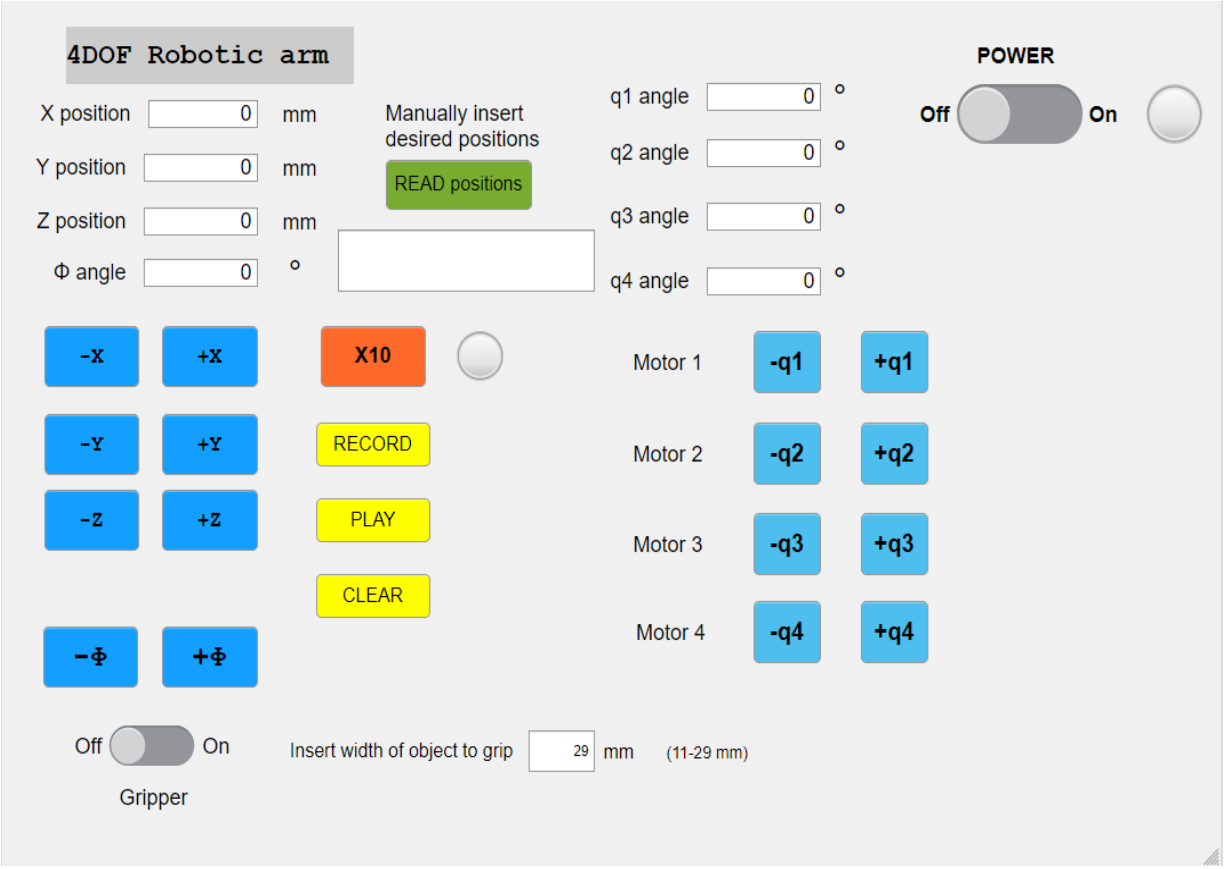
3.7. Izrada korisničkog sučelja (GUI-a)

Kod izbora korisničkog sučelja odluka je pala na MATLAB App Designer. Iako je upravljanje Dynamixel-ovim motorima znatno bolje odrađeno i napravljeno za programiranje u ROS-u, Arduinu i Pythonu. Sama izrada korisničkog sučelja je nešto jednostavnija pomoću MATLAB-a u donosu na spomenute programe.

App Designer je interaktivno razvojno okruženje za dizajniranje izgleda aplikacije i programiranje njenog ponašanja. Samo sučelje se izrađuje pomoću drag & drop metode , a potom se svaka ubačena komponenta posebno programira. Svaka komponenta služi kao input ili kao output informacija. Na input komponente se vežu callback-ovi, što su zapravo funkcije koje se pozovu kada se dotični input aktivira. Moguće je izraditi vlastite funkcije, a varijable se spremaju pod „properties“ te imaju sufiks app. . Kod za dodane komponente se sam generira te se samo dodaju naredbe koje će on izvršavati.



Slika 3.45 Korisničko sučelje App Designer-a



Slika 3.46 Korisničko sučelje robotske ruke

3.8. Izrada programa u MATLAB-u

Ovo poglavlje pokriva kod u programu koji se ne odnosi na korisničko sučelje, već na samu kinematiku i kretanje robota, te njegove mogućnosti. Ovdje će se samo ukratko proći kroz ideje kako bi robot trebao raditi i na tok izrade samoga osnovnog programa, sam kod sa svim natuknicama i komentarima se nalazi u prilogu rada. Dio koda izradi sam MATLAB te on neće biti u prilogu.

