Usklađeno dvoručno robotsko sklapanje

Šarić, Ante

Master's thesis / Diplomski rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:706912

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-27**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Ante Šarić

Zagreb, 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Dr. sc. Bojan Jerbić, dipl. ing.
Dr. sc. Zdenko Kovačić, dipl. ing.

Student:

Ante Šarić

Zagreb, 2015.

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

Ovom prilikom bi želio zahvaliti: Voditelju rada prof.dr.sc. Bojanu Jerbiću, prof.dr.sc. Zdenku Kovačiću i dr.sc. Marku Švaci na stručnim savjetima i pomoći tijekom izrade diplomskog rada.

Posebno bi želio zahvaliti svojoj obitelji na razumijevanju, potpori i pomoći tijekom izrade ovog rada, tako i cijelog studija.

Ante Šarić

PRAZNO ZA DIPLOMSKI ZADATAK

Sadržaj

Popis slika III

Popis tablica V

Popis oznaka VI

Popis sistemskih varijabli VI

Sažetak VII

Summary VIII

1 Uvod 1

1.1 Industrijski roboti 1

1.2 PID regulator 2

2 Opis sustava 3

2.1 LR Mate 200iC [5] 3

2.2 M-10iA [6] 5

3 Mjerenje odziva preko TCP/IP protokola i digitalnih signala 8

3.1 Mjerenje odziva preko TCP/IP protokola bez kretanja robota i pokretanja paralelnih programa 8

3.2 Mjerenje odziva preko TCP/IP protokola sa kretanjem robota i s pokrenutim paralelnim programima 12

3.3 Mjerenje odziva preko digitalnih signala 13

4 Praćenje robota [7] 16

4.1 Radni prostor 16

4.2 Kalibracija koordinatnih sustava robota i kalibracija vrha alata 17

4.2.1 Kalibracija vrha alata 18

4.2.2 Kalibracija zajedničkih koordinatnih sustava 18

4.3 Program za kretanje master robota 20

4.4 Program za kretanje *slave* robota 22
4.5 Program za regulaciju 24
5 Regulacija 27
5.1 PI regulator 27
5.2 PD regulator 30
5.3 PID regulator 32
5.4 Testiranje PID regulatora na većim brzinama 35
5.4.1 100 mm/s 35
5.4.2 150 mm/s 38
5.4.3 200 mm/s 40
5.4.4 300 mm/s 42
6 Provjera sa vizijskim sustavom 44
6.1 TP program za mjerenje greške 50
7 Testiranje spajanja 52
8 Zaključak 54
Prilozi 56
Literatura 57
DODATAK A 58
DODATAK B 62
DOCATAK C 66
DODATAK D 69
DODATAK E 72
DODATAK F 76
DODATAK G 81
DODATAK H 87

Popis slika

Slika 1: Industrijski robot sa 6 osi 1
Slika 2: Dimenzije LR Mate 200iC/5L robota 3
Slika 3: Dimenzije M-10iA robota 5
Slika 4: LR Mate 200iC/5L i M-10iA roboti 7
Slika 5: Sustav s dva robota u Laboratoriju za projektiranje izradbenih i montažnih sustava 7
Slika 6: Diagram toka mjerenja odziva preko TCP/IP protokola bez kretanja robota 10
Slika 7: Odziv mjeren preko TCP/IP-a bez kretanje robota 11
Slika 8: Odziv mjeren preko TCP/IP-a sa kretanjem robota 12
Slika 9: Diagram toka mjerenja odziva preko TCP/IP protokola bez kretanja robota 14
Slika 10: Odziv mjeren preko digitalnih signala bez kretanje robota 15
Slika 11: Tlocrt konfiguracije robota 16
Slika 12: Konfiguracija robota u prostoru 17
Slika 13: Kalibracija vrha alata robota 18
Slika 14: Smještanje robota u isti koordinatni sustav 19
Slika 15: Diagram toka programa *master* robota 21
Slika 16:Diagram toka programa za kretanje *slave* robota 23
Slika 17: Diagram toka programa za regulaciju brzine 26
Slika 18: Brzina *slave* robota kod PI regulatora na 50 mm/s brzine *master* robota 27
Slika 19: Greška *slave* robota kod PI regulatora na 50 mm/s brzine *master* robota 28
Slika 20: Putanje *master* i *slave* robota kod PI regulatora po y osi i putanja *slave* robota po x osi za vrijeme spajanja pri brzini *master* robota od 50 mm/s 28
Slika 21: Greška PI regulatora u trenutku spajanja 29
Slika 22: Brzina slave robota kod PD regulatora na 50 mm/s brzine master robota 30
Slika 23: Greška slave robota kod PD regulatora na 50 mm/s brzine master robota 30
Slika 24: Putanje *master* i *slave* robota kod PD regulatora po y osi i putanja *slave* robota po x osi za vrijeme spajanja pri brzini *master* robota od 50 mm/s 31
Slika 25: Brzina slave robota kod PD regulatora na 50 mm/s brzine master robota 32
Slika 26: Greška slave robota kod PID regulatora na 50 mm/s brzine master robota 32
Slika 27: Putanje *master* i *slave* robota kod PID regulatora po y osi i putanja *slave* robota po x osi za vrijeme spajanja pri brzini master robota od 50 mm/s 33
Slika 28: Greška PID regulatora u trenutku spajanja 33

