SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Luka Fućek, Antun Vukičević, Josip Vukičević

Razvoj i izrada šesteronogog hodača kao razvojne platforme

29. travnja 2015.

Ovaj rad izrađen je u Laboratoriju za Robotiku i Inteligentne Sustave Upravljanja, na Zavodu za Automatiku i Računalno Inženjerstvo pod vodstvom prof. dr. sc. Stjepana Bogdana i predan je za natječaj za dodjelu Rektorove nagrade u akademskoj godini 2014/2015.

Sadržaj

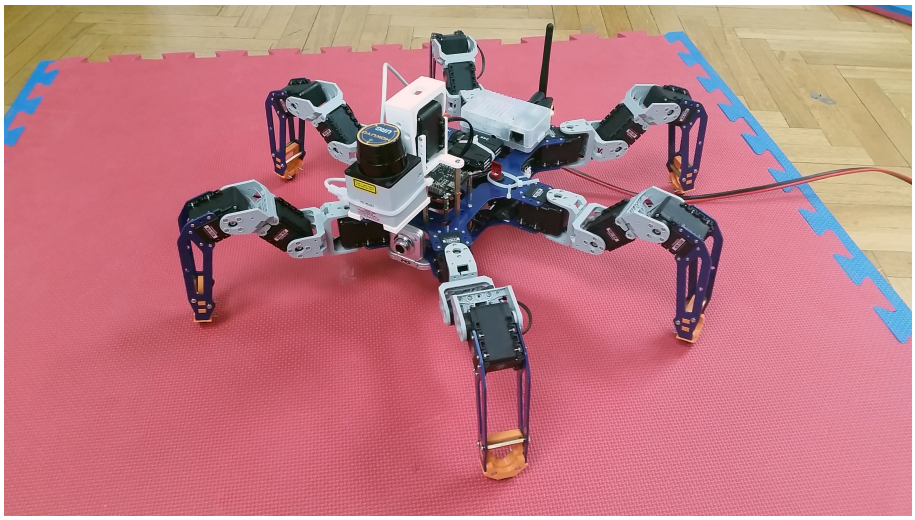
Popis slika

Popis tablica

1. Uvod

Iako ne tako nova grana robotike, roboti hodači predstavljaju granu koja trenutno doživljava veliki rast u razvoju. Lokomocija ovakvih robota je inspirirana prirodom što akademsku zajednicu svakim danom privlači sve više dok sama atraktivnost ovakvih robota ni hobiste diljem svijeta ne ostavlja nezainteresiranim. Velikom rastu u razvoju dodatno doprinosi sve mnogobrojnija i jeftinija proizvodnja elektromehaničkih komponenti potrebnih za realizaciju ovakvih robota. Najčešći hodači su bipedi, quadrupedi te hexapodi koji redom hodaju uz pomoć dvije, četiri te šest nogu. Dok je za kretanje bipeda i quadrupeda potrebna dinamička kontrola za hodanje, hexapodi svojim brojem nogu mogu osigurati da u svakom trenutku hoda na podlozi budu prizemljene barem tri noge što osigurava statičku stabilnost. Zbog redundantnosti hexapod robota moguće su i kompenzacije u slučaju otkazivanja jedne ili više nogu. Tada do izražaja dolaze sljedovi koračanja nogu ili *gaitovi*. Nadalje, ukoliko nisu sve noge potrebne za stabilan stav robota, one koje se ne koriste mogu se koristiti za manipulaciju teretom ili za dohvat novih gazišta. Iz tog su razloga hexapod hodači privlačni za istraživanje hoda po neravnim terenima, nedostupnim uobičajenim mobilnim robotika koji za kretanje koriste kotače.

Za razliku od mobilnih robota na kotačima, upravljanje robota hodača predstavlja veći problem nego što se na prvi pogled čini. Živim bićima je kretanje udova intuitivno te ne zahtjeva aktivno razmišljanje o zakretima zglobova što nije slučaj pri implementaciji sličnih kretanja kod robota. Iako je realizacija vrlo kompleksna, u konfiguraciji hexapod robota leži iznimno veliki potencijal za savladavanje i najteže dostupnih terena. Robot mora biti svjestan okoline kako bi se u njoj mogao snaći no to više ne znači samo prepoznati zidove i prepreke već se očekuje potpuno poznavanje okolnog terena po kojemu robot treba hodati. Zato se koriste senzori poput laserskih radara pomoću kojih se može rekonstruirati kompletan okolni prostor u obliku trodimenzionalne mape. Tek uz poznatu mapu prostora moguće je kvalitetno planirati hod robota po neravnim terenima. Ukoliko se uz mapu prostora koriste i senzori sile koju svaka noga osjeća pri dodiru s podlogom te mjerni inercijalni sustav za povratnu vezu orijentacije robota, moguće je osmisliti autonomni sustav koji bi se snalazio i u najtežim terenima bez ljudske interakcije.



2. Opći i specifični ciljevi rada

Inženjeri ili studenti željni rada na hexapod robotskim hodačima većinom grade vlastite platforme zbog zatvorenog načinom izrade dostupnih hexapoda te njihove neorijentiranosti razvoju. Opći cilj ovog rada je riješiti taj problem tj. razviti i izgraditi šesteronogog hodača kao razvojnu platformu otvorenog i modularnog tipa. Njezina otvorenost omogućit će dostupnost korištenih algoritama i metoda korisniku, dok će njezina modularnost omogućiti dodavanje novih ili izmjenu postojećih. Time se ostvaruje potpuna personalizacija platforme ovisno o potrebi, bilo da korisniku neki dijelovi nisu potrebni ili ju želi nadograditi novim.

Kako bi se postigao navedeni opći cilj, potrebno ga je razložiti na više specifičnih. Tako ovaj rad ima sljedeće specifične ciljeve:

- Fizička konstrukcija - Kako bi se cijela robotska platforma držala na mjestu te se osiguralo čvrsto i pouzdano pozicioniranje komponenti na njoj potrebno je napraviti fizičku konstrukciju. Detaljna razrada fizičke konstrukcije opisan je u poglavlju 3.

- Algoritam hodanja - Zakretima zglobova fizičke konstrukcije je potrebno upravljati na način da se razvojna platforma u konačnici kreće u željenom smjeru. Algoritam hodanja se brine o upravo tim zakretima. Planira korake te ih međusobno sinkronizira raznim metodama. Detaljnija razrada algoritma hodanja se nalazi u poglavlju 5.1.

- Simultana lokalizacija i mapiranje - Mapiranje okoline platforme kao i lociranje unutar napravljene mape važni su za snalaženje platforme u prostoru. Jedan takav algoritam upogonjen je na razvojnoj platformi. Detaljna razrada simultane lokalizacije i mapiranja opisana je u poglavlju 5.2.

- Inercijalni mjerni sustav - Ugradnja inercijalog mjernog sustava pruža platformi informacije o prostornoj orijentaciji platforme kao i njenoj kutnoj brzini te linearnoj akceleraciji. Uz pomoć tih informacija moguć je razvoj naprednijih algoritama upravljanja platformom. Detaljna razrada inercijalnog mjernog sustava opisana je u poglavlju 5.3.

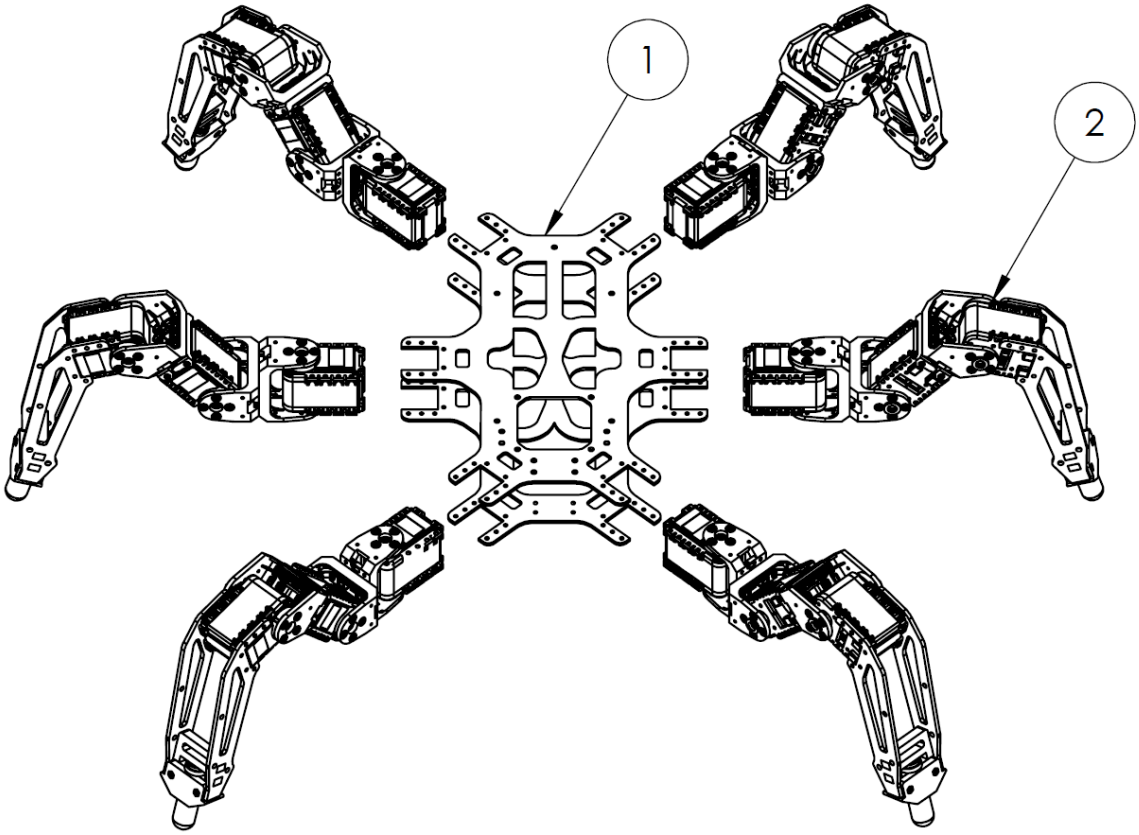
- Senzori sile - Mjerenje sile koju svaka noga osjeća pri dodiru s podlogom radi se pomoću senzora sile. Oni otvaraju mogućnost razvoja algoritama za hod po neravnim terenima i slično. Detaljna razrada Senzora sile opisana je u poglavlju 5.4.

- Robotski simulator - Proces testiranja na stvarnoj opremi često je vremenski i financijski zahtjevan jer se svaka greška skupo plaća. Riješenje je preseliti se u virtualni svijet robotske simulacije. Detaljna razrada realizacije robotske platforme opisan je u poglavlju 5.5.

3. Tehnička izvedba

Tehnička izvedba predstavlja koraka realizacije novih produkata u kojem se točno određuje fizički odnos i odabir komponenti. U njezinoj izvedbi se pazi na ulogu, specifikacije i potrebe svake komponente jer pogreške mogu dovesti do smanjene funkcionalnosti ili neželjenog ponašanja.

Realizacija tehničke izvedbe razvojne platforme temelji se na SolidWorks[1] 3D CAD okruženju gdje njegova CAD[2] orijentacija predstavlja najveću prednost nad ostalim 3D okruženjima. Cjeline osnovne fizičke konstrukcija razvojne platforme prikazane su Slikom 1. Osnovnom konstrukcijom smatraju se modularno napravljene cjeline na koje se dalje grade nove komponente.



Slika 1: Prikaz osnovnih konstrukcijskih cjelina razvojne platforme

Tablica 1: Popis osnovnih konstrukcijskih cjelina razvojne platforme

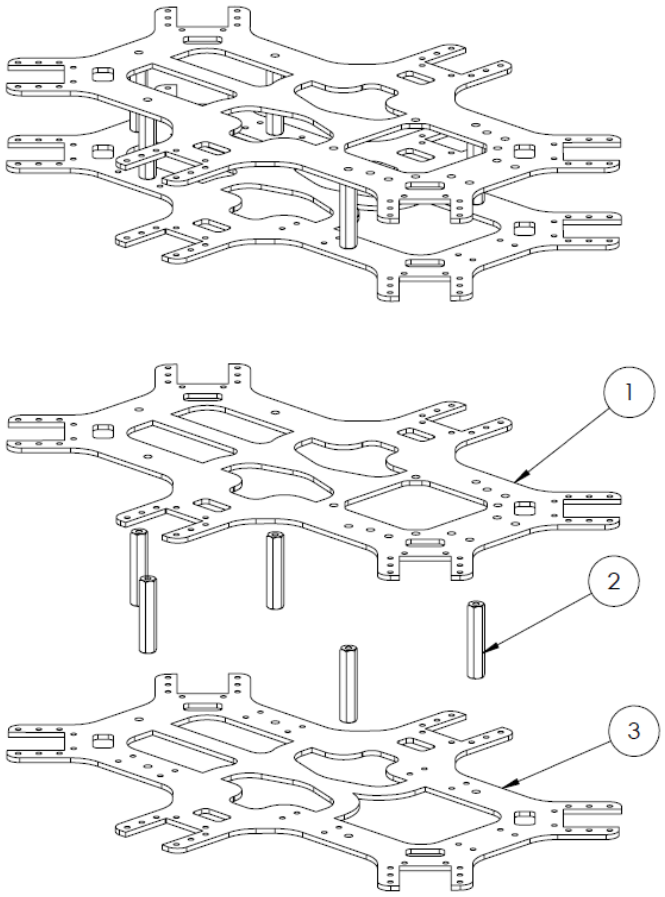
Redna oznaka na Slici 1	Naziv cjeline
1	Tijelo
2	Noga

[1] Detaljnije informacije o SolidWorksu nalaze se u poglavlju 4.6.5

[2] Computer-aided design (CAD) je proces dizajna u kojem računalo pomaže pri njegovom stvaranju, modifikaciji, analizi i optimizaciji - link: `http://en.wikipedia.org/wiki/Computer-aided_design`

3.1. Tijelo

Za Tijelo razvojne platforme se može reći da predstavlja glavni dio konstrukcije. Ono svojim izgledom i pozicijom igra ulogu trupa koji drži cijelu platformu na mjestu te osigurava čvrsto i pravilno pozicioniranje ostalih cjelina. Izgrađeno je od dvije odvojene aluminijske ploče debljine dva milimetra, koje su se izradile CNC[3] rezanjem. Aluminij svojom čvrstinom i malom masom odlično igra uloge na platformi. Visinu tijela određuju električni aktuatori te do pet odstojnika duljine 30 milimetara koji se stavljaju po potrebi. Masa Tijela minimizirana je micanjem nepotrebnog materijala dok je dizajn modularno orijentiran, tako da postoji više načina i pozicija na koji se ostale cjeline i komponente mogu pričvrstiti. Modularnost uvelike pridonosi namještanju centra mase platforme u njezino geometrijskom središte. Tehnička izvedba Tijela prikazana je Slikom 2.



Slika 2: Prikaz tehničke izvedbe Tijela

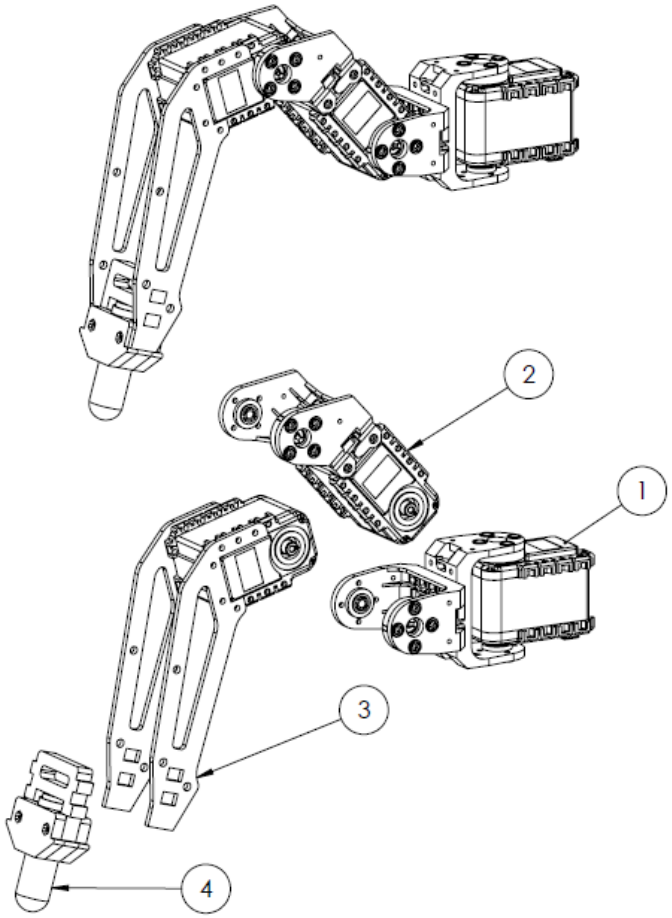
Tablica 2: Popis dijelova tehničke izvedbe Tijela

Redna oznaka na Slici 2	Naziv dijela
1	Gornja ploča
2	Odstojnici
3	Donja ploča

[3]Computer numerical control (CNC) je vrlo precizna kompjuterska kontrola mašine za rezanje, gdje su potrebne naredbe spremljene na odvojeni memorijski medij - link: `http://en.wikipedia.org/wiki/Numerical_control`

3.2. Noga

Noga razvojne platforme se pojavljuje šest puta u osnovnoj konstrukciji platforme. Od tih šest nogu tri se nalaze na desno a tri na lijevoj strani, gdje su one na desnoj zrcalno simetrične onima na desnoj i obrnuto. Zrcalna simetrija je uzrokovala suprotnu orijentaciju motora, što je dodatno programski riješeno. Pri konstrukciji Noge se pazilo na kompaktnost i što manju masu kako u njenom vrhu (alatu) ne bi dolazilo do neželjenih odstupanja, mala masa smanjuje rad motora te tako povećava njegovu preciznost dok kompaktnost eliminira zračnost između komponenti. Tehnička izvedba Noge prikazana je Slikom 3. Tri električna aktuatora omogućuju tri osi slobode te pozicioniranje alata u svaku točku radnog prostora.



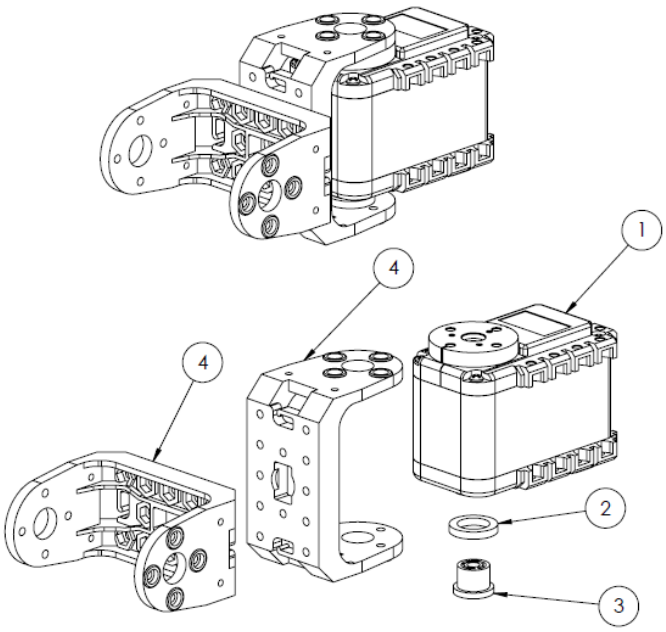
Slika 3: Prikaz tehničke izvedbe Noge

Tablica 3: Popis dijelova tehničke izvedbe Noge

Redna oznaka na Slici 3	Naziv dijela
1	Coxa
2	Femur
3	Tibia
4	Alat

3.2.1. Coxa

Ako se Noga razvojne platforme usporedi s anatomijom ljudske noge, Coxa predstavlja kuk. Dynamixel AX 12A[4] električni aktuator daje jednu os slobode okomitu na os slobode Femura i Tibie. Dva FP04-F2 plastična spojnika daju potrebu duljinu Coxi te omogućuju promjenu duljine po potrebi. Tehnička izvedba Coxe prikazana je Slikom 4.



Slika 4: Prikaz tehničke izvedbe Coxe

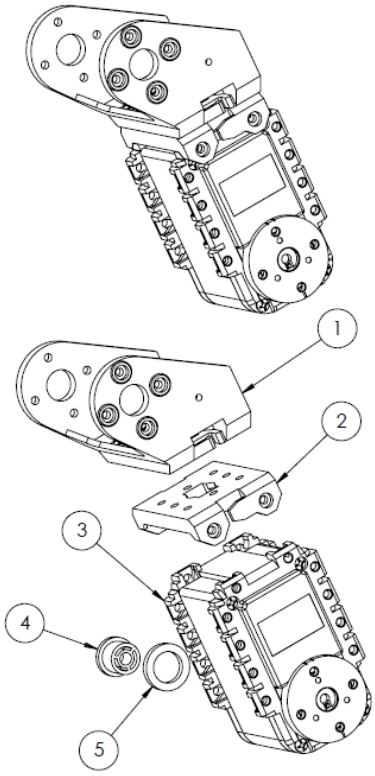
Tablica 4: Popis dijelova tehničke izvedbe Coxe

Redna oznaka na Slici 4	Naziv dijela
1	Dynamixel AX 12A električni aktuator
2	Unutarnji ležaj aktuatora
3	Vanjski ležaj aktuatora
4	FP04-F2 plastični spojnik

3.2.2. Femur

Ako se Noga razvojne platforme usporedi s anatomijom ljudske noge, Femur predstavlja natkoljenicu. Za razliku od Coxe, na Femuru se nalazi Dynamixel AX 18A[5] električni aktuator koji daje jednu os slobode okomitu na os Coxe te omogućuje veći okretni moment na osovini. Potreba za većim okretnim momentom dolazi zbog velikog momenta koju Femur treba svladati. Gledajući njegovu geometrijsku poziciju lako je za vidjeti da on mora svladati najveći dio utjecaja gravitacijske sile. FP04-F1 i FP04-F3 plastični spojnici daju potrebu duljinu Femuru te omogućuju promjenu duljine po potrebi. Tehnička izvedba Femura prikazana je Slikom 5.

[4]Detaljnije informacije o Dynamixel AX 12A električnom aktuatoru nalaze se u poglavlju 4.2.1
[5]Detaljnije informacije o Dynamixel AX 18A električnom aktuatoru nalaze se u poglavlju 4.2.1



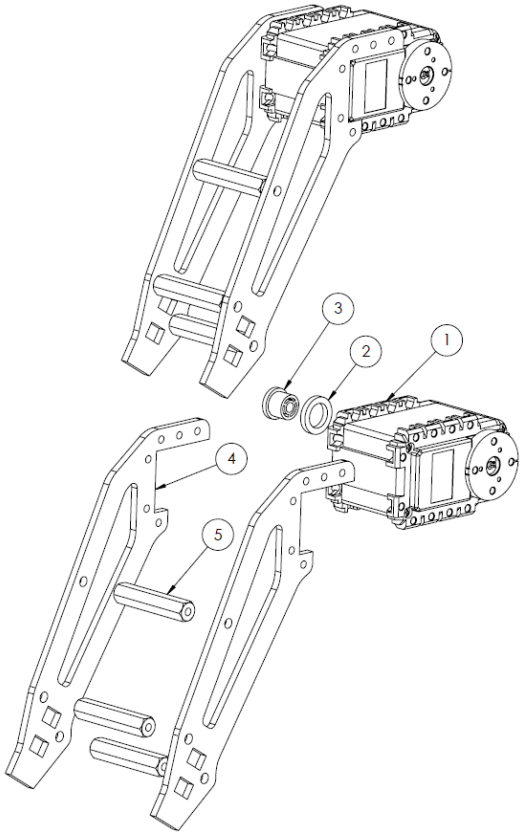
Slika 5: Prikaz tehničke izvedbe Femura

Tablica 5: Popis dijelova tehničke izvedbe Femura

Redna oznaka na Slici 5	Naziv dijela
1	FP04-F1 plastični spojnik
2	FP04-F3 plastični spojnik
3	Dynamixel AX 18A električni aktuator
4	Vanjski ležaj aktuatora
5	Unutarnji ležaj aktuatora

3.2.3. Tibia

Ako se Noga razvojne platforme usporedi s anatomijom ljudske noge, Tibia predstavlja potkoljenicu. Dynamixel AX 12A električni aktuator daje jednu os slobode paralelnu s osi slobode Coxe. Dvije aluminijske ploče koje su kao i ploče s tijela dobivene CNC rezanjem daju potrebu duljinu Tibii te omogućuju promjenu duljine po potrebi. Tri odstojnika duljine 30 milimetara drže Tibiu i Alat kompaktnim. Tibia je zamišljena da može imati različite Alate, ovisno o načinu korištenja Noge. Tehnička izvedba Tibie prikazana je Slikom 6.



Slika 6: Prikaz tehničke izvedbe Tibie

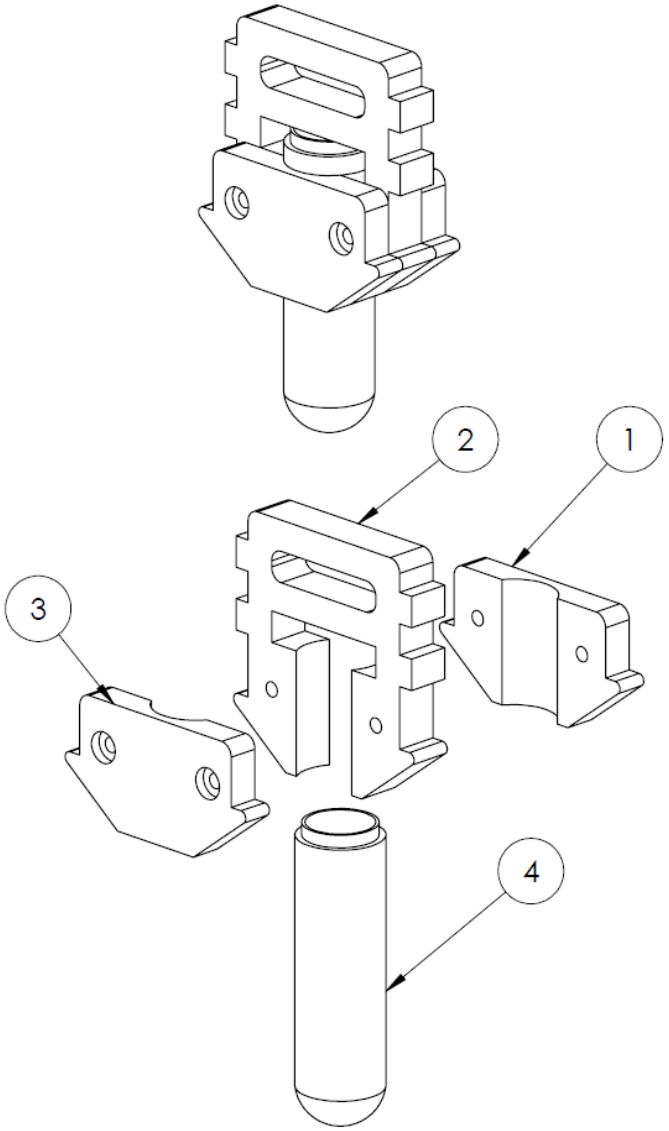
Tablica 6: Popis dijelova tehničke izvedbe Tibie

Redna oznaka na Slici 6	Naziv dijela
1	Dynamixel AX 12A električni aktuator
2	Unutarnji ležaj aktuatora
3	Vanjski ležaj aktuatora
4	Aluminijska ploča
5	Odstojnik duljine 30 milimetara

3.2.4. Alat

Alat je modularni dio Noge koji direktno određuje njenu ulogu. Zbog redundantnosti broja Nogu platforme, jedna ili više njih može se koristiti u druge svrhe. Tako, ako se Alat usporedi s anatomijom ljudske noge, on može predstavljati stopalo ili u drugim ulogama šaku. Platforma je u vrijeme pisanja ovog rada konfigurirana tako da sve Noge imaju ulogu lokomocije. Tehnička izvedba Alata prikazana je Slikom 7. Tri 3D printana dijela drže FSR[6] kompaktnim te omogućuju dodatnu promjenu duljine po potrebi.