Prva i osnovna ideja je bila da se na temelju korisničkog sučelja učitavaju vrijednosti unutarnjih kod direktne odnosno vanjskih kod inverzne kinematike. Zatim bi se te vrijednosti čitale iz polja na kojima su prikazane i stavljale u ranije izvedenu kinematiku. Kada korisnik promijeni kut motoru preko sučelja aktivira se direktna kinematika kako bi se mogle ispisati koordinate izvršnog člana, odnosno ako korisnik unese koordinate određene odmah se na sučelje ispisuju kutovi motora koji su izračunati putem inverzne kinematike. Time se dobiva sukladnost kako bi na sučelju u svakome trenutku za prikazane unutarnje koordinate odgovarale njihove vanjske koordinate. Također su se prije probe u glavnome programu motori i njihove mogućnosti testirale u zasebnim testnim programima. Također bilo je potrebno postavljati određena ograničenja kod unosa podataka i kod samih izračunatih podataka pošto robot ima svoj radni prostor i nije u mogućnosti svaki put otići u zadanu poziciju. Što se ovog dijela tiče tu još postoji prostor za napredak. Također je stavljen dio koda u program da robotska ruka može pamtiti pozicije te se kasnije vraćati na iste i izvoditi razne radnje, također se i pamti stanje hvataljke koja se uključi/isključi u potrebnom položaju. Kako je ovo edukacijska ruka, ovaj programski dio ostavlja još mnogo prostora za napredak i dodavanje vlastitih mogućnosti ili zadataka koje robot može napraviti. Kako je samo korisničko sučelje izrađeno preko App Designer-a ono je izuzetno robusno i ostavlja vrlo malo prostora za grešku. Sami dio koda za inverznu kinematiku bi se još mogao dodatno poboljšati primjenom ne samo jedne, već sve tri spomenute metode za rješavanje inverznog problema , te ih integrirati skupa kako ne bi dolazilo do grešaka prilikom rješavanja. Kako svaka od metoda ima svoje negativne i pozitivne strane, izrada programa koristeći ovu metodu bi zahtijevala mnogo više programiranja i detaljno poznavanje svake od njih. Kako dostupni motori nisu imali mogućnost rada u „Current Control“ ili „Current based Position Control“ režimu, sama hvataljka nije mogla biti isprogramirana preko navedenih režima koji omogućuju puno lakšu i bolju izvedbu hvatanja predmeta. Iz tog razloga je potrebno unijeti širinu objekta koji se planira uhvatiti pomoću hvataljke , kako bi program odredio do koje pozicije mora doći hvataljka da bi prihvatila predmet s 40-60% opterećenja motora, što iz testova odgovara optimalnoj sili pritiska za većinu objekata.

4. PROCJENA TROŠKOVA IZRADE ROBOTSKE RUKE

Kod procjene troškova je važno uzeti u obzir dostupnost alata koje osoba što izrađuje vlastitu robotsku ruku ima. Ova procjena je bazirana na vlastitoj izradi, te je vrlo vjerojatno značajno manja nego što bi to bilo da je rađena bez svih dostupnih alata i uređaja. Kod same izrade veliki dio sredstava odlazi na motore kao što se može vidjeti u tablici 4.1. Ostali materijal koji nisu spomenuti, kao i izrada dijelova su bili tako rečeno besplatni, ako se ne računa trošenje alata i upotrjebljena električna energija. Važno je naglasiti kako je korišten i SOLIDWORKS te MATLAB, što su programi koji se moraju platiti ako osoba nema dostupnu besplatnu licencu. Ako bi se ruka htjela izraditi pomoću CNC laserskoga rezanja te profesionalnog zavarivanja cijena bi bila veća za nekih 20%.

Ako se cijena usporedni s komercijalno dostupnim rukama obrađenima u drugom poglavlju, izrada vlastite ruke ispada najisplativija.

Tablica 4.1 Cijene korištenih komponenti i materijala

Komponenta	Broj komada	Pojedinačna cijena [€]	Ukupna cijena [€]
Dynamixel XC430-W150-T	1	108	108
Dynamixel XC430-W240-T	4	119	476
U2D2	1	28,84	28,84
SMPS2Dynamixel	1	6,3	6,3
MEAN-WELL-LRS-75-12	1	16,61	16,61
Robot Cable 3P-XL	6	3	18
Čelični lim 0.8mm (0,2 m^2)	1	2	2
Čelični lim 3mm (0,1 m^2)	1	5	5
Boja	1	4,5	4
3D print hvataljke	1	3	3
Vijci	/	/	2
UKUPNA CIJENA RUKE	670 €		

