Upravljanje modularnim robotima

Prkačin, Vicko

SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

STUDIJ ELEKTROTEHNIČKE I KOMUNIKACIJSKE TEHNOLOGIJE U POMORSTVU

DIPLOMSKI RAD

UPRAVLJANJE MODULARNIM ROBOTIMA

Dubrovnik, rujan 2017.

SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

STUDIJ ELEKTROTEHNIČKE I KOMUNIKACIJSKE TEHNOLOGIJE U POMORSTVU

DIPLOMSKI RAD

UPRAVLJANJE MODULARNIM ROBOTIMA

Mentor: **doc.dr.sc. Ivana Palunko**

Diplomant: **Vicko Prkačin**

Dubrovnik, rujan 2017.

ZAHVALA:

Zahvaljujem se svojoj mentorici doc.dr.sc. Ivani Palunko, na pruženom znanju, motivaciji za odabir projekta i bezbroj napisanih mailova!

Hvala kolegama Dubravku, Luki i Mateju na svim kafama u Freaky-a i lijepim danima provedenim na Sveučilištu u Dubrovniku. Hvala mom bivšem cimeru i nikad bivšem prijatelju Nikši na satima objašnjavanja kroz cijelo školovanje. Bez tebe bi ovo bio puno mukotrpniji put.

Hvala familiji na krovu nad glavom!

Hvala Bruce Lee-u što mi je uvijek čuvo leđa!

Hvala autoru projekta Albertu, na pomoći i pruženim materijalima. Sretno s nastavkom projekta!

I na kraju, hvala mojim drugovima Maroju, Teu, Dominiku, Pasku, Filipu, Vlahu, Marinu i drugom Dominiku na uvijek ugodnom društvu, nezaboravnim danima i zaboravljivim noćima... bez vas bi ovaj rad bio gotov prije dva mjeseca. Hvala!!

Sažetak:

Razvoj znanosti i tehnologije omogućio je čudesne stvari. Ljudska tehnologija dosegnula je međuzvjezdani prostor i najudaljenije kutke našega planeta. Kako nastavljamo s istraživanjem više smo svjesni koliko je naše znanje krhko. Mnoga područja još su neistražena, ali i nedostupna čovjeku. To nas nije obeshrabrilo, dapače, poslužilo je kao motiv za daljnji napredak i učenje. Rezultat toga su modularni roboti poput DTTO-a. Znanje i tehnologija iza njega opisani su u ovome radu. U V-REP programskom okruženju izrađen je i testiran model robota. Kao najveći nedostatak robota pokazala se ograničena mobilnost. Prema tome su ponuđena rješenja iz domene upravljanja, ali i određene funkcionalne preinake koje će omogućiti dodatnu slobodu u kretanju. Dana rješenja simulirana su u V-REPu i implementirana pri spajanju dva nezavisna DTTO modula. Predstavljene su i prednosti modularnih robota poput robusnosti i prilagodljivosti koje ih čine idealnim kandidatom za nastavak najstarije ljudske misije - istraživanja čudesnih novih svjetova i odlazak tamo gdje nitko nije išao.

Ključne riječi: samo-rekonfigurabilni roboti, modularna robotika, mehatronika, DTTO.

Abstract:

Scientific and technological advances have made wonderful things possible. Human technology has reached interstellar space, as well as the furthest reaches of our planet. As we continue to explore, we become more aware of how fragile our knowledge is. There are still undiscovered places, unreachable to humans. But this never discouraged us, on the contrary, it motivated us to learn and develop. As a result we now have modular robots like DTTO. The knowledge and technology behind it have been presented in this thesis. The robot was modeled and tested in V-REP. The limited mobility of the robot was shown to be the biggest disadvantage. Control solutions and functional modifications were introduced. Given solutions were tested in VREP and implemented on coupling of two independent modules. The numerous advantages of the robots such as their robustness and versatility have made them ideal candidates for the most ancient of humanity's missions – to explore strange new worlds, to seek out new life and civilizations, to boldly go where no man has gone before.

Key words: self-reconfigurable robots, modular robotics, DTTO mechatronics.

S A D R Ž A J

1. UVOD

Robotika je interdisciplinarna grana znanosti koja obuhvaća područja elektrotehnike, mehanike i računarstva. Bavi se dizajnom, izradom, oblikovanjem i upravljanjem robota što podrazumijeva očitanje i obradu podataka sa senzora, njihovo procesiranje te implementaciju algoritama za aktuaciju. Riječ robot skovao je češki pisac Karel Čapek i prvi put je predstavio u svome dramskom djelu *Rossumovi univerzální roboti* [1]. Riječ je izvedena iz slavenske riječi *robota* što znači *rad*. Iz značenja riječi može se iščitati svrha robota i robotike kao znanosti, a to su dizajn i izrada robota za izvršavanje zadataka koje ljudi ne mogu, ne žele ili ne obavljaju dovoljno učinkovito. Ljudska težnja za olakšanjem svakodnevnog života i ostvarivanjem za ljudsku anatomiju nemogućih ciljeva, stara je vjerojatno koliko i sam ljudski rod, mitologije drevnih civilizacija ispunjene su legendama o stvorenjima nadljudskih sposobnosti. Iako je povijest čovječanstva od samih početaka bogata primjerima strojeva koji su bili sposobni ispunjavati najrazličitije zadaće, za početak razvoja robotike kakvu danas poznajemo bilo je potrebno pričekati drugu polovicu 20. stoljeća i razvoj računarstva i elektrotehnike.U tom razdoblju nagli tehnološki napredak doveo je do široke primjene robotike u gotovo sva područja ljudskog djelovanja čime su postavljeni temelji za razvoj suvremenih robota.

Građa suvremenih robota, u većini slučajeva, može se razložiti na tri osnovne cjeline:

- konstrukciju i aktuatore
- osjetilne elemente
- upravljački uređaj.

Konstrukcija robotu daje mehaničku čvrstoću dok mu translacijski ili rotacijski aktuatori daju slobodu gibanja oko ili u smjeru jedne

od triju osi. Osjetilni elementi robotu služe kao čula kojima percipira svijet oko sebe. Ona mu pružaju povratnu informaciju o veličinama važnima za proces koji obavlja i omogućuju funkcioniranje u zatvorenoj upravljačkoj petlji. Upravljački uređaj je središnji dio svakog robota. On je mozak robota i u njega je ugrađeno cjelokupno znanje o radnom procesu i algoritmi upravljanja. Zadatak upravljačkog uređaja je očitati i obraditi informacije s osjetilnih elemenata te prema upravljačkom algoritmu izračunati vrijednosti potrebne aktuatorima za uspješno obavljanje predviđene radne zadaće. Upravljanje robotom podrazumijeva regulaciju njegove kinematike i dinamike. Kinematika geometrijski opisuje gibanje robota, uzimajući u obzir poziciju i orijentaciju cijelog robota, kao i pojedinih dijelova s pripadnim brzinama i ubrzanjima, ne uzimajući u obzir sile i momente koji djeluju na tijelo. Dinamika, s druge strane, opisuje dinamičko ponašanje robota u stvarnim uvjetima gibanja uzimajući u obzir sile i momente koji djeluju na tijelo i pritom generiraju željeno gibanje [2]. Ukupni dinamički model robota objedinjuje dinamiku pojedinih aktuatora i njihov utjecaj na ponašanje robota kao cjeline. Dinamički model osobito je važan segment pri izradi matematičkih modela robota korištenih u računalnim simulacijama. Simulacije su sastavni dio krivulje razvoja i izrade, prvo prototipa, a kasnije i radnih modela robota.

Roboti su danas sastavni dio života modernog čovjeka. Roboti nam izrađuju automobile, obavljaju poslove u okolinama opasnim po zdravlje i život čovjeka, istražuju podmorje i druge planete, a sve više ulaze u kućanstva te osim za rad služe i za zabavu. Slično kao što je industrijalizacija dovela do ukidanja ropstva, možda će daljnji razvoj robotike i pojava inteligentnih robota otvoriti vrata nove ere čovječanstva - bez rada kakvoga danas poznajemo i gdje su tri zakona robotike Isaaca Asimova [3] sastavni dio statuta svake zemlje našeg, a možda i nekih novih planeta.

Modularni samorekonfigurabilni roboti (engl. *modular self-reconfigurable robots*) građeni su od modula koji se mogu promatrati kao zasebne robotske stanice. Svaki modul je robot za sebe i sadrži sve komponente koje čine jedan robot: aktuatore, osjetilne elemente, upravljačku jedinicu i (uglavnom) baterije. Osim osnovnih dijelova, svaki od modula sadrži i neku vrstu komunikacijske tehnologije koja mu omogućuje razmjenu informacija s ostalim modulima te mehanizam za spajanje koji mu omogućuje spajanje i razdvajanje od drugih modula. Aktuatori svakom od modula omogućavaju vlastito kretanje u prostoru, pomicanje drugih modula ili kretnju s drugim modulima. Različitim kombinacijama spajanja i razdvajanja modula, obično njih desetak do sto, modularnom robotu omogućeno je da mijenja oblik. Time se modularni roboti razlikuju i ističu od svih drugih vrsta robota.

Slika 1.1MTRAN III modularni robot [4]

Modularni samorekonfigurabilni roboti moraju zadovoljiti sljedeće kriterije [5:5]:

- *Modularnost* – robot je građen od više nezavisnih modula.
- *Rekonfigurabilnost* – moduli se mogu spajati na više različitih načina i tako formirati robote različitih veličina, oblika i funkcionalnosti.
- *Dinamička rekonfigurabilnost* – moduli se mogu odvajati i spajati dok je robot aktivan.
- *Samo-rekonfigurabilnost* – robot samostalno može mijenjati način na koji su moduli povezani.

Modularna robotika sve više postaje područje zanimanja znanstvenih istraživanja zbog modularne prirode robota koja je zaslužna za jedinstvene osobine koje krase modularne robote. Prva i vjerojatno najvažnija je svestranost – budući da se moduli mogu spajati na više načina modularnom robotu omogućeno je obavljanje širokog niza funkcija, ali i lakša prilagodba promjenama u njegovoj okolini. Iz modularnosti proizlazi i druga važna osobina, a to je robusnost – bez obzira na uzrok mogućeg kvara na nekom od modula, drugi modul može preuzeti njegovu ulogu. Povećanjem broja modula koji su u kvaru očekivan je i pad performansi sustava, ali važna je činjenica da sustav može nastaviti funkcionirati bez obzira na kvarove. Budući da su moduli uglavnom jednake građe lako ih je zamijeniti, a troškovi se zbog masovnosti proizvodnje smanjuju, što omogućuje izradu redudantnih modula koji mogu služiti kao zamjena ukoliko dođe do kvara. Rezultat masovne proizvodnje je relativno niska cijena proizvodnje modula, uzimajući u obzir njihovu složenost. Navedene osobine modularne robote čine prikladnima za primjene u neizvjesnim okolinama koje zahtijevaju prilagodljivost, poput istraživanja svemira, akcija traganja i spašavanja i slično.

2. DTTO Modularni robot

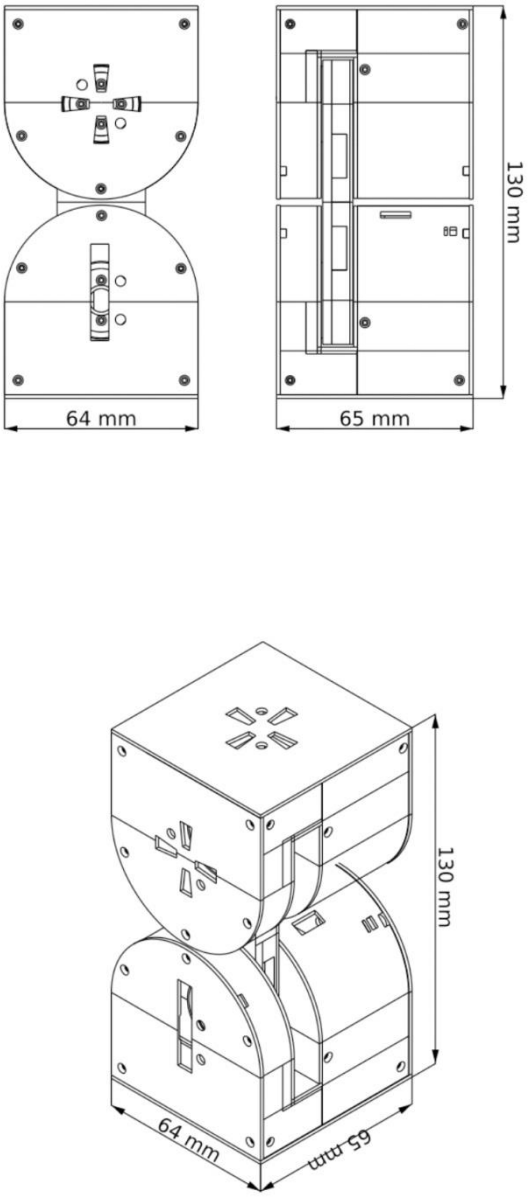
DTTO modularni robot nastao je kao projekt za diplomski rad inženjera elektrotehnike Alberta Molina Pérez s Universitat Rovira i Virgili u Tarragoni, Catalunya [6]. Inspiriran M-TRAN modularnim robotom [7] [8] i njegovim mogućnostima odlučio je i sam napraviti modularni robot, ali s idejom da cijeli projekt bude *open source,* kako u sklopovskom tako i u programskom dijelu, jeftin i otvoren za javnost, što ga čini prvim takvim projektom iz područja modularne robotike u svijetu. Projekt je predstavljen široj javnosti putem HACKADAY.IO platforme, najveće zajednice za razvoj hardwarea na svijetu, gdje je 2016. godine osvojio prvu nagradu za najbolji projekt [9]. Cilj projekta je razvoj modularnog robota koji će biti u mogućnosti pomoći ljudima u kriznim situacijama poput prirodnih katastrofa gdje će prilagodljivost i robusnost modularnih robota najviše doći do izražaja. Danas na projektu sudjeluju stručnjaci sa sveučilišta diljem svijeta. Prototipi robota izrađeni su u Kini, Francuskoj, Indiji, Ujedinjenom Kraljevstvu, SAD-u, a od nedavno prvi prototip ima i Sveučilište u Dubrovniku.

Osnovne značajke DTTO modularnog robota su [10]:

- Male dimenzije i samodostatnost
- U potpunosti ga je moguće isprintati 3D printerom
- Samorekonfigurativnost
- Bluetooth i radiokomunikacija
- Više načina gibanja
- Punjive baterije
- Razvijen simulacijski model
- Niska cijena modula (<55$)
- Open source hardware i software
- Prvi open source samorekonfigurativni modularni robot na svijetu.

2.1. Arhitektura

DTTO robot se sastoji od dvije cjeline, muške i ženske, povezane osovinom. Izgled i osnovne dimenzije DTTO v2 robota prikazane su na *Slici 2.1*. Muški i ženski segment su kockastog oblika s jednom zaobljenom stranicom koja omogućava kretanje robota. Iz dizajna modula je vidljivo da je kao predložak korišten M-TRAN (Modular transformer) robot, razvijen na Nacionalnom institutu za znanost i tehnologiju u Tokiju (AIST) [11]. Ovakav dizajn omogućuje iskorištavanje prednosti kako rešetkastih, tako i lančanih tipova rekonfigurabilnih modularnih robota. Naime, u modularnoj robotici razlikujemo dvije vrste rekonfigurabilnih robota, a to su rešetkasti (engl. *lattice type*) i lančani (engl. *chain type*) rekonfigurabilni roboti [5]. Kod rešetkastog tipa, unutar strukture robota moduli su organizirani slično kao atomi u strukturi kristala. Svaki modul može zauzeti samo diskretnu poziciju unutar definirane strukture rešetke. Roboti se mogu kretati unutar te strukture ponavljajući sekvence spajanja i odvajanja dok ne dosegnu željenu diskretnu poziciju. Budući da su pozicije unutar strukture unaprijed određene, pojednostavljuje se proces rekonfiguracije. Jedan od prvih samo-rekonfigurabilih robota i klasičan predstavnik rešetkastog tipa modularnih robota je Fracta [12]. Za razliku od rešetkastog tipa, kod lančanog tipa rekonfigurativnih robota moduli se mogu slobodno kretati u prostoru, spajanjem modula u lance tvore stablaste strukture ili petlje. Posljedica obilježja da moduli mogu zauzimati proizvoljne pozicije u prostoru je dodatan korak u procesu rekonfiguracije. Gdje je sada, osim razdvajanja, pomicanja i spajanja, potrebno i pronaći modul s kojim se želi izvršiti spajanje jer se modul ne mora nalaziti na diskretnoj poziciji kao što je to slučaj kod rešetkastog tipa rekonfigurativnih robota. Time se značajno komplicira proces rekonfiguracije što je osobitno vidljivo kod velikog broja modula. Primjer lančanog tipa modularnog robota je PolyBot, o kojem se više može pronaći u [13].



Slika 2.1Nacrt i dimenzije DTTO v2 [10]

DTTO, kao i M-TRAN, ubrajamo u skupinu hibridnih rekonfigurabilnih robota (engl. *hybrid type*). Ova skupina kombinira prednosti prethodno opisanih tipova, pa tako zadržava mogućnost slobodnog kretanja karakterističnu za lančani tip, ali prilikom rekonfiguracije, kada se moduli nastoje spojiti, mogu zauzeti samo diskretne pozicije u prostoru što je karakteristično za rešetkasti tip rekonfigurabilnih robota.

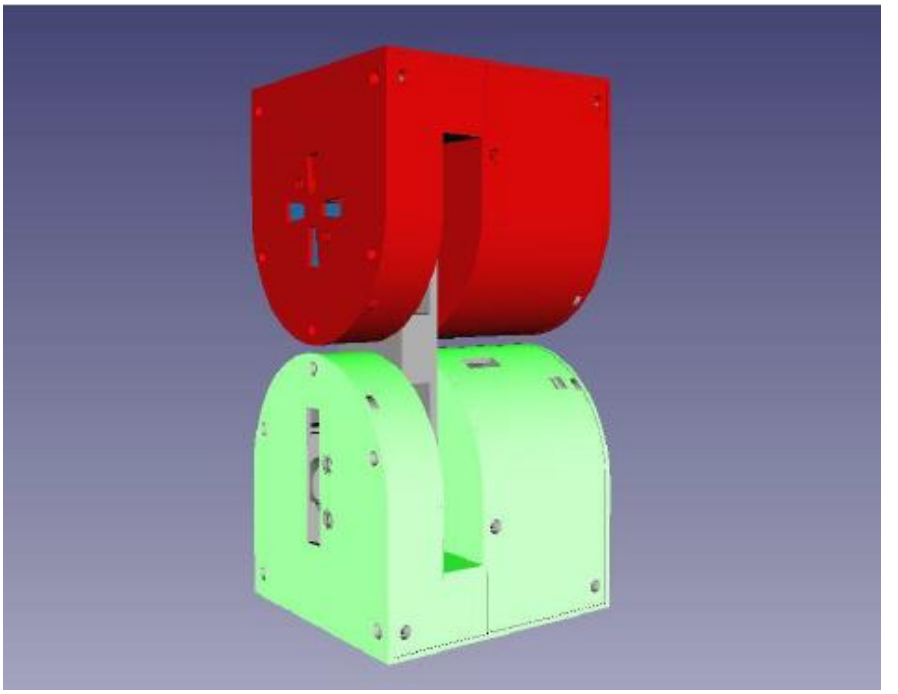
Jedan DTTO v2 modul sastavljen je od ukupno 22 dijela koji su printani Ultimaker 2 Extended + 3D printerom [14], čine osnovnu konstrukciju robota. Osim vanjske konstrukcije, svaki DTTO modul sadrži i:

- 1x Arduino Nano
- 2x TowerPro MG92B servomotora
- 3x TowerPro MG90s servomotora
- 1x Bluetooth HC-05 modul
- 1x NRF2401 modul
- 2x Li-Po baterije 3,7V 600mAh
- 1x LM317 regulator napona
- 24x Neodimijska magneta
- 40x M1.7x4mm vijka.

Arduino Nano je središnji upravljački uređaj robota. Na njega su spojeni svi aktuatori i komunikacijski moduli te implementirani algoritmi upravljanja. Dva TowerPro MG92B servomotora spojena su na središnju osovinu koja povezuje muški i ženski segment modula. Odgovorni su za gibanje modula, te omogućuju njegovo gibanje u smjeru samo jedne osi, tj. jedan stupanj slobode (engl. *degree of freedom - DOF*). Tri TowerPro MG90s servomotora, u kombinaciji s magnetima služe kao mehanizam za spajanje i razdvajanje modula, uz određena ograničenja mogu omogućiti i dodatan stupanj slobode. Za komunikaciju među modulima koriste

se bluetooth i RF modul, dok je napajanje modula osigurano iz dvije litij-polimer baterije.

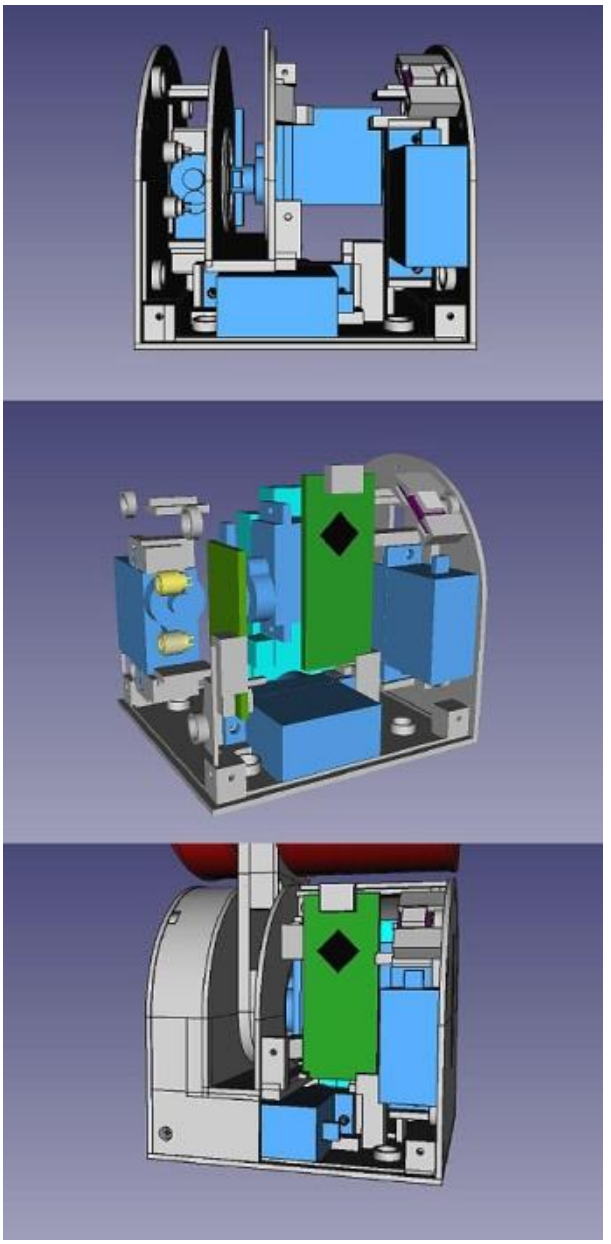
Modul je dizajniran tako da je jedan segment aktivan, a drugi pasivan. U aktivnom modulu, na *Slici 2.2* osjenčan zelenom bojom, osim jednog pogonskog servomotora smještena je većina elektroničkih komponenti kao i tri servomotora mehanizma za spajanje.



Slika 2.2Aktivni segment DTTO v2 [6]

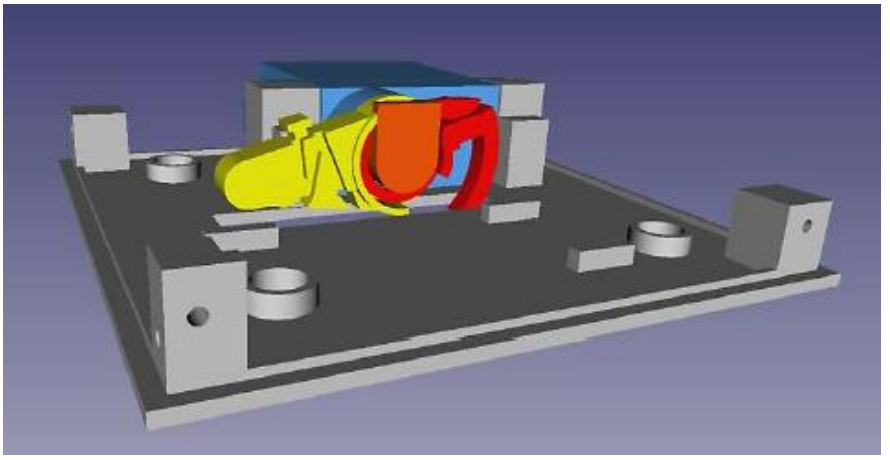
Smještaj komponenti unutar aktivnog segmenta prikazan je na *Slici 2.3.* Sa slike je vidljivo da je najveći izazov pri dizajnu modula bio smjestiti sve komponente unutar ograničenog prostora, pazeći da mnoštvo kabela ne ograničava kretnju modula i ne ometa mehanizam za spajanje. Posebna pozornost posvećena je odabiru i duljini kabela te njihovom smještaju unutar segmenta. Kako bi se lakše moglo pristupiti unutrašnjosti segmenta, dizajniran je tako da

se polovica segmenta otvara uklanjanjem samo tri vijka, bez potrebe rastavljanja ostatka segmenta.



Slika 2.3Unutrašnjost aktivnog segmenta DTTO v2 [6]

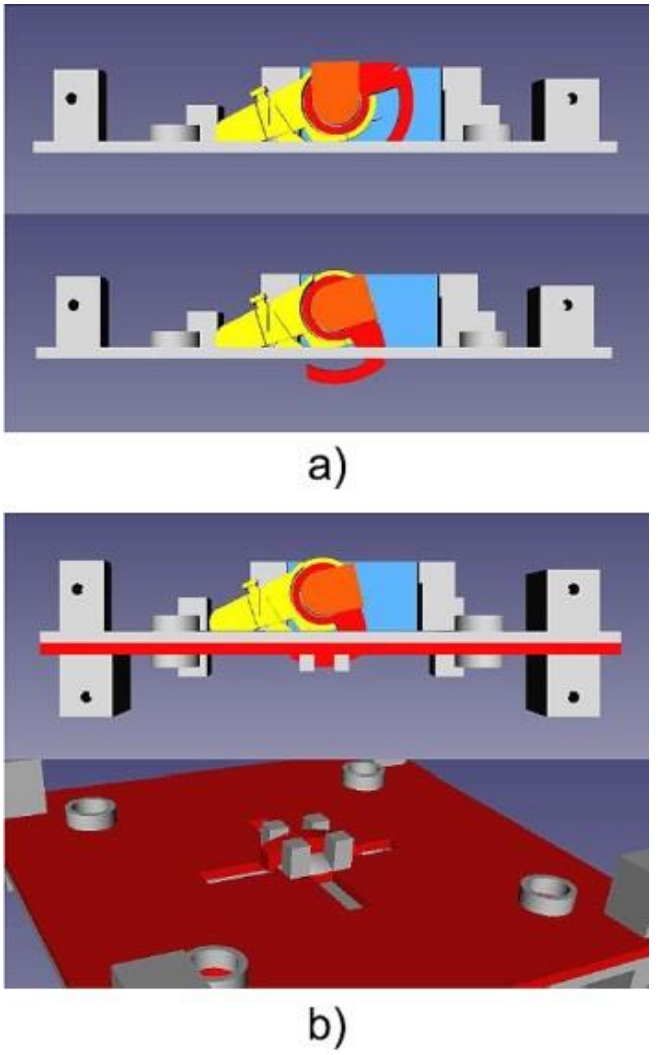
Aktivni segment, za razliku od pasivnog, sadrži tri servomotora sustava za spajanje. Sustav za spajanje omogućuje povezivanje više modula te je ključan u procesu rekonfiguracije. Sastoji se od ukupno 24 neodimijska magneta i tri servomotora. Magneti su zaduženi za poravnanje modula prilikom spajanja kako bi se osiguralo da su moduli međusobno točno na željenoj pozicji. Poravnavanje je glavni preduvjet da mehanizam povezivanja obavi svoju funkciju. Mehanizam za povezivanje sastoji se od servomotora, kukice za povezivanje, poluge za razdvajanje i elastične gumice. On osigurava čvrstu mehaničku vezu između modula, a može je i raskinuti. Mehanizam za povezivanje prikazan je na *Slici 2.4*.



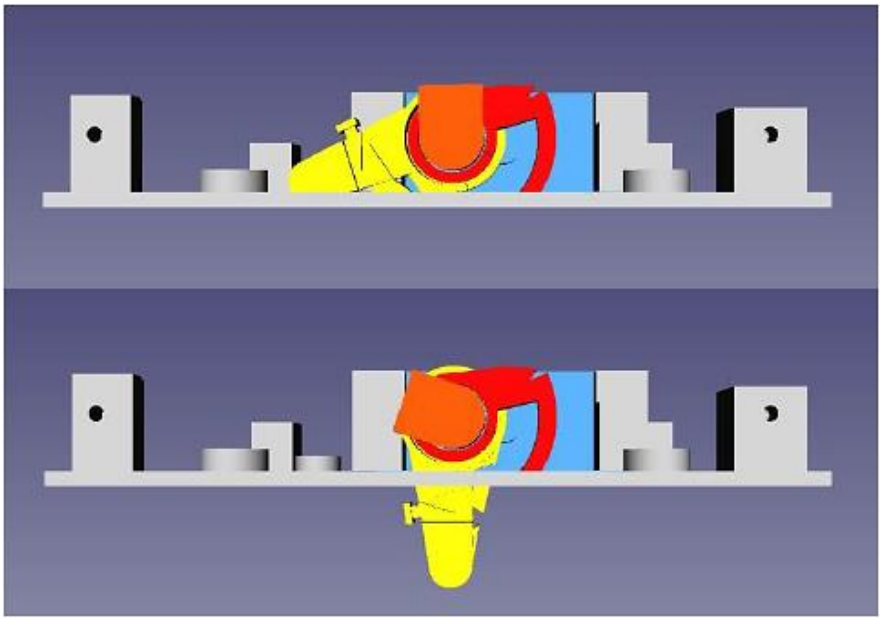
Slika 2.4Mehanizam za povezivanje DTTO v2 [6]

Kada se dva modula približe i magnetima se osigura da su u ravnini, rotacijom željenog servomotora mehanizma za povezivanje za 90 stupnjeva u smjeru kazaljke na satu crvena kukica će se zarotirati, izaći iz aktivnog dijela modula te se zakačiti na odgovarajući dio pasivnog dijela drugog modula s kojim se želi ostvariti povezivanje, kao što je prikazano na *Slici 2.5*.

Kada se module želi razdvojiti, prvo je potrebno mehanizam vratiti u početni položaj rotacijom za 90 stupnjeva u smjeru suprotnom od kazaljke na satu, te ponovno zarotirati mehanizam za 90 stupnjeva u smjeru suprotnom od kazaljke na satu. Rezultat će biti pomicanje poluge van tijela modula koja će drugi modul odgurnuti od sebe silom dovoljnom za svladavanje privlačne magnetske sile među modulima. Opisani proces prikazan je na *Slici 2.6.*

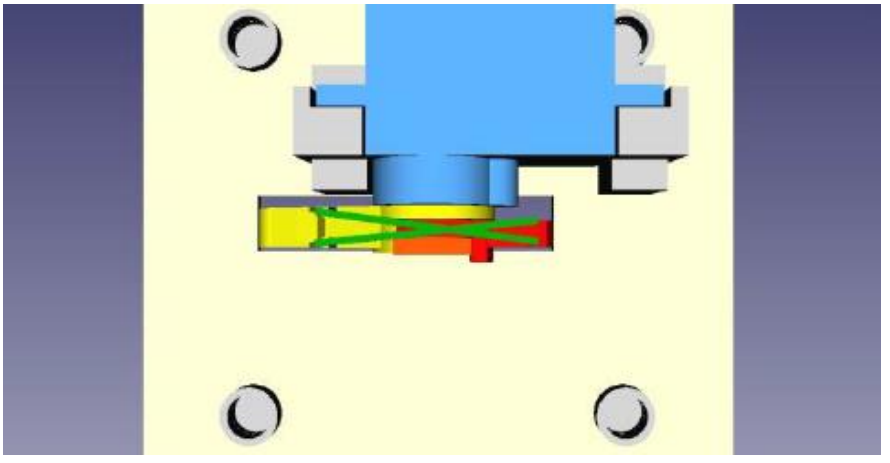


Slika 2.5a) rotacija kukice b) povezivanje dva modula [6]



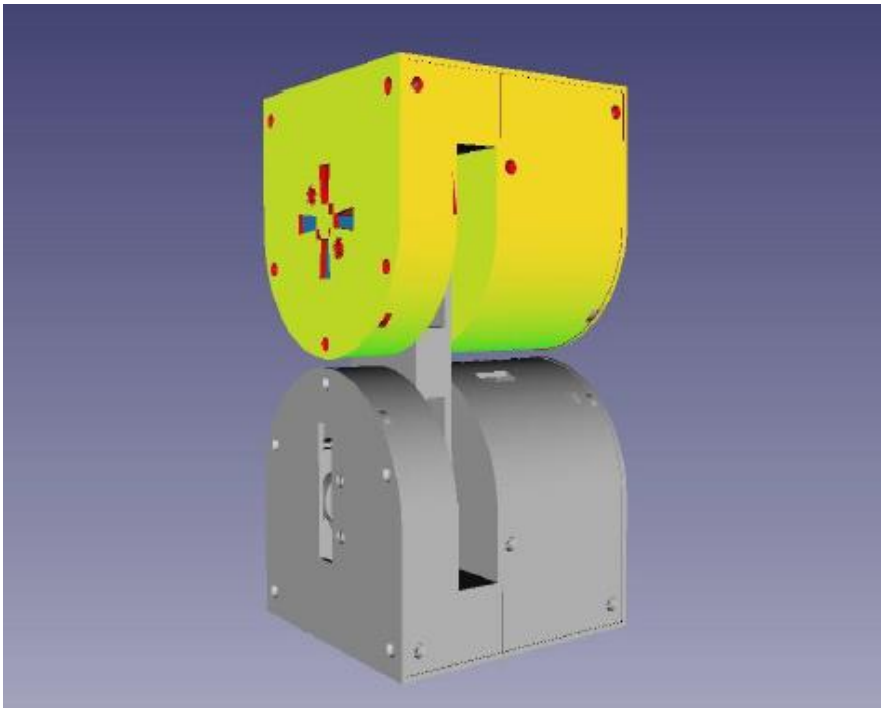
Slika 2.6 Mehanizam za razdvajanje [6]

Elastična gumica, prikazana na *Slici 2.7*, sastavni je dio mehanizma za povezivanje, smještena je između poluge i kukice mehanizma. Njena uloga je osigurati početni položaj mehanizma kada se servomotor nalazi u neutralnom položaju.



Slika 2.7 Elastična gumica mehanizma za povezivanje [6]

U pasivnom segmentu, na *Slici 2.8* osjenčan zelenom bojom, smješteni su samo drugi pogonski motor i baterije.



Slika 2.8 Pasivni segment DTTO v2 [6]

Nejednolik raspored komponenti između segmenata omogućuje ugradnju komponenti u pasivni segment prema potrebi. To mogu biti dodatne baterije, Arduino, razni senzori i slično. Pravilnim rasporedom komponenti unutar pasivnog segmenta može se postići ravnomjerna raspodjela težine modula, koja u ovom obliku nije postignuta.

