Matematičko modeliranje kinematike i sinteza upravljanja mobilnim robotom s četiri zakretna i četiri neovisno pogonjena kotača

Škifić, Niko

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Niko Škifić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Vladimir Milić, mag. ing.

Student:

Niko Škifić

Zagreb, 2024.

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

Zahvaljujem se prije svega roditeljima radi moralne i financijske potpore, svom mentoru **izv. prof. dr. sc. Vladimiru Miliću** što mi je omogućio da napišem ovaj rad, te kolegi Braimiru Ćaranu koji mi je omogućio pristup robotu i pomogao pri izradi rada. Također bih se volio zahvaliti djevojci, svim prijateljima i kolegama na podršci tokom studija.

Niko Škifić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

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

DIPLOMSKI ZADATAK

Student: **Niko Škifić** JMBAG: 1191240473

Naslov rada na hrvatskom jeziku: **Matematičko modeliranje kinematike i sinteza upravljanja mobilnim robotom s četiri zakretna i četiri neovisno pogonjena kotača**

Naslov rada na engleskom jeziku: **Mathematical modelling of kinematics and control synthesis of a mobile robot with four steering and four independently driven wheels**

Opis zadatka:

Razvijene su mnoge strukture mobilnih robota s različitim vrstama pogona kao što su: diferencijalni pogon, višesmjerni pogon, pogon s neovisnim upravljanjem na dva ili više kotača, itd. Posljednjih godina zbog visoke sposobnosti manevriranja električnih motora, mobilni roboti s četiri zakretna i neovisno pogonjena kotača predstavljaju jednu od najistraživanijih struktura s primjenama u područjima transporta, poljoprivrede te mnogim granama industrije.
U diplomskom radu je potrebno:

1. Izvesti nelinearne diferencijalne jednadžbe koje opisuju kinematiku mobilnog robota s četiri pogonska motora s neovisnim upravljanjima zakretanjem kotača. Provesti transformaciju kinematičkog model robota u lančanu formu i pri tome uvesti odgovarajuće pretpostavke u svrhu dobivanja jednostavnijeg modela.
2. Lančanu formu linearizirati za male perturbacije oko referentne trajektorije u obliku vremenski promjenljivog prostora stanja prikladnom za sintezu linearnog vremenski promjenljivog zakona upravljanja prema kvadratnom kriteriju optimalnosti. Za sintezu zakona upravljanja koristiti odgovarajuće ugrađene funkcije u nekom standardnom matematičkom programskom alatu (npr. u MATLAB-u).
3. Načiniti simulacijske modele u odgovarajućem matematičkom programskom alatu (npr. u MATLAB-u) u kojima će se direktno numerički rješavati diferencijalne jednadžbe primjenom ugrađenih funkcija. Simulacijama na računalu analizirati ponašanje zatvorenog sustava upravljanja u slučajevima nelinearnog i lineariziranog modela robota za slučaj nekoliko različitih referentnih trajektorija.
4. Opisati glavne komponente mobilnog robota s četiri pogonska motora s neovisnim upravljanjima zakretanjem kotača izrađenim u Laboratoriju za računalnu inteligenciju (CRTA). Na ovom mobilnom robotu implementirati dobiveni zakon upravljanja te provesti eksperimentalna mjerenja primjenom vanjskog mjernog sustava OptiTrack.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
16. studenoga 2023.	18. siječnja 2024.	22. – 26 . siječnja 2024.

Zadatak zadao: Predsjednik Povjerenstva:

Izv. prof. dr. sc. Vladimir Milić Prof. dr. sc. Ivica Garašić

Sadržaj

Popis slika

Popis simbola

A	amplituda sinusoide	m
a	udaljenost kotača od središta robota u smjeru x robotovog koordinatnog sustava	m
b	udaljenost kotača od središta robota u smjeru y robotovog koordinatnog sustava	m
R	radijus kružnice	m
r	radijus kotača	m
s	koeficijent Gaussove krivulje	-
t	vrijeme	s
u_i	ulazna veličina u ulačanoj formi	-
u_{di}	željena ulazna veličina u ulačanoj formi	-
$\tilde{u}_i$	greška ulazne veličine u ulačanoj formi	-
v	brzina robota	m/s
v_i	brzina kotača i	m/s
$v_x,\ v_y$	brzina robota u smjerovima x i y	m/s
$v_{xi},\ v_{yi}$	brzina kotača i u smjerovima x i y	m/s
$x,\ y$	položaj robota u globalnom koordinatnom sustavu	m
x_c	položaj amplitude Gaussove krivulje	m
x_i	stanje u ulačanoj formi	-
x_{di}	željeno stanje u ulačanoj formi	-
$\tilde{x}_i$	greška pozicije u ulačanoj formi	-
$x_{wi},\ y_{wi}$	položaj kotača i u robotovom koordinatnom sustavu	m

Y	amplituda Gaussove krivulje	m

Konstante

π	pi	3.14

Grčka slova

δ_i	kut zakreta kotača i	rad
ω	brzina zakreta robota	rad/s
ω_i	brzina zakreta kotača i	rad/s
ϕ_i	kut rotacije kotača i	rad/s
θ	kut zakreta robota	rad

Matrice

$\mathbf{A}$(t)	vremenski ovisna matrica koeficijenata sustava
$\mathbf{B}$(t)	vremenski ovisna matrica ulaza sustava
$\mathbf{g}_i$	vektor koeficijenata upravljačkih veličina
$\mathbf{I}$	jedinična matrica
$\mathbf{K}$(t)	vremenski ovisna matrica pojačanja
$\mathbf{P}$	matrica položaja
$\mathbf{P}^+$	pseudoinverz matrice položaja
$\mathbf{Q}$	penalty matrica performansi
$\mathbf{q}$	vektor stanja robota
$\mathbf{R}$	penalty matrica aktuacije
$\mathbf{u}$	vektor ulaznih veličina u ulančanoj formi
$\mathbf{u}_d$	vektor željenih ulaznih veličina u ulačanoj formi
$\tilde{\mathbf{u}}$	vektor greške ulaznih veličina u ulačanoj formi
$\mathbf{X}$	matrica zakreta
$\mathbf{x}$	vektor stanja u ulančanoj formi

$\mathbf{x}_d$ vektor željenih stanja u ulačanoj formi

$\tilde{\mathbf{x}}$ vektor greške pozicije u ulačanoj formi

Sažetak

Ovaj rad je fokusiran na razvoj regulatora za vođenje mobilnog robota duž zadane trajektorije. Robot je posebno opremljen sa četiri zakretna i neovisno pogonjena kotača. Uz određene pretpostavke, jednadžbe kinematike robota pretvaraju se u dvolančani format s trostrukim ulazom, s pojedinačnim generatorom u lancu. Zatim se provodi linearna aproksimacija ovih jednadžbi za slučaj malih odstupanja oko željene trajektorije. Ovaj pristup rezultira modelom vremenski promjenjivog sustava, koji je pogodan za sintezu vremenski promjenjivog linearnog zakona upravljanja temeljenog na principu kvadratne optimalnosti. Najučinkovitija verzija ovog zakona upravljanja pronađena je računalnim simulacijama provedenim u MATLAB-u (MathWorks, Natick, MA, SAD). Ovaj zakon upravljanja se zatim primjenjuje na stvarnom mobilnom robotu, razvijenom u Regionalnom centru izvrsnosti za robotske tehnologije (CRTA). Implementacija regulatora provedena je sustavom Robot Operating System (ROS), a njegova se izvedba procjenjuje kroz eksperimentalna mjerenja pomoću OptiTrack sustava.

Ključne riječi: mobilni robot s četiri zakretna i četiri neovisno pogonjena kotača, ulančana forma, praćenje trajektorije, vremensko zavisni LQR

Abstract

This master's thesis is focused on developing a controller for guiding a mobile robot along the desired trajectory. The robot is uniquely equipped with four steerable and independently powered wheels. By making certain assumptions, equations of the robot kinematics are transformed into a format known as three-input, two-chain, single-generator chained form. A linear approximation of these equations is then performed for the case in which the perturbations around the desired trajectory are small. This approach results in a time-varying system model, which is suitable for the sinthesis of a time-varying linear control law based on the principle of quadratic optimality. The most effective version of this control law is identified through computer simulations conducted in MATLAB (MathWorks, Natick, MA, USA). This control law is then applied to a real-world mobile robot, developed at the Regional Center of Excellence for Robotic Technologies (CRTA). The implementation of the regulator is carried out by Robot Operating System (ROS), and its performance is assessed through experimental measurements using the OptiTrack system.

Keywords: four-wheel steered and drive mobile robot, chained form, trajectory tracking, time-varying LQR

1 Uvod

U dinamičnom i neprekidno rastućem polju robotike, područje mobilne robotike se razvilo kao ključno područje inovacija, spajajući naprednu tehnologiju s praktičnim primjenama koje preoblikuju našu interakciju s fizičkim svijetom. Područje mobilne robotike nije samo ograničeno na automatizaciju rutinskih zadataka, već i transformira industriju, poboljšava kvalitetu života i prednjači u istraživanjima u okruženjima van dosega čovjeka.

Ovaj se rad dodiruje teme tehnoloških napredaka koji su gurnuli mobilnu robotiku među sam vrh tehnološke inovacije, što je detaljnije opisano u poglavlju **2**. To uključuje napredak u tehnologiji senzora, koji omogućuje robotima da percipiraju svoje okruženje sa visokom točnošću i dubinom, te integraciju upravljačkih algoritama koji omogućuju robotima da se autonomno se prilagođavaju novim izazovima [1].

U trećem je poglavlju rada predstavljen mobilni robot za penjanje po zidovima (eng. Wall Climbing Robot - WCR), gdje je opisan njegov princip rada, konstrukcija, te razvoj kroz nekoliko prototipova [2, 3]. Prototip WCR-a s četiri pogonska motora s neovisnim upravljanjima zakretanjem kotača u ravnini, odabran je kako bi se opisala problematika upravljanja neholonomnim mobilnim robotom, s posebnim naglaskom na korištenje povratnih informacija, očitanih sa senzora, za izvršavanje određenih zadataka kretanja.

Za rješenje navedenog problema, u poglavlju **4** su izvedene nelinearne diferencijalne jednadžbe koje opisuju kinematiku mobilnog robota [4, 5, 6]. Zatim je provedena transformacija kinematičkog modela robota u ulančanu formu dvolančanog formata s trostrukim ulazom i pojedinim generatorom u lancu, pri čemu je bilo potrebno uvođenje odgovarajućih pretpostavki u svrhu dobivanja jednostavnijeg modela [7, 8].

Lančana je forma, u poglavlju **5**, linearizirana za male perturbacije oko referentne trajektorije [9, 10, 11] u obliku vremenski promjenljivog prostora stanja prikladnom za sintezu linearnog vremenski promjenljivog zakona upravljanja prema kvadratnom kriteriju optimalnosti. U MATLAB-u su zatim izrađeni simulacijski modeli u kojima je u zakonu upravljanja korišten vremenski varijabilni linearni kvadradni regulator (eng. Time-Varying Linear Quadratic Regulator - TV LQR) [12], čija se uporaba pri praćenju trajektorije kod mobilnih robota pokazala optimalnom, što je moguće vidjeti u [13], gdje je odstupanje od željene trajektorije zanemarivo, iako dolazi do grešaka u ulaznom signalu. Zakon upravljanja izveden je prema primjeru iz [14], pa je simulacijama na računalu analizirano ponašanje zatvorenog sustava upravljanja u slučajevima nelinearnog i linearizirano modela robota za slučaj nekoliko različitih referentnih trajektorija.

Nakon što su pomoću simulacijskih modela dobiveni zadovoljavajući rezultati, odabrana je jedna od referentnih trajektorija, te se u šestom poglavlju zakon upravljanja implementirao na stvarnom robotu pomoću sustava ROS (Robot Operating System) [15, 16]. Prvo su analizirani rezultati dobiveni iz informacija očitanih sa stanja zglobova robota, koji su se u nekoliko iteracija približavali željenoj putanji. Mjerenja su rađena za slučajeve kada je LQR vremenski varijabilan i kada je invarijabilan, te su greške u dva slučaja međusobno uspoređene. Kada je postignuto potpuno poklapanje rezultata, stvarna putanja robota izmjerena je pomoću OptiTrack sustava i rezultati su uspoređeni sa zadanom putanjom.

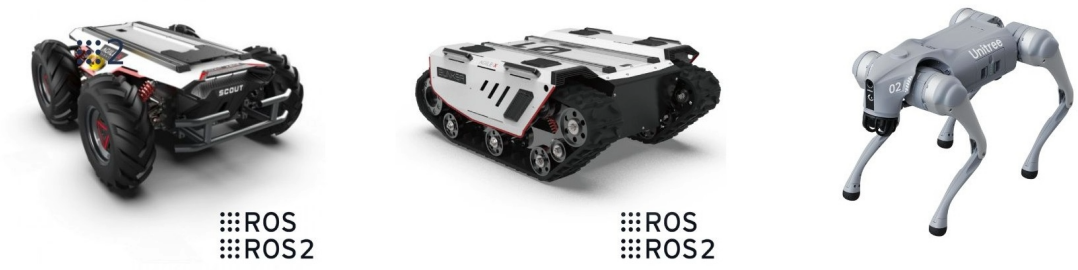
2 Mobilna robotika

Mobilna robotika je brzorastuće područje koje kombinira mehaniku, umjetnu inteligenciju, i tehnologiju robotike kako bi stvorilo robote sposobne za efektivno kretanje u prostoru i izvođenje zadataka bez ljudske intervencije. Ovi roboti su dizajnirani za navigaciju i interakciju s dinamičkim okruženjima, čime revolucioniraju način izvođenja zadataka u brojnim sektorima, od proizvodnje i logistike do zdravstva i mapiranja okoline. Koristeći napredne senzore, algoritme umjetne inteligencije i robusne mehaničke dizajne, ovi roboti nude povećanu učinkovitost, sigurnost i uštedu troškova.

2.1 Sustavi robota

2.1.1 Lokomotorni sustav

Mobilnog je robota moguće opisati kao skupinu tehnoloških podsustava gdje svaki sustav obavlja ključnu funkciju. Lokomotorni sustav zadužen je za samo kretanje te se sastoji od aktuatora i različitih mehanizama za mobilnost kao što su kotači, noge, gusjenice, ili propeleri kod kretanja zrakom ili vodom. Aktuatori na mobilnim robotima najčešće su elektromotori istosmjerne struje, te rjeđe koračni motori. Osim samog kretanja, lokomotorni sustav ima i ulogu manevriranja, što uključuje brzo i efektivno usmjeravanje i zakretanje, pri čemu robot ne smije izgubiti stabilnost. Pri konstruiranju lokomotornog sustava, potrebno je uzeti u obzir i energetsku učinkovitost, prilagodljivost prostoru u kojem se robot kreće, te integraciju senzora i upravljačkog sustava.



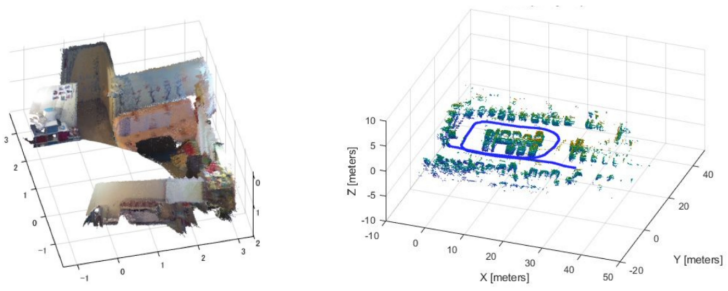
Slika 2.1: Kretanje po tlu; kotači: Scout 2.0 (Agilex), gusjenica: Bunker (Agilex), noge: Go 2 Air (Unitree Robotics)[1]

[1] Generation Robots; https://www.generationrobots.com/en/

Slika 2.2: Kretanje kroz fluid; dron: Crazyflie 2.1 (Bitcraze)[1], podvodni: BlueROV2 (Blue Robotics)[2]

2.1.2 Senzori i navigacijski sustav

Sustav za navigaciju i mapiranje koristi algoritme koji omogućuju robotima da razumiju svoju okolinu, te njom i navigiraju, koristeći tehnike poput SLAM-a (eng. Simultaneous Localization and Mapping). Kako bi lokalizacija i mapiranje bili mogući potrebni su senzori koji očitavaju prostor, poput kamera i LIDAR-a (eng. Light Detection And Ranging) koji koisti laserske senzore. Uzmemo li u obzir SLAM metodologiju moguće je usporediti podatke dobivene s kamera (eng. Visual SLAM - vSLAM) i lasera (LIDAR SLAM); iako kamere mogu pružiti informacije o boji i teksturi, što je prednost za određene primjene poput prepoznavanja objekata, vSLAM često zahtijeva spajanje s drugim senzorima kako bi nadoknadio ograničenja u percepciji dubine, te može imati problema u uvjetima slabog osvjetljenja, dok LIDAR SLAM dosljedno djeluje bez obzira na osvjetljenje i može samostalno stvoriti detaljnije 3D mape, posebno u složenim okruženjima. Generalno LIDAR daje puno točnije rezultate, iako su kamere primjenjivije radi niže cijene. Osim navedenih senzora koji mjere vanjska stanja (eksteroceptivni), propriocepcijski senzori mjere unutarnja stanja robota kao što su brzina robota, kutovi zakreta motora, unutarnja temperatura robota i napon baterije.



Slika 2.3: Visual SLAM (lijevo) i LIDAR SLAM (desno) [3]

[2]Blue Robotics; https://bluerobotics.com/

2.1.3 Upravljački sustav

Upravljački sustav mobilnog robota, koji je ključan za izvršavanje zadataka i postizanje ciljeva, temelji se na tri osnovna područja: percepcija, obrada i kognicija, te akcija. Sustav percepcije prikuplja informacije o okolini, robotu i njihovoj interakciji, koje se obrađuju kako bi se generirale naredbe za aktuatore koji pokreću mehaničku strukturu robota. Kognitivna arhitektura robota odgovorna je za planiranje puta kako bi se postigli ciljevi, čineći je dijelom sustava za donošenje odluka i izvršavanje.

Informacije sa senzora, u kombinaciji s ciljevima robota, vode sustav kognicije i upravljanja u određivanju potrebnih akcija za postizanje ciljeva. Ovaj sustav koordinira sve ulazne podatke i planira kretanje robota. Kognitivni modeli koji predstavljaju robota i njegovu interakciju s okolinom bitni su u ovom procesu. Tehnologije poput računalnog vida, prepoznavanja uzoraka, algoritama mapiranja i algoritama umjetne inteligencije, uključujući planiranje kretanja, koriste se za osiguravanje učinkovite interakcije i izbjegavanje prepreka.

Umjetna inteligencija igra ključnu ulogu u obradi informacija prikupljenih od strane robota. Roboti iskazuju nelinearnu dinamiku, te se koriste tehnike nelinearnog upravljanja za precizno reproduciranje njihovog ponašanja. Jedan od ključnih ciljeva sustava upravljanja je praćenje pozicije robota, pri čemu se greška praćenja koristi za kvantificiranje učinkovitosti upravljanja. Različite strategije upravljanja, temeljene na načinu izračuna greške i reduciranjem iste, primjenjuju se za poboljšanje funkcionalnosti mobilnih robota.

S obzirom na analizirana svojstva (lokalnu stabilnost, odstupanja od fiksnih točaka, fazne dijagrame) i način na koji se pristupa nelinearnim dinamičkim jednadžbama, prema [1] strategije kontrole mogu se klasificirati na slijeći način:

1. Upravljanje temeljeno na globalnoj linearizaciji
2. Upravljanje temeljeno aproksimativnoj linearizaciji
3. Upravljanje temeljeno na Lyapunovljevoj teoriji.

Upravljanje pomoću globalne linearizacije pretvara nelinearnu dinamiku u linearni model prostora stanja. Koristi povratne informacije o stanju za rješavanje problema procjene stanja. Ove metode mogu se dalje podijeliti na temelju teorija poput diferencijalno ravnih sustava i Lie algebre.

Kod aproksimativne linearizacije se lokalni linearni modeli, izvedeni u lokalnim ravnotežama, koriste za rješavanje problema nelinearnog upravljanja. Stabilnost je ključni fokus, a parametri lokalnih upravljača biraju se kako bi se osigurala robusnost

[3]MathWorks; https://ch.mathworks.com/

prema vanjskim smetnjama i nesigurnostima modela.

Cilj upravljanja temeljenog na Lyapunovljevoj teoriji je minimizirati Lyapunovljeve funkcije kako bi se jamčila asimptotska stabilnost kontrolne petlje. Postoje dva glavna pristupa: linearizacija i Lyapunovljeva direktna metoda. Metoda linearizacije omogućuje primjenu linearnih tehnika kontrole, procjenjujući stabilnost na temelju svojstvenih vrijednosti linearne aproksimacije sustava. Lyapunovljeva direktna metoda usredotočuje se na načelo da sustav čija ukupna energija neprestano opada, vodi do točke ravnoteže.

2.2 Podjela prema primjeni

2.2.1 Industrija

Mobilni roboti bitno su utjecali na industrijski sektor, povećanjem efikasnosti, smanjivanjem vremena proizvodnje, kao i produživanjem radnog vremena rada tvornica. U tipičnom proizvodnom okruženju obično se nalazi veći broj industrijskih vozila koja premeštaju materijal ili dijelove koji su u procesu izrade između radnih stanica ili, u naprednijim operacijama, pozicioniraju alate ili robote koji izravno djeluju na dijelove. Postoje mnogi aspekti rada takvih vozila koji moraju biti planirani i koordinirani. Oni uključuju osiguranje da se putevi vozila ne preklapaju, da promet ne postane zagušen, da se materijal dostavlja na prava mjesta u pravo vrijeme i da je protok usklađen s radnim tempom tvornice, te da se vozilima omogući vrijeme za punjenje. Planiranje uključuje određivanje više od samo puteva kojima će vozila slijediti. Također može uključivati osiguranje da vozilo izbjegava drugu opremu ili ljude, omogućujući visoku preciznost priključivanja na transportne trake ili drugu opremu.

Slika 2.4: MiR Hook 250 [4]

2.2.2 Snimanje i mapiranje okoline

Primjenom prethodno navedenih metoda mapiranja, mnogi mobilni roboti koriste se u svrhu skeniranja okoline, bilo radi označavanja prostora za autonomno kretanje, mjerenja neravnina na terenu ili snimanja u svrhu izrada karata. Uzmimo za primjer područje građevine gdje se sve češće koristi fotogrametrija; proces snimanja zračnih fotografija lokacije kako bi se stvorila 3D karta s GPS koordinatama i točnim mjerenjima. Osim dronova, kao primjer vrijedi spomenuti robota za penjanje po zidovima, no taj će robot detaljnije biti opisan u nastavku rada.

Slika 2.5: Foxtechrobot D20 VTOL Fixed Wing Drone [5]

2.2.3 Kućanstvo

Mobilni roboti za kućanstvo dizajnirani su za pomoć u različitim kućanskim poslovima. Uobičajene vrste uključuju robotske usisavače, brisače podova, kosilice, robote za čišćenje prozora i bazena, kao i društvene i sigurnosne robote za socijalnu interakciju, društvo i nadzor doma. Izazovi na ovom području uključuju osiguravanje besprijekorne integracije s drugim pametnim uređajima, čineći tehnologiju pristupačnijom široj bazi potrošača, te poboljšanje na temelju strojnog učenja. Budući trendovi ukazuju na napredniju integraciju, povećanu inteligenciju te veće mogućnosti prilagodbe i personalizacije.

Slika 2.6: iRobot Roomba Combo j5 [6]

[4]Mobile Industrial robots; https://www.mobile-industrial-robots.com/
[5]Foxtechrobot; https://www.foxtechrobot.com/
[6]iRobot; https://www.irobot.com/

2.2.4 Ostala područja primjene

Osim prethodno navedenih područja, uporaba mobilne robotike sve je češća u zdravstvu, agronomiji, ekologiji, uredima ali i za zabavu. Fraunhoferov Care-O-Bot je interesantan primjer mobilnog robota u zdravstvu, koji služi kao mobilni centar za informacije, alat za prikupljanje i dostavu predmeta i pomoćnik za sigurnost ili nadzor. Clearbot je dobar primjer robota korištenog u ekologiji, te može skupiti do 200 kg smeća i 15 L ulja. Osim toga, ima i ulogu nadzorog i spasilačkog autonomnog vozila, te prijenosnika tereta. Kao primjer mobilnih robota za zabavu vrijedi navesti platformu Roborace, koja se bavi autonomnim trkaćim robotima. U suradnji sa Nvidiom razvili su vozilo autonomije stupnja 5, što predstavlja potpunu autonomnost i nema nikakav standardan upravljački hardver (volan, pedale za gas ili kočenje).

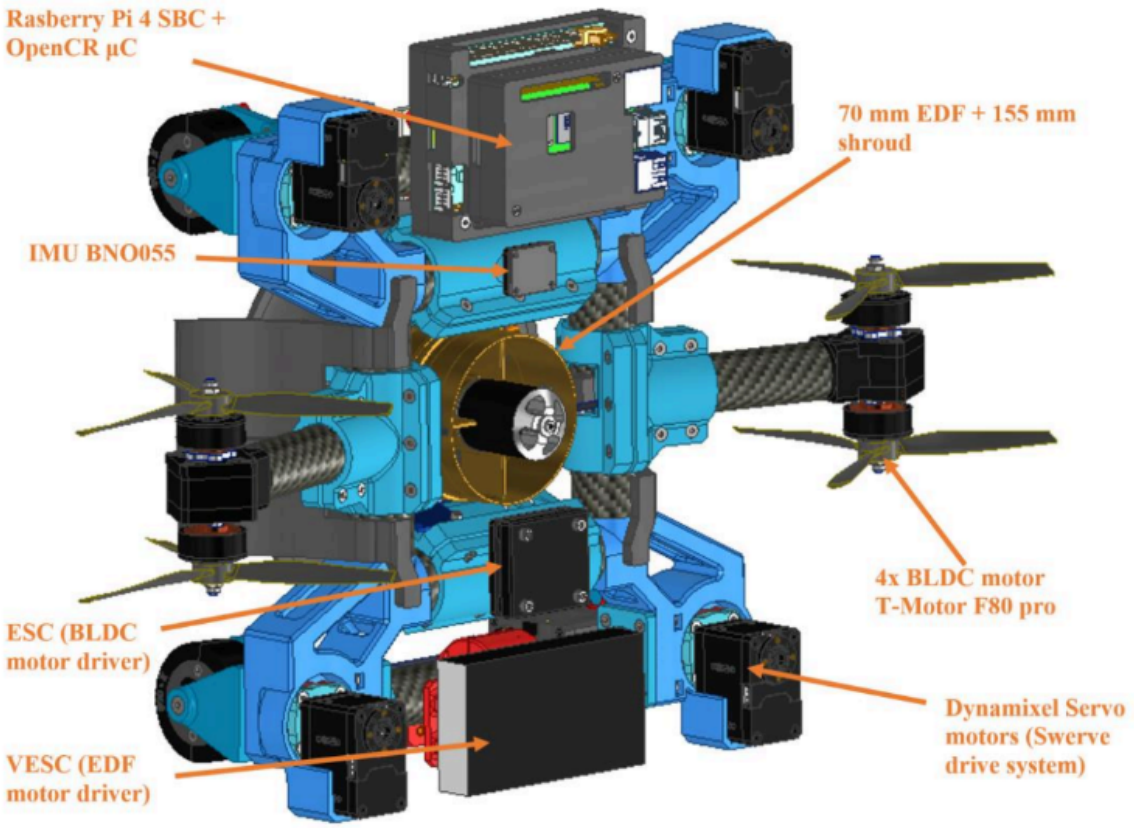
Slika 2.7: Care-O-Bot-4 [7], Clearbot [8], Nvidia Roborace [9]

[7]Care-O-Bot; https://www.care-o-bot.de/
[8]Clearbot; https://www.clearbot.org/
[9]Roborace; https://roborace.com/

3 Opis eksperimentalnog postava mobilnog robota

3.1 Komponente robota

Mobilni robot čijom se kinematikom bavi ovaj rad, je robot za penjanje po zidovima (eng. Wall Climbing Robot - WCR) razvijen u Regionalnom centru izvrsnosti za robotske tehnologije (CRTA). Robot je dizajniran za kretanje po vertikalnim stukturama pri čemu koristi hibridni sustav prijanjanja temeljen na kombinaciji negativnog tlaga i potiska, te se kreće pomoću četiri nezavisno pogonjena i zakretana kotača, čime je omogućeno kretanje u svim smjerovima. Dizajn robota je kompaktan, s kućištem izrađenim 3D tiskanjem, od cijevi karbonskih vlakana i ASA (akrilonitril stiren akrilat) materijala. Dimenzije robota su 380 x 300 mm (bez propelera), s ukupnom težinom od 3.25 kg i ukupnom nosivosti od 1.5 kg.

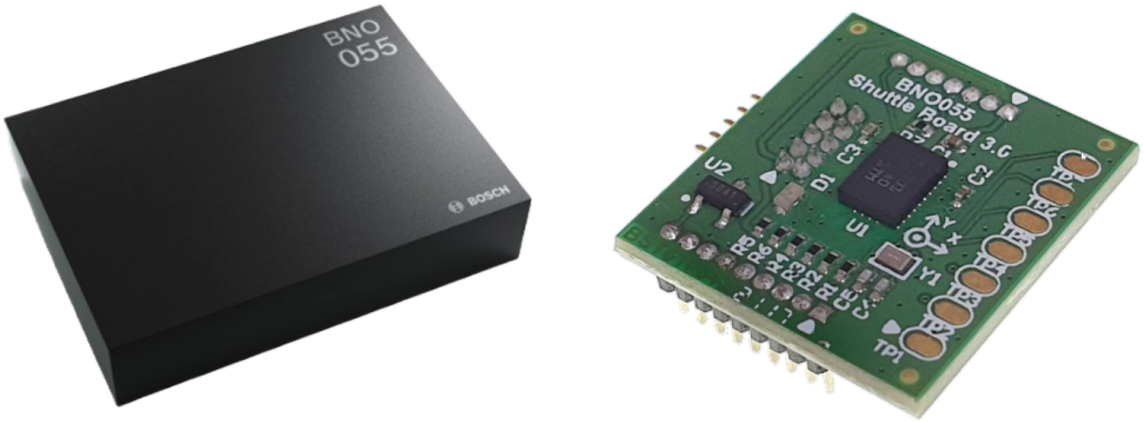


Slika 3.1: WCR - CAD model [2]

Kotači su pokretani Dynamixel XC430 pametnim aktuatorom s integriranim istosmjernim motorom, regulatorom, senzorom, reduktorom, pri čemu je jedan servo modul umrežen. Serija XC omogućuje rotaciju za 360 stupnjeva pomoću beskontaktnog magnetskog enkodera i sklopa sa šupljim dijelom kućišta radi lakšeg zakretanja [1]. Energetski sustav iznosi 24V, s laboratorijskim napajanjem od 3kW pomoću kabela.

Slika 3.2: Dynamixel XC430 [1]

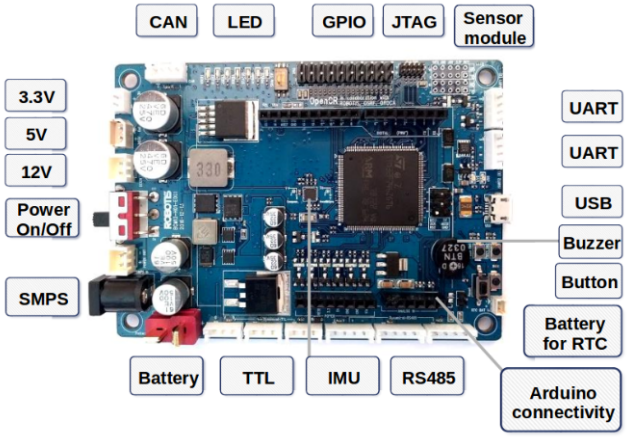
Senzorski sustav, osim navedenog enkodera motora, sadrži i Boschov BNO055. Bosch BNO055 je IMU (eng. Inertial Measurement Unit), što znači da mjeri i objavljuje specifičnu silu tijela, kutnu brzinu i orijentaciju tijela, koristeći troosni 14-bitni akcelerometar, troosni 16-bitni žiroskop zatvorenog kruga, troosni geomagnetski senzor i 32-bitni mikrokontroler sa softverom BSX3.0 FusionLib [2].



Slika 3.3: Bosch BNO055 [2]

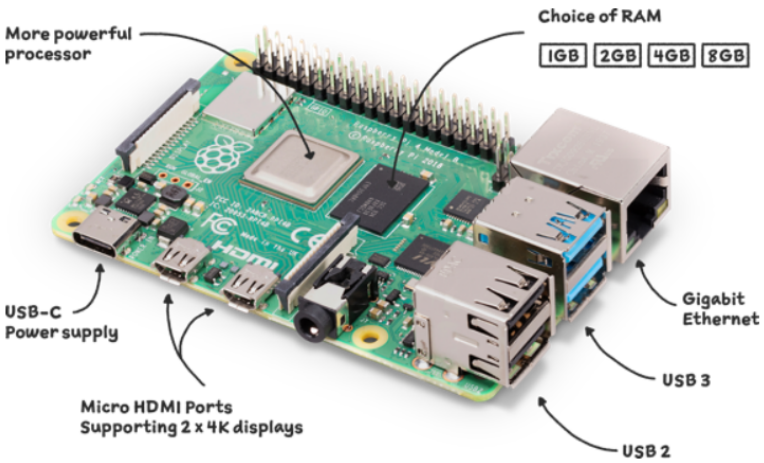
[1] Dynamixel; https://www.robotis.us/dynamixel/
[2] Bosch Sensortec; https://www.bosch-sensortec.com/

Motor je upravljan pomoću OpenCR ploče, koja pruža hardver i softver za otvorenu upotrebu za sustave koji koriste ROS. Ima ugrađen mikrokontroler STM32F7 s procesorom ARM Cortex-M7, koji pomoću FPU-a (eng. Floating Point Unit) može izvoditi operacije s brojevima tipa float. Osim toga sadrži i senzorni modul kompatibilan s Boschovim IMU-om i enkoderima pametnih aktuatora Dynamixel [3].