5. ZAKLJUČAK

U radu je napravljena detaljna analiza tržišta s već dostupnim edukacijskim rukama, te je potom napravljena objektivna usporedba izabranih ruku, odnosno svakoj je dodijeljena konačna ocjena i kao objektivno najbolja ruka na tržištu je Wlkata Mirobot. Nakon analize tržišta je rad krenuo opisivati tijek izrade vlastite robotske ruke. Kod izrade vlastite ruke je bitno da ona ne zahtijeva skupe dijelove i da je sama izrada i struktura programa jednostavna za objasniti studentima ili učenicima srednjih škola. Izrada je krenula od idejnog koncepta ruke te odabira o broju stupnjeva slobode gibanja koje će posjedovati. Dalje je uslijedio odabir motora koji će se koristiti za izradu robotske ruke, te je on pao na Dynamixel XC430-W240-T i Dynamixel XC430-W150-T servo motore. Razlog odabira ovih motora je taj što su bili dostupni, a uz to su idejno osmišljeni i napravljeni upravo za primjenu u području mehatronike i robotike. Rad je zatim objasnio postavljanje osnovnih parametara motora koji su potrebni za kasniju upotrebu u programu. Zatim se detaljno objasnio i riješio direktni kao i inverzni kinematski problem. Nakon postavljenih temelja krenulo se u izradu konstrukcije robotske ruke kao i njene hvataljke, hvataljka je u potpunosti konstruirana u SOLIDWORKS programu. Rad u SOLIDWORKS-u kao i u MATLAB-u je bio poprilično jednostavan zbog dobro stečenog znanja tijekom studiranja. Izrada konstrukcije od tankog čeličnog lima se pokazala poprilično zahtjevnijom nego što se na početku činila. Postoji puno nepredviđenih komplikacija koji nastanu ili stvari na koje se ne misli odmah na početku , te one značajno oduže samo vrijeme izrade. Kada je robotska ruka izrađena i sklopljena , postavljena su potrebna pojačanja regulatora položaja za ispravan rad. Na kraju je ukratko objašnjena izrada korisničkoga sučelja i programa. Program kao takav ostavlja dosta prostora za napredak, poput planiranja putanje kojom se robot kreće, te mogućnost biranja vrste kretanja robotske ruke (najbrži, najekonomičniji, pravocrtni itd.).

Gledajući razne izvore na ovu temu može se naići na mnogo pojedinaca koji su opisali kako je to napraviti vlastitu ruku, te kako i po nekoliko mjeseci ili godina već rade na tome.

Unatoč tome što je tema rada uspješno odrađena, važno je imati na umu kako ovdje još ostaje dosta prostora za napredak i poboljšanja koja bi jednu edukacijsku ruku postavile na poziciju s koje bi mogla ozbiljno konkurirati onima dostupnima na tržištu.

LITERATURA

[1] J. Wallén, The history of the industrial robot., Linköping University: Electronic Press, 2008.

[2] M. L. W. K. G. &. B. R. Merdan, Robotics in education: Research and practices for robotics in STEM education, Springer, 2016.

[3] »mybotshop,« [Mrežno]. Available: https://www.mybotshop.de/DOBOT-Magician-Basic-Robot-arm. [Pokušaj pristupa 12 8 2024].

[4] »mybotshop,« [Mrežno]. Available: https://www.mybotshop.de/Interbotix-PincherX-100_1. [Pokušaj pristupa 12 8 2024].

[5] »wlkata,« Wlkata, [Mrežno]. Available: https://www.wlkata.com/. [Pokušaj pristupa 12 8 2024].

[6] »nyrio,« Niryo, [Mrežno]. Available: https://niryo.com/product/6-axis-robotic-arm/. [Pokušaj pristupa 12 8 2024].

[7] »shop.ozobot.com,« [Mrežno]. Available: https://shop.ozobot.com/products/ora-ozobot-robotic-arm?srsltid=AfmBOoqUiyQV0rdq-HLy-w5K-G7xsYBofKwMN2U6NBe_AikPQ_4KxR2. [Pokušaj pristupa 13 8 2024].

[8] J. J.Craig, Introduction to Robotics Mechanics and Control, Pearson Education, Inc., 2005.

[9] »robotis.com,« Robotis, [Mrežno]. Available: https://emanual.robotis.com/docs/en/dxl/x/xc430-w240/. [Pokušaj pristupa 13 8 2024].

[10] »robotis.com,« Robotis, [Mrežno]. Available: https://emanual.robotis.com/docs/en/dxl/x/xc430-w150/. [Pokušaj pristupa 13 8 2024].

[11] M. W. H. S. &. V. M. Spong, Robot modeling and control., John Wiley & Sons., 2020.

[12] J. Angeles, Fundamentals of robotic mechanical systems: theory, methods, and algorithms., New York: Springer New York, 2003.

[13] K. E. &. S. Y. Clothier, »A geometric approach for robotic arm kinematics with hardware design, electrical design, and implementation.,« *Journal of Robotics,* svez. 2010, p. 10, 2010.

[14] »Youtube,« [Mrežno]. Available: https://www.youtube.com/watch?v=_fVgZASO0Cw&t=1149s&ab_channel=EngineeringEducatorAcademy. [Pokušaj pristupa 15 8 2024].

[15] M. G. H. S. Alok Kumar Trivedi, »PLA based biocomposites for sustainable products,« *Sciencedirect,* svez. 6, br. 4, p. 14, 2023.