Poveznicu između aktivnog i pasivnog segmenta čini osovina modula, prikazana na *Slici 2.9*. Na nju su pričvršćena oba pogonska servomotora, a kroz njenu unutrašnjost, do baterija koje se nalaze u pasivnom segmentu, provučeni su kabeli za napajanje i kabeli drugog pogonskog servomotora. Činjenica da je većina komponenti

koncentrirana unutar aktivnog segmenta značajno smanjuje broj kabela koji prolaze osovinom modula.

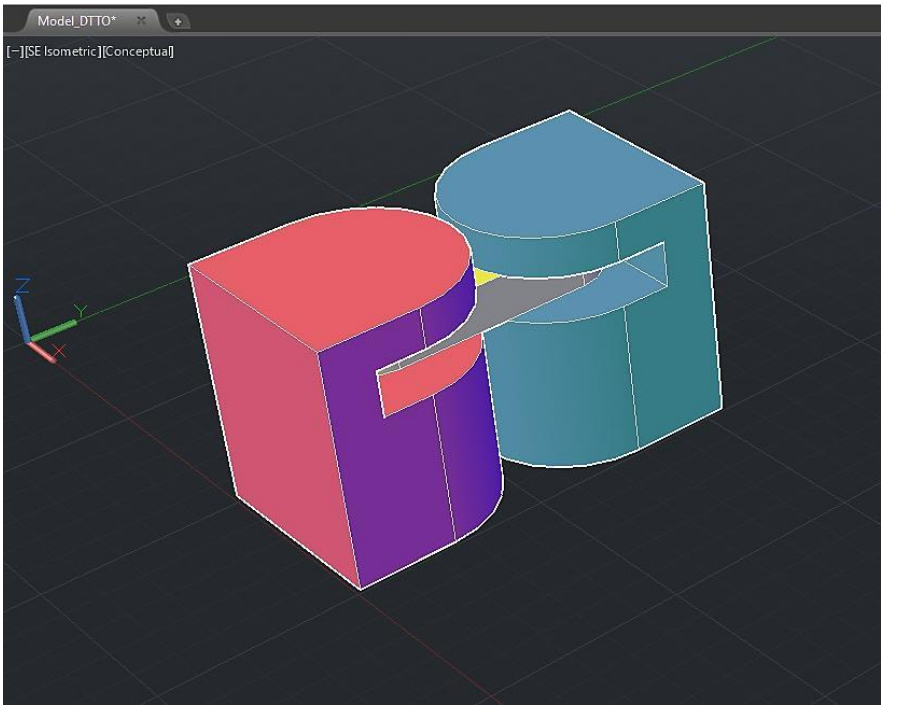
Slika 2.9 Osovina DTTO v2 [6]

2.2. Modeliranje modularnog DTTO robota

V-REP je vritualna platforma za razvoj, simulaciju i testiranje robota i rješenja vezanih uz robotiku i automatizaciju. Osim što sadrži već razvijene modele poznatih robota, ugrađeno razvojno okruženje V-REP -a omogućuje relativno jednostavan razvoj novih modela robota, njihovu vizualizaciju i dinamičko testiranje u proizvoljno namještenim scenarijima [15]. Unutar V-REP-a koristi se Lua programski jezik, ali omogućeno je povezivanje i s vanjskim aplikacijama koje koriste druge programske jezike, kao što su C, Java, Python, Matlab, Octave i Urbi, a također je omogućeno i sučelje prema ROS-u (Robot Operating System). Sve navedeno čini V-REP vrlo moćnim alatom za brzi razvoj i testiranje rješenja vezanih uz robotiku i automatizaciju.

Iako V-REP sadrži osnovne geometrijske oblike koji se mogu dodavati pri razvoju novog modela, za složenije oblike ipak je najjednostavnije koristiti neku prikladniju aplikaciju. Tako je pri izradi modela DTTO-a korišten AutoCAD. U AutoCAD-u je prema službenim nacrtima napravljen vjeran 3D model DTTO robota koji je potom unešen u V-REP. Izgled AutoCAD modela prikazan je na *Slici 2.10.* CAD model poslužio je kao osnova za razvoj dinamičkog modela DTTO robota. Pri izradi modela robota potrebno je pridržavati se određenih načela i pravila koje je propisala Coppeliarobotics, autor V-REP-a, kako bi se dobio brz, stabilan i vizualno atraktivan simulacijski model. Razvoj modela robota u V-REP-u može se podijeliti u četiri faze:

- Izrada vizualnog modela
- Izrada aktuatora
- Izrada dinamičkog modela
- Definiranje modela



Slika 2.10 CAD model DTTO robota

Pri izradi vizualnog modela najvažnije je da prethodno ubačeni CAD model nije pretežak jer će simulacija biti vizualno usporena, a simulacijski proračuni će se sporije izvršavati. Zato je potrebno s modela ukloniti detalje koji nemaju utjecaj na njegovo ponašanje. Imajući to na umu, i sam CAD model je dizajniran minimalistički pa nije bilo potrebe za dodatnim pojednostavljenjima, iako V-REP nudi razne alate koji to omogućuju prema potrebi .

Sljedeći korak u izradi modela je dodavanje aktuatora. U V-REP-u postoje četiri vrste aktuatora: rotacijski (engl. *revolute*), translacijski (engl. *prismatic*), vijčani (engl. *screws)* i kuglasti (engl. *spherical*). U slučaju DTTO-a korištena su dva rotacijska aktuatora koja će simulirati dva glavna servomotra.

Aktuatori su postavljeni u *Torque or force mode* te njihovo ponašanje ovisi o proračunu dinamike V-REP-a. U ovom modu postoji više načina rada:

- Motor je onemogućen – motor se slobodno može rotirati, a gibanje je ograničeno samo zadanim granicama.
- Motor je omogućen, a upravljačka petlja je onemogućena – motor će nastojati dosegnuti zadanu brzinu uz maksimalan okretni moment koji mu je dostupan. Ako je okretni moment jako velik, zadana brzina će se postići gotovo trenutno te se tada upravlja brzinom motora. U suprotnom će motor davati definirani okretni moment sve dok se ne dosegne željena brzina, te se tada upravlja okretnim momentom motora.
- Motor je omogućen, upravljačka petlja je omogućena – tada korisnik na raspolaganju ima tri vrste upravljanja:
    - Motorom se može proizvoljno upravljati iz skripte za upravljanje motorom (engl. *joint control callback script*).
    - Može se upravljati pozicijom motora pomoću PID regulatora zaduženog za upravljanje brzinom motora koristeći jednadžbu (2.1):

$$v = \frac{K_p e_i + K_i \sum e_i \Delta t + K_d (e_i - e_{i-1})/\Delta t}{\Delta t} \qquad (2.1)$$

      uz konstantnu silu, a gdje su K_p, K_i i K_d proizvoljno odabrani parametri regulatora.
    - Motor se ponaša poput sustava opruge i prigušivača, gdje će mu brzina biti konstantna, a sila će se računati po formuli (2.2):

$$F = K e_i + C(e_i - e_{i-1})/\Delta t \qquad (2.2)$$

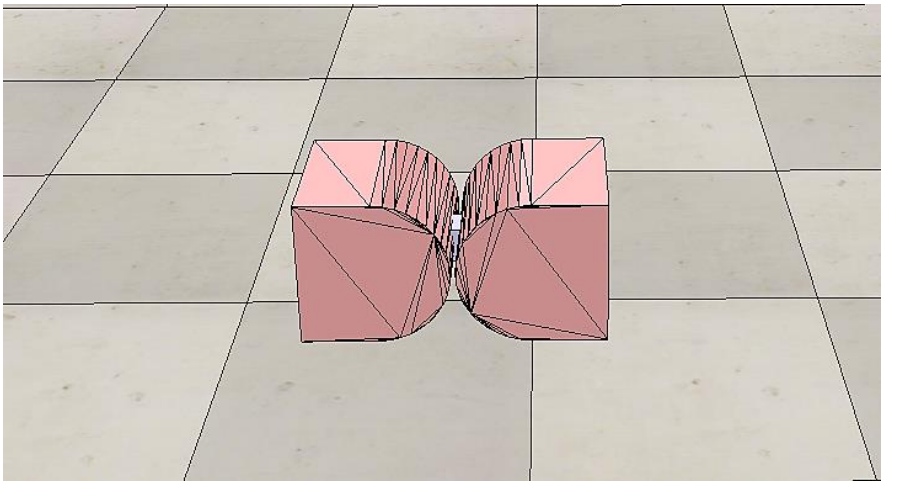
      gdje su K i C proizvoljno odabrani parametri sustava.

Obzirom da je DTTO aktuiran upravljanjem pozicijom dvaju glavnih servomotora i u simulacijskom modelu koristi se upravljanje pozicijom aktuatora. Iz toga razloga aktuatori su postavljeni u *Torque or force mode* s omogućenom upravljačkom petljom te se njihovom pozicijom upravlja pomoću PID regulatora prema referentnoj poziciji proslijeđenoj iz glavne upravljačke skripte (engl. *child script*).

Nakon dodavanja aktuatora, sljedeći korak u krivulji razvoja V-REP modela robota je izrada dinamičkog modela. Ovo je ključni korak u dizajnu ako želimo da robot pravilno reagira na sudare, padove i slično. Proračun dinamike odnosi se na sve oblike u sceni koji su dinamički omogućeni, a to znači da će na njih djelovati sve vanjske sile i momenti. Za svaki oblik se može naknadno odrediti hoće li reagirati na sudare ili neće odabirom *respondable* ili *non-respondable* opcije u postavkama oblika.V-REP razlikuje pet vrsta oblika koji se mogu dodati u scenu, a to su:

- *Pure shapes* – jednostavni geometrijski oblici, kao što su kocke, kugle, cilindri i slično.
- *Pure compound shapes*–skupina jednostavnih oblika.
- *Convex shapes* – konveksni oblici.
- *Compound convex shapes* – skupina konveksnih oblika.
- *Random shapes* – nepravilni oblici.

Ovisno o vrsti oblika ovisit će vrijeme i stabilnost proračuna dinamike. Što je oblik jednostavniji, izvedbe će biti bolje pa je poželjno da se što više koriste prve dvije skupine opisanih oblika. Iz tog razloga za proračun dinamike koristit će se pojednostavljeni oblik CAD modela, dok će se za vizualizaciju koristiti već uneseni CAD model. Izgled pojednostavljenog oblika DTTO robota, koji će služiti za proračun dinamike, prikazan je na *Slici 2.11*.

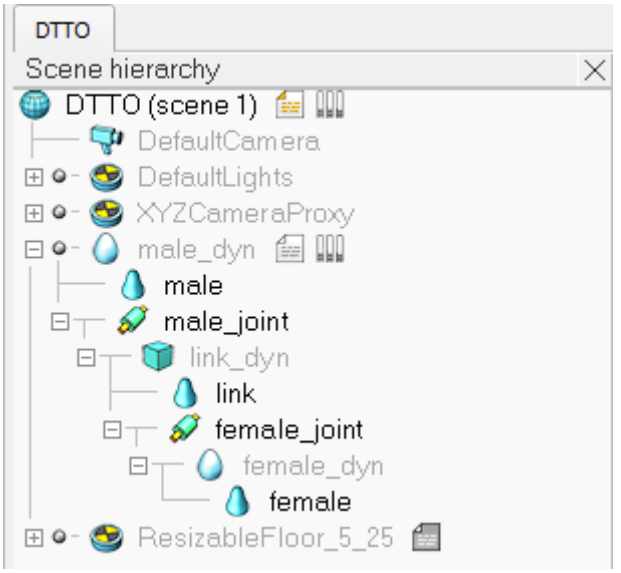


Slika 2.11Dinamički model DTTO v2 robota

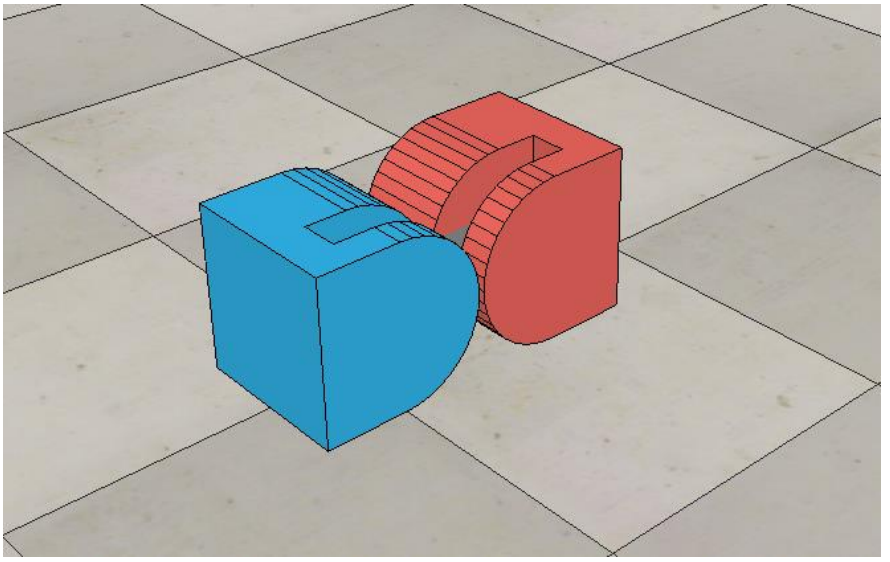
Pojednostavljenjem je originalni unešeni CAD model, koji pripada skupini nepravilnih oblika, pretvoren u jednostavni konveksni oblik koji je optimiziran za dinamičke proračune. Sa *Slike 2.11* je vidljivo da dinamički model izgledom nije istovjetan stvarnome modelu robota. Ovo pojednostavljenje je bilo potrebno napraviti kako bi se dobio stabilan i brz dinamički proračun, a u simulacijama nije značajno utjecalo na ponašanje robota. Kada je dinamički model završen, preostaje posljednji korak u razvoju modela DTTO robota, a to je definiranje modela.

Definiranje modela podrazumijeva izradu hijerarhijske strukture između dinamičkih i odgovarajućih vizualnih dijelova robota te njihovo povezivanje aktuatorima. Ovaj korak može se zamisliti kao sastavljanje virtualnog robota. Povezivanje se ostvaruje uspostavom roditelj – dijete (engl. *parent – child*) veze između odgovarajućih dijelova. Izgled izrađene hijerarhijske strukture DTTO robota prikazan je na *Slici 2.12*. Pri definiranju modela važno je voditi računa o nazivima dijelova strukture. Potrebno je da budu jednostavni i prepoznatljivi kako bi pisanje koda koje slijedi nakon izrade modela bilo što lakše, kako za programera, tako i za ostale

zainteresirane strane koje bi taj kod proučavale. Na *Slici 2.13* prikazan je konačan V-REP model DTTO v2 robota.



Slika 2.12 Hijerarhijska struktura DTTO v2 robota



Slika 2.13 V-REP model DTTO v2 robota

3. PRINCIP GIBANJA MODULARNIH ROBOTA

Modularnost omogućuje modularnim robotima različite mogućnosti gibanja koje ovise o broju dostupnih modula i načinu njihove rekonfiguracije. Zahvaljujući tom svojstvu, modularni roboti mogu se prema potrebi prilagođavati terenu po kojem se gibaju i uvjetima koji na njemu vladaju, što ih čini iznimno zanimljivima za potrebe istraživanja i akcije traganja i spašavanja. Tisuće godina evolucije dovele su do to toga da su stvorenja iz naše okoline gotovo savršeno prilagođena uvjetima u kojima žive. Zato se robotika i srodne znanosti pri traženju rješenja za određeni problem često oslanjaju na primjere iz prirode. Ponekad su to pokušaji oponašanja lokomotornih sposobnosti pojedinih životinja, ali u posljednje vrijeme sve se više nastoji algoritmima opisati ponašanje jedinki i skupina te implementirati rješenja iz njihovog svijeta u ljudsku okolinu [16] [17]. Za modularnu robotiku podjednako su zanimljiva oba područja – lokomotorni sustavi iz perspektive gibanja robota, a proučavanje ponašanja skupina iz perspektive suradnje jedinki na ostvarivanju zajedničkog cilja i grupne inteligencije.

Gibanje po kopnu u prirodi možemo podijeliti na dva načina: gibanje uz pomoć udova i gibanje stezanjem i opuštanjem tijela. Prvi način gibanja karakterističan je za kukce i sisavce, dok se drugim služe zmije i gusjenice. Već spomenuta prilagodljivost modularnih robota omogućuje oba načina gibanja pa se uz dovoljan broj modula može ostvariti prvi način, ali u ovome radu će se usredotočiti na drugi jer je osnova za gibanje svakog pojedinog modula.

Zmije postoje diljem našeg planeta, a njihova karakteristična anatomija omogućuje im kretanje po najrazličitijim podlogama – betonu, travi, pijesku, drveću, stijenama pa čak i vodi. Stoga ne čudi da su poslužile kao inspiracija za izradu robota u samim začetcima

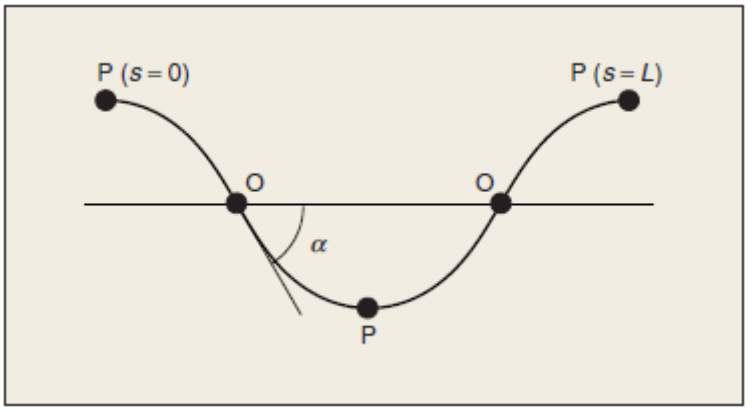
moderne robotike. Među pionirima na ovome području je profesor Hirose sa Instituta za tehnologiju u Tokiju. U svome radu *Biologically Inspired Robots [18]* objavljenom 1993. u Oxford University Press predstavio je krivulju kojom je opisao gibanje zmije (engl. *serpenoid curve*). Glavna karakteristika ove krivulje je da se zakrivljenost krivulje duž krivulje mijenja po sinusnom zakonu, što se može predstaviti sljedećom jednadžbom [19:90]:

$$K(s) = -\frac{2\pi\alpha}{L}\cos\left(\frac{2\pi}{L}s\right) \tag{3.1}$$

gdje je:

- α – kut zakrivljenosti
- L – valna duljina kriuvlje za vrijeme jednog perioda
- s– udaljenost koja odgovara promatranoj točki na krivulji.

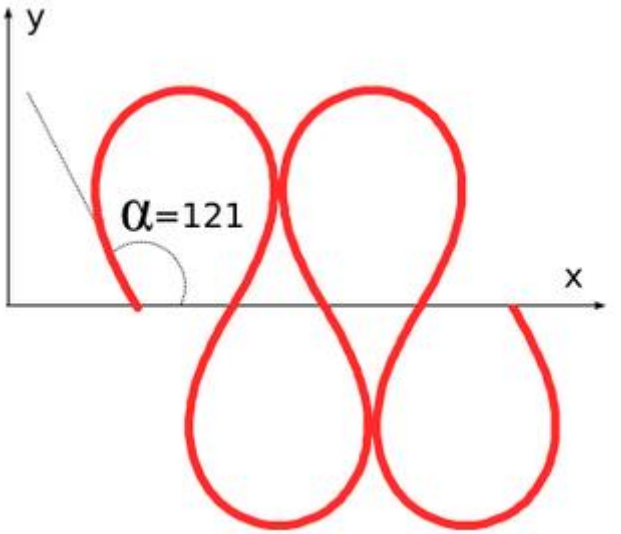
Izgled opisane krivulje dan je *Slici 3.1*.



Slika 3.1 Vijugava krivulja [19]

Kut zakrivljenosti α odgovara kutu pod kojim tangenta ljubi krivulju u točki $s = 0$ i njime je definiran oblik krivulje. Maksimalan kut zakrivljenosti koji je moguće postići, a da ne dođe do preklapanja krivulje sa samom sobom je $\alpha = 121°$ [6:70].

Na *Slici 3.2* prikazana je krivulja s maksimalnim kutem zakrivljenosti $\alpha = 121°$.



Slika 3.2 Vijugava krivulja za α=121°[20]

Zakrivljenost krivulje $K(s)$ se definira kao promjena kuta pod kojim je povučena tangenta na krivulju, odnosno [6:69]:

$$K(s) = \frac{d\alpha_s}{ds} \tag{3.2}$$

Konačno, krivulju je moguće opisati i točakma koje je definiraju u prostoru, pa prema tome koordinate krivulje u Kartezijevom koordinatnom sustavu mogu se dobiti koristeći sljedeće izraze [20]:

$$x(s) = \int_0^s \cos\left(\alpha\cos\left(\frac{2\pi}{L}s\right)\right) ds \tag{3.3}$$

$$y(s) = \int_0^s \sin\left(\alpha\cos\left(\frac{2\pi}{L}s\right)\right) ds \tag{3.4}$$

3.1. Implementacija gibanja

Kako za izraze (3.3) i (3.4) ne postoji analitičko rješenje, nego ih je potrebno rješavati numerički, razvijeno je više metoda koje ih dovoljno dobro aproksimiraju. Cilj tih metoda je što jednostavnija implementacija opisanog gibanja u najnižu razinu upravljanja, odnosno algoritme za kontrolu aktuatora robota. Neka od rješenja su [6:71]:

- Ručno zadavanje pozicija – potrebno je razviti niz tablica u kojima će se definirati kut koji svaki aktuator mora zauzeti u određenom trenutku. Ovo je vrlo mukotrpno rješenje i skoro je neizvedivo za veći broj modula.
- Inverzna kinematika – potrebno je raditi proračun za sljedeću poziciju aktuatora ovisno o njegovom trenutnom položaju i orijentaciji.
- Sinusnom aproksimacijom – prema Fourieru, periodična funkcija može se aproksimirati sumom sinusnih funkcija različitih amplituda, faza i frekvencija. To vrijedi i za vijugavu krivulju. Ranije je spomenuta osnovna karakteristika vijugave krivulje: zakrivljenost vijugave krivulje duž krivulje mijenja se po sinusnom zakonu. Upotrebnom sinusnih funkcija s pravilnom odabranim parametrima može se postići upravo opisano svojstvo. Prema tome, ako se pravilno smješteni aktuatori robota pobude sinusnim signalima pravilno odabranih parametara, osigurat će se gibanje robota koje aproksimira vijugavu krivulju [21].

Treća metoda odabrana je kao rješenje za aktuaciju servomotora DTTO robota. Kako kod simulacijskog modela, tako i kod realnog robota, pogonski servomotori upravljani su po poziciji. Pozicije aktuatora mijenjaju se po sinusnom načelu, pa je upravljački signal koji se daje aktuatorima oblika:

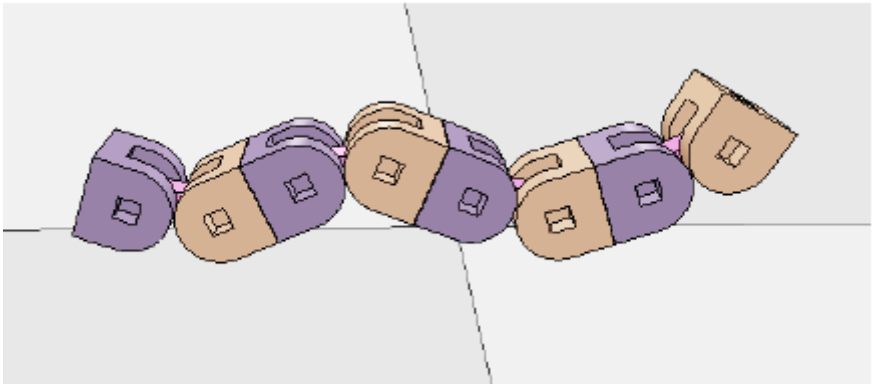
$$p = A\sin(wt + \theta + \theta_s) \qquad (3.5)$$

gdje je:

- p – pozicija aktuatora
- A – amplituda sinusnog signala
- w – kružna frekvencija $\left[\frac{rad}{s}\right]$
- t – vrijeme $[s]$
- θ –faza sinusnog signala $[rad]$
- θ_s–pomak u fazi sinusnog signala u odnosu na drugi aktuator $[rad]$

Promjenom navedenih parametara utječe se na brzinu gibanja robota, kao što će biti pokazano u *Potpoglavlju 3.2*. Simulacije će se vršiti u V-REPu.

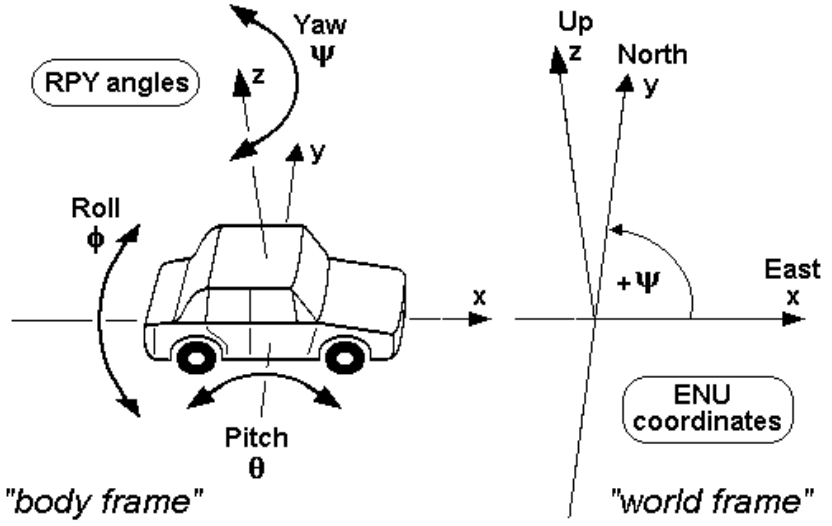
Osim što je pogodna za aktuaciju svakog pojedinog modula, opisana sinusna metoda aktuacije može se koristiti i za gibanje više spojenih modula u zmijastoj konfiguraciji [6]. Pri tome je potrebno implementirati sinusno načelo upravljanja na aktuatore svakog modula te osigurati pomak u fazi između njih, ali i aktuatora susjednih modula. Na taj se način realizira putujući sinusni val duž tijela robota, kao što je prikazano na *Slici 3.3.*



Slika 3.3 Zmijasta konfiguracija četiri DTTO modula [6]

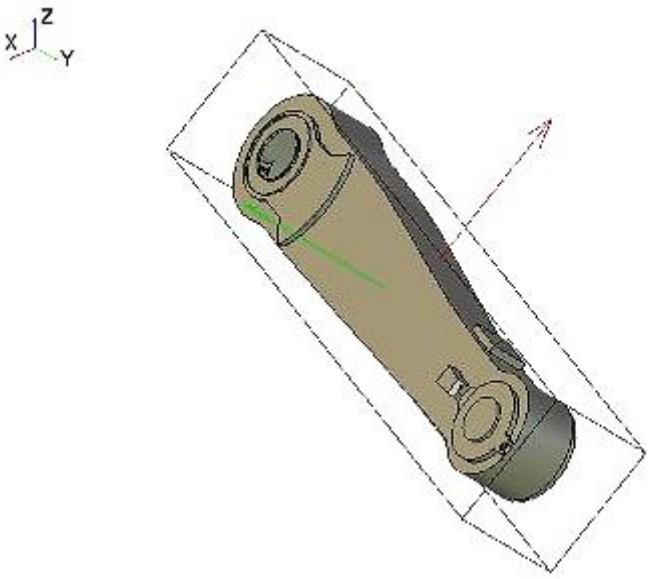
3.2.Simulacija gibanja

V-REP koristi ENU (East-North-Up) konvenciju za referentni koordinatni sustav. Za nju vrijedi da je pozitivan smjer X osi u smjeru gibanja, pozitivan smjer Y osi je usmjeren na lijevo od tijela, dok pozitivan smjer Z osi pokazuje prema gore, kao što je prikazano na *Slici 3.4*.



Slika 3.4 ENU konvencija koordinatnog sustava [22]

Svaki oblik i predmet u V-REP sceni ima svoj referentni okvir i prividni pravokutnik koji ga omeđuje. Koordinatni sustav tijela sastoji se od tri koordinatne osi *x, y* i *z* koje su redom označene crvenom, zelenom i plavom bojom. On je uvijek smješten u geometrijsku sredinu tijela iz koje se računa pozicija i orijentacija tijela. Postoji više načina za određivanje orijentacije referentnog okvira u odnosu na položaj tijela, a za sve simulacije koristit će se konvencija kod koje je referentni okvir tijela u ravnini s glavnim osima tijela, kao što je prikazano na *Slici 3.5*.



Slika 3.5 Konvencija za orijentaciju referentnog okvira tijela [23]

Eulerovi kutevi su tri kuta koja opisuju orijentaciju krutoga tijela. Postoji dvanaest različitih konvencija za Eulerove kutove, a V-REP koristi sljedeću: rotacija tijela definirana je preko tri elementalne rotacije koje se opisuju kutevima *alpha*, *beta* i *gamma*, na sljedeći način:

$$Q = R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma) \tag{3.6}$$

gdje su $R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma)$ elementalne rotacije oko *x*, *y* i *z* koordinatnih osi referentnog koordinatnog sustava. Jednaka transformacija može se postići rotacijom oko vlastitog referentnog okvira sljedećim redoslijedom: rotacijom za *alpha* oko vlastite *x* osi, potom rotacijom za *beta* oko vlastite *y* osi i konačno rotacijom za *gamma* oko vlastite *z* osi [24]. Kut *alpha* odgovara kutu valjanja (engl. *roll*), kut *beta* odgovara kutu posrtanja (engl. *pitch*), a kut *gamma* odgovara kutu zaošijanja (engl. *yaw*).

Inicijalna simulacijska scena postavljena je kako slijedi:

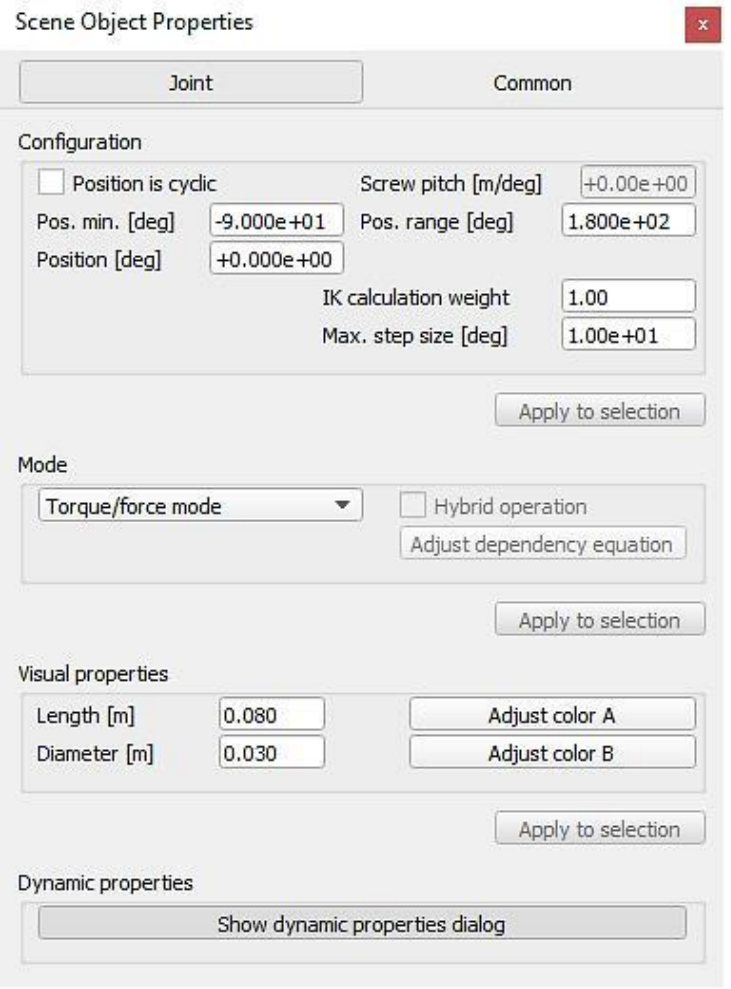
- Kao referentni objekt za poziciju i orijentaciju DTTO robota odabran je muški segment robota.
- Inicijalna pozicija robota je u ishodištu referentnog koordinatnog sustava.
- Smjer gibanja robota je u smjeru X osi referentnog koordinatnog sustava, pri čemu je za pozitivan smjer gibanja odabran smjer gibanja muškog segmenta u pozitivnom smjeru osi.
- Za simulaciju dinamike korišten je Bullet 2.78 engine.
- Promatrat će se i bilježiti podaci o poziciji robota i položaju aktuatora za prvih 60 sekundi simulacije.
- Inicijalni parametri sinusne funkcije su:
  - $A = \frac{\pi}{4}\ rad = 45°$
  - $w = 2\frac{rad}{s}$
  - $\theta_s = \frac{\pi}{2}\ rad = 90°$

  Aktuacija se odvija prema sljedećim jednadžbama:

$$p\_male = A\sin(wt + \theta_s) \tag{3.7}$$

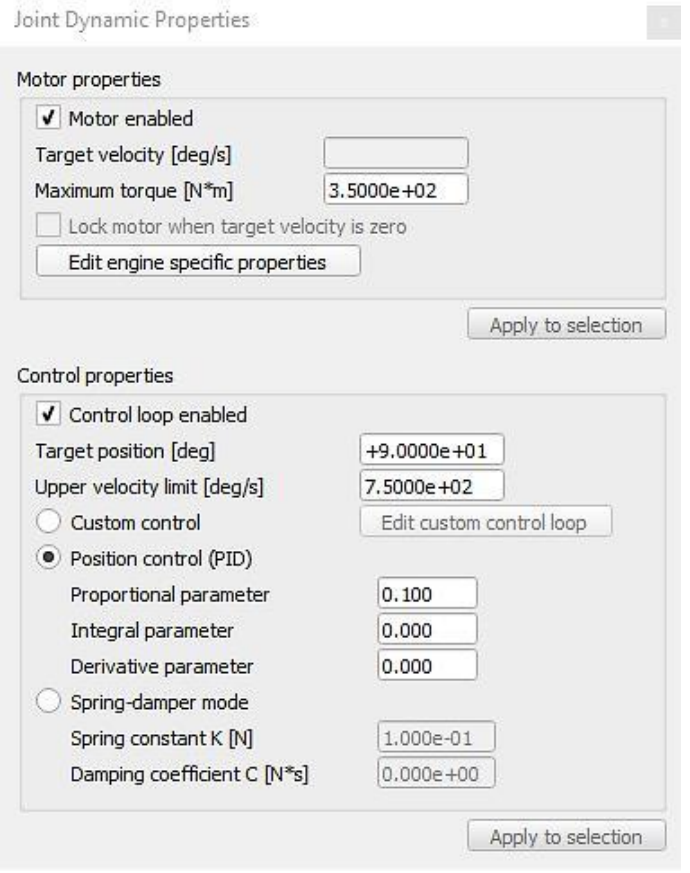
$$p\_female = A\sin(wt) \tag{3.8}$$

gdje p_male označava poziciju aktuatora muškog segmenta u radijanima, a p_female poziciju aktuatora ženskog segmenta, također u radijanima. Iz izraza (3.7) i (3.8) je vidljivo da je pomak u fazi dan aktuatoru muškog segmenta. Svojstva glavnih aktuatora i njihovi dinamički parametri prikazani su na *Slici 3.6* i *Slici 3.7*. Sa slike je vidljivo da hod motora odgovara hodu modificiranih MG92B TowerPro motora [25], o kojima će više riječi biti u *Poglavlju 7*.