[6]Force-sensing resistor (FSR) je otpornik koji mijenja svoj otpor ovisno o primijenjenoj sili. Detaljnije informacije o FSR-u nalaze se u poglavlju 4.3.2



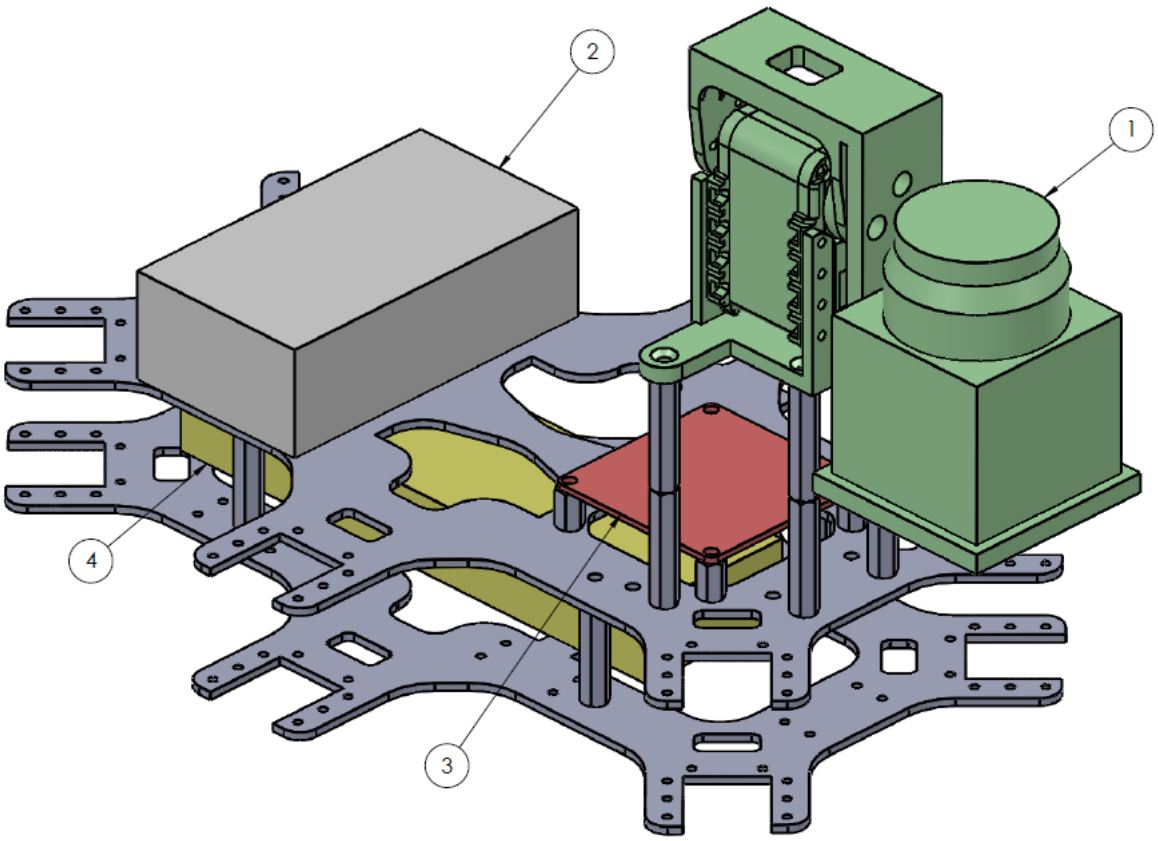
Slika 7: Prikaz tehničke izvedbe Alata

Tablica 7: Popis dijelova tehničke izvedbe Alata

Redna oznaka na Slici 7	Naziv dijela
1	Prvi 3D printani dio
2	Drugi 3D printani dio
3	Treći 3D printani dio
4	FSR

3.3. Ostale komponente

U kategoriju ostalih komponenti ulaze komponente koje se spajaju na Tijelo razvojne platforme. Njihova uloga jednako je važna kao i ranije navedenih cjelina ali oni predstavljaju kraj građevnog lanca tako da se na njih ne može dalje graditi. Slika 8 ih prikazuje na predviđenom položaju Tijela.



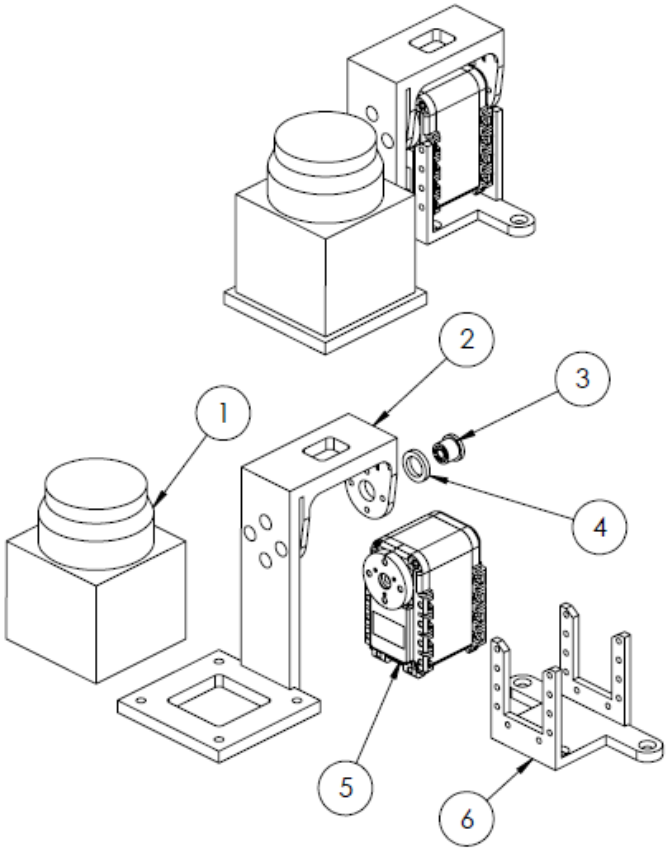
Slika 8: Prikaz ostalih komponenti spojenih na predviđene pozicije Tijela

Tablica 8: Popis ostalih komponenti spojenih na predviđene pozicije Tijela

Boja / redna oznaka na Slici 8	Naziv komponente
Zelena / 1	Postolje za laserski radar Hokuyo URG-04LX
Siva / 2	Glavna upravljačka jedinica Odroid U3
Crvena / 3	Sporedna upravljačka jedinica MultiWii
Žuta / 4	Baterija Zippy Compact

Postolje za laserski radar Hokuyo URG-04LX[7] nalazi se uzdignuto pomoću odstojnika na prednjem dijelu tijela kako bi radar imao pregledno i čisto vidno polje terena ispred razvojne platforme. Dynamixel AX 12A električni aktuator postavljen je na postolje da bi se laser mogao rotirati oko željene osi. Postolje je zbog svojeg jedinstvenog dizajna u potpunosti dobiveno 3D pintom.

[7] Detaljnije informacije o laserskom radaru Hokuyo URG-04LX nalaze se u poglavlju 4.3.1



Slika 9: Prikaz tehničke izvedbe postolja za laserski radar

Tablica 9: Popis dijelova tehničke izvedbe postolja za laserski radar

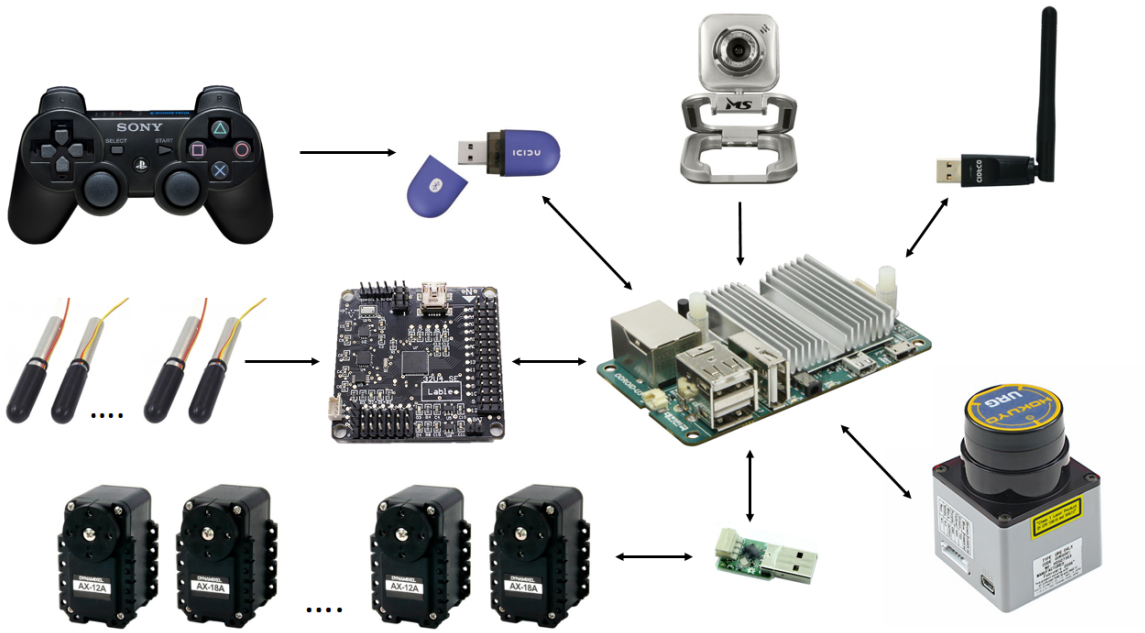
Redna oznaka na Slici 9	Naziv dijela
1	Laserski radar Hokuyo URG-04LX
2	Prvi 3D printani dio
3	Vanjski ležaj aktuatora
4	Unutarnji ležaj aktuatora
5	Dynamixel AX 12A električni aktuator
6	Drugi 3D printani dio

Glavna upravljačka jedinica Odroid U3[8] te Baterija Zippy Compact[9] svojim položajem kompenziraju masu laserskog radara i sporedne upravljačke jedinice MultiWii[10], kako bi centar mase platforme bio čim bliže njegovom geometrijskom središtu. Zbog lakšeg i učestalog micanja, glavna upravljačka jedinica i baterija spojene su s Tijelom preko čičak trake.

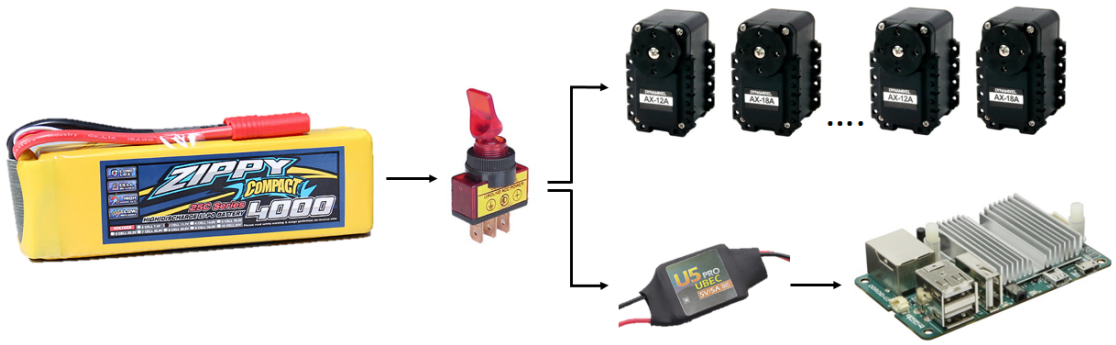
[8]Detaljnije informacije o glavnoj upravljačkoj jedinici Odroid U3 nalaze se u poglavlju 4.1.1
[9]Detaljnije informacije o bateriji Zippy Compact nalaze se u poglavlju 4.5.1
[10]Detaljnije informacije o sporednoj upravljačkoj jedinici MultiWii nalaze se u poglavlju 4.1.2

3.4. Funkcionalna shema

Kako bi se što bolje razumio funkcionalni aspekt načina rada razvojne platforme potrebno je poznavati njenu funkcionalnu shemu. Shema je prikazana na Slici 10 i Slici 11, gdje je iz praktičnih razloga shema napajanja odvojena na posebnoj fotografiji.



Slika 10: Funkcionalna shema razvojne platforme bez napajanja



Slika 11: Funkcionalna shema napajanja razvojne platforme

Slika 10 prikazuje funkcionalnu shemu platforme bez napajanja. U samom središtu sheme jest glavna upravljačka jedinica Odroid U3. Glavna upravljačka jedinica jako je bitan dio platforme te je svaka komponenta na shemi povezana s njom, posredno ili neposredno. Neposredno na glavnu upravljaču jedinicu, preko serijske USB komunikacije spajaju se: *Wi-Fi* modul za komunikaciju Hardkernel, *Bluetooth* modul za komunikaciju Icidu, USB2AX modul za komunikaciju, web kamera MS 301, laserski radar Hokuyo URG-04LX, sporedna upravljačka jedinica MultiWii MicroWii ATmega32U4. Ostale komponente koje su spojene posredno su: Playstation 3 upravljač koji se bežično spaja na *Bluetooth* modul, devetnaest električnih aktuatora Dynamixel AX-12A i AX-18A koji se

putem serijske digitalne komunikacije spajaju na USB2AX modul te šest senzora za očitanje sile Lynxmotion koji se putem analogne komunikacije spajaju na sporednu upravljačku jedinicu.

Funkcionalna shema napajanja razvojne platforme prikazana je Slikom 11. Napajanje cijele platforme omogućuje Litij-ion baterija Zippy Compact. Baterija se sastoji od tri ćelije koje zajedno daju napon od 11.1 volti. Direktno na bateriju spaja se prekidač za paljenje i gašenje platforme. Dobiveni napon na bateriji dovodi se preko prekidača na aktuatore. Taj napon previsok je za glavnu upravljačku jedinici stoga se koristi DC/DC pretvarač HKU5 5V/5A UBEC kako bi se napon prilagodio (pretvara se s 11.1 V na 5 V) i doveo do nje. Glavna upravljačka jedinica napaja sve ostale komponente.

4. Materijali

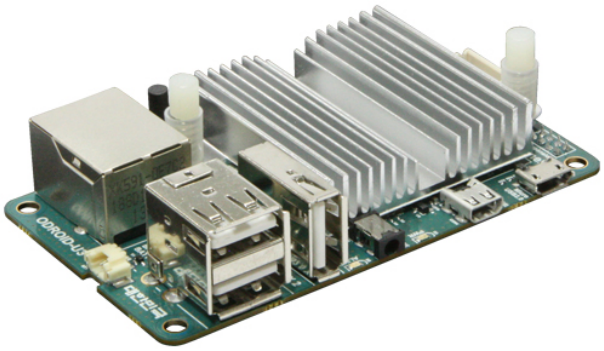
Materijali korišteni u izradi platforme mogu se podijeliti u 6 većih skupina: mikrokontroleri, aktuacija, senzorika, komunikacija i upravljanje, napajanje i programska podrška.

4.1. Mikrokontroler

U svijetu mobilne robotike, mikrokontroleri su obavezan dio svake platforme. Definiraju se kao mala računala integrirana na tiskanoj pločici koja sadrži processor, memoriju te programabilnu ulazno/izlane periferiju.

4.1.1. Odroid U3

Glavna upravljačka jedinica razvojne platforme je ugradbeno računalo bazirano na Linux operativnom sustavu (Slika 12).



Slika 12: Odroid U3 - glavna upravljačka jedinica razvojne platforme

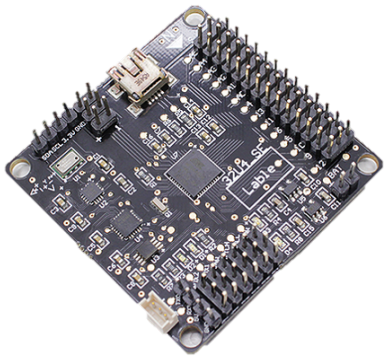
O ovom malom računalu nije potrebno mnogo govoriti što je ujedno i njegova najbolja osobina. Rukovanje takvim računalom je gotovo jednako kao i sa svakim drugim PC-em. To znači da, kada se na njega instalira odgovarajući operativni sustav, moguće je koristiti veliki broj alata bez posebnih prilagodbi. Zahvaljujući ubrzanom napretku ARM procesora, ovakvom računalu nikako ne nedostaje procesorske snage što se i pokazalo kroz višemjesečno korištenje.

Tablica 10: Tehničke specifikacije glavne upravljačke jedinice Odroid U3 [6]

Odroid U3	
Procesor	Exynos4412 Prime Cortex-A9 Quad Core 1.7Ghz, 1MB L2 cache
Memorija	2048MB(2GB) LP-DDR2 880Mega data rate
Video	HDMI monitor
IO Port	GPIO, UART, I2C, SPI
Pohrana podataka	MicroSD, eMMC
OS	Prilagođene Linux distribucije - Ubuntu, Android i mnogi drugi
Napajanje	5V 2A
Dimenzije	83 x 48 mm
Težina	48g s hladnjakom

4.1.2. MultiWii MicroWii ATmega32U4

Sporedna upravljačka jedinica razvoje platforme je ugradbeno računalo MultiWii MicroWii ATmega32U4 (Slika 13), u daljnjem tekstu MultiWii.



Slika 13: MultiWii MicroWii ATmega32U4 - sporedna upravljačka jedinica razvojne platforme

Uspoređujući s glavnom upravljačkom jedinicom sporedna je procesorski bitno slabija te ne posjeduje operativni sistem. Prednosti ovakve upravljačke jedinice su jednostavnost programiranja, senzorika te velik broj analognih/digitalnih pinova. Jednostavnost se postiže programiranjem u programskom okruženju Arduino[11] dok od senzora, upravljačka jedinica posjeduje:

1. Inercijalni mjerni sustav (IMU) koji se sastoji od:
   (a) Žiroskopa - senzor za mjerenje promjene kuta položaja tijela za kojega je vezan
   (b) Magnetometra - senzor za mjerenje jakosti magnetskog polja
   (c) Akcelerometra - senzor za mjerenje promjene linearne brzine kretanja tijela za kojega je vezan
2. Barometar - senzor za mjerenje atmosferskog tlaka

Analogni pinovi potrebni su kako bi se mogla uspostaviti komunikacija sa šest senzora za očitanje sile. Svaki od senzora spaja se na jedan analogni pin (A0 do A5). Digitalna komunikacija s glavnom upravljačkom jedinicom uspostavljena je preko USB komunikacije. Od tehničkih specifikacija upravljačke jedinice, bitno je nabrojati:

Tablica 11: Tehničke specifikacije sporedne upravljačke jedinice MultiWii [8]

MultiWii	
Procesor	ATmega32U4 - 16MHz
Ulazni napon	5.0 V
Broj analognih pinova	6 kom
Broj digitalnih pinova	8 kom
Težina	14 g
Dimenzije	50 x 50 x 16 mm

[11] Arduino programsko okruženje bazira se na C++ sintaksi te je napravljeno kako bi olakšalo korištenje novim programerima - link: `http://www.arduino.cc/`

4.2. Aktuacija

Aktuatori na razvojnoj platformi služe kao i mišići kod živih bića, daju joj mogućnost kretanja pretvarajući električnu u mehaničku energiju.

4.2.1. Dynamixel servo motori

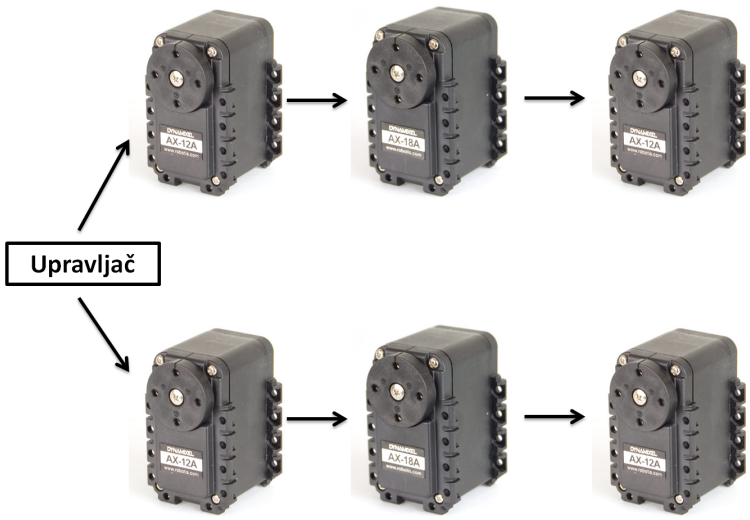
Za ovakav robotski sustav potrebni su aktuatori dovoljno kompaktni, snažni i pametni kako bi se razvojna platforma mogla kretati nesmetano i glatko. Drugim riječima poželjno je imati što manje ograničenja sa strane aktuatora kako bi algoritmi u razvoju došli do izražaja. Samo nekoliko godina unatrag, za ovakav robotski sustav bilo je potrebno razvijati posebne aktuatore po mjeri. Oni bi se sastojali od elektromotora, prijenosa, enkodera, mikrokontrolera, učinske elektronike, senzora za mjerenje struje, momenta, temperature itd. Danas se sve nabrojeno može naći na jednom mjestu. Zato su odabrani Dynamixel servo motori kao električni aktuatori (Slika 14). Točnije, odabrana su dva različita modela Dynamixel servo motora kako bi se zadovoljili određeni zahtjevi na snagu pojedinih zglobova razvojne platforme. Tehničke specifikacije su prikazane u tablici 12.

Slika 14: Dynamixel električni aktuatori AX-12A (gore) i AX-18A (dolje)

Tablica 12: Tehničke specifikacije odabranih Dynamixel servo motora [20]

	AX-12A	**AX-18A**
Napon	11.1V (9-12V)	11.1V (9-12V)
Max moment	1.2 Nm	1.8 Nm
Max brzina	59 RPM	97 RPM
Težina	55 g	54.5 g
Dimenzije	32 x 50 x 40 mm	32 x 50 x 40 mm
Rezlučljivost	0.29°	0.29°
Prijenos	1/254	1/254
Domet	300°	300°
Max struja	900 mA	2200 mA

Kada se spominje "pamet" ovakvih električnih aktuatora misli se na nekoliko specifičnih karakteristika. Prva pogodna karakteristika je serijska komunikacija što znači da nije potrebno imati zasebno ožičenje za svaki aktuator (Slika 15). U robotici je ova karakteristika iznimno važna jer se aktuatori često lančano nadograđuju jedan na drugoga.



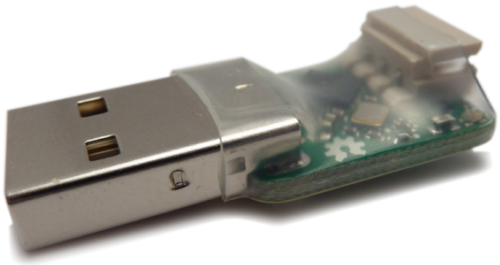
Slika 15: Prikaz serijskog povezivanja električnih aktuatora

Sljedeća važna stavka je integrirani mikrokontroler koji se brine za sve regulacije vezane za kretanje motora. S njim se komunicira TTL protokolom te se koristeći Dynamixel SDK [12] mogu upisivati i čitati vrijednosti iz registara putem C++ programskog jezika. Tako se upisivanjem vrijednosti u registre mogu zadavati veličine kao što su maksimalna brzina, maksimalni moment, željena pozicija i još mnogo toga.

[12]Dynamixel SDK je biblioteka u C++ programskom jeziku koja znatno olakšava komunikaciju s motorima - link: `http://support.robotis.com/en/software/dynamixelsdk.htm`

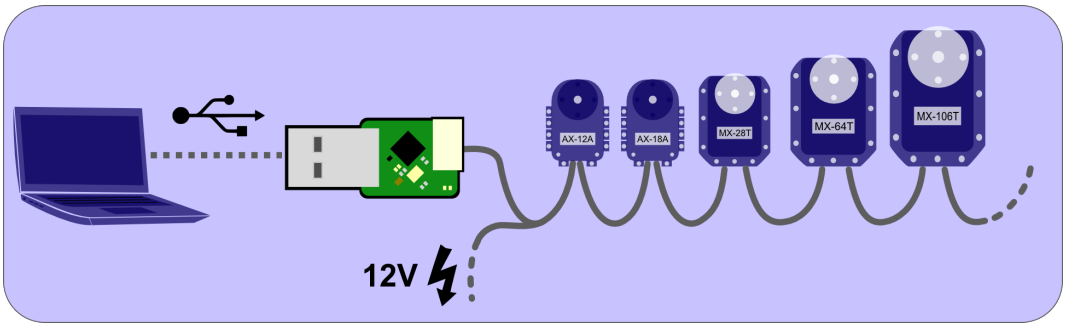
4.2.2. USB2AX

S aktuatorima Dynamixel serije se može komunicirati posebnim mikrokontrolerima koji podržavaju TTL komunikaciju ili koristeći posrednike koji će određene poruke pretvarati u one koje Dynamixel integrirani mikrokontroler razumije. Za slučaj ove razvojne platforme nije potrebno uvoditi dodatni mikrokontroler jer se cijeli proračun može vršiti na glavnoj upravljačkoj jedinici. Kao posrednik između glavne upravljačke jedinice i Dynamixel aktuatora odabran je USB2AX (Slika 32). Taj uređaj je razvijen kao manja verzija originalnog USB2Dynamixel posrednika. Kako se koristi modificirani uređaj (USB2AX), tj. ne koristi se originalni posrednik za komunikaciju s Dynamiel aktuatorima (USB2Dynamixel), potrebno je modificirati i Dynamixel SDK za primjenu u C++ programskom jeziku. Zato se koristi već pripremljena biblioteka pyUSB2AX [13] koja predstavlja izvedbu originalnog Dynamixel SDK u python programskom jeziku uz provedene potrebne prilagodbe za USB2AX.



Slika 16: USB2AX posrednik u komunikaciji [30]

Iznimno je važno paziti na povezivanje ovog uređaja s aktuatorima. Aktuatori se napajaju s 12V što bi moglo nanijeti značajnu štetu računalu na kojeg se USB2AX spaja preko USB-a. Zato treba pripaziti da naponska razina od 12V nikako ne dospije do USB2AX uređaja (Slika 17).



Slika 17: Shema povezivanja računala i aktuatora koristeći USB2AX posrednik [30]

[13] pyUSB2AX je modificirana Dynamixel SDK biblioteka napisana u python programskom jeziku - link: `https://github.com/jthorniley/pyusb2ax`

4.3. Senzorika

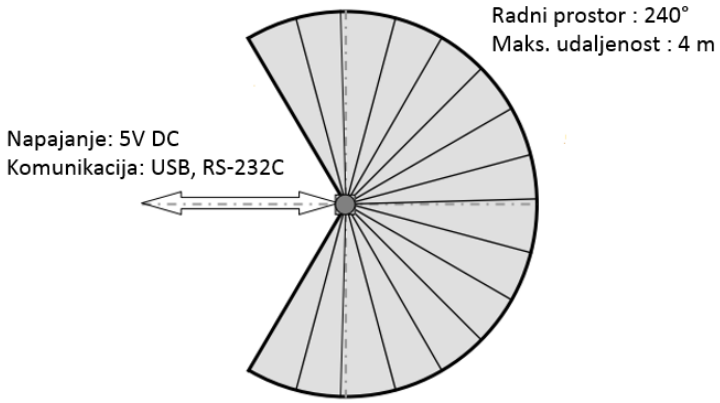
Kako bi razvojna platforma bila svjesna okoline koja ju okružuje te imala mogućnost snalaženja unutar nje potrebe su joj informacije o okolini koje joj pružaju senzori.

4.3.1. Laserski radar Hokuyo URG-04LX

Jedan od bitnijih senzora na razvojnoj platformi je laserski radar Hokuyo URG-04LX (Slika 18). Uz pomoć njega robot može biti svjestan objekata i terena koji se nalazi u neposrednoj blizini.

Slika 18: Laserski radar Hokuyo URG-04LX

Svrha radar je određivanje udaljenost do nekoga objekta putem snopa infracrvenih laserskih zraka. Snop zraka dobiva se na način da se jedna zraka kružnim slijedom "ispucava" unutar dvodimenzionalnog radnog prostora radara. Mjerenje udaljenosti bazira se na razlici faza poslane i primljene zrake, čime se minimizira utjecaj boje i refleksije pogođenog objekta [10]. Hokuyo svoj radni prosto definira kako je prikazano Slikom 19. Nakon odrađenog skeniranja (prijeđeni je cijeli radni prostor) šalju se informacije o očitanoj udaljenosti svake laserske zrake unutar skeniranja. Taj posao obavlja se frekvencijom 10 Hz. Informacije se šalju direktno glavnoj upravljačkoj jedinici putem USB komunikacije, kojom radar ujedno i dobiva 5 V napajanje.



Slika 19: Radni prostor laserskog radara Hokuyo URG-04LX

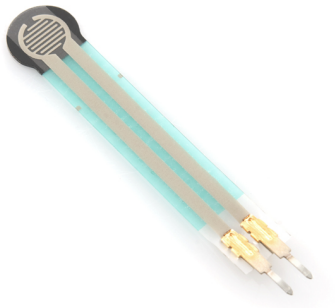
Na platformi je kućište radara pričvršćeno na servo motor (Dynamixel AX-12A) koji omogućuje njegovu rotaciju oko jedne osi u oba smjera. Os rotacije prolazi ishodištem svake laserske zrake. Rotacija radara otvara mogućnosti korištenja naprednijih algoritama za 3D mapiranje prostora i lokalizaciju unutar napravljene mape. Ostale tehničke informacije o Hokuyo radaru mogu se vidjeti u Tablici 13.