Slika 3.4: OpenCr [3]

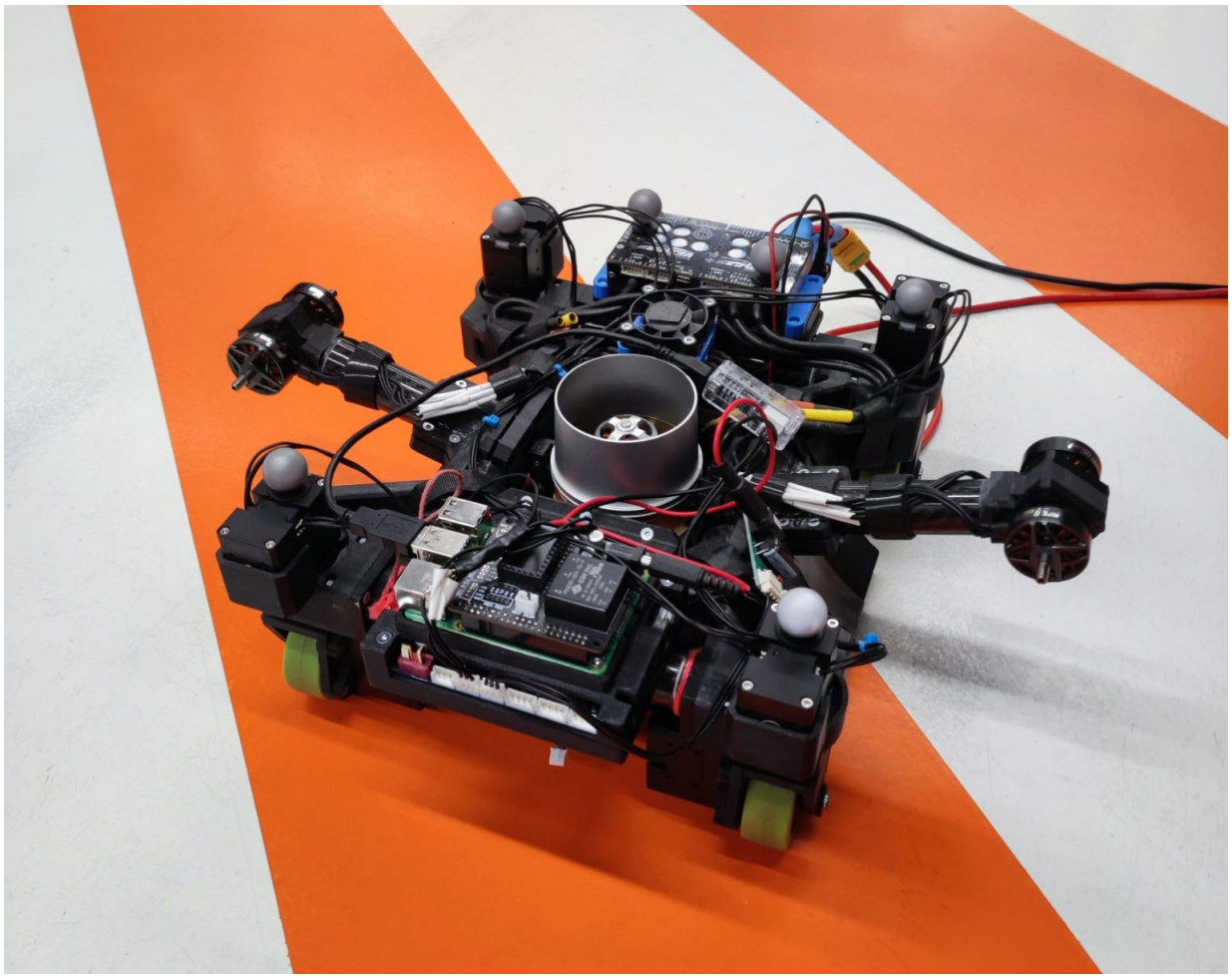
Mikrokontroleri komuniciraju s Raspberry Pi4B jednopločnim računalom s 8 GB RAM memorije. Uz dva USB 2 utora, model Pi4B ima dva USB 3 utora koji mogu prenositi podatke do deset puta brže od svog prethodnika, što je značajno s obzirom da se komunikacija vrši putem USB protokola [4]. Raspberry Pi4B radi na Linuxu i koristi ROS posrednički softver. Daljinsko upravljanje robotom moguće je preko WiFi-a u ROS Master-Slave postavci ili joystickom povezanim putem Bluetootha.



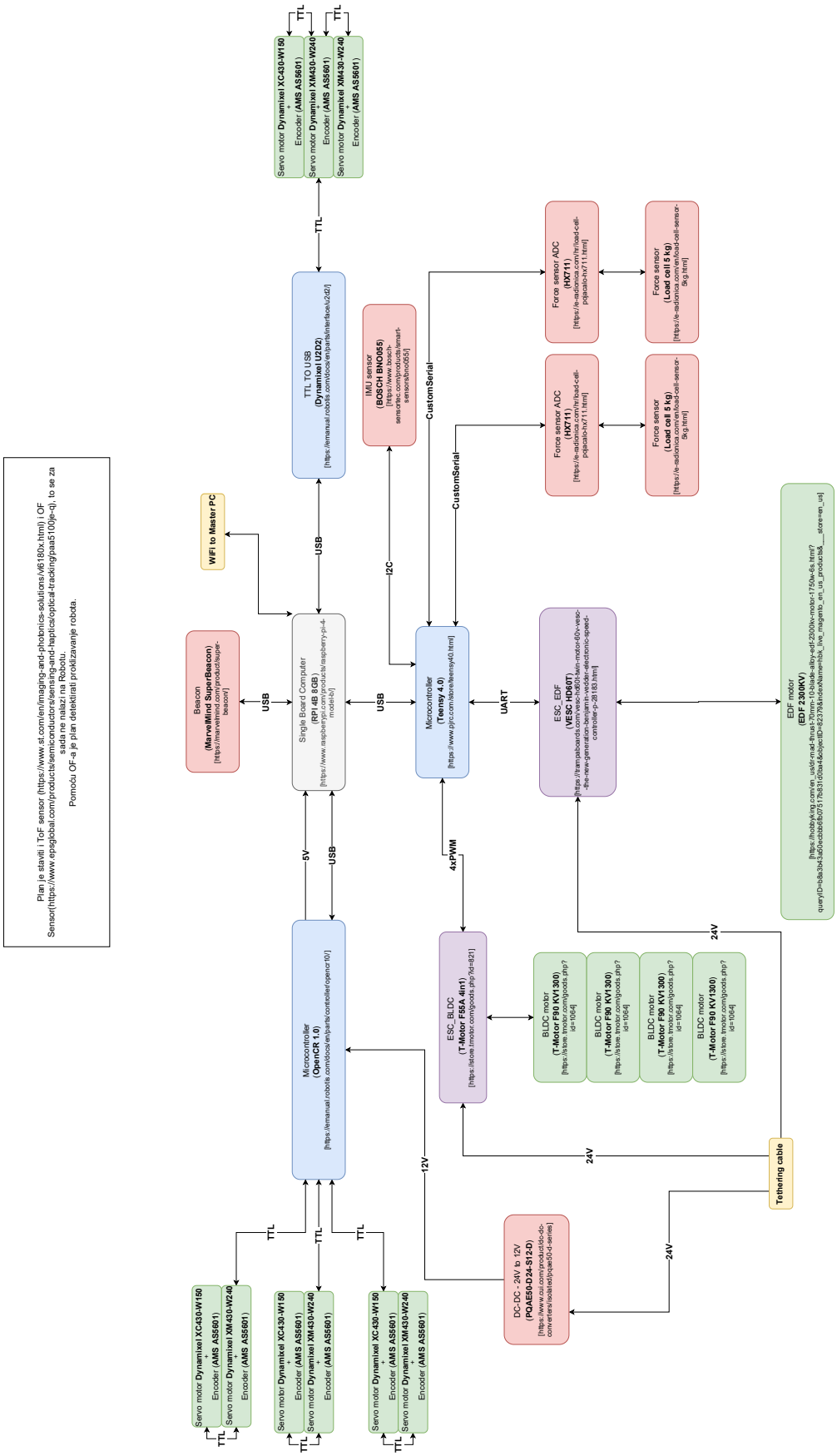
Slika 3.5: Raspberry Pi4B [4]

[3]Robotis; https://www.robotis.us/
[4]Raspberry Pi; https://www.raspberrypi.com/

Hibridni se sustav prianjanja sastoji od jedinice električnog kanaliziranog ventilatora (eng. Electric Ducted Fans - EDF) promjera 70 mm s kućištem širine 155 mm i razmakom kućišta 7,5 mm. Prianjanje se upravlja NXP iMXRT1062 mikrokontrolerom. Za mjerenje sile prianjanja generirane EDF jedinicom, robot je opremljen s dva senzora sile. Osim EDF-a, robot uključuje pogonsku jedinicu drona s dva para BLDC motora (T-Motor F80 Pro) opremljenih s propelerima duljine 15.2 cm. Ova pogonska jedinica strateški je pozicionirana blizu centra gravitacije robota kako bi se poboljšao i sustav prianjanja i ukupna pokretljivost S obzirom da se ovaj rad ne bavi kretanjem po vertikalnim površinama, komponente sustava prijanjanja neće biti detaljno opisane, te je detaljan opis moguće pronaći u [2].



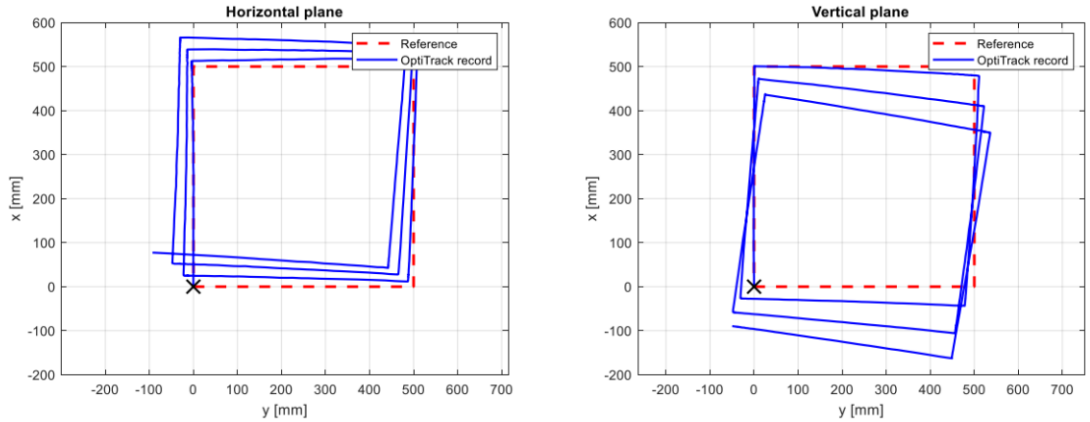
Slika 3.6: WCR



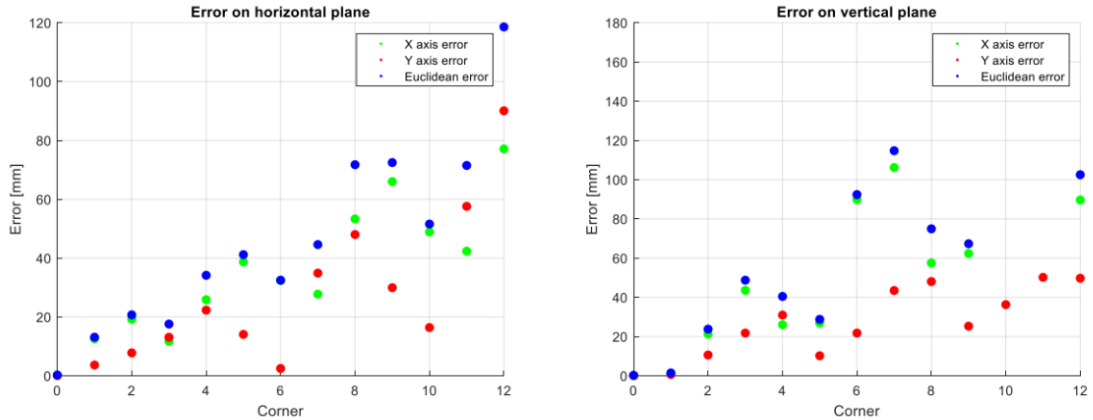
Slika 3.7: Blok dijagram robota

3.2 Mjerenja performansi robota

Preliminarna ispitivanja su provedena na horizontalnoj i vertikalnoj površini. Robot je daljinski upravljan u otvorenoj petlji koristeći Bluetooth joystick i uspješno je vožen na betonskoj i glatkoj površini. Kako bi se ispitalo ponašanje odometrije kinematike okretnog pogona, provedena su dva eksperimentalna mjerenja. Robotu je dodijeljeno 13 točaka koje ocrtavaju kvadratni oblik, a morao se kretati prema tim točkama koristeći odometriju iz enkodera motora i modela kinematike okretnog pogona kao povratnu petlju. Kao lokalni planer za ta mjerenja koristilo se $v = v_{max} \tanh(e)$, kako vi se osiguralo da robot ne prelazi maksimalnu brzinu v_{max} fiksirana na 0.15 m/s. Argument funkcije tangens hiperbolni e je greška udaljenosti između željene i trenutne lokacije robota. Za usporedbu odometrijskih mjerenja s apsolutnim vrijednostima korišten je OptiTrack kao vanjski sustav mjerenja. OptiTrack ima točnost manju od 0.5 mm što je sasvim dovoljno za ovu vrstu mjerenja[2].



Slika 3.8: Testiranje kretanja po horizontalnoj (lijevo) i vertikalnoj (desno) površini [2]



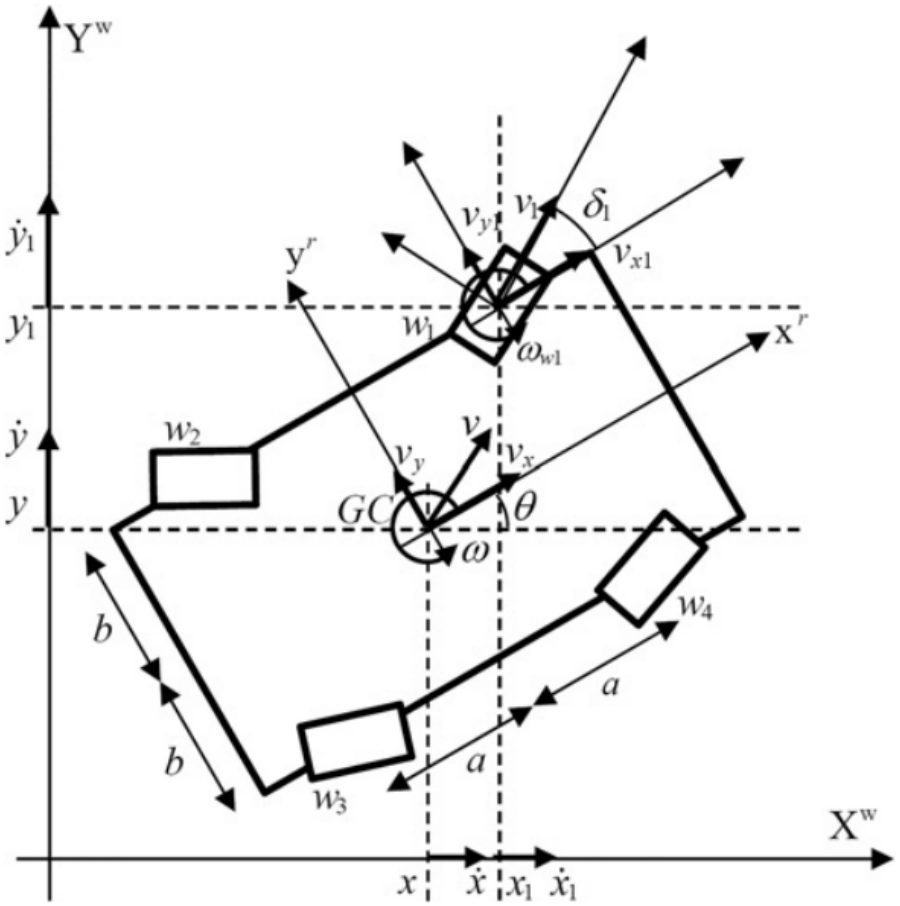
Slika 3.9: Greška u odometriji u kutevima kvadrata na horizontalnoj (lijevo) i vertikalnoj (desno) površini [2]

Na slici 3.8 u horizontalnoj ravnini je vidljivo da na svakom kutu kvadrata dolazi do greške, koja nastaje radi nepreciznog kuta zakreta. Na slici 3.9 se vidi kako se ta pogreška akumulira u x i y osima, te raste svakim zakretom. U vertikalnoj ravnini na slici 3.8 primjećuje se ista greška, no u rezultatima na slici 3.9 akumulirana greška vidno manja. Do smanjenja greške dolazi radi klizanja robota prema dolje što se može primjetiti na slici 3.8, gdje putanja robota između dvije točke nije potpuno ravna. Iz priloženih grafova da se zaključiti da se akumulira greška kod odometrije robota, što će utjecati i na rezultate u nastavku, te je potrebno daljnje istraživanje.

4 Kinematika mobilnog robota

4.1 Kinematički model

U ovom će odlomku biti prikazana kinematika mobilnog robota s četiri zakretna i četiri neovisno pogonjena kotača prema članku [4], u kojoj se u obzir ne uzimaju pokreti pri nagibu, kotrljanju i vertikalnom pomaku. Iz članka [4] prikazana je konfiguracija robota u globalnom koordinatnom sustavu na slici 4.1.



Slika 4.1: Konfiguracija robota u globalnom koordinatnom sustavu [4]

Na svakom se kotaču nalazi motor za pokretanje te dodatni motor za zakretanje. x, y i θ predstavljaju položaje robota unutar globalnog koordinatnog sustava, w_i označava kotač broj i, a δ_i predstavlja kut skretanja tog kotača. a i b su udaljenosti između središta robota i svakog kotača. S obzirom da je položaj težišta WCR-a jednak težištu robota, te da svi kotači imaju jednak radijus i masu, pozicije kotača su definirane na slijedeći način:

$$(x_{w1}, y_{w1}) = (a, b),$$
$$(x_{w2}, y_{w2}) = (-a, b),$$
$$(x_{w3}, y_{w3}) = (-a, -b),$$
$$(x_{w4}, y_{w4}) = (a, -b),$$

gdje x_{wi}, y_{wi} predstavljaju koordinate kotača u robotovom koordinatnom sustavu i vrijedi:

$$a = b = 112.5 \text{ mm}.$$

Nadalje, koristit ćemo izraze v i v_i za linearnu brzinu robota i kotača i, odnosno ω i ω_i za kutne brzine, pri čemu vrijedi:

$$v = \sqrt{v_x^2 + v_y^2},$$
$$v_i = \sqrt{v_{xi}^2 + v_{yi}^2}.$$

Ukoliko ne dolazi do klizanja, odnosi brzine između svakog kotača i tijela robota dobivaju se ograničavanjem stupnjeva slobode gibanja za kruto tijelo:

$$v_{xi} = v_i \cos(\delta_i) = v_x - y_{wi}\omega, \tag{4.1}$$
$$v_{yi} = v_i \cos(\delta_i) = v_y + x_{wi}\omega. \tag{4.2}$$

Unošenjem definiranih parametara u jednadžbe 4.1 i 4.2 slijedi:

$$\mathbf{P} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \mathbf{X} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}, \tag{4.3}$$

pri čemu su: $\mathbf{P} = \begin{bmatrix} 1 & 0 & -b \\ 0 & 1 & a \\ 1 & 0 & -b \\ 0 & 1 & -a \\ 1 & 0 & b \\ 0 & 1 & -a \\ 1 & 0 & b \\ 0 & 1 & a \end{bmatrix}$, $\mathbf{X} = \begin{bmatrix} \cos(\delta_1) & 0 & 0 & 0 \\ \sin(\delta_1) & 0 & 0 & 0 \\ 0 & \cos(\delta_2) & 0 & 0 \\ 0 & \sin(\delta_2) & 0 & 0 \\ 0 & 0 & \cos(\delta_3) & 0 \\ 0 & 0 & \sin(\delta_3) & 0 \\ 0 & 0 & 0 & \cos(\delta_4) \\ 0 & 0 & 0 & \sin(\delta_4) \end{bmatrix}$.

Sada je potreban pseudoinverz matrice $\mathbf{P}$ koji glasi:

$$\mathbf{P}^{+} = \begin{bmatrix} \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{b}{L} & \frac{a}{L} & -\frac{b}{L} & -\frac{a}{L} & \frac{b}{L} & -\frac{a}{L} & \frac{b}{L} & \frac{a}{L} \end{bmatrix},$$

gdje je:

$$L = 4a^2 + 4b^2.$$

Množenjem jednadžbe 4.3 pseudoinverzom $\mathbf{P}^{+}$ s lijeve strane dobivamo:

$$\mathbf{I} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \mathbf{P}^{+}\mathbf{X} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\cos(\delta_1)}{4} & \frac{\cos(\delta_2)}{4} & \frac{\cos(\delta_3)}{4} & \frac{\cos(\delta_4)}{4} \\ \frac{\sin(\delta_1)}{4} & \frac{\sin(\delta_2)}{4} & \frac{\sin(\delta_3)}{4} & \frac{\sin(\delta_4)}{4} \\ W_1 & W_2 & W_3 & W_4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{4.5}$$

pri čemu vrijedi:

$$W_i = \frac{x_{wi}\sin\delta_i - y_{wi}\cos\delta_i}{4x_{wi}^2 + 4y_{wi}^2}$$

Kombinacijom tri stanja pozicije robota, kuta rotacije i kuta zakreta svakog kotača, dobiva se jedanaest varijabli stanja koje predstavljaju položaj i stanje mobilnog robota:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \\ \dot{\delta}_1 \\ \dot{\delta}_2 \\ \dot{\delta}_3 \\ \dot{\delta}_4 \end{bmatrix} = \begin{bmatrix} c_1/4 & c_2/4 & c_3/4 & c_4/4 & 0 & 0 & 0 & 0 \\ s_1/4 & s_2/4 & s_3/4 & s_4/4 & 0 & 0 & 0 & 0 \\ W_1 & W_2 & W_3 & W_4 & 0 & 0 & 0 & 0 \\ r^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{4.6}$$

pri čemu su c_i i s_i jednaki:

$$c_i = \cos(\delta_i + \theta)$$
$$s_i = \sin(\delta_i + \theta).$$

4.2 Ulančana forma

U ovom se radu promatra slučaj u kojem se svaki kotač rotira jednakom brzinom, i gdje se prednji kotači i zadnji kotači kreću međusobno paralelno. U tom slučaju uvođenjem jednakosti:

$$v_1 = v_4 = v_2 = v_3$$
$$\delta_1 = \delta_4, \quad \delta_2 = \delta_3,$$

u jednadžbu 4.6, dobiva se slijedeći niz jednadžbi:

$$\dot{x} = \frac{1}{2}\left(\cos(\delta_1 + \theta) + \cos(\delta_2 + \theta)\right) v_1 \tag{4.7}$$
$$\dot{y} = \frac{1}{2}\left(\sin(\delta_1 + \theta) + \sin(\delta_2 + \theta)\right) v_1 \tag{4.8}$$
$$\dot{\theta} = \left(\frac{x_{w1}\sin\delta_1}{2x_{w1}^2 + 2y_{w1}^2} + \frac{x_{w2}\sin\delta_2}{2x_{w2}^2 + 2y_{w2}^2}\right) v_1 \tag{4.9}$$
$$\dot{\phi_1} = \frac{1}{r} v_1 \tag{4.10}$$
$$\dot{\delta_1} = \omega_1 \tag{4.11}$$
$$\dot{\delta_2} = \omega_2. \tag{4.12}$$

S obzirom da jednadžba 4.10 nije nužna za opis gibanja robota, uklanjanjem izraza dobivamo sustav sa pet varijabli stanja; x, y, θ, δ_1 i δ_2 i tri upravljačke ulazne varijable v_1, ω_1 i ω_2 kojeg je moguće zapisati u slijedećem obliku:

$$\dot{\xi} = \mathbf{g}_1(\xi)v_1 + \mathbf{g}_2(\xi)\omega_1 + \mathbf{g}_3(\xi)\omega_2,$$

pri čemu su:

$$\mathbf{g}_1(\xi) = \begin{bmatrix} \frac{1}{2}\left(\cos(\delta_1 + \theta) + \cos(\delta_2 + \theta)\right) \\ \frac{1}{2}\left(\sin(\delta_1 + \theta) + \sin(\delta_2 + \theta)\right) \\ \frac{x_{w1}\sin\delta_1}{2x_{w1}^2 + 2y_{w1}^2} + \frac{x_{w2}\sin\delta_2}{2x_{w2}^2 + 2y_{w2}^2} \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{g}_2(\xi) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{g}_3(\xi) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.13}$$

Ulančana forma je kanonska struktura koja se često koristi kod upravljanja u zatvorenom krugu. U nastavku slijedi transformacija sustava u ulančanu formu za dvolančani format s trostrukim ulazom ($m = 3$), s pojedinačnim generatorom u lancu [8]:

$$\begin{aligned} \dot{z}_1^0 &= u_1 \\ \dot{z}_2^0 &= u_2 \qquad \dot{z}_3^0 = u_3 \\ \dot{z}_{21}^1 &= z_2^0 u_1 \quad \dot{z}_{31}^1 = z_3^0 u_1, \end{aligned}$$

koja je radi jednostavnosti zapisana na slijedeći način:

$$\dot{x}_1 = u_1 \tag{4.14}$$

$$\dot{x}_2 = u_2 \tag{4.15}$$

$$\dot{x}_3 = x_2 u_1 \tag{4.16}$$

$$\dot{x}_4 = u_3 \tag{4.17}$$

$$\dot{x}_5 = x_4 u_1. \tag{4.18}$$

Ulančana froma ima temeljnu linearnu strukturu koja se može prikazati u slučaju kada je u_1 funkcija vremena i više se ne promatra kao upravljačka varijabla. U tom slučaju sustav jednadžbi (4.14-4.18) postaje jednovarijabilni vremenski ovisan linearni sustav. Transformacija se odvija uvođenjem navedenih supstitucija:

$$x_1 = x, \qquad x_3 = \theta, \qquad x_5 = y,$$

nakon čega se izrazi deriviraju, te se sređivanjem dolazi do ulančane forme i izraza za transformaciju upravljačkih veličina.

$$\dot{x}_1 = \dot{x} = u_1 \tag{4.19}$$

$$u_1 = \frac{1}{2}\left(\cos\left(\delta_1 + \theta\right) + \cos\left(\delta_2 + \theta\right)\right) v_1 \tag{4.20}$$

$$v_1 = \frac{2u_1}{\cos\left(\delta_1 + \theta\right) + \cos\left(\delta_2 + \theta\right)} \tag{4.21}$$

$$\dot{x}_5 = \dot{y} = x_4 u_1 \tag{4.22}$$

$$x_4 = \frac{v_1(\sin(\delta_1 + \theta) + \sin(\delta_2 + \theta))}{2u_1} \tag{4.23}$$

$$x_4 = \tan\left(\frac{\delta_1 + \delta_2}{2} + \theta\right) \tag{4.24}$$

$$\dot{x}_4 = u_3 \tag{4.25}$$

$$u_3 = \left[\tan\left(\frac{\delta_1 + \delta_2}{2} + \theta\right)\right]' \tag{4.26}$$

$$u_3 = \frac{\frac{2u1\left(\frac{x_{w1}\sin(\delta_1)}{{x_{w1}}^2+{y_{w1}}^2} + \frac{x_{w2}\sin(\delta_2)}{{x_{w2}}^2+{y_{w2}}^2}\right)}{\cos(\delta_1+\theta)+\cos(\delta_2+\theta)} + \omega_1 + \omega_2}{\cos(\delta_1 + \delta_2 + 2\theta) + 1} \tag{4.27}$$

$$u_3 = \frac{v_1\left(\frac{x_{w1}\sin(\delta_1)}{{x_{w1}}^2+{y_{w1}}^2} + \frac{x_{w2}\sin(\delta_2)}{{x_{w2}}^2+{y_{w2}}^2}\right) + \omega_1 + \omega_2}{\cos(\delta_1 + \delta_2 + 2\theta) + 1} \tag{4.28}$$

$$\dot{x}_3 = \dot{\theta} = x_2 u_1 \tag{4.29}$$

$$x_2 = \frac{\frac{x_{w1}\sin(\delta_1)}{x_{w1}^2+{y_{w1}}^2} + \frac{x_{w2}\sin(\delta_2)}{{x_{w2}}^2+{y_{w2}}^2}}{\cos(\delta_1 + \theta) + \cos(\delta_2 + \theta)} \tag{4.30}$$

$$\dot{x}_2 = u_2 \tag{4.31}$$

$$u_2 = \left[\frac{\frac{x_{w1}\sin(\delta_1)}{{x_{w1}}^2+{y_{w1}}^2} + \frac{x_{w2}\sin(\delta_2)}{{x_{w2}}^2+{y_{w2}}^2}}{\cos(\delta_1 + \theta) + \cos(\delta_2 + \theta)}\right]' \tag{4.32}$$

Radi jednostavnijeg zapisa u nastavku se uvode slijedeće supstitucije:

$$\begin{aligned} c_i &= \cos(\delta_i + \theta) \\ s_i &= \sin(\delta_i + \theta) \\ B_i &= \frac{x_{wi}\cos(\delta_i)}{{x_{wi}}^2 + {y_{wi}}^2} \\ D_i &= \frac{x_{wi}\sin(\delta_i)}{{x_{wi}}^2 + {y_{wi}}^2} \\ K &= 1 + \cos(\delta_1 + \delta_2 + 2\theta). \end{aligned}$$

$$\begin{aligned} u_2 =& \frac{B_1\omega_1 + B_2\omega_2}{c_1 + c_2} \\ &+ \frac{(D_1 + D_2)((c_1 + c_2)(2\omega_1 s_1 + 2\omega_2 s_2) + v_1(D_1 + D_2)(s_1 + s_2))}{2(c_1 + c_2)^2} \end{aligned} \tag{4.33}$$

Konačno, dobivamo ulančanu formu:

$$x_1 = x \tag{4.34}$$

$$x_2 = \frac{D_1 + D_2}{c_1 + c_2} \tag{4.35}$$

$$x_3 = \theta \tag{4.36}$$

$$x_4 = \tan\left(\frac{\delta_1 + \delta_2}{2} + \theta\right) \tag{4.37}$$

$$x_5 = y \tag{4.38}$$

sa ulaznim transformacijama:

$$v_1 = \frac{2u_1}{c_1 + c_2} \tag{4.39}$$

$$\begin{aligned}\omega_1 =& \frac{B_2(2u_1(D_1 + D_2) - Ku_3(c_1 + c_2)) + u_2(c_1 + c_2)^2}{(B_1 - B_2)(c_1 + c_2) + (D_1 + D_2)(s_1 - s_2)} \\ &+ \frac{(D_1 + D_2)(-Ks_2u_3(c_1 + c_2) - (u_1(D_1 + D_2)(s_1 - s_2)))}{(c_1 + c_2)((B_1 - B_2)(c_1 + c_2) + (D_1 + D_2)(s_1 - s_2))}\end{aligned} \tag{4.40}$$

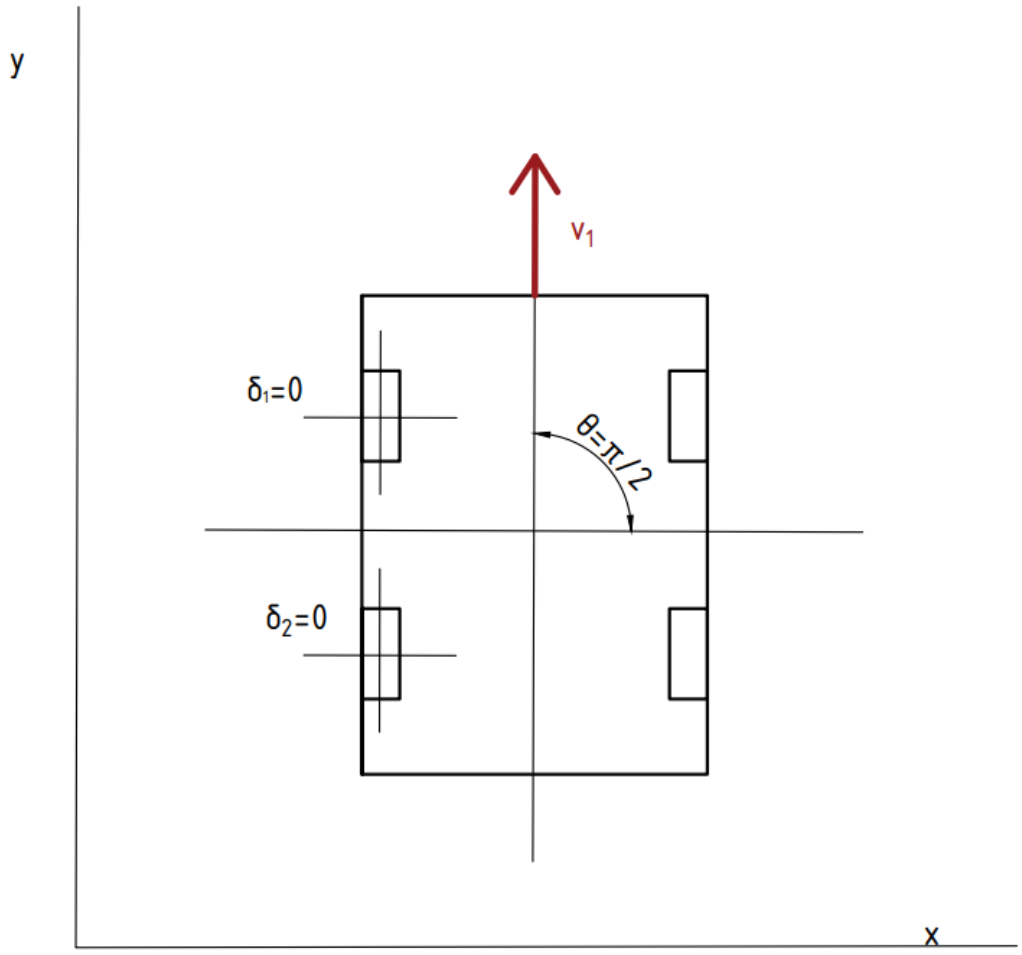
$$\begin{aligned}\omega_2 =& \frac{B_1(-2u_1(D_1 + D_2) + Ku_3(c_1 + c_2)) - u_2(c_1 + c_2)^2}{(B_1 - B_2)(c_1 + c_2) + (D_1 + D_2)(s_1 - s_2)} \\ &+ \frac{(D_1 + D_2)(Ks_1u_3(c_1 + c_2) - (u_1(D_1 + D_2)(s_1 - s_2)))}{(c_1 + c_2)((B_1 - B_2)(c_1 + c_2) + (D_1 + D_2)(s_1 - s_2))}.\end{aligned} \tag{4.41}$$

4.3 Ograničenja

U jednadžbama 4.35 i 4.37 primjećujemo da se u nazivniku nalazi član kojeg je potrebno ograničiti.

$$\begin{aligned} c_1 + c_2 &\neq 0 \\ \cos(\delta_1 + \theta) + \cos(\delta_2 + \theta) &\neq 0 \\ \delta_1 + \delta_2 + 2\theta &\neq \pi + 2k\pi \quad \forall k \in \mathbb{Z} \end{aligned}$$

Isto ograničenje dobivamo iz 4.39 - 4.41, iz čega je moguće zaključiti da sustav nije dobro definiran u trenutku kada je $\dot{x} = u_1 = 0$, to jest kada se robot kreće pravocrtno u smjeru osi y.