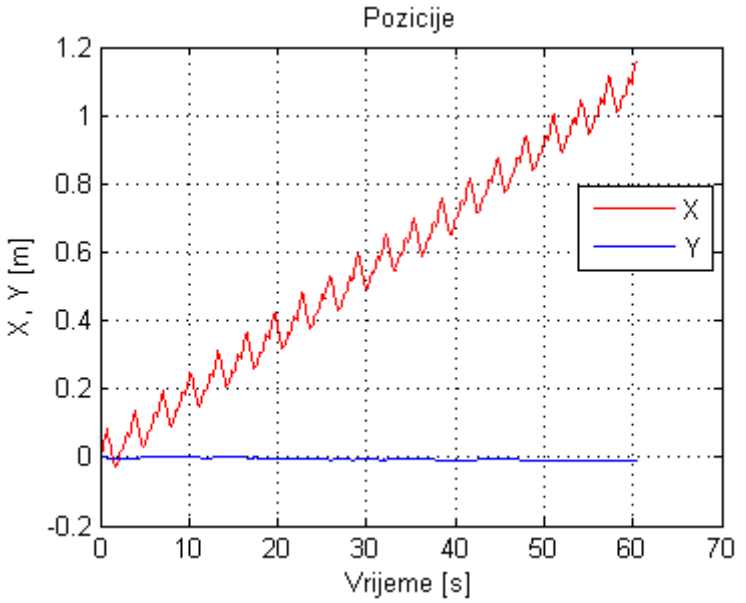
Slika 3.6 Svojstva glavnih aktuatora

Također, može se vidjeti da je brzina aktuatora ograničena na nominalnu brzinu MG92B servomotora, prema podacima dostupnima iz [25]. Okretni moment aktuatora u simulaciji je povećan u odnosu na definirani okretni moment servomotora jer je povećana masa modula kako bi se dobio stabilniji simulacijski model.

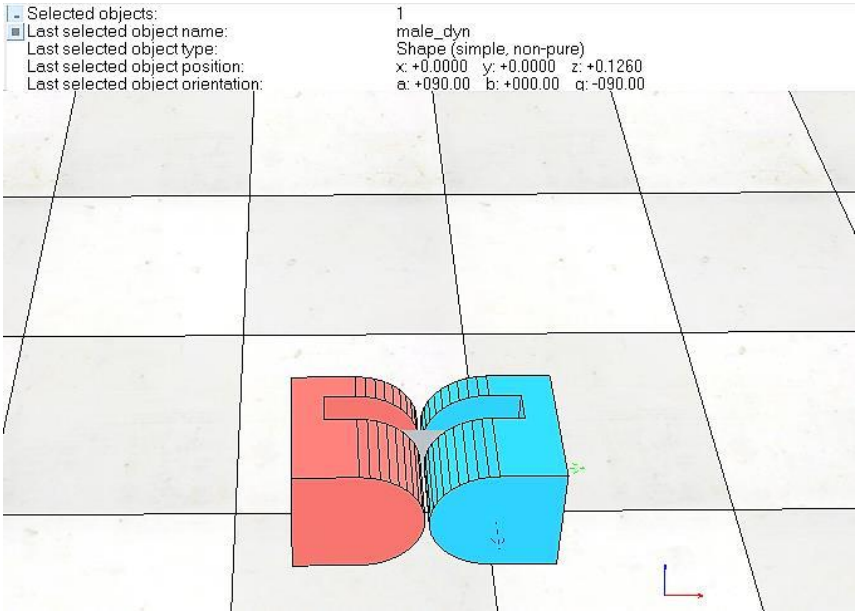


Slika 3.7 Dinamički parametri glavnih aktuatora

Ovako određeni aktuatori, uz već navedene parametre sinusne pobude, DTTO v2 simulacijskom modelu osiguravaju da u 60 sekundi prijeđe udaljenost od 1.116 metara (rezultat je jednak za deset uzastopnih mjerenja), kao što je vidljivo na *Slici 3.8.* To odgovara brzini od $v = 1.86\ \frac{cm}{s}$. Može zaključiti da model dobro aproksimira brzinu realnog robota koja prema [6:88] iznosi $v = 1.63\ \frac{cm}{s}$, ako se u obzir uzme da na brzinu robota značajno utječe i trenje, a podatci o podlozi na kojoj je mjerenje brzine izvršeno nisu dostupni. Izgled testne scene dan je na *Slici 3.9.*



Slika 3.8 Brzina inicijalnog simulacijskog modela



Slika 3.9 Početno stanje simulacije

3.2.1. Utjecaj faznog pomaka aktuatora na gibanje modula

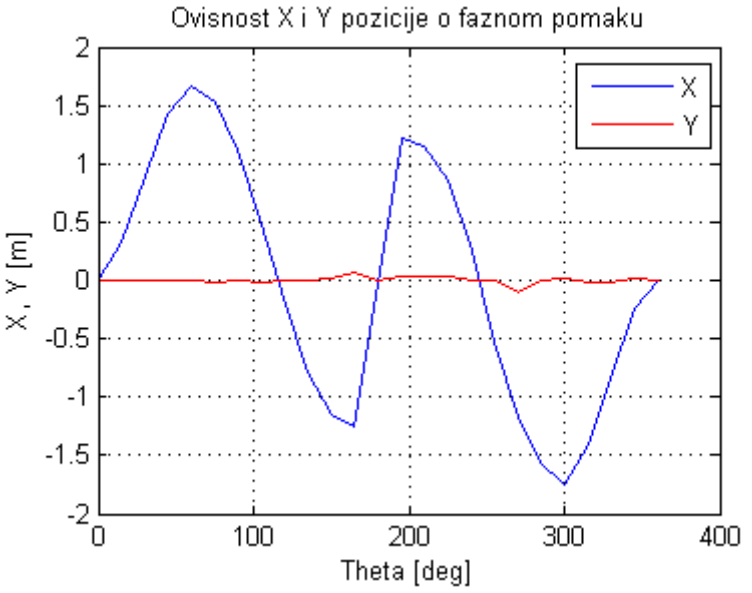
Kako bi se DTTO mogao gibati po prethodno definiranom sinusnom načelu potrebno je postojanje pomaka u fazi između njegovih aktuatora θ_s. Obavljene su simulacije za različite fazne pomake u rasponu od 0° do 360°, uz $A = \frac{\pi}{4}\ rad$ i $w = 2\frac{rad}{s}$, Pozicije modula nakon 60 sekundi simulacije prikazane su u *Tablici 3.1*, izračunata je brzina gibanja modula prema tim podatcima. Za svaki fazni pomak snimljen je graf pozicija modula i pozicija aktuatora.

Tablica 3.1 Ovisnost pozicija modula o faznom pomaku između aktuatora

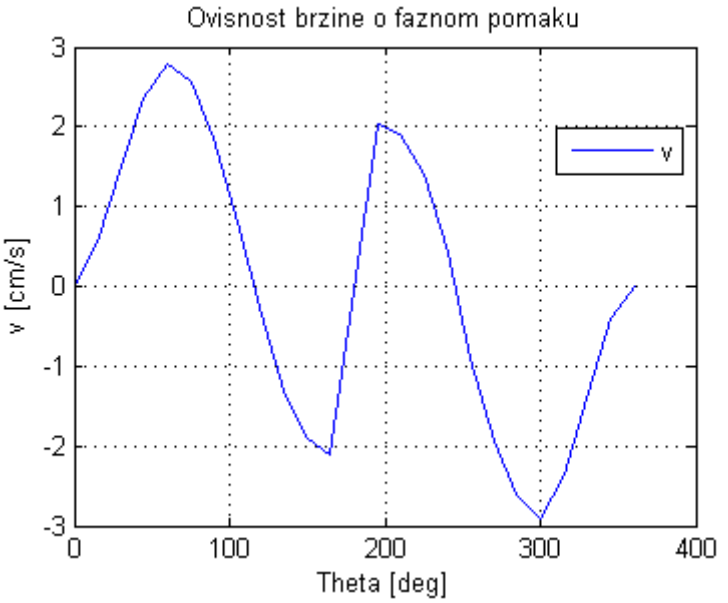
$\boldsymbol{\theta_s}$	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v\ \left[\frac{cm}{s}\right]}$
0	0	0	0
15	0.33	0	0.55
30	0.9	0	1.50
45	1.42	0	2.36
60	1.67	0	2.78
75	1.53	-0.01	2.55
90	1.12	0	1.86
105	0.48	-0.02	0.8
120	-0.21	0	-0.35
135	-0.79	0	-1.31
150	-1.15	0.02	-1.91
165	-1.26	0.07	-2.1
180	-0.04	0	-0.06
195	1.22	0.04	2.03
210	1.15	0.03	1.91

225	0.85	0.04	1.41
240	0.26	0	0.43
255	-0.54	0	-0.9
270	-1.16	-0.1	-1.93
285	-1.56	0	-2.6
300	-1.75	0.01	-2.91
315	-1.41	-0.02	-2.35
330	-0.84	-0.02	-1.4
345	-0.24	0.02	-0.4
360	0	0	0

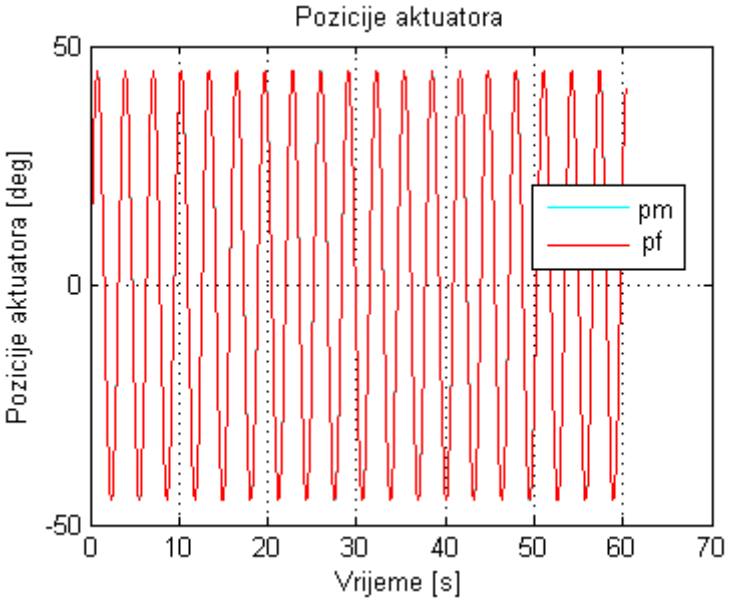
Grafički prikaz rezultata mjerenja pozicija dan je na *Slici 3.10*, a brzine na *Slici 3.11*.



Slika 3.10 Grafički prikaz izmjerenih pozicija

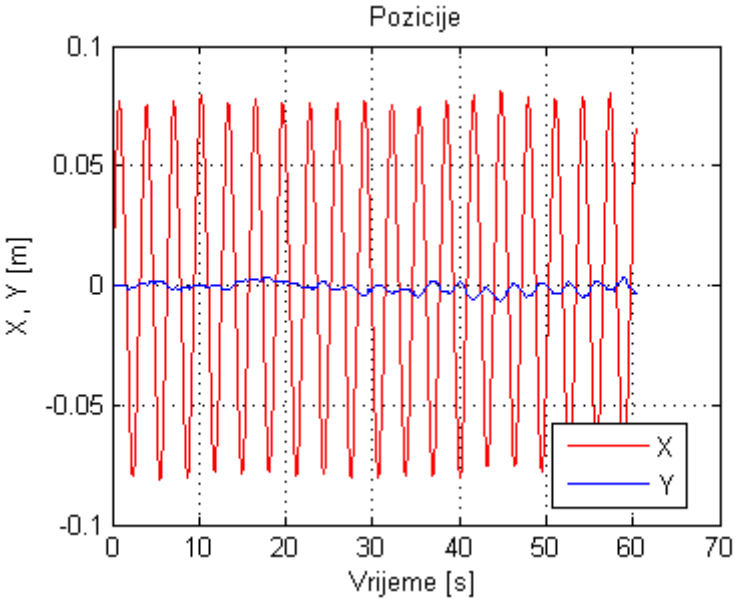


Slika 3.11 Grafički prikaz ovisnosti brzine o faznom pomaku



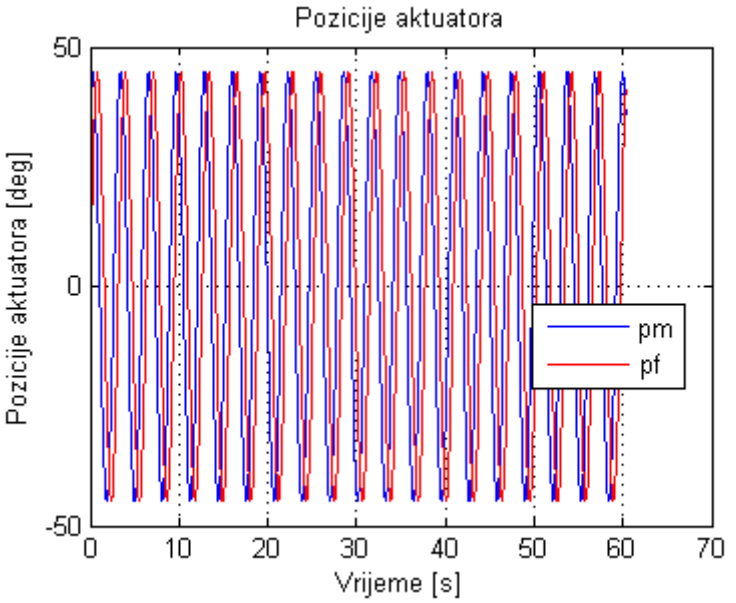
Slika 3.12 Pozicije aktuatora za θ_s=0°

Iz rezultata mjerenja možemo zaključiti da se pozicija robota u ovisnosti o faznom pomaku aktuatora ponaša približno sinusoidalno. Kada faznog pomaka između aktuatora nema, *Slika 3.12*, nije moguće ostvariti ni gibanje robota jer robot oscilira u mjestu oko nultog položaja, kao što je vidljivo na grafu pozicija robota *Slika 3.13*.

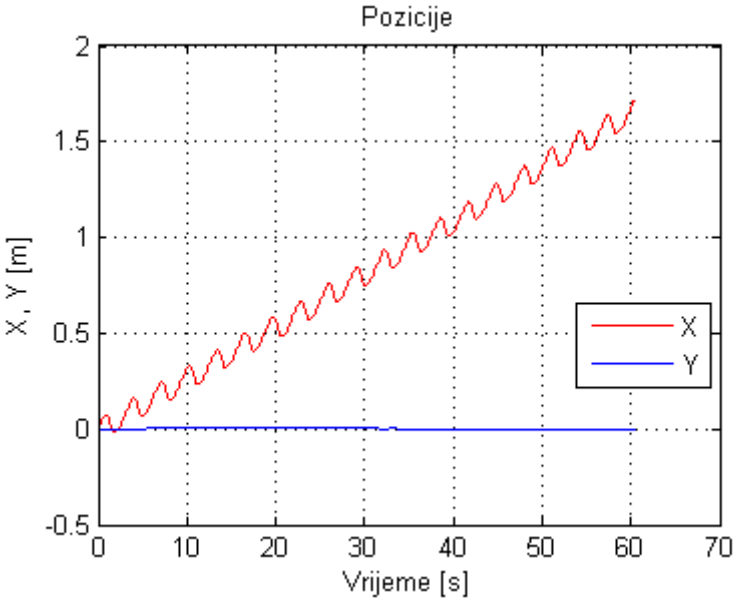


Slika 3.13 Pozicije modula za θ_s = 0°

Povećanjem faznog pomaka u pozitivnom smislu ostvaruje se gibanje robota u pozitivnom smjeru osi X. Povećanjem faznog pomaka povećava se i brzina modula, a time i prijeđena udaljenost za 60 sekundi simulacije. Maksimalan pomak u pozitivnom smjeru X osi ostvaren je za fazni pomak $\theta_s = 60°$ i on iznosi $X(60s) = 1.67m$ što odgovara brzini od $v = 2.78\ \frac{cm}{s}$. Pozicije aktuatora za $\theta_s = 60°$ dane su na *Slici 3.14*, a pozicije modula na *Slici 3.15*. Povećanje faznog pomaka nakon $\theta_s = 60°$ dovodi do smanjenja brzine modula koja konačno pada na nulu za $\theta_s \sim 120°$.



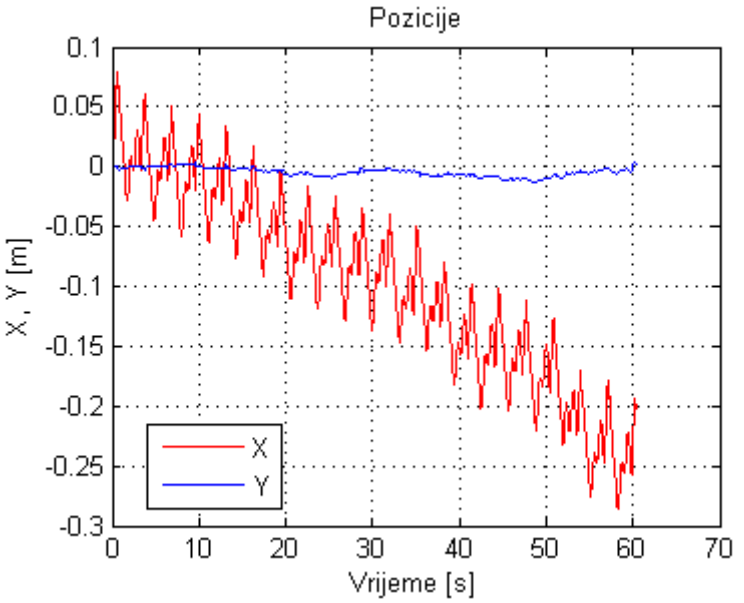
Slika 3.14 Pozicije aktuatora za θ_s=60°



Slika 3.15 Pozicije modula za θ_s = 60°

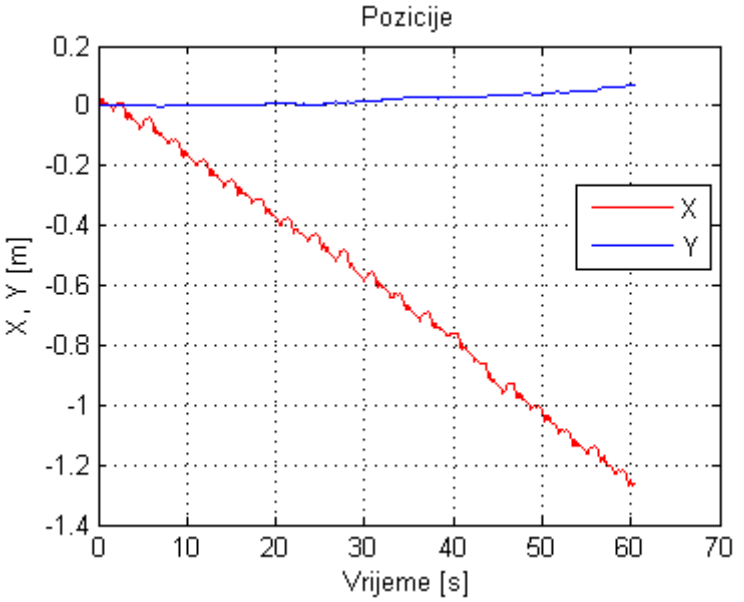
Za fazni pomak od $\theta_s = 90°$ ostvarena je brzina $1.86\ \frac{cm}{s}$. Ta brzina odgovara već mjerenoj brzini za iste parametre u prethodnom poglavlju što ukazuje na ponovljivost mjerenja i valjanost simulacijskog modela. Za fazne pomake od $\theta_s = 0°$ do $\theta_s = 105°$ nema značajnijeg odstupanja robota od gibanja po X osi, odnosno pomaci robota po Y osi su jedan centimetar ili manje (uglavnom nula).

Za fazni pomak od $\theta_s = 120°$ modul se giba u negativnom smjeru osi X. Time je ušao u drugu fazu gibanja u kojoj će za faze od $\theta_s = 120°$ do $\theta_s = 180°$ uvijek ići u negativnom smjeru. Pozicija modula za $\theta_s = 120°$ prikazana je na *Slici 3.16*.

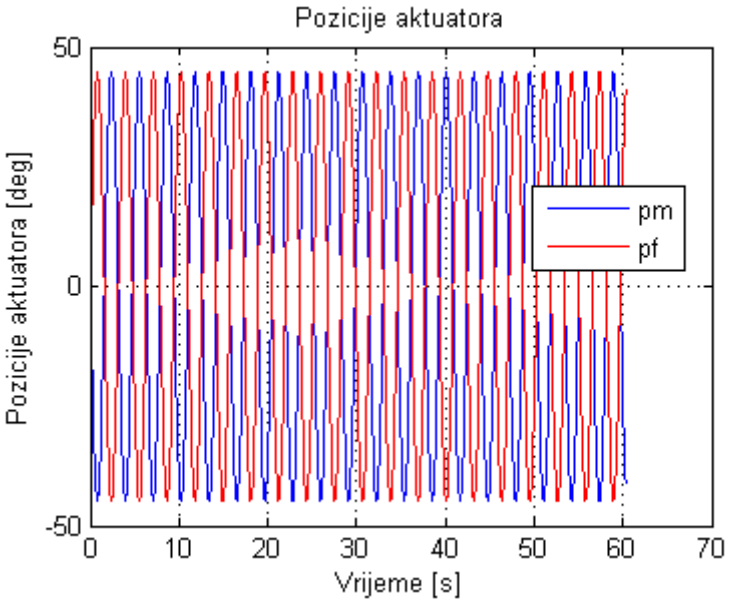


Slika 3.16 Pozicije modula za θ_s = 120°

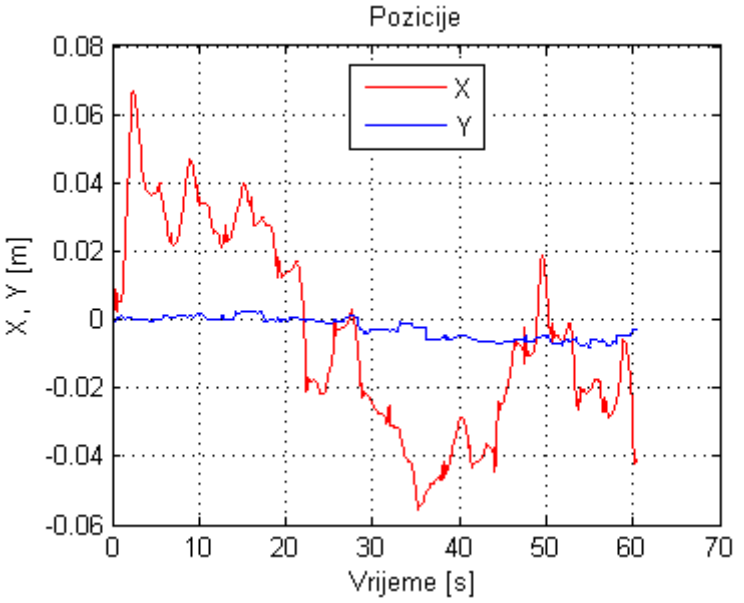
Slično kao i kod pozitivne poluperiode gibanja, u negativnoj je najmanja brzina ostvarena za $\theta_s = 120°$ te se povećava povećanjem faznog pomaka. Maksimalnu vrijednost postiže za $\theta_s = 165°$. Pozicije modula za $\theta_s = 165°$ prikazane su na *Slici 3.17.* Daljnjim povećanjem faznog pomaka ponovno se smanjuje brzina gibanja te za $\theta_s = 180°$ ponovno postiže minimalnu vrijednost. Na toj vrijednosti aktuatori su u protufazi pa nije moguće ostvariti kontrolirano gibanje u bilo kojem smjeru. Pozicija robota zbog toga oscilira oko nulte vrijednosti. Položaji aktuatora za $\theta_s = 180°$ prikazani su na *Slici 3.18*, a položaji modula na *Slici 3.19*. U rasponu faznog pomaka od $\theta_s = 120°$ do $\theta_s = 180°$ javlja se nešto veći pomak modula u smjeru Y osi koji doseže čak sedam centimetara i to upravo za fazni pomak kojim je ostvarena najveća udaljenost u negativnom smjeru X osi, vidljivo na *Slici 3.17.*



Slika 3.17 Pozicije modula za θ_s = 165°



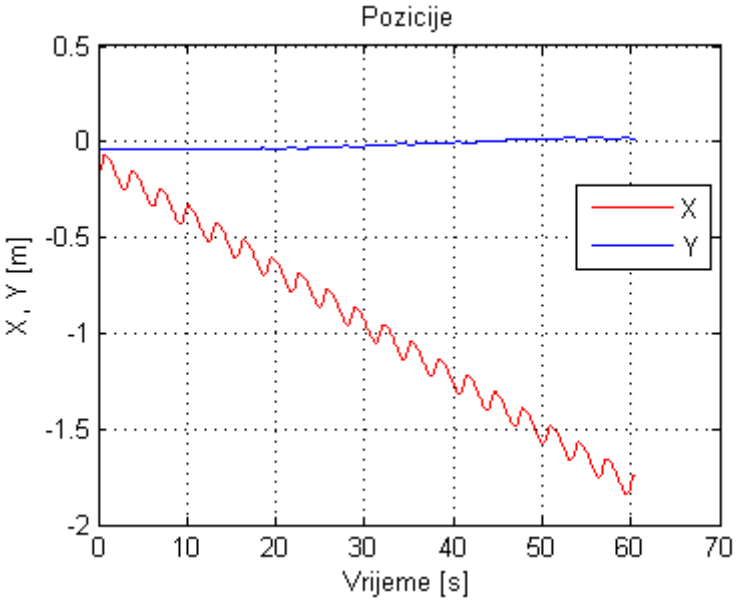
Slika 3.18 Pozicije aktuatora za θ_s=180°



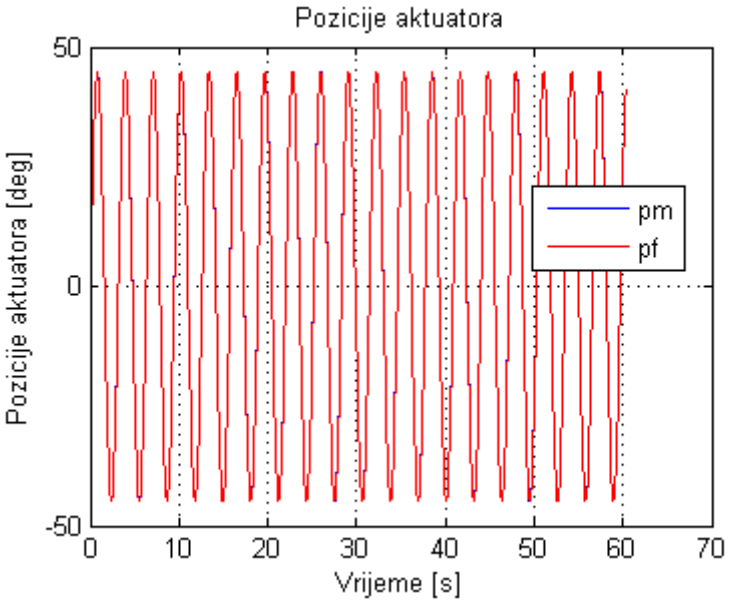
Slika 3.19 Pozicije modula za θ_s = 180°

Povećanjem faznog pomaka preko θ_s = 180° modul ponovno započinje s gibanjem u pozitivnom smjeru X osi. Modul će zadržati to gibanje sve do θ_s = 240°. U ovom intervalu ponašanje je slično onome za interval faznih pomaka od θ_s = 0° do θ_s = 120°; povećanjem faznog pomaka povećava se prijeđena udaljenost, a maksimum je ostvaren za θ_s = 195°. Iznos maksimuma je manji nego za prvi pozitivni interval, a mogu se uočiti i značajna odstupanja u smjeru Y osi.

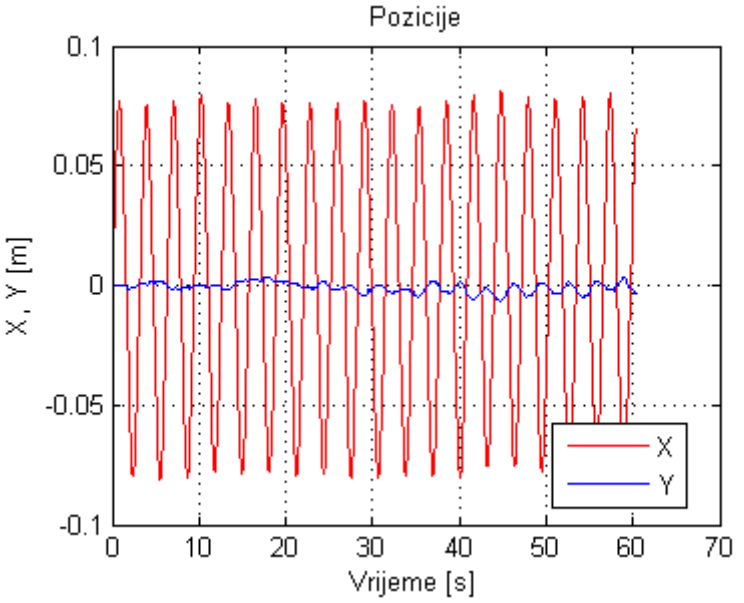
Sa θ_s = 255° modul započinje gibanje u negativnom smjeru koje će zadržati sve do θ_s = 360°. Zanimljivo je da je na ovom intervalu ostvaren i globalni ekstrem funkcije i to za θ_s = 300°. Za ovaj fazni pomak ostvarena je brzina u negativnom smjeru osi X od 2.91 $\frac{cm}{s}$, što odgovara pomaku od -1.75 metra, kao što je vidljivo na *Slici 3.20*.



Slika 3.20 Pozicije modula za θ_s = 300°



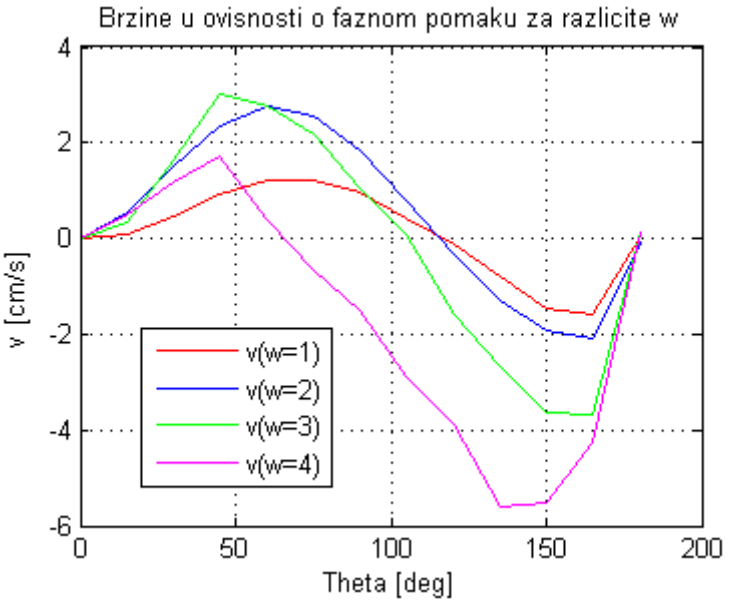
Slika 3.21 Pozicije aktuatora za θ_s=360°



Slika 3.22 Pozicije modula za θ_s = 360°

Za fazni pomak $\theta_s = 360°$ ponovno nema faznog pomaka između aktuatora modula te nije moguće ostvariti gibanje prikazano na *Slici 3.21*. Zbog toga će modul oscilirati oko nulte pozicije, kao što je prikazano na *Slici 3.22*. Ovime je opisano ponašanje modula za sve moguće fazne pomake među aktuatorima. Interesantno je uočiti da se svi značajni događaji zbivaju na višekratnicima kuta $\theta_s = 60°$; tako je maksimalan pomak u pozitivnom smjeru X osi ostvaren na $\theta_s = 60°$, a minimalan na $\theta_s = 120°$, $\theta_s = 180°$, $\theta_s = 240°$ i $\theta_s = 360°$. Maksimalan pomak u negativnom smjeru X osi ostvaren je za $\theta_s = 300°$.

Kako bi se dobio bolji uvid u ponašanje brzine modula u ovisnosti o faznom pomaku napravljenje su simulacije promjene brzine modula u ovisnosti o faznom pomaku za kružne frekvencije $w = 1$, $w = 2$, $w = 3$ i $w = 4$, a rezultati su prikazani na *Slici 3.23*.



Slika 3.23 Brzine modula u ovisnosti o faznom pomaku za w=1, w=2, w=3 i w=4

Iz grafa je vidljivo da se za sve četiri kružne frekvencije ostvaruje približno sinusoidalno ponašanje brzine u ovisnosti o faznom pomaku. Za $w = 1$ i $w = 2$ vladanje brzine je gotovo istovjetno; pozitivni ekstrem ostvaren je za θ_s = 60°, nultočke su u θ_s = 0°, θ_s ~120° i θ_s = 180°, dok je negativni ekstrem u θ_s = 165°. Očekivano, veća brzina je ostvarena za $w = 2$. Najveća brzina ostvarena je za $w = 3$ i to pri θ_s = 45°, ali modul započinje s kretanjem u negativnom smjeru pri manjem faznom pomaku (θ_s = 105°) u odnosu na $w = 1$ i $w = 2$. Povećanjem kružne frekvencije na $w = 4$ nije ostvarena veća brzina gibanja modula, ona je manja nego kod $w = 2$, bar što se tiče pozitivnog smjera gibanja. Vidljivo je da za $w = 4$ modul započne s gibanjem u negativnom smjeru X osi već u okolici θ_s = 60°. Iz svega navedenog može se zaključiti da je najbolja kružna frekvencija za aktuaciju modula iz perspektive faznog pomaka $w = 2$ jer pruža najveću brzinu kretanja za najveći raspon faznih pomaka θ_s.

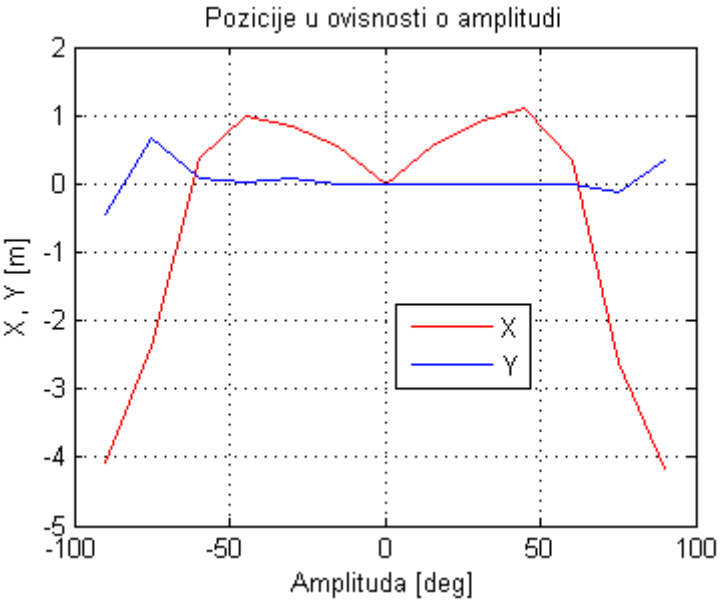
3.2.2. Utjecaj amplitude aktuatora na gibanje modula

Nakon opisa utjecaja faznog pomaka na gibanje modula u prethodnom poglavlju, opisat će se utjecaj zadane amplitude gibanja aktuatora na gibanje modula. Aktuatori su u gibanju ograničeni vrijednostima $A = 90°$ i $A = -90°$ pa su simulacije izrađene samo za zadani interval u kojem je jedino i moguće gibanje aktuatora. Ostali parametri aktuacije su $w = 2\frac{rad}{s}$ i $\theta_s = \frac{\pi}{2}\ rad$. Rezultati simulacija prikazani su u *Tablici 3.2*.