Tablica 13: Tehničke specifikacije laserskog radara Hokuyo URG-04LX [10]

Hokuyo URG-04LX	
Radni prostor	240°
Maksimalni domet	4000 mm
Kutna razlučivost	0.36°
Ulazni napon	5 V
Maksimalna struja	800 mA
Težina	160 g
Dimenzije	50 x 50 x 70 mm

4.3.2. FSR

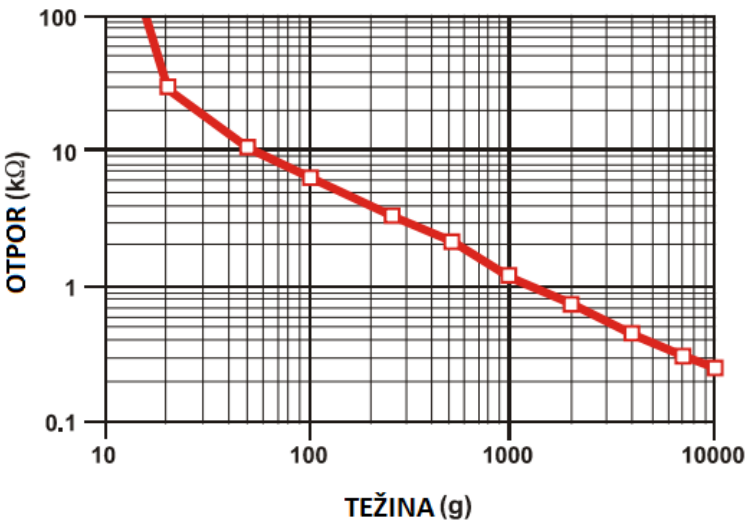
Force sensing resistor (u daljnjem tekstu FSR ili senzor sile) prikazan Slikom 20 je otpornik promjenjivog iznosa koji se mijenja ovisno o primijenjenoj sili na njegovu površinu.



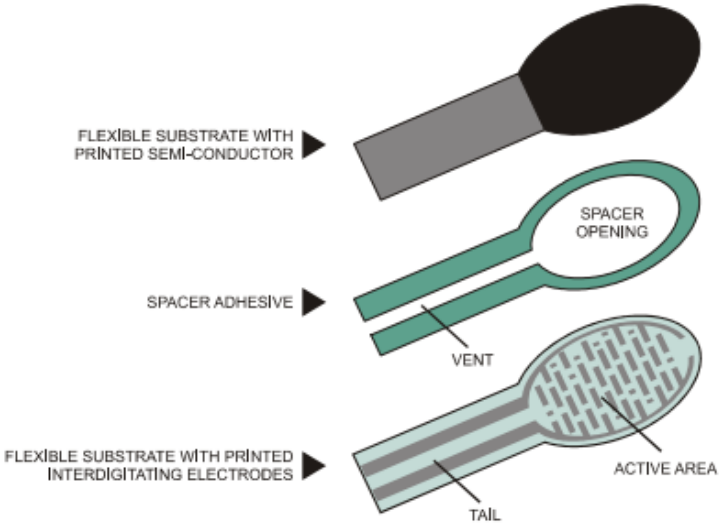
Slika 20: Senzor sile Lynxmotion FSR-01

Promjena otpora obrnuto proporcionalna je povećanju sile tako da on teži u ∞ pri iznosu sile 0 i u 0 pri iznosu sile ∞. Odnosa težine i otpora prikazan je Slikom 21. Takvo ponašanje postignuto je njegovom višeslojnom konstrukcijom prikazanom Slikom 22. Fleksibilni sloj poluvodiča na vrhu odvojen je od fleksibilnog sloja vodiča u praznom hodu. Primjenom sile i međusobnim doticanjem slojeva zatvara se strujni krug određenog otpora. Različite vrijednosti otpora kruga Ohmovim zakonom[14] uzrokuju različit napon kruga te se njegovim mjerenjem dobiva vrijednost sile.

[14]Ohmov zakon je temeljni zakon elektrotehnike. Govori o odnosu jakosti električne struje, napona i otpora u strujnom krugu - link: `http://hr.wikipedia.org/wiki/Ohmov_zakon`



Slika 21: Otpor senzora sile u ovisnosti o primijenjenoj težini [21]

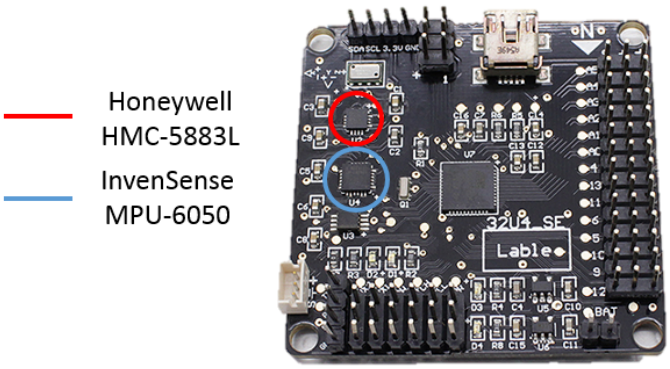


Slika 22: Prikaz tehničke izvedbe senzora sile [21]

4.3.3. Inercijalni mjerni sustav

Inercijalni mjerni sustav nalazi se na sporednoj upravljačkoj jedinici razvojne platforme MultiWii. Takav sustav sastoji se od 3 različita senzora: žiroskopa, magnetometra i akcelerometra. Žiroskop i akcelerometar nalaze se unutar zajedničke elektroničke kompenente InvenSense MPU-6050, dok se magnetometar nalazi u zasebnoj Honeywell HMC-5883L. Obje komponente mogu se vidjeti na Slici 23.

InvenSense MPU-6050 komunicira s periferijom putem digitalne komunikacije. Ima velik izbor različitih područja rada senzora. Žiroskop se može koristiti na područjima ±250, ±500, ±1000, ili ±2000°/sec. Akcelerometar se može koristiti na ±2g, ±4g, ±8g, ili ±16g [13]. Područje rada senzora izabire se ovisno o uvjetima u kojima se koristi. Dinamičniji sustav koristiti će postavke s većim područjem mjerenja kutne brzine i linearne akceleracije, posljedica toga su ne preciznija



Slika 23: Inercijalni mjerni sustav sporedne upravljačke jedinice

mjerenja i veći šum. Onaj tromiji koristiti će postavke s manjim područjem mjerenja kute brzine i linearne akceleracije ali preciznijim mjerenjima. Magnetometar također pruža digitalnu komunikaciju s periferijom i ima fiksno područje rada [11].

Ovakav sustav senzora pruža razvojnoj platformi informacije o trenutnoj orijentaciji u prostoru, linearnoj akceleraciji kao i kutnoj brzini platforme.

4.3.4. Web kamera MS Industrial 301

Web kamera postavljena na razvojnu platformu je MS Industrial 301 (Slika 24). Svojom veličinom i fleksibilnošću pokazala se kao jako praktičan izbor za platformu. Uz pomoć nje korisnik je u mogućnosti vidjeti direktno putem videa što se nalazi ispred platforme.

Slika 24: Web kamera MS Industrial 301

Video kojega kamera može snimati je u VGA[15] rezoluciji i snima se brzinom od 30 slika u sekundi [12]. Takva, niža rezolucija je sasvim dovoljna za potrebe razvojne platforme. Uz pomoć USB komunikacije snimljeni se video direktno šalje glavnoj upravljačkoj jedinici. Kamera također pruža

[15] *Video Graphics Array* (VGA) rezolucija je jedna od standardnih rezolucija korištenih u starijom računalima i mobitelima. Iznosi 640 x 480 piksela.

opciju snimanja zvuka koji se provodi zasebnom audio konekcijom. Ostale tehničke informacije vezanu za kameru mogu se vidjeti u Tablici 14.

Tablica 14: Tehničke specifikacije web kamere MS Industrial 301 [12]

MS Industrial 301	
Tehnologija	CMOS
Video rezolucija	640 x 480 piksela
Broj slika u sekundi	30
Digitalni zoom	Ne
Autofokus	Da
Ulazni napon	5 V

4.4. Komunikacija i upravljanje

Komunikacija između ne trivijalnih sustava razvojne platforme i njihovo upravljanja vrši se preko sljedećih komponenta.

4.4.1. Komunikacija

Kako bi razvojnoj platformi mogli pristupiti na što jednostavniji način koriste se dva tipa komunikacije. Prvo je potrebno osigurati da razvojna platforma ima stabilnu komunikaciju s drugim računalom kako bi se omogućilo nesmetano programiranje i nadzor. Za tu svrhu se koristi Hardkernel WiFi module 3 (Slika 25). Tako je moguće imati potpunu kontrolu glavne upravljačke jedinice tijekom izvođenja pokusa.

Slika 25: Hardkernel WiFi module 3 [6]

Osim WiFi konekcije potreban je još jedan tip komunikacije kako bi se lokalno povezao upravljač kojim se šalju jednostavne instrukcije za gibanje platforme. U ovom slučaju odabrana je Bluetooth komunikacija.

Slika 26: Bluetooth modul

Oba tipa komunikacije lako se ostvaruju zahvaljujući raširenosti Ubuntu operacijskog sustava. Oba modula se USB-om spajaju na glavnu upravljačku jedinicu a operacijski sustav se brine za sve ostalo. Na korisniku je dalje samo da ispravno konfigurira module za spajanje na željene mreže.

4.4.2. Upravljanje

Za direktno upravljanje gibanja razvojne platforme potreban nam je upravljač s upravljačkim palicama za zadavanje preciznih instrukcija gibanja te barem 10 tipkala koji reagiraju na silu pritiska različitih intenziteta. PlayStation 3 DualShock sixaxis upravljač se pokazao kao odličan kandidat jer zadovoljava sve gore navedeno uz dodatne opcije kao što su integrirani unutarnji inercijski mjerni sustav te LED signalizacija.

Slika 27: PlayStation 3 DualShock sixaxis upravljač [24]

Povezivanje je bežično putem bluetooth komunikacije opisane u prošlom poglavlju. Važno je napomenuti da se Linux kernel u ovom slučaju neće pobrinuti za komunikaciju s upravljačem već je potrebno instalirati *driver*[16] namijenjen za upravo taj upravljač. Objašnjenje funkcije svakog tipkala i upravljačke palice na upravljaču se nalazi u zasebnom poglavlju.

[16]Za instalaciju *driver*-a potrebno je pratiti upute na sljedećem linku: `http://www.raspians.com/Knowledgebase/ps3-dualshock-controller-install-on-the-raspberry-pi/`

4.5. Napajanje

Kako bi sve komponente radile potrebno im je napajanje. Na razvojnoj platformi za napajanje se brinu sljedeće komponente.

4.5.1. Baterija Zippy Compact

Komponenta koja pruža napajanje cijeloj razvojnoj platformi je baterija Zippy, serije Compact. Baterija je prikazana na Slici 28. Kemijski sastav baterije je Litij-ion a ona se sastoji od tri ćelije povezane u seriju. Tako povezane ćelije tvore napon od 11.1 V i daju mogućnost maksimalne izlazne struje pražnjenja do 100 A [9]. Kada se baterija želi puniti potrebno je koristiti poseban punjač. Ovakve karakteristike uobičajene su za baterije s Litij-ion kemijskim sastavom. Ostale tehničke karakteristike baterije dane su u Tablici 15.

Slika 28: Baterija Zippy Compact

Tablica 15: Tehničke specifikacije baterije Zippy Compact [9]

Zippy Compact	
Kapacitet	4000 mAh
Broj ćelija	3
Nazivni napon	11.1 V
Maksimalna struja pražnjenja	100 A
Maksimalna struja punjena	20 A
Težina	286 g
Dimenzije	148 x 34 x 20 mm

Autonomija razvojne platforme, pri punom radu (konstantnom kretanju aktuatora i s upaljenim svim senzorima), iznosi ~ 30 minuta. Aktuatori su najveći potrošači električne energije a platforma se rijetko koristi na punom opterećenju. Uzevši to u obzir, autonomija razvojne platforme može doseći i preko 60 minuta.

4.5.2. Pretvarač HKU5 5V/5A UBEC

Glavna upravljačka jedinica na razvojnoj platformi ne može primati 11 V napona koju baterija daje direktno na svom izlazu pa se koristi DC/DC pretvarač kao bi spustio razinu napona na 5 V. Ostale komponente koje zahtijevaju 5 V, dobivaju napajanje s glavne upravljačke jedinice. Pretvarač korišten na platformi je HKU5 5V/5A UBEC (Slika 29).

Slika 29: DC/DC pretvarač HKU5 5V/5A UBEC

Tablica 16: Tehničke specifikacije DC/DC pretvarač HKU5 5V/5A UBEC [7]

DC/DC pretvarač HKU5 5V/5A UBEC	
Ulazni napon	7.2 V - 21 V
Izlazni napon	5 V
Izlazna struja	5 A
Težina	11.5 g
Dimenzije	49 x 20 x 12 mm

4.6. Programska podrška

U kategoriju programske podrške ulaze sva programska okruženja i operacijski sustavi korišteni u ovom radu.

4.6.1. Linux - Ubuntu 14.04

Slika 30: Logo Linux jezgre za operacijske sustave [15] (lijevo), logo Ubuntu distribucije [28] (desno)

Glavna upravljačka jedinica Odroid U3 je dizajnirana za korištenje s Linux [17] operacijskim sustavom. Odabrana distribucija Linux-a je Ubuntu 14.04 jer je to ujedno i najraširenija distribucija Linux operacijskih sustava s najraširenijom zajednicom. Zajednica je od iznimne važnosti kod razvoja jer se gotovo svako rješenje problema tehničke naravi može pronaći kao tema na jednom od brojnih foruma. Operacijskom sustavu se prepušta briga o mnogim stvarima kao što su internet i bežična konekcija, upravljanje procesima i još mnogo toga. Uz to se ne ograničavamo na isključivo konzolno korištenje već je po potrebi moguće i osposobiti modificirano Ubuntu grafičko sučelje za lakši rad i postavke sustava (Slika 31).

Slika 31: Grafičko sučelje Ubuntu distribucije Linux-a za Odroid U3

Iako je grafičko sučelje korisno za slučajeve kada se niti na jedan način ne može povezati s upravljačkom jedinicom, uobičajeni način komunikacije s njom je preko *Secure Shell* [18] mrežnog protokola. Detaljniji opis ove metode povezivanja je opisan u poglavlju 5.

[17] Linux je jezgra(kernel) računalnog operacijskog sustava sličnog Unixu. Često se naziv "Linux" koristi i za sam operacijski sustav

[18] SSH je kriptirani mrežni protokol za komunikaciju s udaljenim računalima na siguran način

4.6.2. ROS

Slika 32: ROS logo

Robot Operating System (u daljnjem tekstu ROS) je fleksibilan okvir za pisanje softvera za robotske sustave ali i šire od toga. Radi se o skupu alata, biblioteka i konvencija kojima je cilj pojednostaviti zadatak stvaranja kompleksnih i robusnih robotskih sustava na velikom broju različitih platformi [22]. Odabrana je posljednja distribucija ROS-a: ROS Indigo Igloo.

Za korištenje razvojne platforme nužno je minimalno poznavanje ROS-a što uključuje sljedeće pojmove:

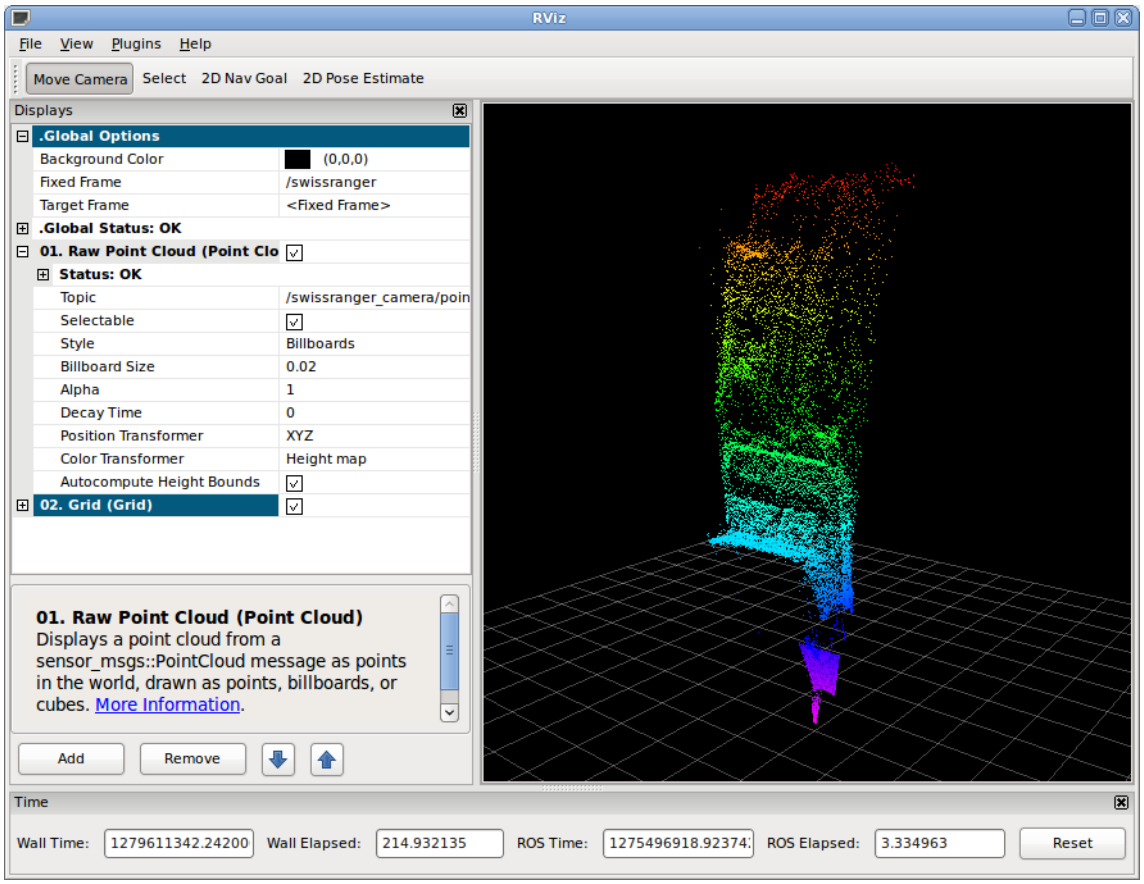
- **roscore** - Predstavlja skup *node*-ova i programa koji su nužni za ispravan rad ROS-a. Obično je to prva naredba koja se pokreće pri radu s ROS-om jer je to osnova svake daljnje komunikacije u okviru ROS-a.
- **node** - Predstavlja izvršnu programsku cjelinu koja obavlja neki rad. Podržani programski jezici su C++ te Python. Najčešće se prije rada pretplaćuje na *topic* te tako prima podatke, a tek nakon objavljuje podatke na neki drugi *topic*.
- **topic** - Predstavlja sabirnice namijenjen za komunikaciju između *node*-ova. *Node*-ovi preko *topic*-a razmjenjuju unaprijed definirane poruke ili *msg*-e.
- **msg** - Unaprijed definirane poruke koje se koriste kao kalup za strukturiranje podataka. Tek kada se podaci strukturiraju kao što određena poruka definira mogu se slati na *topic*.

Osim navedenih pojmova ROS se odlikuje raznim svojstvima i alatima. Za sve detalje valja posjetiti službenu internet stranicu [22].

4.6.3. Rviz

Slika 33: Rviz logo [23]

Rviz je paket koji dolazi u sklopu operativnog sistema ROS i služi za vizualizaciju podataka koje razvojna platforma dobiva od senzora ili algoritama. U tom pogledu uvelike olakšava razvoj novih algoritama i omogućuje bolje poznavanje podataka koje senzori šalju na platformu. Rviz se preplaćuje na Ros topic-e te s njih čita sve potrebne podatke za prikazivanje. Prednost ovakvog programa je u tome što se može u svakom trenutku pretplatiti na topic a da pritom ne poremeti podatke koji se nalaze na nju. Te iste podatke mogu nesmetano koristiti svi drugi dijelovi ROS operativnog sustava [23]. Od senzora na platformi, laserski radar i inercijalni mjerni sustav naviše koriste spomenuti softvere. Primjer korištenja vidljiv je na Slici 34.



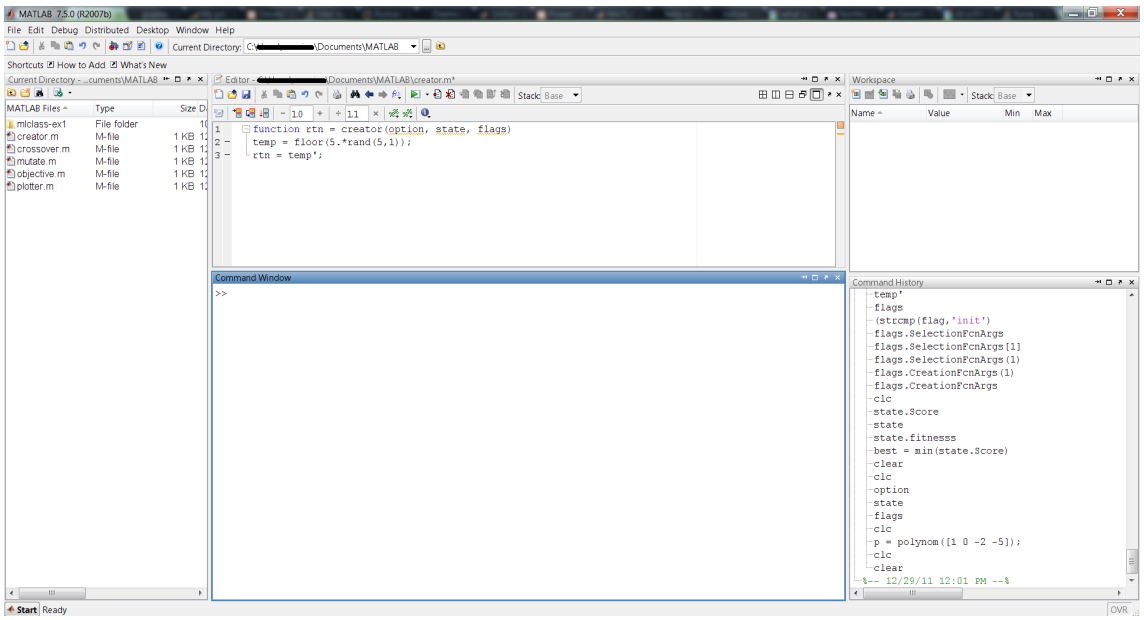
Slika 34: Prikaz Rviz korisničkog sučelja

4.6.4. Matlab

Slika 35: Mathworks Matlab logo [16]

Matlab je programski jezik visoke razine apstrakcije i interaktivno okruženje za programiranje koje uvelike olakšava rješavanje matematičkih problema bilo koje vrste. Osim samog računa, Matlab pomaže i pri vizualizaciji dobivenih rezultata te se može koristiti i za grafički prikaz rezultata dobivenih u raznim pokusima.

U ovom slučaju Matlab je korišten za proračun kinematičkih jednadžbi udova na razvojnoj platformi. Račun se vršio nad kvadratnim matricama četvrtog reda uz mnogo trigonometrije što bi bilo gotovo nemoguće ručno izračunati. Sam račun je objašnjen u poglavlju 5.1.1.



Slika 36: Mathworks Matlab sučelje

4.6.5. SolidWorks

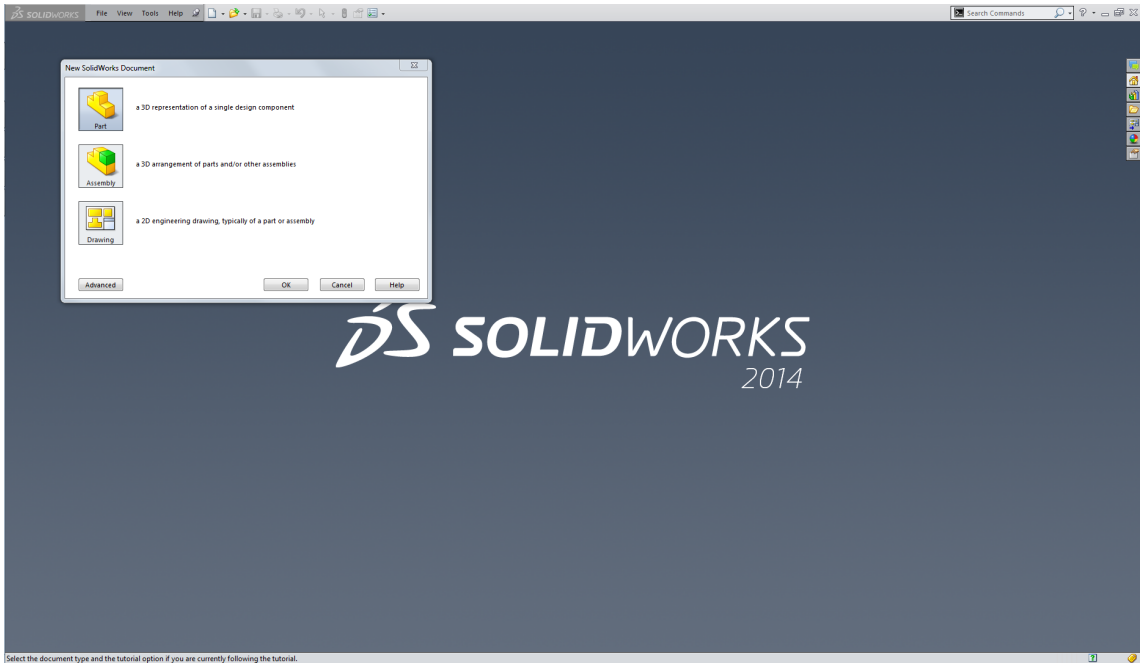
Slika 37: Dassault Systèmes SolidWorks logo [25]

SolidWorks je 3D CAD okruženje rađeno samo za Windows operacijske sustave. Unutar Solidworksa nalaze se alati za CAD dizajn, simuliranje, tehničku komunikaciju i električni dizajn novog produkta. U ovom radu je on primarno korišten za CAD dizajn platforme.

Postupak dizajna sastoji se od dvije faze:

- Part faza - Predstavlja fazu u kojoj se svaki part (dio) zasebno modelira te sprema u memoriju računala.
- Assembly faza - Predstavlja fazu u kojoj se ranije napravljeni dijelovi slažu u novi assembly (montažu). Svaki dio u montaži treba imati strogo definirani fizički odnos s drugim dijelovima montaže, kako bi se mogle simulirati kretnje i ponašanje u stvarnosti.

3D vizualizacijom platforme lakše je uočiti moguće pogreške, postaviti komponente na pravo mjesto, testirati njihovo ponašanje, poslati potreban CAD dizajn na CNC rezanje itd.



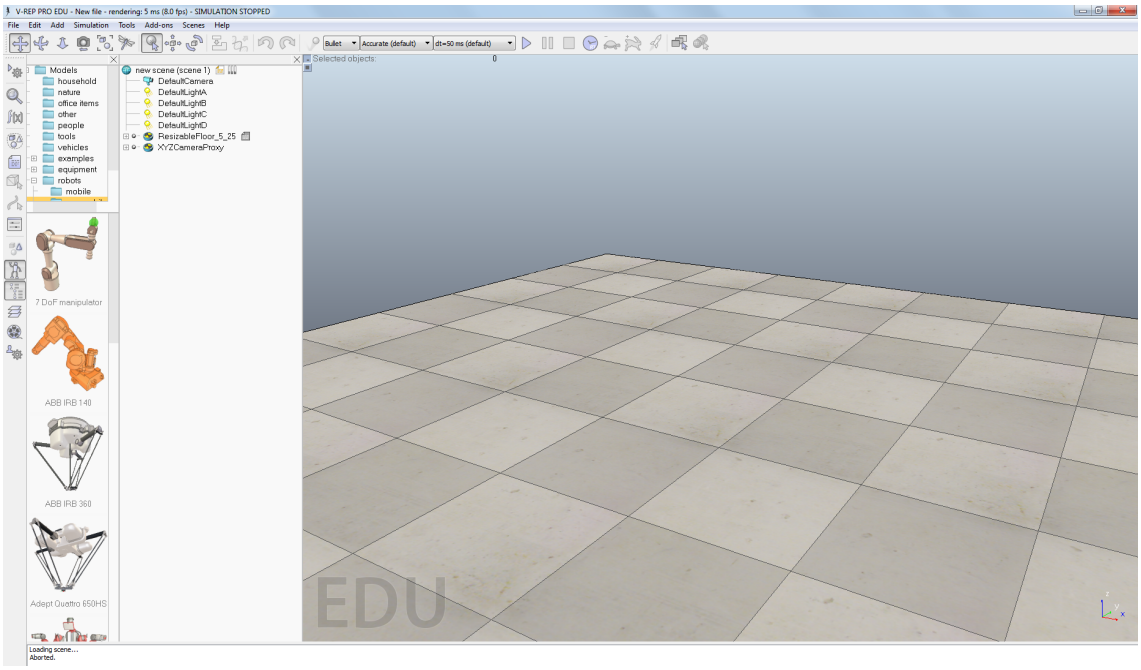
Slika 38: Dassault Systèmes SolidWorks sučelje

4.6.6. V-rep

Slika 39: Coppelia Robotics v-rep logo [19]

Virtual robot experimentation platform (u daljnjem tekstu v-rep) je *open source* simulacijska platforma robota koja se može koristiti na Windows i Linux operativnim sustavima. Mogućnost rada na Linux operativnom sustavu omogućila je integraciju v-repa i razvojne platforme. Simulatori ovakve vrste koriste se kako se bi se testirali i razvili novi robotski sustavi. Puno je lakše i isplativije testirati nove sustave u simulatoru te kasnije kad se ostvare željeni rezultati sve prebaciti na stvaran sustav.