PRILOZI

I. MATLAB kod robotske ruke

I. MATLAB kod robotske ruke

```
%definiranje svojstva odnosno varijabli koje će se koristiti
properties (Access = private)
    %naziv biblioteke
    lib_name = '';
    % Postavljanje adresa za kontrolu
    ADDR_PRO_TORQUE_ENABLE = 64;
    ADDR_PRO_GOAL_POSITION = 116;
    ADDR_PRO_PRESENT_POSITION = 132;
    ADDR_PRO_PRESENT_LOAD = 126;
    % Duljina podatka u bajtovima
    LEN_PRO_GOAL_POSITION = 4;
    LEN_PRO_PRESENT_POSITION = 4;
    LEN_PRO_PRESENT_LOAD = 2;
    % Verzija protokola
    PROTOCOL_VERSION = 2.0;
    % Inicijaliziranje motora, frekvencije osvježavanja,
    serijskog
    % porta itd.
    DXL1_ID = 1; % Dynamixel#1 ID: 1 XC430-W240-T
    DXL2_ID = 2; % Dynamixel#2 ID: 2 XC430-W240-T
    DXL3_ID = 3; % Dynamixel#1 ID: 3 XC430-W240-T
    DXL4_ID = 4; % Dynamixel#2 ID: 4 XC430-W240-T
    DXL5_ID = 5; % Dynamixel#2 ID: 5 gripper XC430-W150-T
    BAUDRATE = 3000000;
    DEVICENAME = 'COM4';
    TORQUE_ENABLE = 1; % Value for enabling the torque
    TORQUE_DISABLE = 0; % Value for disabling the torque
    groupwrite_num = 0;
    groupread_num = 0;
    port_num = 0;

    %GOAL pozicije (0~4095)
    dxl_1_goal_position = 0;
    dxl_2_goal_position = 0;
    dxl_3_goal_position = 0;
    dxl_4_goal_position = 0;
    dxl_5_goal_position = 0;

    % Varijable za INVERSE_KINEMATICS
    %input
    x_inv = 0;
    y_inv = 0;
    z_inv = 0;
    fi_inv = 0;

    %output
    q1_inv = 0;
    q2_inv = 0;
```

```
q3_inv = 0;
q4_inv = 0;
q1_inv_str = 'q1_inv';
q2_inv_str = 'q2_inv';
q3_inv_str = 'q3_inv';
q4_inv_str = 'q4_inv';
q1_inv_deg = 0;
q2_inv_deg = 0;
q3_inv_deg = 0;
q4_inv_deg = 0;

% Varijable za FORWARD_KINEMATICS
%output
x=0;
y=0;
z=0;
fi=0;

%input
q1=90;
q2=90;
q3=-90;
q4=0;
q1_str='q1';
q2_str='q2';
q3_str='q3';
q4_str='q4';
q1_rad=0;
q2_rad=0;
q3_rad=0;
q4_rad=0;

%Udaljenosti između članaka [mm]
L1=125;
L2=150;
L3=57.5;
L4=116;

%pomoćne varijable
times10 = 0;
rec = zeros(100,5); %100 mogućih pozicija
rec_counter = 0;
gripper_state = 0;
out_of_reach_state = 0;
play_state = 0
end
```

```
methods (Access = private)

% funkcija za izračun inverzne kinematike
function Inverse_Kinematic(app)
     %uzimamo iz polja poziciju EE
     if(app.play_state==0) %dodano za play
     app.x_inv = app.XpositionEditField.Value;
     app.y_inv = app.YpositionEditField.Value;
     app.z_inv = app.ZpositionEditField.Value;
     app.fi_inv = app.angleEditField.Value;
     end
     %dodano za play
     app.XpositionEditField.Value=app.x_inv;
     app.YpositionEditField.Value=app.y_inv;
     app.ZpositionEditField.Value=app.z_inv;
     app.angleEditField.Value=app.fi_inv;
     %računjanje inverzne kinematike
     %q1_inv
     %-----------------------------------------------------
     app.q1_inv = atan2(app.y_inv,app.x_inv);
     if(app.q1_inv<1e-3 && app.q1_inv>0)
     app.q1_inv=0;
     %provjera granica
     elseif (app.q1_inv>pi || app.q1_inv<0)
     app.EditField_error.Value = 'OUT OF REACH';
     app.out_of_reach_state=1;
     end
     %-----------------------------------------------------
     %pomoćne varijable za računanje q3
     %q3_inv
     A = (app.x_inv-app.L4*cos(app.q1_inv)*cos(app.fi_inv));
     B = (app.y_inv-app.L4*sin(app.q1_inv)*cos(app.fi_inv));
     C = (app.z_inv-app.L1-app.L4*sin(app.fi_inv));
     D = ((A^2+B^2+C^2-(app.L2)^2-
     (app.L3)^2)/(2*app.L2*app.L3));
     if(D>1 || D<-1)                %dodao
          app.EditField_error.Value = 'OUT OF REACH';
          app.out_of_reach_state=1;
     end
     if(abs(D)<=1)
          app.q3_inv = atan2(-sqrt(1-D^2),D);
     end
     if(abs(app.q3_inv)<1e-3)
     app.q3_inv=0;
     %provjera granica
     elseif (app.q3_inv>pi/2 || app.q3_inv<-pi/2)
     app.EditField_error.Value = 'OUT OF REACH';
     app.out_of_reach_state=1;
     end
     %pomoćne varijable za računanje q2
     a = app.L3*sin(app.q3_inv);
```

```
b = app.L2+app.L3*cos(app.q3_inv);
c = app.z_inv-app.L1-app.L4*sin(app.fi_inv);
r = sqrt(a^2+b^2);
if((r^2-c^2)<0)
app.EditField_error.Value = 'OUT OF REACH';
app.out_of_reach_state=1;
elseif((app.q3_inv<pi/2 && app.q3_inv>-pi/2) &&
(app.q1_inv<pi && app.q1_inv>0))
app.out_of_reach_state=0;
app.q2_inv = atan2(c,sqrt(r^2-c^2))-atan2(a,b); %q2_inv
     if(app.q2_inv<1e-3)
     app.q2_inv=0;
     %provjera granica
     elseif(app.q2_inv>pi || app.q2_inv<0)
     app.EditField_error.Value = 'OUT OF REACH';
     app.out_of_reach_state=1;
     end
app.q4_inv = app.fi_inv-app.q2_inv-app.q3_inv; %q4_inv
end
%ako su svi kutovi kako treba ovaj dio koda se izvršava