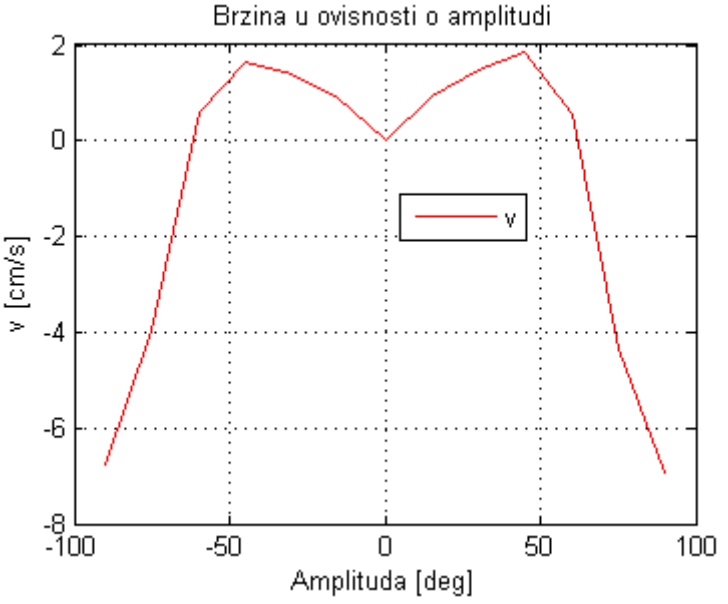
Tablica 3.2 Ovisnost pozicija modula o amplitudi aktuatora

A	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v}\ \left[\frac{\boldsymbol{cm}}{\boldsymbol{s}}\right]$
-90	-4.07	-0.46	-6.78
-75	-2.38	0.67	-3.97
-60	0.36	0.07	0.6
-45	0.99	0.03	1.65
-30	0.84	0.07	1.4
-15	0.54	0	0.9
0	0	0	0
15	0.56	-0.02	0.93
30	0.89	-0.01	1.48
45	1.12	0	1.87
60	0.34	0	0.57
75	-2.60	-0.13	-4.33
90	-4.16	0.35	-6.93

Grafički prikaz rezultata iz *Tablice 3.2* dan je na *Slici 3.24* i *Slici 3.25*.



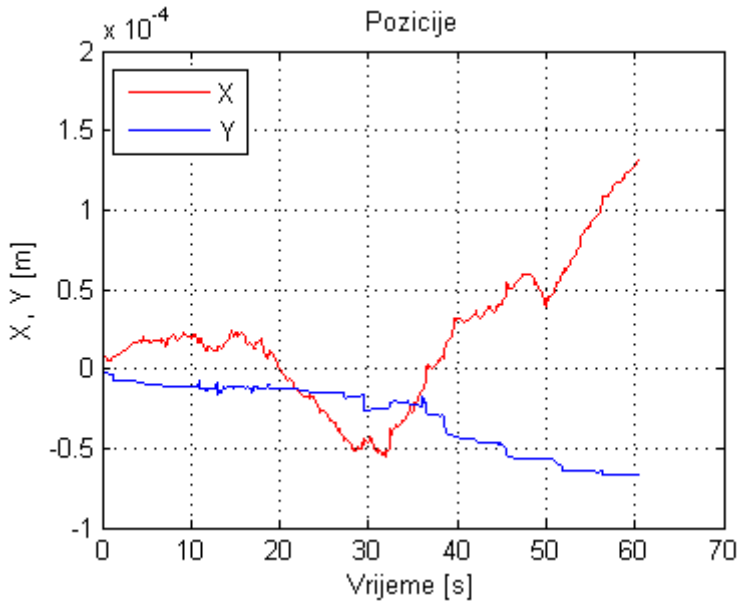
Slika 3.24 Pozicije modula u ovisnosti o amplitudi



Slika 3.25 Brzina modula u ovisnosti o amplitudi

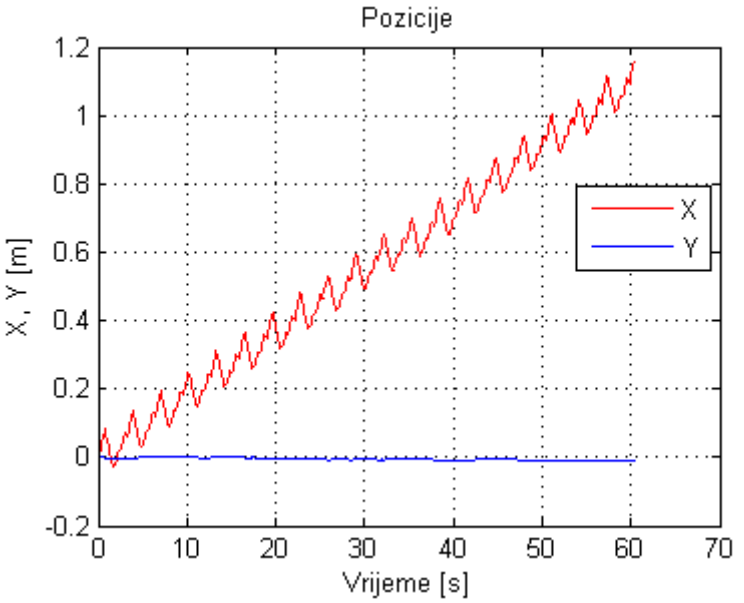
Na slikama se prvo uočava simetričnost grafa s obzirom na nulu. Simetričnost pokazuje da, bez obzira je li aktuator pobuđen negativnom ili pozitivnom amplitudom, smjer gibanja i doseg gibanja robota je gotovo istovjetan; zaključuje se da smjer gibanja i prijeđeni put modula ne ovise o predznaku amplitude sve dok su oba aktuatora pobuđena amplitudom jednakog predznaka. Ako su aktuatori pobuđeni jednakim iznosom amplitude, ali suprotnih predznaka ,tada će postojati dodatni fazni pomak između aktuatora koji može rezultirati oscilacijom modula u mjestu, drugačijom brzinom gibanja ili gibanjem modula u negativnom smjeru osi, kao što je slučaj prikazan na *Slici 3.28*.

Za $A = 0°$ očekivano, nema gibanja modula, kao što je vidljivo na *Slici 3.26*.

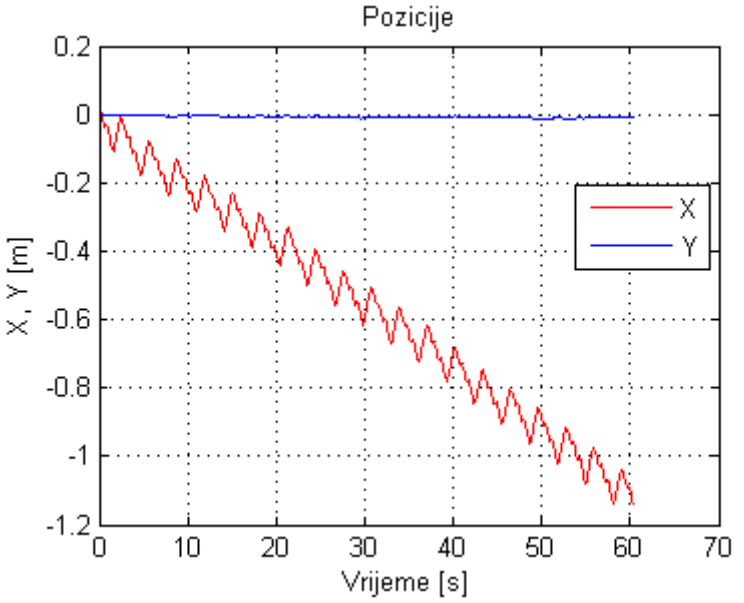


Slika 3.26 Pozicije modula za A=0°

Male promjene u poziciji nastaju kao posljedica okretnog momenta koji motori moraju generirati kako bi zadržali zadanu poziciju ($A = 0°$).



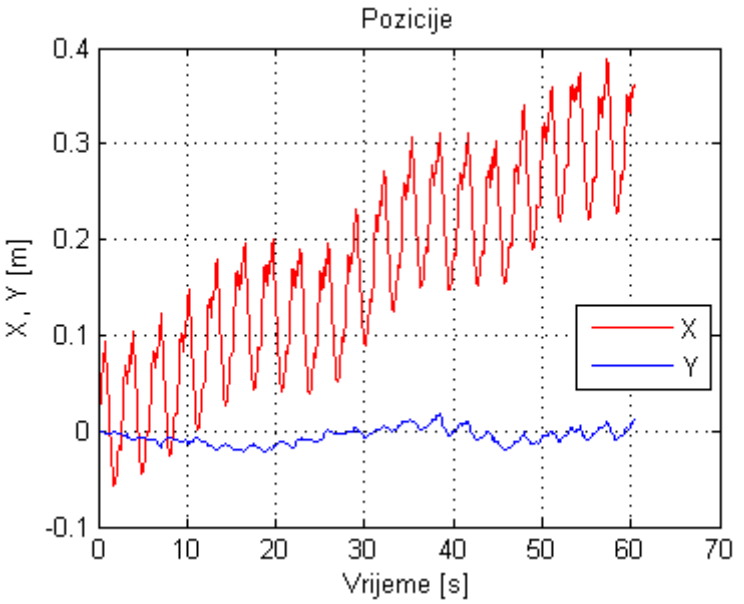
Slika 3.27 Pozicije modula za A=45°



Slika 3.28 Pozicije modula za A_m=-45° i A_f=45°

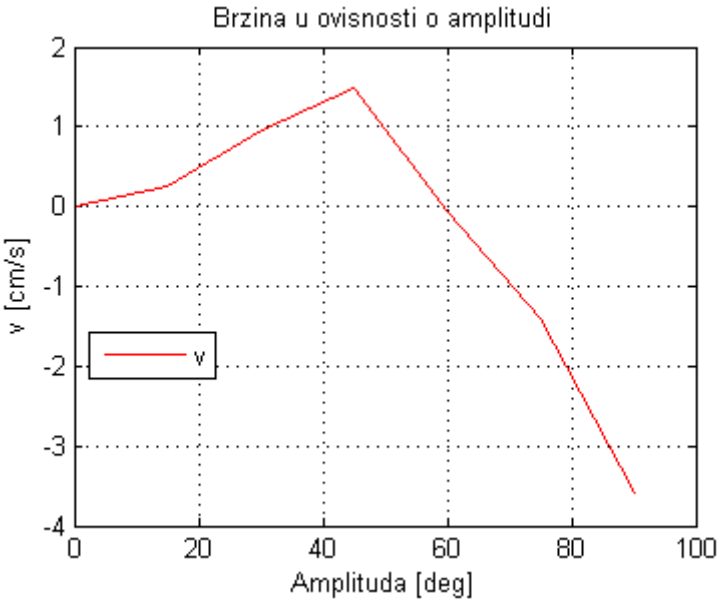
Sa *Slike 3.24* vidljivo je da se najveći pomak u pozitivnom smjeru X osi postiže za $A = 45°$. Na *Slici 3.27* prikazane su pozicije modula upravo za taj slučaj. Ostvareni pomak iznosi 1.12 metar, što odgovara brzini od $v = 1.86\ \frac{cm}{s}$. Na *Slici 3.28* prikazane su pozicije modula za slučaj kada je muški aktuator pobuđen amplitudom $A_m = -45°$, a ženski aktuator amplitudom $A_f = 45°$. Vidljivo je da će ovakva pobuda aktuatora rezultirati jednakim iznosom pomaka, ali u negativnom smjeru osi X, kako je ranije navedeno.

Za $A = 60°$ kod modula se javlja oscilatorno ponašanje te modul ne ostvaruje značajniji pomak, kako u pozitivnom, tako ni u negativnom smjeru X osi, vidljivo na *Slici 3.29*.

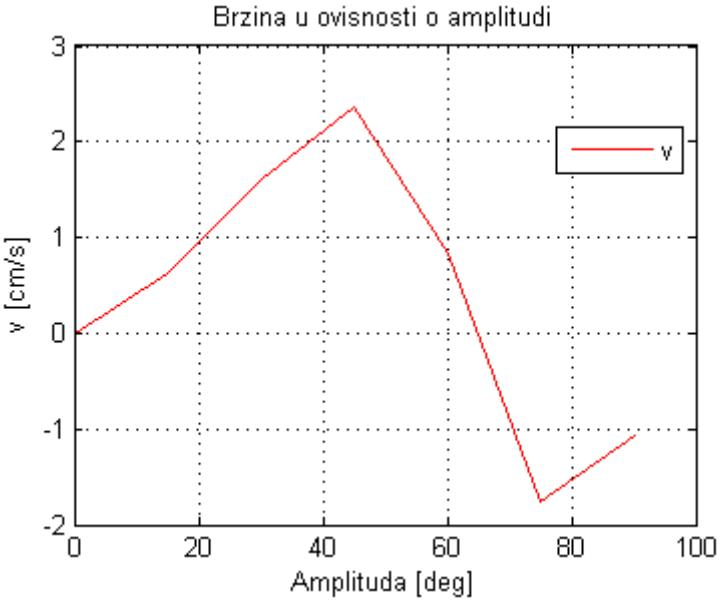


Slika 3.29 Pozicije modula za A=60°

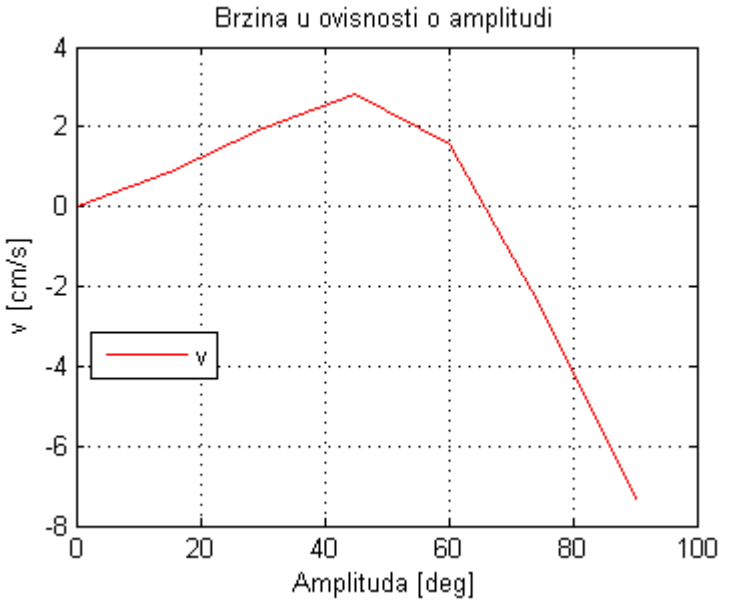
Ovo ponašanje nije vezano uz fazni pomak modula jer je uočljivo na raznim faznim pomacima ($\theta_s = 30°$ *Slika 3.30*, $\theta_s = 45°$ *Slika 3.31*, $\theta_s = 60°$ *Slika 3.32*).



Slika 3.30 Brzina modula za θ_s=30°

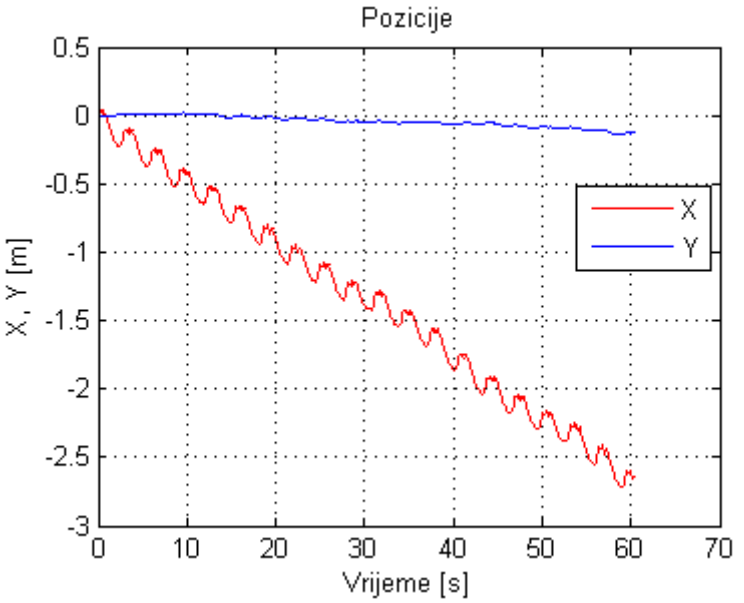


Slika 3.31Brzina modula za θ_s=45°

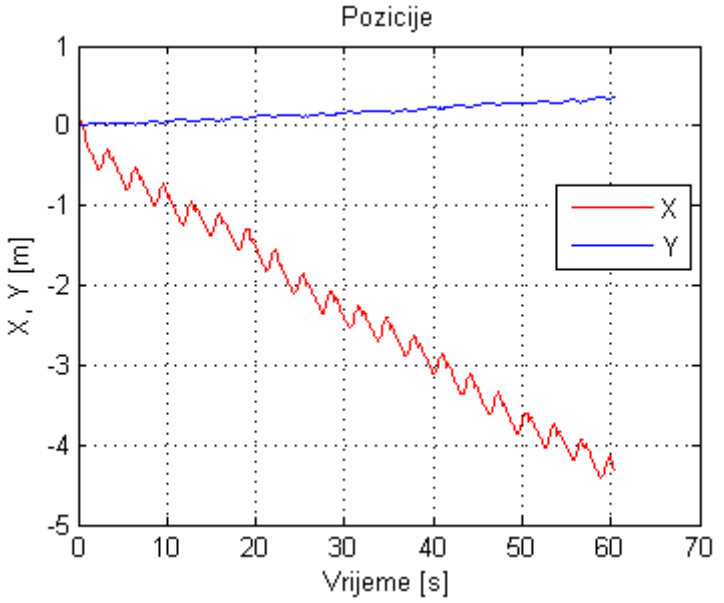


Slika 3.32 Brzina modula za θ_s=60°

Iz grafova zaključujemo da je za sve fazne pomake ponašanje brzine u ovisnosti o amplitudi približno jednako, brzina raste od $A = 0°$ do $A = 45°$ gdje doseže maksimum, potom se smanjuje i u okolici $A = 60°$ mijenja predznak što znači da je modul započeo s gibanjem u negativnom smjeru X osi. Za više amplitude ($A = 75°$ i $A = 90°$) brzina modula se značajno povećava u negativnom smislu, što je vidljivo na *Slici 3.33* za $A = 75°$, i na *Slici 3.34* za $A = 90°$. Posljedica toga je gubitak fluidnosti gibanja te je unesena doza nepredvidljivosti u gibanje. To najbolje pokazuju odstupanja u smjeru Y osi koja su značajno veća od odstupanja za druge simulirane amplitude. Iako je brzina povećana, ne smije se zanemariti da se modul giba u smjeru suprotnom od željenoga i da je to gibanje skokovite naravi. Zaključuje se da je na visokim amplitudama dijelom izgubljeno upravljanje nad ponašanjem modula te je povećana entropija, pa visoke amplitude nisu prikladne za aktuaciju modula.



Slika 3.33Pozicije modula za A=75°



Slika 3.34 Pozicije modula za A=90°

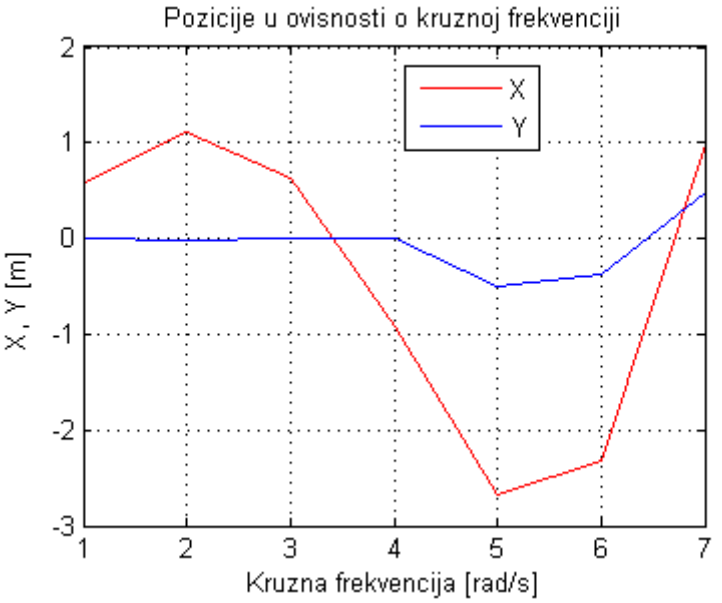
3.2.3. Utjecaj kružne frekvencije aktuatora na gibanje modula

Nakon što je opisan utjecaj faznog pomaka i amplitude na pozicije aktuatora preostaje ispitati utjecaj kružne frekvencije. Simulacije su obavljenje prema postavkama opisanima na početku cjeline i za raspone kružnih frekvencija od $w = 1\frac{rad}{s}$ do $w = 7\frac{rad}{s}$. Ispitana je i promjena amplitude aktuatora na različitim kružnim frekvencijama. Ispitivanjem se želi potvrditi ili opovrgnuti postavljena teza da se maksimalan put modula ostvaruje za $A = \frac{\pi}{4}\ rad$. Rezultati simulacija uz konstantne parametre $A = \frac{\pi}{4}\ rad$ i $\theta_s = \frac{\pi}{2}\ rad$ dani su u *Tablici 3.3,* dok su rezultati ispitivanja promjene amplitude na različitim kružnim frekvencijama dani na kraju poglavlja u *Tablici 3.4*, *Tablici 3.5, Tablici 3.6* i *Tablici 3.7.*

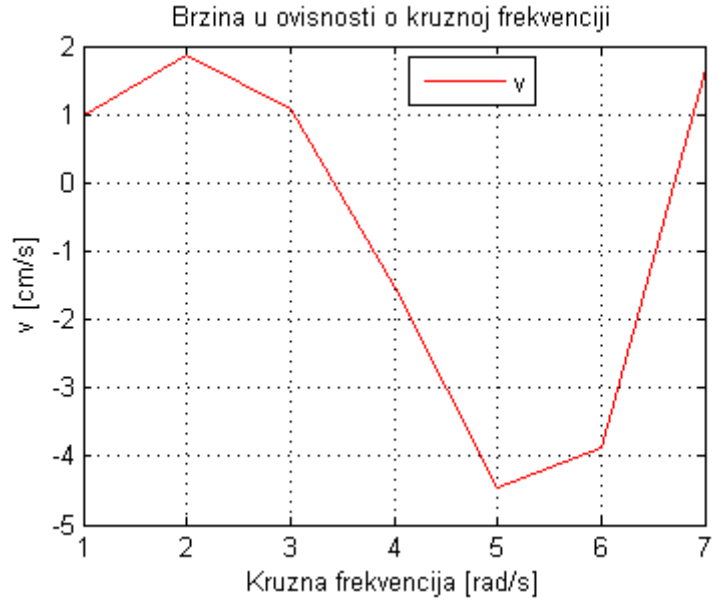
Tablica 3.3 Ovisnost pozicija modula o kružnoj frekvenciji aktuatora

w	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v}\ \left[\frac{\boldsymbol{cm}}{\boldsymbol{s}}\right]$
1	0.58	0	0.97
2	1.12	-0.01	1.87
3	0.64	0	1.07
4	-0.91	0	-1.52
5	-2.67	-0.5	-4.45
6	-2.33	-0.37	-3.88
7	0.99	0.49	1.65

Grafički prikaz rezultata iz *Tablice 3.3* prikazan je na *Slici 3.35* i *Slici 3.36.*

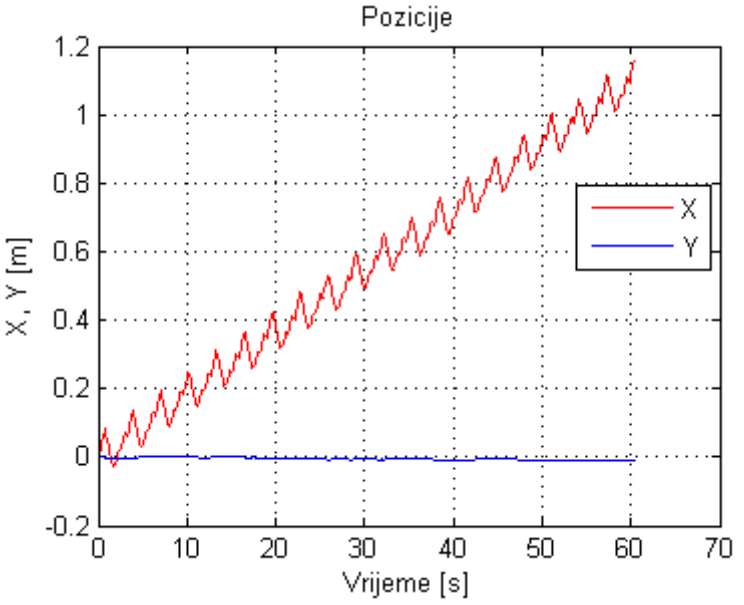


Slika 3.35 Pozicije u ovisnosti o kružnoj frekvenciji



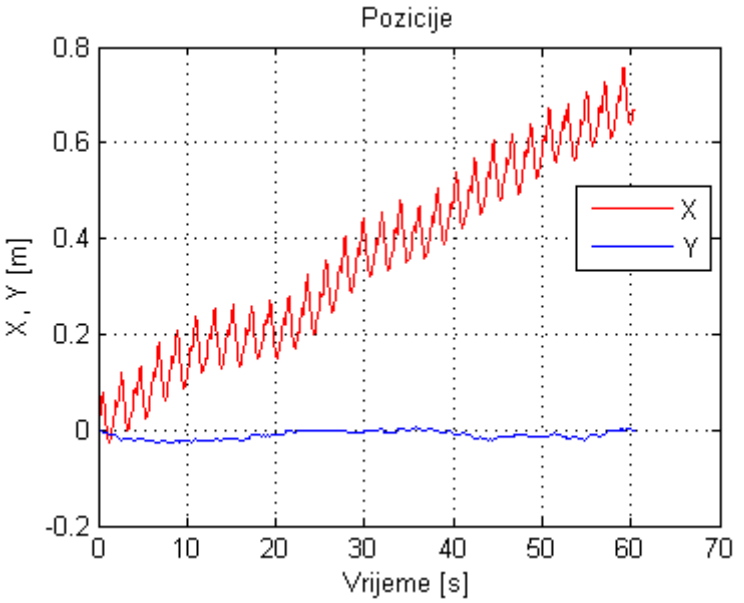
Slika 3.36 Brzina u ovisnosti o kružnoj frekvenciji

Iz prethodnih dijagrama može se iščitati da se najveća brzina postiže za $w = 2$. Dijagram pozicija za navedenu kružnu frekvenciju prikazan je na *Slici 3.37.*

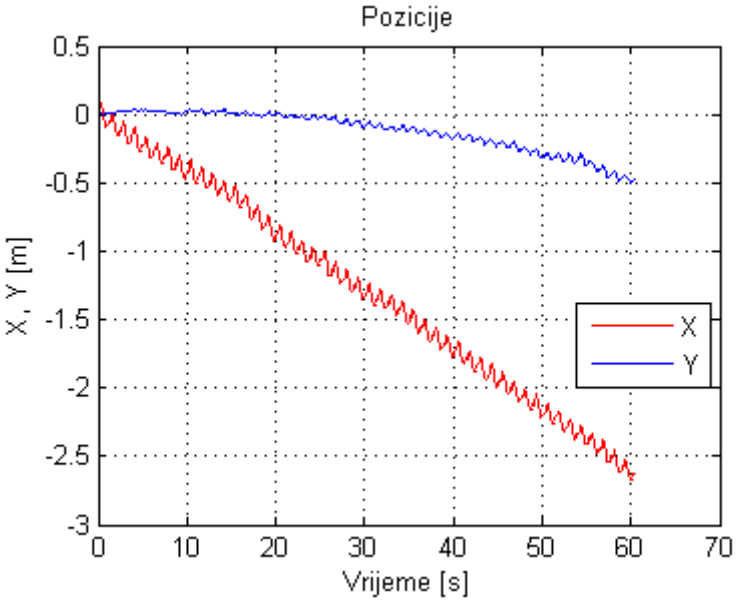


Slika 3.37 Pozicije za kružnu frekvenciju w=2

Iz tablice se može vidjeti da je brzina gibanja modula za $w = 2$ gotovo dvostruko veća nego za $w = 1$. Daljnjim povećanjem kružne frekvencije, na $w = 3$, ne ostvaruje se povećanje brzine modula, vidljiv je utjecaj skliza pri gibanju modula pa je prijeđena udaljenosta manja, kao što je vidljivo na *Slici 3.38*. Slična je situacija i za $w = 4$, gdje je važno primijetiti da je modul započeo gibanje u negativnom smjeru X osi. Maksimalan put u negativnom smjeru X osi ostvaren je za $w = 5$ i iznosi -2.67 metara, što odgovara brzini od čak $v = -4.45\ \frac{cm}{s}$. Unatoč činjenici da je to značajno veća brzina od svih drugih izmjerenih, u obzir se mora uzeti i činjenica da je modul napravio i neželjeni pomak u smjeru osi Y od gotovo pola metra, vidljivo na *Slici 3.39.*



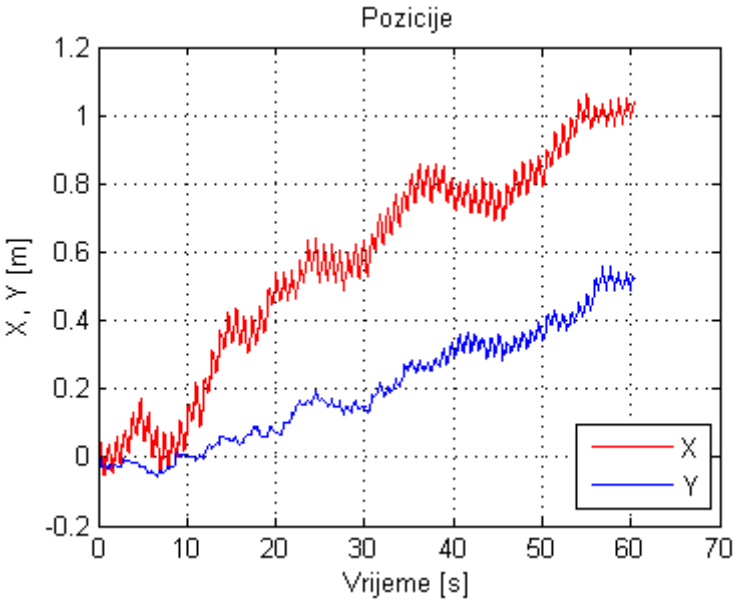
Slika 3.38 Pozicije za kružnu frekvenciju w=3



Slika 3.39 Pozicije za kružnu frekvenciju w=5

Činjenica da se za $w = 5$ ostvaruje značajan neželjeni pomak u smjeru osi Y, kao i činjenica da se modul giba u negativnom smjeru osi X, čini ovu frekvenciju neprikladnom za aktuaciju modula. Sličan rezultat postignut je i za $w = 6$, uz nešto manji prijeđeni put. Na *Slici 3.40* prikazan je dijagram pozicija za $w = 7$. Iz dijagrama se može vidjeti da se pri ovoj kružnoj frekvenciji modul ne giba u željenom smjeru te je njime gotovo nemoguće upravljati . Pomak u pozitivnom smjeru X osi iznosi samo jedan metar, dok je neželjeni pomak po Y osi gotovo pola metra.

Iz svega navedenog zaključujemo da sve više kružne frekvencije ($w = 4$, $w = 5$, $w = 6$ i $w = 7$) nisu prikladne za aktuaciju modula jer daju neželjen smjer gibanja, a od nižih kružnih frekvencija je za aktuaciju najprikladnija $w = 2$ jer omogućuje najveću brzinu gibanja modula. Ako se želi promijeniti smjer gibanja modula, tako da se umjesto u pozitivnom smjeru X osi giba u neagtivnom, dovoljno je promijeniti predznak kružne frekvencije i za iste iznose ostvarit će se jednaki rezultati gibanja.



Slika 3.40 Pozicije za kružnu frekvenciju w=7

U prethodnom *Potpoglavlju 3.2.2*, opisan je utjecaj promjene amplitude na gibanje modula i zaključeno je da se maksimalan put ostvaruje za $A = \frac{\pi}{4}\, rad$, te da u okolici $A = \frac{\pi}{3}\, rad$ modul započinje s gibanjem u negativnom smjeru pa više amplitude nisu prikladne aktuaciju modula. U nastavku će se dana pretpostavka ispitati za različite kružne frekvencije i to $w = 1$, $w = 2$, $w = 3$ i $w = 4$. Rezultati simulacija prikazani su u *Tablici 3.4*, *Tablici 3.5*, *Tablici 3.6* i *Tablici 3.7*. Usporedba brzina modula za navedene kružne frekvencije prikazana je na *Slici 3.45*.