Slika 4.2: Prikaz kretanja u kojem robot nije dobro definiran

5 Praćenje trajektorije

5.1 Formulacija problema praćenja trajektorije i tranformacija u vremenski varijantni sustav

Željena trajektorija je izvediva kada ju je moguće dobiti iz sustava jednadžbi koji kao referencu imaju mobilnog robota. U ovom slučaju izabiremo prethodno dobivenu ulančanu formu.

$$\dot{x}_{d1} = u_{d1} \tag{5.1}$$
$$\dot{x}_{d2} = u_{d2} \tag{5.2}$$
$$\dot{x}_{d3} = x_{d2}u_{d1} \tag{5.3}$$
$$\dot{x}_{d4} = u_{d3} \tag{5.4}$$
$$\dot{x}_{d5} = x_{d4}u_{d1}. \tag{5.5}$$

Za prikaz ulančane forme, pogreške stanja i ulaza, prema [4], prikazane su na slijedeći način:

$$\tilde{x}_i = x_i - x_{di}, \quad i = 1, ..., 5, \qquad \tilde{u}_j = u_j - u_{dj}, \quad j = 1, 2, 3,$$

pa nelinearne jednadžbe greške glase:

$$\dot{\tilde{x}}_1 = \tilde{u}_1 \tag{5.6}$$
$$\dot{\tilde{x}}_2 = \tilde{u}_2 \tag{5.7}$$
$$\dot{\tilde{x}}_3 = x_2u_1 - x_{d2}u_{d1} \tag{5.8}$$
$$\dot{\tilde{x}}_4 = \tilde{u}_3 \tag{5.9}$$
$$\dot{\tilde{x}}_5 = x_4u_1 - x_{d4}u_{d1}. \tag{5.10}$$

Linearizacijom oko željene trajektorije dobivamo vremenski varijantni sustav:

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}(t)\tilde{\mathbf{x}} + \mathbf{B}(t)\tilde{\mathbf{u}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & u_{d1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{d1} & 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_{d2} & 0 & 0 \\ 0 & 0 & 1 \\ x_{d4} & 0 & 0 \end{bmatrix} \tilde{\mathbf{u}}, \tag{5.11}$$

pri čemu je sustav upravljiv za $u_{d1} \neq 0$. Uvrštavanjem zakona upravljanja:

$$\tilde{\mathbf{u}} = \mathbf{u}_d - \mathbf{K}(t)(\mathbf{x} - \mathbf{x}_d) \tag{5.12}$$

u jednadžbe stanja linearnih vremenski-varijantnog sustava, pri čemu je $\mathbf{K}$(t) matrica pojačanja određena pomoću linearnog kvadratnog regulatora, dobiva se:

$$\dot{\tilde{\mathbf{x}}} = (\mathbf{A}(t) - \mathbf{B}(t)\mathbf{K}(t))\tilde{\mathbf{x}} + \mathbf{B}(t)\mathbf{u}_d. \tag{5.13}$$

5.2 Sinteza vremenski varijantnog zakona upravljanja prema LQ kriteriju

Regulatori s vremenski varijabilnim linearnim kvadratnim povratnim informacijama su sada praktični zbog bržih računala s velikim kapacitetom memorije. Oni poboljšavaju brzinu i točnost za brze manevriranje zrakoplova, svemirskih letjelica ili robota u usporedbi s regulatorima s vremenski nepromjenjivim koeficijentima, te nisu pretjerano složeniji kada je sustav u digitalnom obliku (zero-order-hold) [12].

Vremenski promjenjivi regulatori mogu se kategorizirati po preciznosti; na meke ili krute. Meki regulatori dizajnirani su korištenjem kvadratne kazne na pogreškama, dok je zadaća krutih regulatora pratiti zadane uvjete bez greške [12]. S obzirom da stvarni sustavi imaju smetnje, kruti su regulatori teško izvedivi, pa se u ovome radu koristi meki regulator, gdje je potrebno pronaći vektor upravljanja $\mathbf{u}(t)$ koji minimizira vrijednosnu funkciju u periodu između t_0 i t_f:

$$\mathbf{J} = \mathbf{x}^T(t_f)\mathbf{F}(t_f)\mathbf{x}(t_f) + \int_{t_0}^{t_f} (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{Q}^T\mathbf{R}\mathbf{u} + 2\mathbf{x}^T\mathbf{N}\mathbf{u})dt. \tag{5.14}$$

pri čemu matrica $\mathbf{Q}$ utječe na performanse, a $\mathbf{R}$ na djelovanje aktuatora. Matrica $\mathbf{N}$ utječe na nusprodukt vektora $\mathbf{x}$ i $\mathbf{u}$, što ćemo u ovom slučaju zanemariti. $\mathbf{F}(t_f)$ je simetrična matrica koeficijenata koja predstavlja vrijednosti u poslijednjem trenutku t_f.

U diskretnom vremenu, uz $\mathbf{N} = 0$, ta jednadžba glasi:

$$\mathbf{J} = \mathbf{x}_{k_f}^T\mathbf{H}_{k_f}\mathbf{x}_{k_f} + \sum_{k=0}^{k_f}(\mathbf{x}_k^T\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k). \tag{5.15}$$

gdje je:

$$\mathbf{K}_k = ({\mathbf{B}_k}^T \mathbf{P}_k \mathbf{B}_k + \mathbf{R})^{-1} ({\mathbf{B}_k}^T \mathbf{P}_k \mathbf{A}_k), \tag{5.16}$$

pri čemu je $\mathbf{P}$ simetrična matrica nejednaka 0, a računa se iterativnim unazadnim postupkom rješavanja Riccatijeve jednadžbe:

$$\mathbf{P}_{k-1} = \mathbf{A}_k^T \mathbf{P}_k \mathbf{P}_k - (\mathbf{A}_k^T \mathbf{P}_k \mathbf{B}_k)(\mathbf{R} + \mathbf{B}_k^T \mathbf{P}_k \mathbf{B}_k)^{-1} (\mathbf{B}_k^T \mathbf{P}_k \mathbf{B}_k) + \mathbf{Q}, \tag{5.17}$$

gdje vrijedi $\mathbf{P}_{k_f} = \mathbf{H}_{k_f}$. Konačno, prema [12] dobiven je zakon upravljanja:

$$\mathbf{u}(t) = \mathbf{K}(t)(\mathbf{x}(t) - \mathbf{x}_d(t)) + \mathbf{u}_d(t). \tag{5.18}$$

Za rješavanje Riccatijeve jednadžbe kojom smo dobili matricu $\mathbf{K}(t)$ korištena je MATLAB-ova gotova funkcija `lqr()`, a kod je dan u prilogu **A**.

5.3 Kružnica kao referentna trajektorija

Za referentnu trajektoriju odabiremo kružnicu koja kreće iz točke 0, 0 u kartezijevom sustavu pa su time koordinate definirane sijedećim izrazima:

$$x_d(t) = R\sin(\omega t), \qquad \theta_d(t) = \omega t, \qquad y_d(t) = R(1 - \cos(\omega t)),$$

pri čemu R predstavlja radius kružnice, a ω period putanje, te vrijedi:

$$R\omega = v_{max}.$$

Analogno postupku iz **5.1**, uvodi se supstitucija:

$$x_{d1} = x_d(t), \qquad x_{d3} = \theta_d(t), \qquad x_{d5} = y_d(t),$$

nakog koje slijedi transformacija sustava u ulančanu formu.

$$x_{d1} = R\sin(\omega t) \tag{5.19}$$

$$u_{d1} = \dot{x}_{d1} \tag{5.20}$$

$$u_{d1} = R\omega\cos(\omega t) \tag{5.21}$$

$$x_{d5} = R(1 - \cos(\omega t)) \tag{5.22}$$

$$x_{d4} = \frac{\dot{x}_{d5}}{u_{d1}} \tag{5.23}$$

$$x_{d4} = \frac{R\omega \sin(\omega t)}{R\omega \cos(\omega t)} \tag{5.24}$$

$$x_{d4} = \tan(\omega t) \tag{5.25}$$

$$u_{d3} = \dot{x}_{d4} \tag{5.26}$$

$$u_{d3} = \omega \sec^2(\omega t) \tag{5.27}$$

$$x_{d3} = \omega t \tag{5.28}$$

$$x_{d2} = \frac{\dot{x}_{d3}}{u_{d1}} \tag{5.29}$$

$$x_{d2} = \frac{\omega}{R\omega \cos(\omega t)} \tag{5.30}$$

$$x_{d2} = \frac{\sec(\omega t)}{R} \tag{5.31}$$

$$u_{d2} = \dot{x}_{d2} \tag{5.32}$$

$$u_{d2} = \frac{\cos(\omega t)\left(\omega \tan(\omega t)\left(\tan^2(\omega t) + 2\right) + 1\right)}{R} \tag{5.33}$$

Ulančana forma glasi:

$$x_{d1} = R \sin(\omega t) \tag{5.34}$$

$$x_{d2} = \frac{\sec(\omega t)}{R} \tag{5.35}$$

$$x_{d3} = \omega t \tag{5.36}$$

$$x_{d4} = \tan(\omega t) \tag{5.37}$$

$$x_{d5} = R(1 - \cos(\omega t)), \tag{5.38}$$

sa ulaznim transformacijama:

$$u_{d1} = R\omega\cos(\omega t) \tag{5.39}$$

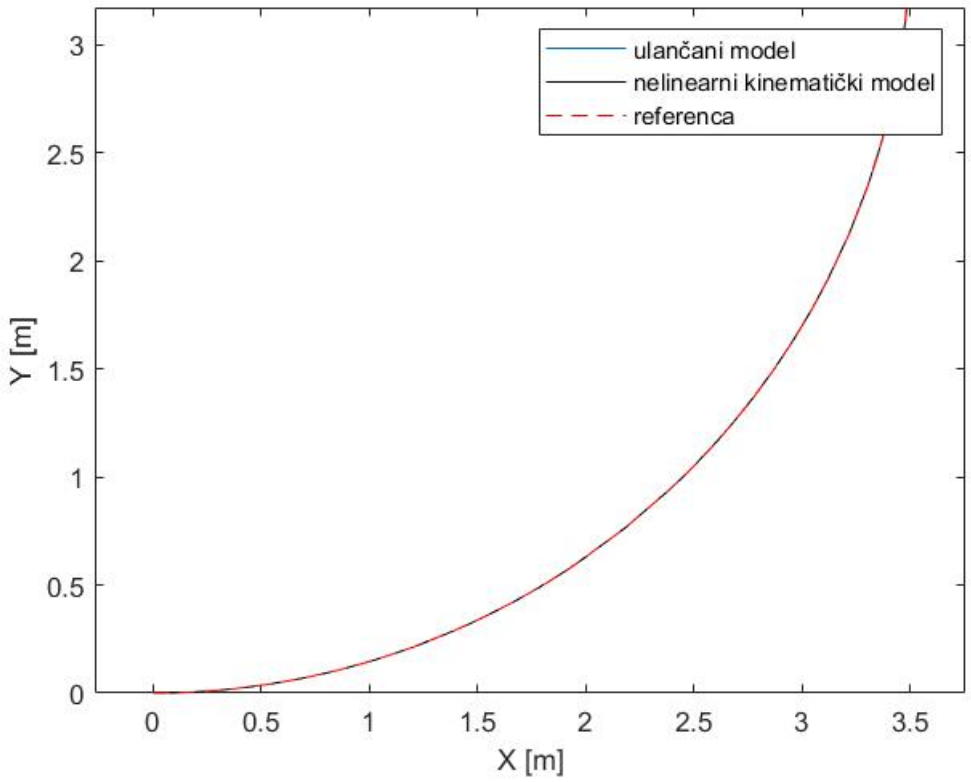
$$u_{d2} = \frac{\cos(\omega t)\left(\omega\tan(\omega t)\left(\tan^2(\omega t)+2\right)+1\right)}{R} \tag{5.40}$$

$$u_{d3} = \omega\sec^2(\omega t). \tag{5.41}$$

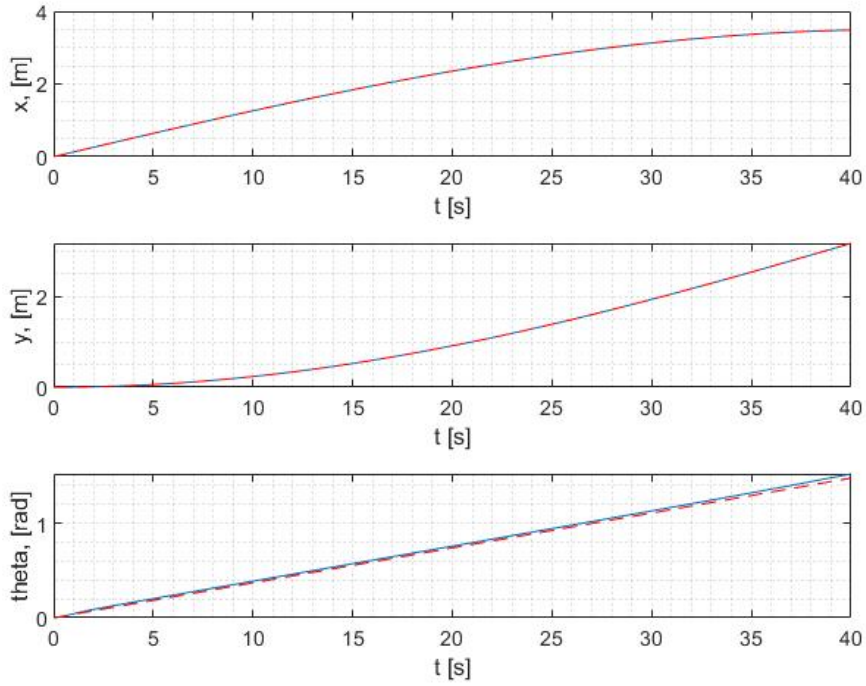
5.3.1 Upravljanje pomoću linearne aproksimacije kružnice

Rezultati S obzirom da je na četvrini putanje kružnice $u_{d1} = 0$, sustav u tom trenu nije upravljiv, pa ćemo promatrati do te točke. Na slikama u nastavku prikazani su rezultati praćenja trajektorije četvrtine kružnice radijusa 3.5 m. Simulacija je napravljena za ulančanu formu i nelinearni model pri čemu su koordinate početne točke jednake ishodištu:

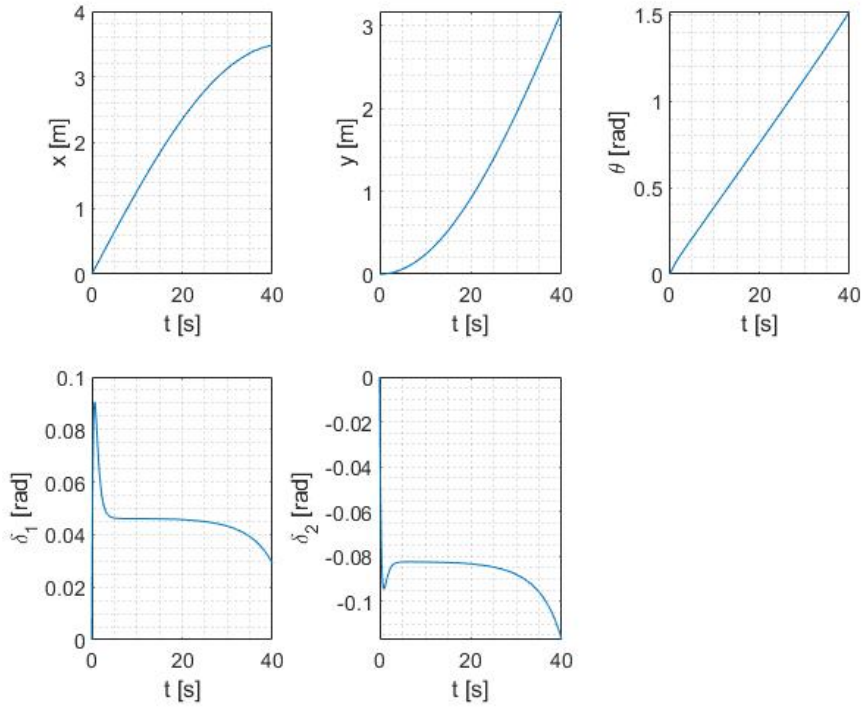
$$x(0) = 0, \qquad y(0) = 0.$$



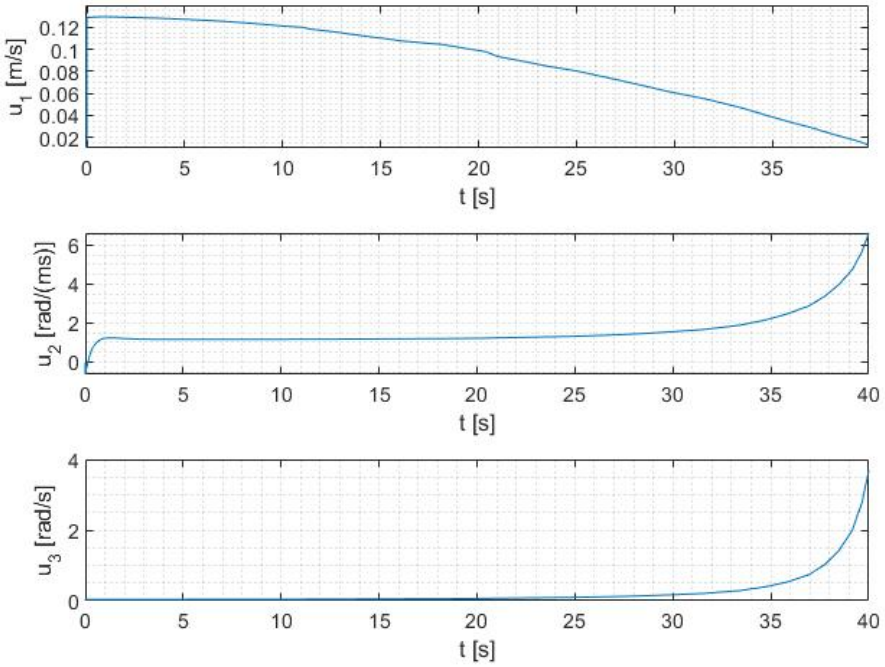
Slika 5.1: Praćenje trajektorije četvrtine kružnice



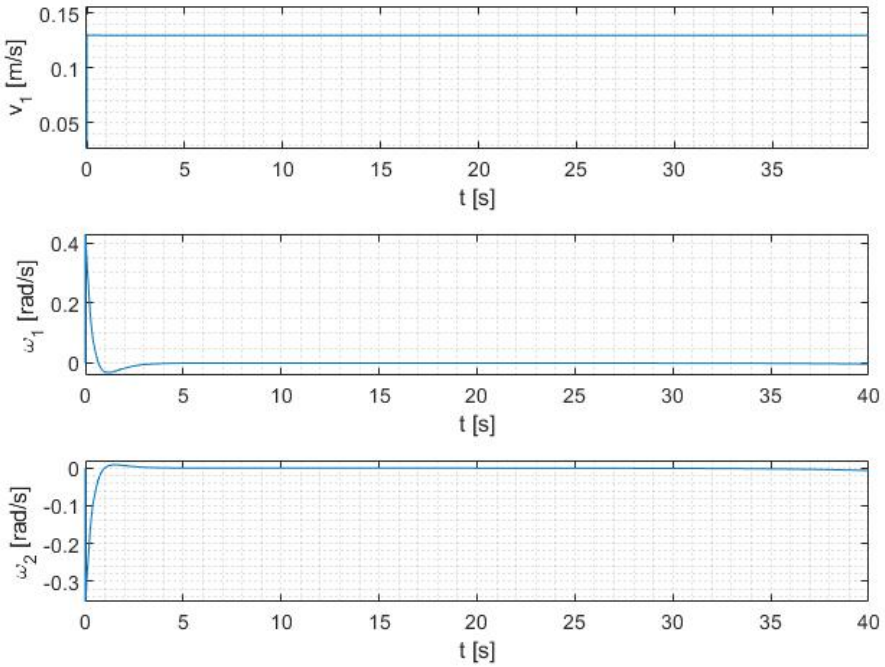
Slika 5.2: Kretanja u smjerovima x,y i kutu zakreta za ulančanu formu trajektorije četvrtine kružnice



Slika 5.3: Kretanja u smjerovima x,y i kutu zakreta za nelinearni model trajektorije četvrtine kružnice



Slika 5.4: Vrijednosti upravljačkih varijabli ulančane forme trajektorije četvrtine kružnice



Slika 5.5: Vrijednosti upravljačkih varijabli nelinearnog modela trajektorije četvrtine kružnice

Po rezutatima iz grafova primjećujemo da robot u simulaciji dobro prati zadanu trajektoriju, te da vrijednosti upravljačkih varijabli u_2 i u_3 naglo rastu pred kraj putanje, to jest kada se približavaju trenutku u kojem sustav nije upravljiv.

5.4 Gaussova krivulja kao referentna trajektorija

Za novu referentnu trajektoriju odabiremo Gaussovu krivulju koja kreće iz točke 0, 0 u kartezijevom sustavu pa su time koordinate definirane slijedećim izrazima:

$$x_d(t) = v_x t, \qquad \theta_d(t) = \arctan\left(\frac{\dot{y_d}(t)}{\dot{x_d}(t)}\right), \qquad y_d(t) = Ye^{-s(x_d - x_c)^2},$$

pri čemu Y predstavlja maksimum u y osi, a x_c vrijednost u x osi pri kojoj dolazi do maksimuma u y. v_x predstavlja brzinu u smjeru x, ali ćemo radi jednostavnsti u jednadžbama stanja koristiti $v_x = v_{max}$, dok će se kod ulaznih varijabli koristiti stvarna željena brzina u smjeru x, dobivena iz:

$$v_{max} = \sqrt{v_x^2 + v_y^2} \tag{5.42}$$

$$v_{max} = \sqrt{\dot{x}_d^2 + \dot{y}_d^2} \tag{5.43}$$

$$v_{max} = \sqrt{\dot{x}_d^2 + (4e^{-2s(x_d - x_c)^2} s^2 Y^2 (x_d - x_c)^2)\dot{x}_d^2)} \tag{5.44}$$

$$v_{max} = \sqrt{\dot{x}_d^2 (4y_d^2 s^2 (x_d - x_c)^2 + 1)} \tag{5.45}$$

$$\dot{x}_d = \frac{v_{max}}{\sqrt{4y_d^2 s^2 (x_d - x_c)^2 + 1}} = v_x. \tag{5.46}$$

Slijedi transformacija u ulančanu formu uvođenjem supstitucije:

$$x_{d1} = x_d(t), \qquad x_{d3} = \theta_d(t), \qquad x_{d5} = y_d(t).$$

Postupak iz **5.1** ponavljamo sa prethodno spomenutom supstitucijom, pa dobivamo ulančanu formu Gaussove trajektorije:

$$x_{d1} = v_x t \tag{5.47}$$

$$x_{d2} = \frac{2sYe^{s(x_c - v_x t)^2}\left(2s(x_c - v_x t)^2 - 1\right)}{4s^2Y^2(x_c - v_x t)^2 + e^{2s(x_c - v_x t)^2}} \tag{5.48}$$

$$x_{d3} = \tan^{-1}\left(2sY(x_c - v_x t)e^{-s(x_c - v_x t)^2}\right) \tag{5.49}$$

$$x_{d4} = -2sY(x_c - v_x t)e^{-s(x_c - v_x t)^2} \tag{5.50}$$

$$x_{d5} = Ye^{-s(v_x t - x_c)^2} \tag{5.51}$$

sa ulaznim transformacijama:

$$u_{d1} = v_x \tag{5.52}$$

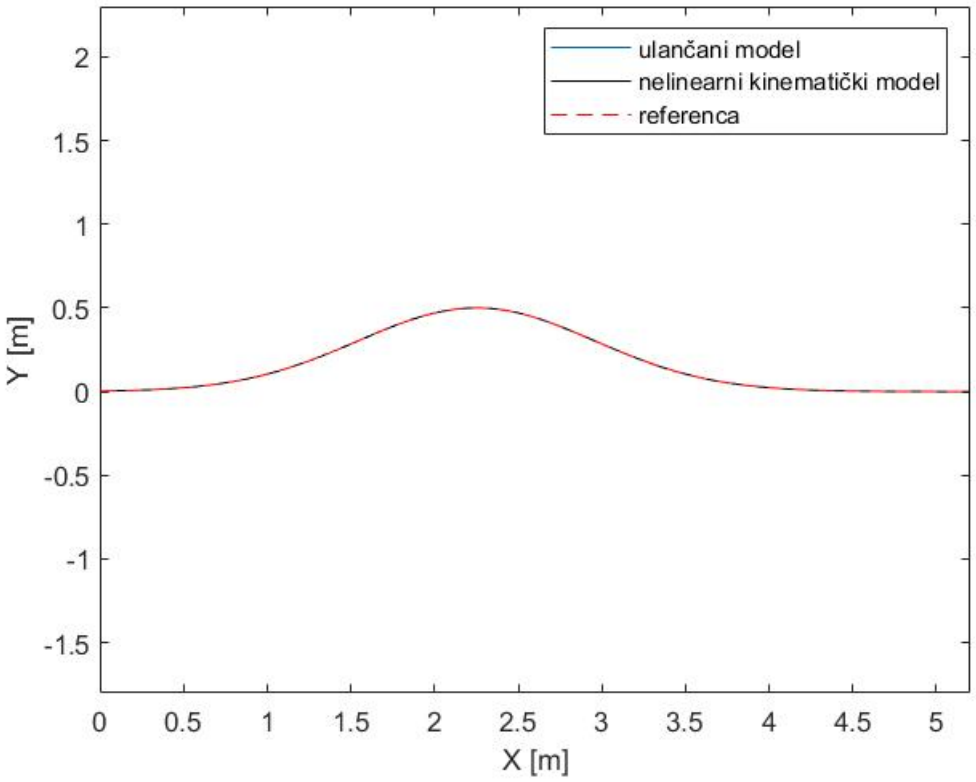
$$u_{d2} = -\frac{4s^2 v_x Y (v_x t - x_c) e^{s(x_c - v_x t)^2} e^{2s(x_c - v_x t)^2}}{\left(4s^2 Y^2 (x_c - v_x t)^2 + e^{2s(x_c - v_x t)^2}\right)^2} \cdot \left(2s(x_c - v_x t)^2 - 3\right) - 4sY^2 \left(s(x_c - v_x t)^2 \left(2s(x_c - v_x t)^2 - 1\right) + 1\right) \tag{5.53}$$

$$u_{d3} = 2sv_x Y e^{-s(x_c - v_x t)^2} \left(2s(x_c - v_x t)^2 - 1\right) \cdot \tag{5.54}$$

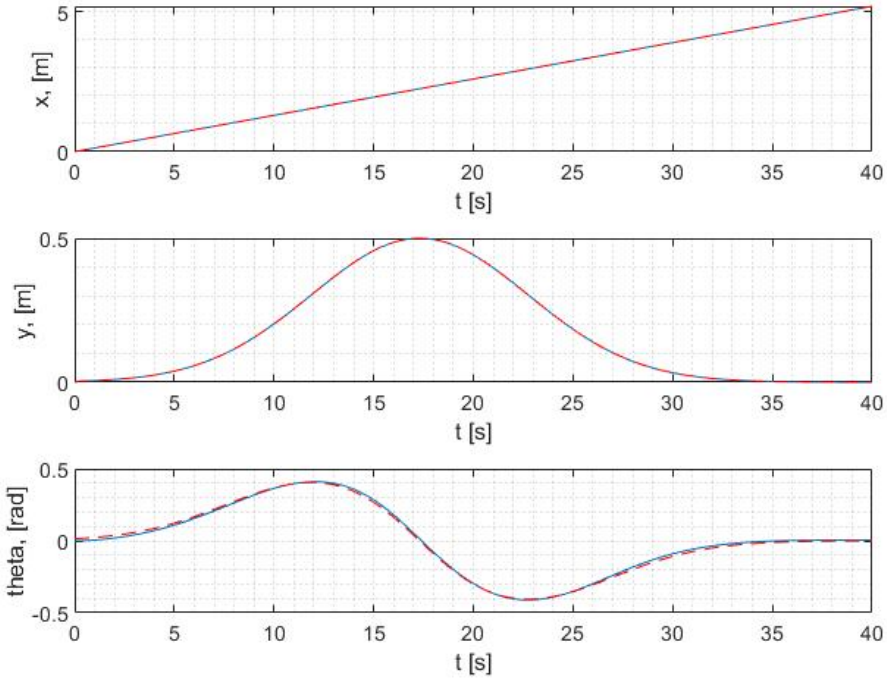
5.4.1 Upravljanje pomoću linearne aproksimacije Gaussove krivulje

Analogno zakonu upravljanja iz poglavlja 5.3.1 simulira se praćenje trajektorije za vrijednosti $x_c = 2.25$ m i $Y = 0.5$ m uz koeficijent $s = 1$ i maksimalnu brzinu $v_{max} = 0.13$ m/s. Simulacija je napravljena za ulančanu formu i nelinearni model pri čemu su koordinate početne točke:

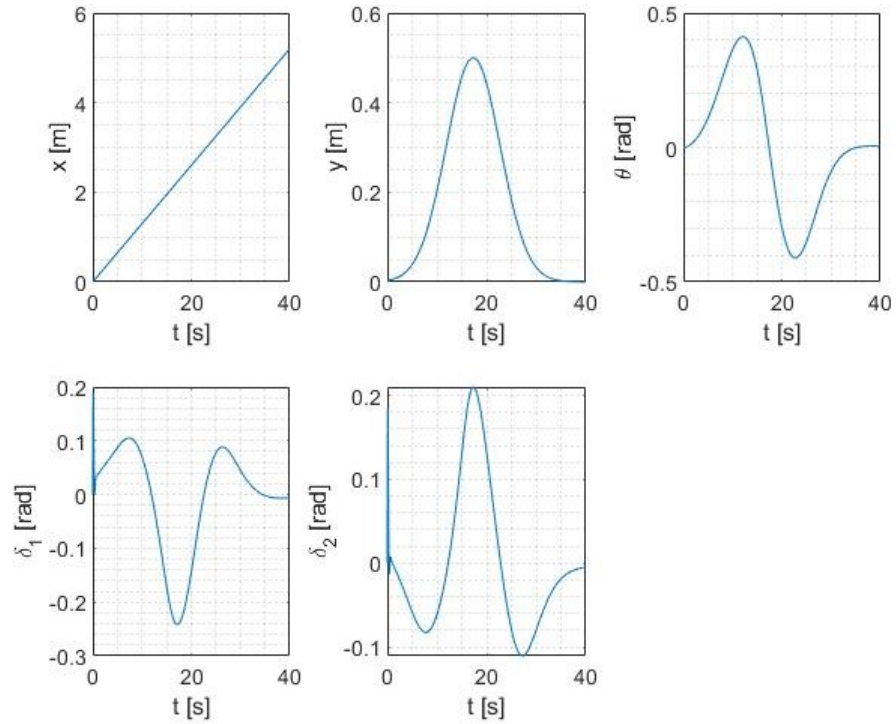
$$x(0) = 0, \qquad y(0) = 0.$$



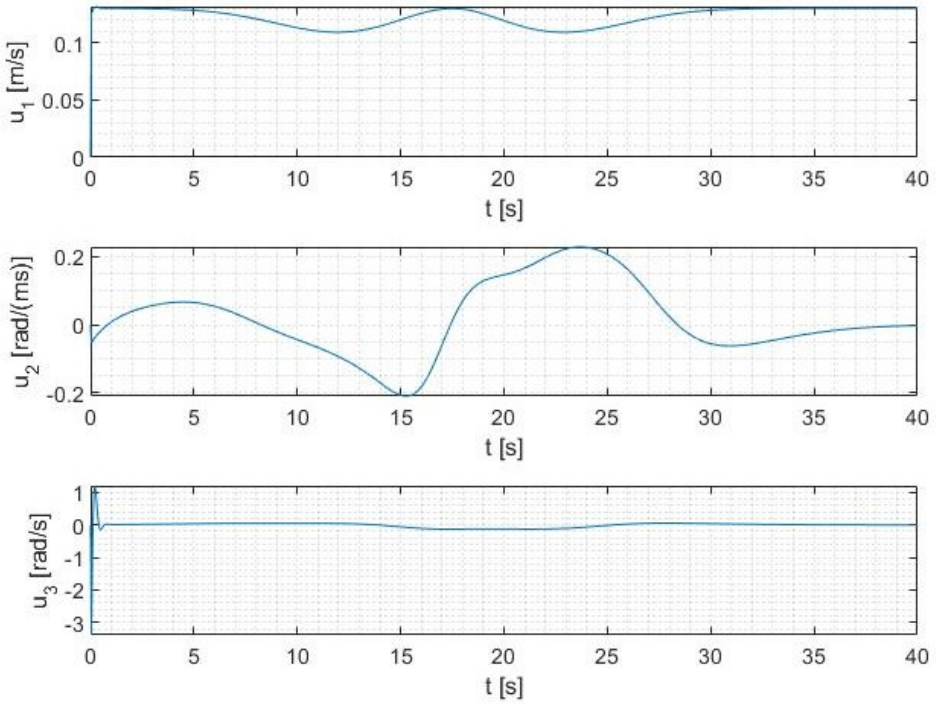
Slika 5.6: Praćenje trajektorije Gaussove krivulje