if(app.out_of_reach_state==0)
app.EditField_error.Value = 'IN REACH';
%prebacivanje iz rad u deg
app.q1_inv_deg = rad_to_degree(app,app.q1_inv);
app.q2_inv_deg = rad_to_degree(app,app.q2_inv);
app.q3_inv_deg = rad_to_degree(app,app.q3_inv);
app.q4_inv_deg = rad_to_degree(app,app.q4_inv);
%upisivanje kutova u GUI
app.q1angleEditField.Value = round(app.q1_inv_deg,1);
app.q2angleEditField.Value = round(app.q2_inv_deg,1);
app.q3angleEditField.Value = round(app.q3_inv_deg,1);
app.q4angleEditField.Value = round(app.q4_inv_deg,1);
%slanje kuteva servo motorima
app.dxl_1_goal_position =
convert_angle(app,app.q1_inv_str);
app.dxl_2_goal_position =
convert_angle(app,app.q2_inv_str);
app.dxl_3_goal_position =
convert_angle(app,app.q3_inv_str);
app.dxl_4_goal_position =
convert_angle(app,app.q4_inv_str);
% Add parameter storage for Dynamixel#1 goal position
groupBulkWriteAddParam(app.groupwrite_num, app.DXL1_ID,
app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
typecast(int32(app.dxl_1_goal_position), 'uint32'),
app.LEN_PRO_GOAL_POSITION);
% Add parameter storage for Dynamixel#2 goal position
groupBulkWriteAddParam(app.groupwrite_num, app.DXL2_ID,
app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
typecast(int32(app.dxl_2_goal_position), 'uint32'),
app.LEN_PRO_GOAL_POSITION);
```

```
        % Add parameter storage for Dynamixel#3 goal position
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL3_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_3_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Add parameter storage for Dynamixel#4 goal position
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL4_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_4_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Bulkwrite goal positions
        groupBulkWriteTxPacket(app.groupwrite_num);
        % Clear bulkwrite parameter storage
        groupBulkWriteClearParam(app.groupwrite_num);
        end
end
% dio koda za računanje direktne kinematike
function Forward_Kinematic(app)
        %provjera stanja play_state
        %čitanje kutova iz GUI-a
        app.q1 = app.q1angleEditField.Value;
        app.q2 = app.q2angleEditField.Value;
        app.q3 = app.q3angleEditField.Value;
        app.q4 = app.q4angleEditField.Value;
        %pretvaranje u radiane
        app.q1_rad = degree_to_rad(app,app.q1);
        app.q2_rad = degree_to_rad(app,app.q2);
        app.q3_rad = degree_to_rad(app,app.q3);
        app.q4_rad = degree_to_rad(app,app.q4);
        %računanje koordinata izvršnog člana
        app.x =
        cos(app.q1_rad)*(app.L3*cos(app.q2_rad+app.q3_rad)+app.L2*
        cos(app.q2_rad)+app.L4*cos(app.q2_rad+app.q3_rad+app.q4_ra
        d));
        app.y =
        sin(app.q1_rad)*(app.L3*cos(app.q2_rad+app.q3_rad)+app.L2*
        cos(app.q2_rad)+app.L4*cos(app.q2_rad+app.q3_rad+app.q4_ra
        d));
        app.z =
        app.L1+app.L3*sin(app.q2_rad+app.q3_rad)+app.L2*sin(app.q2
        _rad)+app.L4*sin(app.q2_rad+app.q3_rad+app.q4_rad);
        app.fi = app.q2_rad+app.q3_rad+app.q4_rad;
        if(abs(app.x)<0.1)
        app.x=0;
        end
        if(abs(app.y)<0.1)
        app.y=0;
        end
        if(abs(app.z)<0.1)
        app.z=0;
        end
```

```
        app.XpositionEditField.Value = round(app.x,1);
        app.YpositionEditField.Value = round(app.y,1);
        app.ZpositionEditField.Value = round(app.z,1);
        app.angleEditField.Value =
        round(rad_to_degree(app,app.fi),2);
        % pokretanje motora 1-4
        app.dxl_1_goal_position = convert_angle(app,app.q1_str);
        app.dxl_2_goal_position = convert_angle(app,app.q2_str);
        app.dxl_3_goal_position = convert_angle(app,app.q3_str);
        app.dxl_4_goal_position = convert_angle(app,app.q4_str);
        % Add parameter storage for Dynamixel#1 goal position
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL1_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_1_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Add parameter storage for Dynamixel#2 goal position
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL2_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_2_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Add parameter storage for Dynamixel#3 goal position
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL3_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_3_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Add parameter storage for Dynamixel#4 goal position