Tablica 3.4 Ovisnost pozicija modula o amplitudi aktuatora za w=1

A	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v}\ \left[\frac{\boldsymbol{cm}}{\boldsymbol{s}}\right]$
0	0	0	0
15	0.19	0	0.32
30	0.45	0	0.75
45	0.58	0	0.97
60	0.59	0	0.98
75	-0.20	0	-0.33
90	-1.29	-0.02	-2.15

Tablica 3.5 Ovisnost pozicija modula o amplitudi aktuatora za w=2

A	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v}\left[\frac{\boldsymbol{cm}}{\boldsymbol{s}}\right]$
0	0	0	0
15	0.56	-0.02	0.93
30	0.89	-0.01	1.48
45	1.12	0	1.87
60	0.34	0	0.57
75	-2.60	-0.13	-4.33
90	-4.16	0.35	-6.93

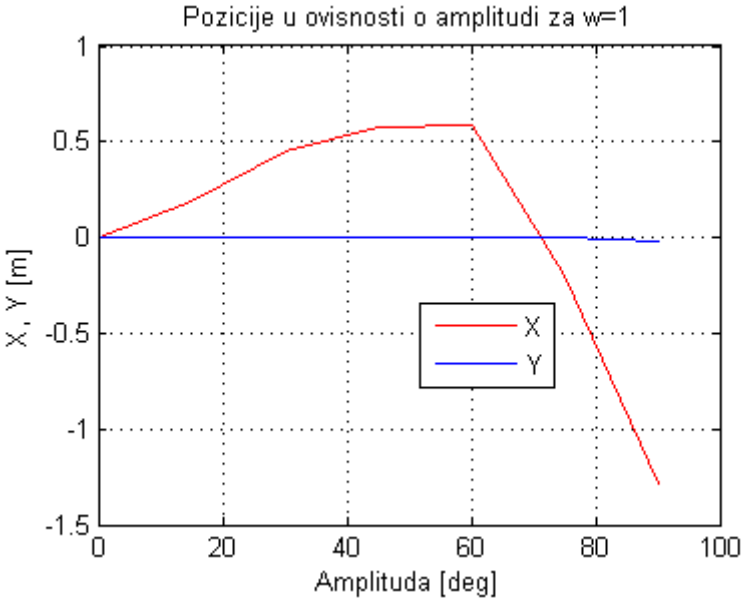
Tablica 3.6 Ovisnost pozicija modula o amplitudi aktuatora za w=3

A	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v}\left[\frac{\boldsymbol{cm}}{\boldsymbol{s}}\right]$
0	0	0	0
15	0.75	0	1.25
30	1.19	0.04	1.98
45	0.64	0	1.07
60	-2.02	-0.29	-3.37
75	-5.05	-1.25	-8.42
90	-5.02	0.86	-8.37

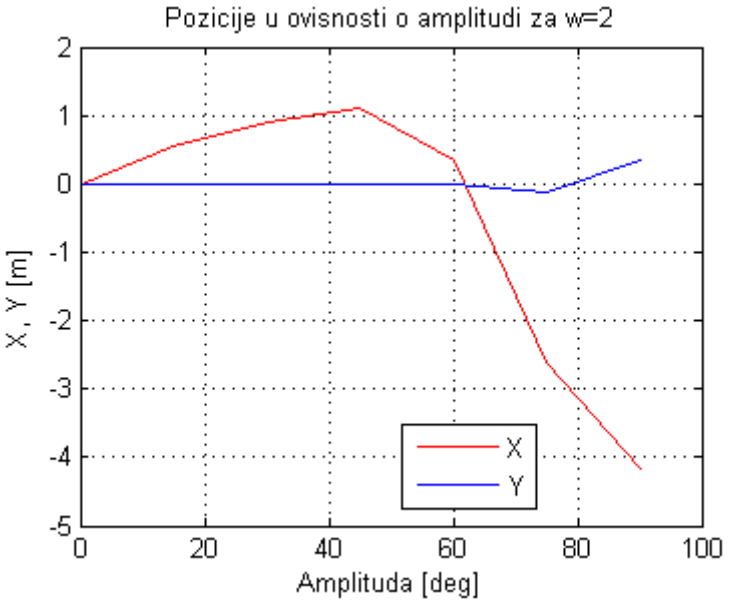
Tablica 3.7 Ovisnost pozicija modula o amplitudi aktuatora za w=4

A	**X(60s) [m]**	**Y(60s) [m]**	$\boldsymbol{v}\left[\frac{\boldsymbol{cm}}{\boldsymbol{s}}\right]$
0	0	0	0
15	1	0.01	1.67
30	1.08	0.02	1.81
45	-0.91	0	-1.52
60	-3.52	-0.59	-5.87
75	-5.08	-0.80	-8.47
90	0.77	-0.83	1.28

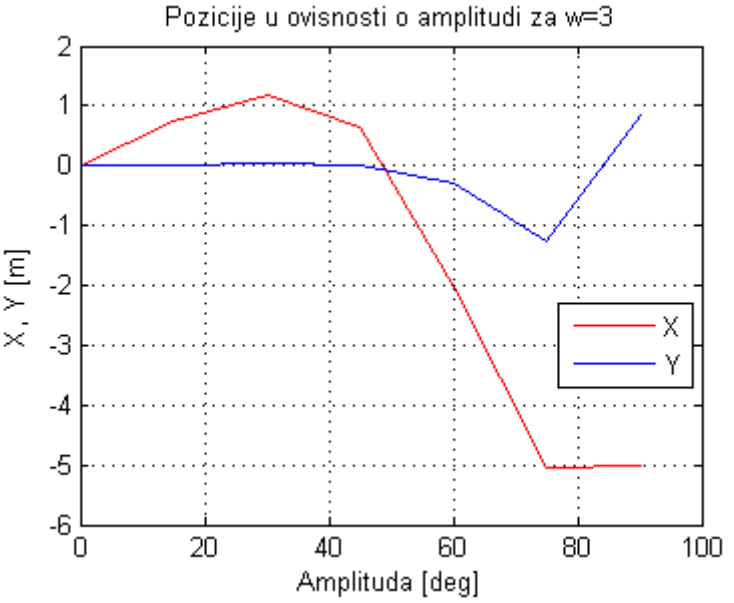
Grafički prikaz navedenih podataka dan je na *Slici 3.41*, *Slici 3.42*, *Slici 3.43* i *Slici 3.44.*



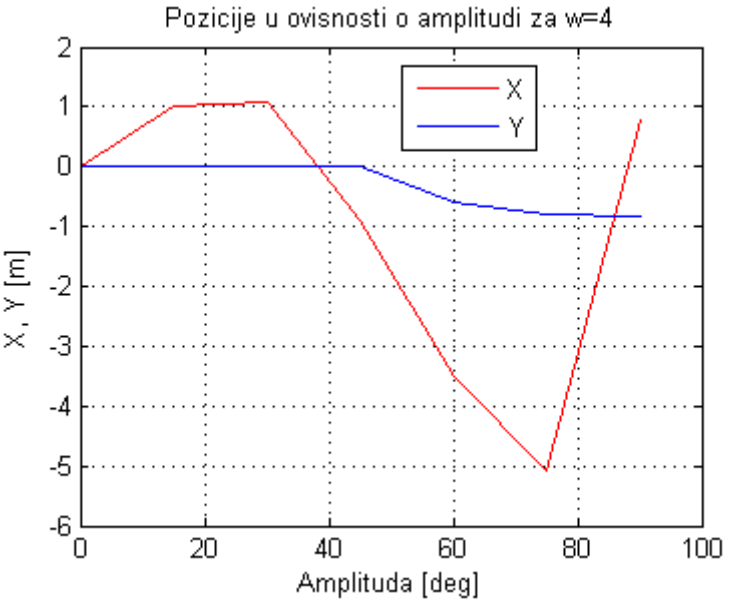
Slika 3.41Pozicije modula u ovisnosti o amplitudi za w=1



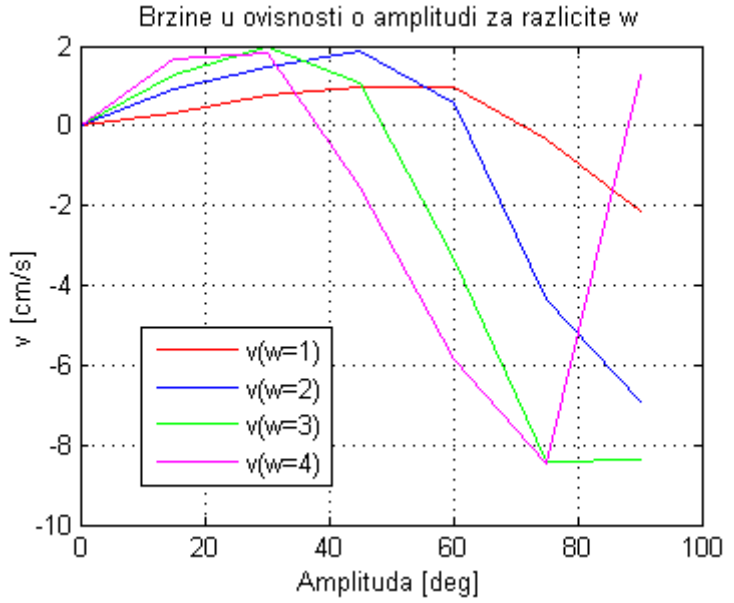
Slika 3.42 Pozicije modula u ovisnosti o amplitudi za w=2



Slika 3.43 Pozicije modula u ovisnosti o amplitudi za w=3



Slika 3.44 Pozicije modula u ovisnosti o amplitudi za w=4



Slika 3.45 Usporedba brzina u ovisnosti o amplitudi za w=1, w=2, w=3 i w=4

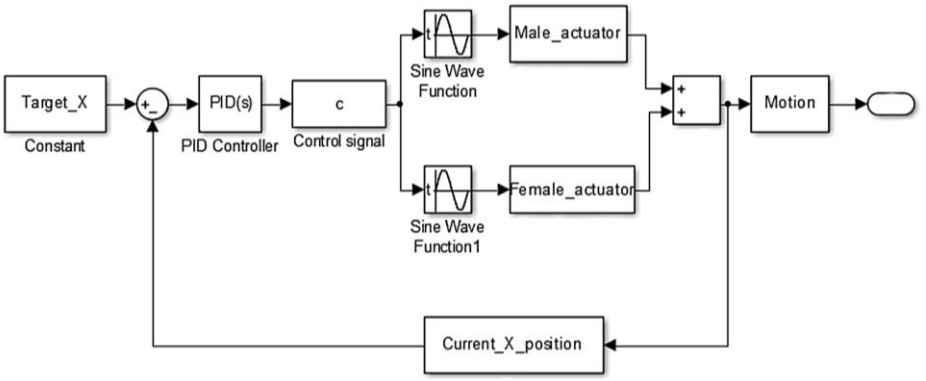
Iz rezultata simulacija može se zaključiti da iako se brzina u ovisnosti o amplitudi ponaša jednako za sve fazne pomake između aktuatora, to se ne događa kada se u obzir uzme ovisnost brzine o amplitudi aktuatora za različite kružne frekvencije. Sa *Slike 3.45* može se vidjeti da se maksimalna brzina ostvaruje za kružne frekvencije od $w = 2$ i to pri amplitudi $A = 45°$ i $w = 3$, pri amplitudi $A = 30°$. Najniža brzina ostvarena je za $w = 1$, ali je za tu kružnu frekvenciju ostvareno gibanje u pozitivnom smjeru X osi za amplitude do $A = 75°$ što nije slučaj kod drugih testiranih kružnih frekvencija. Što se ostalih kružnih frekvencija tiče, promjena smjera gibanja modula ostvarena je za $w = 2$ u okolici $A = 60°$, za $w = 3$ u okolici $A = 45°$, a za $w = 4$ u okolici $A = 40°$. S povećanjem kružne frekvencije smanjuje se raspon amplituda kojima se može aktuirati modul, a za koje se ostvaruje željeno ponašanje modula, tj. gibanje u željenom smjeru. Tako za $w = 1$ raspon amplituda je približno $\Delta A = 70°$, dok za $w = 4$ iznosi manje od $\Delta A = 40°$. Simulacije još jednom dokazuju da modulom nije moguće upravljati pri višim amplitudama narinutim aktuatorima, te da se ta granica maksimalne amplitude za koju je ostvareno željeno gibanje smanjuje povećavanjem kružne frekvencije.

4. UPRAVLJANJE MODULARNIM ROBOTOM LINEARNIM REGULATORIMA

Opisom principa gibanja i njegove implementacije obrađena je najniža razina upravljanja robotom koja je zadužena za upravljanje na razini aktuatora. Na sljedećoj razini upravljanja potrebno je upravljati pozicijom robota, za tu ulogu odabran je PID regulator.

PID regulator je vjerojatno najpoznatiji algoritam iz teorije automatskog upravljanja. Osnovni preduvjet za njegovu implementaciju je povratna veza koja osigurava informaciju o trenutnoj vrijednosti veličine sustava kojom se želi upravljati. Kontinuiranom usporedbom vrijednosti povratne veze sa zadanom referentnom vrijednosti izračunava se vrijednost pogreške koja odgovara razlici tih vrijednosti. Ovisno o iznosu pogreške te P, I i D člana regulatora formira se upravljački signal čiji je zadatak pobuditi sustav s ciljem svođenja signala pogreške na nulu.

Za simulacijski model PID je implementiran programski unutar V-REP-a u Lua programskom okruženju. Načelna shema regulatora dana je na *Slici 4.1*.



Slika 4.1 Načelna izvedba PID regulatora

S prethodne slike vidljivo je da je referenca željena pozicija po X osi *Target_X*. Kako bi se formirao signal pogreške *e_x*, od reference je oduzeta trenutna pozicija modula po X osi *Current_X_position*. Signal pogreške je ulazna veličina *PID Controllera*, koji je zadužen za formiranje upravljačkog signala *c*. Upravljački signal apliciran je na sinusni signal koji se šalje aktuatorima. Aktuatori omogućavaju gibanje modula koje rezultira promjenom njegove trenutne pozicije po X osi koja se putem negativne povratne veze uspoređuje s referencom čime je zatvorena upravljačka petlja. Programska implementacija navedenog regulatora ostvarena je prema pseudokodu prikazanom na *Slici 4.2*.

```
--zeljena pozicija po X
    target_x=1
--Parametri regulatora:
            Kp=1
            Ki=0
            int=0
            dt=0.05
--Dohvat trenutne X pozicije:
    current_x_position=sim.getObjectPosition(dtto)
--Regulator:
    e=target_x-current_x_position
    int=int+e*dt
    c=Kp*e+Ki*int
```

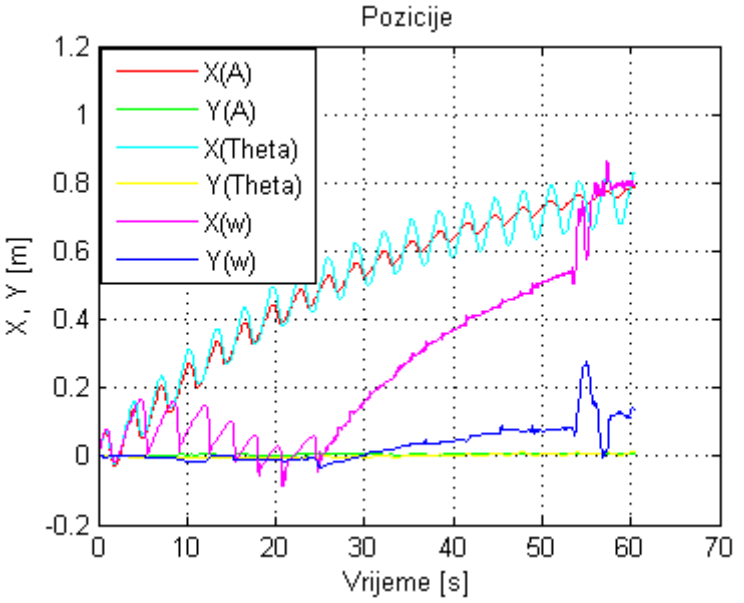
Slika 4.2 Pseudokod za implementaciju PI regulatora

Iz koda je vidljivo da je realiziran PI regulator. Derivativni član je izostavljen jer bi nagle promjene upravljačkog signala izazvale neželjeno ponašanje sustava koji je preinertan da bi pravodobno na njih reagirao. Sljedeći korak u implementaciji PI regulatora je odabrati veličinu kojom se želi upravljati pozicijom robota. Ujedno, to je i najkompleksniji dio implementacije regulatora. U prethodnom *Poglavlju 3* opisani su principi i pravila na kojima počiva princip gibanja robota. Iz njih se može zaključiti da na brzinu robota utječu sve tri varijable sinusnog signala – amplituda, kružna frekvencija i fazni pomak među aktuatorima. Testirani su regulatori temeljeni na upravljanju amplitudom, faznim pomakom i

kružnom frekvencijom, s parametrima koji omogućuju najbrže i najtočnije gibanje modula:

- $A = \frac{\pi}{4}\ rad$
- $w = 2\frac{rad}{s}$
- $\theta_s = \frac{\pi}{3}\ rad.$

Rezultati simulacije za regulator temljen na upravljanju amplitudom, regulator temljen na upravljanju faznim pomakom i regulator temljen na upravljanju kružnom frekvencijom prikazani su na *Slici 4.3*, uz parametre regulatora Kp=1 i Ki=0.

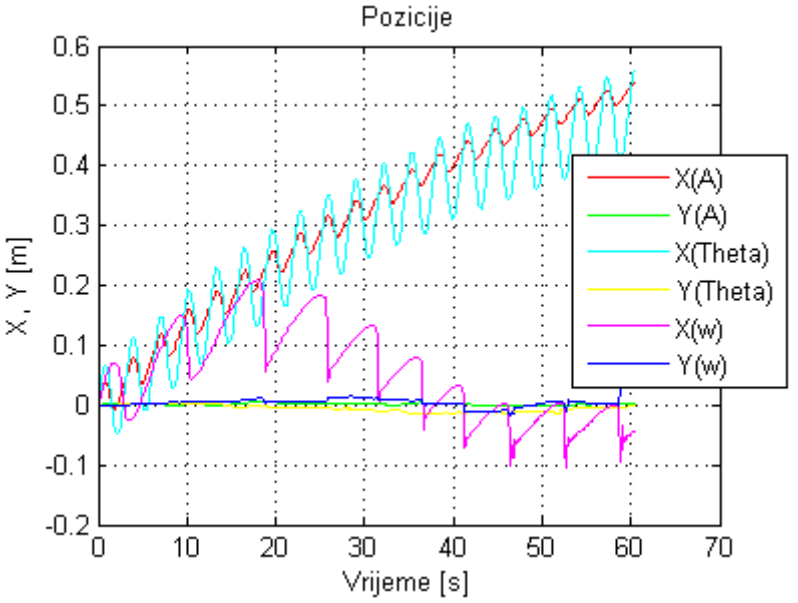


Slika 4.3 Pozicije modula za testirane regulatore uz Kp=1 i Ki=0

S prethodne slike možemo uočiti da se za amplitudni i fazni regulator postižu slični rezultati - modul se asimptotski približava poziciji X=1 koja je odabrana kao referenca. Tu poziciju modul nikad neće doseći jer se njegova brzina smanjuje kako joj se približava, jer amplituda, odnosno faza modula teže ka nuli. Iz ponašanja modula s regulatorom kružne frekvencije može se

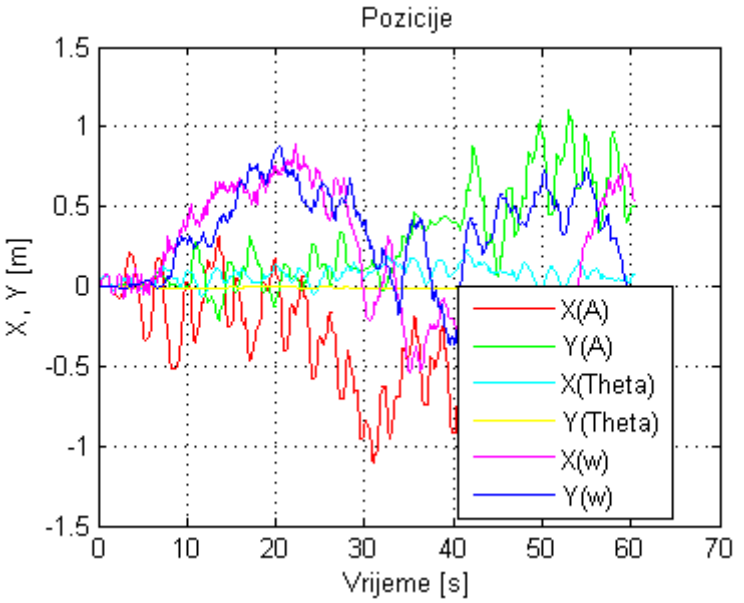
zaključiti da je neupravljiv za ovako realiziran regulator jer je ponašanje modula nepredvidljivo.

Na *Slici 4.4* prikazani su rezultati simulacije uz Kp=0.5. Očekivano, rezultati su bliski onima za Kp=1, ali je dosegnuta pozicija za 60 sekundi simulacije nešto manja jer su faza i amplituda manje što znači i manju brzinu kretanja. Za regulator kružne frekvencije može se vidjeti da ni u ovom slučaju ne daje zadovoljavajuće rezultate.



Slika 4.4 Pozicije modula za testirane regulatore uz Kp=0.5 i Ki=0

Na *Slici 4.5* prikazani su rezultati simulacije za navedene regulatore uz parametre Kp=0.5 i Ki=0.5. Za razliku od prethodnih situacija kada su amplitudni i fazni regulator pokazivali zadovoljavajuće vladanje, to sada nije tako. Uvođenje integralnog člana u ovome iznosu učinilo je modul neupravljivim za sva tri testirana regulatora.



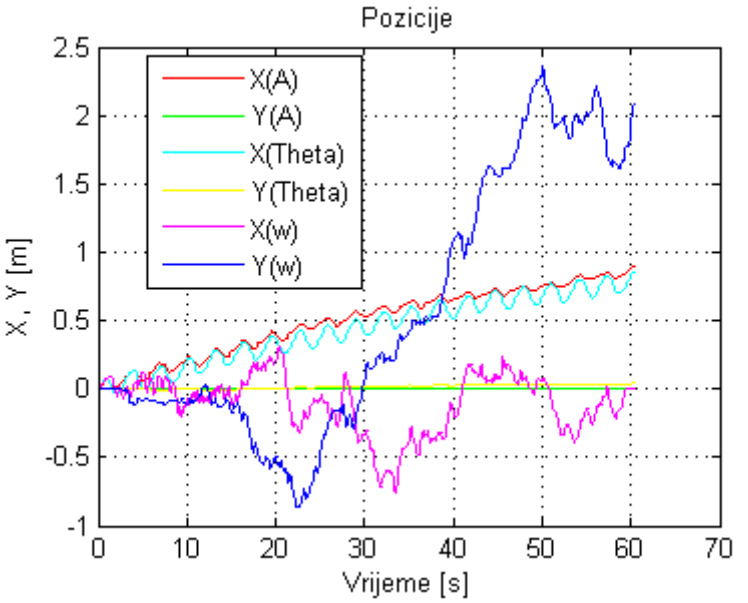
Slika 4.5 Pozicije modula za testirane regulatore uz Kp=0.5 i Ki=0.5

Uzrok ovakvog ponašanja je zasićenje prisutno u sustavu u vidu ograničenog hoda glavnih aktuatora koje dovodi do integralnog nakupljanja (engl. *integral windup*). Zbog velike pogreške integralni član PI regulatora brzo raste te prijeđe određenu vrijednost ispod koje se više ne može vratiti. Posljedica toga je velik doprinos integralnog člana upravljačkom signalu koji će davati nepotrebno visok nalog upravljanoj veličini što rezultira preupravljanjem i vodi sustav u nestabilnost. Rješenje problema leži u postavljanju limita na vrijednost integralnog člana regulatora, tzv. *anti-windup*. Jedan od načina implementacije *anti-windupa* je određivanje granice koju integralni član regulatora može dosegnuti, iznad te granice član se resetira na nulu ili se gasi integralno regulacijsko djelovanje postavljanjem pojačanja na nulu. Implementirano anti-windup rješenje prikazano je na *Slici 4.6*, a rezultati implementacije na *Slici 4.7*.

```
--PI X:
    e_x=(target_x-x_male)

    if math.abs(int_x)>1 then
        int_x=0
    else
        int_x=int_x+e_x*dt
    end
    c_x=Kp*e_x+Ki*int_x
```

Slika 4.6 Anti-windup rješenje

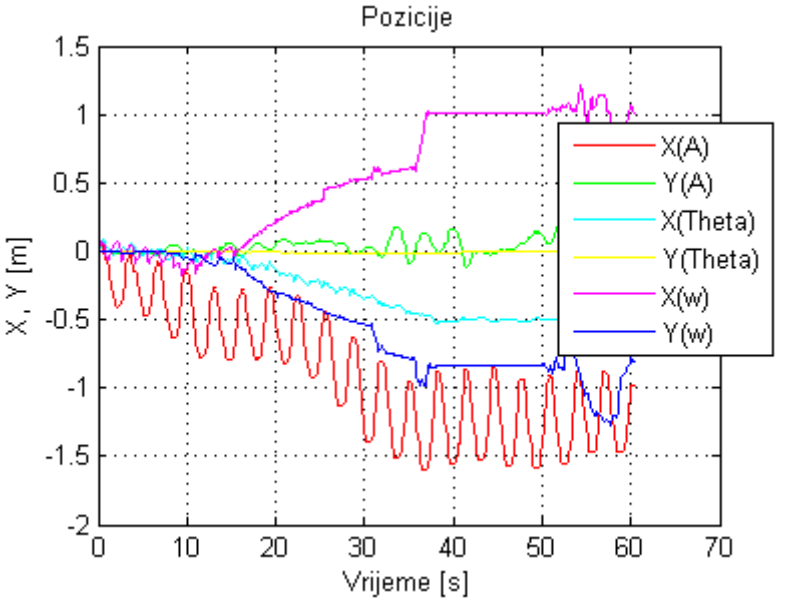


Slika 4.7 Pozicije modula za testirane regulatore uz Kp=0.5 i Ki=0.5 + anti-windup

Iz rezultata simulacije prikazanih na *Slici 4.7* vidljivo je da *anti-windup* zaista djeluje, te je omogućio je kretanje modula s dosegom većim nego za Kp=0.5 i Ki=0, za regulatore temeljene na upravljanju amplitudom i faznim pomakom. Što se tiče regulatora

temeljenog na upravljanju kružnom frekvencijom, sa slike je vidljivo da ni u ovom slučaju ne daje zadovoljavajuće rezultate.

Sljedeća simulacija obavljena je za parametre regulatora Kp=2 i Ki=0, a rezultati su prikazani na *Slici 4.8.*



Slika 4.8 Pozicije modula za testirane regulatore uz Kp=2 i Ki=0

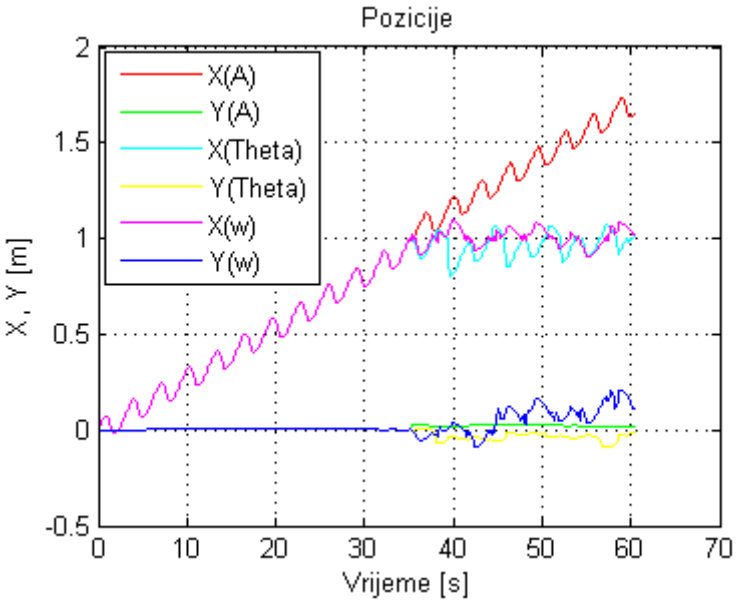
Iz rezultata možemo vidjeti da modul nije upravljiv ni za jedan od testiranih regulatora. Uzrok je previsok iznos pojačanja koje parametre aktuacije postavlja van optimalnog područja djelovanja.

Iz provedenih simulacija možemo zaključiti da, iako su regulatori temeljeni na upravljanju amplitudom i faznim pomakom pokazali zadovoljavajuće ponašanje, zbog naglog smanjenja brzine kretanja sa smanjenjem signala pogreške ne omogućuju dostizanje referentne pozicije. Tim više što rezultati aktuacije ovise o zadanoj referenci, pa iako su rezultati za referencu X=1 bili zadovoljavajući, za više iznose reference modul se pokazao neupravljivim jer upravljački signal poprima veće iznose koji parametre aktuacije postavljaju van optimalnog područja rada, slično kao kod

prevelikog iznosa pojačanja. Za sve provedene testove regulator temeljen na upravljanju kružnom frekvencijom pokazao se neprikladnim. Zbog svega navedenog, kao i činjenice da su u *Poglavlju 3* ustanovljeni optimalni parametri za aktuaciju modula, normiran je iznos signala pogreške. Kada pogreška postoji signal pogreške iznosi jedan, a kada signal pogreške padne na nulu iznosi nula. Ovakvo rješenje uklanja spomenuti problem preupravljanja za više iznose reference, smanjuje entropiju u vezi gibanja modula jer se modul giba po već ustanovljenim pravilima. Programska implementacija ovoga rješenja dana je na *Slici 4.9.*

```
--PI X:
    e_x=(target_x-x_male)/math.abs(target_x-x_male)
        if math.abs(int_x)>1 then
            int_x=0
        else
            int_x=int_x+e_x*dt
        end
    c_x=Kp*e_x+Ki*int_x
```

Slika 4.9 Normiranje iznosa signala pogreške



Slika 4.10 Pozicije za Kp=1 i normiran iznos pogreške

Na *Slici 4.10* prikazani su rezultati simulacije za testirane regulatore, iznos parametara regulatora Kp=1 i normiran iznos signala pogreške. Sa slike je vidljivo da u ovom slučaju svi regulatori dostižu referentnu vrijednost istom brzinom i dok regulator temeljen na upravljanu faznim pomakom i regulator temeljen na upravljanju kružnom frekvencijom osciliraju oko referentne točke, kod regulatora temeljenog na upravljanju amplitudom dolazi do prekoračenja referente točke (engl. *overshoot*). Prekoračenje se javlja kao posljedica načina gibanja modula kojim je vrlo teško postići željenu poziciju tolikom preciznošću. Osmišljena su dva rješenja navedenog problema koja će biti opisana u nastavku.

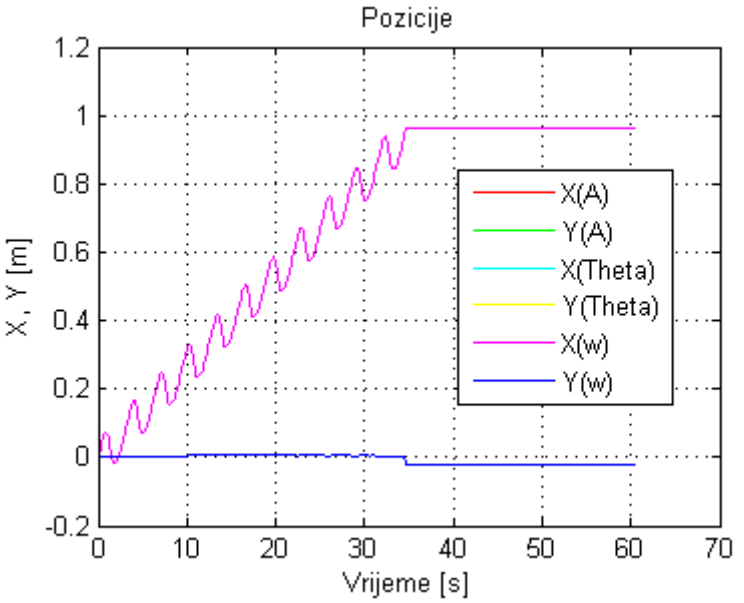
Prvo rješenje je definiranje zaštitne zone u blizini referentne pozicije u kojoj se djelovanje regulatora premosti kako bi se dobilo željeno ponašanje modula. Kod kojim je ovo rješenje implementirano dan je na *Slici 4.11*.

```
if math.abs(x_male)>=math.abs(target_x)-0.05 then
    pm=0
    pf=0
    sim.setJointTargetPosition(male_joint,pm)
    sim.setJointTargetPosition(female_joint,pf)
else
    pm=(a_m*math.sin((c_x*w_m*t)+p_s))
    pf=(a_f*math.sin((c_x*w_f*t)))

    sim.setJointTargetPosition(male_joint,pm)
    sim.setJointTargetPosition(female_joint,pf)
end
```

Slika 4.11 Implementacija prvog rješenja problema pozicioniranja

Iz koda se vidi da regulator upravlja robotom sve dok se robot ne približi na pet centimetara od željene pozicije, tada se djelovanje regulatora premosti naredbom za postavljanje aktuatora u inicijalni položaj. U ovom slučaju širina zaštitne zone je pet centimetara. Rezultati simulacije gibanja modula s navedenim rješenjem prikazani su na *Slici 4.12*.



Slika 4.12 Pozicije modula s implementiranim prvim rješenjem problema pozicioniranja

S prethodne slike može se vidjeti da se robot kreće sve do ulaska u zaštitnu zonu gdje je njegovo kretanje zaustavljeno na devedeset šestom centimetru, a tu poziciju zadržava do kraja simulacije. Rezultati simulacije jednaki su za sva tri tipa regulatora.

Drugo rješenje može se nazvati hibridnim rješenjem jer je temeljeno na dva različita izračuna signala pogreške. Sve dok se modul ne približi na definiranu udaljenost od željene pozicije greška se izračunava prema normiranom principu čime se osigurava brzina i očekivan ishod aktuacije. Kada modul prekorači definiranu vrijednost i približi se željenom cilju, signal pogreške izračunava se na klasičan način pomoću čega je osigurana preciznost pozicioniranja. Kod kojim je ovo rješenje implementirano prikazan je na *Slici 4.13*, dok su rezultati simulacije za tri predstavljena regulatora prikazani na *Slici 4.14*.

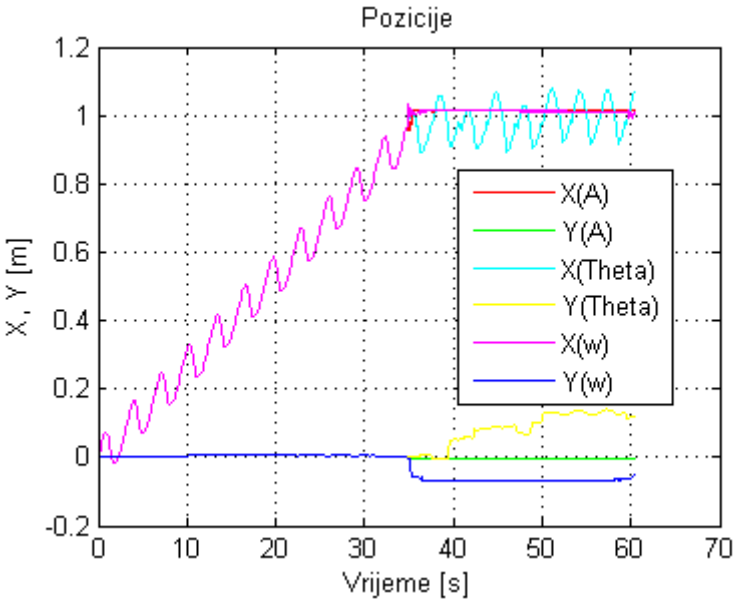
```
--PI X:
    if math.abs(x_male)>=math.abs(target_x)-0.03 then
        e_x=(target_x-x_male)
    else
        e_x=(target_x-x_male)/math.abs(target_x-x_male)
    end
        if math.abs(int_x)>1 then
            int_x=0
        else
            int_x=int_x+e_x*dt
        end
    c_x=Kp*e_x+Ki*int_x

--Aktuacija po sinus nacelu:

    pm=(c_x*a_m*math.sin((w_m*t)+p_s))
    pf=(c_x*a_f*math.sin((w_f*t)))

    sim.setJointTargetPosition(male_joint,pm)
    sim.setJointTargetPosition(female_joint,pf)
```

Slika 4.13 Implementacija hibridnog rješenja problema pozicioniranja



Slika 4.14 Pozicije modula s implementiranim hibridnim rješenjem problema pozicioniranja

Sa *Slike 4.14* vidljivo je da je ponašanje modula za sva tri tipa regulatora jednako do trenutka u kojem je dostignuta tražena pozicija. U tom trenutku aktuatori upravljani regulatorom temeljenom na upravljanju amplitudom zauzet će inicijalni položaj i zadržati ga do kraja simulacije uz točnost pozicioniranja od jednog centrimetra. Slična točnost postiže se i regulatorom temeljnom na upravljanju kružnom frekvencijom, ali uz veće odstupanje po osi Y. Modul s regulatorom temeljenom na upravljanju faznim pomakom neće ostati statičan nego će oscilirati oko referentne pozicije, također uz značajna ostupanja po osi Y.

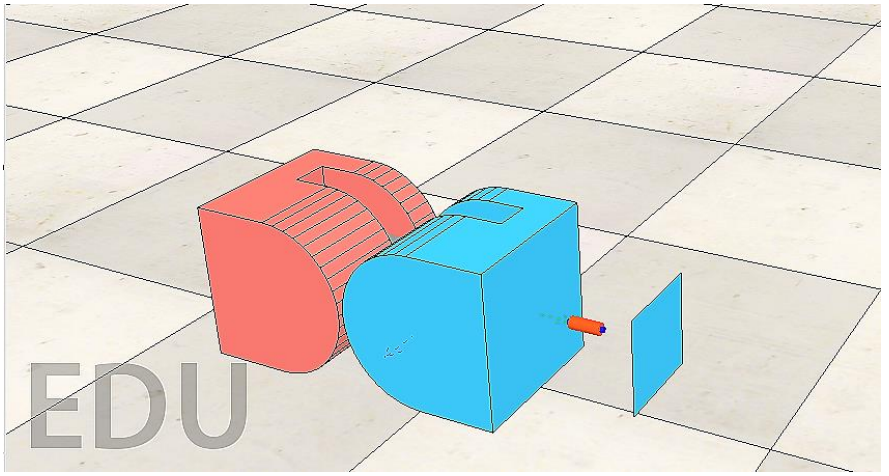
Iz svega navedenog u ovome poglavlju može se izvući zaključak da se linearnim regulatorom, kao što je PID regulator, ne može upravljati nelinearnim sustavom kao što je DTTO modularni robot bez određenih programskih rješenja koja su prezentirana. Najveću prepreku u tome predstavlja specifičan način gibanja robota koji mu onemogućuje precizno pozicioniranje. S obzirom na to da se radi o relativnom inertnom sustavu nije bilo potrebe za implementacijom derivacijskog člana regulatora koji bi svojim djelovanjem samo unosio dodatan šum u upravljački signal te time pogoršao ponašanje robota. Konačno, za regulator kojim će se upravljati pozicijom robota odabran je regulator temeljen na upravljanju kružnom frekvencijom uz korištenje zaštitne zone u blizini referentne točke. Odabran je zato što, osim upravljanja apsolutnom pozicijom robota, omogućuje i automatski odabir smjera kretanja pa nisu potrebna dodatna programska rješenja ako je referentna pozicija negativnog predznaka. Regulator nudi jednaku preciznost pozicioniranja kao i druga dva regulatora za isto opisano rješenje.