Slika 29: Brzina slave robota kod PD regulatora na 100 mm/s brzine master robota 35
Slika 30: Greška slave robota kod PID regulatora na 100 mm/s brzine master robota 36
Slika 31: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 100 mm/s 36
Slika 32: Greška za vrijeme spajanja na 100 mm/s 37
Slika 33: Brzina slave robota kod PD regulatora na 150 mm/s brzine master robota 38
Slika 34: Greška slave robota kod PID regulatora na 150 mm/s brzine master robota 38
Slika 35: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 150 mm/s 39
Slika 36: Greška za vrijeme spajanja na 150 mm/s 39
Slika 37: Brzina slave robota kod PD regulatora na 200 mm/s brzine master robota 40
Slika 38: Greška slave robota kod PID regulatora na 200 mm/s brzine master robota 40
Slika 39: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 200 mm/s 41
Slika 40: Greška za vrijeme spajanja na 200 mm/s 41
Slika 41: Brzina slave robota kod PD regulatora na 300 mm/s brzine master robota 42
Slika 42: Greška slave robota kod PID regulatora na 300 mm/s brzine master robota 42
Slika 43: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 300 mm/s 43
Slika 44: Očitanje prve vrijednosti 45
Slika 45: Očitanje druge vrijednosti 46
Slika 46: Greška sustava dobivena preko vizijskog procesa 47
Slika 47: Brzina robota 48
Slika 48: Greška robota 48
Slika 49: Putanja *slave* i *master* robota 49
Slika 50: Greška za vrijeme spajanja –podatci iz robota 49
Slika 51: Diagram toga TP programa 51
Slika 52: Predmeta za spajanje 52
Slika 53: Greška u odnosu na poziciju x osi za vrijeme spajanja 52

Popis tablica

Tabela 1: Specifikacije LR Mate 200iC/5L robota 4
Tabela 2: Specifikacije M-10iA robota 6
Tabela 3: Rezultati mjerenja odziva preko TCP/IP-a bez kretanja robota 11
Tabela 4: Rezultati mjerenja odziva preko TCP/IP-a sa kretanjem robota 12
Tabela 5:Rezultati mjerenja odziva preko digitalnih signala bez kretanja robota 15
Tabela 6: Usporedba rezultata mjerenja odziva signala 15
Tabela 7: Utjecaj nezavisnog pove canja parametara 27
Tabela 8: Mjerenja greške u trenutku spajanja PI regulatora 29
Tabela 9: Mjerenja greške u trenutku spajanja PID regulatora 34
Tabela 10: Usporedba grešaka za vrijeme spajanja PI i PID 34
Tabela 11: Usporedba rezultata vizijskog sutava i podataka iz robota 50

Popis oznaka

Oznaka	Jedinica	Opis
Kp	-	Proporcijonalno pojačanje
Ki	-	Integracijsko pojačanje
Kd	-	Derivacijsko pojačanje
e	mm	Greška
e_1	mm	Greška u prethodnom koraku
T	s	Vrijeme diskretizacije
a_{max}	mm/s^2	Maksimalno ubrzanje
v	mm/s	Brzina robota
v_1	mm/s	Brzina robota u prethodnom koraku

Popis sistemskih varijabli

Sistemska varijabla	Opis
$SCR.$ITP_TIME	brzina ciklusa procesora u ms [4-124]
$GROUP[1].$SPEED	maksimalna brzina robota u mm/s
$MCR_GRP[1].$PRGOVERRIDE	brzina robota u postotku od $GROUP[1].$SPEED [0-100]
$IDL_CPU_PCT	opterećenje procesora [0-100]
$PARAM_GROUP[1].$SV_OFF_TIME[9]	vrijeme čekanje prije gašenja servo motora u ms

Sažetak

U ovom radu će se obraditi regulacija dva industrijska robota preko TCP/IP protokola pomoću diskretnog PID regulatora koji omogućava montažu u pokretu. Prvo se tražilo vrijeme diskretizacije za regulator koje je dobiveno mjerenjem kašnjenja slanja podatka preko TCP/IP-a. Sustav zbog velikog kašnjenja dobivanja podataka i brzine kretanja robota nije previše pogodan za PID regulaciju. Sustav se testirao s tri vrste regulatora (PI, PD, PID) na brzini od 50 mm/s i na kraju je odabran najoptimalniji regulator (PID) za daljnja testiranja pri većim brzinama.