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL4_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_4_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Bulkwrite goal positions
        groupBulkWriteTxPacket(app.groupwrite_num);
        % Clear bulkwrite parameter storage
        groupBulkWriteClearParam(app.groupwrite_num);
end
%funkcija za promjenu kuta iz stupnjeva u brojeve za servo
motore
function goal_pos = convert_angle(app,kut)
        if (strcmp(kut,'q1')==1)
                goal_pos = round(11.375*app.q1);
                if(abs(goal_pos)<1)
                        goal_pos=1;
                end
        elseif(strcmp(kut,'q1_inv')==1)
                goal_pos = round(11.375*app.q1_inv_deg);
                if(abs(goal_pos)<1)
                        goal_pos=1;
                end
        elseif (strcmp(kut,'q2')==1)
                goal_pos = round(2048-11.375*app.q2);
                if(abs(goal_pos)<1)
```

```
                goal_pos=1;
            end
        elseif(strcmp(kut,'q2_inv')==1)
            goal_pos = round(2048-11.375*app.q2_inv_deg);
            if(abs(goal_pos)<1)
                goal_pos=1;
            end

        elseif (strcmp(kut,'q3')==1)
            goal_pos = round(1024-11.375*app.q3);
            if(abs(goal_pos)<1)
                goal_pos=1;
            end
        elseif(strcmp(kut,'q3_inv')==1)
            goal_pos = round(1024-11.375*app.q3_inv_deg);
            if(abs(goal_pos)<1)
                goal_pos=1;
            end
        elseif(strcmp(kut,'q4')==1)
            goal_pos = round(1024+11.375*app.q4);
            if(abs(goal_pos)<1)
                goal_pos=1;
            end

        elseif(strcmp(kut,'q4_inv'))
            goal_pos = round(1024+11.375*app.q4_inv_deg);
            if(abs(goal_pos)<1)
                goal_pos=1;
                end

        end


end

%pretvorba deg u rad
function rad = degree_to_rad(~,degree)
      rad=degree*(pi/180);

end

%pretvorba rad u deg
function deg = rad_to_degree(~,radian)
       deg=radian*(180/pi);

end

%funkcija koja računa vrijednost za gripper na temelju unesene
%širine objekta
function  gripper_close(app)
```

```
        app.dxl_5_goal_position
        =round(26.34*app.GripperValueEditField.Value + 1130);
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL5_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_5_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Bulkwrite goal positions
        groupBulkWriteTxPacket(app.groupwrite_num);

        % Clear bulkwrite parameter storage
        groupBulkWriteClearParam(app.groupwrite_num);

end

%funkcija koja otvara gripper u potpunosti
function gripper_open(app)
        app.dxl_5_goal_position = 1902;
        groupBulkWriteAddParam(app.groupwrite_num, app.DXL5_ID,
        app.ADDR_PRO_GOAL_POSITION, app.LEN_PRO_GOAL_POSITION,
        typecast(int32(app.dxl_5_goal_position), 'uint32'),
        app.LEN_PRO_GOAL_POSITION);
        % Bulkwrite goal positions
        groupBulkWriteTxPacket(app.groupwrite_num);

        % Clear bulkwrite parameter storage
        groupBulkWriteClearParam(app.groupwrite_num);

end

end %kraj metoda

% Code that executes after component creation
function startupFcn(app)
        %provjera operativnog sistema
        if strcmp(computer, 'PCWIN')
                app.lib_name = 'dxl_x86_c';
        elseif strcmp(computer, 'PCWIN64')
                app.lib_name = 'dxl_x64_c';
        elseif strcmp(computer, 'GLNX86')
                app.lib_name = 'libdxl_x86_c';
        elseif strcmp(computer, 'GLNXA64')
                app.lib_name = 'libdxl_x64_c';
        elseif strcmp(computer, 'MACI64')
                app.lib_name = 'libdxl_mac_c';
        end
         %učitavanje biblioteke za rad s motorima
        loadlibrary(app.lib_name, 'dynamixel_sdk.h','addheader',
        'port_handler.h', 'addheader', 'packet_handler.h',
        'addheader', 'group_bulk_read.h', 'addheader',
        'group_bulk_write.h');
```

```
    %sljedeći dio koda je na engl. jer je intuitivnije za
    %razumijeti

    % Get methods and members of PortHandlerLinux or
    PortHandlerWindows
    app.port_num = portHandler(app.DEVICENAME);

    % Initialize PacketHandler Structs
    packetHandler();

    % Initialize groupBulkWrite Struct
    app.groupwrite_num = groupBulkWrite(app.port_num,
    app.PROTOCOL_VERSION);

    % Initialize Groupbulkread Structs
    app.groupread_num = groupBulkRead(app.port_num,
    app.PROTOCOL_VERSION);

    % Open port
    openPort(app.port_num)


    % Set port baudrate
    setBaudRate(app.port_num, app.BAUDRATE)
```