Mogućnost simuliranja fizike objekata unutar v-repa uvelike pomaže pri dizajnu i vizualizaciji robotskog sustava. Prva testiranje platforme odrađena su u različitom simulatoru koji nije imao tu sposobnost, gdje je zapravo i uočena potreba za njome. Za sve detalje može se posjetiti službena stranica [19].



Slika 40: Coppelia Robotics v-rep sučelje

5. Metode

Metode predstavljaju glavini ovoga rada. U njima je opisana realizacija svih ciljeva postavljenih u poglavlju 2.

5.1. Algoritam hoda

Upravljanje kretanjem ovakvog robota zahtjeva određena znanja robotike kao što su principi inverzne i direktne kinematike te planiranje putanja po kojima se kreću vrhovi nogu. Tek tada je ovom razvojnom platformom moguće upravljati na prihvatljiv način. U nastavku su opisane korištene metode algoritma hoda.

5.1.1. Kinematika

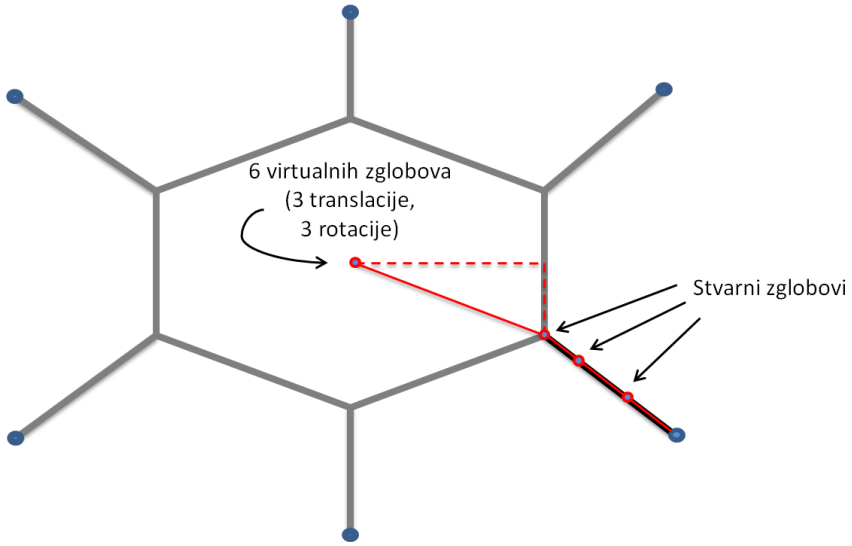
Razvojna platforma je mobilni robot koji se giba pomoću 6 manipulatora (u daljnjem tekstu noge), od kojih svaki ima 3 osi slobode. Upravljanje zakretima svih 18 zglobova problem je kojemu rješenje nije trivijalno. Ukoliko se želi postići glatko i koordinirano kretanje robota u svakom trenutku je nužno znati kakvim zakretom zglobova je potrebno kretati noge da one završe u točno određenim točkama u prostoru. Drugim riječima, za zadane točke u prostoru, definirane pripadajućim vektorima konfiguracije alata potrebno je pronaći pripadajuće vrijednosti varijabli zglobova. Definicija problema inverzne kinematike: *Ako su zadane vrijednosti položaja p i orijentacije alata R, tada je potrebno pronaći vrijednosti varijabli u prostoru zglobova* $R^n(q_i, 1 \leq i \leq n)$ *koje zadovoljavaju jednadžbu manipulatora.* [14]. No prije inverzne kinematike potrebno je odrediti izraze direktne kinematike. Oni govore gdje će završiti alat manipulatora R te koje će biti orijentacije p uz zadane varijable u prostoru zglobova q_i. Direktnu kinematiku je potrebno analizirati za svaku od 6 nogu robota. Naravno, sve noge su iste konfiguracije pa će se tako račun jedne noge lako prilagoditi za ostale.

Sada se postavlja pitanje kako pristupiti matematičkom opisivanju jedne noge. Za translacijsko gibanje (na mjestu, poput naginjanja u naprijed) robota dovoljno je uzeti u obzir samo 3 zgloba svake noge i kretati ih bez da je bilo koja noga "svjesna" pozicije ostalih nogu. Drugim riječima, ako se želi da robot translatira u naprijed dovoljno je da vrhovi svih šest nogu "guraju" unatrag (gledano iz perspektive robota). Moglo bi se reći da svaka noga ima 3 osi slobode pa kada bi se sve noge kretale u istom smjeru i tijelo se može gibati u samo 3 osi slobode. Uz određene alternacije takvih kretanja nogu unaprijed i unatrag moguće je postići i hodanje unaprijed no više o tome u sljedećim poglavljima.

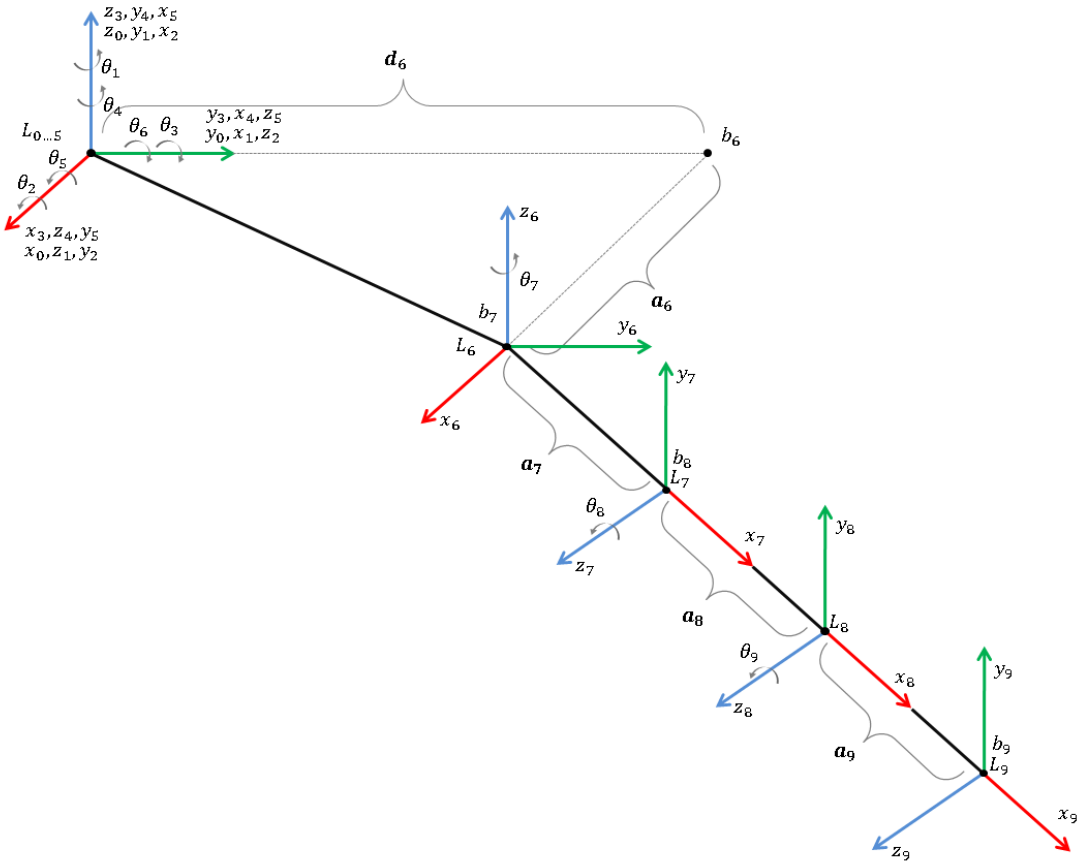
Ako se robot treba rotirati (naginjati) oko svoje 3 osi treba promijeniti pogled na noge. Potrebno je postići sklad u planiranju pokreta svih nogu tako da sve skupa rade na naginjanju tijela. Nije dovoljno promatrati samo 3 zgloba svake noge pa se uvode virtualni zglobovi. Točnije 6 virtualnih zglobova (3 translacijska i 3 rotacijska). Oni su indirektno odgovorni za rotaciju i translaciju tijela na mjestu. Prvo se virtualnim zglobovima odredi koja je željena translacija i rotacija tijela a zatim se gleda gdje bi zbog toga trebali biti postavljeni *coxa*[19] zglobovi svih nogu te kako da svaka noga od

[19] Detaljan opis zglobova svake noge nalazi se u poglavlju 3.2

tamo dođe do željene točke na podu. Dakle, virtualni zglobovi su tu samo zbog planiranja i računa dok su stvarni zglobovi tu da se obavi stvarni (fizički) pomak robota. Kako bi opisana metoda bila jasnija valja pogledati slike 41 i 42.

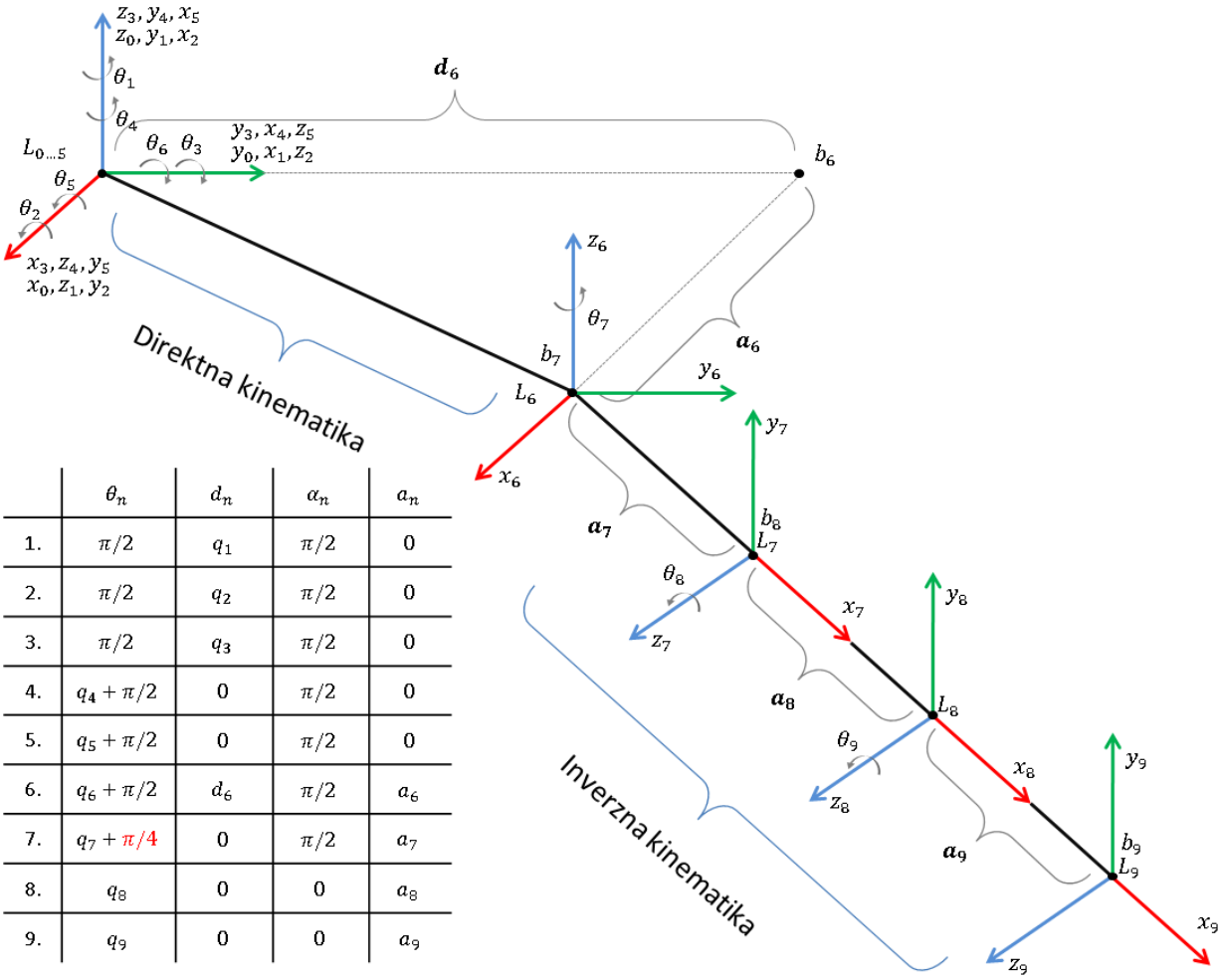


Slika 41: 6 virtualnih zglobova (3 translacije i 3 rotacije) i 3 stvarna zgloba jedne noge



Slika 42: Koordinatni sustavi jedne noge (6 virtualnih i 3 stvarna)

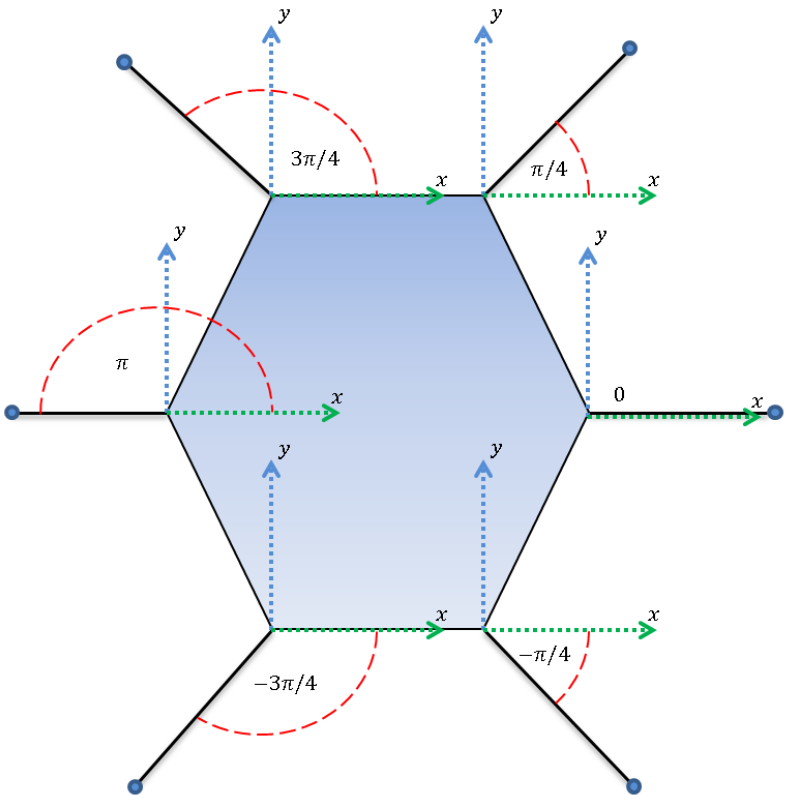
Kao što je opisano, prvo se za zadanu translaciju i rotaciju tijela direktnom kinematikom gleda gdje bi trebali završiti kukovi robota a onda se inverznom kinematikom računa potreban zakret stvarnih zglobova kako bi noga završila u željenoj točki prostora. Na slici 42 je dodan 10. koordinatni sustav koji predstavlja koordinatni sustav vrha noge. Denavit-Hartenbergovom metodom (DH parametri u daljnjem tekstu) [14] se za ovakvu konfiguraciju dobivaju parametri prikazani slikom 43. Na slici q_i predstavlja varijable virtualnih i stvarnih zglobova (njihov zakret), a_i predstavlja duljine članaka a d_i predstavlja širine zglobova.



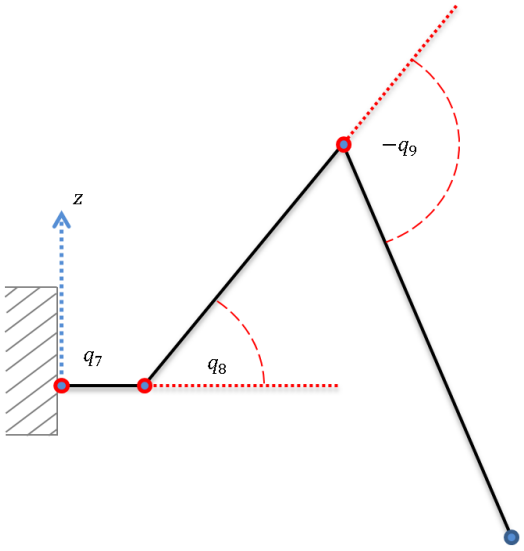
	θ_n	d_n	α_n	a_n
1.	$\pi/2$	q_1	$\pi/2$	0
2.	$\pi/2$	q_2	$\pi/2$	0
3.	$\pi/2$	q_3	$\pi/2$	0
4.	$q_4 + \pi/2$	0	$\pi/2$	0
5.	$q_5 + \pi/2$	0	$\pi/2$	0
6.	$q_6 + \pi/2$	d_6	$\pi/2$	a_6
7.	$q_7 + \pi/4$	0	$\pi/2$	a_7
8.	q_8	0	0	a_8
9.	q_9	0	0	a_9

Slika 43: DH parametri potrebni za proračun direktne i inverzne kinematike

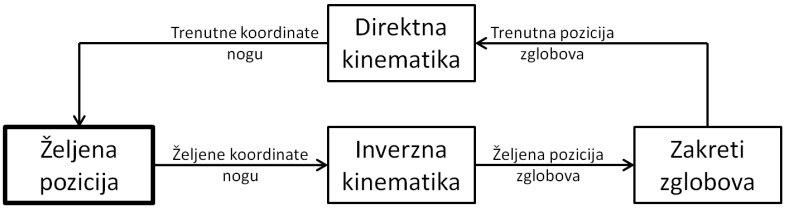
Matlab je korišten za sve proračune direktne i inverzne kinematike. Prikaz dobivenih rezultata je prevelik za prikaz u matričnom obliku. On se može vidjeti u priloženom kodu u *Leg.cpp* fajlu. Na dobivenim DH parametrima treba obratiti pažnju na zacrvenjeni parametar $\theta_7 = q_7 + \pi/4$. $\pi/4$ koji predstavlja zakret kuka prve desne noge robota. On je za svaku nogu drugačiji te je to ujedno i jedini parametar koji se mijenja kako bi se račun prilagodio za svaku nogu. Svi zakreti dani su slikom 44. Svaka rotacija zglobova definirana je njegovim zakretom oko z osi. Tako se određuju i orijentacije zakreta zglobova svake noge (Slika 45). Ovako postavljeni model robota predstavlja najniži nivo apstrakcije upravljanja robotom. On se brine da vrh svake noge završi na točno određenim koordinatama metodom prikazanom na slici 46.



Slika 44: Koordinatni sustavi kukova (*coxa*) svih nogu uz njihov zakret u odnosu na referentni koordinatni sustav



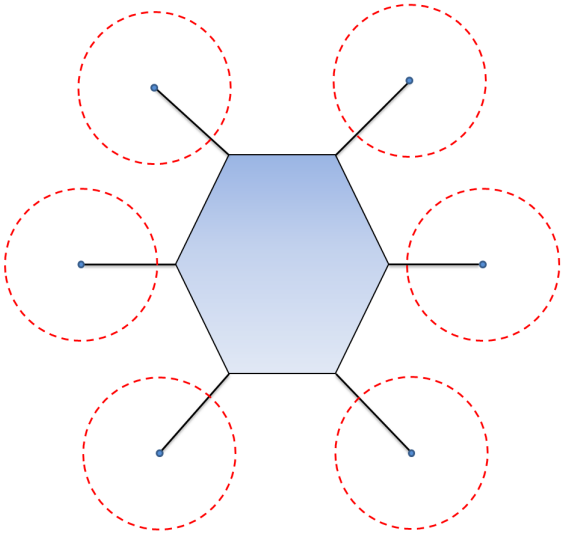
Slika 45: Orijentacija zakreta zglobova jedne noge



Slika 46: Shema metode poveznice željene pozicije vrha noge i zakreta zglobova

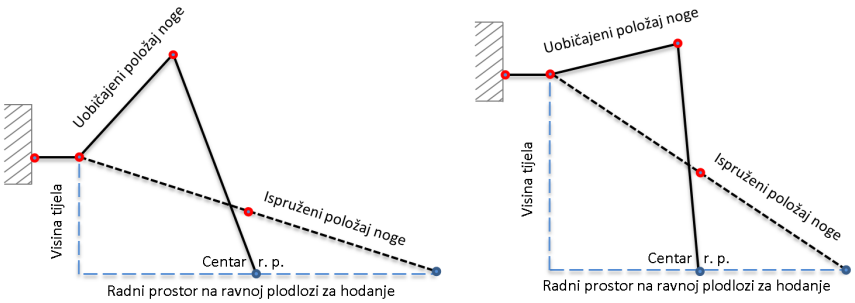
5.1.2. Radni prostor

Prije nego se krene u plan hoda robota potrebno je definirati prostor u kojemu će se njegove noge sigurno kretati bez kolizija (u daljnjem tekstu radni prostor). Nakon mnogo testiranja i pokusa pokazalo se da je radni prostor svake noge najbolje opisati kružnicom na plohi po kojoj hoda (Slika 47).



Slika 47: Radni prostori svih nogu

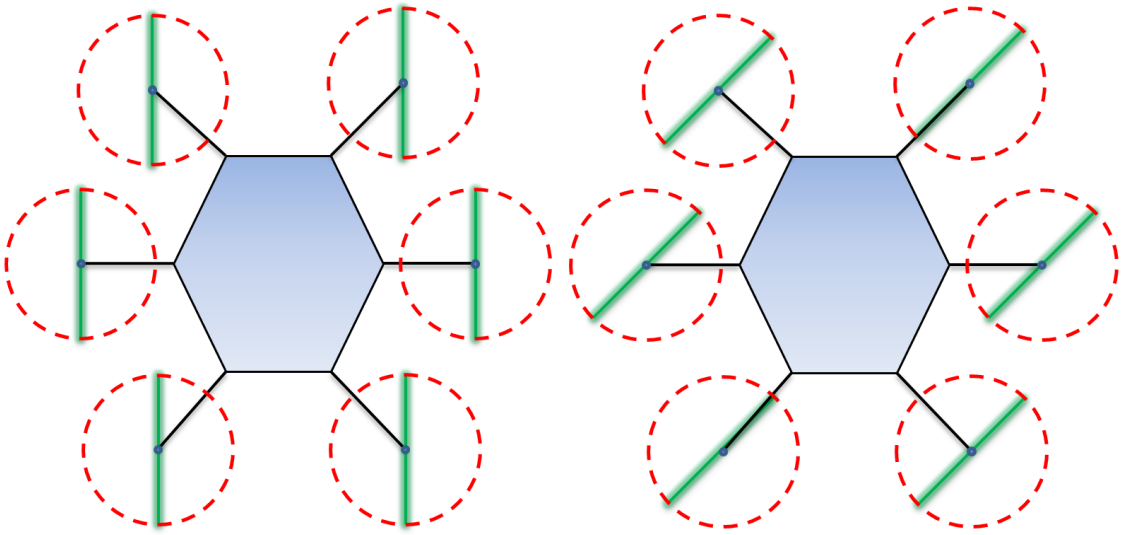
Radni prostor treba biti promjenjivog radijusa i središta kako bi se osiguralo da noga za svaku rotaciju i translaciju tijela može doći u svaki dio radnog prostora. Pritom treba paziti da ne dođe do kolizije među nogama. Primjer promjenjivog radijusa i središta radnog prostora može se prikazati na primjeru različitih visina robota (Slika 48). Kako se tijelo diže, tako noga ima manji domet pa se središte mora pomaknuti prema sredini robota te se radijus mora smanjiti kako noga ne bi došla preblizu tijelu.



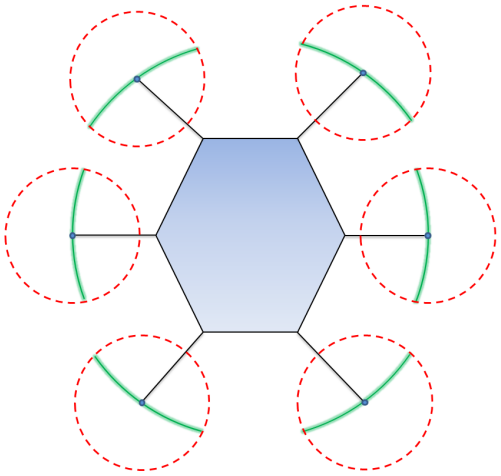
Slika 48: Primjer promjenjivog radnog prostora s obzirom na visinu robota

5.1.3. Jezgra

Nakon definiranja radnog prostora svake noge treba odrediti putanje po kojima se one moraju kretati kako bi se robot rezultantno gibao u željenom smjeru. Ako se noga koja dodiruje tlo iz perspektive robota kreće u natrag, intuitivno je jasno da ona tako "gura" robota prema naprijed gledano iz perspektive globalnog koordinatnog sustava. Slika 49 pokazuje po kojim putanjama se trebaju kretati noge robota kako bi se on u konačnici kretao ravno unaprijed odnosno pod 45°. Ako se robot rotira oko svoje z osi putanje trebaju biti kako pokazuje slika 50.



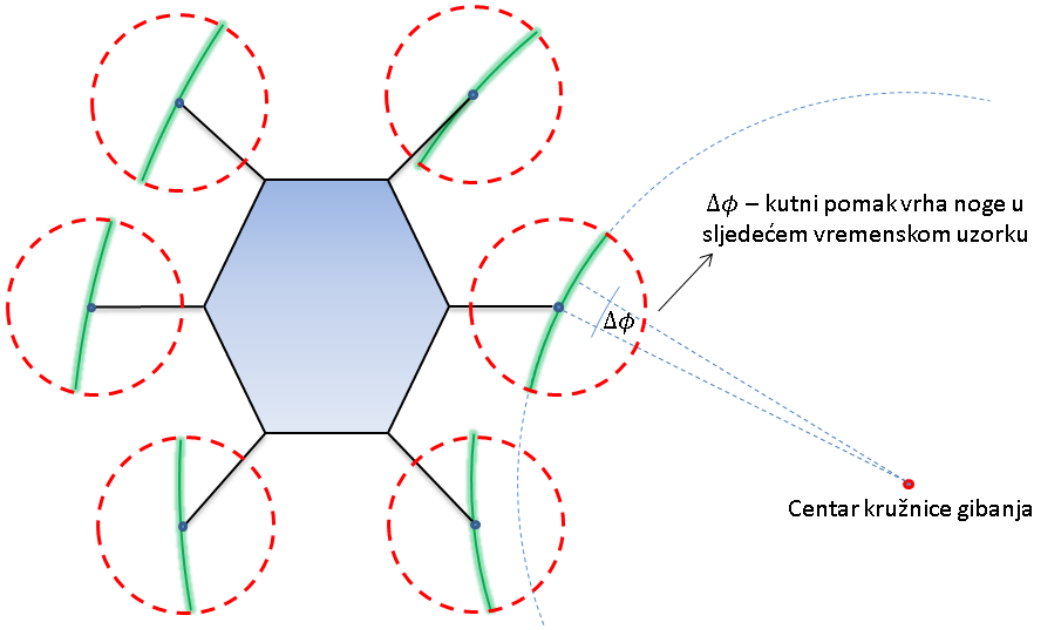
Slika 49: Kretanje robota ravno unaprijed (lijevo) odnosno pod 45°(desno). Zelenim linijama su pokazane putanje vrhova nogu



Slika 50: Rotiranje robota tj. kretanje ono svoje z osi. Zelenim linijama su pokazane putanje vrhova nogu

Poopćenjem prethodnih metoda kretanja može se zaključiti da je svako kretanje robota moguće opisati kretanjem po kružnici. Pravocrtno kretanje robota se može opisati kretanjem po kružnici beskonačnog polumjera (ili aproksimativno dovoljno velikog polumjera) i središta negdje u beskonačnosti (važno je gdje zbog smjera kretanja) dok se svako zakrivljeno kretanje može opisati kružnicom

konačnog radijusa i središta. Slika 49 dodatno pojašnjava ovu metodu planiranja putanja.



Slika 51: Kretanje robota u bilo kojem smjeru uz bilo kakvo zakrivljenje putanje. Zelenim linijama su pokazane putanje vrhova nogu

Sustav je implementiran u diskretnoj domeni što znači da se u svakom diskretnom trenutku noga pomiče po x i y osi prateći isplaniranu putanju za zadani referentni smjer i zakret kretanja robota. Pravilo po kojemu se svaka noga kreće u x-y ravnini dano su izrazima 1 i 2.