5. DODATAN STUPANJ SLOBODE PRI KRETANJU DTTO MODULARNOG ROBOTA

Glavni aktuatori DTTO modularnog robota omogućuju jedan stupanj slobode kretanja. Dodatan stupanj slobode može se postići upotrebom mehanizma za spajanje, kao što je prezentirano u [26]. Iako prezentirano rješenje omogućuje rotaciju modula, proces je izuzetno spor pa mu je potrebno pronaći alternativu.

Jedno od mogućih alternativnih rješenja je dodavanje aktuatora i rotacijske plohe na bazu robota. Prednost ovakvog rješenja je što se robot do željene orijentacije može zarotirati mnogo brže i u manje prostora.

Implementacija predloženog rješenja u V-REP-u prikazana je na *Slici 5.1*.



Slika 5.1 Dodavanje rotacijske plohe u V-REP-u

Rješenje se sastoji od jednog rotacijskog aktuatora i ravne plohe. Dimenzije plohe tako su odabrane da ni u jednom položaju ploha ne izlazi van okvira baze kako ne bi smetala pri kretanju robota. Time je ponešto narušena stabilnost robota u vertikalnom položaju, osobito pri rotaciji, ali s obzirom na to da je ploha postavljena

blisko bazi robota i male je debljine, ona nema značajniji utjecaj na ponašanje robota.

Kod rješenja predloženog u [26] rotacija robota obavlja se prema algoritmu:

1. *Izvuci kukicu mehanizma za spajanje.*
2. *Zarotiraj aktuator aktivnog segmenta kako bi se kukica odvojila od podloge.*
3. *Uvuci kukicu mehanizma za spajanje.*

Ako se pretpostavi da su poznate X i Y koordinate željenog odredišta, rješenje s rotacijskom plohom omogućuje da se one dosegnu na dva načina koja će biti opisana u nastavku.

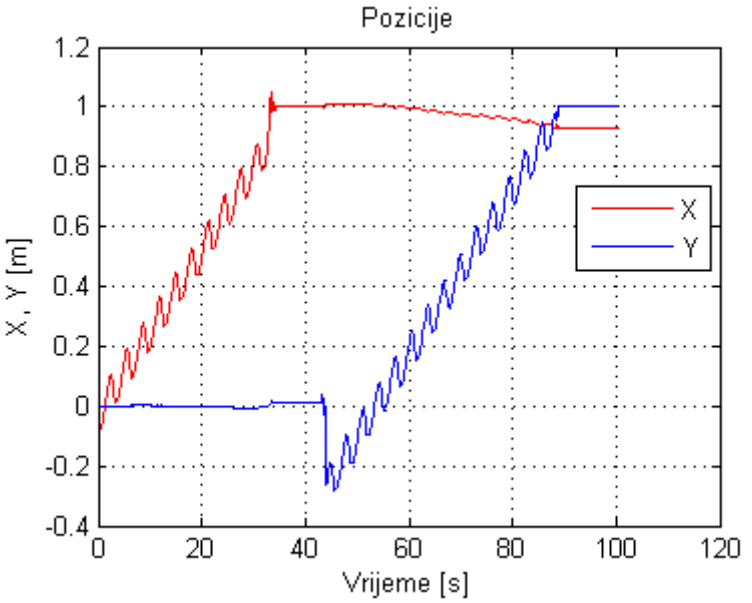
5.1. Hod paralelan koordinatnim osima

Prvi način podrazumijeva gibanje modula paralelno osima referentnog koordinatnog sustava uz rotaciju modula za 90 stupnjeva, kao što je opisano sljedećim algoritmom:

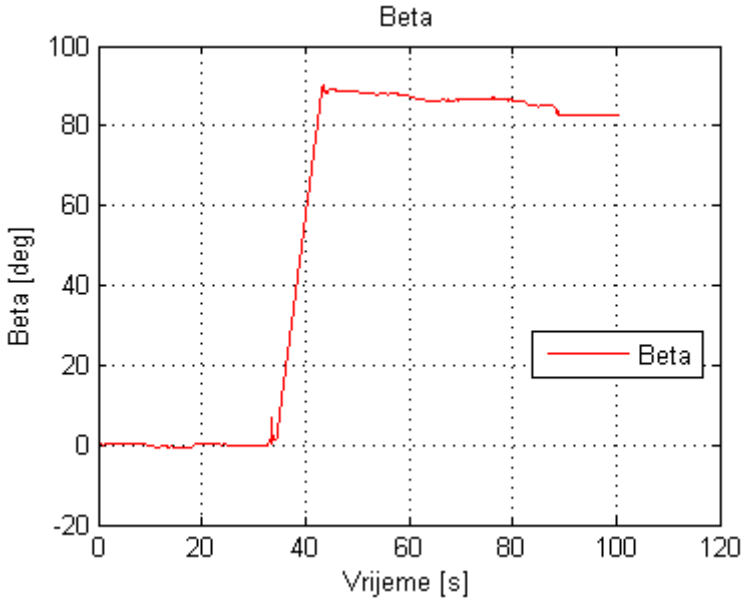
1. *Aktuiraj modul po sinusnom načelu dok se ne dosegne željena X koordinata.*
2. *Uspravi modul.*
3. *Zarotiraj modul za 90 stupnjeva.*
4. *Vrati modul u horizontalni položaj .*
5. *Aktuiraj modul po sinusnom načelu dok se ne dosegne željena Y koordinata.*

Rezultati simulacije za opisani algoritam dani su na *Slici 5.2* i *Slici 5.3*, a za simulaciju su korišteni sljedeći parametri:

- $A = \frac{\pi}{4}$
- $w = 2$
- $\theta_s = \frac{\pi}{3}$
- $target\ x = 1;\ target\ y = 1$



Slika 5.2 Prikaz pozicija za hod paralelan koordinatnim osima

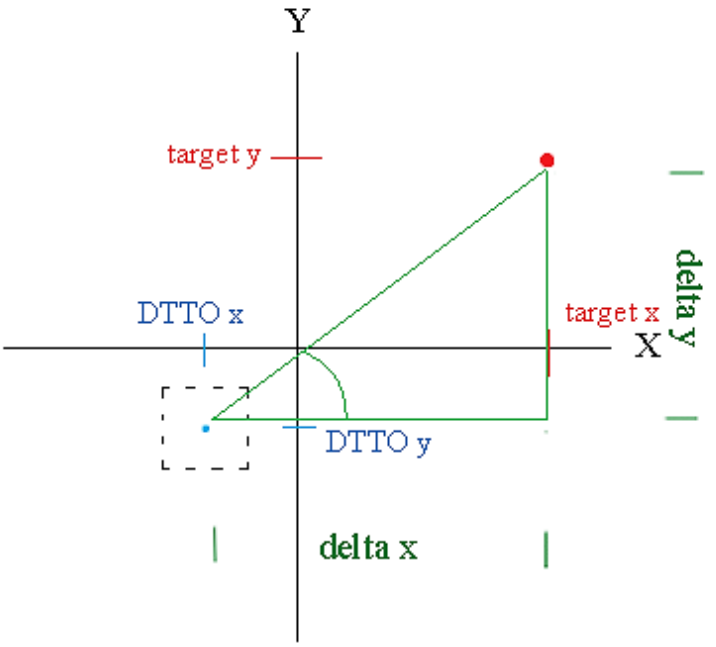


Slika 5.3 Promjena beta kuta orijentacije kod hoda paralelnog koordinatnim osima

S prethodnih slika vidljivo je da se modul iz ishodišta giba po X osi sve do tražene koordinate (X=1) koju dostiže u trenutku t=33 s, pri čemu modul dobro održava smjer gibanja uz minimalna odstupanja po Y osi. Nakon što je dosegnuo traženu X koordinatu modul se uspravlja, rotira za 90 stupnjeva kao što je prikazano na *Slici 5.3*, ponovno vraća u horizontalni položaj te započinje aktuaciju u smjeru osi Y. Pri spuštanju modula prema horizontalnom položaju, zbog nemogućnosti aktuatora rotacijske plohe da strogo zadrži traženu poziciju, javlja se mali pomak u orijentaciji modula ($\Delta\beta < 1°$) koji će rezultirati otklonom u smjeru osi X pri gibanju po osi Y. U konačnici taj otklon uzrokuje pogrešku pozicioniranja po X osi od 7cm. Pozicioniranje po osi Y je praktično bez pogreške, a realizirano je linearinim regulatorom koji je temeljen na upravljanju kružnom frekvencijom uz korištenje zaštitne zone. Vrijeme potrebno da modul dosegne zadane koordinate iznosi t=89 s.

5.2. Hod pod izračunatim kutem

Drugi način zahtijeva izračun kuta pod kojim se modul mora gibati kako bi dosegnuo traženu poziciju, ovisno o trenutnoj poziciji modula. Grafički prikaz veličina od značaja za ovaj proračun dan je na *Slici 5.4*.



Slika 5.4 Karakteristične točke za izračun kuta gibanja

Na prethodnoj slici mogu se uočiti karakteristične točke važne za proračun kuta gibanja, a to su:

- DTTO x i DTTO y – trenutne pozicije modula
- Target x i Target y – željene pozicije modula
- Delta x i Delta y – razlika između željenih i trenutnih pozicija.

Kako bi se dobio traženi kut gibanja potrebno je izračunati označeni kut između hipotenuze i katete zelenog trokuta, što se može napraviti po jednadžbi:

$$\varepsilon = arc\cos\left(\frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}}\right) \tag{5.1}$$

Navedeni kut ovisi o trenutnoj i željenoj poziciji modula pa je potrebno izračunati vrijednosti Δx i Δy za sve moguće pozicije modula, što je ostvareno kodom prikazanim na *Slici 5.5*.

```
if x_male<0 then
    if target_x>0 then
        delta_x=math.abs(x_male)+target_x
    else
        delta_x=math.abs(target_x)+x_male
    end
else
    if target_x>0 then
        delta_x=target_x-x_male
    else
        delta_x=x_male+math.abs(target_x)
    end
end

if y_male<0 then
    if target_y>0 then
        delta_y=math.abs(y_male)+target_y
    else
        delta_y=math.abs(target_x)+y_male
    end
else
    if target_y>0 then
        delta_y=target_y-y_male
    else
        delta_y=y_male+math.abs(target_y)
    end
end
```

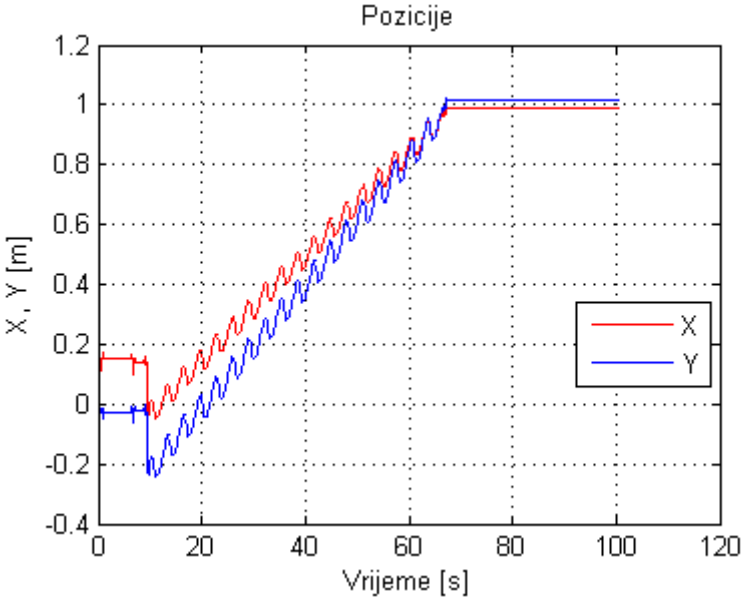
Slika 5.5 Kod za izračun Δx i Δy

Kada je kut izračunat modul se može aktuirati po sljedećem algoritmu:

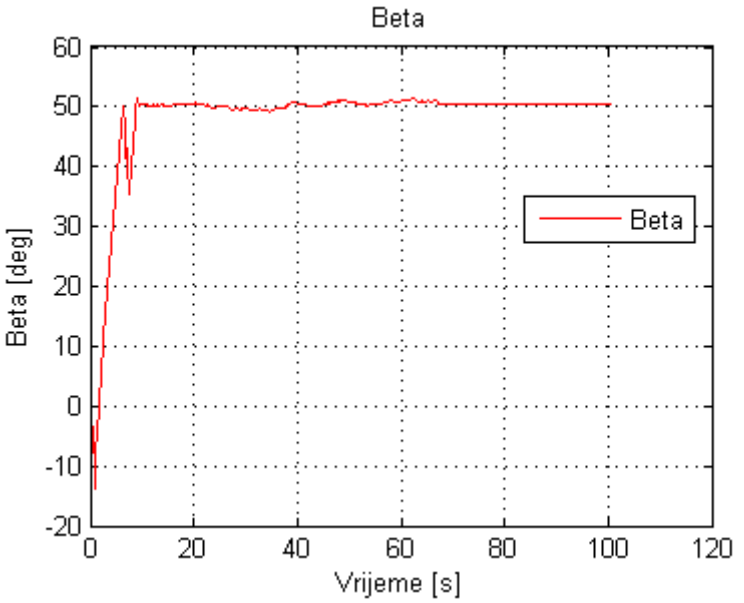
1. *Uspravi modul.*
2. *Zarotiraj modul za izračunati kut vodeći računa o kvadrantu.*
3. *Vrati modul u horizontalni položaj.*
4. *Aktuiraj modul po sinusnom načelu dok se ne dosegnu željene koordinate.*

Rezultati simulacije za opisani algoritam dani su na *Slici 5.6* i *Slici 5.7*, a za simulaciju su korišteni sljedeći parametri:

- $A = \frac{\pi}{4}$
- $w = 2$
- $\theta_s = \frac{\pi}{3}$
- $target\ x = 1;\ target\ y = 1$



Slika 5.6 Rezultati simulacije za okret pod izračunatim kutem



Slika 5.7 Promjena beta kuta orijentacije kod hoda pod izračunatim kutem

Na *Slici 5.6* možemo vidjeti da se odmah na početku simulacije pozicija modula promijeni kao posljedica uspravljanja čime modul više nije u ishodištu koordinatnog sustava. To opravdava razvoj algoritma prikazanog na *slici 5.5.* Da je modul zadržao poziciju u ishodištu koordinatnog sustava, navedeni algoritam ne bi bio potreban jer bi kut kretanja ovisio samo o koordinatama odredišta. Nakon uspravljanja i rotacije za traženi kut, vidljivo na *Slici 5.7*, modul se vratio u horizontalni položaj i započeo aktuaciju po sinusnom načelu prema odredištu. Može se uočiti da se krivulje koje prikazuju gibanje modula sijeku približno u točki 1 što je verifikacija algoritma upravljanja. Otklon koji se javlja prilikom spuštanja modula u horizontalni položaj prisutan je i u ovom slučaju. Tražena pozicija postignuta je uz preciznost unutar 1.5 cm i po X i po Y osi. Vrijeme potrebno da modul dosegne potrebne koordinate iznosi t=67s.

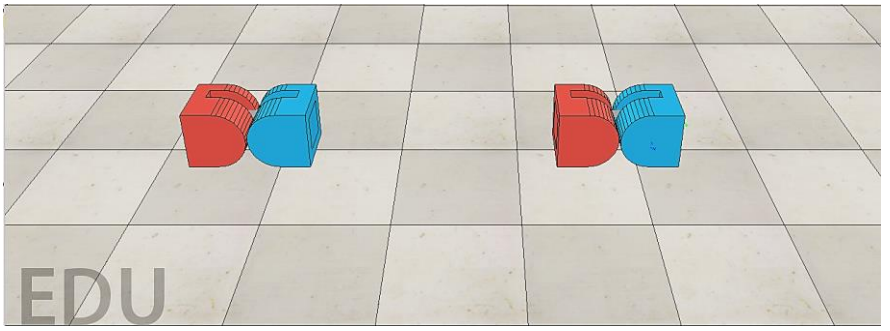
Nakon što su prezentirana oba algoritma za rotaciju modula pomoću rotacijske plohe možemo zaključiti da drugi algoritam omogućava dostizanje tražene pozicije u kraćem vremenu, što je očekivano s obzirom na to da je put koji modul mora prijeći manji. Točnost pozicioniranja kod prvog algoritma ovisit će o preciznosti implementiranih regulatora za pozicioniranje i o pogrešci koja se javlja uslijed odstupanja od željene orijentacije prilikom spuštanja modula u horizontalni položaj. Kod drugog algoritma točnost pozicioniranja, uz pretpostavku da je traženi kut pravilno izračunat, ovisit će isključivo o pogrešci koja se javlja prilikom spuštanja modula i sposobnosti modula da održi pravocrtno gibanje. Oba algoritma pokazala su zadovoljavajuće rezultate za simulirane uvjete.

Iako rotacijska ploha omogućuje bržu promjenu orijentacije modula i ovo rješenje ima određene nedostatke. Prvenstveno, da bi prezentirani algoritmi funkcionirali potrebno je da podloga bude ravna što je u realnim uvjetima rijetkost, u suprotnom potrebno je razviti kompleksnije algoritme upravljanja. Drugi važan nedostatak je što ovo rješenje zahtijeva razvoj novog mehanizma za spajanje modula pa nije kompatibilno s trenutnim izvedbama robota.

6. PRIMJENA RAZVIJENIH UPRAVLJAČKIH ALGORITAMA PRI SPAJANJU DVAJU NEZAVISNIH MODULA

Sve prednosti modularnih robota proizlaze iz interakcije među modulima. Dok jedan modul ima vrlo ograničene mogućnosti kretanja i obavljanja zadataka, sposobnosti više njih povezanih u koordiniranu cjelinu rastu do neslućenih razmjera. Zato je proces pronalaska i spajanja modula ključan u modularnoj robotici.

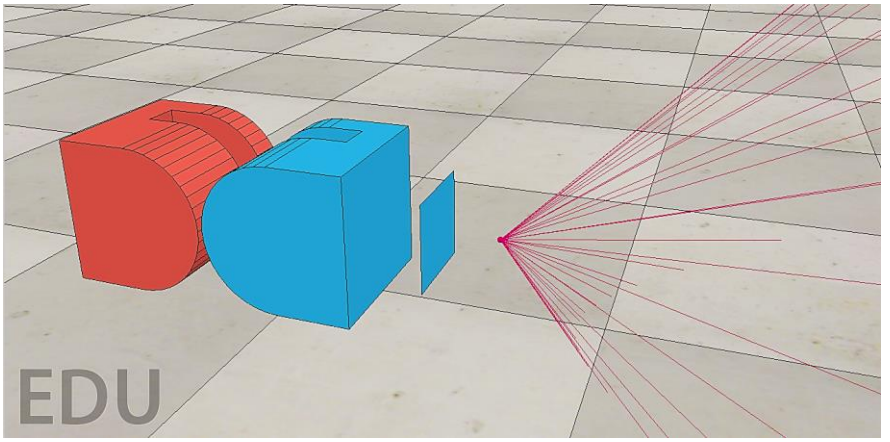
U nastavku je prezentiran proces otkrivanja i spajanja dva DTTO modula. Izgled testne scene prikazan je na *Slici 6.1*.



Slika 6.1 Izgled testne scene za spajanje dva DTTO modula

Sa slike se može vidjeti da je ovo idealan scenarij za spajanje, jer se moduli nalaze na istome pravcu koji odgovara smjeru njihova gibanja. Segmenti koji su predviđeni za spajanje su aktivni i pasivni što omogućuje modulima da se povežu. Kada se želi izvršiti povezivanje modula ovo je pozicija u koju se moduli moraju postaviti kako bi povezivanje bilo moguće zbog čega ovaj scenarij možemo nazvati univerzalnim. U testnoj simulaciji jedan modul će biti statičan, dok će drugi izvesti oktrivanje modula, približiti mu se i izvršiti proces spajanja.

Kako bi povezivanje bilo moguće, potrebno je izvršiti nadogradnju osnovnog simulacijskog modela. Osnovnom modelu DTTO robota dodan je ultrazvučni senzor za mjerenje udaljenosti, a programski je implementiran mehanizam za spajanje. Mehanizam za povezivanje simulirat će elektromagnetna baza robota, implementiran prema [27]. Kada se želi izvršiti povezivanje elektromagnet će biti aktivan, a kada se veza želi raskinuti elektromagnet će biti deaktiviran. Za površinu elektromagneta iskorištena je već ugrađena rotacijska ploha. Izgled navedenog rješenja dan je na *Slici 6.2*.



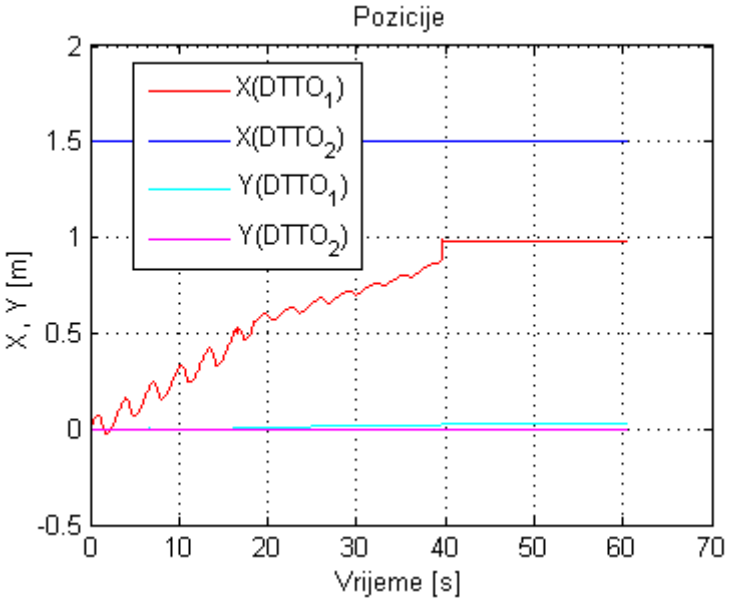
Slika 6.2 Implementacija senzora za mjerenje udaljenosti

Odabrano mjesto za ugradnju senzora za mjerenje udaljenosti je dosta nezgodno, ali jedino moguće. Problem leži u načinu gibanja robota zbog kojeg će tijekom aktuacije, osim svoje okoline senzor detektirati i podlogu po kojoj se giba što može otežati razvoj algoritama za povezivanje. Rješenje leži u postavljanju mrtvih zona za koje se neće uzimati u obzir očitanja senzora. Ovaj je pristup prikladan za simulacijski model i iskorišten u ovome slučaju, a za realnu implementaciju to ne mora biti slučaj te je potrebno iznaći druga rješenja. Senzori za udaljenost ugrađeni su u aktivni i pasivni segment simuliranih modula.

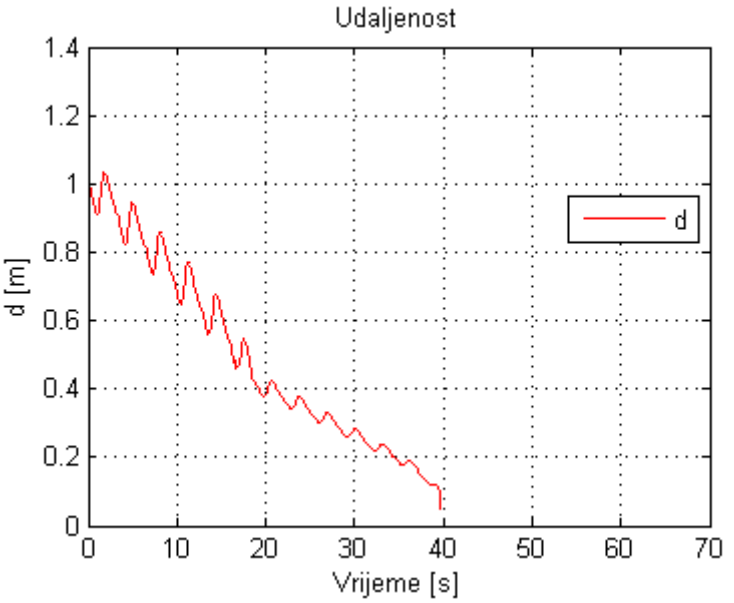
Za aktuaciju mobilnog modula korišteni su sljedeći paramteri:

- $A = \frac{\pi}{4}$
- $w = 2$
- $\theta_s = \frac{\pi}{3}$
- $Kp = 1.$

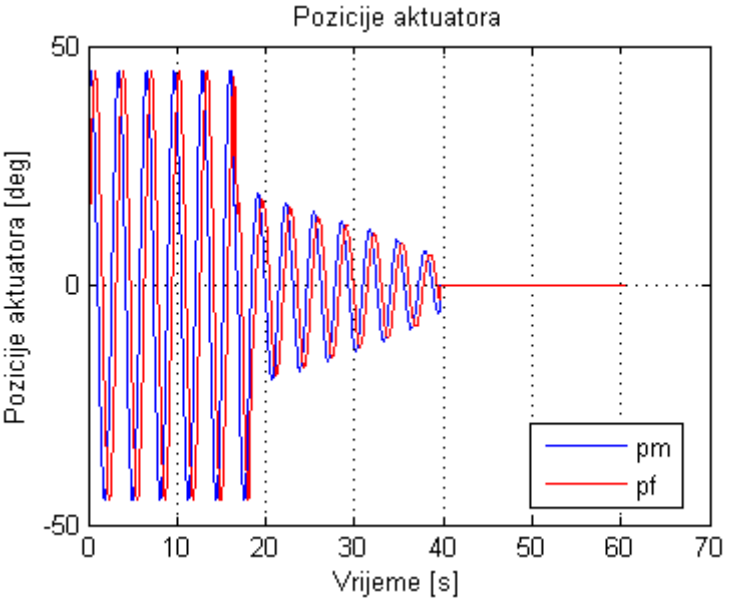
Za upravljanje pozicijom modula korišten je regulator temeljen na upravljanju amplitudom s hibridnim rješenjem, opisan u *Poglavlju 4.*, pri čemu je kao referenca poslužila izmjerena udaljenost iz ultrazvučnog senzora. Odabran je jer je potrebno da pri spajanju modula lica njihovih baza budu paralelna. Kako se detektirana udaljenost bude smanjivala, smanjivat će se i amplituda gibanja s tendencijom odlaska u nulu, čime je navedeni uvjet ispunjen. Inicijalna udaljenost između modula iznosi 1.5 m. Pozicije modula za simulaciju spajanja prikazane su na *Slici 6.3*, promjena detektirane udaljenosti na *Slici 6.4*, a položaj aktuatora na *Slici 6.5*.



Slika 6.3 Pozicije modula za simulaciju spajanja

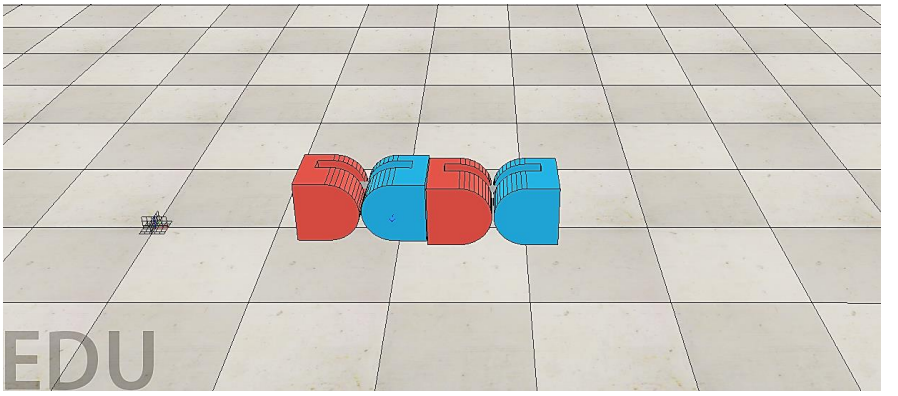


Slika 6.4 Detektirana udaljenost



Slika 6.5 Položaji aktuatora za simulaciju spajanja

Na *Slici 6.3* DTTO_1 modul predstavlja mobilni modul, a DTTO_2 statični modul. Kada se pojavi signal sa senzora udaljenosti modul će započeti s aktuacijom u smjeru detektiranog modula, što je prikazano crvenom linijom na grafu. Kako se mobilni modul približava statičnom tako se smanjuje detektirana udaljenost prikazana na *Slici 6.4*. U trenutku t=20 s vidi se promjena u vladanju aktuatora uzrokovana implementiranim regulatorom. Hibridni regulator je namješten tako da se pogreška do detektirane udaljenosti od 0.5 m računa po normiranom načelu, a za manje udaljenosti ona predstavlja signal pogreške. Posljedica toga je smanjenje amplitude gibanja aktuatora uočljivo na *Slici 6.5*, čime je smanjena i brzina gibanja modula. Modul nastavlja gibanje sve do t=39 s gdje se njegova pozicija trenutno promijeni pod djelovanjem privlačne magnetske sile koja spoji module. Od toga trenutka moduli ostaju spojeni i statični, kao na *Slici 6.6*.



Slika 6.6 Rezultat spajanja dva modula

Na *Slici 6.3* uočljiv mali pomak mobilnog modula po osi Y, posljedica je načina na koji je modul aktuiran. Prilikom gibanja, glavni aktuatori preko tijela robota djeluju na podlogu i odguruju se čime se ostvaruje pomak modula. U tom procesu aktuatori modula u jednom djeliću vremena djeluju jedan protiv drugog rezultat čega su sile koje dovode do malih odstupanja modula po osi Y. Odstupanje dovodi do toga da moduli nisu savršeno poravnani u trenutku

spajanja, kao što je vidljivo na *Slici 6.6*. To možda predstavlja problem kod virtualne izvedbe robota, ali kod realne izvedbe to nije tako. Kod stvarne izvedbe robota za poravnanje su zaduženi neodimijski magneti. Na bazu robota, u svaki od četiri kuta ugrađen je po jedan magnet čime se postiže savršeno poravnanje robota u trenutku spajanja. Savršeno poravnati moduli su preduvjet za ispravno funkcioniranje sustava za povezivanje.

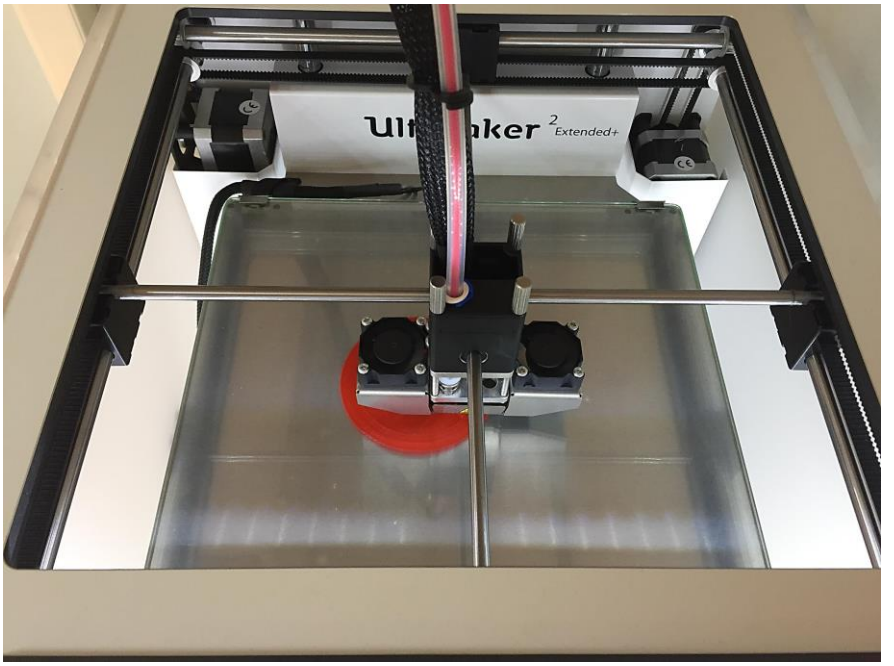
7. IZRADA MODULA DTTO ROBOTA

Nakon što su opisani algoritmi upravljanja i konstrukcija robota, predstavit će se proces izrade jednog DTTO v2 modula. Modul je izrađen prema uputama danima u [28].

Izrađeni DTTO modul sastoji se od:

- 1x Arduino Nano
- 22x Plastična dijela
- 2x TowerPro MG92B servomotora
- 3x TowerPro MG90s servomotora
- 1x Bluetooth HC-05 modul
- 1x NRF2401 modul
- 2x Li-Po baterije 3,7V 600mAh
- 1x LM317 regulator napon
- 24x Neodimijska magneta
- 40x M1.7x4mm vijka
- 1x Prekidača
- 4x Otpornika od 520 ohma
- 3x Elastične gumice

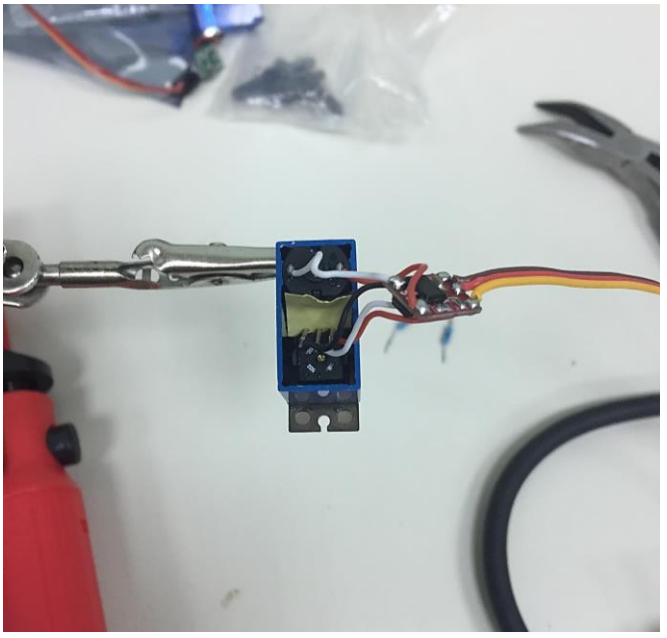
Prvi korak u izradi modula je printanje i obrada svih plastičnih dijelova. Datoteke potrebne za printanje preuzete su s [29], dok je popis dijelova dostupan u [30]. Print je obavljen na Ultimaker 2 Extended + printeru. Više informacija o printeru moguće je pronaći na [14]. Za print je odabrana PLA tehnologija koja je pokazala određene nedostatke. Prvenstveno slaba mehanička svojstva koja su bila posljedica izgubljenih slojeva prilikom printanja. Uzrok je ovisnost kvalitete printa o ambijentalnoj temperaturi. Ultimaker 2 Extended + omogućuje print i u ABS tehnologiji koja bi trebala pružiti bolja mehanička svojstva. Proces printanja jednog od dijelova robota može se vidjeti na *Slici 7.1*.