%dio koda s tipkama

```
% Value changed function: Switch
function SwitchValueChanged(app, event)
    value_power = app.Switch.Value;
    if strcmp(value_power,'On')
    %POWER gumb
    app.Lamp_POWER.Color = 'g';
    % Enable Dynamixel#1 Torque
    write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
    app.DXL1_ID, app.ADDR_PRO_TORQUE_ENABLE,
    app.TORQUE_ENABLE);

    % Enable Dynamixel#2 Torque
    write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
    app.DXL2_ID, app.ADDR_PRO_TORQUE_ENABLE,
    app.TORQUE_ENABLE);
    % Enable Dynamixel#3 Torque
    write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
    app.DXL3_ID, app.ADDR_PRO_TORQUE_ENABLE,
    app.TORQUE_ENABLE);

    % Enable Dynamixel#4 Torque
```

```
write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
app.DXL4_ID, app.ADDR_PRO_TORQUE_ENABLE,
app.TORQUE_ENABLE);
% Enable Dynamixel#5 Torque
write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
app.DXL5_ID, app.ADDR_PRO_TORQUE_ENABLE,
app.TORQUE_ENABLE);
app.q1angleEditField.Value = app.q1;
app.q2angleEditField.Value = app.q2;
app.q3angleEditField.Value = app.q3;
app.q4angleEditField.Value = app.q4;
%postavljanje parametara motora 1
write4ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL1_
ID,112,20)
%postavljanje parametara motora 2
write4ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL2_
ID,112,10)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL2_
ID,84,700)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL2_
ID,82,200)
%postavljanje parametara motora 3
write4ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL3_
ID,112,20)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL3_
ID,84,600)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL3_
ID,80,1000)
%postavljanje parametara motora 4
write4ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL4_
ID,112,30)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL4_
ID,84,700)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL4_
ID,82,200)
write2ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL4_
ID,80,900)
%postavljanje parametara motora 5
write4ByteTxRx(app.port_num,app.PROTOCOL_VERSION,app.DXL5_
ID,112,20)

%pokretanje direktne kinematike nakon uključivanja
Forward_Kinematic(app);
else
app.Lamp_POWER.Color = [0.90,0.90,0.90];
% Disable Dynamixel#1 Torque
write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
app.DXL1_ID, app.ADDR_PRO_TORQUE_ENABLE,
app.TORQUE_DISABLE);
% Disable Dynamixel#2 Torque
```

```
        write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
        app.DXL2_ID, app.ADDR_PRO_TORQUE_ENABLE,
        app.TORQUE_DISABLE);
        % Disable Dynamixel#3 Torque
        write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
        app.DXL3_ID, app.ADDR_PRO_TORQUE_ENABLE,
        app.TORQUE_DISABLE);
        % Disable Dynamixel#4 Torque
        write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
        app.DXL4_ID, app.ADDR_PRO_TORQUE_ENABLE,
        app.TORQUE_DISABLE);
        % Disable Dynamixel#5 Torque
        write1ByteTxRx(app.port_num, app.PROTOCOL_VERSION,
        app.DXL5_ID, app.ADDR_PRO_TORQUE_ENABLE,
        app.TORQUE_DISABLE);

        end
end

% Button pushed function: XButton_plus

function XButton_plusPushed(app, event)
        if(app.times10 == 0)
                app.XpositionEditField.Value =
                app.XpositionEditField.Value + 1;
        else
                app.XpositionEditField.Value =
                app.XpositionEditField.Value + 10;
        end
        Inverse_Kinematic(app);
end

% Button pushed function: XButton_minus
function XButton_minusPushed(app, event)
        if(app.times10 == 0)
                app.XpositionEditField.Value =
                app.XpositionEditField.Value - 1;
        else
                app.XpositionEditField.Value =
                app.XpositionEditField.Value - 10;
        end
        Inverse_Kinematic(app);
end

% Button pushed function: YButton_minus
function YButton_minusPushed(app, event)
        if(app.times10 == 0)
        app.YpositionEditField.Value =
        app.YpositionEditField.Value - 1;
        else
```

```
        app.YpositionEditField.Value =
        app.YpositionEditField.Value - 10;
        end
        Inverse_Kinematic(app);
end

% Button pushed function: YButton_plus
function YButton_plusPushed(app, event)
        if(app.times10 == 0)
        app.YpositionEditField.Value =
        app.YpositionEditField.Value + 1;
        else
        app.YpositionEditField.Value =
        app.YpositionEditField.Value + 10;
        end
        Inverse_Kinematic(app);
end

% Button pushed function: ZButton_minus
function ZButton_minusPushed(app, event)
        if(app.times10 == 0)
        app.ZpositionEditField.Value =
        app.ZpositionEditField.Value - 1;
        else
        app.ZpositionEditField.Value =
        app.ZpositionEditField.Value - 10;
        end
        Inverse_Kinematic(app);
end

% Button pushed function: ZButton_plus
function ZButton_plusPushed(app, event)
        if(app.times10 == 0)
        app.ZpositionEditField.Value =
        app.ZpositionEditField.Value + 1;
        else
        app.ZpositionEditField.Value =
        app.ZpositionEditField.Value + 10;
        end
        Inverse_Kinematic(app);
end

% Button pushed function: Button_fi_minus
function Button_fi_minusPushed(app, event)
        if(app.times10 == 0)
        app.angleEditField.Value = app.angleEditField.Value - 1;
        else
        app.angleEditField.Value = app.angleEditField.Value - 10;
        end
        Inverse_Kinematic(app);
end
```

```
% Button pushed function: Button_fi_plus
function Button_fi_plusPushed(app, event)
     if(app.times10 == 0)
     app.angleEditField.Value = app.angleEditField.Value + 1;
     else