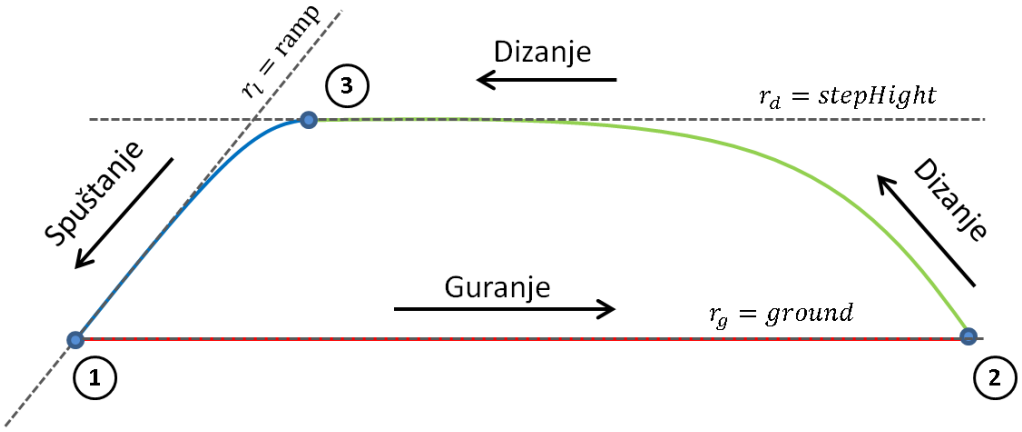
$$x_{k+1} = r\cos(\phi_k + \Delta\phi) + x_{mc} \tag{1}$$

$$y_{k+1} = r\sin(\phi_k + \Delta\phi) + y_{mc} \tag{2}$$

Gdje su r udaljenost od trenutnog vrha noge do središta kružnice gibanja, ϕ_k trenutni kut od središta kružnice gibanja do vrha noge, $\Delta\phi$ kutni pomak robota po toj kružnici a x_{mc} i x_{mc} koordinate središta kružnice gibanja. Lako se može primijetiti da se određivanjem središta kružnice gibanja može potpuno opisati željeno gibanje robota a mijenjanjem $\Delta\phi$-a možemo upravljati brzinom kretanja robota.

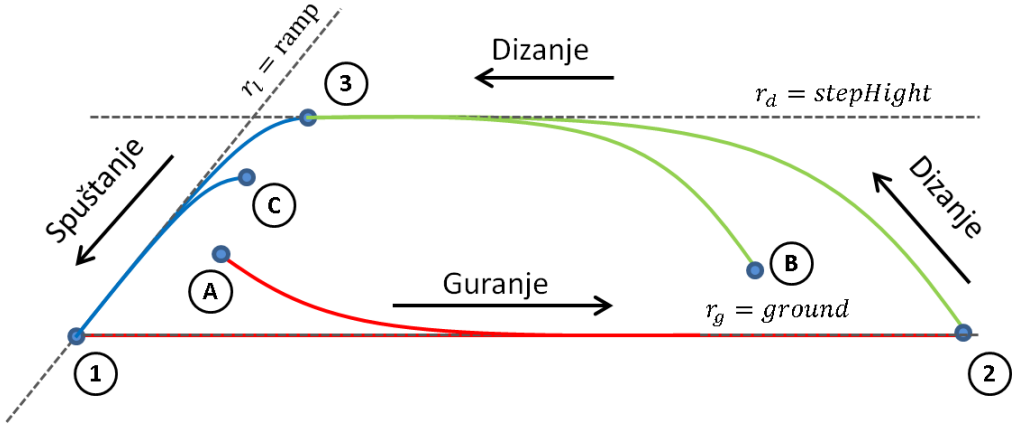
Jasno je da se robot neće kontinuirano kretati u željenom smjeru ako krećemo noge po isključivo ovim pravilima. Kretao bi se dok mu noge ne bi naišle na kraj radnog prostora. Sada je potrebno definirati korak. Korak će se uvijek kretati po putanji isplaniranoj na gore navedeni način (promatrano iz dvodimenzionalne perspektive x-y ravnine po kojoj robot hoda). Drugim riječima, projekcija trodimenzionalne putanje koraka na dvodimenzionalnu x-y plohu će uvijek biti ista za zadano kretanje robota kao što je pokazano na slikama 49, 50 i 51. Ono što će se mijenjati je smjer kretanja po isplaniranoj putanji te visina noge tj. njena z komponenta. U grubo, to se kretanje može podijeliti u 2 faze. Prva faza je kretanje noge po putanji u pozitivnom smjeru dok je noga na podu. Druga faza je kretanje po putanji u negativnom smjeru dok je noga u zraku. Ako se druga faza podijeli na fazu

dizanja noge i fazu spuštanja noge sveukupno se dobiju tri glavne faze koraka (Slika 52). Točka 1 predstavlja prijelaz iz faza kad je noga u zraku u fazu kada je noga prizemljena a obrnuto vrijedi za točku 2. Točka 3 je dodatna međutočka koja odvaja faze dizanja i spuštanja noge. Dodatno treba napomenuti da u ustaljenom stanju hodanja točke 1 i 2 predstavljaju rubove radnog prostora noge kako bi se radni prostor dobro iskoristio.



Slika 52: 3 glavne faze svakog koraka

Na slici 52 su dodatno isprekidano označene referentne vrijednosti visine noge u koraku tj. njegove z komponente. U prvoj fazi referentna visina noge je $z = 0$. U drugoj fazi referentna visina noge je $z = stepHight$ što ujedno označava i visinu svakog koraka. U trećoj fazi referentna vrijednost je promjenjiva s obzirom na udaljenost do kraja koraka. Odabrana je spuštajuća rampa koja treba završiti u $z = 0$. Praćenje ovih referenca implementirano je pomoću PT1 članova različitih vremenskih konstanti za svaku fazu koraka kako bi prijelazi bili glatki i bez nepotrebnih trzaja nogu. Korak se mogao dizajnirati i unaprijed isplaniranim trajektorijama ili polinomima koji bi isto predstavljali dobro rješenje do tada bi nastajali problemi s trzajima pri naglim promjenama referentnog smjera kretanja robota i slično. Na ovaj način se postiže da noga može biti zatečena u bilo kojem položaju u prostoru te će u ovisnosti o fazi u kojoj se nalazi samo z komponentom težiti u referentnu z vrijednost kao što je prikazano na slici 53 dok se x i y komponente ponašaju po pravilima 1 i 2.

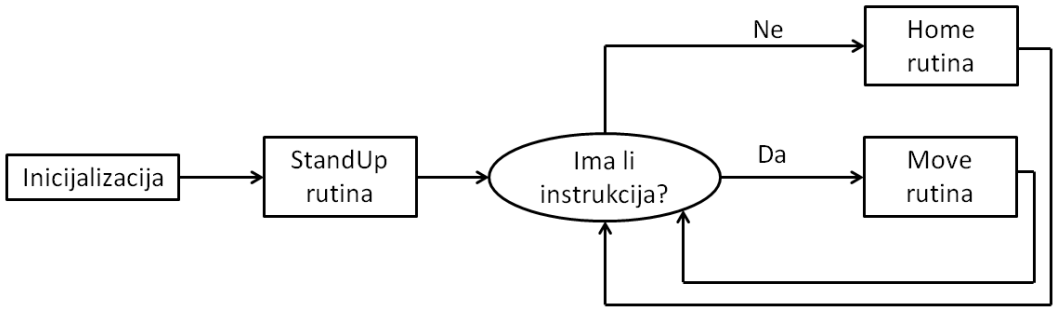


Slika 53: Prikaz težnji z komponente noge u referentnu vrijednost bez obzira na zatečeni položaj

Gore navedene metode su temelj jezgre hodanja. One su obuhvaćene u 3 glavne rutine kojima se

robot upravlja. Svaka od njih više ne tretira noge nezavisno već na robota gleda kao na cjelinu kako bi se u konačnici artikulirano kretao:

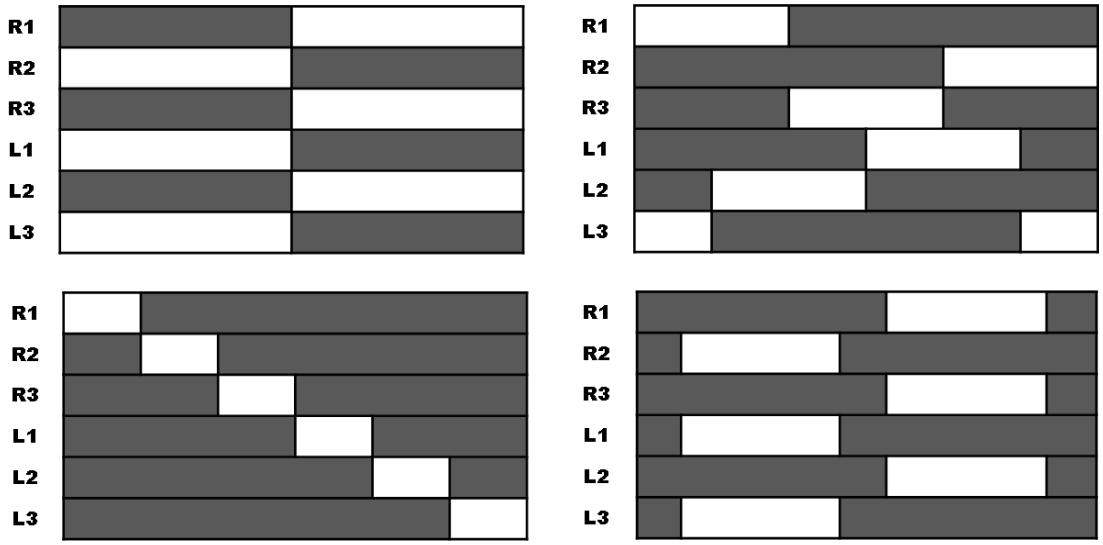
- *StandUp* rutina - Rutina ustajanja. Pri pokretanju algoritma, nakon uspostavljanja svih komunikacija robot se automatski ustaje. Nakon izvršenja ove rutine robot je spreman za korištenje.
- *Move* rutina - pokreće se isključivo ukoliko algoritam izvana prima instrukcije za kretanje. Vrši sve gore navedene metode algoritma upravljanja kako bi se u konačnici robot kretao u sladu sa zadanim instrukcijama.
- *Home* rutina - pokreće se uglavnom kada algoritam izvana ne prima instrukcije za kretanje te tada vrača robota u *home* poziciju kako bi bio spreman za nadolazeće instrukcija. Sadržajno gotovo jednaka *move* rutini no ne prima vanjske instrukcije o usmjerenju već uvijek robota stavlja u početnu poziciju.



Slika 54: Prikaz tijeka izvođenja rutina algoritma hodanja.

5.1.4. Gait

Nakon isplaniranih putanja potrebno je uskladiti faze izvršavanja koraka svake noge. Robot će se kvalitetno kretati samo ako su barem 3 noge prizemljene na podlogu a često je poželjno i više. To znači da se svaka noga ne može kretati nezavisno već je nužno poštivati određeni slijed izmjene koraka među nogama ili *gait*. *Gait* hodanja diktira kada bi koja noga trebala biti prizemljena a kada u zraku. *Gait* sljedovi su najčešće inspirirani prirodom pa su tako po uzoru na razne kukce definirani gaitovi dani slikom 55. Zatamljena područja govore kada bi noga trebala prizemljena "gurati" a u ostalim područjima bi noga trebala biti u zraku i pripremati se za nadolazeće područje "guranja" tj. putovati u zraku do točke gdje bi se ponovno trebala prizemljiti. I tako u krug.



Slika 55: Tripod gait (gore lijevo), ripple gait (gore desno), wave giat (dolje lijevo), modificirani tripod gait (dolje desno)

Na slici 55 su prikazani samo neki od mogućih *gait*-ova šesteronogog hodača. Za brže kretanje robota odabiru se *gait*-ovi s 3 ili 4 konstantno prizemljene noge dok se za sporije i sigurnije kretanje koriste *gait*-ovi s 5 ili 6 konstantno prizemljenih nogu. Tako se masa uvijek raspodjeljuje na više nogu te se smanjuje napor svake noge zasebno. Upravo ovo područje je vrlo zanimljivo i plodno za eksperimentiranje. Kao primjer modifikacije na slici su tripod i modificirani tripod *gait*. Problem kod običnog tripoda je što dolazi do velikog trzaja u hodu pri izmjeni funkcija svih nogu. Taj problem je riješen u modificiranom tripodu gdje se prvo sve noge prizemlje, neko vrijeme guraju sve noge skupa a tek nakon kratkog perioda se određene noge dižu u zrak. Tako se smanji udar zbog istovremenog dizanja prizemljenih nogu te spuštanja nogu koje su u zraku.

5.1.5. Upute za korištenje algoritma hodanja

Sve dosada objašnjene metode vezane za algoritam hodanja implementirane su u ROS okruženju u *C++* programskom jeziku. Osnovni *node* je *move_node*. U njemu su implementirani kinematika, radni prostor, *gait engine*, jezgra algoritma koja predstavlja planiranje putanja i koraka te još mnogo metoda koje izlaze iz okvira ovog rada. Programska podrška je organizirana tako da se odlikuje lakom nadogradivošću. Svaka noga robota ima svoju instancu *Leg* klase što znači da se trenutni kod lako može prilagoditi i hodačima s različitim brojem nogu. Jednostavno treba primijeniti broj instanca *Leg* klase unutar *Body* klase. Također se lako mogu implementirati novi *gait*-ovi minimalnim dodatcima *gait* klasi. Za sve informacije kako izvršiti nadogradnje preporučeno je pročitati detaljne komentare unutar koda. Za pokretanje *move_nodea* prvo je potrebno pokrenuti *roscore* sljedećom naredbom u terminalu:

```
$ roscore
```

move_node se zatim pokreće sljedećom naredbom u terminalu:

```
$ rosrun altera_hexapod move_node
```

Tako nastaju sljedeći *topici*:

- vrep_in_joint_angles_topic - ulazni podaci za vizualizaciju robota u simulaciji (izlazni podaci iz algoritma hodanja). Podržana poruka je tipa *joint_angles.msg* te se njome određuju zakreti svih zglobova u simulaciji.
- vrep_out_joint_angles_topic - izlazni podaci iz simulacije (ulazni podaci algoritma hodanja). Podržana poruka je tipa *joint_angles.msg* te ona govori o trenutnim zakretima zglobova u simulaciji.
- dynamixel_in_joint_angles_topic - isto kao i vrep_in_joint_angles_topic no predviđeno za zadavanje zakreta stvarnih motora.
- dynamixel_out_joint_angles_topic - isto kao i vrep_out_joint_angles_topic no predviđeno za čitanje zakreta stvarnih motora.
- foot_state_topic - *move_node* na ovaj *topic* objavljuje stanja svih nogu (koordinate vrhova nogu te faza u kojoj se noga nalazi). Podržana poruka je tipa *foot_state.msg*
- commands_topic - *move_node* preko ovog *topica* prima instrukcije za kretanje (više o tome u nastavku).

Sljedeći važan *node* je *commands node* koji u suradnji s *joy node*-om i *driverom* za upravljač šalje instrukcije *move nodeu* te se na taj način upravlja kretanjem robota. No prije je potrebno upariti upravljač s *bluetooth*modulom. To se vrši sljedećom naredbom:

```
$ sixpair
```

Nakon uspješno uparivanja upravljača i *bluetooth* modula potrebno je pokrenuti sljedeće naredbe u zasebnim terminalima:

```
$ sixad -s #pokretanje drivera za ps3 upravljac
$ rosrun joy joy_node #povezivanje upravljaca i ros-a
$ rosrun altera_hexapod commands_node #prilagodba podataka s upravljaca te slanje
    instrukcija na odgovarajuce topic-e
```

Tako nastaju sljedeći *topici*:

- joy - rezultat pokretanja *joy_nodea*. ROS-ov standardni *topic* za objavljivanje statusa generičkih upravljača.
- commands_topic - rezultat pokretanja *commands_nodea* koji se pretplaćuje na *joy topic* te obrađuje sirove podatke s upravljača kako bi u *commands_topic* poslao instrukcije za kretanje robota. Podržana poruka je tipa *commands.msg*

Na upravljaču je sada potrebno pritisnuti *PS* tipkalo kako bi se povezao sa *bluetooth* modulom. Jednom kada se svi koraci inicijalizacije izvrše upravljačem se upravlja gibanjem robota kao što je prikazano na slici

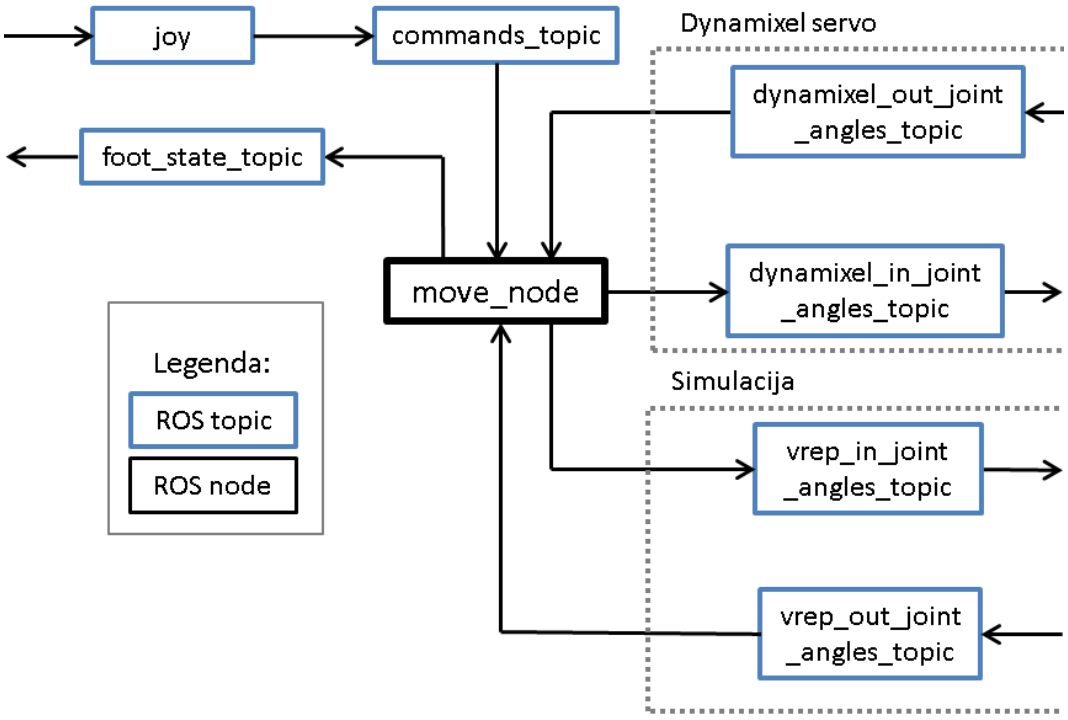
Slika 56: Objašnjenje uloga tipkala i palica upravljača u algoritmu hodanja

U ovom trenutku je algoritam spreman za korištenje. *move_node* sada čeka da uspostavi komunikaciju sa simulatorom ili stvarnim aktuatorima preko odgovarajućih *topica*. U trenutku uspostavljanja komunikacije automatski se pokreće rutina ustajanja te je robot spreman za gibanje.

Gore opisani postupak se može obaviti i korištenjem *roslaunch* alata. On služi za pokretanje više *nodeova* odjednom uz dodatnu mogućnost podešavanja parametara. Tako se pokretanjem sljedeće linije pokreću *move_node*, *joy_node* te *commands_node*:

```
$ roslaunch altera_hexapod start.launch
```

Uređivanjem *start.launch* datoteke se mogu mijenjati mnogi parametri algoritma hodanja. Detaljna objašnjenja tih parametara se nalaze u komentarima koda.



Slika 57: Shema organizacije *topica* i *nodeova* algoritma hodanja unutar ROS okruženja

5.2. Simultana lokalizacija i mapiranje

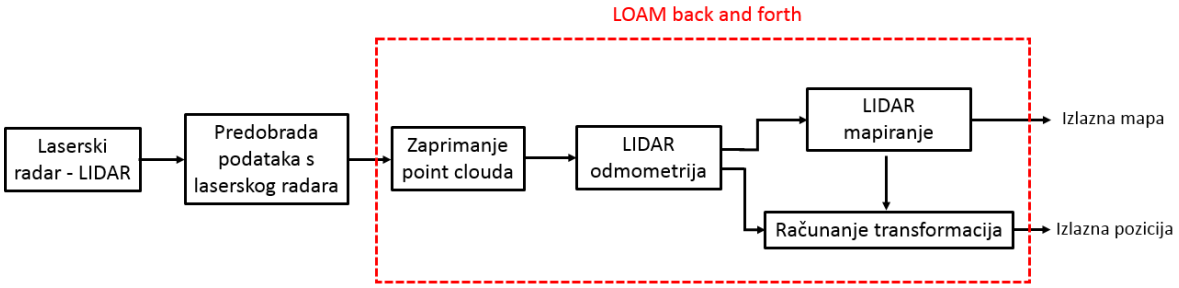
Problem simultane lokalizacije i mapiranja (eng *Simultaneous Localization And Mapping*), u daljnjem tekstu SLAM, ispituje je li moguće mobilnom robotu koji je stavljen u nepoznati teren na nepoznatu lokaciju postepeno izgradi mapu prostora dok se simultano pozicionira unutar nje. Rješenje ovoga problema pruža robotu da uistinu postane autonoman i smatra se jednim od najtežih problema u svijetu mobilne robotike [4].

SLAM je proces u kojemu mobilni robot može izgraditi mapu okoline koja ga okružuje dok u isto vrijeme koristi tu mapu kako bi odredio svoju poziciju unutar nje. Unutar SLAM algoritma trajektorija i pozicija svih objekata okoline estimiraju se online bez potrebe za bilo kakvim a priori znanjem o lokaciji platforme ili okolini [4]. Kako bi izgradnja mape bila moguća potreban je senzor koji algoritmu daje informacije o okolini. Najčešće korišteni senzori su sonar i rotirajući laserski radar.

Veliki dio nedavnog razvoja SLAM algoritama koncentrirao se na smanjenje procesorske snage potrebne za izvršavanje algoritma i očuvanje točnosti koju algoritam pruža [5]. Uz sve brži razvoj tehnologije, senzori potrebni za implementaciju SLAM algoritma postaju sve manji i kvalitetniji kao i upravljačke jedinice na kojima se oni izvode. Mapiranje rotirajućim laserskim radarom danas se smatra uobičajenim postupkom, čak i za manje mobilne robote kao što je razvojna platforma opisana u ovome radu.

5.2.1. LOAM back and forth

SLAM algoritam korišten na razvojnoj platformi je *LOAM back and forth* [31]. Glavni razlog korištenja ovog algoritma je njegova kompatibilnost s ROS operativnim sistemom te mogućnosti pokretanja na ugradbenom računalu. Uspoređujući s ostalim javno dostupnim algoritmima *LOAM back and forth* pokazao se kao najbolja opcija za platformu.

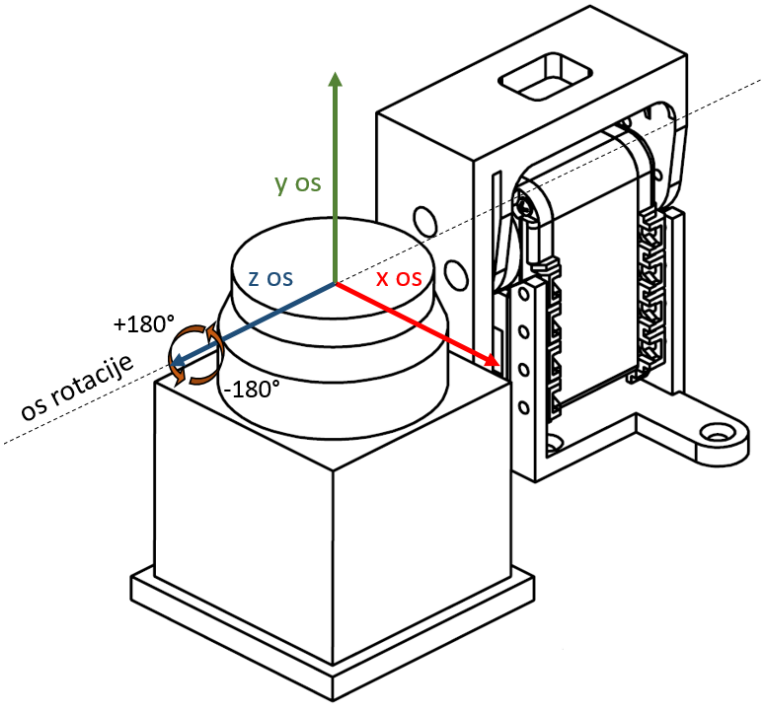


Slika 58: Blokovska shema LOAM back and forth algoritma

Slika 58 prikazuje blokovsku shemu načina rada LOAM back and forth algoritma. Crveno označeni pravokutnik na slici prikazuje način rada spomenutog algoritma. Algoritam postiže malu količinu drifta i zahtjeva nisku razni kompjuterske snage bez potrebe za visoko preciznim mjerenjima ili dodatnim unutarnjim mjerenjima. Osnova ideja koja je omogućila takve performanse je podjela tipično kompleksnog problema simultane lokalizacije i mapiranja u dva manja algoritma. Prvi algoritam izvodi odometriju s viskom frekvencijom ali malom pouzdanošću estimacije kretanja. Drugi se algoritam izvodi na nižoj frekvenciji od prvoga i služi za precizno spajanje i primanje ulaznog

point clouda[20] [32]. Oba algoritma izvlače točke u prostoru locirane na oštrim rubovima ili ravnim dijelovima objekata kako bih kasnije pridružili ostalim točkama koje čine mapu. Kao dodatnu opciju algoritam nudi korištenje inercijalnog mjernog sustava za bolju estimaciju brzih kretnja. Na razvojnoj platformi inercijalni mjerni sustav nije korišten u sklopu *LOAM back and forth* algoritma. Planira se njegovo korištenje u daljem razvoju platforme.

Ulazni podatak u algoritam je rotirajući *point cloud* koji rotira oko z osi statičnog koordinatnog sustava laserskog radara. Taj se koordinatni sustav nalazi u polazištu svake laserske zrake i miruje s obzirom na radar, a "putuje" s njim s obzirom na dobivenu mapu. Željena orijentacija koordinatnog sustava vidljiva je na Slici 59. Izlazni podaci iz SLAM algoritma su dobivena *point cloud* mapa i transformacija iz ishodišnog koordinatnog sustava u trenutni koordinatni sustav. Kako se robot giba tako se dobivena mapa puni sa sve većim brojem podataka a transformacija pozicionira koordinatni sustav robota unutar dobivene mape.



Slika 59: Prikaz rotacije laserskog radara za $\pm 180°$ oko z-osi

5.2.2. Rotacija radara i predobrada podataka

Najveći izazov u implementaciji SLAM algoritma na razvojnu platformu bio je dobivanje rotirajućeg rotirajući *point clouda* kao ulazne veličine u algoritam. Kako bi se on mogao dobiti bila je potrebna rotacija radara kao i predobrada dobivenih podataka s njega. Na blokovskoj shemi (Slika 58) rotirajući radar predstavlja prvi bok s lijeve strane pod imenom "Laserski radar - LIDAR" dok predobradu predstavlja drugi blok s lijeve strane pod nazivom "Predobrada podataka s laserskog radara".

Radar korišten na razvojnoj platformi za potrebe algoritma je Hokuyo URG-04LX. Kako je već spomenuto u cjelini 3.3 i 4.3.1 tijelo radara pričvršćeno je za servo motor (Dynamixel AX-12A)

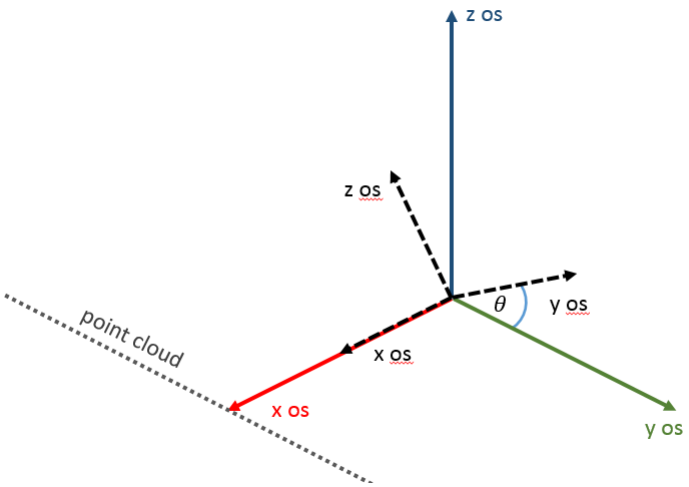
[20] Point cloud je digitalni zapis skupa točaka unutar jednog koordinatnog sustava.

koji rotira radar oko jedne osi u oba smjera. Takva postava lasera i servo motora uobičajena je za SLAM algoritme. Rotacija radara je potreba iz razloga što on daje informacije o udaljenosti unutar dvodimenzionalnog radnog prostora. Ako se želi napraviti trodimenzionalna mapa prostora potrebna je rotacija radnog prostora. Kod nekih algoritama potrebno je radar rotirati kontinuirano za puno krug (360°) oko osi rotacije, drugi algoritmi zahtijevaju rotaciju oko više osi itd. *LOAM back and forth* algoritam zahtjeva rotaciju radara za $\pm 180°$ oko jedne osi. Zbog slabije kvalitete laserskog radara brzina okretanja radara postavljena je na 0.125 okretaja u minuti. Kvalitetniji laseri postiži i do četiri puta veće brzine okretanja. Frekvencija skeniranja lasera je 10 Hz. Iz ova dva podatka dobiva se kutna razlučivost mape koja iznosi $\frac{360°}{8*10} = 4.5°$. Ukoliko bi se korišteni radar rotirao većom brzinom gubila bi se kutna razlučivost dobivene mape zbog sve manjeg broja skeniranja unutar jednog okretaja za $\pm 180°$.