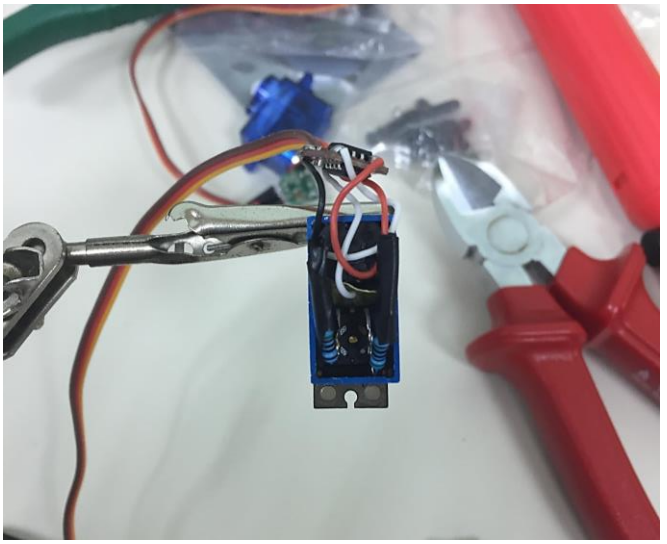
Slika 7.1 Printanje dijela DTTO robota

Nakon što su svi dijelovi isprintani i obrađeni potrebno je izvršiti prilagodbu TowerPro MG92B servomotora[25]. Oni u izvornom obliku imaju hod od -90 stupnjeva do 90 stupnjeva, s incijalnim položajem u nuli. Kako bi bili prikladni za aktuaciju DTTO modula potrebno je opseg pokreta proširiti do 180 stupnjeva. Proširenje opsega pokreta postiže se lemljenjem otpornika od 520 ohma na stezaljke potenciometra servomotora. Na *Slici 7.2* prikazani su MG92B servomotori, a na *Slici 7.3* i *Slici 7.4* postupak rastavljanja i prilagodbe. U prvoj verziji DTTO robota [6] korišteni su TowerPro SG92R servomotori o kojima je više informacija moguće pronaći na [31]. U drugoj verziji SG92R motore bilo je potrebno zamijeniti zbog porasta dimenzija modula. Veće dimenzije zahtijevale su i veći okretni moment te kvalitetniji sustav prijenosa što je osigurano MG92B servomotorima.

Slika 7.2 TowerPro MG92B servomotori

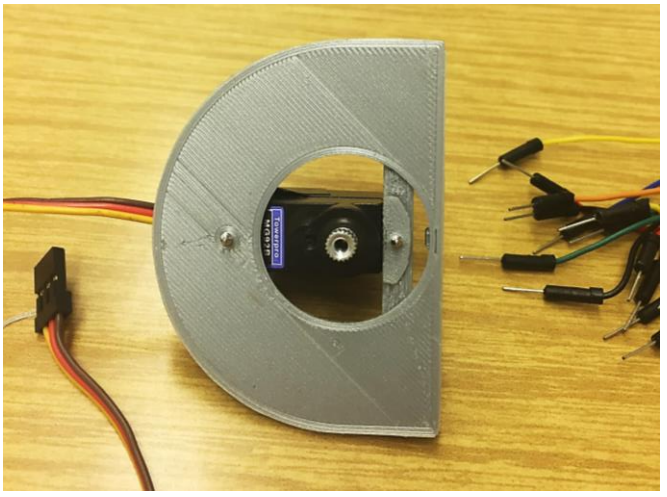


Slika 7.3 Rastavljanje TowerPro MG92B servomotora



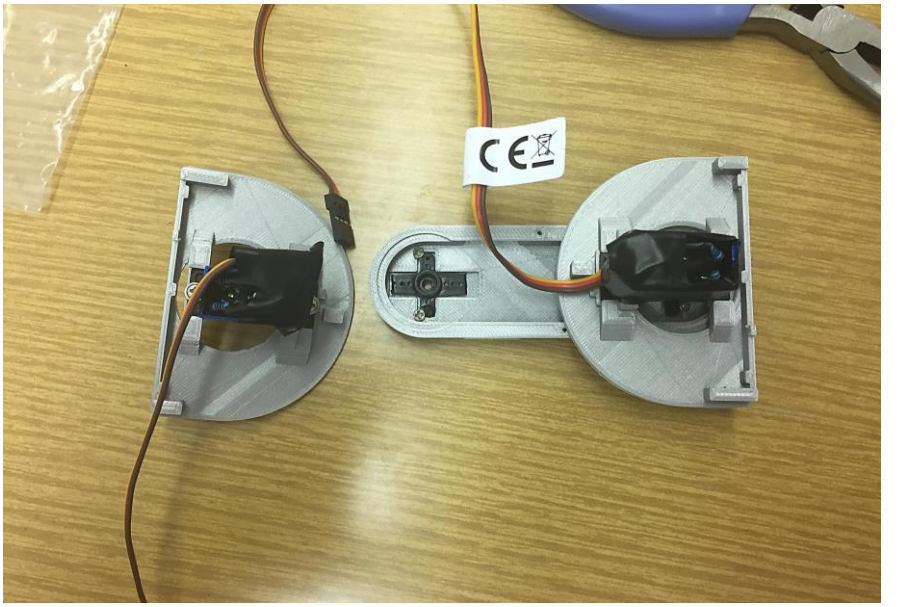
Slika 7.4 TowerPro MG92B servomotor s ugrađenim otpornicima

Kada je modifikacija motora završena, može se pristupiti sastavljanju modula. Prvi korak u sastavljanju modula je ugradnja MG92B servomotora na isprintani dio *broj 4*, prikazano na *Slici 7.5*. Kako ne bi smetali pri aktuaciji, vijke koji vire iz plastike potrebno je skratiti.



Slika 7.5 Ugradnja pogonskog servomotora

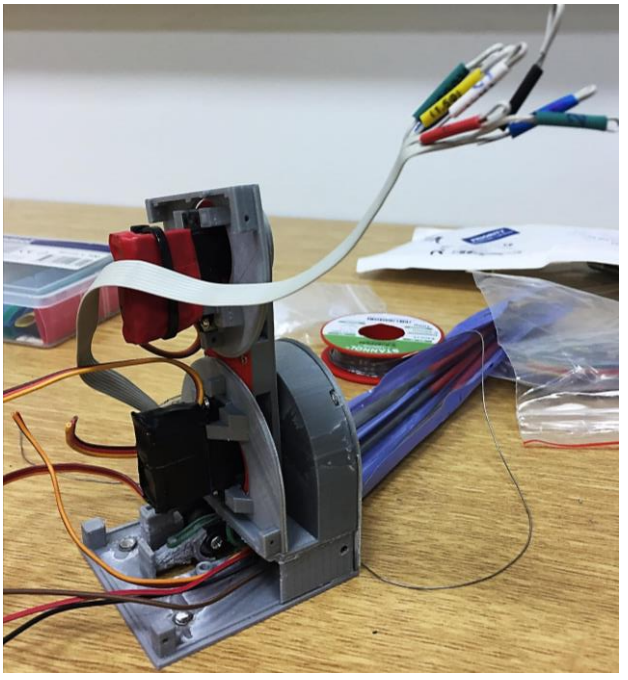
S pogonskim servomotorima dolaze i plastični križići koji služe za prijenos rotacijskog gibanja s motora na tijelo. Njih je potrebno vijcima učvrstiti na osovinu modula, odnosno dio *broj 12*. U ovome koraku važno je križiće pravilno pozicionirati kako bi, jednom kada se modul sastavi, bilo ostvareno gibanje u punome opsegu pokreta. Izgled instalacije križića i pogonskog servomotora dan je na *Slici 7.6.*



Slika 7.6 Instalacija križića i pogonskih servomotora na osovinu modula

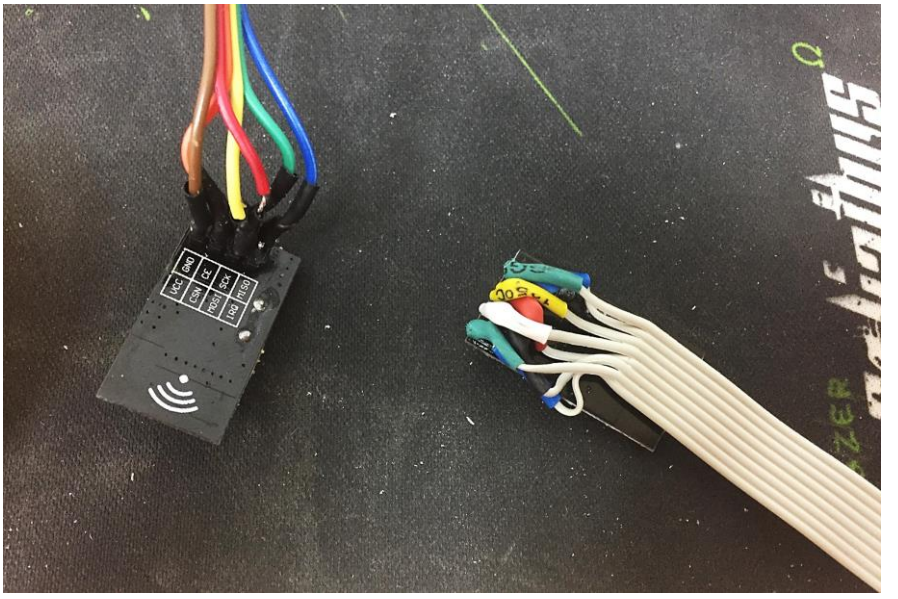
Kada su motori pravilno centrirani i gibanje testirano, može se nastaviti s pričvršćivanjem ostalih plastičnih dijelova. Prvi na redu je dio *broj 3* koji se uz pomoć četiri vijka spoji s dijelom *broj 4.* U ovome koraku modul poprima prve obrise i formiraju se aktivni (muški) i pasivni (ženski) segment modula. Sljedeći korak u izradi modula je instalacija magneta sustava za povezivanje. Zadatak magneta je osigurati poravnanje modula pri spajanju. Ukupno je 24 neodimijska magneta potrebno zalijepiti na dijelove *broj 5*, *broj 7* i *broj 9*. U ovome koraku važno je paziti na orijentaciju magneta

kako bi se pri spajanju modula osigurala privlačna sila samo između aktivnog i pasivnog segmenta. Aktivni dio mehanizma za spajanje čine tri TowerPro MG90s servomotora [32]. Oni pružaju nešto manji okretni moment od pogonskih MG92B servomotora, ali sasvim dovoljan kako bi održao čvrstu vezu među modulima. Na sva tri TowerPro MG90s servomotora ugrađena je kukica, poluga i elastična gumica koje imaju zadatak uspostavljanja ili raskidanja veze između pasivnih i aktivnih segmenata modula u interakciji. Servomotori mehanizma za povezivanje ugrađuju se na bazu aktivnog segmenta (dio *broj 5*) i njegove bočne stranice, dijelovi *broj 7* i *broj 9*. Kada su servomotori ugrađeni na za to predviđena mjesta, dijelovi *broj 5* i *broj 9* mogu se povezati s dijelom *broj 3*. Izgled modula u ovoj fazi prikazan je na *Slici 7.7*.



Slika 7.7 Izgled modula sa spojenim dijelovima broj 5 i broj 9

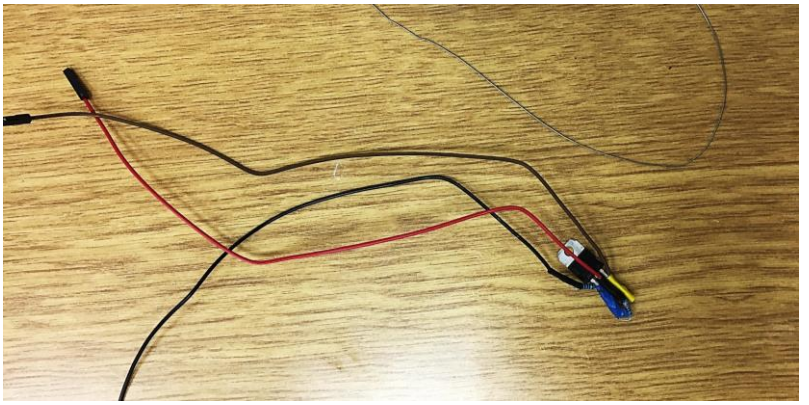
Na *Slici 7.7* može se uočiti mehanizam za povezivanje ugrađen u bazu robota. Sivi plosnati kabel uočljiv na prethodnoj slici pripada NRF2401 modulu instaliranom s desne strane aktivnog segmenta. Upotreba plosnatog kabela bila je potrebna zbog ograničenog prostora unutar aktivnog segmenta. Korištenjem plosnatog kabela omogućeno je da se kabel provuče preko servomotora mehanizma za povezivanje čime je osiguran dodatan prostor u ostatku segmenta. Sada je još preostalo lemljenje kabela na ostale elektroničke komponente te povezivanje komponenti u cjelinu. Prvi na redu je HC-05 bluetooth modul na *Slici 7.8* prikazan zajedno s NRF2401 radiomodulom.



Slika 7.8 HC-05 bluetooth modul i NRF2401 radio modul

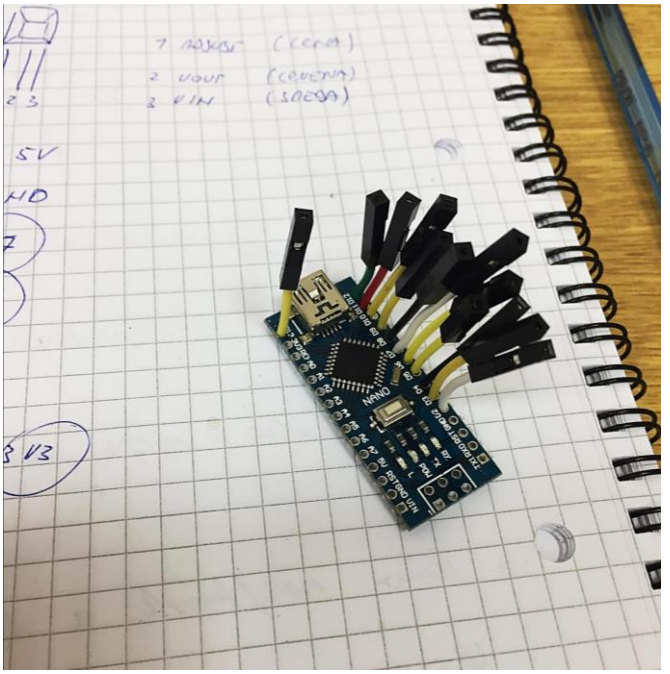
Kod lemljenja korišteni su raznobojni kabeli kako bi se modul kasnije lakše povezao s Arduinom i kako bi se minimizirala vjerojatnost pogreške. Sljedeća na redu elektronička komponenta je LM317 regulator napona [33]. Za napajanje modula koriste se dvije Li-Po baterije napona 3,7 V i kapaciteta 600 mAh. Njihovim povezivanjem u seriju ostvaren je napon od 7.4 V. Iako je taj napon

prikladan za napajanje Arduina, prevelik je za napajanje servomotora čiji je nominalni napon od 4.8V do 6.6V, pa se servomotori napajaju preko regulatora napona. LM317 je podesivi regulator napona, a vrijednost izlaznog napona određuje se dodavanjem otpornika između stezaljki regulatora. Između izlazne stezaljke i stezaljke za prilagodbu zalemljen je otpornik od 330 ohma, dok je između stezaljke za prilagodbu i negativnog pola baterija zalemljen otpornik od 1.2k ohma. Time je ostvaren izlazni napon regulatora od približno 6V. Izgled LM317 regulatora napona sa zalemljenim otpornicima dan je na *Slici 7.9.*



Slika 7.9 LM317 regulatora napona

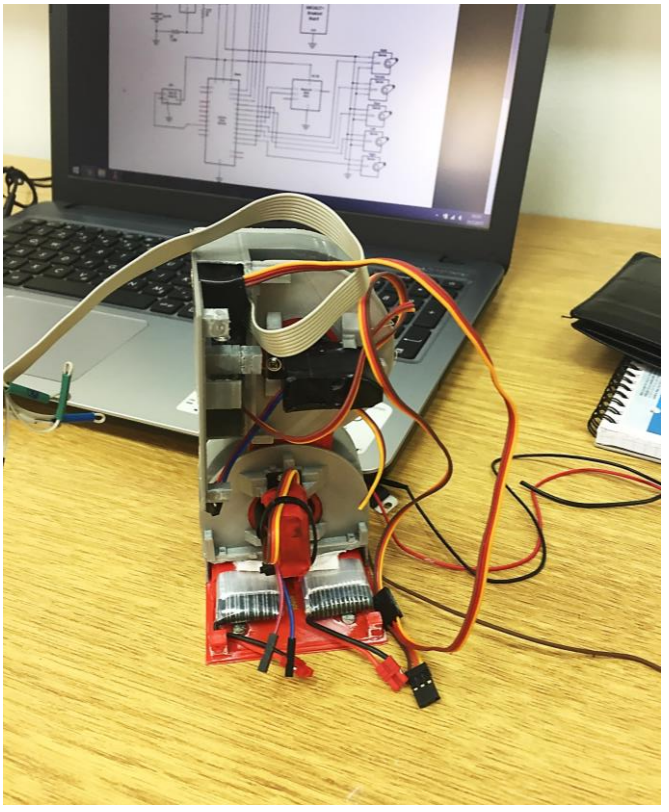
Uloga središnjeg upravljačkog uređaja dana je Arduinu Nano [34]. Na njega će se povezati svi komunikacijski uređaji i upravljačke stezaljke servomotora. Kako bi spajanje komponenti bilo lakše i olakšala se zamjena u slučaja kvara, na Arduino su zalemljene ženske priključnice, vidljivo na *Slici 7.10*. Ugradnjom priključnica na Arduino omogućeno je spajanje komponenti jednostavnim priključivanjem muških priključnica koje su zalemljenje na spojne kabele komponenti. Ovakvim rješenjem izbjegnuto je izravno lemljenje spojnih kabela komponenti na Arduino u ograničenom prostoru aktivnog segmenta modula.



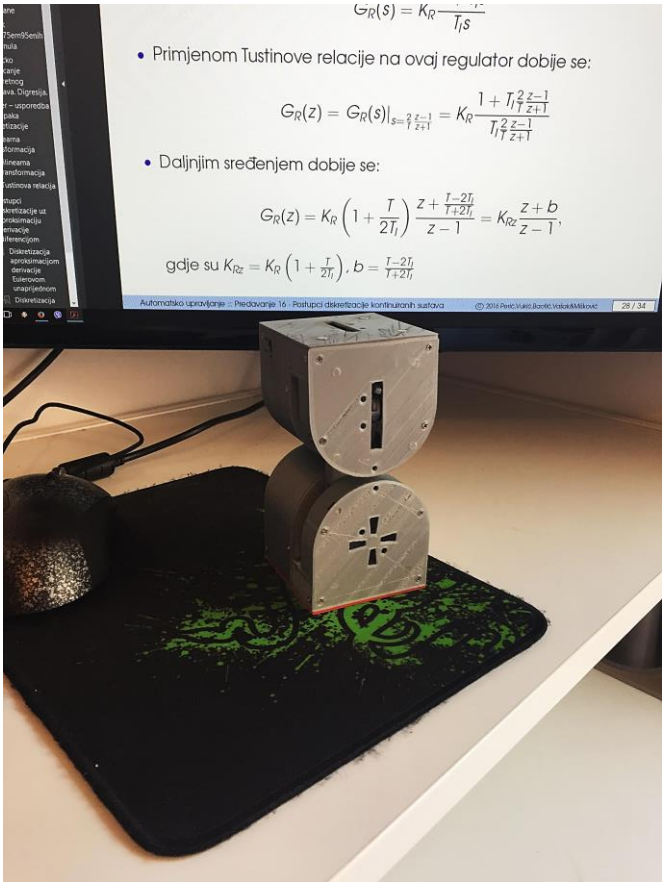
Slika 7.10 Arduino Nano DTTO modularnog robota

Preostaje povezati sve elektroničke komponente prema shemi spajanja preuzetoj sa službene stranice projekta [35] i smjestiti ih unutar modula. Kako su sve komponente osim baterija smještene u aktivnom segmentu, postupak rezultira šumom kabela koju je potrebno organizirati tako da ne ometaju gibanje modula i rad mehanizma za povezivanje. Napajanje je iz pasivnog segmenta u aktivni dovedeno kroz osovinu modula. Pri provlačenju kabela za napajanje važno je posvetiti pozornost položaju i duljini kabela jer se nalaze tik u glavne aktuatore. Ako se položaj ili duljina kabela krivo odaberu, rotacija segmenata uzrokovat će struganje kabela između pomičnih dijelova robota što će dovesti do trošenja izolacije i u konačnici presijecanja kabela. Kada su sve elektroničke komponente ugrađene može se zatvoriti konstrukciju robota preostalim isprintanim dijelovima. Na *Slici 7.11* prikazan je izgled modula prije ugradnje elektroničkih komponenti, a na *Slici 7.12* konačan izgled dovršenog modula.

Nakon završetka poslova vezanih uz sklopovlje, potrebno je modul priključiti na računalo i isprogramirati. Arduino kod za aktuaciju modula razvio je autor projekta Alberto Molina Pérez. Kod je dodan u *Prilog 1*, a dostupan je za preuzimanje na [36]. U kodu je moguće odabrati parametre aktuacije i način gibanja modula. Kada su svi parametri odabrani, kod se pomoću Arduino programa podigne na Arduino Nano. Ako je sve pravilno spojeno i isprogramirano, robot će započeti s gibanjem prema zadanim parametrima. Time je proces izrade modula dovršen.



Slika 7.11 Izgled modula prije ugradnje komponenti



Slika 7.12 Konačan izgled DTTO modularnog robota

8. ZAKLJUČAK

U radu je predstavljen DTTO robot. Opisane su njegova konstrukcija, način gibanja, proces izrade i simulirano ponašanje u virtualnom okruženju za simuliranje dinamičkih sustava V-REPu. Bit projekta sadržana je u njegovoj jednostavnosti i otvorenosti. Zahvaljujući *open source* pristupu i niskoj cijeni izrade DTTO je postao dostupan široj javnosti. To ga čini prvim takvim projektom iz područja modularne robotike u svijetu.

Iako naizgled jednostavan, u DTTO projekt ugrađena su znanja iz područja matematike, elektronike, mehatronike i računarstva. Odabir vrste kretanja i aktuatora kojima će se kretanje ostvariti je osnova od koje kreće razvoj svakog robota. Zato je najveći dio ovoga rada posvećen upravo načinu gibanja robota i principima koji iza njega stoje. U *Poglavlju 3.* pokazano je da se relativno kompleksni principi gibanja prisutni u prirodi mogu aproksimirati jednostavnim sinusnim gibanjem. Sinusno gibanje je potom implementirano na prethodno razvijen virtualni model DTTO robota.

U V-REP- u je simulirano ponašanje robota za različite parametre sinusnih signala kojima su pobuđeni rotacijski aktuatori, a rezultati su predstavljeni u obliku grafova i tablica. Iz rezultata simulacija mogu se izvući sljedeći zaključci:

1. Brzina modula u ovisnosti o faznom pomaku između aktuatora ponaša se približno sinusoidalno.
2. Na različitim kružnim frekvencijama sinusoida ovisnosti brzine modula o faznom pomaku ekstreme i nultočke ostvarivat će na različitim faznim pomacima.
3. Optimalni rezultati gibanja iz perspektive prijeđenog puta ostvareni su za $w = 2\,\frac{rad}{s}$ i $\theta_s = \frac{\pi}{3}\ rad = 60°$.

4. Konstrukcija robota postavlja granice za amplitude koje se mogu ostvariti na $A = \frac{\pi}{2}\ rad = 90°$ i$A = -\frac{\pi}{2} rad = -90°$.
5. Neovisno o faznom pomaku maksimalna brzina kretanja ostvaruje se za $A = \frac{\pi}{4} rad = 45°$.
6. Brzina modula u ovisnosti o amplitudi različita je za različite kružne frekvencije, a maksimalne brzine ostvarene su za $A = \frac{\pi}{4}\ rad = 45°$ pri $w = 2\frac{rad}{s}$ i $A = \frac{\pi}{6}\ rad = 30°$ pri $w = 3\frac{rad}{s}$.
7. Za amplitude preko $A = \frac{\pi}{3}\ rad = 60°$ ostvaruje se neželjeno vladanje robota za sve kružne frekvencije.
8. Povećanjem kružne frekvencije smanjuje se raspon amplituda za koje se ostvaruje željeno ponašanje robota.
9. Najveća brzina kretanja u ovisnosti o kružnoj frekvenciji ostvaruje se za $w = 2\frac{rad}{s}$.
10. Više kružne frekvencije ($w = 4, w = 5, w = 6\ i\ w = 7$) pokazale su se neprikladnima za aktuaciju robota jer dovode do neželjenog gibanja.

Nakon postizanja najniže razine upravljanja, u *Poglavlju 4.* razvijeni su upravljački algoritmi za upravljanje pozicijom robota temeljeni na linearnim regulatorima. Predstavljena su tri tipa regulatora: regulator temeljen na upravljanju amplitudom, regulator temeljen na upravljanju kružnom frekvencijom i regulator temeljen na upravljanju faznim pomakom između aktuatora. Sva tri regulatora pokazala su se neprikladnima u izvornom obliku jer ne omogućuju dostizanje referentne pozicije. Stoga su predstavljena dva rješenja problema pozicioniranja. Prvo rješenje podrazumijeva određivanje zaštitne zone u okolici referentne pozicije u kojoj se djelovanje regulatora premosti kako bi se dobilo željeno ponašanje modula. Ono se pokazalo učinkovitim i omogućilo je uporabu sva tri tipa regulatora. Drugo rješenje nazvano je hibridnim rješenjem

jer je temeljeno na dva načina izračuna signala pogreške. Do određene udaljenosti od referente pozicije signal pogreške izračunava se po normiranom načelu čime su osigurane brzina i točnost gibanja modula, a kada se modul približi unutar definirane udaljenosti pogreška se izračunava prema klasičnom načelu čime je osigurana preciznost pozicioniranja. Hibridno rješenje pokazalo se učinkovitim rješenjem problema pozicioniranja s dva različita ishoda. Za regulator temeljen na upravljanju amplitudom i regulator temeljen na upravljanju kružnom frekvencijom hibridno rješenje omogućuje zaustavljanje robota u okolici referente točke, dok će za regulator temeljen na upravljanju faznim pomakom modul oscilirati oko referente pozicije. Konačno, kao regulator koji će se implementirati u razvijeni V-REP model DTTO robota odabran je regulator temeljen na upravljanju kružnom frekvencijom sa zaštitnom zonom jer omogućuje automatski odabira smjera gibanja robota u ovisnosti o predznaku referente pozicije.

Kao velik nedostatak DTTO robota pokazala se ograničenost njegova gibanja. Zato je u *Poglavlju 5.* predstavljeno dodavanje dodatnog stupnja slobode. Rješenje je temeljeno na dodavanju rotacijske plohe na bazu robota koja će omogućiti promjenu orijentacije robota. Predstavljeno rješenje implementirano je u V-REPu te su razvijena dva algoritma za njegovu primjenu. Prvi algoritam podrazumijeva gibanje robota u ravnini s koordinatnim osima do dostizanja tražene pozicije, uz okret od 90 stupnjeva. Drugo rješenje temeljeno je na izračunu kuta gibanja ovisno o traženoj i trenutnoj poziciji. Predstavljeni su rezultati simulacije za oba algoritma. Prvi algoritam pokazao je nešto veću pogrešku pozicioniranja. Uzrok pogreške je nemogućnost aktuatora rotacijske plohe da strogo održi poziciju prilikom spuštanja modula u horizontalni položaj. Drugi algoritam pokazao je veću preciznost pozicioniranja i omogućio dostizanje željene pozicije u kraćem vremenu, što je i očekivano jer je put koji modul mora prijeći kraći. Dodavanje rotacijske plohe omogućilo je dodatan stupanj slobode

uz određene nedostatke. Prvenstveno moduli s ovim rješenjem ne bi bili kompatibilni s trenutnom verzijom robota, a bilo bi potrebno razviti i novi sustav za povezivanje. Veliki nedostatak je i činjenica da bi rotacijska ploha u verziji kojoj je predstavljena omogućavala dodatan stupanj slobode samo na ravnoj podlozi.

Snaga modularnih robota proizlazi iz njihove brojnosti. Što je modula više, to je više mogućnosti njihove primjene. Zato je algoritam za povezivanje modula ključan u njihovoj eksploataciji. U *Poglavlju 6.* predstavljen je najjednostavniji, univerzalni slučaj povezivanje dvaju modula. Mehanizam za spajanje implementiran je u obliku elektromagneta. Ugrađen je i ultrazvučni senzor udaljenosti koji mobilnom modulu omogućuje pronalazak statičnog modula. Kada mobilni modul detektira statični modul, započinje gibanje u njegovom smjeru. Ključan faktor u ovome procesu je algoritam za pozicioniranje, a kao najprikladniji regulator pokazao se regulator temeljen na upravljanju amplitudom sa hibridnim rješenjem problema pozicioniranja. Prilikom spajanja dolazi do male pogreške jer moduli nisu apsolutno poravnani što ne predstavlja značajan problem jer u realnim izvedbama do te pogreške ne može doći. Konačno, priča o DTTO modularnom robotu zaokružena je procesom izrade jednog modula opisanom u *Poglavlju 7.*

Ovaj rad daje uvid u konstrukciju DTTO v2 modula i nudi neka od mogućih rješenja problema do kojih može doći u njegovoj eksploataciji. U međuvremenu je s razvojem krenula i DTTO v3, koja bi trebala donijeti značajna poboljšanja u odnosu na trenutnu verziju. Prvenstveno novi dizajn modula, izrađen u ABS tehnologiji s manjim i snažnijim aktuatorima, infracrveni senzori koji će omogućiti detekciju orijentacije modula pri spajanju, novi sustav za povezivanje i dodana rotacijska ploha koja omogućuje dodatan stupanj slobode. Neka od navedenih poboljšanja mogu se vidjeti na stranici projekta[37]. Konkretan razvoj projekta tek je započeo, a njegov puni potencijal tek se treba pokazati.

Literatura:

[1] Karel Čapek: *R.U.R. (Rossum's Universal Robots)*, Dover, New York, 2001.

[2] Hrvatska enciklopedija, http://www.enciklopedija.hr/natuknica.aspx?ID=53100 , 15.9.2017

[3] Isaac Asimov. *Runaround*, Street and Smith Publications, 1942.

[4] Mark Yim, Paul White, Michael Park, Jimmy Sastra. *Modular Self-Reconfigurable Robots, Encyclopedia of Complexity and Systems Science*, Larkspur, Springer, 2009.

[5] Kasper Stoy, David Brandt, David J. Christensen. *Self-Reconfigurable Robots An Introduction*, Cambridge, The MIT Press, 2010.

[6] Alberto Molina Pérez. *Diseño y Construcción de un Robot Modular Reconfigurable,* Tarragona, Universitat Rovira i Virgili, 2016.

[7] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita and S. Kokaji, "M-TRAN: self-reconfigurable modular robotic system," in *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 431-441, Dec. 2002.

[8] Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, Akiya Kamimura, Satoshi Murata, Shigeru Kokaji. *Self-reconfigurable M-TRAN structures and walker generation,* Elsevier, 2006.

[9] Hackaday, https://hackaday.com/2016/11/05/dtto-explorer-modular-robot-wins-2016-hackaday-prize/, 15.9.2017

[10] Hackaday, https://hackaday.io/project/9976-dtto-explorer-modular-robot, 15.9.2017

[11]AIST project MTRAN, https://unit.aist.go.jp/is/frrg/dsysd/mtran3/history.htm, 15.9.2017

[12] S.Murata, H. Kurokawa and S. Kokaji, "Self-assembling machine," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA, 1994, pp. 441-448 vol.1.

[13] M. Yim, D. G. Duff and K. D. Roufas, "PolyBot: a modular reconfigurable robot," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, San Francisco, CA, 2000, pp. 514-520 vol.1.

[14] Ultimaker: 3D Printers, https://ultimaker.com/en/resources/18767-unboxing, 15.9.2017

[15] Coppeliarobotics, http://www.coppeliarobotics.com/helpFiles/index.html, 15.9.2017

[16] Iida, Fumiya, and Auke Jan Ijspeert. "Biologically inspired robotics." *Springer Handbook of Robotics*. Springer International Publishing, 2016. 2015-2034.

[17] Pfeifer, Rolf, Max Lungarella, and Fumiya Iida. "Self-organization, embodiment, and biologically inspired robotics." *science* 318.5853 (2007): 1088-1093.

[18] Owen, T. (1994). Biologically Inspired Robots: Snake-Like Locomotors and Manipulators by Hirose ShigeoOxford University Press, Oxford, 1993

[19] Shiego Hirose, Hiroya Yamada. *Snake-Like Robots: Machine Design of Biologically Inspired Robots*, IEEE Robotics & Automation Magazine, 2009.

[20] Juan González Gómez (Obijuan) *Curva Serpentinoide*, http://nbviewer.jupyter.org/github/Obijuan/My-Ipython-notebooks/blob/master/serpenoid-curve/Serpenoid-Curve-es.ipynb, 15.9.2017

[21] San-Millán, Rodríguez, Andrés. *Diseño, construcción y control de una serpiente robótica*. Proyecto de Final de Carrera, Universidad de Castilla-La Mancha. Ciudad Real, 2011.

[22] Qniemiec - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10893168, 15.9.2017

[23] Coppeliarobotics, http://www.coppeliarobotics.com/helpFiles/en/shapeReferenceFrames.htm, 15.9.2017

[24] Coppeliarobotics, http://www.coppeliarobotics.com/helpFiles/en/eulerAngles.htm, 15.9.2017

[25] TowerPro, http://www.towerpro.com.tw/product/mg92b/, 15.9.2017

[26] DTTO modular robot rotating snake configuration, https://www.youtube.com/watch?v=Y83nx73XyuU, 15.9.2017

[27] Coppeliarobotics, www.coppeliarobotics.com/scenesAndModels/simpleMagnetDemo.ttt, 15.9.2017

[28] Hackaday, https://hackaday.io/project/9976/instructions, 15.9.2017

[29] Hackaday, https://cdn.hackaday.io/files/9976426813184/DttoModularRobotv2-3Dparts.zip, 15.9.2017

[30] Hackaday, https://cdn.hackaday.io/files/9976426813184/List%20of%203D%20printed%20Partsv4.pdf, 15.9.2017

[31] Towerpro, http://www.towerpro.com.tw/product/sg92r-7/, 15.9.2017

[32] Towerpro, http://www.towerpro.com.tw/product/mg90s-3/, 15.9.2017

[33] Texas Instruments. *LM317 3-Terminal Adjustable Regulator*, september 1997, revised september 2016.

[34] Arduino, https://store.arduino.cc/arduino-nano, 15.9.2017

[35] Hackaday, https://cdn.hackaday.io/files/9976426813184/Schematicsv1.pdf , 15.9.2017

[36] Github, https://github.com/otrebla333/Dtto-Modular-Robot/tree/master/Software, 15.9.2017

[37] Hackaday, https://hackaday.io/project/9976/logs, 15.9.2017

[38] Matlab Answers, https://www.mathworks.com/matlabcentral/answers/466-removing-white-space-in-emf-plots, 15.9.2017

Prilog 1.