     app.angleEditField.Value = app.angleEditField.Value + 10;
     end
     Inverse_Kinematic(app);
end

% Button pushed function: q1Button_minus
function q1Button_minusPushed(app, event)
     if(app.times10 == 0)
     app.q1angleEditField.Value = app.q1angleEditField.Value -
     1;
     else
     app.q1angleEditField.Value = app.q1angleEditField.Value -
     10;
     end
     Forward_Kinematic(app);
end

% Button pushed function: q1Button_plus
function q1Button_plusPushed(app, event)
     if(app.times10 == 0)
     app.q1angleEditField.Value = app.q1angleEditField.Value +
     1;
     else
     app.q1angleEditField.Value = app.q1angleEditField.Value +
     10;
     end
     Forward_Kinematic(app);
end

% Button pushed function: q2Button_minus
function q2Button_minusPushed(app, event)
     if(app.times10 == 0)
     app.q2angleEditField.Value = app.q2angleEditField.Value -
     1;
     else
     app.q2angleEditField.Value = app.q2angleEditField.Value -
     10;
     end
     Forward_Kinematic(app);
end

% Button pushed function: q2Button_plus
function q2Button_plusPushed(app, event)
     if(app.times10 == 0)
```

```
        app.q2angleEditField.Value = app.q2angleEditField.Value +
        1;
        else
        app.q2angleEditField.Value = app.q2angleEditField.Value +
        10;
        end
        Forward_Kinematic(app);
end

% Button pushed function: q3Button_minus
function q3Button_minusPushed(app, event)
        if(app.times10 == 0)
        app.q3angleEditField.Value = app.q3angleEditField.Value -
        1;
        else
        app.q3angleEditField.Value = app.q3angleEditField.Value -
        10;
        end
        Forward_Kinematic(app);
end

% Button pushed function: q3Button_plus
function q3Button_plusPushed(app, event)
        if(app.times10 == 0)
        app.q3angleEditField.Value = app.q3angleEditField.Value +
        1;
        else
        app.q3angleEditField.Value = app.q3angleEditField.Value +
        10;
        end
        Forward_Kinematic(app);
end

% Button pushed function: q4Button_minus
function q4Button_minusPushed(app, event)
        if(app.times10 == 0)
        app.q4angleEditField.Value = app.q4angleEditField.Value -
        1;
        else
        app.q4angleEditField.Value = app.q4angleEditField.Value -
        10;
        end
        Forward_Kinematic(app);
end

% Button pushed function: q4Button_plus
function q4Button_plusPushed(app, event)
        if(app.times10 == 0)
        app.q4angleEditField.Value = app.q4angleEditField.Value +
        1;
        else
```

```
     app.q4angleEditField.Value = app.q4angleEditField.Value +
     10;
     end
     Forward_Kinematic(app);
end

% Button pushed function: READpositionsButton
function READpositionsButtonPushed(app, event)
     Inverse_Kinematic(app);
end

% Button pushed function: RECORDButton
function RECORDButtonPushed(app, event)
%kada se pritisne gumb RECORD zapamte se vrijednosti kutova s
%GUI-a
     app.rec_counter = app.rec_counter +1; %brojač pozicija


     app.rec(app.rec_counter,1) =
     app.XpositionEditField.Value;

     app.rec(app.rec_counter,2) =
     app.YpositionEditField.Value;

     app.rec(app.rec_counter,3) =
     app.ZpositionEditField.Value;

     app.rec(app.rec_counter,4) =
     app.angleEditField.Value;

     app.rec(app.rec_counter,5) = app.gripper_state;



end

% Button pushed function: CLEARButton
function CLEARButtonPushed(app, event)
     app.rec = zeros(100,5);   %postavlja sve vrijednosti u
nula odnosno briše
     app.rec_counter = 0;
end

% Value changed function: GripperSwitch
function GripperSwitchValueChanged(app, event)
     status = app.GripperSwitch.Value;
     if strcmp(status,'On')
          app.gripper_state=1;
          gripper_close(app);
     else
          gripper_open(app);
```

```
            app.gripper_state=0;
        end

end

% Button pushed function: PLAYButton
function PLAYButtonPushed(app, event)
        app.play_state=1;
        for m=1:1:app.rec_counter-1
            app.x_inv = app.rec(m,1);
            app.y_inv = app.rec(m,2);
            app.z_inv = app.rec(m,3);
            app.fi_inv = app.rec(m,4);
            g_state = app.rec(m,5);
            %pozove da se vrati u prvu točku, kako god
            Inverse_Kinematic(app);

            pause(2)
            if(g_state==1)
                gripper_close(app);
            else
                gripper_open(app);
            end

            %očitanje prethodne pozicije za računanje jed. pravca
            u prostoru
            x_inv_poc=app.XpositionEditField.Value;
            y_inv_poc=app.YpositionEditField.Value;
            z_inv_poc=app.ZpositionEditField.Value;

            %računanje jednadžbe pravca i određivanje lambde
            K=[x_inv_poc;y_inv_poc;z_inv_poc];  %prethodna točka

            app.x_inv = app.rec(m+1,1);
            app.y_inv = app.rec(m+1,2);
            app.z_inv = app.rec(m+1,3);
            app.fi_inv = app.rec(m+1,4);
            g_state = app.rec(m+1,5);

            L=[app.x_inv;app.y_inv;app.z_inv];

            dist=sqrt((app.x_inv-x_inv_poc)^2 + (app.y_inv-
            y_inv_poc)^2 + (app.z_inv-z_inv_poc)^2);
            %udaljenosti se diskretizira kako bi putanja bio
            „pravac"
            if(dist<100)
                 v=0.1;
            else
                 v=0.05;
            end
            for lambda=0:v:1
```

```
                P=K+lambda*(L-K);

                app.x_inv=P(1,1);
                app.y_inv=P(2,1);
                app.z_inv=P(3,1);
                Inverse_Kinematic(app);
            end
            pause(2)
            if(g_state==1)
               gripper_close(app);
            else
               gripper_open(app);
            end
        end

        app.play_state=0;

end
```