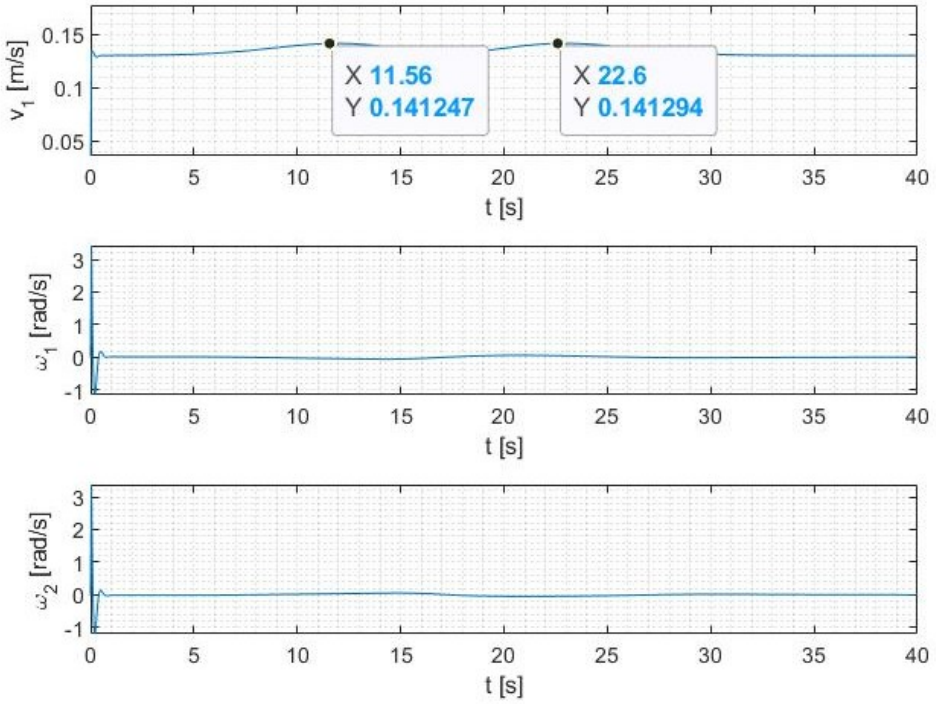
Slika 5.7: Kretanja u smjerovima x,y i kutu zakreta za ulančanu formu trajektorije Gaussove krivulje



Slika 5.8: Kretanja u smjerovima x,y i kutu zakreta za nelinearni model trajektorije Gaussove krivulje



Slika 5.9: Vrijednosti upravljačkih varijabli ulančane forme trajektorije Gaussove krivulje

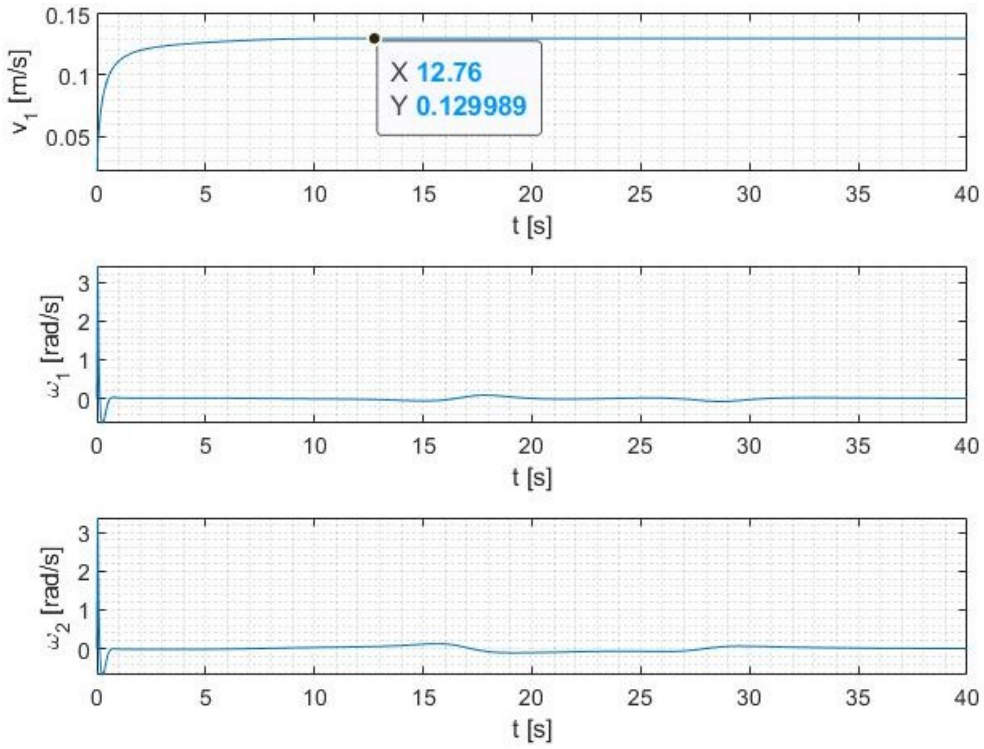


Slika 5.10: Vrijednosti upravljačkih varijabli nelinearnog modela trajektorije Gaussove krivulje

Vidljivo je da robot dobro prati putanju Gaussove krivulje, no primjćujemo u grafu sa slike 5.10 da vrijednosti v_1, prelaze maksimalnu brzinu od 0.13 m/s. Ograničavanjem brzine v_1 pomoću funkcije tangens hiperbolni;

$$v_1 = 0.13 \tanh(2.5 v_1)$$

dobivamo nove vrijednosti upravljačkih varijabli prikazane na slici 5.11, dok ne dolazi do značajnih promjena kod ostalih veličina.



Slika 5.11: Vrijednosti upravljačkih varijabli nelinearnog modela trajektorije Gaussove krivulje nakon ograničenja brzine

5.5 Sinusoida kao referentna trajektorija

Kao treća referentna trajektorija odabrana je sinusoida koja kreće iz točke 0, 0 u kartezijevom sustavu pa su koordinate definirane izrazima:

$$x_d(t) = v_x t, \qquad \theta_d(t) = \arctan\left(\frac{\dot{y_d}(t)}{\dot{x_d}(t)}\right), \qquad y_d(t) = A\sin(\omega t),$$

pri čemu je A amplituda sinusoide, a ω period. v_x predstavlja brzinu u smjeru x, ali ćemo radi jednostavnosti u jednadžbama stanja koristiti $v_x = v_{max}$, dok će se kod ulaznih varijabli koristiti stvarna željena brzina u smjeru x, dobivena iz:

$$v_x = v_{max}\cos(\theta_d) \tag{5.55}$$

Ponovno se uvodi supstitucija za transformaciju sustava:

$$x_{d1} = x_d(t), \qquad x_{d3} = \theta_d(t), \qquad x_{d5} = y_d(t),$$

Postupak iz 5.1 se ponavlja, te se dobiva ulančana forma sinusoide:

$$x_{d1} = v_x t \tag{5.56}$$

$$x_{d2} = -\frac{A\omega^2\sin(\omega t)}{A^2\omega^2\cos^2(\omega t) + v_x^2} \tag{5.57}$$

$$x_{d3} = \tan^{-1}\left(\frac{A\omega\cos(\omega t)}{v_x}\right) \tag{5.58}$$

$$x_{d4} = \frac{A\omega\cos(\omega t)}{v_x} \tag{5.59}$$

$$x_{d5} = A\sin(\omega t). \tag{5.60}$$

Uvrštavanjem 5.58 u 5.55 dobiva se izraz za stvarnu željenu brzinu u smjeru x:

$$v_x = v_{max}\cos\left(\tan^{-1}\left(\frac{A\omega\cos(\omega t)}{v_x}\right)\right). \tag{5.61}$$

Ulazne transformacije su slijedeće:

$$u_{d1} = v_x \tag{5.62}$$

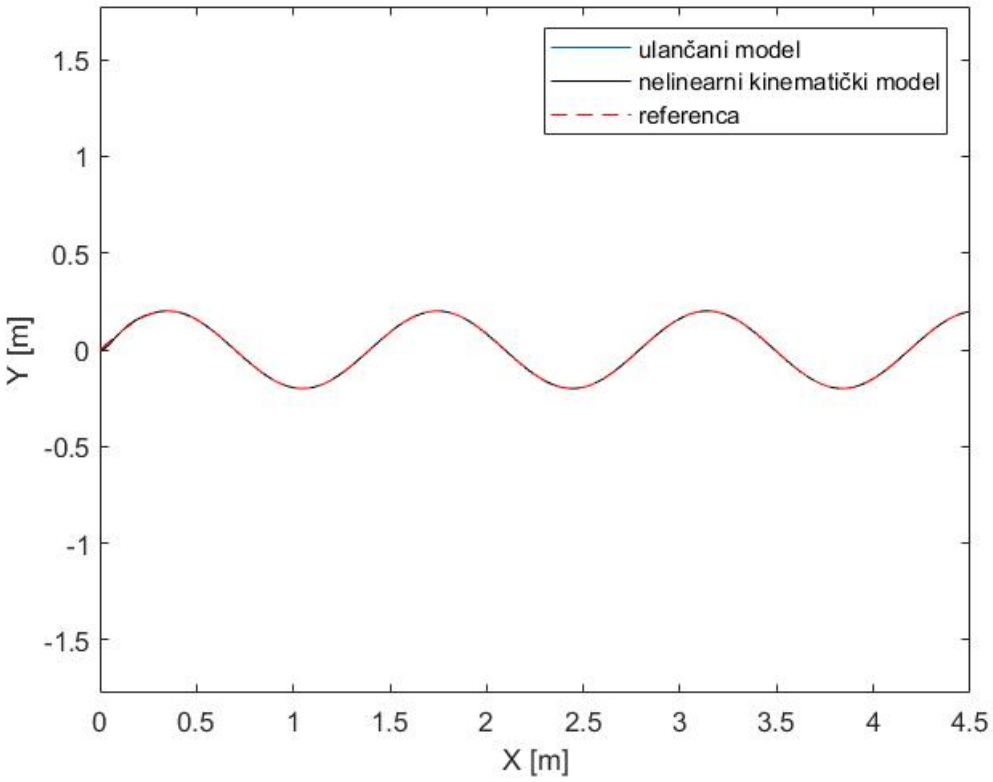
$$u_{d2} = \frac{A\omega^3 \cos(\omega t)\left(A^2\omega^2\cos(2\omega t) - 3A^2\omega^2 - 2v_x^2\right)}{2\left(A^2\omega^2\cos^2(\omega t) + v_x^2\right)^2} \tag{5.63}$$

$$u_{d3} = -\frac{A\omega^2 \sin(\omega t)}{v_x}. \tag{5.64}$$

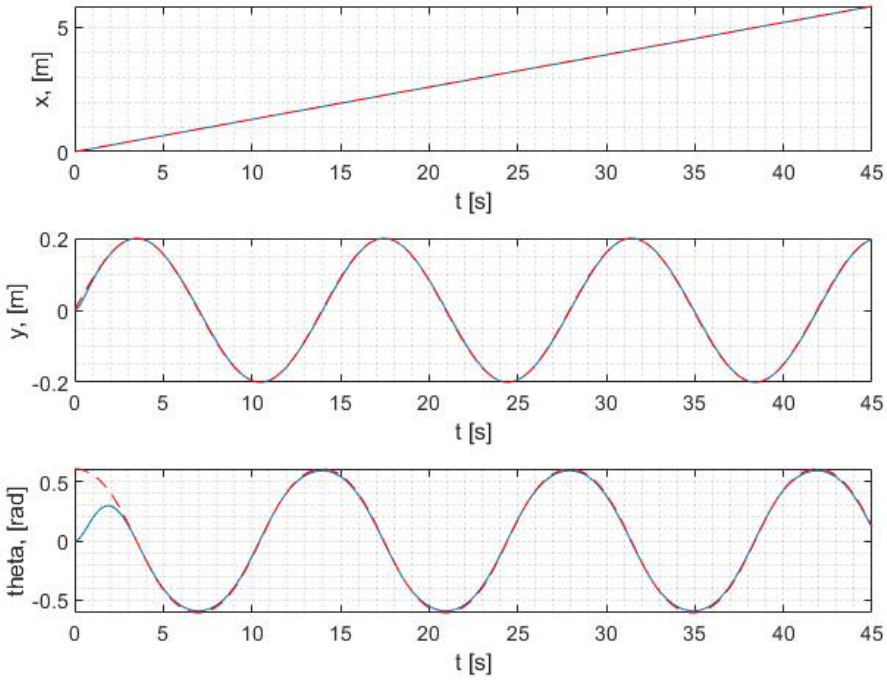
5.5.1 Upravljanje pomoću linearne aproksimacije sinusoide

Analogno zakonu upravljanja iz poglavlja **5.3.1** i **5.4.1** simulira se praćenje trajektorije za sinusoidu amplitude $A = 0.2$ m i period $\omega = 0.45$ i maksimalnu brzinu $v_{max} = 0.13$ m/s. Simulacija je napravljena za ulančanu formu i nelinearni model početnim koordinatama u ishodištu:

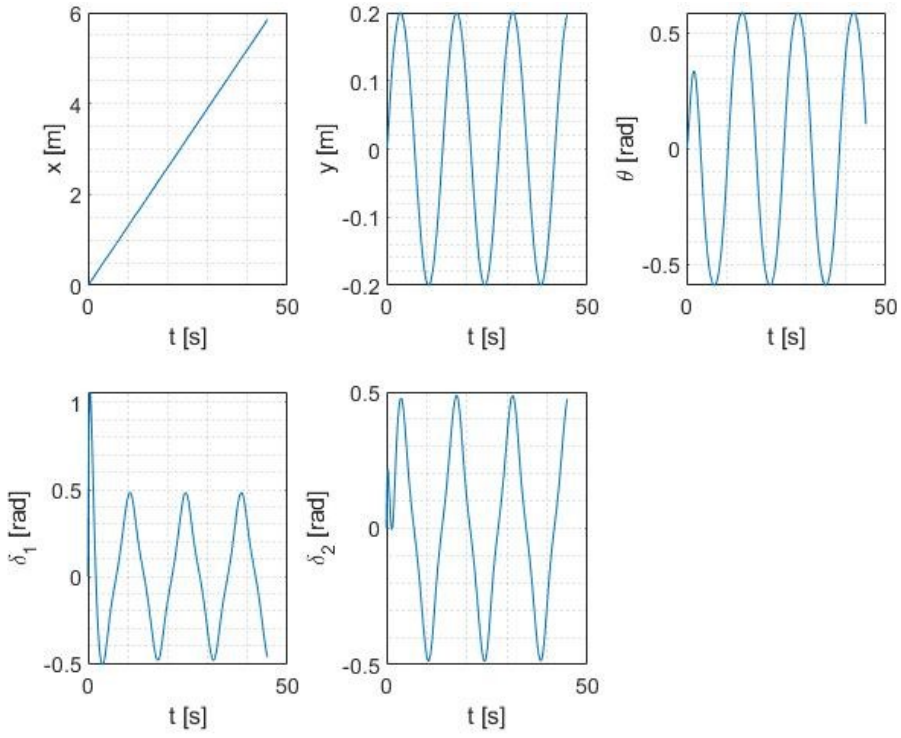
$$x(0) = 0, \qquad y(0) = 0.$$



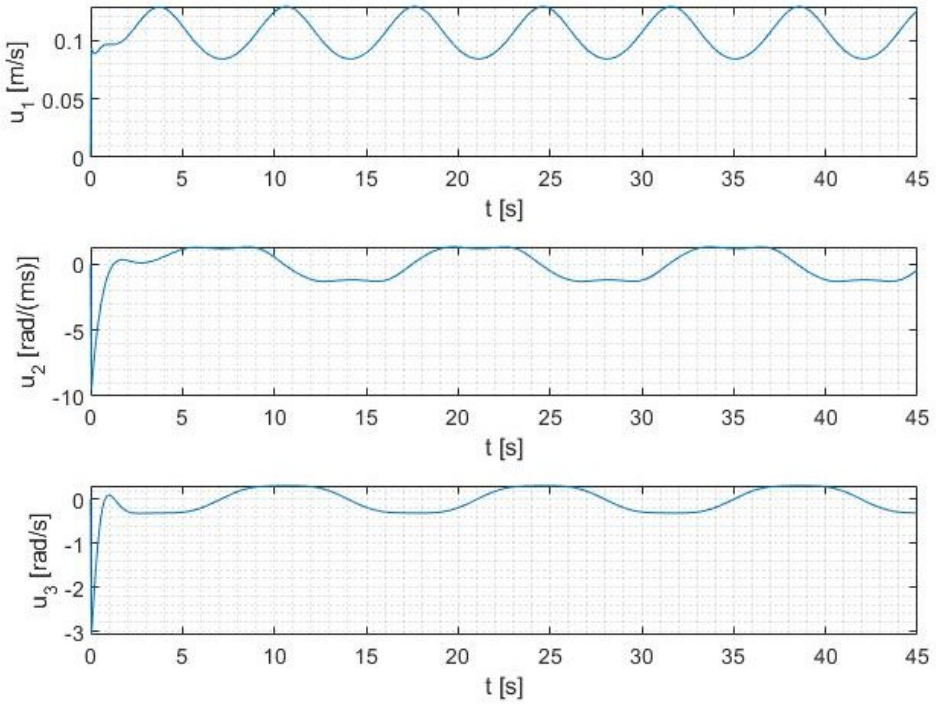
Slika 5.12: Praćenje sinusoidne trajektorije



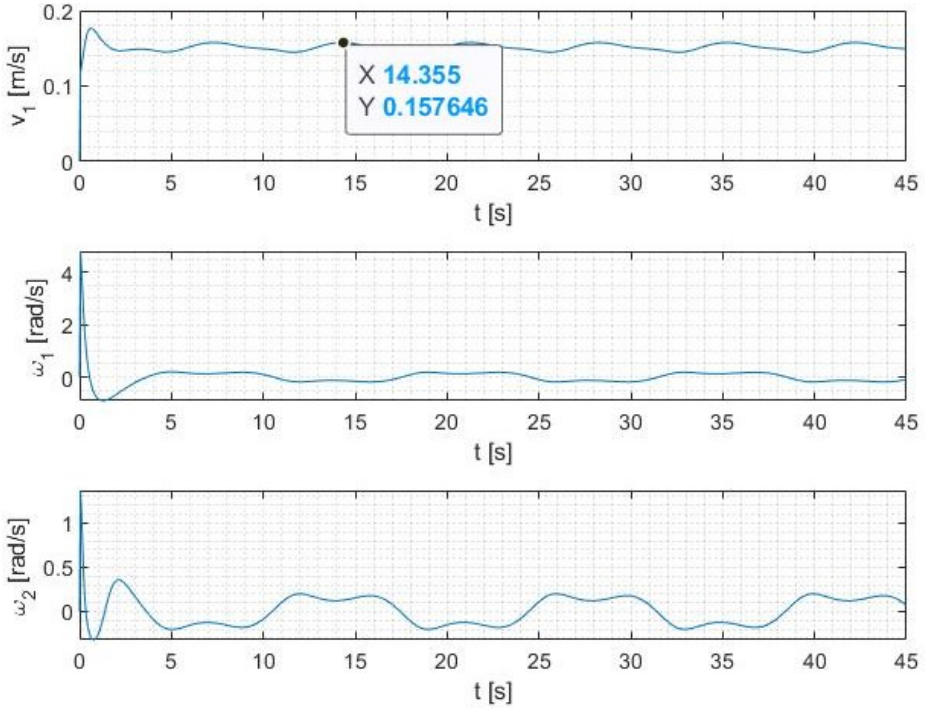
Slika 5.13: Kretanja u smjerovima x,y i kutu zakreta za ulančanu formu sinusoidne trajektorije



Slika 5.14: Kretanja u smjerovima x,y i kutu zakreta za nelinearni model sinusoidne trajektorije



Slika 5.15: Vrijednosti upravljačkih varijabli ulančane forme sinusoidne trajektorije

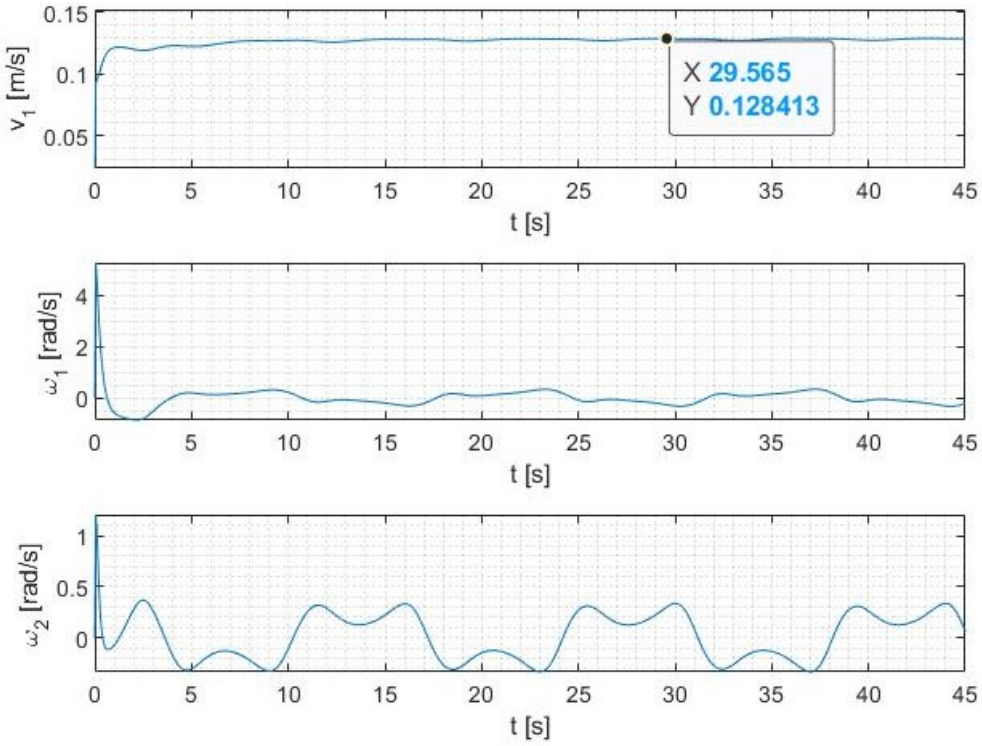


Slika 5.16: Vrijednosti upravljačkih varijabli nelinearnog modela sinusoidne trajektorije

Kao i kod Gaussove krivulje, u simulaciji robot dobro prati putanju, ali vrijednosti v_1 ponovno prelaze maksimalnu brzinu. Ograničenje pomoću tangensa hiperbolnog u ovom slučaju nije dalo dobre rezultate, pa je nominalna brzina smanjena na 0.1 m/s, te se uvodi sigmoidna funkcija koja bi osigurala da maksimalna brzina ne prelazi 0.13 m/s;

$$v_1 = \frac{0.13}{(1 + \mathrm{e}^{-10v_1})}.$$

Dobivene nove vrijednosti upravljačkih varijabli prikazane su na slici 5.17, dok su promjene kod ostalih veličina zanemarive.



Slika 5.17: Vrijednosti upravljačkih varijabli nelinearnog modela sinusoidne trajektorije nakon ograničenja brzine

6 Robotski operativni sustav

Robotski operativni sustav (ROS) je softverski okvir otvorenog koda, inicijalno namjenjen za Linux, a dizajniran je za omogućavanje robotima izvedbu raznih funkcionalnosti. Razvio se iz robotskog sustava Switchyard, kojeg je 2007. godine kreirao Morgan Quigley. ROS služi kao univerzalna softverska platforma za programere i korisnike robota. Ova zajednička platforma poboljšava suradnju omogućujući lakše dijeljenje koda i ideja. Važno je istaknuti da eliminira potrebu za opsežnim razvojem softverske infrastrukture prije aktiviranja robota. S ROS-om je značajno smanjena potreba za razvojem svega iz temelja, osobito za one koji rade s robotima podržanim od strane ROS-a. To omogućuje veći fokus na šire aspekte robotike umjesto na detalje niže razine programiranja i razvoj upravljačkih programa.

ROS (Robot Operating System) je sveobuhvatan okvir koji se sastoji od nekoliko ključnih komponenata za razvoj robotike [15]:

1. skup upravljačkih programa za komunikaciju sa senzorima i aktuatorima, koji podržavaju širok spektar hardvera i komercijalnih robotskih sustava,
2. kolekcija osnovnih algoritama robotike za zadatke poput mapiranja, navigacije, interpretacije podataka senzora, planiranja kretanja i manipulacije objektima,
3. računalna infrastruktura za upravljanje podacima, povezivanje različitih komponenata složenih robotskih sustava i integraciju prilagođenih algoritama,
4. robusni skup alata za vizualizaciju stanja robota i algoritama, otklanjanje pogrešaka i snimanje podataka senzora,
5. širok raspon resursa, uključujući wiki za dokumentaciju mnogih aspekata okvira, web-mjesto za pitanja i odgovore za podršku zajednice, te živuću mrežu korisnika i programera.

6.1 Struktura ROS sustava

ROS je strukturno podijeljen na tri cjeline, odnosno tri razine. Prva razina predstavlja razinu sustava datoteka. Na ovoj razini definiraju se struktura direktorija i datoteke potrebne za funkcioniranje sustava. Druga razina je razina povezivanja procesa na kojoj se odvija komunikacija između procesa, odnosno čvorova i povezanih sustava. Treća razina

je razina zajednice. Ova razina presudna je za funkcioniranje ROS-a jer omogućuje njegovo brzo širenje i razvoj [1].

6.1.1 Razina sustava datoteka

Koncepti na razini sustava datoteka (eng. Filesystem level) u ROS-u uglavnom obuhvaćaju razne resurse koji se nalaze na disku, a navedeni su u nastavku teksta [1].

1. Packages (paketi), su najosnovniji elementi za izgradnju i objavljivanje, tj osnovne jedinice za organizaciju softvera u ROS-u. Paket može uključivati ROS procese izvođenja (nodes), knjižnice, skupove podataka, konfiguracijske datoteke ili bilo koje druge povezane stavke.

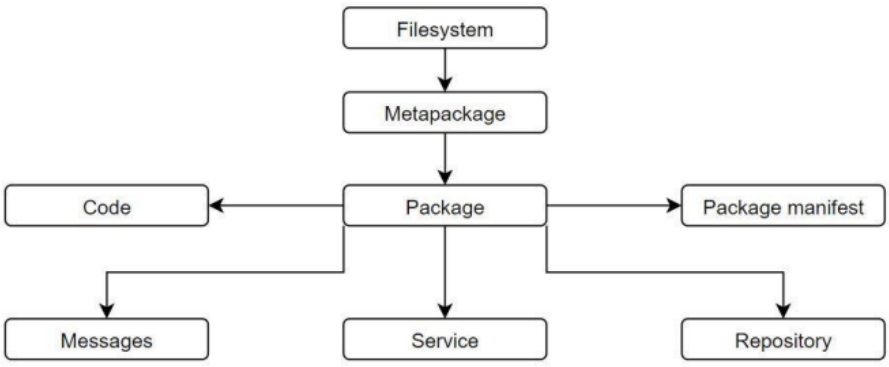
2. Metapackages (metapaketi) su specijalizirani paketi koji služe za predstavljanje grupe povezanih paketa.

3. Package Manifests (manifesti paketa) pružaju bitne metapodatke o paketu, kao što su njegovo ime, verzija, opis, informacije o licenci, ovisnosti i druge pojedinosti poput izvezenih paketa.

4. Repositories (repozitoriji) su su skupovi paketa koji dijele zajednički sustav za kontrolu verzija (eng. Version Control System - VCS).

5. Messages (poruke) definiraju strukture podataka koje se prenose u ROS-u. Spremaju se u formatu "mypackage/msg/MyMessageType.msg" .

6. Services (usluge), pohranjeni u formatu "mypackage/srv/MyServiceType.srv" opisuju strukture podataka za zahtjeve i odgovore usluga.



Slika 6.1: Razina sustava datoteka [16]

[1] ROS; https://wiki.ros.org/

6.1.2 Razvojna razina

Razvojna razina (eng. Computation level) ROS-a je partnerska mreža različitih čvorova koji surađuju u obradi podataka. Svaki umreženi čvor može pristupiti mreži, te ostvariti međusobnu interakciju s drugim čvorovima pomoću koje može slati informacije i vidjeti informacije koje drugi čvorovi šalju.

1. Nodes (čvorovi) su osovni procesi koji obavljaju određene radnje. Modularni dizajn ROS-a uključuje više čvorova za različite funkcije, kao što su kontrola motora, obavljanje lokalizacije ili pružanje grafičkih prikaza.

2. Master locira i omogućuje komunikaciju između čvorova, poruka i servisa, te nadzire njihov rad. Ključan je za održavanje informacija o registraciji za teme i usluge, omogućujući čvorovima da dinamički uspostavljaju veze. Čvorovi se direktno povezuju, a Master pruža bitne informacije za pretraživanje, slično DNS poslužitelju.

3. Parameters Server (Parametarski server) pohranjuje podatke na središnjem mjestu, što čvorovima omogućuje pristup serveru i njegovima parametrima.

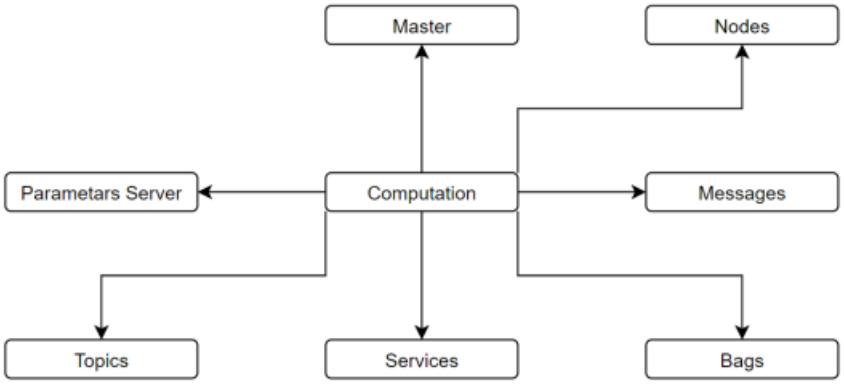
4. Topics (teme) su putevi za poruke temeljene na sustavu publisher/subscriber (izdavač/pretplatnik) gdje publisher predstavlja čvor koji objavljuje poruku na temu, a subscriber se na nju pretplaćuje. Teme omogućuju anonimnu i odvojenu interakciju između izdavača i pretplatnika.

5. Bags (spremnici) se koriste za pohranu i ponovno reproduciranje podataka ROS poruka, posebno korisne za spremanje podataka senzora i drugih informacija ključnih za razvoj i testiranje algoritama.

Ova arhitektura podržava odvojeni rad, gdje su imena glavno sredstvo za izgradnju većih i složenijih sustava. Imena za čvorove, teme, usluge i parametre su ključna, a biblioteke klijenata ROS-a podržavaju preusmjeravanje imena putem naredbenog retka za fleksibilno rekonfiguriranje u različitim topologijama razvojnog grafa.

6.1.3 Razina zajednice (Community level)

Koncepti na razini ROS zajednice (eng. Community level) obuhvaćaju resurse koji olakšavaju razmjenu softvera i znanja unutar različitih ROS zajednica. Ovi resursi uključuju distribucije, repozitorije, Wiki, mailing liste, ROS Answers, blog i sustav za prijavu grešaka.



Slika 6.2: Razvojna razina [16]

1. ROS distribucije su zbirke softverskih sklopova dostupnih za instalaciju. One pojednostavljuju proces instalacije softvera i održavaju konzistentne verzije kroz različite softverske pakete.

2. ROS koristi federiranu mrežu kodnih repozitorija, omogućujući različitim institucijama da neovisno razvijaju i objavljuju svoje komponente softvera za robote.

3. ROS Wiki je glavna platforma za dokumentiranje informacija o ROS-u. To je suradnički prostor gdje svatko može doprinijeti dokumentaciji, napraviti ispravke ili ažuriranja, pisati tutorijale i slično.

4. Sustav za prijavu grešaka zvan Tickets, koristi se za prijavljivanje i praćenje softverskih grešaka.

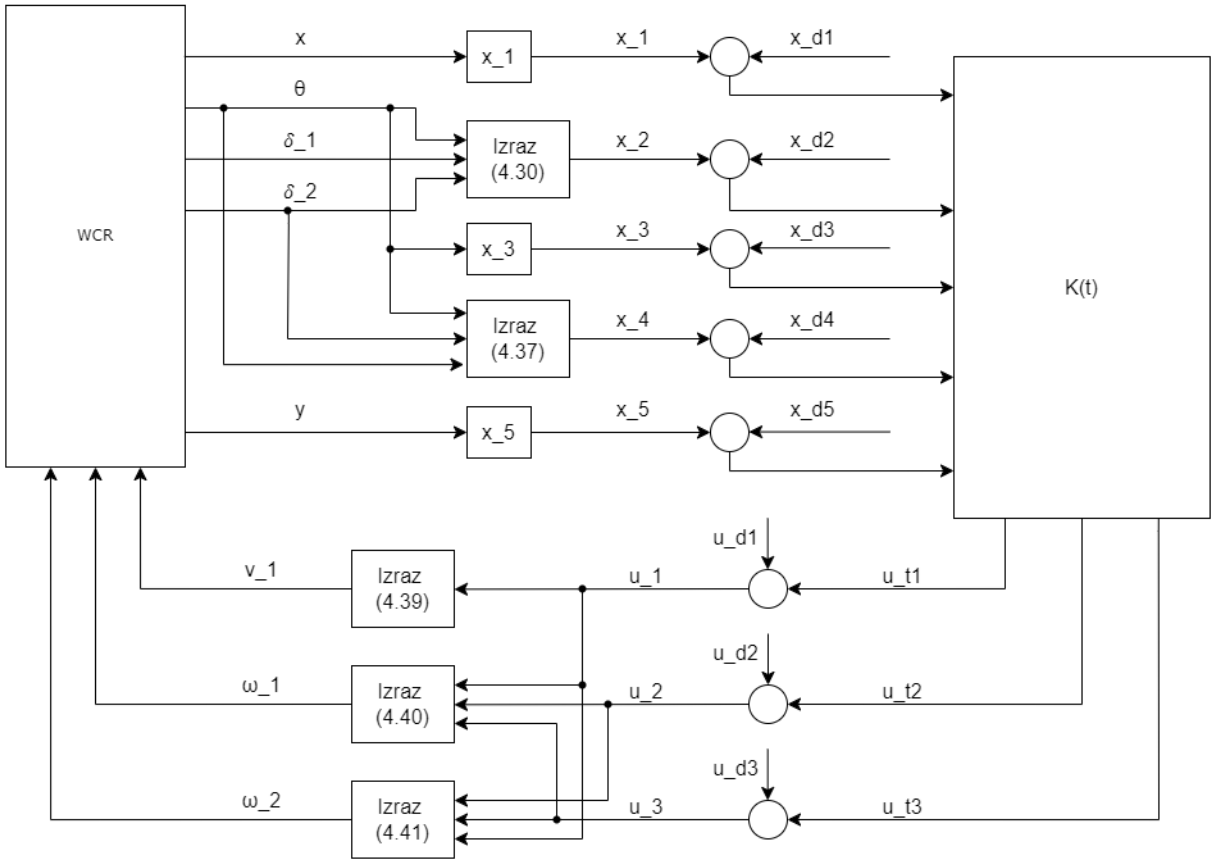
5. Mailing lista ros-users služi kao glavni komunikacijski kanal za nova ažuriranja ROS-a i forum za postavljanje pitanja o ROS softveru.