Glavni cilj predobrade je pretvorba dobivenih podataka s radara u podatke koji odgovaraju *LOAM back and forth* algoritmu. Pod time se podrazumijeva pretvorba iz statičnog *laser scan*[21] paketa podataka u rotirajući *point cloud* pritom vodeći računa o orijentaciji koordinatnih sustava (sukladno onome što algoritam zahtjeva).

Prvi korak u predobradi bila je uspostava komunikacije između Hokuyo lasera i ROS operativnog sustava. Za komunikaciju je zadužen programski paket *Hokuyo node*, koji podatke s laserskog radara pakira u *laser scan* paket podataka [3] te ih objavljuje na ROS *topicu* pod nazivom */scan*. Ovako dobiveni paketi standardni su za rad unutar ROS-a. Iz toga razloga lako se mogu koristiti u druge svrhe ako se za to pokaže potreba. Upute za uspostavu komunikacije mogu se pronaći u cjelini 5.2.3.

Nakon uspostavljene komunikacije podaci s lasera obrađuju se unutar za to napisanog *node*-a pod nazivom *laserscan_to_pointcloud*. Node prisluškuje *topic /scan* te nakon svakog pristizanja poruke pokreće obradu iste. U sklopu *nodea* korištena su dva programska paketa pod nazivom *Laser geometry* [26] i *Tf2* [27]. Programski paket *Tf2* služi kako bi se napravila željena transformacija iz rotirajućeg koordinatnog sustava laserskog radara u stacionarni koordinatni sustav radara. Programski paket *Laser geometry* pretvara poruke tipa *laser scan* u *point cloud* tip poruka koristeći dobivenu transformaciju od *Tf2*.



Slika 60: Stacionarni i rotirajući koordinatni sustav laserskog radara

[21] Laser scan je paketa podataka karakterističan za laserske radare korištene u ROS-u.

Na Slici 60 najbolje se vidi statični koordinatni sustav laserskog radara oko čije x osi rotira rotirajući koordinatni sustav radara. Kut rotacije θ jednak je zaokretu servo motora koji rotira radar. Informaciju o zaokretu *node laserscan_to_pointcloud* dobiva s *topica /servo_position*. Nakon definirane transformacije iz rotirajućeg u statični koordinatni sustav, *Laser geometry* transformira *laser scan* u *point cloud* tip poruka. Primjer dobivenog *point clouda* također se vidi na Slici 60.

Uspoređujući orijentaciju koordinatnih sustava na Slici 60 i 59 vidimo kako se one ne podudaraju. Iz toga razloga potreban je zadnji korak predobrade, rotacija dobivenog *point clouda* kako bi se on mogao koristiti u *LOAM back and forth algoritmu*. Rotacija je složena, iznosi -90° oko x i y osi te se izvodila upotrebom matrica rotacija R_x i R_y.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} R_y = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Kada je rotacija napravljena dobiveni se *point cloud* može poslati *LOAM back nad forth algoritmu* na obradu.

5.2.3. Upute za korištenje radara i SLAM algoritma

Kako bi korisnik mogao uspostaviti komunikaciju između ROS-a i Hokuyo laserskog radara, uz uvjet da je prethodno spojen putem SSH protokola na glavnu upravljačku jedinicu Odroid, potrebno je pratiti sljedeće korake:

1. Otvaranje novog terminala na glavnoj upravljačkoj jedinici Odroid
2. Pokretanje ROS *launch* datoteke za uspostavu komunikacije, upisivanjem sljedeće naredbe u prethodno otvorenom terminalu:

```
$ roslauch laserscan_to_pointcloud hokuyo_node_start.launch
```

3. Za provjeru rada korisnik može otvoriti programski paket Rviz te učitati konfiguracijsku datoteku *laserscan.config*. Prikazani podatci su ROS poruke s *topica /scan*. Rviz se pokreće na vlastitome računalu.

Ako su svi koraci izvedeni ispravno, komunikacije između ROS-a i Hokuyo radara trebali bi biti uspostavljena putem USB komunikacije na portu */dev/ttyACM1*. Podatci dobiveni s radara objavljuju se na *topicu /scan*, te su pakirani u ROS poruke oblika *sensor_msgs/LaserScan*. Shematski prikaz ROS komunikacije laserskog rada vidi se na Slici 61.

Za korištenje SLAM algoritma nešto je dulji postupak nego kod korištenja radara. Uz uspostavljenu komunikaciju s laserskim radarom, prate se koraci:

1. Otvaranje novog terminala na glavnoj upravljačkoj jedinici Odroid
2. Pokretanje ROS *nodea* za početak komunikacije s servo motorima, upisivanjem sljedeće naredbe u prethodno otvorenom terminalu:

```
$ rosrun altera_hardware servo_control.py
```

3. Otvaranje novog terminala na glavnoj upravljačkoj jedinici Odroid

4. Pokretanje ROS *nodea* za početak obrade *laser scana*, upisivanjem sljedeće naredbe u prethodno otvorenom terminalu:

```
$ rosrun laserscan_to_pointcloud laserscan_to_pointcloud
```

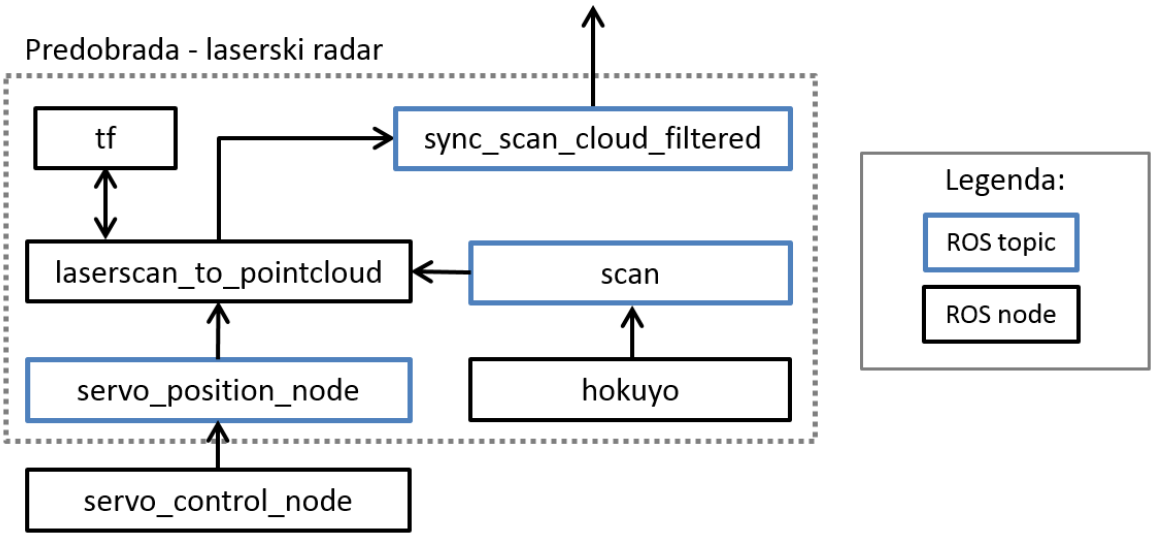
5. Otvaranje novog terminala na glavnoj upravljačkoj jedinici Odroid

6. Pokretanje ROS *launch* datoteke za početak rada *LOAM back and forth* algoritma:

```
$ roslaunch loam_back_and_forth loam_back_and_forth.launch
```

7. Dobivena mapa prostora zajedno s pozicijom razvojne platforme unutar nje može se pregledati unutar programskog paketa Rviz, učitavanjem konfiguracijske datoteke *loam_back_and_forth.config*. Rviz se pokreće na vlastitome računalu.

Ako su svi koraci dobro izvedeni, unutar programskog paketa Rviz trebala bi se pojaviti mapa prostora kojeg radar očitava zajedno s koordinatni sustavom radara koji predstavlja njegovu poziciju unutar mape. Shematski prikaz ROS komunikacije algoritma velik je i nespretan za prikazivanje pa se na Sllici 61 može vidjeti prikaz ROS komunikacije predobrade. Izlazni podatci s *topica sync_scan_cloud_filterd* su ulazni podatci za algoritam.



Slika 61: Shematski prikaz ROS komunikacije predobrade podataka s laserskog radara

5.3. Inercijalni mjerni sustav

U sklopu sporedne upravljačke jedinice nalazi se inercijalni mjerni sustav. Zadaća ovakvog sustava je pružanje informacija razvojnoj platformi o njenoj orijentaciji, kutnoj brzini i linearnoj akceleraciji. Te se informacija kasnije mogu primjenjivati u raznim algoritmima hodanja, stabilizacije, mapiranja i ostalog.

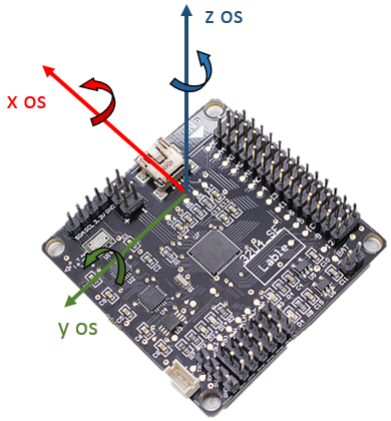
Sporedna upravljačka jedinica programira se u Arduino programskom okruženju. Unutar okruženja korištena je *FreeIMU* biblioteka [29] za dobivanje potrebnih informacija iz inercijalnog mjernog sustava. Kao i sve ostale komponente na razvojnoj platformi potrebna joj je komunikacija s glavnom upravljačkom jedinicom i ROS operativnim sustavom. Komunikacija se odvija preko serijske USB komunikacije a programski paket zadužen za komunikaciju s ROS-om je *Rosserial* [2]. ROS poruke koje se šalju sa sporedne upravljačke jedinice su *Float64* tipa te se mogu rasporediti od tri skupine:

1. Orijentacijski kvaternion[22] - orijentacija u prostoru prikazana je pomoću kvaterniona. Kvaternioni se sastoji od svog realnog (w) i kompleksnog (i, j, k) dijela i koriste se za prikazivanje orijentacije zbog svoje jednostavnosti korištenja

2. Kutna brzina - mjeri se oko tri osi sustava x, y i z. Mjerna jedinica za kutnu brzinu je rad/sec

3. Linearna akceleracija - kao i kutna brzina, linearna se akceleracija mjeri u sve tri osi sustava. Mjerna jedinica je m/s^2

Dobivene poruke putem *Rosserial*-a, potrebno je obraditi i pakirati u *sensor_msgs/Imu* tip poruka. Pod pojmom obrada misli se na promjenu orijentacije dobivenih podataka. Obrada se odvija uz pomoć za to napisanog *nodea imu_node*. Referentna orijentacija kvaterniona prikazana je na Slici 62. Kutna brzina smatra se pozitivnom ukoliko se sustav rotira u smjeru naznačenom na Slici 62. Linearna akceleracija je također orijentirana kao i koordinatni sustav kvaterniona. Kada je obrada završena, dobiveni rezultati objavljuju se na *topic /IMU_topic*. Ovako pakirani podaci s inercijalnog mjernog sustava uobičajeni su za ROS operativni sustav te lako mogu koristiti u neke druge namjene ako se za to pokaže potreba.



Slika 62: Orijenacija koordinatnih sustava inercijalnog mjernog sustava

[22] U matematici, kvaternioni su algebarsko proširenje kompleksnih brojeva.

5.3.1. Upute za korištenje inercijalnog mjernog sustava

Za uspostavu komunikacije između ROS-a i inercijalnog mjernog sustava, uz uvjet da je prethodno korisnik spojen putem SSH protokola na glavnu upravljačku jedinicu Odroid, potrebno je pratiti sljedeće korake:

1. Otvaranje novog terminala na glavnoj upravljačkoj jedinici Odroid
2. Pokretanje ROS *nodea* za uspostavu komunikacije, upisivanjem sljedeće naredbe u prethodno otvorenom terminalu:

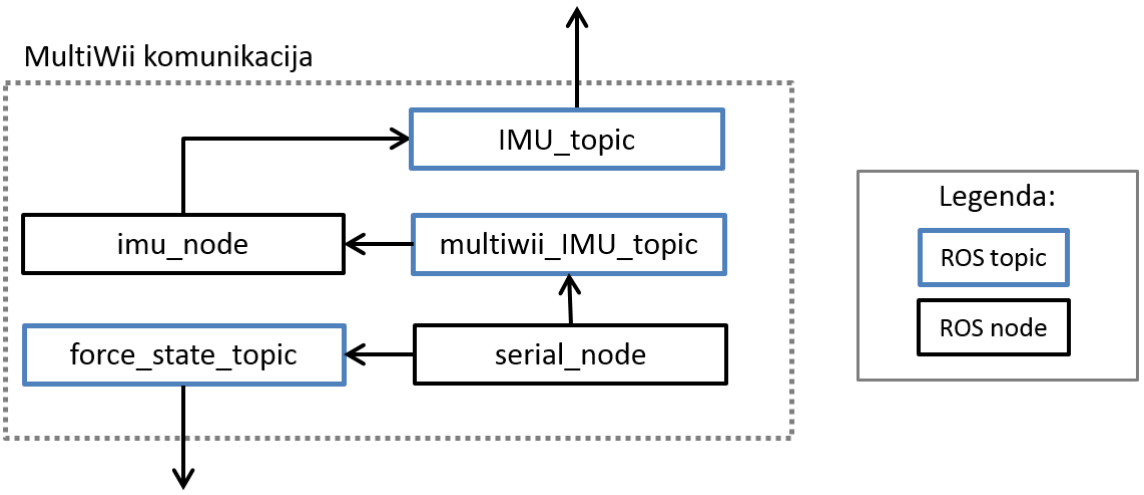
```
$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM2
```

3. Otvaranje novog terminala na glavnoj upravljačkoj jedinici Odroid
4. Pokretanje ROS *nodea* za obradu dobivenih podataka, upisivanjem sljedeće naredbe u prethodno otvorenom terminalu:

```
$ rosrun imu_pkg imu_node
```

5. Prostorna orijentacija inercijalnog mjernog sustava može se prikazati unutar programskog paketa Rviz. Potrebno je učitati konfiguracijsku datoteku pod nazivom *imu.config*. Rviz se pokreće na vlastitome računalu.

Ako su svi koraci pravilo izvedeni trebala bi se uspostaviti komunikacija između ROS operativnog sustava i inercijalnog mjernog sustava. Shematski prikaz komunikacije može se vidjeti na Slici 63.



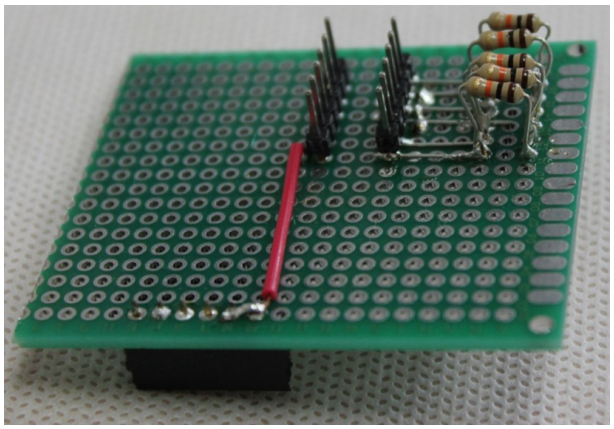
Slika 63: Shematski prikaz komunikacije između ROS-a i inercijalnog mjernog sustava

5.4. Senzori sile

Robotska platforma svojom masom opterećuje alat svake noge određenom silom. Zbog fizičkih nepravilnosti platforme i terena kojom ona hoda te sile nisu jednake. Nejednake sile mogu dovesti do preopterećenja električnih aktuatora koje uzrokuje neželjeno ponašanje. Kako bi se razvili algoritmi koji će rješavati taj i slične probleme potrebni su senzori sile, u ovom slučaju njih šest (jedan na vrhu alata svake noge).

5.4.1. Tiskana pločica

Senzor sile radi na principu promjenjivog otpornika kojem se mjeri napon kruga, kako je i opisano u poglavlju 4.3.2. Sporedna upravljačka jedinica MultiWii osigurava 5V i na svojim analognim pinovima mjeri napon kruga senzora sile. Kako proporcionalno povećanju sile otpor senzora teži u 0, potrebno je osigurati izvor od struje kratkog spoja. Tiskana pločica [23] je odlično rješenje za takve potrebe, tako da je i za ovu situaciju napravljena jedna, prikazana Slikom 64.



Slika 64: Tiskana pločica

Pločica je namijenjena da se stavi kao štit sporednoj upravljačkoj jedinici, čime su izbjegnuta dodatna ožičenja. Na njoj se se nalazi svih šest strujnih krugova senzora sile, jedan za svaku nogu. Kao zaštita od struje kratkog spoja se koriste otpornici veličine 10kohm. Slika 65 prikazuje shemu strujnog kruga senzora sile.

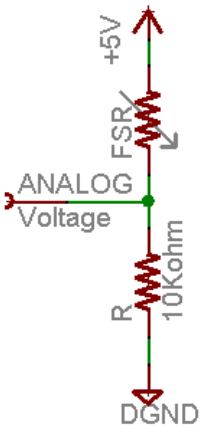
5.4.2. Upute za korištenje senzora sile

Sporedna upravljačka jedinica MultiWii spojena je na glavnu Odroid, tako da je prvi korak spojiti se na Odriod SSH protokolom. Dodatno treba provjeriti dali *roscore* radi kako bi se *serial_node* mogao spojiti s njim pri pokretanju. *serial_node* pokreće se upisivanjem sljedeće naredbe u terminal:

```
$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM2
```

Valja naglasiti kako je *serial_node* također odgovoran za dohvat podataka s inercijalnog mjernog sustava, tako da ga je dovoljno pokrenuti samo u jednu svrhu.

[23] Tiskana pločica (PCB) naziv je za sredstvo kojim se mehanički i električki povezuju elektroničke komponente - link `http://hr.wikipedia.org/wiki/Tiskana_plo%C4%8Dica`



Slika 65: Shema strujnog kruga senzora sile [1]

Dobiveni podaci sa senzora sile su cijeli brojevi u rasponu od 0-1024 koji odgovaraju naponu strujnog kruga V_0 senzora sile (0 predstavlja 0V dok 1024 predstavlja 5V). Vrijednost sile može se dobiti korištenjem grafa na Slici 21, gdje je potrebno iz dobivenog napona prvo izračuna otpor senzora sile R_{FSR} formulom 3 te očitati mjerenu silu s grafa.

$$\begin{aligned} V_{cc} &= 5V \\ R &= 10000ohm \\ R_{FSR} &= \frac{R}{V_0}(V_{cc} - V_0) \end{aligned} \tag{3}$$

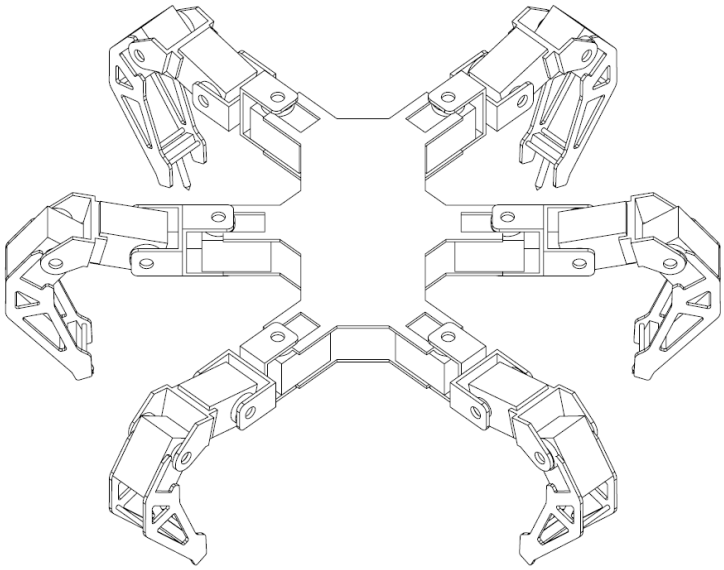
5.5. Robotski simulator

Robotski sustav sastoji se softverskog i hardverskog dijela koji prvo moraju biti osmišljeni, napravljeni, testirani i u konačnici stavljeni u upotrebu. Proces testiranja na stvarnoj opremi često je vremenski i financijski zahtjevan jer se svaka greška skupo plaća. Rješenje takve situacije je preseliti se iz stvarnog svijeta u virtualni, onaj koji se nalazi na računalu. Cilj virtualnog svijeta je da bude što sličniji ovome u kojem živimo, da postoje strogo definirana pravila i odnosi objekata te njihovih atributa. Upravo se taj virtualni svijet naziva simulatorom, u ovom slučaju konkretno robotskim simulatorom.

5.5.1. Izgradnja modela

Simulator nema svrhu ako ne postoje objekti koje će se u njemu simulirati. Objekte se može samostalno modelirati ili skinuti neki od postojećih modela s Interneta. Pošto je ova razvojna platforma robotski sustav jedinstvenog izgleda i proporcija, napravljen je vlastit model.

Izgradnja modela počinje dizajnom pojednostavljenog modela u SolidWorksu. Kod simulatora nije bitno detaljno modelirati objekte, već modelirati one atribute koji su najbitniji za njihov vizualni identitet. Tako se, na primjer, ne će modelirati svaka rupa na Dynamixel aktuatoru već će se on predstaviti kao kvadar. Pojednostavljeni model razvojne platforme prikazan je Slikom 66.



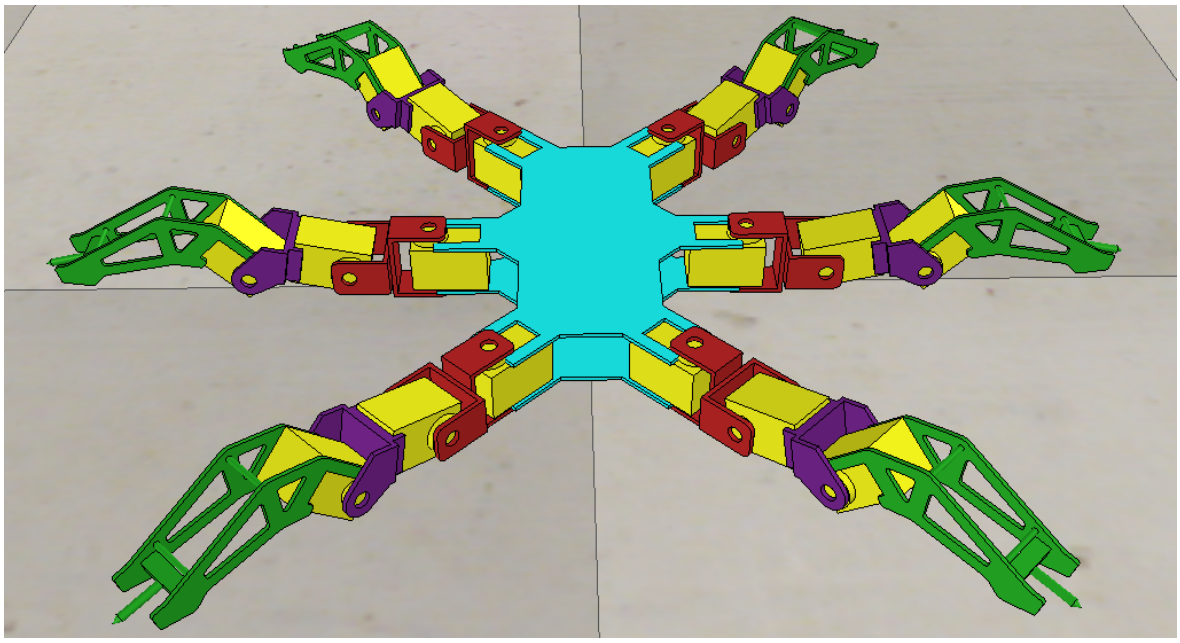
Slika 66: Pojednostavljeni model razvojne platforme

Tako napravljen model se prebacuje u v-rep robotski simulator. Unutar njega napravljeni model za sad ne posjeduje ni jedno svojstvo, on je bezdimenzionalan. Kako bi se razumio sljedeći korak potrebno je ukratko objasniti rad v-rep simulatora. U njemu postoje dva modela koji predstavljaju jedan objekt:

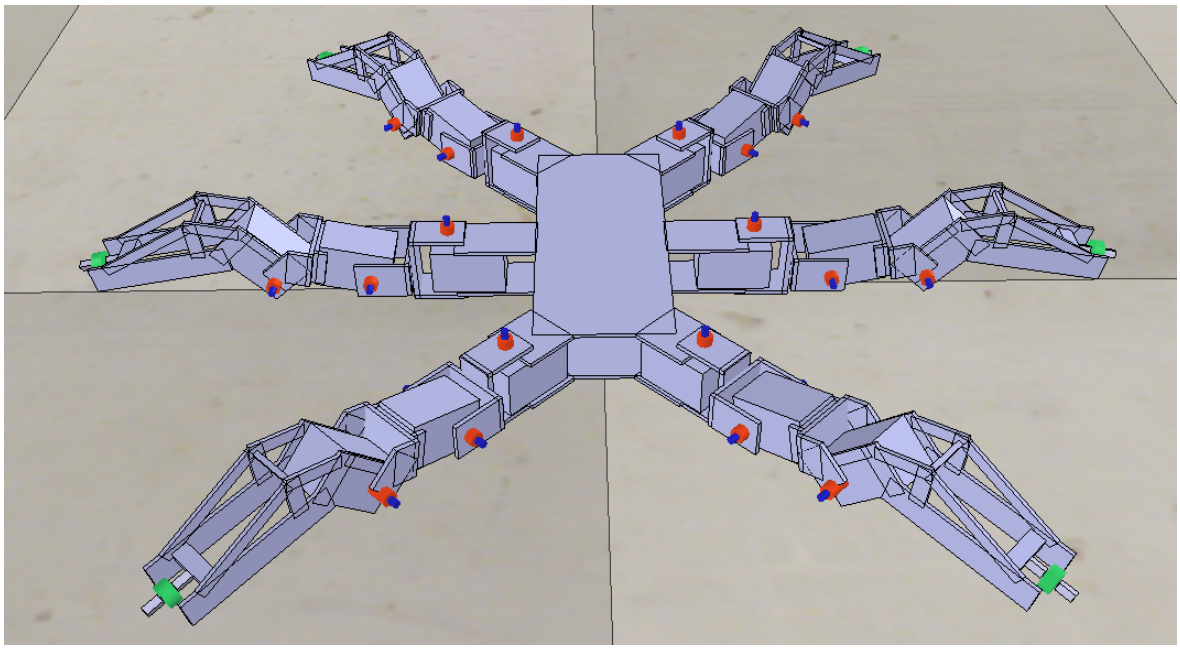
- Dinamički simulirani objekti - Predstavljaju objekte na kojima se odvijaju svi proračuni simulatora. Oni imaju definirane svoje dimenzije, masu, gustoću itd. Pojednostavljeno se može reći da se oni ponašaju kao u stvarnom svijetu.

- Statični objekti - Predstavlja objekte suprotne dinamički simulirani objektima. Oni su bezdimenzionalan te se većinom koriste samo kao lijepo "lice" dinamičkim objektima.

Trenutni model će predstavljati statički model. Razlog tome je to što je on još uvijek previše kompleksan za simulator i dinamičku simulaciju. Sljedeći korak je logičan, potrebno je napraviti dinamički model na temelju ovog pojednostavljenom modela. Njemu se točno određuju potrebni dinamički parametri (masa, centar mase, proporcije, itd.) Statični model je prikazan Slikom 67, dok je dinamički model prikazan Slikom 68. Na dinamičkom modelu se mogu primijetiti zglobovi (narančaste boje) te senzori sile (zelena boje) koji su modelirani po specifikacijama stvarnih komponenti.