Arduino kod za aktuaciju DTTO modularnog robota, zasluga Alberta Molina Péreza.

```
/*
 * Código módulos Dtto:  robot modular transformable
 * Alberto Molina Pérez
 * */


/////LIBRERIAS/////

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif
#include <SoftwareSerial.h>
#include <Servo.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

/////DEFINITIONS/////

#define DttoType 0//0 for MASTER module; 1 for the rest of SLAVE
modules
#define METALGEAR 1 //0 for NO, 1 for YES (If you are using the
MG92B,use 1. If using SG92R, use 0.

#define PIN 19 //Pin Neopixel

#define CE_PIN 2  //RF pin
#define CSN_PIN 4  //RF pin

int SCenterM =83;     //Center position of Male and Female main
motors
int SCenterF =83;
#define SCenterB 83   //Center, min and max position of coupling
mechanism
#define SMinB 5
#define SMaxB 150

/////CONTROL SINUSOIDAL (snake movement) /////


#define vel 0.002 //Speed (w)
//
int ampl = 40; //Amplitude
float desfase = 3.14; // Phase between modules(motores*2)
float desfaseint = 3.14;
unsigned long t = 0;    //Time auxiliar variables
unsigned long t2 = 0;
```

```
unsigned long t_start=0;
byte reset=1;



/////INICIALIZACIONES/////

Adafruit_NeoPixel strip = Adafruit_NeoPixel(1, PIN, NEO_GRB +
NEO_KHZ800);  //number of pix, arduino PIN, pixel type flags

SoftwareSerial bluetooth(7,8); //Creating software serial for
bluetooth module (RX,TX)

//NRF module configuration
const uint64_t pipes[2] = {0xE8E8F0F0E1LL,0xF0F0F0F0D2LL}; // Define
the transmit pipe rf
char RFData[6];
char addmodRF[2]={'a','+'};
RF24 radio(CE_PIN, CSN_PIN); // Create a Radio

/////VARIABLES PROGRAMA/////

//bt
char rxChar;    // Serial port receiving variable
int ledpin = 13;  // Pin donde se encuentra conectado el led (pin
13)

String rxDataBT; //Datos recibidos por BT - BT received data
char rxDataBT2[6]; //Datos recibidos por BT convertidos a char array
- BT data converted to char array
char rxDataRF[6]; //Data recibida por RF, valorar si es para ese
modulo - NRF received data
char rxData[6]; //Data internal para el modulo - Clean received data
char rxServo;
int angle;
int w0;
int w1;
int w2;
int i=0;
int j=0;

int modnum=1;
int modtotal=1;

//SERVO definitions (Calibrate according to real module for
accuracy)
Servo servoM;
int posM = 82;//MIN 001, MAX 172
Servo servoF;
int posF = 83;//MIN002, MAX 174
Servo servoB;
int posB = 82;
Servo servoR;
int posR= 80;
Servo servoL;
int posL = 80;

///// Rotating movement variables
```

```
#define vel2 0.001 //Speed (w)
byte stepc = 0;
byte inicio=0;
byte next = 0;

  unsigned long started_at = millis();
  bool timeout2 = true;
  int sync_ok=0;
  int sync_data[2];


void setup()
{

    if (METALGEAR) //Correcting for MG92B motors
    {
      SCenterM =90;
      SCenterF =90;
    }
  // put your setup code here, to run once:
    Serial.begin(9600);
    delay(3000);

    radio.begin();  //Starts NRF (radio)
    //radio.setRetries(15,5); //Intentos de reenvio en caso de mala
comunicacion

    if (DttoType==0)      //If module is a master: Start bluetooth
and writing channel NRF radio
    {
      bluetooth.begin(9600);
      bluetooth.println("Reset");
      radio.openWritingPipe(pipes[0]);
      radio.openReadingPipe(1,pipes[1]);
      //radio.openWritingPipie(pipes[2]);
      radio.startListening();
    }
    else                  //IF module is a slave: Start reading
channel NRF radio
    {
      radio.openWritingPipe(pipes[1]);
      radio.openReadingPipe(1,pipes[0]);
      //radio.openReadingPipe(2,pipes[2]);
    }

    if (DttoType==1) getmodulenum(); //If slave, ask master module
for a number
    //Start Neopixel LED
    strip.begin();
    strip.setBrightness(255);
    strip.setPixelColor(0,0,100,0);
    strip.show();




}
```

```
void loop()
{

  if(DttoType==0)
  {
    assignmodulenum();
    listenBluetooth();
  }
  else
  {
    listenRF();
  }
      //rxData[1]='r';
      //rxData[2]='m';
      switch (rxData[1])

                  {
                     case 's': //start, initial position
                         start();
                       break;
                     case 'r': //Run, sinusoidal control
                         snake();
                       break;
                     case 'e'://Escape
                         escape();
                       break;
                     case 'm'://manual movement
                         manual();
                       break;
                     case 'g'://attach module
                         gancho();
                       break;
                     case 't'://test communications
                         test_comm();
                       break;
                     case 'a'://modificar Amplitud
                         amplitud();
                       break;
                     case 'f'://modificar Fase
                         fase();
                       break;
                     case '+':
                         modtotal++;
                         rxData[1]='e';
                         Serial.println(modtodal);
                         break;
                     case 'h'://test communications
                         modtotal=1;
                         rxData[1]='e';
                       //break;
                     default:
                       break;
                   }
  }


void amplitud()
{
```

```
      w0=(rxData[2]-'0')*100;
      w1=(rxData[3]-'0')*10;
      w2=rxData[4]-'0';
      ampl=w0+w1+w2;
      rxData[1]='e';
}

void fase()
{
      w0=(rxData[2]-'0')*100;
      w1=(rxData[3]-'0')*10;
      w2=rxData[4]-'0';
      desfaseint=w0+w1+w2;
      desfaseint=(desfaseint*3.1415)/180;
      desfase=desfaseint;
      rxData[1]='e';
}

void getmodulenum()
{
  int peticion=1234;
  detachServos();
  while (modnum==1)
  {
      radio.stopListening();
      radio.write(&peticion,sizeof(int));
      radio.startListening();
      unsigned long started_waiting_at = millis();
      bool timeout = false;
      while ( ! radio.available() && ! timeout )       // Esperamos
200ms
      {
          if (millis() - started_waiting_at > 500 )timeout = true;
      }
      if ( ! timeout )
      {   // Leemos el mensaje recibido
           int got_num;
           radio.read(&got_num, sizeof(int) );
           modnum=got_num;
           modtotal=modnum-1;
      }
  }
}


void assignmodulenum()
{

      //detachServos();
      if ( radio.available())
      {   // Leemos el mensaje recibido
           int got_peticion;
           int addmod;
           radio.read(&got_peticion, sizeof(int) );
           if (got_peticion==1234)
            {
              radio.stopListening();
              modtotal++;
              radio.write(&modtotal,sizeof(int));
```

```
              radio.write(&addmodRF,6);
              radio.startListening();

            }
      }
}

void listenBluetooth()
{

  if( bluetooth.available()>1 ) //Aqui hay num de bytes disponibles
up to 64 bytes
  {
    //  Detach servos para que no crear interferencias con
softwareSerial.
      //radio.stopListening();
      detachServos();
      rxDataBT= bluetooth.readStringUntil('\0');
      //radio.startListening();
      rxDataBT.toCharArray( rxDataBT2,6 );
       //Check DATA syntax??
      if(rxDataBT2[0]=='1')
        {
            for(i=0;i<6;i++)
            {
              rxData[i]=rxDataBT2[i];
            }

        }
      else if (rxDataBT2[0]=='a')
        {
            for(i=0;i<6;i++)
            {
              RFData[i]=rxDataBT2[i];
              rxData[i]=rxDataBT2[i];
            }
            radio.stopListening();
            radio.write( RFData, 6 );
            radio.startListening();
        }
      else
        {
          for(i=0;i<6;i++)
            {
              RFData[i]=rxDataBT2[i];
            }
            radio.stopListening();
            radio.write( RFData, 6 );
            radio.startListening();
        }
  }
  //
}


void syncMaster()
{
            sync_data[0]='S';
            delay(1);
```

```
            radio.stopListening();
            radio.write( &sync_data, sizeof(int) );
            radio.startListening();

}

void listenRF()
{
  if ( radio.available() )
  {
    // Read the data payload until we've received everything
   detachServos();
   while (radio.available())
   {
      // Fetch the data payload
      radio.read( rxDataRF, 6 );
    }

   if((rxDataRF[0]== (modnum+48))||(rxDataRF[0]=='a') )
   {
    for(i=0;i<6;i++)
            {
              rxData[i]=rxDataRF[i];
            }
   }
   //attachServos();
  }
}

void gancho()
{
  int rxData2OLD;
  int rxData3OLD;

  if ((rxData[2]!=rxData2OLD)||(rxData[3]!=rxData3OLD))
  {
    switch(rxData[2])
    {
      case 'b':
        servoB.attach(10);
        if (rxData[3]=='0')posB=SCenterB;
        else if (rxData[3]=='1')posB=SMinB;
        else if (rxData[3]=='2')posB=SMaxB;
        servoB.write(posB);
        break;
      case 'r':
        servoR.attach(3);
        if (rxData[3]=='0')posR=SCenterB;
        else if (rxData[3]=='1')posR=SMinB;
        else if (rxData[3]=='2')posR=SMaxB;
        servoR.write(posR);
        break;
      case 'l':
        servoL.attach(9);
        if (rxData[3]=='0')posL=SCenterB;
        else if (rxData[3]=='1')posL=SMinB;
        else if (rxData[3]=='2')posL=SMaxB;
        servoL.write(posL);
        break;
```

```
      default:
        break;
    }
    rxData2OLD=rxData[2];
    rxData3OLD=rxData[3];
    delay(150);

  }
  detachServos();
}

void start()

{
  attachServos();
  posM = SCenterM;
  posF = SCenterF;
  posB = SCenterB;
  posR = SCenterB;
  posL = SCenterB;
  writeServos();
  delay(250);
  detachServos();

}

void snake()
{
  attachServos();
  if(reset) t_start=millis();
  t=millis()- t_start;
  reset=0;
  //Serial.println(posM);
  //Serial.println(posF);
  switch(rxData[2])
  {
    case 'm':
      posM=SCenterM+ampl*sin((vel*t)+(modnum*desfase));

posF=SCenterF+ampl*sin((vel*t)+((modnum*desfase)+((desfase)/2)));
      servoM.write(posM);
      servoF.write(posF);
      break;
    case 'f':
      posM=SCenterM+ampl*sin((vel*t)-(modnum*desfase));
      posF=SCenterF+ampl*sin((vel*t)-
((modnum*desfase)+((desfase)/2)));
      servoM.write(posM);
      servoF.write(posF);
      break;
    case 'c':
        switch(modtotal)
            {
            case 3:
              if (stepc == 0)
              {
                if (modnum==1)
                {
                  servoB.write(5);
```

```
    delay(1000);
    stepc=1;
    delay(1000);
    servoM.write(0);
    servoF.write(178);
    delay(1000);
  }
  else if (modnum==2)
  {
    servoB.write(5);
    delay(1000);
    stepc=3;
    delay(400);
    servoM.write(0);
    servoF.write(85);
    delay(600);
    delay(1000);
  }
  else if (modnum==3)
  {
    delay(1000);
    stepc=2;
    servoM.write(85);
    servoF.write(178);
    delay(1000);
    delay(1000);
    servoB.write(5);
  }
  delay(1000);
  t_start=millis();
  next=0;

}
if(next)
{
  stepc++;
  if (stepc ==4) stepc =1;
  delay(500);
  t_start=millis();
  next=0;
}
  t=millis()- t_start;

switch(stepc)
{
  case 1:
    posM=85*sin((vel2*t));
    if (posM>83) next++;
    servoM.write(posM);
    servoF.write(178);
    break;
  case 2:
    posM=85-85*sin((vel2*t));
    if(posM<1)posM=0;
    posF=178-93*sin((vel2*t));
    if(posF<87)next++;
    servoM.write(posM);
    servoF.write(posF);
```

```
      break;
    case 3:
      posF=85+93*sin((vel2*t));
      if(posF>176)next++;
      servoM.write(0);
      servoF.write(posF);

      break;
    default:
    break;
  }
  break;
break;
case 4:
  if (stepc == 0)
  {
    if (modnum==1)
    {
    stepc=1;
    servoB.write(5);
    delay(2500);
    servoF.write(178);
    servoM.write(2);
    delay(1000);
    }
    else if (modnum==2)
    {
      stepc=3;
      servoB.write(5);
      delay(3500);
    }
    else if (modnum==3)
    {
      stepc=1;
      servoB.write(5);
      delay(500);
      servoM.writeMicroseconds(700);
      delay(1000);
      servoF.writeMicroseconds(2288);
      delay(2000);
    }
    else if (modnum==4)
    {
      stepc=3;
      servoF.write(178);
      delay(1500);
      servoF.write(85);
      delay(2000);
      servoB.write(5);

    }
    delay(1000);
    t_start=millis();
    next=0;

  }
  if(next)
  {
    stepc++;
```

```
                if (stepc ==5) stepc =1;
                delay(400);
                t_start=millis();
                next=0;
              }
                t=millis()- t_start;

              switch(stepc)
              {
                case 1:
                  posM=85*sin((vel2*t));
                  if (posM>83) next++;
                  servoM.write(posM);
                  servoF.write(178);
                  break;
                case 2:
                  posF=178-93*sin((vel2*t));
                  if(posF<87)next++;
                  servoM.write(86);
                  servoF.write(posF);

                  break;
                case 3:
                  posM=85-85*sin((vel2*t));
                  if(posM<1)next++;
                  servoM.write(posM);
                  servoF.write(87);
                  break;
                case 4:
                  posF=85+93*sin((vel2*t));
                  if(posF>176)next++;
                  servoM.write(1);
                  servoF.write(posF);
                  break;
                default:
                break;
              }
              break;
            break;
             }
    default:
      break;
  }
  //detachServos();
}

void manual()
{
    //Lectura angulo introducido
      w0=(rxData[3]-'0')*100;
      w1=(rxData[4]-'0')*10;
      w2=rxData[5]-'0';
      angle=w0+w1+w2;
      if ((angle<=180)&&(angle>=0))
      {
           switch (rxData[2])
           {
              case 'm':
                  servoM.write(angle);
```

```
                break;
              case 'f':
                  servoF.write(angle);
                break;
              default:

                break;
            }
      }
}
void escape()
{
  t=0;
  detachServos();
  strip.setPixelColor(0,0,100,0);
  strip.show();
  reset=0;
  stepc=0;
  next=0;
  inicio=0;
}

void attachServos()
{
  servoM.attach(6);
  servoF.attach(5);
  servoB.attach(10);
  servoR.attach(3);
  servoL.attach(9);
}

void writeServos()
{
  servoM.write(posM);
  servoF.write(posF);
  servoB.write(posB);
  servoR.write(posR);
  servoL.write(posL);
}

void detachServos()
{
  servoM.detach();
  servoF.detach();
  servoB.detach();
  servoR.detach();
  servoL.detach();
}

void test_comm()
{

      if((millis()- started_at)>500) strip.setPixelColor(0,0,0,100);
      else strip.setPixelColor(0,100,0,0);
      if((millis()- started_at)>1000) started_at=millis();
      strip.show();

}
```

Prilog 2.

V-REP kod korišten za testiranje aktuacije i rotacije modula.

Napomena: dio koda za simuliranje magneta preuzet je iz [27].

```
if (sim_call_type==sim.childscriptcall_initialization) then

    dtto=sim.getObjectHandle("male_dyn")
    male_joint=sim.getObjectHandle("male_joint")
    female_joint=sim.getObjectHandle("female_joint")
    rotation_joint_male=sim.getObjectHandle("rotation_joint_male")

rotation_joint_female=sim.getObjectHandle("rotation_joint_female")
    female_segment=sim.getObjectHandle("female")
    male_segment=sim.getObjectHandle("male")
    link=sim.getObjectHandle("link_dyn")
    magnet_male=sim.getObjectHandle("magnet_male")
    magnet_female=sim.getObjectHandle("magnet_female")
    sensor_male=sim.getObjectHandle('Proximity_sensor_male')
    sensor_female=sim.getObjectHandle('Proximity_sensor_female')
    graphHandle=sim.getObjectHandle("pogreska_i_control")

    --Parametri aktuacije:
        t=0
        situation=1           --situation=1 rotacija pod izracunatim
kutem, situation=2 rotacija pod 90
        nominalForce=0.3

        w_m=2
        w_f=2
        amp_m=45
        amp_f=45
```

```
        phs=180
        phs_s=60
        a_m=amp_m*(math.pi/180)
        a_f=amp_f*(math.pi/180)
        p=phs*(math.pi/180)
        p_s=phs_s*(math.pi/180)
        pr=0
            condition=0
            rot_end=0
            pos=0
            det=0

        target_x=1
        target_y=1
        delta_x=0
        delta_y=0

    --Parametri regulatora:
        Kp=1
        Kp_rot=1
        Ki=0
        int=0
        dt=0.05

end


if (sim_call_type==sim.childscriptcall_actuation) then

    t=t+sim.getSimulationTimeStep()

    --Pozicije aktuatora:
    male_joint_position=sim.getJointPosition(male_joint)
```

```
    mj_pos=male_joint_position*(180/math.pi)
    female_joint_position=sim.getJointPosition(female_joint)
    fj_pos=female_joint_position*(180/math.pi)
    rotation_position_male=sim.getJointPosition(rotation_joint_male)
    rm_pos=rotation_position_male*(180/math.pi)

rotation_position_female=sim.getJointPosition(rotation_joint_female)
    rf_pos=rotation_position_female*(180/math.pi)
    --Pozicije segmenta:
    position_dtto=sim.getObjectPosition(dtto,-1)
    position_female=sim.getObjectPosition(female_segment,-1)
        x_male=position_dtto[1]
        y_male=position_dtto[2]
        x_female=position_female[1]
        y_female=position_female[2]
    --Orijentacije segmenta:
    orientation_dtto=sim.getObjectOrientation(dtto,-1)
    orientation_male=sim.getObjectOrientation(male_segment,-1)
    orientation_female=sim.getObjectOrientation(female_segment,-1)
    orientation_link=sim.getObjectOrientation(link,-1)
        alpha_dtto=orientation_dtto[1]
        a_dtto=orientation_dtto[1]*(180/math.pi)         --alpha dtto
u stupnjevima
        beta_dtto=orientation_dtto[2]
        b_dtto=orientation_dtto[2]*(180/math.pi)          --beta dtto
u stupnjevima


        alpha_male=orientation_male[1]
        a_male=orientation_male[1]*(180/math.pi)
        beta_male=orientation_male[2]
        b_male=orientation_male[2]*(180/math.pi)
        gamma_male=orientation_male[3]
        g_male=orientation_male[3]*(180/math.pi)
```

```
        alpha_female=orientation_female[1]
        a_female=orientation_female[1]*(180/math.pi)
        beta_female=orientation_female[2]
        b_female=orientation_female[2]*(180/math.pi)

        alpha_link=orientation_link[1]
        a_link=orientation_link[1]*(180/math.pi)
        beta_link=orientation_link[2]
        b_link=orientation_link[2]*(180/math.pi)
        gamma_link=orientation_link[3]
        g_link=orientation_link[3]*(180/math.pi)

    --PI X:
        e_x=(target_x-x_male)/math.abs(target_x-x_male)
        int_x=int+e_x*dt
        c_x=Kp*e_x+Ki*int_x
    --PI Y:
        e_y=(target_y-y_male)/math.abs(target_y-y_male)
        int_y=int+e_y*dt
        c_y=Kp*e_y+Ki*int_y

    --Senzori i magneti:
        r_m,dist_m,pt_m,obj_m=sim.handleProximitySensor(sensor_male)
r_f,dist_f,pt_f,obj_f=sim.handleProximitySensor(sensor_female)

        if r_m>0 then
            dist_m=dist_m+0.02 -- add some offset, otherwise we have
an infinite force when objects touch
            force=nominalForce/(dist_m*dist_m)
            -- Get tranformation matrix, without translational
components:
            m_m=sim.getObjectMatrix(magnet_male,-1)
            m_m[4]=0
```

```
            m_m[8]=0
            m_m[12]=0
            -- Multiply, so we have the force vector in absolue
coordinates:
                forceVector=sim.multiplyVector(m_m,{0,0,force})
            -- Apply the force the the center of mass for exactly
one time step:
            sim.addForceAndTorque(magnet_male,forceVector,{0,0,0})
        end

        if r_f>0 then
            dist_f=dist_f+0.02 -- add some offset, otherwise we have
an infinite force when objects touch
            force=nominalForce/(dist_f*dist_f)
            -- Get tranformation matrix, without translational
components:
            m_f=sim.getObjectMatrix(magnet_female,-1)
            m_f[4]=0
            m_f[8]=0
            m_f[12]=0
            -- Multiply, so we have the force vector in absolue
coordinates:
                forceVector=sim.multiplyVector(m_f,{0,0,force})
            -- Apply the force the the center of mass for exactly
one time step:
            sim.addForceAndTorque(magnet_female,forceVector,{0,0,0})
        end

if situation==1 then

    epsilon=math.acos(delta_x/math.sqrt((delta_x^2)+(delta_y^2)))
    eps=epsilon*(180/math.pi)

    if x_male<0 then
```

```
        if target_x>0 then
            delta_x=math.abs(x_male)+target_x
        else
            delta_x=math.abs(target_x)+x_male
        end
    else
        if target_x>0 then
            delta_x=target_x-x_male
        else
            delta_x=x_male+math.abs(target_x)
        end
    end

    if y_male<0 then
        if target_y>0 then
            delta_y=math.abs(y_male)+target_y
        else
            delta_y=math.abs(target_x)+y_male
        end
    else
        if target_y>0 then
            delta_y=target_y-y_male
        else
            delta_y=y_male+math.abs(target_y)
        end
    end




    if  mj_pos<91 and mj_pos>89 then
        sim.setJointTargetPosition(female_joint,0)
--upravljanje modula
        sim.setJointTargetPosition(male_joint,0)
```

```
    end


    --PI Fi:
    if eps-math.abs(b_dtto)<0.5 then
        e_fi=0
    else
        e_fi=(eps-math.abs(b_dtto))/(eps-math.abs(b_dtto))
    end
    c_fi=Kp_rot*e_fi



result_m,parameter_rot_male=sim.getObjectInt32Parameter(rotation_joi
nt_male,2001)

result_sensor_m,parameter_sensor_male=sim.getObjectInt32Parameter(se
nsor_male,4002)


            if       target_x>0       and       target_y>0       then
--I. kvadrant

sim.setJointTargetPosition(rotation_joint_male,0.5*(math.pi/180))
                if rm_pos>0 then
                    pr=pr+(c_fi*0.5*(math.pi/180))

sim.setJointTargetPosition(rotation_joint_male,pr)
                    if e_fi==0 then

sim.setJointTargetPosition(male_joint,90*(math.pi/180))
                    end
                end
            end


            if       target_x<=0       and       target_y>0       then
--II. kvadrant

sim.setJointTargetPosition(rotation_joint_male,99*(math.pi/180))
```

```
                if rm_pos>98 then
                    pr=pr+(c_fi*0.5*(math.pi/180))

sim.setJointTargetPosition(rotation_joint_male,pr)
                    if e_fi==0 then

sim.setJointTargetPosition(male_joint,90*(math.pi/180))
                    end
                end
            end


            if      target_x<=0      and      target_y<=0      then
--III. kvadrant
                sim.setJointTargetPosition(rotation_joint_male,-
91*(math.pi/180))
                if rm_pos<=-90.1 then
                    pr=pr-(c_fi*0.5*(math.pi/180))

sim.setJointTargetPosition(rotation_joint_male,pr)
                    if e_fi==0 then

sim.setJointTargetPosition(male_joint,90*(math.pi/180))
                    end
                end
            end


            if       target_x>0       and       target_y<=0       then
--IV. kvadrant
                sim.setJointTargetPosition(rotation_joint_male,-
1*(math.pi/180))
                 if rm_pos<=-1 then
                     pr=pr-(c_fi*0.5*(math.pi/180))

sim.setJointTargetPosition(rotation_joint_male,pr)
                    if e_fi==0 then
```

```
sim.setJointTargetPosition(male_joint,90*(math.pi/180))
                    end
                end
            end

            if    math.abs(a_link)>88  and  math.abs(a_link)<92  and
math.abs(g_link)>88        and        math.abs(g_link)<91        and
math.abs(math.abs(b_dtto)-eps)<1 then
                det=1

sim.setObjectInt32Parameter(sensor_male,4002,female_segment)

sim.setObjectInt32Parameter(sensor_female,4002,male_segment)
            end

            if det==1 and parameter_sensor_male==28 then
                sim.setJointTargetPosition(male_joint,0)
                if mj_pos>-1 and mj_pos<1 then
                    rot_end=2
                end
            end

            if rot_end==2 then
                if   math.abs(y_male)>=math.abs(target_y)-0.03   and
math.abs(x_male)>=math.abs(target_x)-0.03 then
                    pm=0
                    pf=0
                    sim.setJointTargetPosition(male_joint,pm)
                    sim.setJointTargetPosition(female_joint,pf)
                else
                    pm=(a_m*math.sin((w_m*t)+p_s))
                    pf=(a_f*math.sin((w_f*t)))

                    sim.setJointTargetPosition(male_joint,pm)
```

```
                    sim.setJointTargetPosition(female_joint,pf)
                end
            end


end

if situation==2 then
    --PI Fi
    eps=90

    if eps-math.abs(b_dtto)<0.5 then
        e_fi=0
    else
        e_fi=(eps-math.abs(b_dtto))/(eps-math.abs(b_dtto))
    end
    c_fi=Kp_rot*e_fi



result_m,parameter_rot_male=sim.getObjectInt32Parameter(rotation_joi
nt_male,2001)

result_sensor_m,parameter_sensor_male=sim.getObjectInt32Parameter(se
nsor_male,4002)

    --Aktuacija po sinus nacelu:

    if math.abs(x_male)>=math.abs(target_x)-0.12 and det==0 then
        sim.setObjectInt32Parameter(sensor_male,4002,-1)
        sim.setObjectInt32Parameter(sensor_female,4002,-1)
    else
        sim.setObjectInt32Parameter(sensor_male,4002,female_segment)
        sim.setObjectInt32Parameter(sensor_female,4002,male_segment)
```

```
    end
    --Rotacija i uspravljanje za gibanje u x smjeru:
    if  math.abs(x_male)>=(math.abs(target_x)-0.12) then
        sim.setJointTargetPosition(male_joint,90*(math.pi/180))
        sim.setJointTargetPosition(female_joint,0)
    else
        pm=((a_m)*math.sin(((w_m)*t)+p+p_s))
        pf=((a_f)*math.sin(((w_f)*t)+p))

        sim.setJointTargetPosition(male_joint,pm)
        sim.setJointTargetPosition(female_joint,pf)
    end

    if      math.abs(a_male)>179   and   math.abs(a_male)<181   and
math.abs(b_male)<10 then
        sim.setJointTargetPosition(male_joint,0)                  --
uspravljanje modula
    end

    if    math.abs(a_link)>179    and    math.abs(a_link)<181    and
math.abs(b_link)<1 then                  -- ako je modul u uspravnom
polozaju
        pr=pr+(c_fi*0.5*(math.pi/180))
        sim.setJointTargetPosition(rotation_joint_male,pr)
        sim.setJointForce(rotation_joint_male,1.550e+03)
        if e_fi==0 then
            sim.setJointTargetPosition(male_joint,90*(math.pi/180))
        end
    end

    if       math.abs(a_link)>88    and    math.abs(a_link)<92    and
math.abs(g_link)>88        and        math.abs(g_link)<91        and
math.abs(math.abs(b_dtto)-eps)<1 then
            det=1
        sim.setObjectInt32Parameter(sensor_male,4002,female_segment)
```

```
        sim.setObjectInt32Parameter(sensor_female,4002,male_segment)
    end

    if det==1 and parameter_sensor_male==28 then
        sim.setJointTargetPosition(male_joint,0)
        if mj_pos>-1 and mj_pos<1 then
                    rot_end=2
        end
    end

    if rot_end==2 then
        if math.abs(y_male)>=math.abs(target_y)-0.03 then
            pm=0
            pf=0
            sim.setJointTargetPosition(male_joint,pm)
            sim.setJointTargetPosition(female_joint,pf)
        else
            pm=(a_m*math.sin((c_y*w_m*t)+p_s))
            pf=(a_f*math.sin((c_y*w_f*t)))

            sim.setJointTargetPosition(male_joint,pm)
            sim.setJointTargetPosition(female_joint,pf)
        end
    end

end
    sim.addStatusbarMessage('beta='..b_dtto)
    sim.addStatusbarMessage('pr='..rm_pos)
    sim.addStatusbarMessage('epsilon='..eps)
    sim.addStatusbarMessage('x='..x_male)
    sim.addStatusbarMessage('male joint position='..mj_pos)
end
```

Prilog 3.

V-REP kod korišten kod spajanja modula.

Napomena: dio koda za simuliranje magneta preuzet je iz [27].

```
if (sim_call_type==sim.childscriptcall_initialization) then

    dtto=sim.getObjectHandle("male_dyn")
    male_joint=sim.getObjectHandle("male_joint")
    female_joint=sim.getObjectHandle("female_joint")
    rotation_joint_male=sim.getObjectHandle("rotation_joint_male")

rotation_joint_female=sim.getObjectHandle("rotation_joint_female")
    female_segment=sim.getObjectHandle("female")
    male_segment=sim.getObjectHandle("male")
    link=sim.getObjectHandle("link_dyn")
    magnet_male=sim.getObjectHandle("magnet_male")
    magnet_female=sim.getObjectHandle("magnet_female")
    sensor_male=sim.getObjectHandle('Proximity_sensor_male')
    sensor_female=sim.getObjectHandle('Proximity_sensor_female')
    graphHandle=sim.getObjectHandle("pogreska_i_control")

    --Parametri aktuacije:
        t=0
        nominalForce=0.15

        w_m=2
        w_f=2
        amp_m=45
        amp_f=45
        phs=180
        phs_s=60
```

```
        a_m=amp_m*(math.pi/180)
        a_f=amp_f*(math.pi/180)
        p=phs*(math.pi/180)
        p_s=phs_s*(math.pi/180)
        pr=0
            condition=0
            rot_end=0
            pos=0
            det=0

        target_x=1
        target_y=1
        delta_x=0
        delta_y=0

    --Parametri regulatora:
        Kp=1
        Kp_rot=1
        Ki=0
        int=0
        dt=0.05

end


if (sim_call_type==sim.childscriptcall_actuation) then

    t=t+sim.getSimulationTimeStep()

    --Pozicije aktuatora:
    male_joint_position=sim.getJointPosition(male_joint)
    mj_pos=male_joint_position*(180/math.pi)
    female_joint_position=sim.getJointPosition(female_joint)
```

```
    fj_pos=female_joint_position*(180/math.pi)
    rotation_position_male=sim.getJointPosition(rotation_joint_male)
    rm_pos=rotation_position_male*(180/math.pi)

rotation_position_female=sim.getJointPosition(rotation_joint_female)
    rf_pos=rotation_position_female*(180/math.pi)
    --Pozicije segmenta:
    position_dtto=sim.getObjectPosition(dtto,-1)
    position_female=sim.getObjectPosition(female_segment,-1)
        x_male=position_dtto[1]
        y_male=position_dtto[2]
        x_female=position_female[1]
        y_female=position_female[2]
    --Orijentacije segmenta:
    orientation_dtto=sim.getObjectOrientation(dtto,-1)
    orientation_male=sim.getObjectOrientation(male_segment,-1)
    orientation_female=sim.getObjectOrientation(female_segment,-1)
    orientation_link=sim.getObjectOrientation(link,-1)
        alpha_dtto=orientation_dtto[1]
        a_dtto=orientation_dtto[1]*(180/math.pi)         --alpha dtto
u stupnjevima
        beta_dtto=orientation_dtto[2]
        b_dtto=orientation_dtto[2]*(180/math.pi)          --beta dtto
u stupnjevima


        alpha_male=orientation_male[1]
        a_male=orientation_male[1]*(180/math.pi)
        beta_male=orientation_male[2]
        b_male=orientation_male[2]*(180/math.pi)
        gamma_male=orientation_male[3]
        g_male=orientation_male[3]*(180/math.pi)


        alpha_female=orientation_female[1]
        a_female=orientation_female[1]*(180/math.pi)
```

```
        beta_female=orientation_female[2]
        b_female=orientation_female[2]*(180/math.pi)

        alpha_link=orientation_link[1]
        a_link=orientation_link[1]*(180/math.pi)
        beta_link=orientation_link[2]
        b_link=orientation_link[2]*(180/math.pi)
        gamma_link=orientation_link[3]
        g_link=orientation_link[3]*(180/math.pi)

    --PI X:
        e_x=(target_x-x_male)/math.abs(target_x-x_male)
        int_x=int+e_x*dt
        c_x=Kp*e_x+Ki*int_x
    --PI Y:
        e_y=(target_y-y_male)/math.abs(target_y-y_male)
        int_y=int+e_y*dt
        c_y=Kp*e_y+Ki*int_y

    --Senzori i magneti:
        r_m,dist_m,pt_m,obj_m=sim.handleProximitySensor(sensor_male)
r_f,dist_f,pt_f,obj_f=sim.handleProximitySensor(sensor_female)

        if r_m>0 then
            dist_m=dist_m+0.02 -- add some offset, otherwise we have
an infinite force when objects touch
            force=nominalForce/(dist_m*dist_m)
            -- Get tranformation matrix, without translational
components:
            m_m=sim.getObjectMatrix(magnet_male,-1)
            m_m[4]=0
            m_m[8]=0
            m_m[12]=0
```

```
            -- Multiply, so we have the force vector in absolue
coordinates:
                forceVector=sim.multiplyVector(m_m,{0,0,force})
            -- Apply the force the the center of mass for exactly
one time step:
            sim.addForceAndTorque(magnet_male,forceVector,{0,0,0})
        end


        if r_f>0 then
            dist_f=dist_f+0.02 -- add some offset, otherwise we have
an infinite force when objects touch
            force=nominalForce/(dist_f*dist_f)
            -- Get tranformation matrix, without translational
components:
            m_f=sim.getObjectMatrix(magnet_female,-1)
            m_f[4]=0
            m_f[8]=0
            m_f[12]=0
            -- Multiply, so we have the force vector in absolue
coordinates:
                forceVector=sim.multiplyVector(m_f,{0,0,force})
            -- Apply the force the the center of mass for exactly
one time step:
            sim.addForceAndTorque(magnet_female,forceVector,{0,0,0})
        end


    --PI X:
        if dist_m<0.5 then
            e_x=dist_m
            if dist_m<0.1 then
                e_x=0
            end
        else
            e_x=dist_m/dist_m
        end
```

```
            if math.abs(int_x)>1 then
                int_x=0
            else
                int_x=int_x+e_x*dt
            end
        c_x=Kp*e_x+Ki*int_x
   --Aktuacija po sinus nacelu:
    if r_m>0 then
        pm=(c_x*a_m*math.sin((w_m*t)+p_s))
        pf=(c_x*a_f*math.sin((w_f*t)))

        sim.setJointTargetPosition(male_joint,pm)
        sim.setJointTargetPosition(female_joint,pf)
    end

    sim.addStatusbarMessage('udaljenost='..dist_m)
    sim.addStatusbarMessage('x='..x_male)

end
```

Prilog 4.

Matlab kod za plotanje grafova.

Napomena: dio koda za prilagodbu veličine grafa preuzet je iz [38].

```
figure_graph= figure('Position',[1 1 400 300],'Color',[1 1
1]);

    plot(var1,var2,'b')
        xlabel('var1 []')
        ylabel('var2 []')
        title('Naslov')
        legend('var1')
    grid on
        AxesHandle=findobj(figure_graph,'Type','axes');
        set(AxesHandle,'OuterPosition',[0,0,1,1]);
    clearvars
```

Popis slika

Popis tablica

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam diplomski rad izradio samostalno, isključivo znanjem stečenim na Odjelu za elektrotehniku i računarstvo, služeći se navedenim izvorima podataka i uz stručno vodstvo mentorice doc.dr.sc. Ivane Palunko, kojoj se još jednom srdačno zahvaljujem.

Vicko Prkačin