6. ROS Answers je stranica s pitanjima i odgovorima posvećena rješavanju upita vezanih za ROS.

7. Blog ros.org pruža redovite ažuriranja, uključujući fotografije i videozapise.

7 Implementacija pomoću sustava ROS i mjerenje OptiTrackom

Odabrana verzija ROS-a je ROS Noetic Ninjemys, koji je namijenjen za operativni sustav Linux Ubuntu 20.04. Programski kod pisan je u editoru Visual Studio Code u programskom jeziku Python 3. Rezultati su mjereni OptiTrack sustavom u laboratoriju CRTA-e. Na slici 7.1 je prikazan blok-dijagram upravljanja s povratnom vezom koji je bilo potrebno implementirati na robota.



Slika 7.1: Blok-dijagram upravljanja s povratnom vezom

7.1 Povezivanje i upravljanje robotom

Prije povezivanja s robotom bilo je potrebno napraviti radni prostor unutar željene mape (u ovom slučaju catkin_wcr) naredbom:

```
$ catkin_make
```

koja, ukratko, sastavlja i povezuje kod napisan za ROS čvorove, usluge i ostale komponente. Zatim je bilo potrebno provesti naredbu:

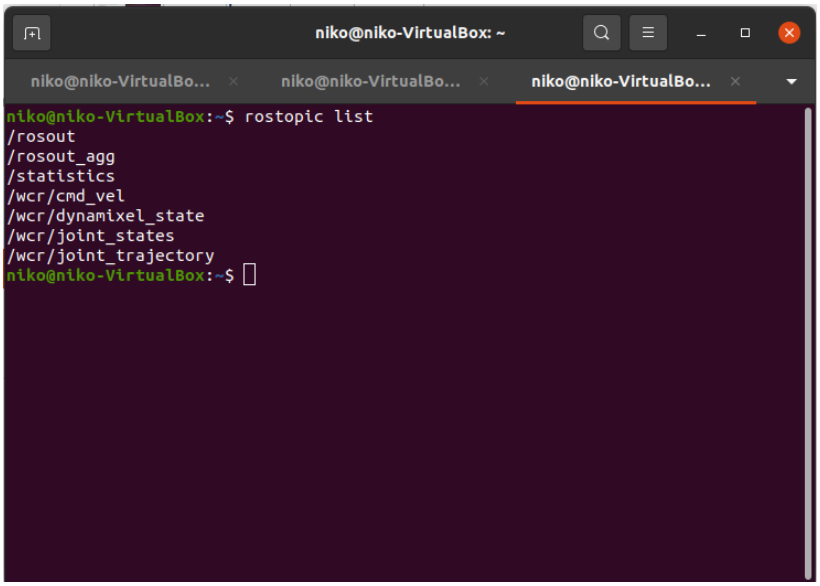
```
$ source ~/catkin_wcr/devel/setup.bash
```

i kopirati ju u .bashrc. Master čvor se pokreće na robotu, te se pokretanjem može vidjeti IP adresa robota. Za povezivanje je potrebno spojiti se na istu WiFi mrežu koju koristi robot te u .bashrc dodati robotovu IP adresu na slijedeći način:

```
export ROS_MASTER_URI=http://192.168.1.156:11311/
export ROS_HOSTNAME=192.168.1.9
export ROS_IP=192.168.1.9                              .
```

Na slici 7.2 prikazane su sve teme (Topics) naredbom:

```
$ rostopic list
```



Slika 7.2: Rostopic list

Ukoliko su potrebne poruke (Messages) objavljene na određenoj temi koristi se naredba:

```
$ rostopic echo /topic_name
```

gdje `/topic_name` predstavlja ime željene teme. Od navedenih tema izvojit ćemo one korištene za upravljanje robotom.

- `wcr/cmd_vel` je tema na koju se robot pretplaćuje, pa se na nju objavljuju informacije o željenom kretanju robota. Ona sadrži informacije o brzinama, točnije šest informacija u svakom diskretiziranom trenutku, po jedna za linearnu i kutnu brzinu u smjeru svake osi Kartezijevog koordinatnog sustava.

- `wcr/joint_states` je tema na koju robot objavljuje informacije o odometriji robota. Tu se nalazi osam veličina koje predstavljaju poziciju i brzinu svakog od četiri kotača. Na ovu se temu pretplaćujemo, te dobivene vrijednosti koristimo za izračun željenih brzina kretanja.

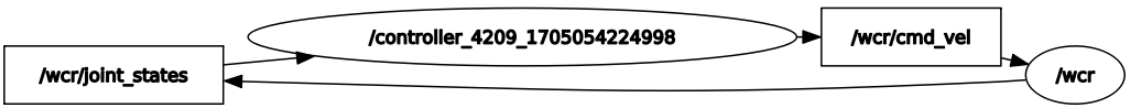
Upravljački čvor je izrađen u programskom jeziku Python pod nazivom `wcr_ns_controller.py` i prikazan je u predlošku. Prvo su definirane biblioteke koje se koriste za pisanje programskog jezika. Korišteno je sedam biblioteka; `rospy` omogućuje korištenje ROS sustava, `math` služi za korištenje matematičkih operacija, `numpy` za matrično računanje, `pandas` za čitanje .txt dokumenata, te su iz poruka `std_msgs.msg`, `sensor_msgs.msg` i `geometry_msgs.msg` uvedeni, redom; `String`, `JointState` i `Twist`. `String` omogućuje objavljivanje i čitanje poruka tipa string, `JointState` će se koristiti za čitanje podataka o odometriji, a `Twist` za objavljivanje željenih veličina (brzina kretanja).

Upravljački se čvor pretplaćuje na temu `wcr/joint_states`, gdje iz funkcije `jointStateCallback()`, koja kao ulazni argument učitava podatke sa senzora, izravno dobiva vrijednosti δ_1 i δ_2, te računa brzine u smjerovima x i y, te brzinu zakreta oko osi z prema izrazu 4.5. Te se vrijednosti, zatim, u funkciji `controller()` koriste prema blok-dijagramu sa slike 7.1 kako bi se izračunale željene upravljačke vrijednosti, koje se objavljuju na temu `wcr/cmd_vel`. U funkciji `main()` se poziva funkcija `controller()` čime se pokreće upravljački čvor, a to se vrši u naredbom u terminalu:

```
$ rosrun wcr_controller wcr_ns_controller.py
```

Na slici 7.3 može se vidjeti grafički prikaz čvorova koji su pokrenuti i tema na koje objavljuju ili se pretplaćuju, a generiran je naredbom:

```
$ rqt_graph
```

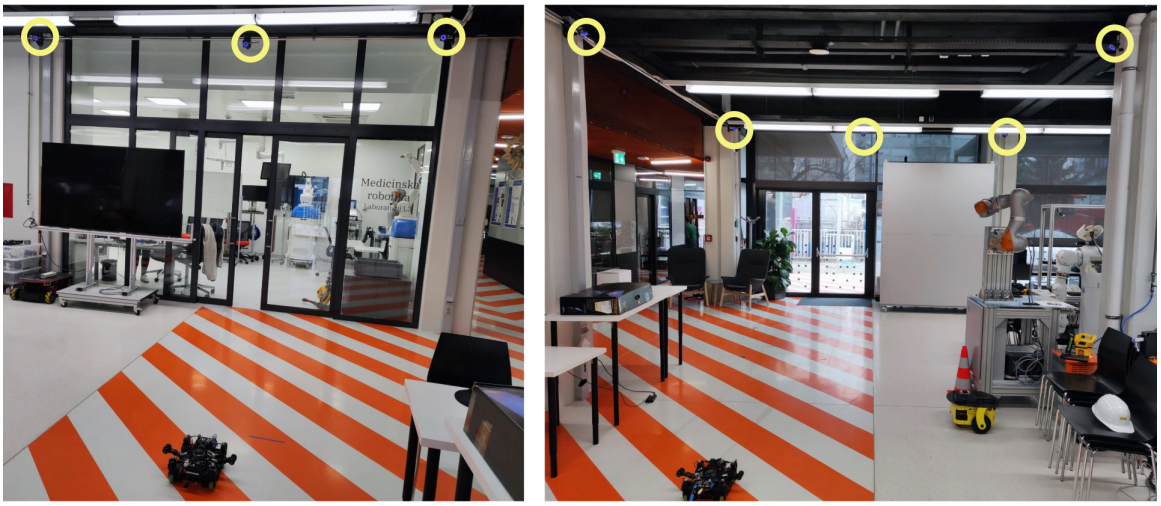


Slika 7.3: Prikaz čvorova za upravljanje mobilnim robotom

7.2 Konfiguracija OptiTracka

OptiTrack je sustav za snimanje i praćenje pokreta koji se široko koristi u raznim područjima kao što su animacija, virtualna stvarnost, robotika, biomehanika i sport. OptiTrack sustavi poznati su po visokoj razini točnosti i malom kašnjenju u praćenju kretanja. Koristi specijalizirane kamere i reflektirajuće markere za praćenje kretanja objekata ili ljudi u 3D prostoru. Sustav se sastoji od više infracrvenih kamera koje bilježe položaj reflektirajućih markera postavljenih na objekt ili osobu koja se prati. Ove su kamere sinkronizirane kako bi stvorile kohezivan i točan prikaz pokreta, koji se mogu precizno prevesti u virtualno okruženje. U robotici, OptiTrack se koristi za prostorna mjerenja, navigaciju robota i kontrolu robotskih pokreta s visokom preciznošću, te je kompatibilan sa nizom sustava, uključujući ROS. Sustav se može skalirati na male i velike prostore, prilagođavajući se različitim potrebama i slučajevima korištenja [1].

Slika 7.4: OptiTrack Prime x 13 [1]



Slika 7.5: Postav sustava OptiTrack u CRTA-i

[1] OptiTrack; https://optitrack.com/

Postavljen OptiTrack sustav sastoji se od osam infracrvenih Prime x 13 kamera (slika 7.4) rezolucije 1280 x 1024 piksela s mogućnosti snimanja do 240 slika u sekundi. Tolerancija preciznosti pozicije u 3D prostoru iznosi +/-0.2 mm, dok je tolerancija kuta manja od 0.5 stupnjeva. Domet snimanja aktivnih markera je 25 m, što pokriva cijeli mjerni prostor u labosu.

Kako bi u ROS-u dobili podatke iz OptiTracka potrebno je pokrenuti VRPN (Virtual Reality Peripheral Network) paketom `vrpn-client-ros`, koji se instalira naredbom:

```
$ sudo apt-get install ros-noetic-vrpn-client-ros
```

nakon čega slijedi konfiguracija VRPN clienta. Kada je VRPN client konfiguriran, pokreće se naredbom:

```
$ roslaunch wcr_controller sample.launch server:=192.168.1.35
```

čime se objavljuje nova tema koja sadrži informacije o poziciji s OptiTracka, na koju se pretplaćujemo kako bi započeli snimanje.

7.3 Mjerenja

7.3.1 Prvo mjerenje

Inicijalno je odabrano vrijeme diskretizacije 0.045 s (frekvencija 22.22 Hz), a vremenski zavisna matrica $\mathbf{K}(t)$ učitana je iz MATLAB-ove simulacije u kojoj su ulazni argumenti `lqr()` funkcije $\mathbf{Q}$ i $\mathbf{R}$ slijedeći:

$$\mathbf{Q} = \text{diag}(10^5,\ 1,\ 10^2,\ 1,\ 10^4), \tag{7.1}$$

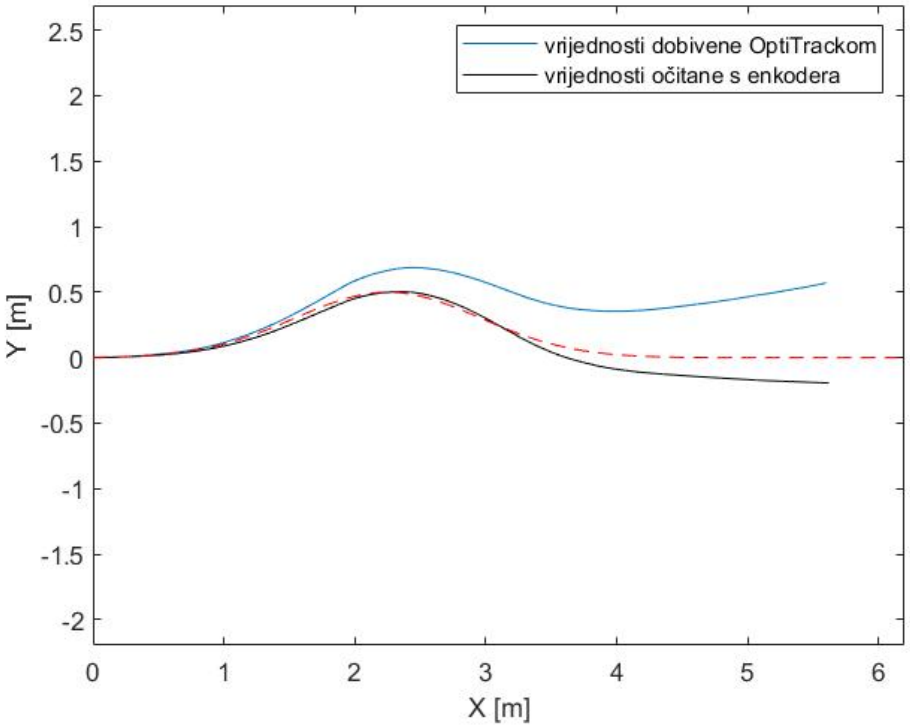
$$\mathbf{R} = \text{diag}(10^3,\ 1,\ 1), \tag{7.2}$$

pri čemu $\mathbf{Q}$ penalizira stanja, a $\mathbf{R}$ djelovanje motora. Odabrana trajektorija koju robot prati u stvarnom prostoru je Gaussova krivulja, pa su korišteni podaci i jednadžbe iz potpoglavlja **5.4**. Prije pokretanja robota, potrebno je započeti snimanje naredbom:

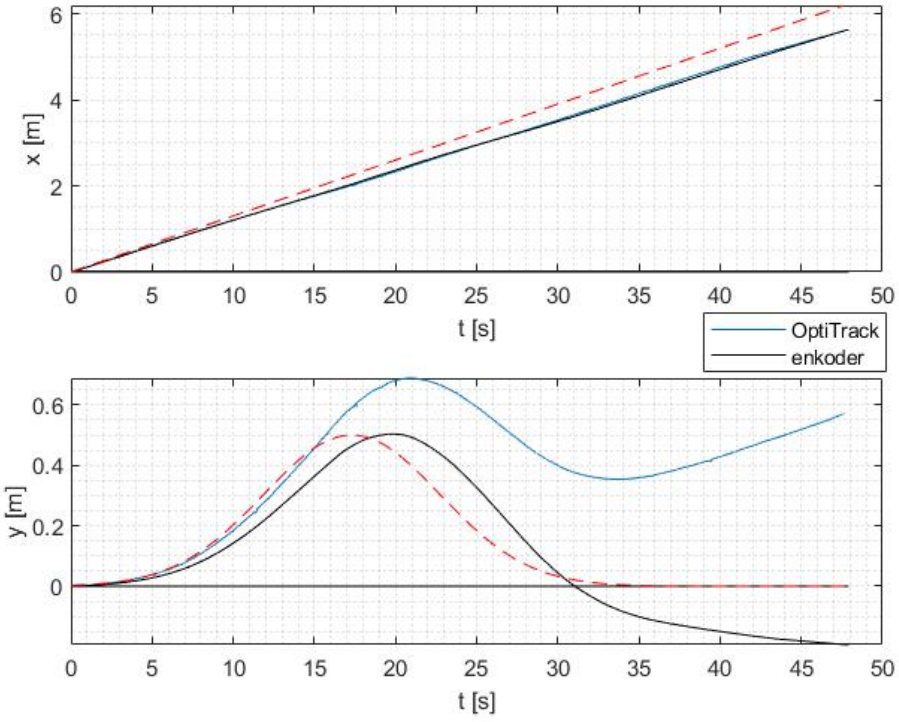
```
$ rosbag record -a
```

Nakon odrađenog snimanja podaci sa željenih tema se spremaju u obliku .csv tablice pomoću:

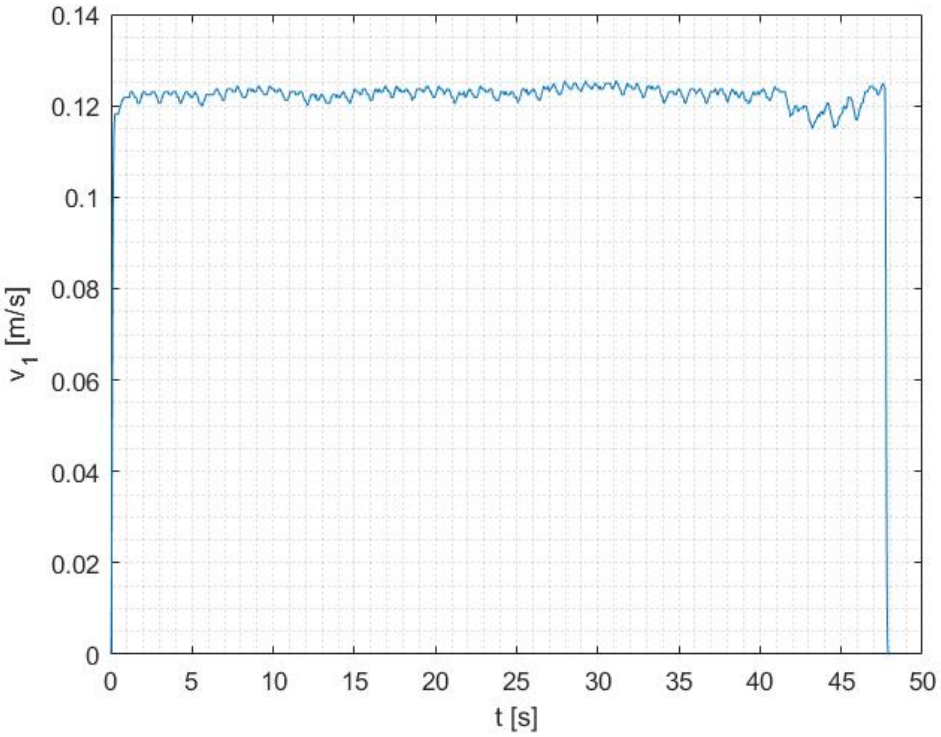
```
$ rosrun rosbag_to_csv rosbag_to_csv.py
```



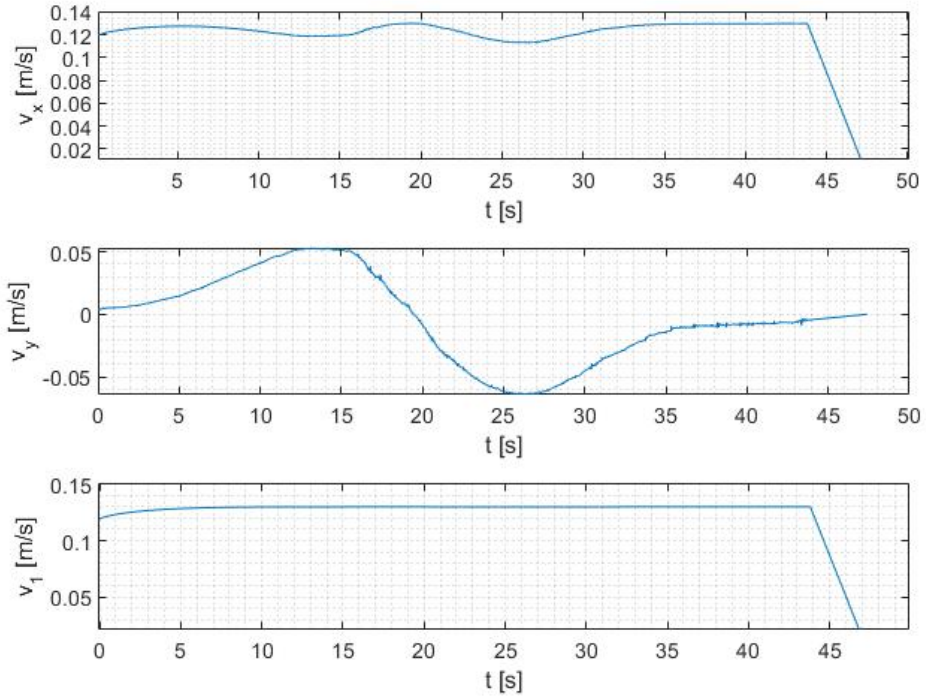
Slika 7.6: Rezultati praćenja trajektorije kod prvog mjerenja



Slika 7.7: Odstupanja u osima x i y kod prvog mjerenja



Slika 7.8: Brzina dobivena s enkodera kod prvog mjerenja



Slika 7.9: Upravljačke veličine prvog mjerenja

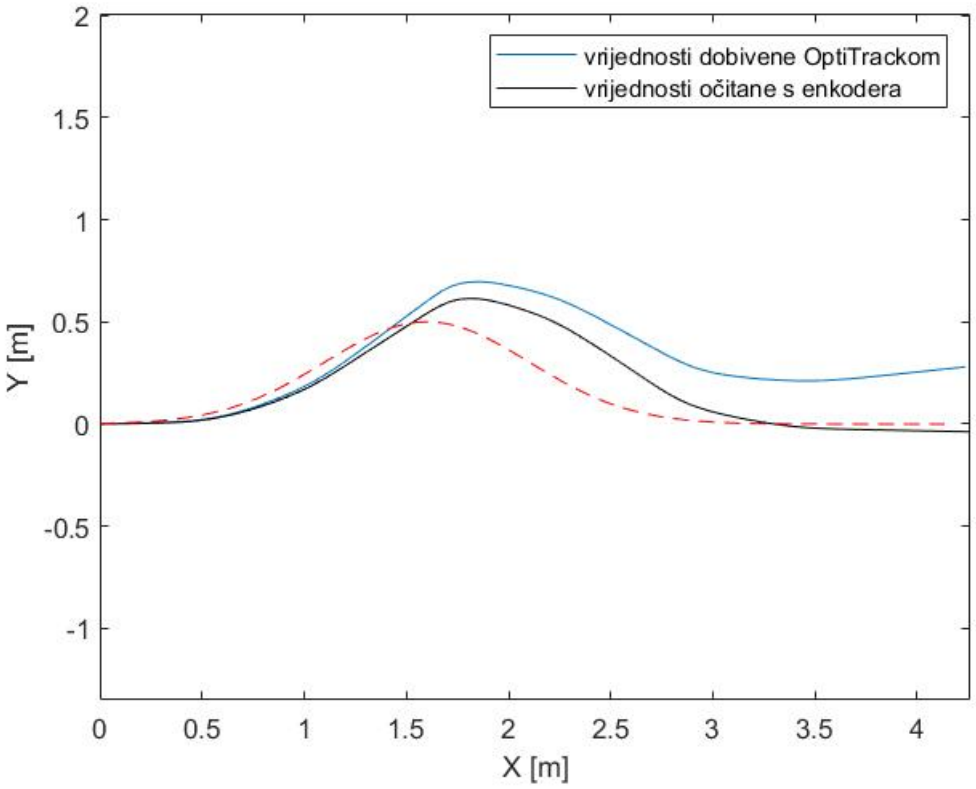
Na slikama 7.6 - 7.9. primjećujemo znatna odstupanja od reference, kako u stvarnosti tako i na enkoderu. S obzirom da dolazi do greške u obe osi kretanja, pretpostavlja se da greška nastaje radi ograničenja brzine funkcijom tangens hiperbolni u potpoglavlju **5.4**. Po rezultatima dobivenima s enkodera, moguće je primijetiti da robot "kasni" za željenom trajektorijom, radi čega će se smanjiti i vrijeme diskretizacije.

7.3.2 Drugo mjerenje

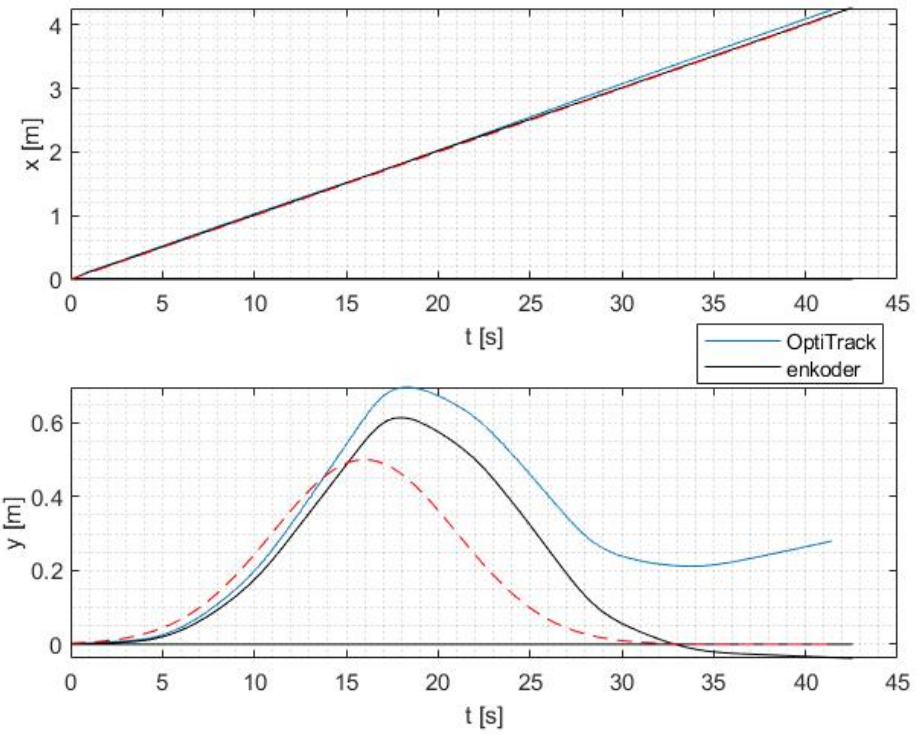
Kako brzina nebi prelazila ograničenje od 0.13 m/s, nominalna brzina je smanjena na 0.1 m/s, radi čega su skraćeni amplituda i duljina trajektorije, te su koeficijenti Gaussove krivulje slijedeći:

$$s = 2,$$
$$Y = 0.5,$$
$$x_c = 1.6,$$

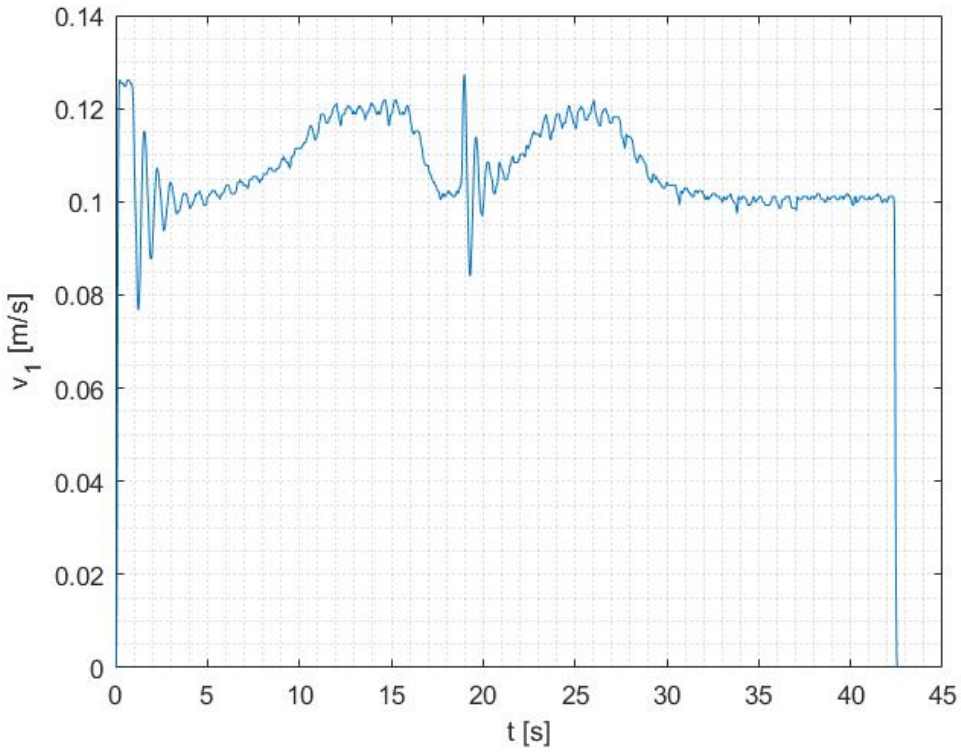
a vrijeme diskretizacije je smanjeno na 0.016 s (62.5 Hz).



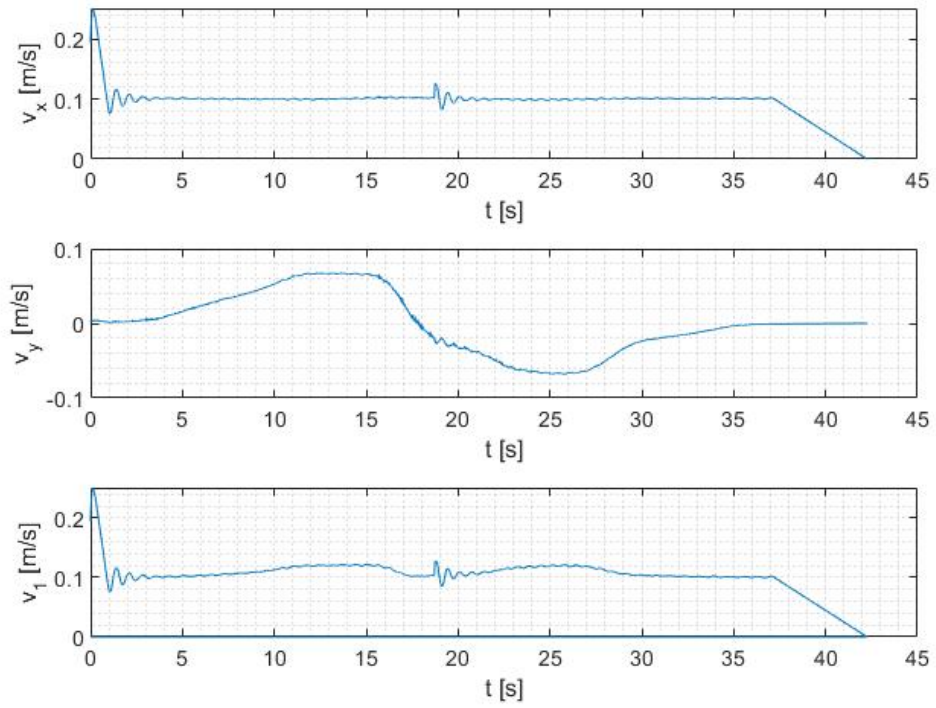
Slika 7.10: Rezultati praćenja trajektorije kod drugog mjerenja



Slika 7.11: Odstupanja u osima x i y kod drugog mjerenja



Slika 7.12: Brzina dobivena s enkodera kod drugog mjerenja



Slika 7.13: Upravljačke veličine drugog mjerenja

Rezultati na slikama 7.10 - 7.13 su vidno točniji u usporedbi s prvim mjerenjem, naročito vrijednosti dobivene s enkodera. Robot sada prati kretanje u smjeru osi x gotovo bez greške, no odstupanja od putanje su i dalje velika. Na slici 7.12 primjećujemo da dolazi do naglih skokova kod brzine v_1, što nam govori da bi se nominalna brzina trebala dodatno smanjiti. U daljnjim će mjerenjima biti prikazani rezultati samo za enkoder, dok podudaranje sa referencom ne bude potpuno.