Slika 67: Statični model robotske platforme unutar v-rep simulatora



Slika 68: Dinamički model robotske platforme unutar v-rep simulatora

5.5.2. Upute za korištenje simulatora

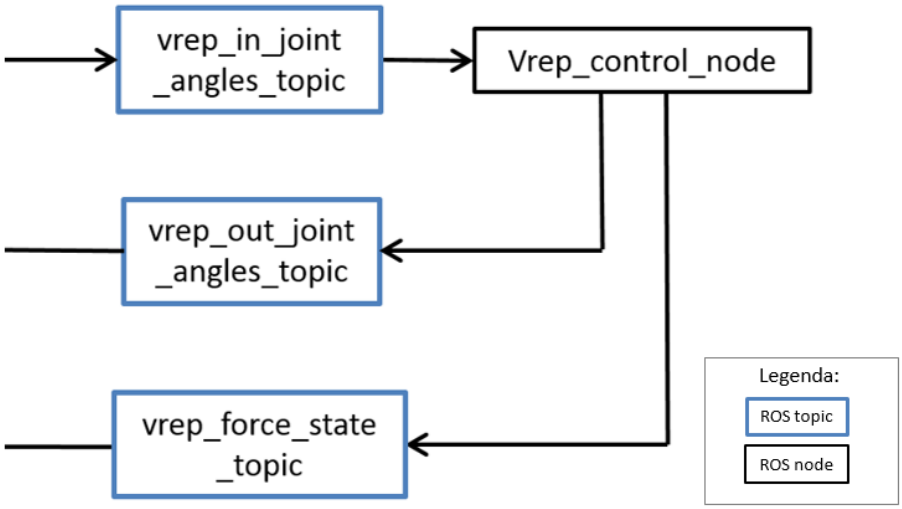
Napravljeno simulacijsko okruženje predviđeno je da se koristi na Linux operativnom sustavu, tako da je prvi korak instalacija v-rep simulatora na neku od Linux distribucija. Instalaciju je najbolje napraviti prateći upute sa službene v-rep stranice [19]. Nakon uspješne instalacije, potrebno je provjeriti dali *roscore* radi kako bi se vrep mogao spojiti s njim pri pokretanju. Vrep se pokreće pozicioniranjem u instalacijsku mapu te pozivanjem naredbe:

```
$ ./vrep.sh
```

Unutar simulatora potrebno je otvoriti napravljeno simulacijsko okruženje pod imenom *Altera_hexapod.ttt*. Korišten simulator fizike je ODE[24], postavka simulatora *Very Accurate* te vrijeme uzorkovanja 33,3 ms. Promjena navedenih parametara uvelike može utjecati na rad simulatora tako da je preporučljivo ne mijenjati ih.

Nakon što je modela ubačen u simulaciju te njeni parametri ispravno postavljeni, potrebno je omogućiti komunikaciju simulatora s drugim metodama razvojne platforme. U tu svrhu razvijen je ROS *node* pod nazivom *vrep_controle_node*. Njegova uloga je prenošenje podataka u i iz simulacije, što omogućuje potpunu kontrolu i dobivanje potrebnih podataka. Shema ROS povezanosti prikazana je Slikom 69. Pri razvoju *nodea* od velike pomoći je bio službeni V-rep User Manual [18] te jedna domaća zadaća s University of Edinburgh fakulteta [17]. Ulazni parametar za pozivanje *nodea* je samo IPv4 adresa [25] računala na kojem je otvorena simulacija. Ako je konekcija uspostavljena terminal će ispisati *ClientID*. *Node* se poziva sljedećom naredbom u terminalu:

```
$ rosrun vrep_controle_node vrep_controle "IPv4adresa"
```



Slika 69: Shema ROS povezanosti *vrep_controle_nodea*

[24] Open Dynamic engine (ODE) je biblioteka otvorenog tipa za simulaciju dinamike krutih tijela - link`http://www.ode.org/`

[25] Internet Protocol (IP) adresa je numerička oznaka svakog uređaja spojenog na Internet - link`http://hr.wikipedia.org/wiki/IP_broj`

Dodatno valja objasniti da se pokretanjem *vrep_controle_nodea* stvaraju sljedeći *topici*:

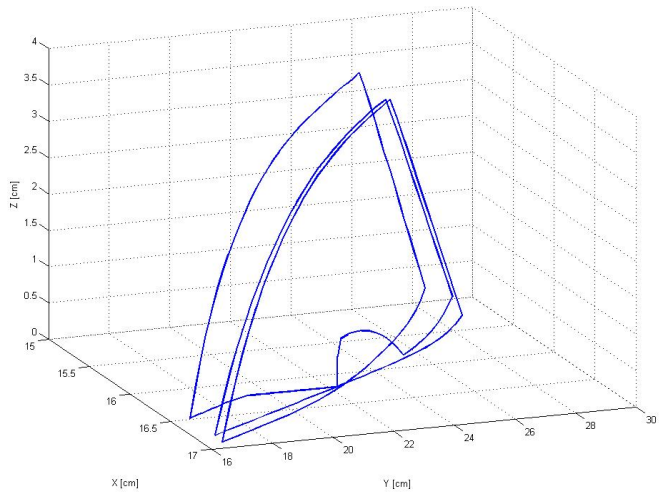
- *vrep_in_joint_angles_topic* - ulazni podaci algoritma hoda za simulator koji govore željenu poziciju električnih aktuatora. Podržana poruka je *joint_angles.msg*
- *vrep_force_state_topic* - izlazni podaci sa senzora sile simulatora koji su pakirani tako da simuliraju rad stvarnih senzora sile. Podržana poruka je *force_sensor.msg*
- *vrep_out_joint_angles_topic* - izlazni podaci položaja električnih aktuatora simulatora. Podržana poruka je *joint_angles.msg*

6. Rezultati

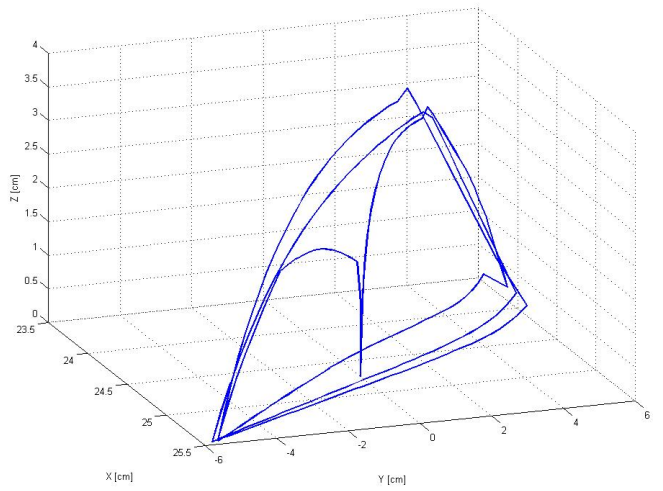
Rad i funkcionalnost svih metoda spomenutih u poglavlju 5. testiran je na stvarnom sustavu te su prikazani dobiveni rezultati.

6.1. Algritam hodanja

Kao rezultat planiranja koraka, na slikama 70 i 71 se jasno može vidjeti trodimenzionalna putanja noge R1 i R2 pri hodu robota. Kako su upravljačke veličine zadavane ručno preko palica PS3 upravljača, one nisu savršene pa tako i putanja nogu prati pomake upravljačke palice u stvarnom vremenu. Zato se putanje svakog vremenski susjednog koraka umalo razlikuju.

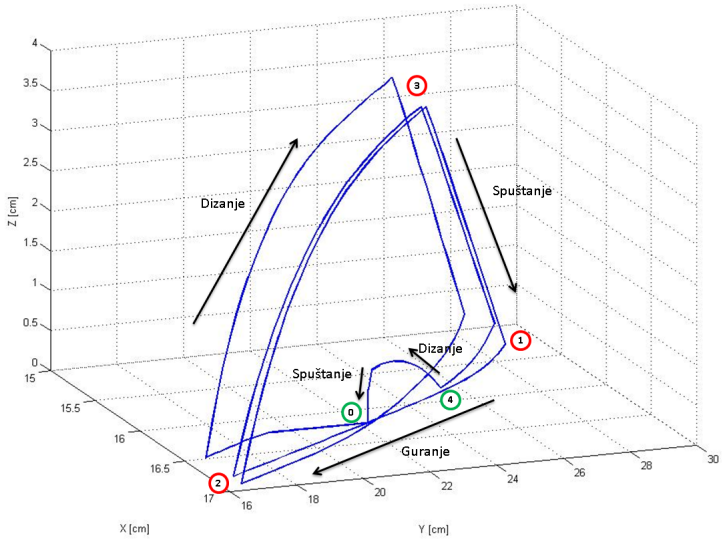


Slika 70: Snimljena putanja noge R1 u trodimenzionalnom prostoru pri hodanju robota

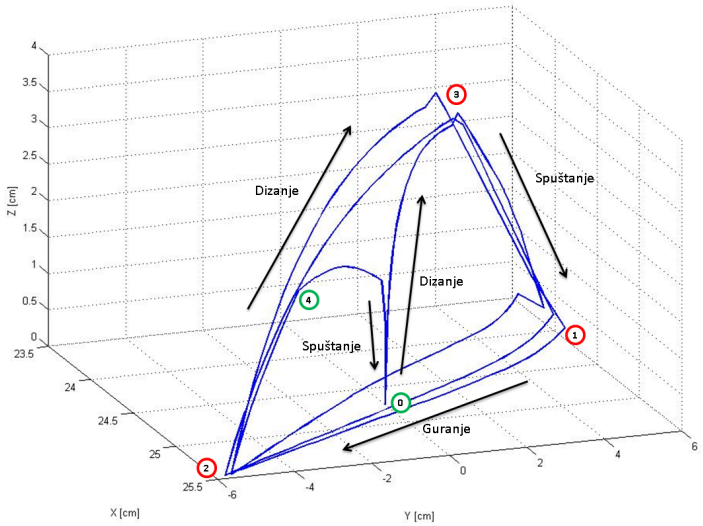


Slika 71: Snimljena putanja noge R2 u trodimenzionalnom prostoru pri hodanju robota

Na slici 73 se osim kompletnog koraka jasno vide i faze koraka objašnjene u potpoglavlju 5.1.3. U točki 1 noga kreće s fazom guranja, u točki 2 s fazom dizanja a u točki 3 s fazom spuštanja. Dodatne dvije točke su točka 0 i točka 4. Točka 0 predstavlja početak hoda, odnosno dizanje noge iz početne pozicije i od te točke se dalje razvija korak prolaskom kroz sve uobičajene faze. Do sada objašnjeno predstavlja rad *move* rutine. Iznimka se događa kada korisnik pusti palicu na upravljaču te algoritam hodanja više ne prima nove instrukcije. Tada se *move* rutina zaustavlja u fazi dizanja noge R2, hod staje te se pokreće *home* rutina koja nogu vrača u početni položaj (do točke 0). Rad *home* rutine se može vidjeti od točke 4 do točke 0 (početna pozicija). Slično se vidi i na slici 72. Razlika je u tome što je početna faza noge R1 faza 1 tj. guranje dok je početna faza noge R2 faza 2 tj. faza dizanja. Također je puštanjem palice upravljača noga R1 prekinuta u fazi guranja što rezultira nešto drukčijom putanjom *home* rutine usporedno s putanjom *home* rutine noge R2 koja je prekinuta u fazi 2 tj fazi guranja.

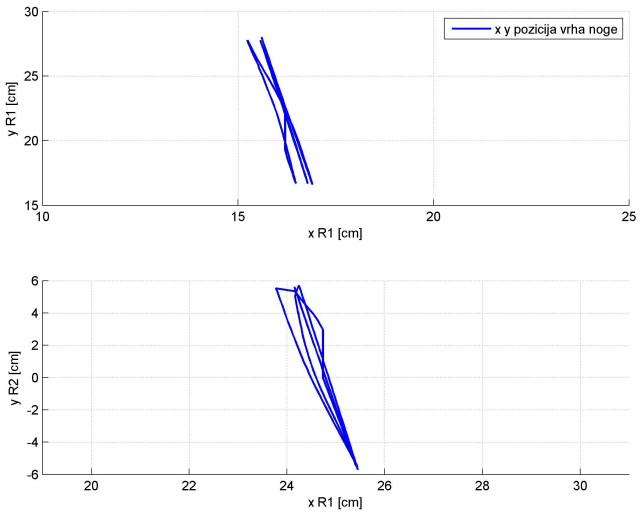


Slika 72: Snimljena putanja noge R1 u trodimenzionalnom prostoru pri hodanju robota

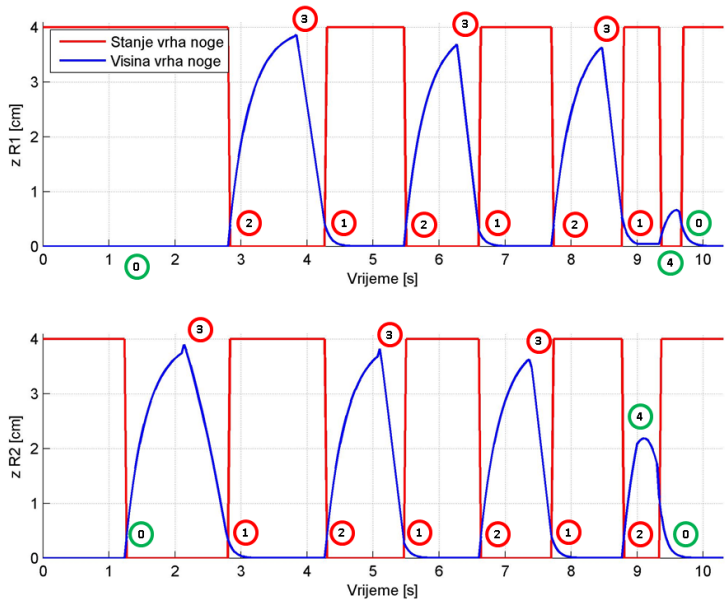


Slika 73: Snimljena putanja noge R2 u trodimenzionalnom prostoru pri hodanju robota

Na slici 74 se može vidjeti projekcija gore opisanih putanja na x-y ravninu. Ovakva putanja potvrđuje kretanje robota u zadanom smjeru. Faze koraka opisane istim točkama kao na slikama 72 i 73 se najbolje mogu vidjeti na slici 75 koja prikazuje putanje nogu R1 i R2 po z osi. Stanje vrha noge opisano je kao 0 za nogu u zraku te > 0 za prizemljenu nogu.

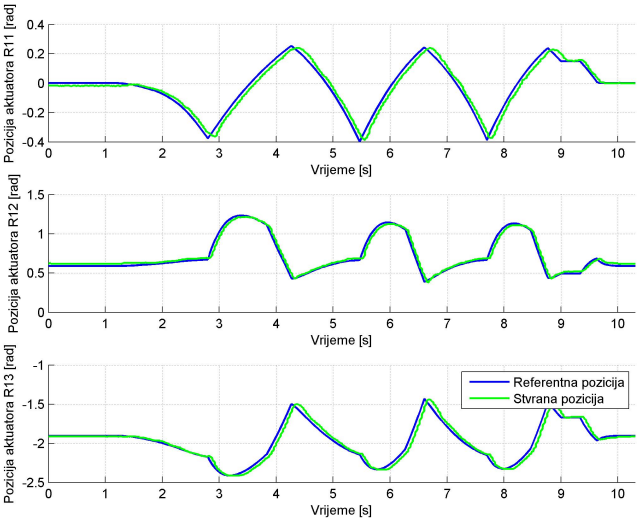


Slika 74: Snimljena putanja noge R1 (gore) te noge R2 (dolje) u dvodimenzionalnom prostoru pri hodanju robota (x-y ravnina)

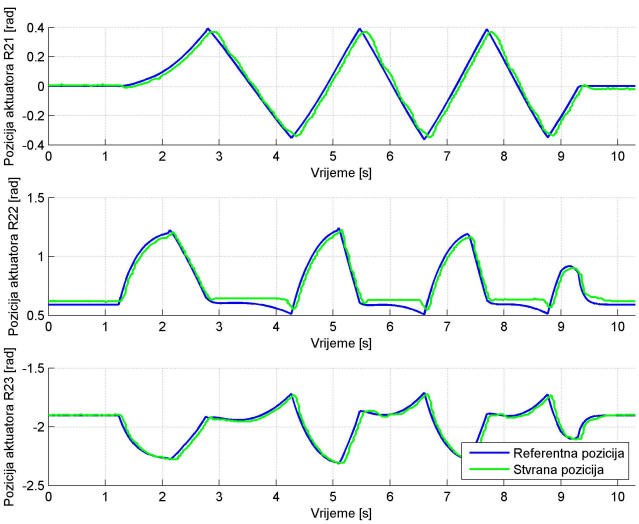


Slika 75: Snimljena putanja po z osi noge R1 (gore) te noge R2 (dolje) u vremenu

Kao što je opisano u poglavlju 5.1.1 sve gore navedene putanje se planiraju u domeni koordinata vrhova nogu a nakon se te koordinate direktnom i inverznom kinematikom prikazuju u domeni zakreta zglobova svake noge. Kako bi sve putanje bile praćene uz što je moguće manju pogrešku, električni aktuatori trebaju biti u stanju precizno pratiti zadane zakrete zglobova. Praćenje zadanih referenci aktuatora je prikazano slikama 76 i 77. Može se primijetiti malo odstupanje u obliku kašnjenja uzrokovano prijelaznom pojavom motora. Dodatnu grešku unosi zračnost nastala zbog velikog prijenosa servo motora. No unatoč malom odstupanju, korišteni aktuatori vrlo dobro prate zadani referentni položaj u vremenu.



Slika 76: Praćenje zadanih referenci zglobova noge R1



Slika 77: Praćenje zadanih referenci zglobova noge R2

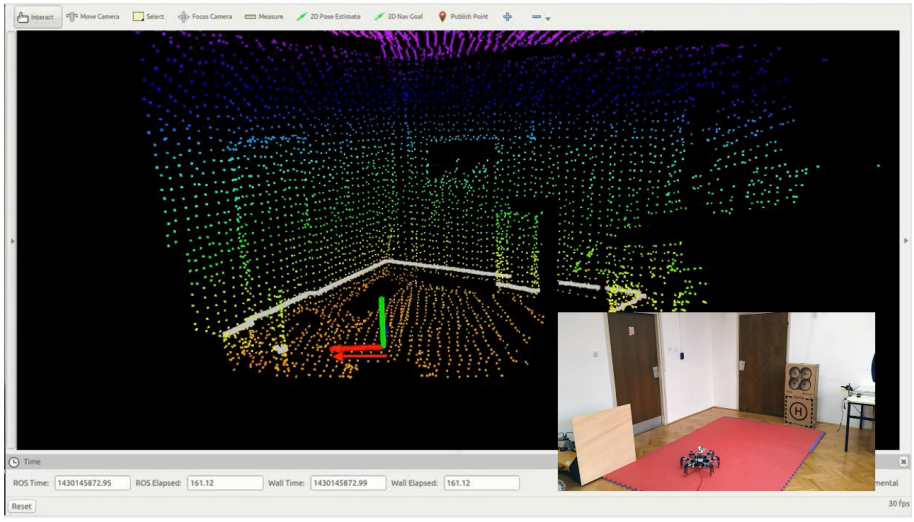
Stvarno kretanje robota se najbolje može vidjeti u priloženim video isječcima:

- Video 1 - Prikazana je rutina ustajanja a zatim translacijsko, rotacijsko i kombinirano gibanje razvojne platforme.
- Video 2 - Prikazano translacijsko, rotacijsko i kombinirano gibanje razvojne platforme na mjestu a zatim i kombinacija hoda s translacijom i rotacijom na mjestu.
- Video 3 - Prikazan hod s redom *tripod*, modificiranim *tripod*, *ripple* te *wave gaitovima*.

6.2. Simultana lokalizacija i mapiranje

Rezultati predobrade podataka za korištenje *LOAM back and forth* algoritma kao i rezultati dobiveni korištenjem algoritma mogu se vidjeti na video isječku pod nazivom "Simultana lokalizacija i mapiranje". Slika na videu se sastoji od dva bitna dijela, programskom okruženja Rviz i snimke okoline unutar koje se kreće razvojna platforma. Ovakva organizacije je bila potreba kako bi se mogla pronaći povezanost između okoline koju je algoritam mapirao i stvarne okoline koja ga okružuje. Ista stvar vrijedi i za poziciju robota.

Bijela "crta" odnosno rotirajući *point cloud* koji će se vidjeti na cijelom trajanju videa je rezultat predobrade podataka dobivenih s laserskog radara. *Point cloud* se također vidi na Slici 78. On predstavlja jedno skeniranje radara rotirano oko z osi statičnog koordinatnog sustava radara. Na snimci se vidi kako *point cloud* pravilno iscrtava prostor koji se nalazi u okolici platforme.

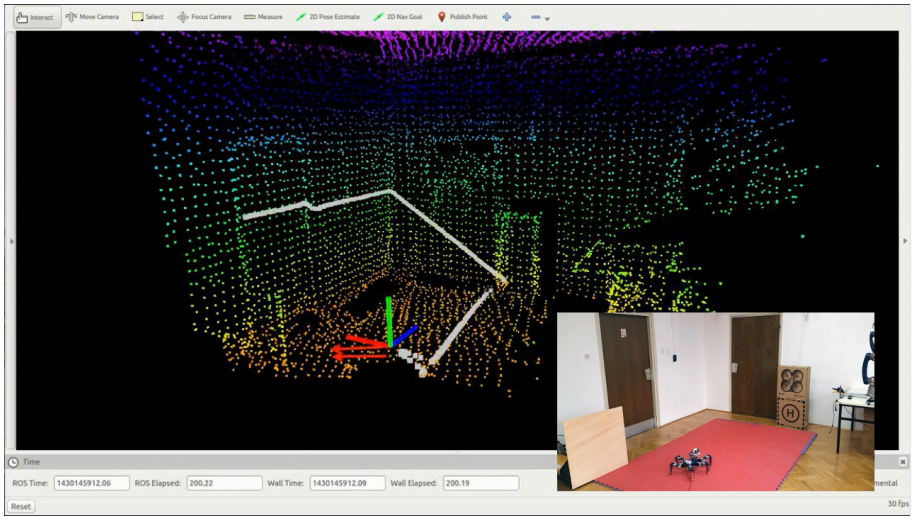


Slika 78: Prikaz očitanog prvog pomaka platforme i napravljene mape nakon očitanja

Na samome početku video isječka vidi se napravljena mapa prostora zajedno s pronađenom pozicijom platforme unutar nje. Mapa je rađena dok je robot bio statičan. Za početnu izgradnju mape i lokalizaciju algoritmu nije potrebno puno vremena, iz razloga što je robot statičan. Mapa je precizna uz mala odstupanja preciznosti pri vertikalnoj poziciji lasera. Vide se neki dijelovi mape na kojima ne postoje podaci o terenu. Razlog tome mogu biti tri razloga: teren se nalazi van radnoga prostora lasera, teren se nalazi iza nekog objekta koji sprječava zraku da ga mapira ili teren reflektira zraku suprotno od laserskog radara te je on ne očitava. Sva tri slučaja mogu se vidjeti na Slici 78. Primjer prvog slučaja je crno prikazana pozadina iza platforme, drugi slučaj pojavljuje se u pozadini kartonskih kutija s desne strane mape dok se drugi slučaj pojavljuje u gornjem dijelu desnih vrata na mapi.

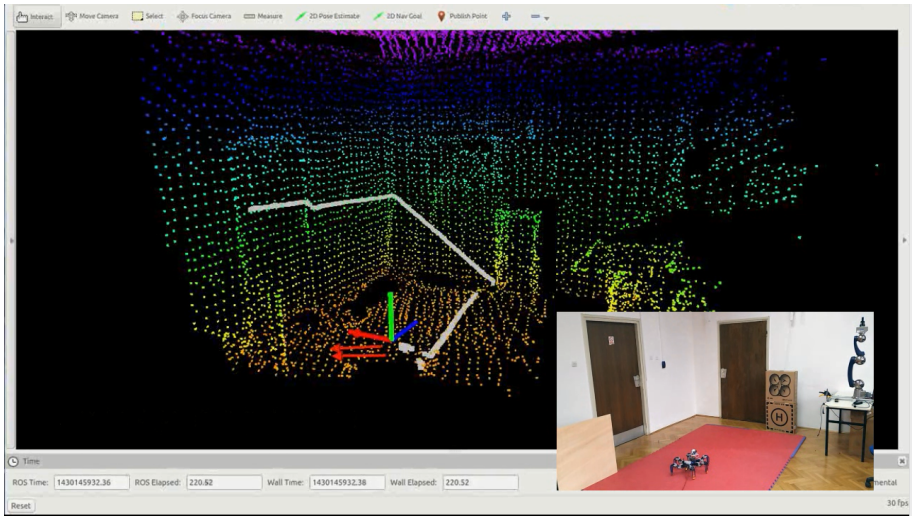
Prvi ozbiljniji posao algoritma dolazi nakon translacije platforme prema naprijed. Već pri početku translacije vidi se kako ulazni *point cloud* više ne iscrtava prikazani prostor koji se nalazi u okolici platforme. Razlog tome je što je algoritmu potrebno dulje vremena da odredi novu poziciju platforme u prostoru. Tek nakon osvježavanja nove pozicije ulazni *point cloud* ponovo prati okolinu. Za određivanje nove pozicije algoritmu je trebalo $\sim$ 30 sekundi. Prvi Pomak platforme jako se dobro vidi unutar Rviz okruženja i prikazan je na Slici 78.

Drugi pomak platforme je rotacija u desnu stranu. Dobivena mapa nakon drugog pomaka može se vidjeti na Slici 79. Kao i za prvi pomak, algoritmu je potrebno ~ 30 sekundi da platformu ponovo pozicionira na ispravnu lokaciju unutar napravljene mape. Nova pozicija robota donosi i više informacija o okolini koje možda nisu bile vidljive u prijašnjim pozicijama pa novo dobivena mapa sadrži više informacija od prethodne. To se dobro može vidjeti na podlozi ispred platforme kao i na gornjem dijelu desnih vrata, prikazanih na Slici 79.



Slika 79: Prikaz očitanog drugog pomaka platforme i napravljene mape nakon očitanja

Zadnji pomak platforme ponovo je translacijsko gibanje prema naprijed. Dobivena mapa nakon pomaka može se vidjeti na Slici 80. Zanimljivo je primijetiti kako je ovoga puta algoritmu bilo potrebno tri puta manje vremena za pronalaženje nove pozicije platforme unutar napravljene mape. Dobivena mapa okoline, usporedno s prijašnjim, sadrži najviše informacija o njoj.



Slika 80: Prikaz očitanog trećeg pomaka platforme i napravljene mape nakon očitanja

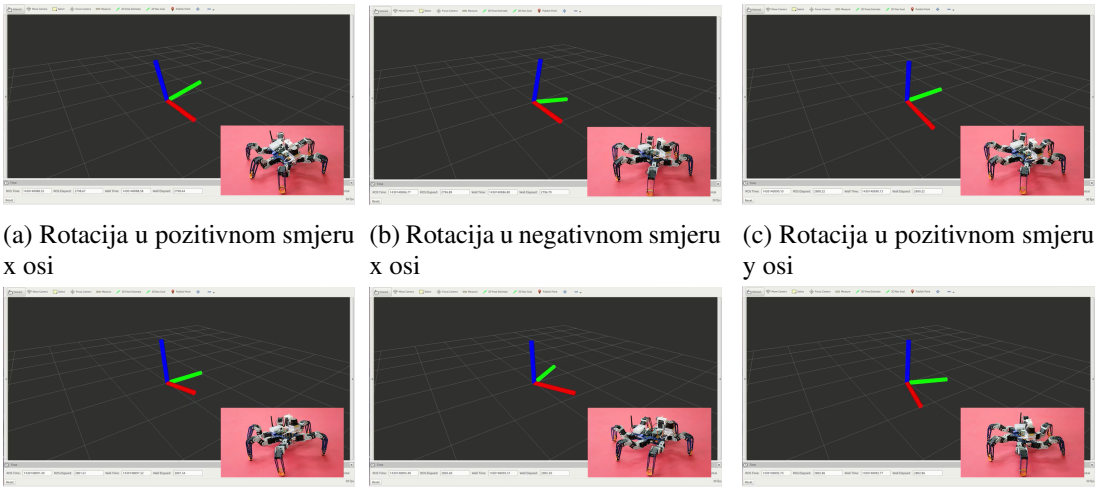
Dobivena mapa prostora dovoljno je precizna kako bi se mogla koristiti u druge svrhe. Jedna od primjena je pružanje informacija platformi o trenu koji ju okružuje u svrhu hoda platforme po neravnom terenu.

6.3. Inercijalni mjerni sustav

Rezultati mjerenja inercijalnog mjernog sustava najbolje se mogu vidjeti u priloženom video isječku pod nazivom Inercijalni mjerni sustav. Na videu je prikazan test rotacije na mjestu, translacije na mjestu i kombinirane rotacije i translacije razvojne platforme uz prikazane orijentacijske podatke koje šalje inercijalni mjerni sustav.

Programsko okruženje prikazano u pozadini videa je Rviz, unutar kojega se prikazuje orijentacija razvojne platforme u stvarnom vremenu. Orijentacija ja predstavljena koordinatnim sustavom koji se rotira u sve tri osi.