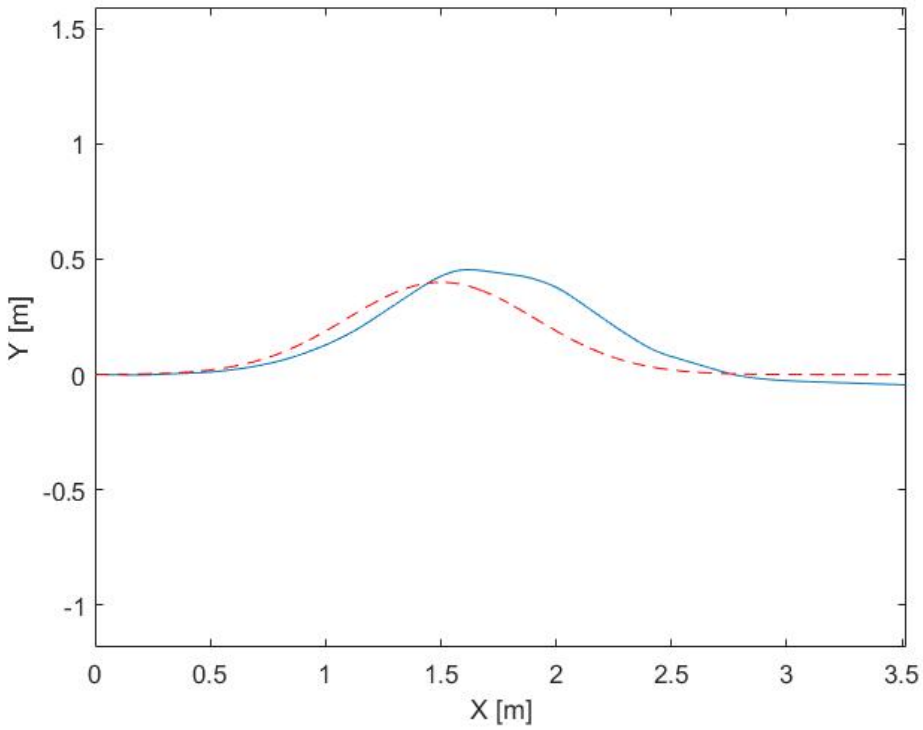
7.3.3 Treće i četvrto mjerenje

Nakon nekoliko iteracija pokazalo se da je optimalna nominalna brzina 0.06 m/s, a vrijednosti penalty matrice **Q** slijedeće:

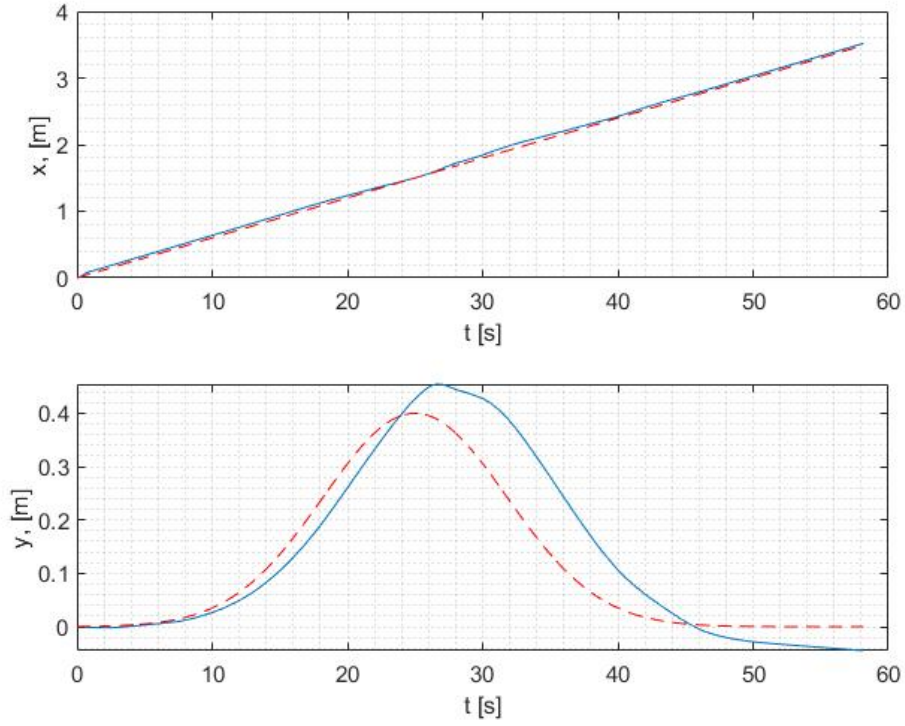
$$\mathbf{Q} = \text{diag}(10^5,\ 1,\ 1,\ 1,\ 10^6). \tag{7.3}$$

Smanjenjem nominalne brzine, izmijenjeni su i koeficijenti Gaussove krivulje:

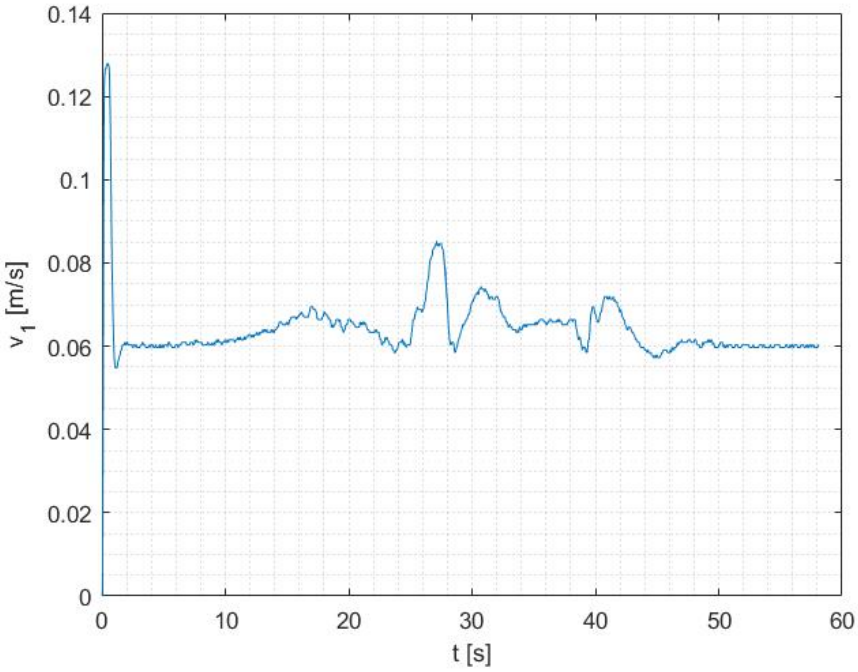
$$s = 3,$$
$$Y = 0.4,$$
$$x_c = 1.4.$$



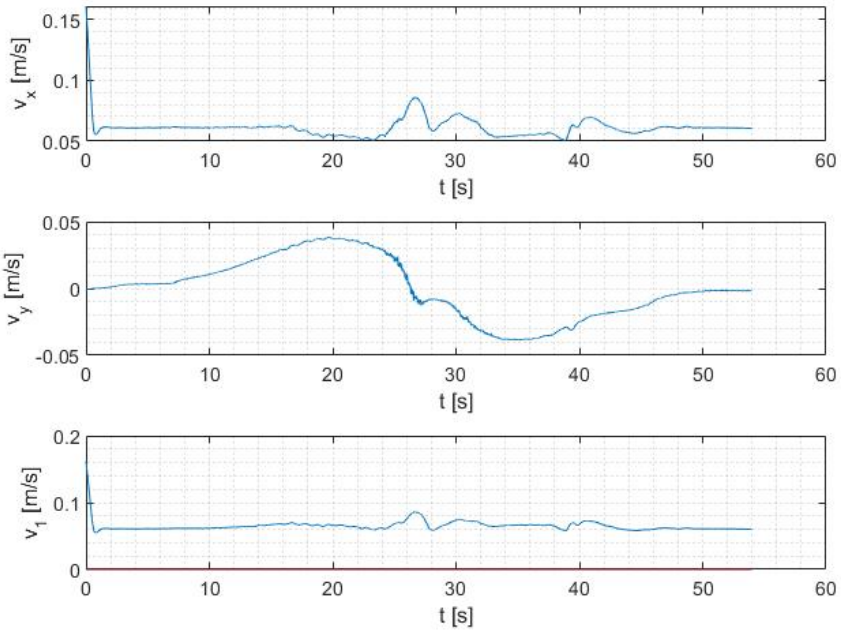
Slika 7.14: Rezultati praćenja trajektorije s enkodera kod trećeg mjerenja



Slika 7.15: Odstupanja u osima x i y s enkodera kod trećeg mjerenja



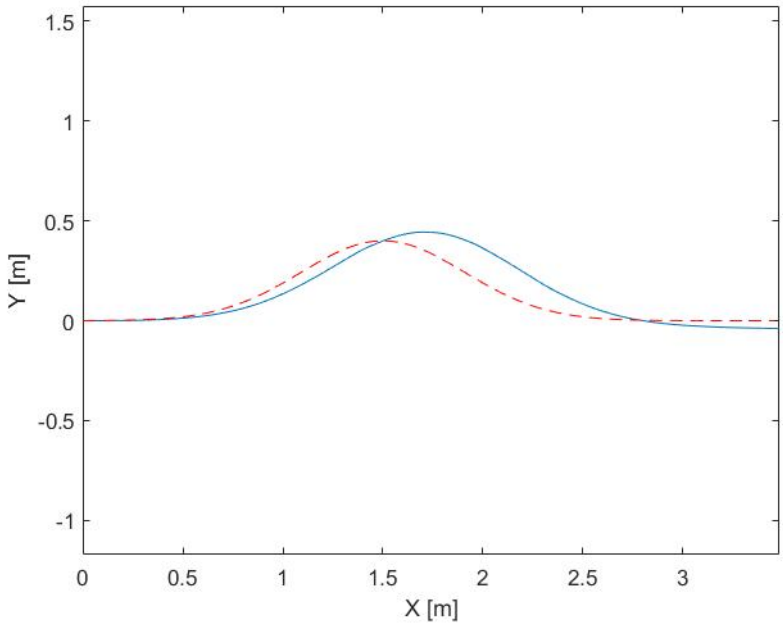
Slika 7.16: Brzina dobivena s enkodera kod trećeg mjerenja



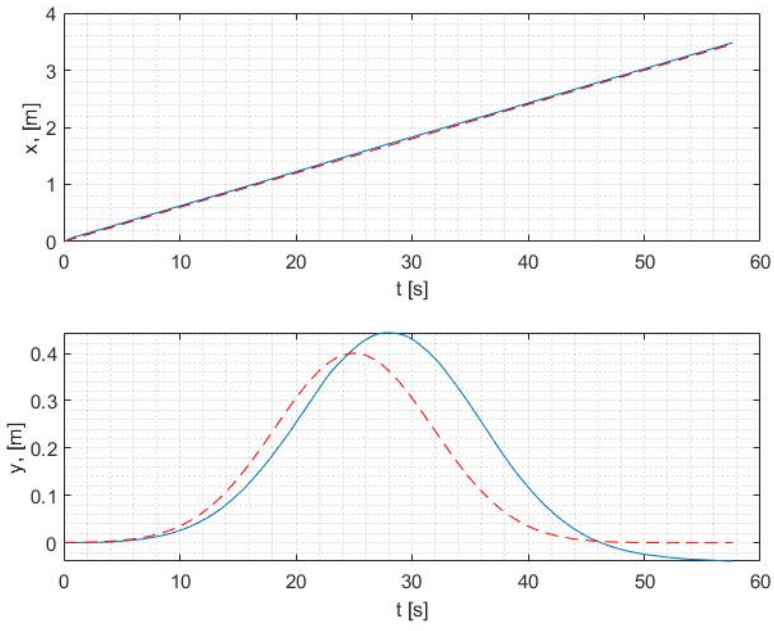
Slika 7.17: Upravljačke veličine trećeg mjerenja

Primjećuje se da se rezultati približavaju željenom cilju, no i dalje nisu zadovoljavajući. S obzirom da daljnja pojačavanja matrice **Q** nisu dala značajne promjene, kod četvrtog je mjerenja korištena kostantna matrica **K**, a odabrana je matrica u zadnjem trenutku prethodnog mjerenja te iznosi:

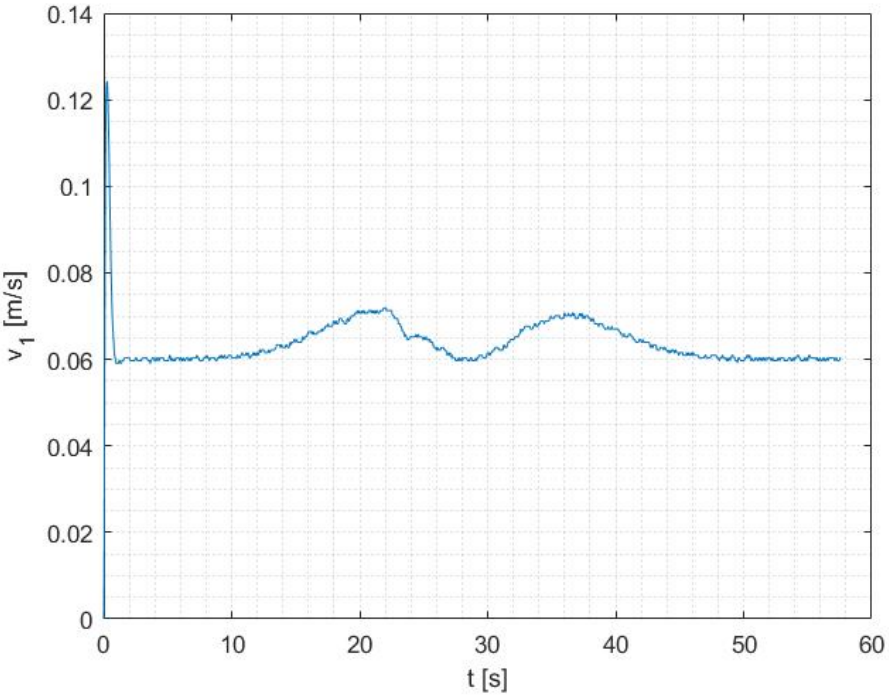
$$\mathbf{K} = \begin{bmatrix} 3.1588 & 0 & 0 & -0.0063 & -1.4876 \\ -0.0828 & 1.0583 & 1 & 0 & 0.0001 \\ 4.7041 & 0 & 0 & 10.9918 & 998.8930 \end{bmatrix}. \tag{7.4}$$



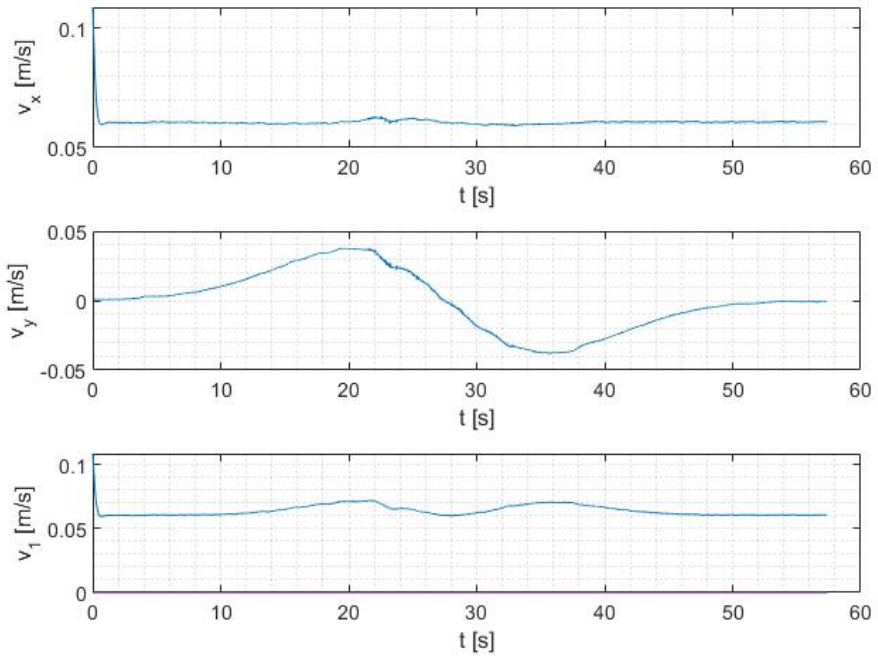
Slika 7.18: Rezultati praćenja trajektorije s enkodera kod četvrtog mjerenja



Slika 7.19: Odstupanja u osima x i y s enkodera kod četvrtog mjerenja



Slika 7.20: Brzina dobivena s enkodera kod četvrtog mjerenja



Slika 7.21: Upravljačke veličine četvrtog mjerenja

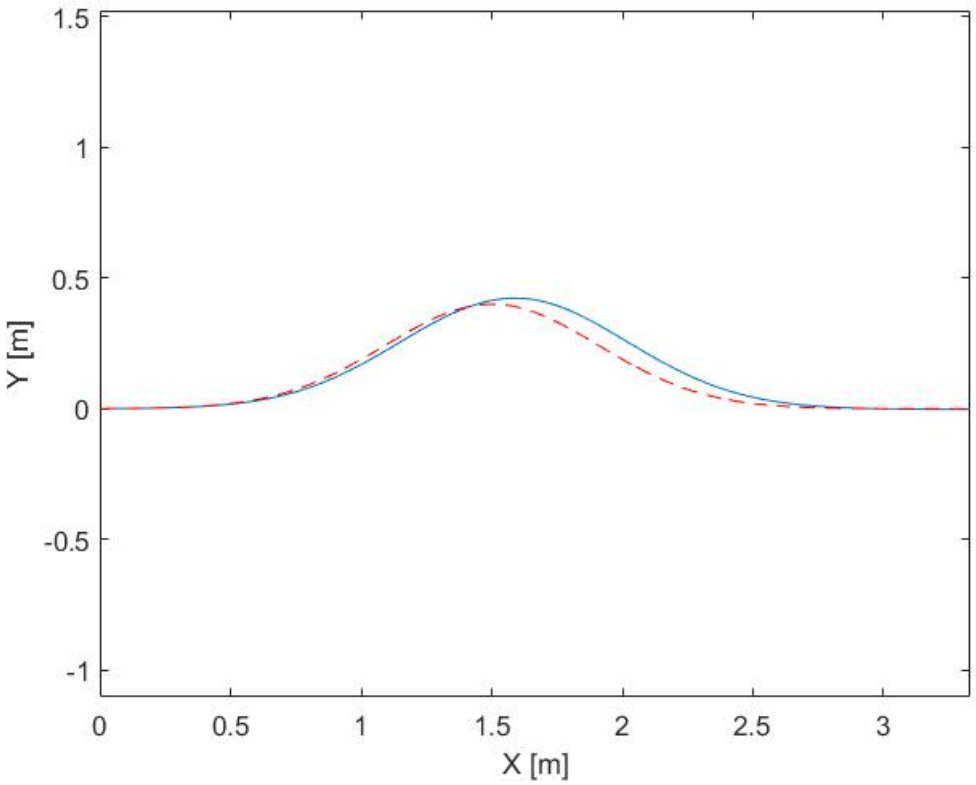
Iako pogreška nije znatno smanjena vidljivo je "glatko" kretanje, a usporedimo li grafove 7.17 i 7.21, primjećujemo da kod regulatora s konstantnom matricom **K** dolazi do manjih oscilacija brzine v_1.

7.3.4 Peto mjerenje

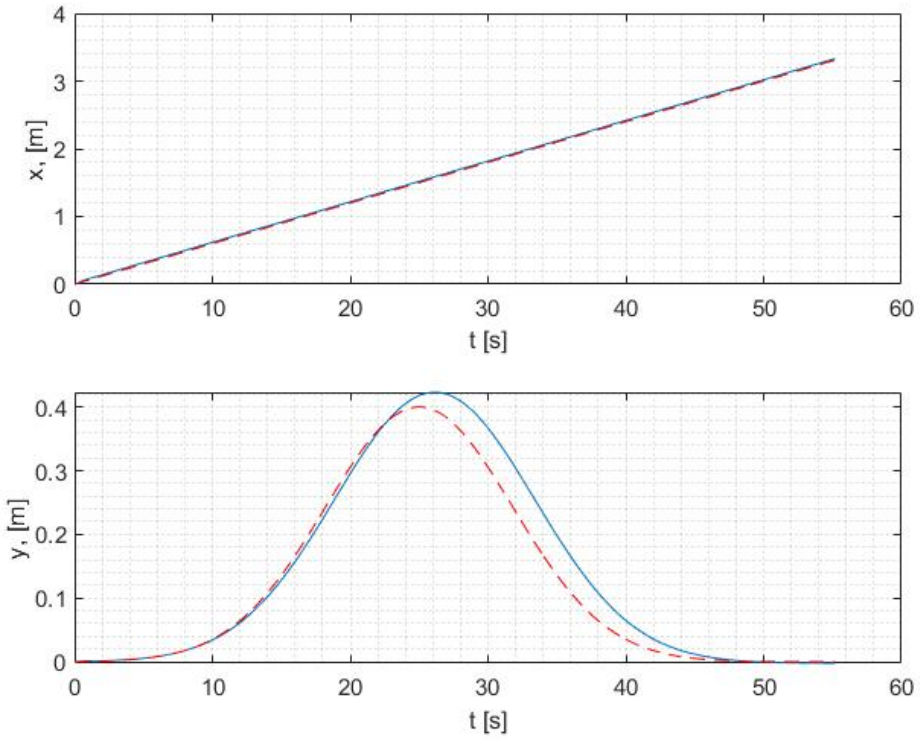
U prethodnom mjerenju vidljivo je da greška nije velika, ali da putanja robota kasni za referentnom Gaussovom krivuljom. Tom smo problemu pristupili podešavanjem PID regulatora Dynamixel servo motora. PID regulator ima tri podesiva parametra: pojačanje K_P, integralnu vremensku konstantu T_I i derivacijsku konstantu T_D. Prisustvo proporcionalnog, integralnog i diferencijalnog djelovanja u ovom regulatoru omugućuje dobijanje željenih performansi kao što su: stabilnost, brzina reakcije, preciznost rada i vrijeme trajanja prijelaznog procesa.

U ovom je slučaju povećana vrijednost koeficijenta proporcionalnog djelovanja (pojačanja) K_P, čime se smanjilo odstupanje od željene vrijednosti, a time i brzina reakcije. Ostale veličine su iste kao i u prethodnom mjerenju.

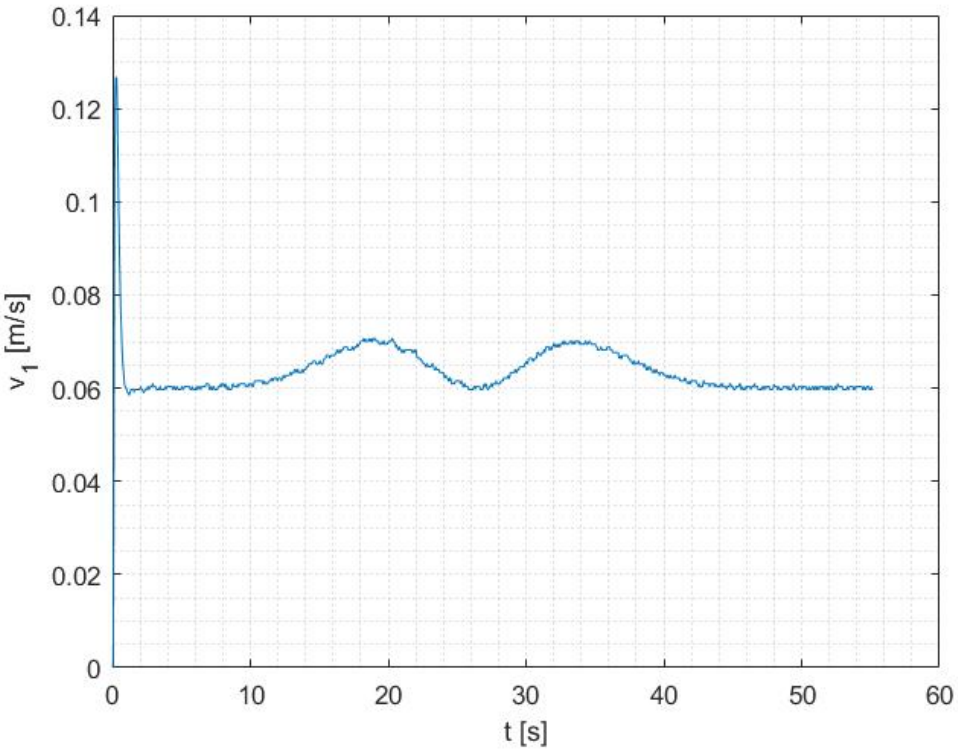
Prema rezultatima sa slika 7.22 i 7.23, vidi se da je greška znatno smanjena, ali da i robot još uvijek kasni za zadanom trajektorijom. Na grafu t, v_x sa slike 7.25. primjećuje se da dolazi do oscilacija upravljačke veličine v_x.



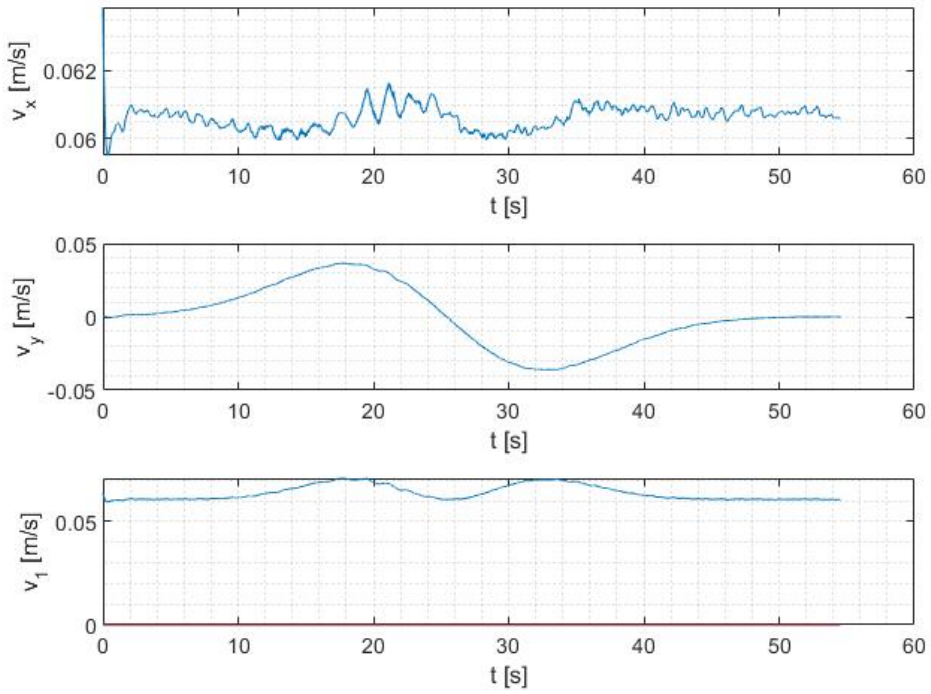
Slika 7.22: Rezultati praćenja trajektorije s enkodera kod petog mjerenja



Slika 7.23: Odstupanja u osima x i y s enkodera kod petog mjerenja



Slika 7.24: Brzina dobivena s enkodera kod petog mjerenja



Slika 7.25: Upravljačke veličine petog mjerenja

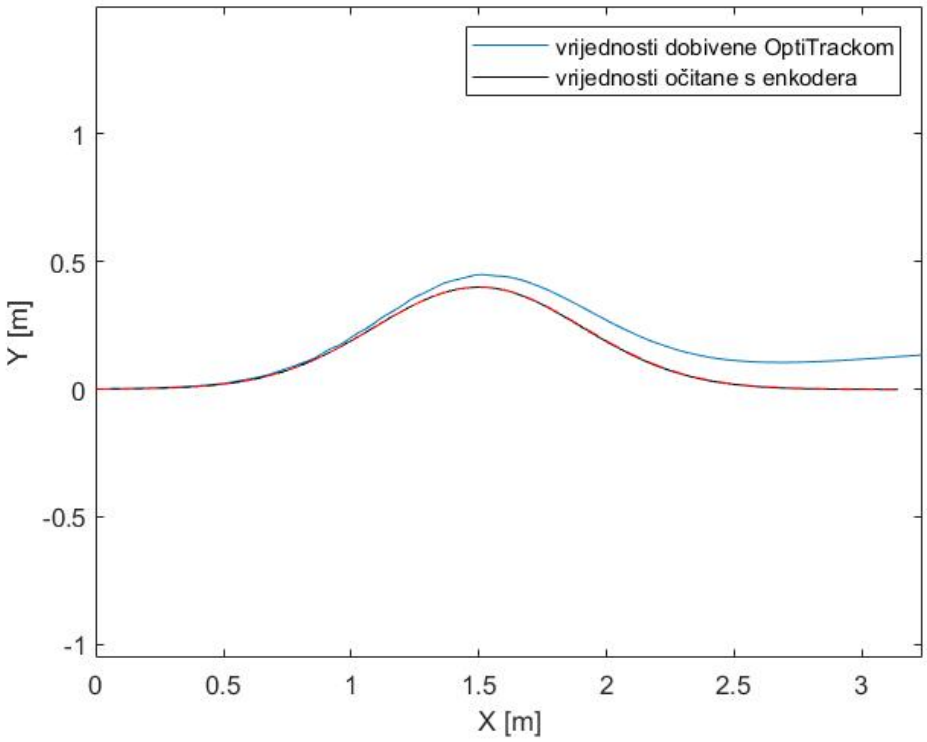
7.3.5 Šesto mjerenje

Pri rješavanju problema daljneg smanjenja kašnjenja robota, odabran je pristup optimizacije koda upravljačkog čvora kako bi se smanjilo vrijeme odziva. To se postiglo ukljanjanjem funkcije `controller()`, tako što se upravljački čvor pretplaćuje na temu `/wcr/joint_states` direktno u funkciji `main()` a svi se izračuni provode u `jointStateCallback()` funkciji iz koje se dobiveni podaci objavljuju na temu `/wcr/cmd_vel`. Novi se upravljački čvor pokreće naredbom;

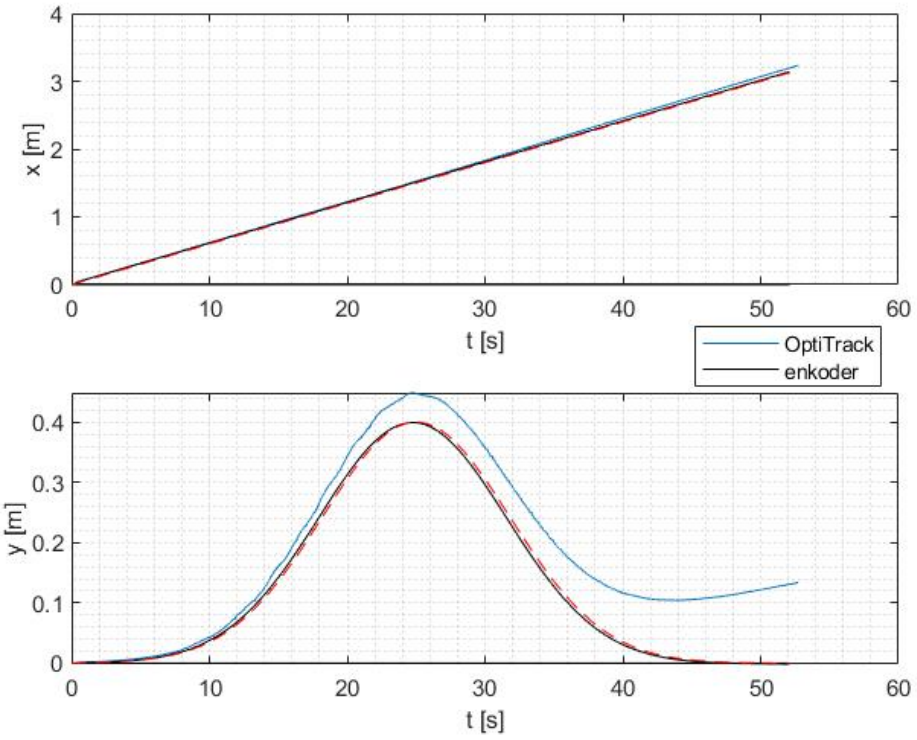
```
$ rosrun wcr_controller wcr_ns_controller3.py
```

a njegov se kod nalazi u prilogu. Rezultati su s obzirom na enkoder zadovoljavajući, pa su izmjereni s OptiTrackom i uspoređeni.

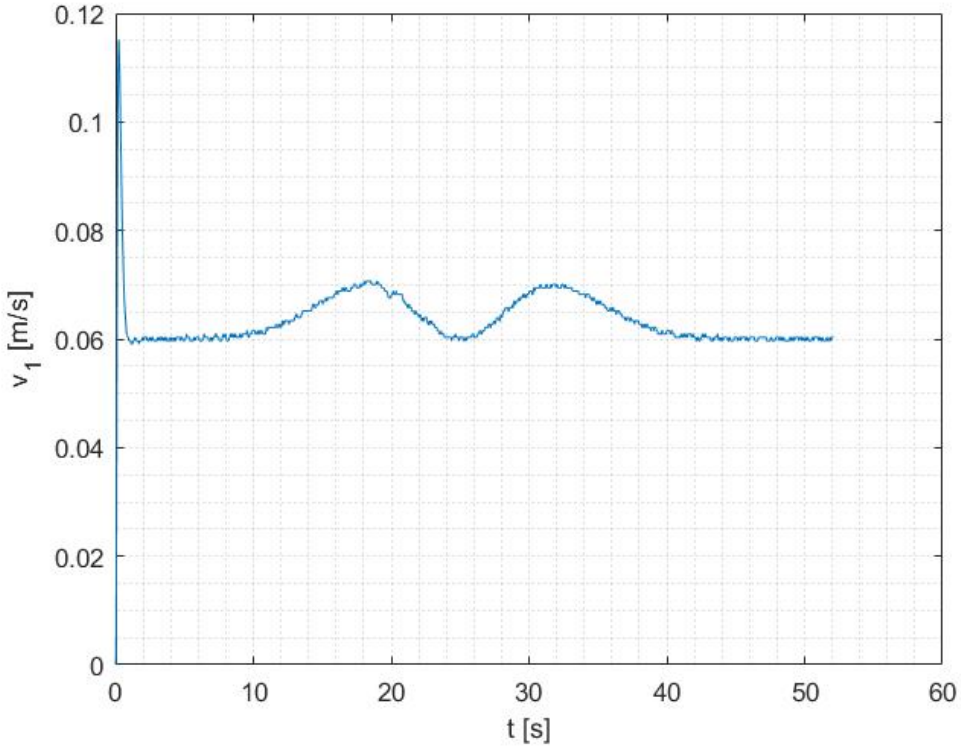
Na slikama 7.26 do 7.29 prikazani su rezultati dobiveni korištenjem konstantne matrice **K** iz izraza (7.4). Iako se vrijednosti s enkodera potpuno poklapaju s referentnom trajektorijom, i dalje je vidno odstupanje od rezultata dobivenih OptiTrackom, koje maksimalno iznosi 12.4 cm. Kako je i spomenuto u poglavlju **3**, to je odstupanje očekivano radi akumulacije greške u odometriji, te bi se u daljnjem istaživanju trebalo potražiti konstrukcijsko rješenje problema.



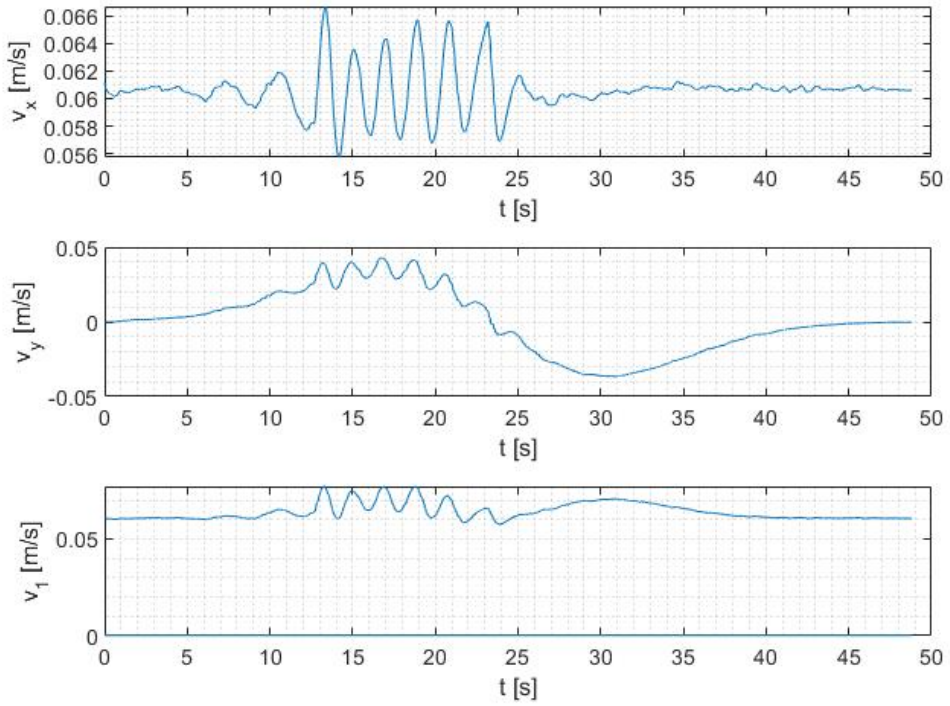
Slika 7.26: Rezultati praćenja trajektorije kod šestog mjerenja



Slika 7.27: Odstupanja u osima x i y kod šestog mjerenja



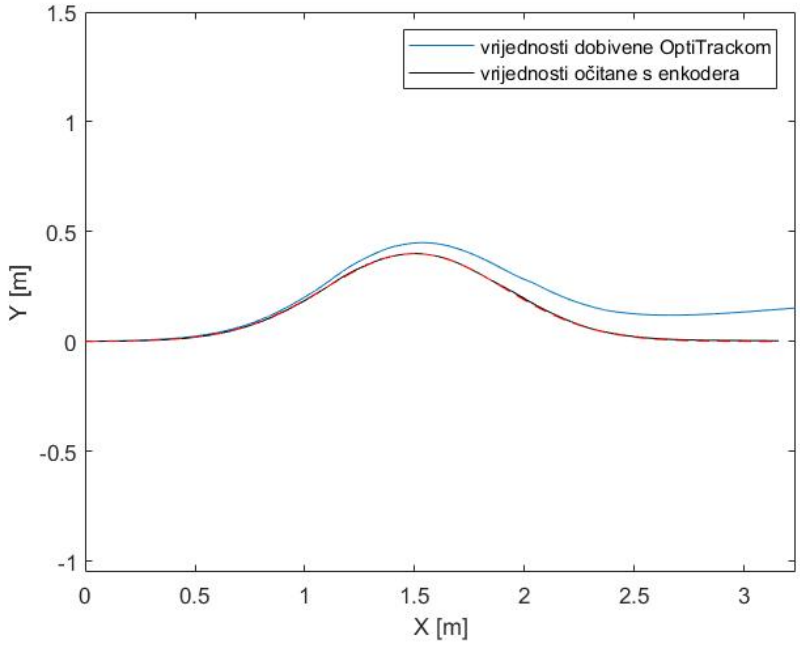
Slika 7.28: Brzina dobivena s enkodera kod šestog mjerenja



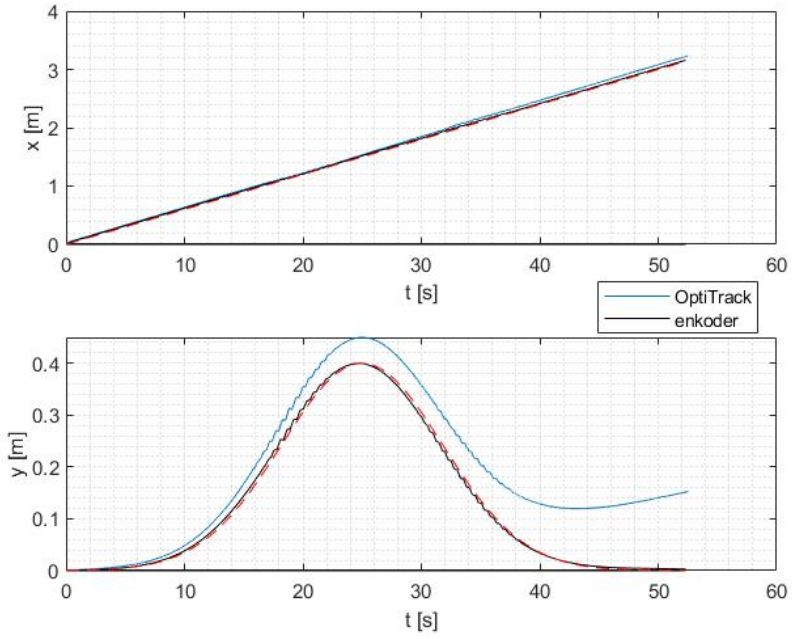
Slika 7.29: Upravljačke veličine šestog mjerenja

7.3.6 Sedmo mjerenje

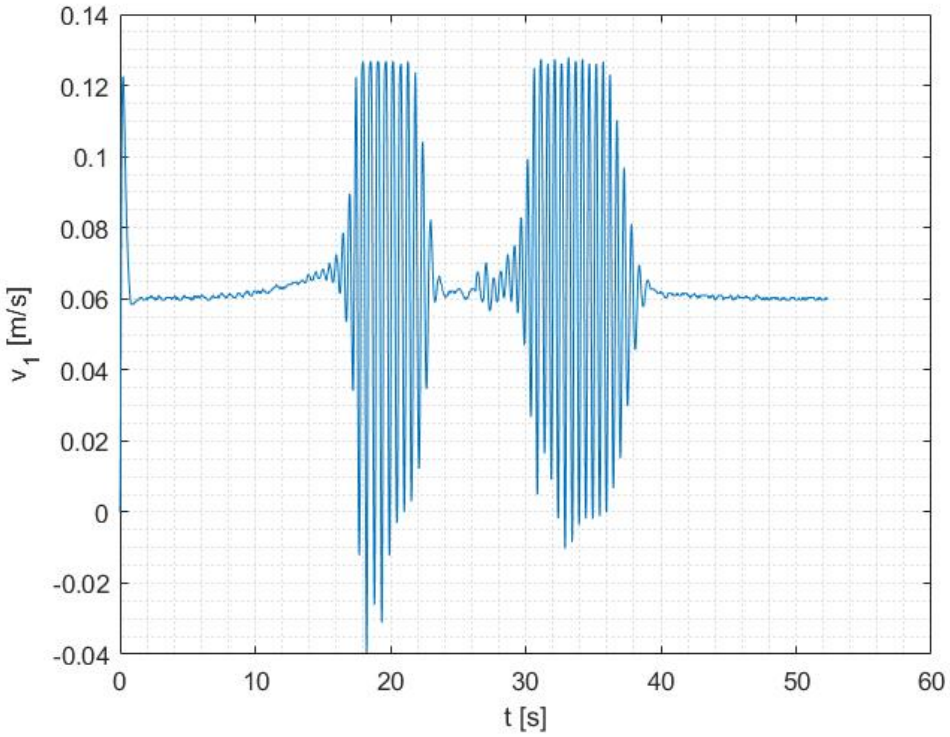
S obzirom da je u ovome radu naglasak na vremensko varijabilnom LQR-u, provedeno je i mjerenje za slučaj kada je matrica $\mathbf{K}$, to jest $\mathbf{K}(t)$, vremenski zavisna, pri čemu su preuzete vrijednosti matrice $\mathbf{Q}$ iz izraza (7.3).



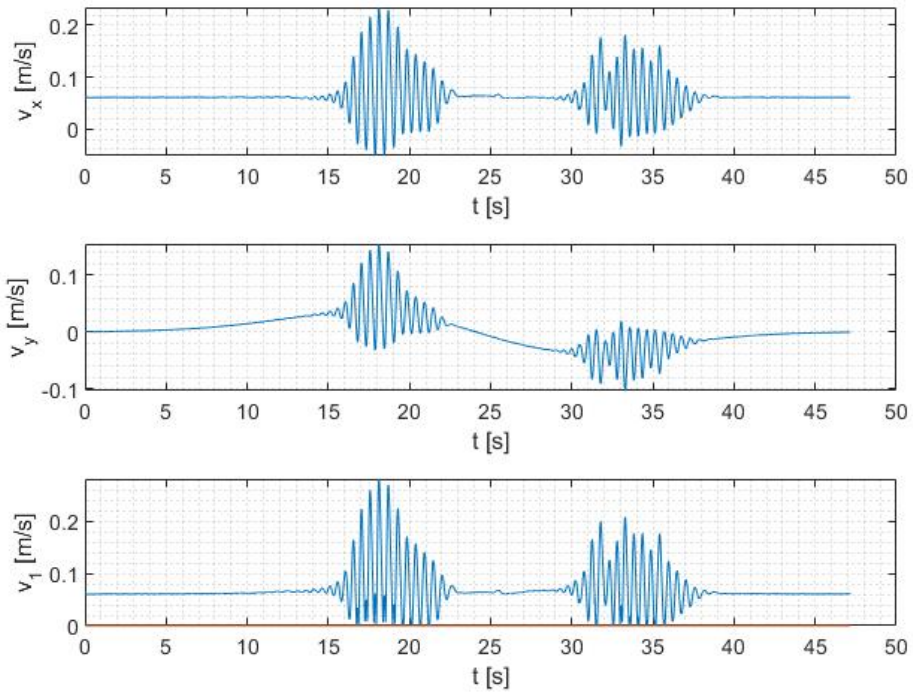
Slika 7.30: Rezultati praćenja trajektorije kod šestog mjerenja



Slika 7.31: Odstupanja u osima x i y kod šestog mjerenja



Slika 7.32: Brzina dobivena s enkodera kod šestog mjerenja

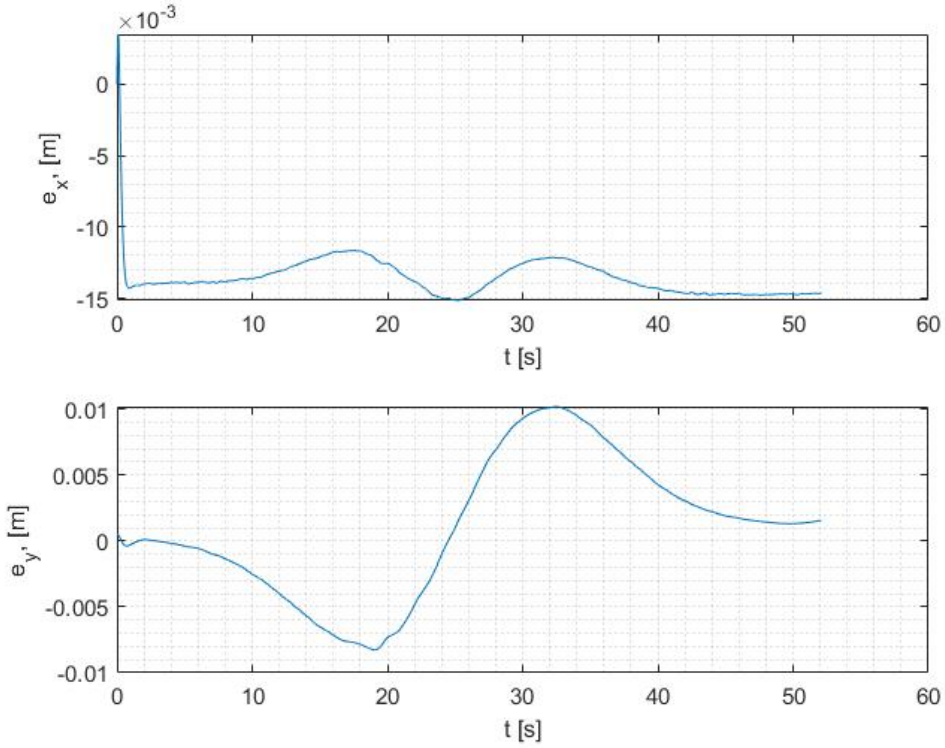


Slika 7.33: Upravljačke veličine sedmog mjerenja

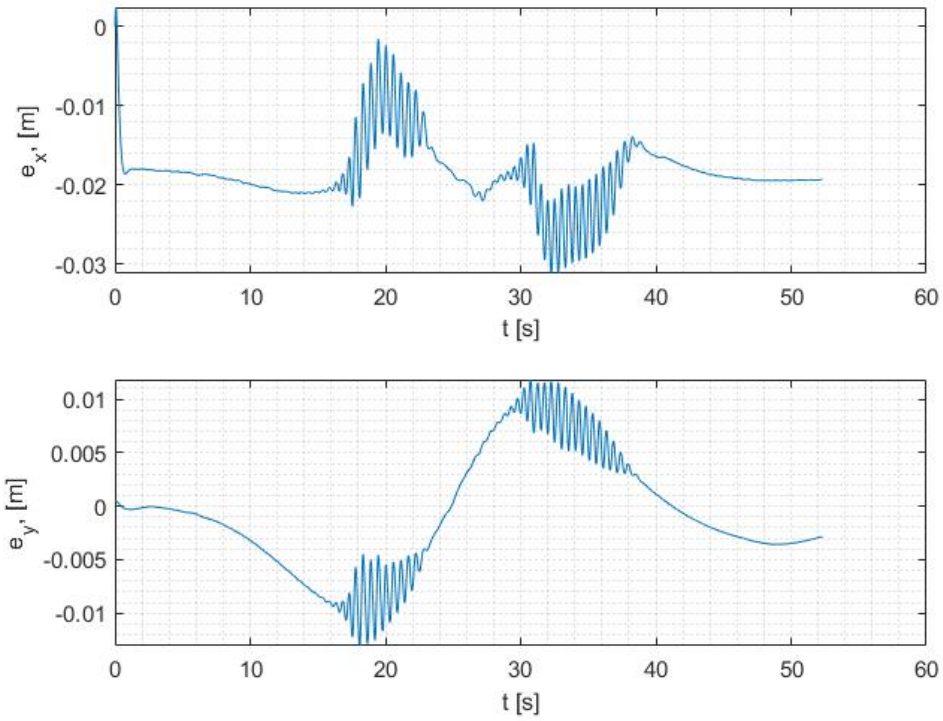
Na grafovima sa slika 7.30 i 7.31 primjećuje se da je odstupanje od rezultata dobivenih OptiTrackom neznatnto veće nego kod konstantnog regulatora i iznosi maksimalno 14.4 cm. Na slici 7.32 vidljiva je znatna razlika u uspredbi sa grafom 7.28, te dolazi do velikih oscilacija u brzini kretanja, što se događa i kod upravljačkih veličina. U stvarnom prostoru to izgleda tako da robot naglo ubrzava te zastaje kad dođe do maksimalne brzine od 0.13 m/s, nakon čega opet naglo ubrzava.

7.3.7 Usporedba vremenski nezavisnog i TV LQR-a

Iako je odstupanje rezultata od trajektorije snimljeno OptiTrackom dobiveno pomoću TV LQR-a veće nego ono dobiveno vremensko nezavisni regulatorom, naizgled je poklapanje s enkoderom savršeno u oba slučaja. Točno odstupanje je izmjereno i prikazano na slikama 7.34 i 7.35. Prvo što primjećujemo je da u oba slučaja najveća greška u smjeru osi y nastaje kod skretanja robota, te je približna jednaka u oba slučaja i iznosi +/-10 mm zanemarimo li oscilaciju kod TV LQR-a. U smjeru osi x greška nastaje na samom početku radi vremena koje je potrebno da brzina robota dođe do nominalne. Kod konstantne se matrice **K** ta početna greška značajno ne mijenja, te je maksimalna u vrhu amplitude Gaussove krivulje.



Slika 7.34: Odstupanje od rezultata enkodera kod konstantne matrice K



Slika 7.35: Odstupanje od rezultata enkodera kod vremenski promjenjive matrice $\mathbf{K(t)}$

Oscilacije kod vremenski zavisne matrice $\mathbf{K}(t)$ nastaju jer robot pokušava sustići putanju za kojom kasni, radi čega ubrzava, ali zastaje kada upravljačka brzina prelazi 0.13 m/s. Radi tih oscilacija maksimalna greška u smjeru x iznosi -30 mm u trenutku drugog skretanja.

8 Zaključak

Cilj ovog rada bio je razviti sustav upravljanja mobilnim robotom s povratnom vezom, tako da robot prati zadanu trajektoriju uz minimalna odstupanja. Jednadžbe kinematike robota s četiri pogonska motora s neovisnim upravljanjima zakretanjem kotača u ravnini transformirane su u lančani oblik, dvolančanog formata s trostrukim ulazom i pojedinim generatorom u lancu, za slučaj kada se svi kotači kreću jednakom brzinom, te da su prednja dva kotača međusobno paralelna, kao i zadnja dva. Kao upravljačke varijable odabrane su brzina kretanja robota v_1, te brzine zakreta prednjih ω_1 i zadnjih ω_2 kotača. Izvedbom ulančane forme dobivena su ograničenja pomoću kojih su izabrane refentne trajektorije. Vremenski varijantni zakon upravljanja proveden je prema linearnom kvadratnom kriteriju, te su provedene simulacije na referentnim trajektorijama u MATLAB-u.

Dobivene su se vrijednosti vremenski varijantne matrice upravljanja zatim implementirale na robota pomoću sustava ROS, te su se mjerila odstupanja od referentne trajektorije Gaussove krivulje s obzirom na vrijednosti dobivene s robotovog enkodera i na stvarne vrijednosti dobivene OptiTrack sustavom. Prvo mjerenje imalo je velika odstupanja i nije dalo zadovoljavajuće rezultate. Nakon prvog mjerenja slijedio je niz iteracija kojima se postupno smanjivala greška. Smanjenjem nominalne brzine kretanja, povećanjem frekvencije sustava i podešavanjem vremenski varijabilne matrice upravljanja dobiveni su točniji rezultati, ali su i dalje značajno odstupali od željenih. Zatim je bilo provedeno mjerenje s konstantnom matricom upravljanja koje, iako nije dalo značajne promjene pri praćenju putanje, je reduciralo nagle promjene pri brzini kretanja robota. Odstupanje od trajektorije znatno se smanjilo nakon podešavanja PID regulatora na motoru robota, pri čemu su dobivene veće oscilacije pri brzini kretanja. Optimizacijom koda upravljačkog čvora, rezultati dobiveni s enkodera su se potpuno poklapali s referencom, ali su i dalje odstupali od rezultata dobivenih OptiTrackom, što ukazuje na akumulaciju greške u odometriji robota.

Uspoređeni su rezultati mjerenja upravljanjem vremenski varijabilnom i konstantnom matricom upravljanja, gdje su kod konstantne matrice odstupanja bila manja radi velikih oscilacija brzine kretanja kod vremenski varijabilne matrice, koja su nastala radi prekoračenja maksimalne brzine robota 0.13 m/s. Za probleme akumulacije greške u odometriji i povećanja maksimalne brzine kretanja, predlaže se daljnje istraživanje u kojem bi se potražilo konstrukcijsko rješenje.

Literatura

[1] F. Rubio, F. Valero, and C. Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, pages 1–14, 2019.

[2] M. Božić, B. Ćaran, M. Švaco, B. Jerbić, and M. Serdar. Mobile Wall-Climbing Robot for NDT inspection of vertical concrete structures. *The International Symposium on Nondestructive Testing in Civil Engineering*, pages 1–14, 2022.

[3] M. Božić, B. Jerbić, and M Švaco. Development of a Mobile Wall-Climbing Robot with a Hybrid Adhesion System. *International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1136–1142, 2021.

[4] M.-H. Lee and T.-H. S. Li. Kinematics, dynamics and control design of 4WIS4WID mobile robots. *The Journal of Engineering*, 2015(1):6–16, 2015.

[5] J. Kasač. Opća teorija sustava. *Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje*, 2007.

[6] G. C. Walsh and L. G. Bushnell. Stabilization of multiple input chained form control systems. *Systems Control Letters*, 25(1):227–234, 1995.

[7] A. De Luca, G. Oriolo, and C. Samson. *Feedback control of a nonholonomic car-like robot*, pages 171–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[8] L. Bushnell, D. Tilbury, and S. S. Sastry. Steering Three-input Chained Form Nonholonomic Systems Using Sinusoids: The Firetruck Example. *Memorandum from UC Berekeley*, (UCB/ERL M92/107):1–27, 1993.

[9] M. F. Selekwa and J. R. Jonathan Nistler. Path Tracking Control of Four Wheel Independently Steered Ground Robotic Vehicles. *IEEE Conference on Decision and Control and European Control Conference*, 50:6355–6360, 2011.

[10] A. Arab, I. Hadžić, and J. Yi. Safe predictive control of four-wheel mobile robot with independent steering and drive. In *2021 American Control Conference (ACC)*, pages 2962–2967, 2021.

[11] X. Liu, W. Wang, X. Li, F. Liu, Z. He, Y. Yao, H. Ruan, and T. Zhang. MPC-based high-speed trajectory tracking for 4WIS robot. *ISA Transactions*, 123(33):413–424, 2022.

[12] A. E. Bryson. Time-Varying Linear-Quadratic Control. *Journal of Optimization Theory and Applications*, 100(3):515–525, 1999.

[13] A. Abbasi and A. J. Moshayedi. Trajectory Tracking of Two-Wheeled Mobile Robots, Using LQR Optimal Control Method, Based on Computational Model of KHEPERA IV. *Journal of Simulation Analysis of Novel Technologies in Mechanical Engineering*, 10(3), 2017.

[14] S. Morales, J. Magallanes, C. Delgado, and R Canahuire. LQR Trajectory Tracking Control of an Omnidirectional Wheeled Mobile Robot. *IEEE Colombian Conference on Robotics and Automation*, 2, 2018.

[15] M. Quigley, B. Gerkey, and W. D. Smart. *Programmig Robots with ROS*. O'Reilly Media, 2018.

[16] B. Ćaran. Planiranje kretanja i ispitivanje točnosti pozicioniranja mobilnog robota; Diplomski rad. *Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje*, 2021.

A MATLAB simulacije

A.1 Četvrtina kružnice

A.1.1 `chained_form` funkcija

```
function dx = chained_form(t, x)

global R om

t

%željena trajektorija
xd = [R*sin(om*t); sec(om*t)/R;  om*t; tan(om*t); R*(1-cos(om*t))];

%željene upravljačke vrijednosti
ud = [om*R*cos(om*t); (cos(om*t)*(1+om*tan(om*t))*(2+(tan(om*t))^2))/R;
    om*(sec(om*t))^2];

%sinteza regulatora
A = [0, 0, 0, 0, 0;
     0, 0, 0, 0, 0;
     0, ud(1), 0, 0, 0;
     0, 0, 0, 0, 0;
     0, 0, 0,  ud(1), 0]

B = [1, 0, 0;
     0, 1, 0;
     xd(2), 0, 0;
     0, 0, 1;
     xd(4), 0, 0]

Q = diag([1e4 10 1e3 10 1e4]);
Rl = diag([1e3 1e0 1e0]);
K = lqr(A, B, Q, Rl)
```

```
%zakon upravljanja
ut = ud +K*(xd-x(1:5));

u = ud - K*(xd-x(1:5));

%% Ulančani model
dx(1,1) = ut(1);
dx(2,1) = ut(2);
dx(3,1) = xd(2)*ut(1)+(x(2)-xd(2))*ud(1);
dx(4,1) = ut(3);
dx(5,1) = xd(4)*ut(1)+(x(4)-xd(4))*ud(1);

dx(6,1) = u(1);
dx(7,1) = u(2);
dx(8,1) = u(3);


%%--------------------------------
%% Nelinearni kinematički model

xx = x(9); y = x(10); theta = x(11); delta1 = x(12); delta2 = x(13);

xw1 = 0.1125; xw2 = -0.1125; yw1 = 0.1125; yw2 = 0.1125;

c1 = cos(delta1+theta); c2 = cos(delta2+theta);
s1 = sin(delta1+theta); s2 = sin(delta2+theta);

B1 = xw1*cos(delta1)/(xw1^2+yw1^2); B2 = xw2*cos(delta2)/(xw2^2+yw2^2);
D1 = xw1*sin(delta1)/(xw1^2+yw1^2); D2 = xw2*sin(delta2)/(xw2^2+yw2^2);

Kk = 1+cos(delta1+delta2+2*theta);


%transformacijske variable
xx1 = xx;
xx2 = (D1+D2)/(c1+c2);
xx3 = theta;
```

```
xx4 = tan((delta1+delta2)/2+theta);
xx5 = y;

XX = [xx1; xx2; xx3; xx4; xx5];

uu = ud + K*(xd-XX);

v1 = 2*uu(1)/(c1+c2);

omega1 = (B2*(2*uu(1)*(D1+D2)-Kk*uu(3)*(c1+c2)) + uu(2)*(c1+c2)^2)/...
    ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) +...
    ((D1+D2)*(-Kk*s2*uu(3)*(c1+c2)-(uu(1)*(D1+D2)*(s1-s2))))/...
    ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)));

omega2 = (B1*(-2*uu(1)*(D1+D2)+Kk*uu(3)*(c1+c2)) - uu(2)*(c1+c2)^2)/...
    ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) +...
    ((D1+D2)*(Kk*s1*uu(3)*(c1+c2)-(uu(1)*(D1+D2)*(s1-s2))))/...
    ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)));

dx(9,1) = 0.5*(cos(delta1+theta)+cos(delta2+theta))*v1;
dx(10,1) = 0.5*(sin(delta1+theta)+sin(delta2+theta))*v1;
dx(11,1) = (xw1*sin(delta1)/(2*xw1^2+2*yw1^2)+...
    xw2*sin(delta2)/(2*xw2^2+2*yw2^2))*v1;% theta
dx(12,1) = omega1;% delta1
dx(13,1) = omega2;% delta2

dx(14,1) = v1;
```

A.1.2 Izvršni kod

```
close all
clear all
clc

global R om

t_init = 0; % pocetno vrijeme
t_final =  40; % konacno vrijeme
```

```
% R*om=v=0.13
R = 3.5;
om =  0.47*pi/t_final;

%integriranje funkcije chained_form
x0 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
[T, X] = ode15s(@chained_form, [t_init t_final], x0);

%referenca
XR = R*sin(om*T);
YR = R*(1-cos(om*T));
ThetaR = om*T;

% upravljačke varijable u ulančanom modelu
Leng = length(T);
U1(1) = x0(6);
U2(1) = x0(7);
U3(1) = x0(8);
for i=1:(Leng-1)
    U1(i+1)=(X(i+1,6)-X(i,6))/(T(i+1)-T(i));
    U2(i+1)=(X(i+1,7)-X(i,7))/(T(i+1)-T(i));
    U3(i+1)=(X(i+1,8)-X(i,8))/(T(i+1)-T(i));
end

% upravljačke varijable u nelinearnom kinematičkom modelu
Leng = length(T);
V1(1) = x0(14);
Omega1(1) = x0(12);
Omega2(1) = x0(13);
for i=1:(Leng-1)
    V1(i+1)=(X(i+1,14)-X(i,14))/(T(i+1)-T(i));
    Omega1(i+1)=(X(i+1,12)-X(i,12))/(T(i+1)-T(i));
    Omega2(i+1)=(X(i+1,13)-X(i,13))/(T(i+1)-T(i));
end
```

```
figure(1)
plot(X(:,1), X(:,5), X(:,9), X(:,10), 'k', XR, YR, '--r')
legend('ulančani model', 'nelinearni kinematički model', 'referenca')
xlabel('X [m]')
ylabel('Y [m]')
axis equal

figure(2)
subplot(311)
plot(T, X(:,1), T, XR, '--r')
xlabel('t [s]')
ylabel('x, [m]')
grid minor

subplot(312)
plot(T, X(:,5), T, YR, '--r')
xlabel('t [s]')
ylabel('y, [m]')
grid minor

subplot(313)
plot(T, X(:,3), T, ThetaR, '--r')
xlabel('t [s]')
ylabel('theta, [rad]')
grid minor


figure(4)
subplot(311)
plot(T, U1)
xlabel('t [s]')
ylabel('u_1 [m/s]')
grid minor

subplot(312)
plot(T, U2)
xlabel('t [s]')
```

```
ylabel('u_2 [rad/(ms)]')
grid minor

subplot(313)
plot(T, U3)
xlabel('t [s]')
ylabel('u_3 [rad/s]')
grid minor

a = axes;
a.Visible = 'off';
tit1.Visible = 'on';

figure(5)
subplot(311)
plot(T, V1)
xlabel('t [s]')
ylabel('v_1 [m/s]')
grid minor

subplot(312)
plot(T, Omega1)
xlabel('t [s]')
ylabel('\omega_1 [rad/s]')
grid minor

subplot(313)
plot(T, Omega2)
xlabel('t [s]')
ylabel('\omega_2 [rad/s]')
grid minor

a = axes;
a.Visible = 'off';
tit1.Visible = 'on';
```

```
figure(6)
subplot(231)
plot(T, X(:,9))
xlabel('t [s]')
ylabel('x [m]')
grid minor

subplot(232)
plot(T, X(:,10))
xlabel('t [s]')
ylabel('y [m]')
grid minor

subplot(233)
plot(T, X(:,11))
xlabel('t [s]')
ylabel('\theta [rad]')
grid minor

subplot(234)
plot(T, X(:,12))
xlabel('t [s]')
ylabel('\delta_1 [rad]')
grid minor

subplot(235)
plot(T, X(:,13))
xlabel('t [s]')
ylabel('\delta_2 [rad]')
grid minor

a = axes;
a.Visible = 'off';
tit1.Visible = 'on';
```

A.2 Eulerova diskretizacija

```
function [tout, xout] = my_euler(FunFcn, tspan, x0, ssize)

% Funkcija za rjesavanje problema pocetnih uvjeta:
% x'(t) = f(x, t), x(0) = x0
% primjenom Eulerove metode

% tout - izlaz --> vrijeme
% xout - izlaz --> vektor varijabli stanja
% FunFcn - ulaz --> sustav dif. jednadzbi f(x, t)
% tspan - ulaz --> vremenski interval
% x0 - ulaz --> vektor pocetnih uvjeta
% sssize - ulaz --> vremenski korak

global N

% Inicijalizacija
t0 = tspan(1); tfinal = tspan(2); h = ssize;
t = t0;
x = x0(:);

% Inicijalizacija izlaza
tout = zeros(N, 1); tout(1) = t;
xout = zeros(N, size(x, 1)); xout(1, :) = x.';
k = 1;

% Algoritam
while (k < N+1)
  if (t + h - tfinal) > 0
    h = tfinal - t;
    tout(k+1) = tfinal;
  else
    tout(k+1) = t0 +k*h;
  end
  k = k+1;
```

```
  s1 = feval(FunFcn, t, x); s1 = s1(:);
  x = x + h*s1;
  t = tout(k);
  xout(k, :) = x.';
end
```

A.3 Gaussova krivulja

A.3.1 chained_gauss funkcija

```
function dx = chained_gauss(t, x)

global s Y xc vm K tau Knew

t
i=int32(t/tau+1);

vx=vm;

%željena trajektorija
xd = [vx*t;
      (2*exp(s*(-t*vx+xc)^2)*s*(-1+2*s*(-t*vx+xc)^2)*Y)/(exp(2*s*(-t*vx+xc)^2)
    +4*s^2*(-t*vx+xc)^2*Y^2);
      atan2(2*exp(-s*(xc-vx*t)^2)*s*Y*(xc-vx*t),1);
      -2*exp(-s*(-xc+vx*t)^2)*s*Y*(-xc+vx*t);
      Y*exp(-s*(vx*t-xc)^2)];

%stvarna brzina
vx = vm/sqrt(4*s^2*xd(5)^2*(xd(1)-xc)^2+1);

%željene upravljačke vrijednosti
ud = [vx;
     -((4*exp(s*(-t*vx+xc)^2)*s^2*vx*(t*vx-xc)*Y*(exp(2*s*(-t*vx+xc)^2)*...
    (-3+2*s*(-t*vx+xc)^2)-4*s*(1+s*(-t*vx+xc)^2*(-1+2*s*(-t*vx+xc)^2))*Y^2))/...
    (exp(2*s*(-t*vx+xc)^2)+4*s^2*(-t*vx+xc)^2*Y^2)^2);
     2*exp(-s*(-t*vx+xc)^2)*s*vx*(-1+2*s*(-t*vx+xc)^2)*Y];
```

```
%sinteza regulatora
A = [0, 0, 0, 0, 0;
     0, 0, 0, 0, 0;
     0, ud(1), 0, 0, 0;
     0, 0, 0, 0, 0;
     0, 0, 0,  ud(1), 0]

B = [1, 0, 0;
     0, 1, 0;
     xd(2), 0, 0;
     0, 0, 1;
     xd(4), 0, 0]

Q = diag([1e5 1 1 1 1e6]);
Rl = diag([1e3 1 1]);
K = lqr(A, B, Q, Rl);

%spremanje vremenski varijabilne matrice
Knew(:,:,i)=K;

%zakon upravljanja
ut = ud - K*(x(1:5)-xd);

u = 2*ud - ut;

%% Ulančani model
dx(1,1) = ut(1);
dx(2,1) = ut(2);
dx(3,1) = xd(2)*ut(1)+(x(2)-xd(2))*ud(1);
dx(4,1) = ut(3);
dx(5,1) = xd(4)*ut(1)+(x(4)-xd(4))*ud(1);

dx(6,1) = u(1);
dx(7,1) = u(2);
dx(8,1) = u(3);
```

```
%%--------------------------------
%% Nelinearni kinematički model

xx = x(9); y = x(10); theta = x(11); delta1 = x(12); delta2 = x(13);

xw1 = 0.1125; xw2 = -0.1125; yw1 = 0.1125; yw2 = 0.1125;

c1 = cos(delta1+theta); c2 = cos(delta2+theta);
s1 = sin(delta1+theta); s2 = sin(delta2+theta);

B1 = xw1*cos(delta1)/(xw1^2+yw1^2); B2 = xw2*cos(delta2)/(xw2^2+yw2^2);
D1 = xw1*sin(delta1)/(xw1^2+yw1^2); D2 = xw2*sin(delta2)/(xw2^2+yw2^2);

Kk = 1+cos(delta1+delta2+2*theta);

%transformacijske varijable
xx1 = xx;
xx2 = (D1+D2)/(c1+c2);
xx3 = theta;
xx4 = tan((delta1+delta2)/2+theta);
xx5 = y;

XX = [xx1; xx2; xx3; xx4; xx5];

uu = ud + K*(xd-XX);

v1 = 2*uu(1)/(c1+c2);
%v1 = 0.13*tanh(2.5*v1); % v1max=0.13 m/s

omega1 = (B2*(2*uu(1)*(D1+D2)-Kk*uu(3)*(c1+c2)) + uu(2)*(c1+c2)^2)/...
    ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) +...
    ((D1+D2)*(-Kk*s2*uu(3)*(c1+c2)-(uu(1)*(D1+D2)*(s1-s2))))/...
    ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)));

omega2 = (B1*(-2*uu(1)*(D1+D2)+Kk*uu(3)*(c1+c2)) - uu(2)*(c1+c2)^2)/...
    ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) +...
    ((D1+D2)*(Kk*s1*uu(3)*(c1+c2)-(uu(1)*(D1+D2)*(s1-s2))))/...
```

```
    ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)));

dx(9,1) = 0.5*(cos(delta1+theta)+cos(delta2+theta))*v1;
dx(10,1) = 0.5*(sin(delta1+theta)+sin(delta2+theta))*v1;
dx(11,1) = (xw1*sin(delta1)/(2*xw1^2+2*yw1^2)+...
    xw2*sin(delta2)/(2*xw2^2+2*yw2^2))*v1;% theta
dx(12,1) = omega1;% delta1
dx(13,1) = omega2;% delta2

dx(14,1) = v1;
```