Na Slici 81 može se vidjeti mjerenja sustava pri rotaciji platforme na mjestu. Platforma se rotira oko svoje x (Slika 81a i 81b), y (Slika 81c i 81d) i z (Slika 81e i 81f) osi u pozitivnom i negativnom smjeru. Pozitivni smjer rotacije oko neke osi definiran je pravilom desne ruke. Inercijalni mjerni sustav prepoznaje spomenute rotacije koje uzrokuju promjenu iznosa izlaznog kvaterniona sustava. Kvaternion je prikazan koordinatnim sustavom. Mjerenja dolaze bez velikog kašnjenja i šuma.



(a) Rotacija u pozitivnom smjeru x osi

(b) Rotacija u negativnom smjeru x osi

(c) Rotacija u pozitivnom smjeru y osi

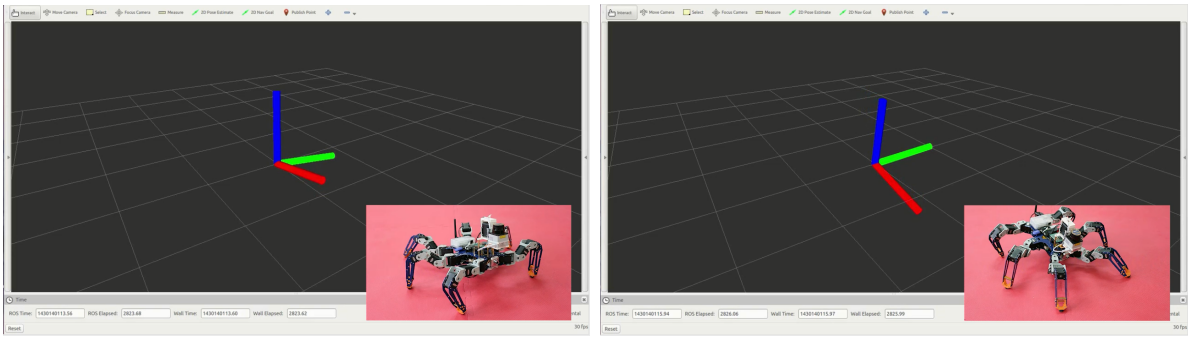
(d) Rotacija u negativnom smjeru y osi

(e) Rotacija u pozitivnom smjeru z osi

(f) Rotacija u negativnom smjeru z osi

Slika 81: Prikaz rada inercijalnog mjernog sustava pri rotaciji platforme

Na drugom dijelu video isječka razvojna platforma translatira svoju položaj tijela, pritom ne mijenjajući svoju orijentaciju. Kako se u Rvizu prikazuje orijentacijski podaci s inercijalnog mjernog sustava, oni će za vrijeme translacije ostati ne promijenjeni. Za vrijeme translacije u videu se vidi malo podrhtavanje koordinatnog sustava, razlog podrhtavanja su vibracije dobivene s električnih aktuatora koji ih stvaraju svojim kretanjem.



(a) Trasnacija u pozitivnom smjeru x osi i rotacija u negativnom smjeru y osi

(b) Trasnacija u negativnom smjeru x osi i rotacija u pozitivnom smjeru y osi

Slika 82: Prikaz rada inercijalnog mjernog sustava pri rotacijskom i translacijskom gibanju platforme

Na slici 82 prikazan je rad inercijalnog mjernog sustava uz kombinirano rotacijsko i translacijsko gibanje platforme. Kako se već može zaključiti orijentacija koordinatnog sustava mijenjati će se samo pri rotacijski kretnjama.

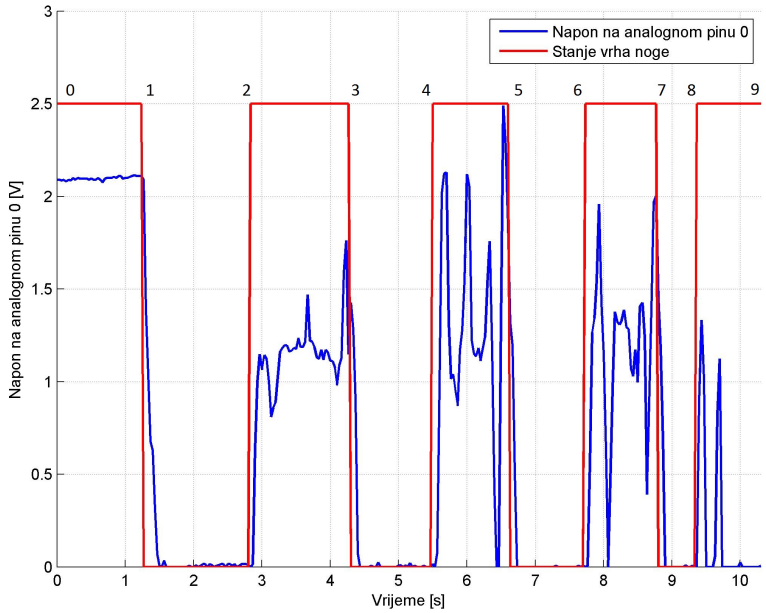
6.4. Senzori sile

Rezultati rada senzora sile se mogu vidjeti u priloženom video isječku Senzor sile. Rad senzora testiran je tako što se rukom simulirao pritisak različitog intenziteta te se u stvarnom vremenu prikazivale dobivene vrijednosti. Intenzitet pritiska može se podijeliti na:

- Bez pritiska - predstavlja period kad je senzor sile neopterećen
- Slab pritisak - predstavlja period kad je senzor sile opterećen pritiskom "slabog do srednjeg" intenziteta
- Jak pritisak - predstavlja period kad je senzor sile opterećen pritiskom "srednjeg do velikog" intenziteta

Senzor sile se tijekom testa predviđeno ponašao, tako što se bez pritiska mjerile vrijednosti približne 0, uz slab pritiska vrijednosti između 300 do 600 te uz jak pritisak vrijednosti veće od 700. Njegovo ponašanje je dinamično te tima osigurava brzu dinamiku prijelaza stanja.

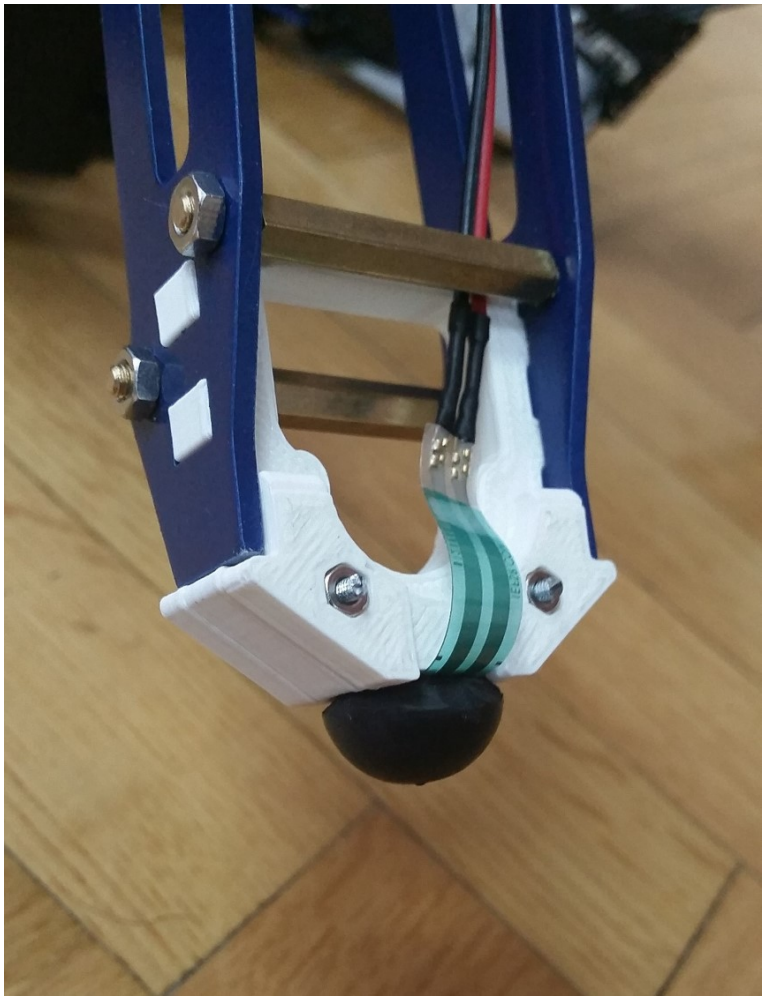
Drugi test senzora sile sastojao se od mjerenja vrijednosti senzora dobivenih tijekom hoda robotske platforme tripod načinom hodanja. Slika 83 prikazuje dobiveni napon analognog pina 0, koji mjeri napon kruga senzora sile R1, uz tripod način hoda robotske platforme. Mjerenja analognog pina su proporcionalna mjerenoj sili (poglavlje 5.4.). Na slici se također nalazi stanje vrha noge dobivena iz algoritma hoda, gdje vrijednosti veće od 0 predstavljaju vrh noge na tlu.



Slika 83: Dobivena očitanja senzora sile noge R2 uz tripod način hoda robotske platforme

Do vremenskog trenutka 1, robotska platforma se nalazi u *home* poziciji te su očitanja napona prilično glatka. Vremenski intervali 1-2, 3-4, 5-6 i 7-8 predstavljaju period kad je stanje noge iznad poda te se su očitanja napona oko vrijednosti 0. Malo vremensko kašnjenje senzora za stanjem noge

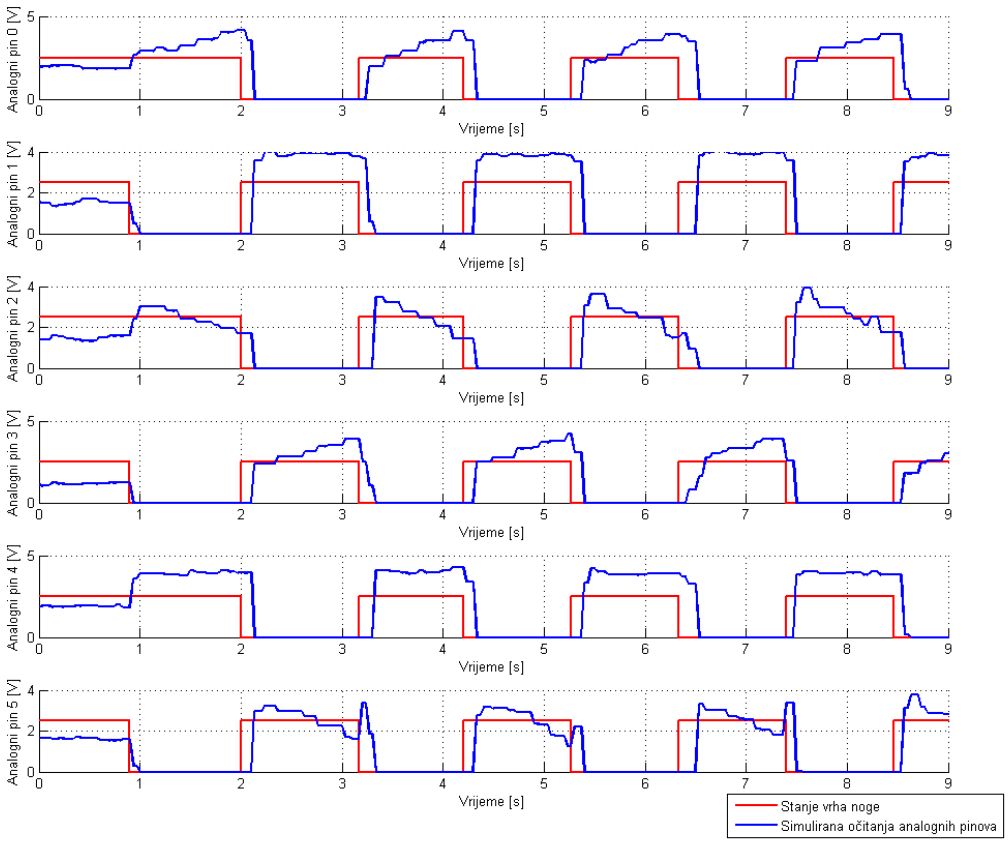
uzrokovano je vremenom potrebnim da ju aktuatori podignu. Vremenski intervali 0-1, 2-3, 4-5, 6-7 i 8-9 predstavljaju period kad je stanje noge pod te su očitanja napona veća od vrijednosti 0. Šiljci očitanja te vrijednosti oko nule unutar tih perioda uzrokovana su raznim nepravilnostima (terenom, odstupanjem centra mase platforme od njenog geometrijskog centra, itd.). Taj problem bi se mogao riješiti uvođenjem povratne veze senzora sile u algoritam hoda te njihovom pametnom obradom, ali to je pitanje izvan okvira ovog rada.



Slika 84: Senzor sile na robotskoj platformi

6.5. Robotski simulator

Ispravnost rada robotski simulator najviše se testirala tijekom njegovog razvoja te postavljanja niza dinamičkih parametara. U svrhu ove razvoje platforme potrebno je obaviti dva test, provjeriti ispravnost rada senzora sile te provjeriti ispravnost rada samog robota. Rezultati prvog testa prikazani su Slikom 85. Kao i u poglavlju 6.4 simulirana mjerenja analognog pina su proporcionalna simuliranoj mjerenoj sili. Analogni pinovi 0-2 mjere napone krugova nogu R1-R3, dok analogni pinovi 3-5 mjere napone krugova nogu L1-L3.

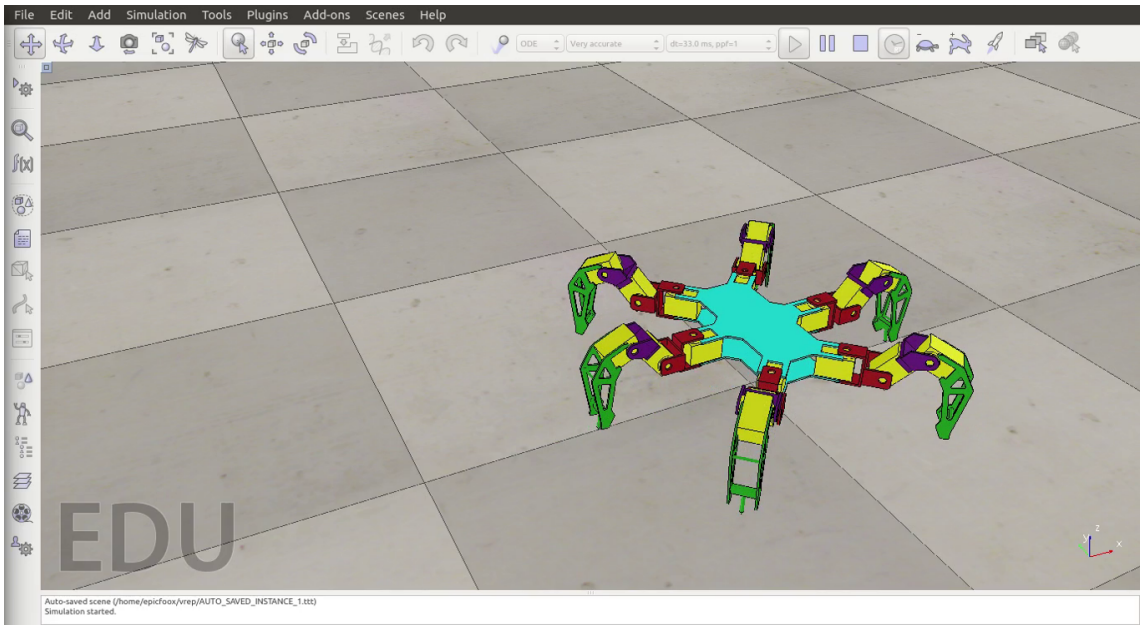


Slika 85: Dobivena simulirana očitanja analognih pinova uz tripod način hoda robotske platforme

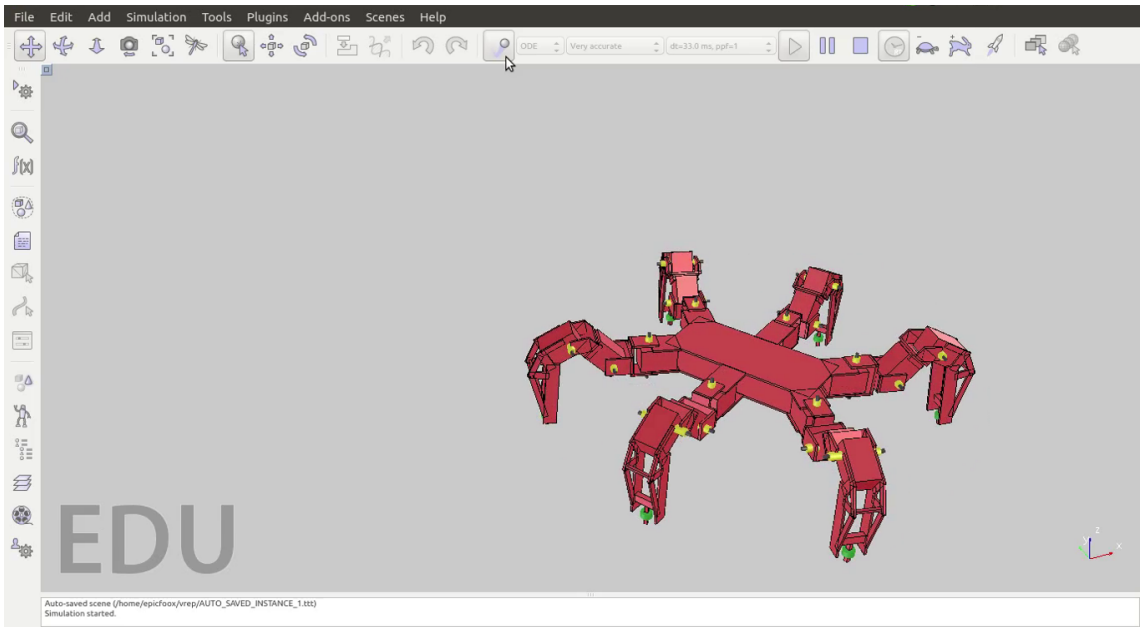
Uspoređivanjem dobivenih rezultat simulacije s onim dobivenim u stvarnosti (Slika 83) mogu se uočiti par sličnosti i razlika. Oba rezultat pokazuju očitanja pinova oko vrijednosti 0 dok je noga iznad tla, stanje vrha noge = 0 te očitanja pinova veća od 0 dok je stanje vrha noge na tlu. Također oba rezultat pokazuju malo kašnjenje motora za stanjem vrha noge, uzrokovano prijelaznom pojavom aktuatora. Očitane vrijednosti iz simulacije su nešto veće od stvarnih vrijednosti, što može biti uzrokovano tehničkom izvedbom senzora sile. Takav i slični problemi se mogu riješiti jednostavnim umjeravanjem senzora sile.

Rezultati provjere ispravnog rada robotske platforme u simulaciji mogu se vidjeti u priloženom video isječku Simulacija. Dobiveni su vrlo dobri rezultati responzivnosti simulacije te dinamike po-

našanja robotske platforme. Preciznije ali sporije ponašanje se može dobiti smanjivanjem vremena uzorkovanja, dok se brže ali nepreciznije ponašanje može dobiti njegovim povećavanjem. Odabrana je vrijednost 33,3 ms zbog frekvencije osvježavanja algoritma hoda koji iznosi 30 Hz.



Slika 86: Test statičkog modela robotske platforme



Slika 87: Test dinamičkog modela robotske platforme

Zaključak

Roboti hodači su neizbježna sastavnica skore budućnosti. Do toga će doći samo ako se konstantno ulaže u razvoj i istraživanje. Šesteronogi hodač ili hexapod robot razvijen je i izgrađen kao razvojna platforma na kojoj će se moći testirati razni algoritmi hodanja po terenima različitih oblika. Pametni Dynamixel servo motori su glavni pokretači razvojne platforme. Oni su se pokazali kao odličan izbor zbog serijskog načina povezivanja te dobrog praćenja zadane pozicije u vremenu. Njima upravlja algoritam hoda koji je nakon testiranja u simulaciji uspješno odradio zadatak i na stvarnom robotu. Simulacija razvojne platforme je neizbježan alat za isprobavanje novih algoritama i metoda. Od senzora, razvojna platforma je opremljena laserskim radarom za mapiranje trodimenzionalnog prostora, senzorima sile za osjet tla pod nogama te mjernim inercijalnim sustavom za povratnu vezu orijentacije robota. Svi nabrojani senzori su spremni za korištenje uz ROS pakete koji će se pobrinuti da svi podaci budu lako dostupni korisniku. Također tu je web kamera koja je spremna za prijenos slike preko gotovih ROS paketa. Cijelim sustavom upravlja središnja upravljačka jedinica Odroid U3 koja se pokazala kao idealno računalo zbog malih dimenzija i velike računalne moći.

Razvojna platforma dizajnirana je kako bi korisnik mogao preskočiti iscrpi proces razvoja i izrade ovakvog robota te se odmah koncentrirati na koristan rad ili razvoj novih algoritama. Samo neka od mogućih istraživanja na ovakvoj platformi su: Dinamična promjena *gaitova* u hodu s obzirom na zahtjeve terena, kompenzacija centra mase translacijom tijela robota te pronalazak novih gazišta u hodu po neravnim terenima. Na navedenim istraživanjima se već radi u okviru diplomskih radova te će se na njima pokazati puni potencijal ovakve platforme.

Literatura

[1] Adafruit. Using an fsr. https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr, 2015.

[2] Paul Bouchier and Mike Purvis. Ros - rosserial. http://wiki.ros.org/rosserial/, Travanj 2015.

[3] Blaise Gassend Brian P. Gerkey, Jeremy Leibs. Ros - hokuyo node. http://wiki.ros.org/hokuyo_node/, Travanj 2015.

[4] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: Part i. *Robotics and Automation Magazine*, 13(2):99–110, 2006.

[5] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: Part ii. *Robotics and Automation Magazine*, 13(3):108 – 117, 2006.

[6] Hardkernel. Odroid u3. http://hardkernel.coml, 2015.

[7] HobbyKing. Hku5 5v/5a ubec. http://hobbyking.com/, Travanj 2015.

[8] HobbyKing. Multiwii microwii atmega32u4. http://hobbyking.com/, Travanj 2015.

[9] HobbyKing. Zippy compact 4000. http://hobbyking.com/, Travanj 2015.

[10] Hokuyo. Laser range finder urg-04lx. http://www.hokuyo-aut.jp/, Travanj 2015.

[11] Honeywell. Hmc-5883l. http://honeywell.com/, Travanj 2015.

[12] MS Industrial. Web camera 301. http://int.ms-start.com/, Travanj 2015.

[13] Invensense. Mpu-6050. http://www.invensense.com/, Travanj 2015.

[14] Bogdan S. Krajči V. Kovačić, Z. *Osnove Robotike*. Graphis, Jurjevska 20, Maksimirska 88, Zagreb, 2002.

[15] Linux. Linux kernel. https://www.linux.com/, 2015.

[16] MathWorks. Matlab. http://www.mathworks.com/, 2015.

[17] University of Edinburgh. Robot learning and sensorimotor control homework. http://wcms.inf.ed.ac.uk/ipab/rlsc/homework, 2015.

[18] Coppelia Robotics. V-rep user manual. http://www.coppeliarobotics.com/helpFiles/, 2015.

[19] Coppelia Robotics. Virtual robot experimentation platform. http://www.coppeliarobotics.com/, 2015.

[20] Robotis. Dynamixel. http://www.robotis.com/xe/dynamixel_en, 2015.

[21] RobotShop. Force sensing resistor. http://www.robotshop.com/media/files/pdf/user-guide-for-fsr-sensor-hfs-01.pdf, 2015.

[22] ROS. Ros. http://www.ros.org/, 2015.

[23] ROS. Rviz. http://wiki.ros.org/rviz/, Travanj 2015.

[24] Sony. Playstation 3 dualshock. http://us.playstation.com/ps3/accessories/dualshock-3-wireless-controller-ps3.html, 2015.

[25] Dassault Systèmes. Solidworks. https://www.solidworks.com/, 2015.

[26] Radu Bogdan Rusu Tully Foote. Ros - laser geometry. http://wiki.ros.org/laser_geometry/, Travanj 2015.

[27] Wim Meeussen Tully Foote, Eitan Marder-Eppstein. Ros - tf2. http://wiki.ros.org/tf2/, Travanj 2015.

[28] Ubuntu. Ubuntu distribucija. http://www.ubuntu.com/, 2015.

[29] Fabio Varesano. Freeimu. http://www.varesano.net/, Travanj 2015.

[30] Xevelabs. Usb2ax. http://www.xevelabs.com/doku.php?id=product:usb2ax:usb2ax, 2015.

[31] Ji Zhang. Ros - loam back and forth. http://wiki.ros.org/loam_back_and_forth/, Travanj 2015.

[32] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. *Robotics: Science and Systems Conference (RSS)*, 2014.

Sažetak

Autori: Luka Fuček, Antun Vukičević i Josip Vukičević

Naslov rada: Razvoj i izrada šesteronogog hodača kao razvojne platforme

U ovom radu opisana je izrada i razvoj šesteronogog hodača kao razvojne platforme uz priložene upute za korištenje. Nakon kratkog uvoda u temu rada predstavljeni su dostignuti opći i specifični ciljevi rada. Šesteronogi hodač osmišljen je kao razvojna platforma otvorenog i modularnog tipa. Otvorenost razvijenih softverskih i hardverskih rješenja omogućuje dostupnost razvijenih algoritama i metoda korisniku, dok će njihova modularnost omogućiti dodavanje novih ili izmjenu postojećih. Time se ostvaruje potpuna personalizacija platforme ovisno o potrebi, bilo da korisniku neki dijelovi nisu potrebni ili ju želi nadograditi novim. Rad je podijeljen u četiri glavne cjeline.

Tehnička izvedba opisuje fizičku konstrukciju i funkcionalne sheme razvojne platforme. Fička konstrukcija navodi sve njezine dijelove popraćene detaljnim grafičkim prikazima i specifikacijama, dok funkcionalne sheme opisuju funkcionalni aspekt njezina načina rada.

Nabrojani su i opisani svi korišteni materijali softverskog i hardverskog tipa kao što su glavna i sporedna upravljačka jedinica te popratni operacijski sustav uz programske alate poput ROS-a, Matlaba, Solidworksa te v-repa. Obrađena su i pitanja aktuacije, senzorike, napajanja i komunikacije razvojne platforme s vanjskim svijetom.

Glavni dio rada sadržan je u metodama. Ovdje je opisan algoritam hodanja uz detaljan prikaz organizacije radnog prostora nogu, kinematičkih proračuna, planiranja koraka te *gaitova*. Za interakciju razvojne platforme s okolinom koriste se i razni senzori. Opisan je laserski radar pomoću kojeg se vrši simultana lokalizacija i mapiranje prostora bez kojega hodanje po nepredvidivim terenima ne bi bilo moguće. Dodatno se koristi senzor sile smješteni na vrh noge te inercijalni mjerni sustav za povratne informacije o orijentaciji razvojne platforme. Na kraju, neizostavna cjelina je robotski simulator bez kojega razvoj ne bi bio moguć jer svaka nova ideja u razvoju prvo prolazi kroz simulaciju.

Rad svih spomenutih metoda testiran je na stvarnom sustavu te su prikazani dobiveni rezultati.

Ključne riječi: šesteronogi hodač, razvojna platforma.

Summary

Authors: Luka Fućek, Antun Vukičević i Josip Vukičević

Title: Research and development of a six-legged robot as a development platform

In this paper, design and development of a six-legged walking robot as a development platform are described and instruction manual is given. After short introduction to the topic of the paper, general and specific goals, achieved within planned framework, are presented. The six-legged walking robot is designed as an open source and modular type development platform. Open source approach enables accessibility of developed algorithms and methods to the user. In the same time their modularity provides user with ability to add new ones or modify the existing software. Hence, a full personalization of the platform is enabled.

This paper is divided in four main sections. Section 3. describes physical construction and functional schemes of the platform. All parts, used for the construction, are presented, followed by detailed graphic representations and specifications. Functional schemes describe functional aspects of the platform. In Section 4. all software and hardware components, such as main and secondary control unit and accompanying operating system with programming tools (ROS, Matlab, Solidworks and v-rep) are listed and described, together with notion on actuation, sensors, power supply and communication modules and protocols.

The main part of the paper is given in Section 5. The walking algorithm is described with detailed presentation of legs workspace organisation, kinematic calculations, steps planning and gaits. Different sensors for interaction with surroundings are also described. Laser radar used for simultaneous localization and mapping is presented (walking on unpredictable terrain would be impossible without laser radar). Thorough descriptions of a force sensor, placed on the top of the leg, and inertial measurement system for orientation feedback, are given. Finally, the unavoidable section is related to the robot simulator. Without the simulator, the development wouldn't be possible because every new idea has to go through the simulation first. Functionalities of all mentioned methods and algorithms were tested on a real system and the results of those tests are shown in Section 6.

Keywords: hexapod robot, development platform.