A.3.2 Pozivna funkcija

```
close all
clear all
clc

global s Y xc vm N tau Knew

t_init = 0; % pocetno vrijeme
t_final = 40; % konacno vrijeme
N = 1e3; % broj vremenskih intervala
tau = (t_final - t_init)/ N; % vremenski korak

s = 1;
Y =  0.5;
xc = 2.25;
vm = 0.13;

%diskretizacija funkcijom my_euler
x0 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
[T, X] = my_euler(@chained_gauss, [t_init t_final], x0, tau);

%referenca
XR = vm*T;
YR = Y*exp(-s*(vm*T- xc).^2);
ThetaR =  atan((-vm*T+ xc)*2.*exp(-s*(-vm*T +xc).^2).*s*Y );
```

```
% upravljačke varijable u ulančanom modelu
Leng = length(T);
U1(1) = x0(6);
U2(1) = x0(7);
U3(1) = x0(8);
for i=1:(Leng-1)
    U1(i+1)=(X(i+1,6)-X(i,6))/(T(i+1)-T(i));
    U2(i+1)=(X(i+1,7)-X(i,7))/(T(i+1)-T(i));
    U3(i+1)=(X(i+1,8)-X(i,8))/(T(i+1)-T(i));
end

% upravljačke varijable u nelinearnom kinematičkom modelu
Leng = length(T);
V1(1) = x0(14);
Omega1(1) = x0(12);
Omega2(1) = x0(13);
for i=1:(Leng-1)
    V1(i+1)=(X(i+1,14)-X(i,14))/(T(i+1)-T(i));
    Omega1(i+1)=(X(i+1,12)-X(i,12))/(T(i+1)-T(i));
    Omega2(i+1)=(X(i+1,13)-X(i,13))/(T(i+1)-T(i));
end

%spremanje vremenski varijabilne matrice u .txt file
Knew
size(Knew)
Ttxt = fopen('tablica.txt','w'); % ili 'wt'
fprintf(Ttxt, '\t %.4f \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f ...
    \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f \t %.4f \n', ...
    pagetranspose(Knew));
fprintf(Ttxt,'\r\n');
fclose(Ttxt);


figure(1)
plot(X(:,1), X(:,5), X(:,9), X(:,10), 'k', XR, YR, '--r')
legend('ulančani model', 'nelinearni kinematički model', 'referenca')
xlabel('X [m]')
```

```
ylabel('Y [m]')
axis equal

figure(2)
subplot(311)
plot(T, X(:,1), T, XR, '--r')
xlabel('t [s]')
ylabel('x, [m]')
grid minor

subplot(312)
plot(T, X(:,5), T, YR, '--r')
xlabel('t [s]')
ylabel('y, [m]')
grid minor

subplot(313)
plot(T, X(:,3), T, ThetaR, '--r')
xlabel('t [s]')
ylabel('theta, [rad]')
grid minor


figure(4)
subplot(311)
plot(T, U1)
xlabel('t [s]')
ylabel('u_1 [m/s]')
grid minor

subplot(312)
plot(T, U2)
xlabel('t [s]')
ylabel('u_2 [rad/(ms)]')
grid minor

subplot(313)
```

```
plot(T, U3)
xlabel('t [s]')
ylabel('u_3 [rad/s]')
grid minor

a = axes;
a.Visible = 'off';
tit1.Visible = 'on';

figure(5)
subplot(311)
plot(T, V1)
xlabel('t [s]')
ylabel('v_1 [m/s]')
grid minor

subplot(312)
plot(T, Omega1)
xlabel('t [s]')
ylabel('\omega_1 [rad/s]')
grid minor

subplot(313)
plot(T, Omega2)
xlabel('t [s]')
ylabel('\omega_2 [rad/s]')
grid minor

a = axes;
a.Visible = 'off';
tit1.Visible = 'on';


figure(6)
subplot(231)
plot(T, X(:,9))
xlabel('t [s]')
```

```
ylabel('x [m]')
grid minor

subplot(232)
plot(T, X(:,10))
xlabel('t [s]')
ylabel('y [m]')
grid minor

subplot(233)
plot(T, X(:,11))
xlabel('t [s]')
ylabel('\theta [rad]')
grid minor

subplot(234)
plot(T, X(:,12))
xlabel('t [s]')
ylabel('\delta_1 [rad]')
grid minor

subplot(235)
plot(T, X(:,13))
xlabel('t [s]')
ylabel('\delta_2 [rad]')
grid minor

a = axes;
a.Visible = 'off';
tit1.Visible = 'on';
```

A.4 Sinusoida

A.4.1 `chained_sin` funkcija

```
function dx = chained_sin(t, x)

global a om N vm tau Ksin

t
i=int32(t/tau+1);

vx=vm;

%željena trajektorija
xd = [vx*t;
      -(a*om^2*sin(om*t))/(vx^2+a^2*om^2*(cos(om*t))^2);
      atan((a*om*cos(om*t))/vx);
      (a*om*cos(om*t))/vx;
      a*sin(om*t)];

%stvarna brzina
vx = vx *cos(xd(3));

%željene upravljačke vrijednosti
ud = [vx;
      (a *om^3 *cos(om*t) *(-3 *a^2 *om^2 - 2 *vx^2 +...
    a^2 *om^2 *cos(2*om*t)))/(2 *(vx^2 +  a^2 *om^2 *(cos(om*t))^2)^2);
      -((a *om^2*sin(om*t))/vx)];

%sinteza regulatora
A = [0, 0, 0, 0, 0;
     0, 0, 0, 0, 0;
     0, ud(1), 0, 0, 0;
     0, 0, 0, 0, 0;
     0, 0, 0,  ud(1), 0]

B = [1, 0, 0;
```

```
     0, 1, 0;
     xd(2), 0, 0;
     0, 0, 1;
     xd(4), 0, 0]

Q = diag([1e4 1 2e2 1 1e4]);
Rl = diag([1e3 1 1]);
K = lqr(A, B, Q, Rl);

%zakon upravljanja
ut = ud - K*(x(1:5)-xd);

u = ud + K*(x(1:5)-xd);

%% Ulančani model
dx(1,1) = ut(1);
dx(2,1) = ut(2);
dx(3,1) = xd(2)*ut(1)+(x(2)-xd(2))*ud(1);
dx(4,1) = ut(3);
dx(5,1) = xd(4)*ut(1)+(x(4)-xd(4))*ud(1);

dx(6,1) = u(1);
dx(7,1) = u(2);
dx(8,1) = u(3);


%%---------------------------------
%% Nelinearni kinematički model

xx = x(9); y = x(10); theta = x(11); delta1 = x(12); delta2 = x(13);

xw1 = 0.1125; xw2 = -0.1125; yw1 = 0.1125; yw2 = 0.1125;

c1 = cos(delta1+theta); c2 = cos(delta2+theta);
s1 = sin(delta1+theta); s2 = sin(delta2+theta);

B1 = xw1*cos(delta1)/(xw1^2+yw1^2); B2 = xw2*cos(delta2)/(xw2^2+yw2^2);
```

```
D1 = xw1*sin(delta1)/(xw1^2+yw1^2); D2 = xw2*sin(delta2)/(xw2^2+yw2^2);

Kk = 1+cos(delta1+delta2+2*theta);

%transformacijske varijable
xx1 = xx;
xx2 = (D1+D2)/(c1+c2);
xx3 = theta;
xx4 = tan((delta1+delta2)/2+theta);
xx5 = y;

XX = [xx1; xx2; xx3; xx4; xx5];

uu = ud + K*(xd-XX);

v1 = 2*uu(1)/(c1+c2);
%v1 = 0.13/(1+exp(-10*v1));

omega1 = (B2*(2*uu(1)*(D1+D2)-Kk*uu(3)*(c1+c2)) + uu(2)*(c1+c2)^2)/...
    ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) +...
    ((D1+D2)*(-Kk*s2*uu(3)*(c1+c2)-(uu(1)*(D1+D2)*(s1-s2))))/...
    ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)));

omega2 = (B1*(-2*uu(1)*(D1+D2)+Kk*uu(3)*(c1+c2)) - uu(2)*(c1+c2)^2)/...
    ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) +...
    ((D1+D2)*(Kk*s1*uu(3)*(c1+c2)-(uu(1)*(D1+D2)*(s1-s2))))/...
    ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)));

dx(9,1) = 0.5*(cos(delta1+theta)+cos(delta2+theta))*v1;
dx(10,1) = 0.5*(sin(delta1+theta)+sin(delta2+theta))*v1;
dx(11,1) = (xw1*sin(delta1)/(2*xw1^2+2*yw1^2)+...
    xw2*sin(delta2)/(2*xw2^2+2*yw2^2))*v1;% theta
dx(12,1) = omega1;% delta1
dx(13,1) = omega2;% delta2

dx(14,1) = v1;
```

A.4.2 Izvršni kod

```
close all
clear all
clc

global a om N vm tau Ksin

t_init = 0; % pocetno vrijeme
t_final = 45; % konacno vrijeme
N = 1e3; % broj vremenskih intervala
tau = (t_final - t_init)/N; % vremenski korak

a =  0.2;
om = 0.45;
vm = 0.1;

%diskretizacija funkcijom my_euler
x0 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
[T, X] = my_euler(@chained_sin, [t_init t_final], x0, tau);

%referenca
XR = vm*T;
YR = a*sin(om*T)
ThetaR =  atan2(a*om*cos(om*T),vm);

% upravljačke varijable u ulančanom modelu
Leng = length(T);
U1(1) = x0(6);
U2(1) = x0(7);
U3(1) = x0(8);
for i=1:(Leng-1)
    U1(i+1)=(X(i+1,6)-X(i,6))/(T(i+1)-T(i));
    U2(i+1)=(X(i+1,7)-X(i,7))/(T(i+1)-T(i));
    U3(i+1)=(X(i+1,8)-X(i,8))/(T(i+1)-T(i));
end

% upravljačke varijable u nelinearnom kinematičkom modelu
```

```
Leng = length(T);
V1(1) = x0(14);
Omega1(1) = x0(12);
Omega2(1) = x0(13);
for i=1:(Leng-1)
    V1(i+1)=(X(i+1,14)-X(i,14))/(T(i+1)-T(i));
    Omega1(i+1)=(X(i+1,12)-X(i,12))/(T(i+1)-T(i));
    Omega2(i+1)=(X(i+1,13)-X(i,13))/(T(i+1)-T(i));
end


figure(1)
plot(X(:,1), X(:,5), X(:,9), X(:,10), 'k', XR, YR, '--r')
legend('ulančani model', 'nelinearni kinematički model', 'referenca')
xlabel('X [m]')
ylabel('Y [m]')
axis equal

figure(2)
subplot(311)
plot(T, X(:,1), T, XR, '--r')
xlabel('t [s]')
ylabel('x, [m]')
grid minor

subplot(312)
plot(T, X(:,5), T, YR, '--r')
xlabel('t [s]')
ylabel('y, [m]')
grid minor

subplot(313)
plot(T, X(:,3), T, ThetaR, '--r')
xlabel('t [s]')
ylabel('theta, [rad]')
grid minor
```

```
figure(4)
subplot(311)
plot(T, U1)
xlabel('t [s]')
ylabel('u_1 [m/s]')
grid minor

subplot(312)
plot(T, U2)
xlabel('t [s]')
ylabel('u_2 [rad/(ms)]')
grid minor

subplot(313)
plot(T, U3)
xlabel('t [s]')
ylabel('u_3 [rad/s]')
grid minor

a = axes;
tit1 = title('Control variables from chained model');
a.Visible = 'off';
tit1.Visible = 'on';

figure(5)
subplot(311)
plot(T, V1)
xlabel('t [s]')
ylabel('v_1 [m/s]')
grid minor

subplot(312)
plot(T, Omega1)
xlabel('t [s]')
ylabel('\omega_1 [rad/s]')
grid minor
```

```
subplot(313)
plot(T, Omega2)
xlabel('t [s]')
ylabel('\omega_2 [rad/s]')
grid minor

a = axes;
tit1 = title('Control variables for normal kinematic model');
a.Visible = 'off';
tit1.Visible = 'on';


figure(6)
subplot(231)
plot(T, X(:,9))
xlabel('t [s]')
ylabel('x [m]')
grid minor

subplot(232)
plot(T, X(:,10))
xlabel('t [s]')
ylabel('y [m]')
grid minor

subplot(233)
plot(T, X(:,11))
xlabel('t [s]')
ylabel('\theta [rad]')
grid minor

subplot(234)
plot(T, X(:,12))
xlabel('t [s]')
ylabel('\delta_1 [rad]')
grid minor
```

```
subplot(235)
plot(T, X(:,13))
xlabel('t [s]')
ylabel('\delta_2 [rad]')
grid minor

a = axes;
tit1 = title('Robot variables for normal kinematic model');
a.Visible = 'off';
tit1.Visible = 'on';
```

B ROS upravljački čvorovi

B.1 Upravljački čvor `wcr_ns_controller.py`

U ovome su kodu podaci koji daju rezultate prvog mjerenja.

```
#!/usr/bin/env python3

import rospy
from std_msgs.msg import String
from sensor_msgs.msg import JointState
from geometry_msgs.msg import Twist
import math
import numpy as np
import pandas as pd



x = 0.0
y = 0.0
th = 0.0

xw1 = 0.1125
xw2 =-0.1125
yw1 = 0.1125
yw2 = 0.1125

v_max = 0.13
Y = 0.5
xc = 2.25
s = 1.0

# učitavanje vremenski varijabilne matrice
X  = pd.read_csv('catkin_wcr/src/wcr_controller/tablica_006.txt', sep="\t", \
    header=None)
del X[0]
K=np.ones((3000, 3, 5))
```

```
for i in range(3000):
  for j in range(1,16):
    if j<6:
      K[i][0][j-1]=X[j][i]
    elif j>5 and j<11:
      K[i][1][j-6]=X[j][i]
    else:
      K[i][2][j-11]=X[j][i]


q = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]


#callback funkcija
def jointStateCallback(data):
        #očitana stanja s motora robota

  global v_x, v_y, omega, delta1, delta2

  q[0] = data.velocity[5]
  q[1] = data.velocity[1]
  q[2] = data.velocity[3]
  q[3] = data.velocity[7]
  q[4] = data.position[4]
  q[5] = data.position[0]
  q[6] = data.position[2]
  q[7] = data.position[6]

  xw1 = 0.1125
  xw2 =-0.1125
  yw1 = 0.1125
  yw2 = 0.1125

  x_w_r = [xw1, xw2, xw2, xw1]
  y_w_r = [yw1, yw2, -yw2, -yw1]

  r = 0.0254
```

```
  W = [0.0,0.0,0.0,0.0]

  v_x = 0.0
  v_y = 0.0
  omega = 0.0
  delta1 = 0.0
  delta2 = 0.0


  for i in range(4):

    v_x = v_x + (math.cos(q[i+4])/4)*(q[i]*r)


    v_y = v_y + (math.sin(q[i+4])/4)*(q[i]*r)


    W[i] = (-y_w_r[i]*math.cos(q[i+4]) + x_w_r[i]*math.sin(q[i+4]))/ \
        (4.0*math.pow(x_w_r[i], 2) + 4.0*math.pow(y_w_r[i], 2))


    omega = omega + W[i]*q[i]*r

  delta1 = q[4]

  delta2 = q[5]


v_x = 0.0
v_y = 0.0
omega = 0.0
delta1 = 0.0
delta2 = 0.0


#controller funkcija
def controller():
```

```
rospy.init_node('controller', anonymous=True)
global v_x, v_y, omega, delta1, delta2

joint_states = rospy.Subscriber("/wcr/joint_states", JointState, \
  jointStateCallback)
cmd_vel_pub = rospy.Publisher("/wcr/cmd_vel", Twist, queue_size = 10)
T_d = 0.045 #vrijeme diskretizacija u sekundama
rate = rospy.Rate(1/T_d)


#nuliramo vrijeme koje koristimo kao početak rada
normalized_time = 0.0

current_time = rospy.Time.now()
current_time = current_time.to_sec()
last_time = rospy.Time.now()
last_time = last_time.to_sec()

x = 0.0
y = 0.0
th = 0.0

while not rospy.is_shutdown():

  current_time = rospy.Time.now()
  current_time = current_time.to_sec()

  dt  = (current_time - last_time)

        #promjena x,y i theta pozicije u globalnom ks

  delta_x = (v_x * math.cos(th) - v_y * math.sin(th)) * dt
  delta_y = (v_x * math.sin(th) + v_y * math.cos(th)) * dt
  delta_th = omega * dt

  x = x + delta_x
  y = y + delta_y
```

```
th = th + delta_th

normalized_time = normalized_time + dt

v_max = 0.13

#referenca

xd = np.zeros((5,1))
xd[0] = v_max *normalized_time
xd[1] = (2.0*math.exp(s*(-normalized_time*v_max + xc)**2.0)*s*\
    (-1.0+2.0*s*(-normalized_time*v_max+xc)**2.0)*Y)/ \
    (math.exp(2.0*s*(-normalized_time*v_max+xc)**2.0)+4.0*math.pow(s, 2)* \
    (-normalized_time*v_max+xc)**2.0*Y**2.0)
xd[2] = math.atan2(2.0*math.exp(-s*(xc-v_max*normalized_time)**2.0)*s*Y* \
    (xc-v_max*normalized_time),1)
xd[3] = -2.0*math.exp(-s*(-xc+v_max*normalized_time)**2.0)*s*Y* \
    (-xc+v_max*normalized_time)
xd[4] = Y*math.exp(-s*(v_max*normalized_time-xc)**2.0)

#stvarna brzina
v_max = v_max/math.sqrt(4.0*s**2.0*xd[4]**2.0*(xd[0]-xc)**2.0+1.0)

ud = np.zeros((3,1))
ud[0] = v_max
ud[1] = -((4.0*math.exp(s*(-normalized_time*v_max+xc)**2.0)*s**2.0*v_max* \
    (normalized_time*v_max-xc)*Y* \
    (math.exp(2.0*s*(-normalized_time*v_max+xc)**2.0)* \
    (-3.0+2.0*s*(-normalized_time*v_max+xc)**2.0)-4.0*s*\
    (1.0+s*(-normalized_time*v_max+xc)**2.0*\
    (-1.0+2.0*s*(-normalized_time*v_max+xc)**2.0))*Y**2.0))/ \
    (math.exp(2.0*s*(-normalized_time*v_max+xc)**2.0)+ \
    4.0*s**2.0*(-normalized_time*v_max+xc)**2.0*Y**2.0)**2.0)
ud[2] = 2.0*math.exp(-s*(-normalized_time*v_max+xc)**2.0)*s*v_max* \
    (-1.0+2.0*s*(-normalized_time*v_x+xc)**2.0)*Y

c1 = math.cos(delta1+th)
```

```
c2 = math.cos(delta2+th)
s1 = math.sin(delta1+th)
s2 = math.sin(delta2+th)

B1 = xw1*math.cos(delta1)/(xw1**2.0+yw1**2.0)
B2 = xw2*math.cos(delta2)/(xw2**2.0+yw2**2.0)
D1 = xw1*math.sin(delta1)/(xw1**2.0+yw1**2.0)
D2 = xw2*math.sin(delta2)/(xw2**2.0+yw2**2.0)

Kk = 1.0+math.cos(delta1+delta2+2.0*th)

#transformacijske varijable
xx = np.zeros((5,1))
xx[0] = x
xx[1] = (D1+D2)/(c1+c2)
xx[2] = th
xx[3] = math.tan((delta1+delta2)/2.0+th)
xx[4] = y

t = int(normalized_time/T_d)

#vremenski varijabilna matrica u trenutku t
Kt = K[t]

ud1=ud[0][0]
xd2=xd[1][0]
xd4=xd[3][0]


#greška
e_x = xd-xx

#zakon upravljanja
u =  ud + np.matmul(Kt, e_x)

v_1 = 2.0*u[0]/(c1+c2)
```

```
    #ograničenje brzine na 0.13 m/s
    v_1 = 0.13* math.tanh(v_1 * 2.5)

    om_1 = (B2* (2.0*u[0]*(D1+D2)-Kk*u[2] *(c1+c2)) + u[1] *(c1+c2)**2.0)/ \
        ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) \
        +((D1+D2)*(-Kk*s2*u[2] *(c1+c2)-(u[0] *(D1+D2)*(s1-s2))))/ \
        ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)))

    om_2 = (B1*(-2.0*u[0]*(D1+D2)+Kk*u[2] *(c1+c2)) - u[1] *(c1+c2)**2.0)/ \
        ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) \
        + ((D1+D2)*(Kk*s1*u[2] *(c1+c2)-(u[0] *(D1+D2)*(s1-s2))))/ \
        ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)))

    #promjena kuteva zakreta
    delta_c_1 = om_1 *dt + delta1
    delta_c_2 = om_2 *dt + delta2

    #nove upravljačke varijable
    v_x_c = 0.5* (math.cos(delta_c_1+th)+math.cos(delta_c_2+th)) *v_1

    v_y_c = 0.5* (math.sin(delta_c_1+th)+math.sin(delta_c_2+th)) *v_1

    #objavljivanje upravljačkih varijabli
    cmd_vel_msg = Twist()
    cmd_vel_msg.linear.x = v_x_c
    cmd_vel_msg.linear.y = v_y_c

    cmd_vel_pub.publish(cmd_vel_msg)

    last_time = rospy.Time.now()
    last_time = last_time.to_sec()

    rate.sleep()


if __name__ == '__main__':
    controller()
```

B.2 Upravljački čvor wcr_ns_controller3.py

U ovom se kodu nalazi upravljački čvor koji daje podatke šestog mjerenja (konstantna matrica **K**.)

```
#!/usr/bin/env python3

import rospy
from std_msgs.msg import String
from sensor_msgs.msg import JointState
from geometry_msgs.msg import Twist
import math
import numpy as np
import pandas as pd



global x, y, th, normalized_time
x = 0.0
y = 0.0
th = 0.0

xw1 = 0.1125
xw2 =-0.1125
yw1 = 0.1125
yw2 = 0.1125



v_max = 0.06
Y = 0.4
xc = 1.5
s = 3.0

normalized_time=0.0

## promjenjiva matrica
# X  = pd.read_csv('catkin_wcr/src/wcr_controller/tablica.txt', sep="\t", \
    header=None)
# del X[0]
```

```
# K=np.ones((3000, 3, 5))
# for i in range(3000):
#   for j in range(1,16):
#     if j<6:
#       K[i][0][j-1]=X[j][i]
#     elif j>5 and j<11:
#       K[i][1][j-6]=X[j][i]
#     else:
#       K[i][2][j-11]=X[j][i]

q = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]


def jointStateCallback(data):
        #očitana stanja s motora robota

  global v_x, v_y, omega, delta1, delta2, th, x, y, normalized_time

  q[0] = data.velocity[5]
  q[1] = data.velocity[1]
  q[2] = data.velocity[3]
  q[3] = data.velocity[7]
  q[4] = data.position[4]
  q[5] = data.position[0]
  q[6] = data.position[2]
  q[7] = data.position[6]

  xw1 = 0.1125
  xw2 =-0.1125
  yw1 = 0.1125
  yw2 = 0.1125

  x_w_r = [xw1, xw2, xw2, xw1]
  y_w_r = [yw1, yw2, -yw2, -yw1]

  r = 0.0254
  W = [0.0,0.0,0.0,0.0]
```

```
v_x = 0.0
v_y = 0.0
omega = 0.0
delta1 = 0.0
delta2 = 0.0


for i in range(4):

  v_x = v_x + (math.cos(q[i+4])/4)*(q[i]*r)


  v_y = v_y + (math.sin(q[i+4])/4)*(q[i]*r)


  W[i] = (-y_w_r[i]*math.cos(q[i+4]) + x_w_r[i]*math.sin(q[i+4]))/ \
      (4.0*math.pow(x_w_r[i], 2) + 4.0*math.pow(y_w_r[i], 2))


  omega = omega + W[i]*q[i]*r


delta1 = q[4]

delta2 = q[5]

T_d = 0.016 #vrijeme diskretizcije

#promjena x,y i theta pozicije u globalnom ks

delta_x = (v_x * math.cos(th) - v_y * math.sin(th)) * T_d
delta_y = (v_x * math.sin(th) + v_y * math.cos(th)) * T_d
delta_th = omega * T_d

x = x + delta_x
y = y + delta_y
```

```
th = th + delta_th


normalized_time = normalized_time + T_d

#nominalna brzina
v_max = 0.06

#referenca

xd = np.zeros((5,1))
xd[0] = v_max *normalized_time
xd[1] = (2.0*math.exp(s*(-normalized_time*v_max + xc)**2.0)*s*\
  (-1.0+2.0*s*(-normalized_time*v_max+xc)**2.0)*Y)/ \
  (math.exp(2.0*s*(-normalized_time*v_max+xc)**2.0)+4.0*math.pow(s, 2)* \
  (-normalized_time*v_max+xc)**2.0*Y**2.0)
xd[2] = math.atan2(2.0*math.exp(-s*(xc-v_max*normalized_time)**2.0)*s*Y* \
  (xc-v_max*normalized_time),1)
xd[3] = -2.0*math.exp(-s*(-xc+v_max*normalized_time)**2.0)*s*Y* \
  (-xc+v_max*normalized_time)
xd[4] = Y*math.exp(-s*(v_max*normalized_time-xc)**2.0)

#stvarna brzina
v_max = v_max/math.sqrt(4.0*s**2.0*xd[4]**2.0*(xd[0]-xc)**2.0+1.0)

ud = np.zeros((3,1))
ud[0] = v_max
ud[1] = -((4.0*math.exp(s*(-normalized_time*v_max+xc)**2.0)*s**2.0*v_max* \
    (normalized_time*v_max-xc)*Y* \
    (math.exp(2.0*s*(-normalized_time*v_max+xc)**2.0)* \
    (-3.0+2.0*s*(-normalized_time*v_max+xc)**2.0)-4.0*s*\
    (1.0+s*(-normalized_time*v_max+xc)**2.0*\
    (-1.0+2.0*s*(-normalized_time*v_max+xc)**2.0))*Y**2.0))/ \
    (math.exp(2.0*s*(-normalized_time*v_max+xc)**2.0)+ \
    4.0*s**2.0*(-normalized_time*v_max+xc)**2.0*Y**2.0)**2.0)
ud[2] = 2.0*math.exp(-s*(-normalized_time*v_max+xc)**2.0)*s*v_max* \
    (-1.0+2.0*s*(-normalized_time*v_x+xc)**2.0)*Y
```

```
c1 = math.cos(delta1+th)
c2 = math.cos(delta2+th)
s1 = math.sin(delta1+th)
s2 = math.sin(delta2+th)

B1 = xw1*math.cos(delta1)/(xw1**2.0+yw1**2.0)
B2 = xw2*math.cos(delta2)/(xw2**2.0+yw2**2.0)
D1 = xw1*math.sin(delta1)/(xw1**2.0+yw1**2.0)
D2 = xw2*math.sin(delta2)/(xw2**2.0+yw2**2.0)

Kk = 1.0+math.cos(delta1+delta2+2.0*th)

#transformacijske varijable
xx = np.zeros((5,1))
xx[0] = x
xx[1] = (D1+D2)/(c1+c2)
xx[2] = th
xx[3] = math.tan((delta1+delta2)/2.0+th)
xx[4] = y


t = int(normalized_time/T_d)

#konstantna matrica
Kt = np.array([[3.1588, 0.0, 0.00, -0.0063, -1.4876], \
  [ -0.0828, 1.0583, 1.0,         0.000, 0.0001], \
  [ 4.7041, 0.000,  0.000, 10.9918,  998.8930]])

e_x = xd-xx

#zakon upravljanja
u =  ud + np.matmul(Kt, e_x)

v_1 = 2.0*u[0]/(c1+c2)
```

```
    om_1 = (B2* (2.0*u[0]*(D1+D2)-Kk*u[2] *(c1+c2)) + u[1] *(c1+c2)**2.0)/ \
           ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) \
           +((D1+D2)*(-Kk*s2*u[2] *(c1+c2)-(u[0] *(D1+D2)*(s1-s2))))/ \
           ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)))

    om_2 = (B1*(-2.0*u[0]*(D1+D2)+Kk*u[2] *(c1+c2)) - u[1] *(c1+c2)**2.0)/ \
           ((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)) \
           + ((D1+D2)*(Kk*s1*u[2] *(c1+c2)-(u[0] *(D1+D2)*(s1-s2))))/ \
           ((c1+c2)*((B1-B2)*(c1+c2)+(D1+D2)*(s1-s2)))

    #promjena kuteva zakreta
    delta_c_1 = om_1 *T_d + delta1
    delta_c_2 = om_2 *T_d + delta2

    #nove upravljačke varijable
    v_x_c = 0.5* (math.cos(delta_c_1+th)+math.cos(delta_c_2+th)) *v_1
    v_y_c = 0.5* (math.sin(delta_c_1+th)+math.sin(delta_c_2+th)) *v_1

    #objavljivanje upravljačkih varijabli
    cmd_vel_msg = Twist()
    cmd_vel_msg.linear.x = v_x_c
    cmd_vel_msg.linear.y = v_y_c

    cmd_vel_pub.publish(cmd_vel_msg)


if __name__ == '__main__':
    rospy.init_node('controller', anonymous=True)
    joint_states = rospy.Subscriber("/wcr/joint_states", JointState,\
      jointStateCallback)
    cmd_vel_pub = rospy.Publisher("/wcr/cmd_vel", Twist, queue_size = 10)
    T_d = 0.016 #vrijeme diskretizacija u sekundama
    rate = rospy.Rate(1/T_d)

    while not rospy.is_shutdown():
       rate.sleep()
```

C Generiranje grafova u MATLAB-u

U nastavku je prikazan kod za generiranje grafa za šesto mjerenje.

```
close all
clear all
clc

s = 3;
Y =  0.4;
xc = 1.5;
vm = 0.06;

%učitavanje vrijednosti mjerenih OptiTrackom
pose1= readtable('konstantna-vrpn_client_node-wcr-pose.csv');

time_pose = pose1.x_header_stamp_secs + pose1.x_header_stamp_nsecs *1e-9 ;
time_pose = time_pose - time_pose(1);

x = pose1.x_pose_position_x;
y = pose1.x_pose_position_y;
th = pose1.x_pose_orientation_z;

%učitavanje vrijednosti s enkodera
joint_states = readtable('joint_call.csv');

time = joint_states.x_header_stamp_secs + joint_states.x_header_stamp_nsecs *1e-9;
time = time -time(1);

delta1=joint_states.delta_1;
delta2=joint_states.delta_2;
delta3=joint_states.delta_3;
delta4=joint_states.delta_4;
```

```
v1=joint_states.v_1*0.0254;
v2=joint_states.v_2*0.0254;
v3=joint_states.v_3*0.0254;
v4=joint_states.v_4*0.0254;

x_joint = zeros(length(time));
y_joint = zeros(length(time));

for i=2:length(time)
    x_joint(i)= x_joint(i-1)+ 0.25*(time(i)-time(i-1))* ...
        (v1(i)*cos(delta1(i))+v2(i)*cos(delta2(i))+ ...
        v3(i)*cos(delta3(i))+v4(i)*cos(delta4(i)));
    y_joint(i)= y_joint(i-1)+ 0.25*(time(i)-time(i-1))*...
        (v1(i)*sin(delta1(i))+v2(i)*sin(delta2(i))+ ...
        v3(i)*sin(delta3(i))+v4(i)*sin(delta4(i)));
end

%referenca
XR = vm*time;
YR = Y*exp(-s*(vm*time- xc).^2);
ThetaR =  atan((-vm*time+ xc)*2.*exp(-s*(-vm*time +xc).^2).*s*Y );

%greška
e_x = XR - x_joint;
e_y = YR - y_joint;

%učitavanje urpavljačkih vrijednosti
cmd_vel = readtable('konstantna-wcr-cmd_vel.csv');

v_x = cmd_vel.x_linear_x;
v_y = cmd_vel.x_linear_y;

v_1 = zeros(length(v_x));
time_vel = cmd_vel.time;
time_vel = time_vel - time_vel(1);
```

```
for i=1: length(v_x)
    v_1(i) = sqrt((v_x(i))^2+(v_y(i))^2);
end

figure(1)
plot(x,y, x_joint, y_joint, 'k', XR, YR, '--r')
legend('vrijednosti dobivene OptiTrackom', 'vrijednosti očitane s enkodera')
xlabel('X [m]')
ylabel('Y [m]')
axis equal

figure(2)
subplot(211)
plot(time_pose, x, time, x_joint, 'k', time, XR, '--r')
xlabel('t [s]')
ylabel('x [m]')
grid minor

subplot(212)
plot(time_pose, y, time, y_joint, 'k', time, YR, '--r')
xlabel('t [s]')
ylabel('y [m]')
grid minor

legend('OptiTrack', 'enkoder')

figure(3)
plot(time, v1)
xlabel('t [s]')
ylabel('v_1 [m/s]')
grid minor

figure(4)
subplot(311)
plot(time_vel, v_x)
xlabel('t [s]')
ylabel('v_x [m/s]')
```

```
grid minor

subplot(312)
plot(time_vel, v_y)
xlabel('t [s]')
ylabel('v_y [m/s]')
grid minor

subplot(313)
plot(time_vel, v_1)
xlabel('t [s]')
ylabel('v_1 [m/s]')
grid minor

figure(5)
subplot(211)
plot(time, e_x)
xlabel('t [s]')
ylabel('e_x, [m]')
grid minor

subplot(212)
plot(time, e_y)
xlabel('t [s]')
ylabel('e_y, [m]')
grid minor
```