Iako sustav ima preveliko kašnjenje pri malim brzinama kretanja *master* robota (50 mm/s) regulator se pokazao dovoljno točnim i stabilnim da bi se proces spajanja mogao izvesti, što je i demonstrirano u radu. Nakon toga se izvršio proces nezavisnog mjerenja rezultata preko vizijskog sustava za utvrđivanje stvarne greške sustava i demonstracija spajanja s realnim objektima.

Ključnje riječi: PID regulacija, robotika, tcp/ip, industrijski roboti

Summary

Abstract – In this paper the control of industrial robots with discrete PID controller over TCP/IP protocol for purposes of assembly in motion is shown. First discretization time is found for controller which is acquired by measuring delay of the signal over TCP/IP. Because of large latency of the combination it is shown that it's not very suitable for PID control. The system has then been tested with three controllers (PI, PD, PID) and in the end the most optimal controller has been chosen for further testing at higher speeds.

Although the system has large latency on smaller motion speeds of master robots (50 mm/s) the controller has been show to be precise and stable enough for the process of assembly to be completed. After that the proces of independent measurement using vision system has been done and the demonstration assembly with real object has been demonstrated.

Key words: PID controler, robotics, tcp/ip protocol, industrial robots

1 Uvod

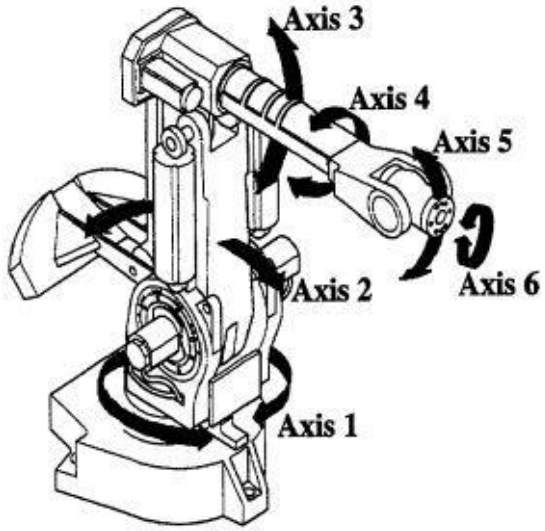
1.1 Industrijski roboti

Industrijski robot je definiran prem ISO 8373 [1] kao automatizirani, reprogramabilni, višenamjenski manipulator kojeg je moguće isprogramirati u tri ili više osi. Sposoban je voditi razne operacije rukovanja (ugradbenim elementima, alatima itd.), ili bilo koje drug operacije posredstvom specijalnih uređaja ugrađenih na završnom dijelu kinematičkog lanca tj. robotske ruke.

Za razliku od uzmi-sastavi uređaja, sposoban je izvoditi najsloženije putanje, i s određenom točnošću, brzinom i orijentacijom doseći bilo koju točku u radnome prostoru bez ikakvog prepravljanja. Upravljan je stoga računalom, koje obrađuje složene kinematičke i dinamičke proračune gibanja pojedinih članaka, čije sinkrono kretanje mora osigurati zadano gibanje vrha robotske ruke.

Robot se sastoji od: baze, ruke, završnog zgloba i izvršnog članka (efektora): hvataljke, alata, zavarivačkog pištolja itd.

Robotska ruka ima definirani radni prostor a nosi završni zglob i izvršni članak kao svoje ekstremitete. [2]



Slika 1: Industrijski robot sa 6 osi

1.2 PID regulator

Proporcionalno-integralno-derivacijski regulator je mehanizam s povratnom petljom i često se koristi u industrijskim sustavima regulacije. PID regulator kontinuirano računa grešku kao razliku između zadane (željene) vrijednosti i varijable procesa. Regulator pokušava minimalizirati tu grešku tijekom vremena namještajući varijablu regulacije koja može biti sve od pozicije ventila, položaja ili u slučaju ovog diplomskog rada brzine.

Općenita jednadžba regulatora je:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}$$

gdje su K_p, K_i i K_d pojačanja regulatora i veća su od 0. u(t) je izlazna varijabla, e je greška sustava.

Proporcionalni dio regulatora predstavlja trenutnu grešku sustava (npr. ako je greška velika i pozitivna, izlaz iz regulatora će biti velik i pozitivan). Integralni dio regulatora predstavlja prijašnje greške (npr. ako izlazna varijabla nije dovoljna da smanji grešku, greška će se nakupiti tijekom vremena i natjerat će sustav da poveća izlaznu varijablu). Derivacijski dio regulatora predviđa buduće greške temeljene na brzini promjene. [3]

Regulator u ovakvom obliku nije primjeren za korištenje na računalu zbog toga što grešku ne dobivamo kontinuirano već određenom frekvencijom. Zbog toga je potrebno dobiti diskretizirani oblik regulatora za primjenu na robotima i računalu.

Općenita jednadžba diskretnog regulatora je:

$$u(T) = K_p e(T) + K_i e(T)q + \frac{K_d}{T}(e - e_1)$$

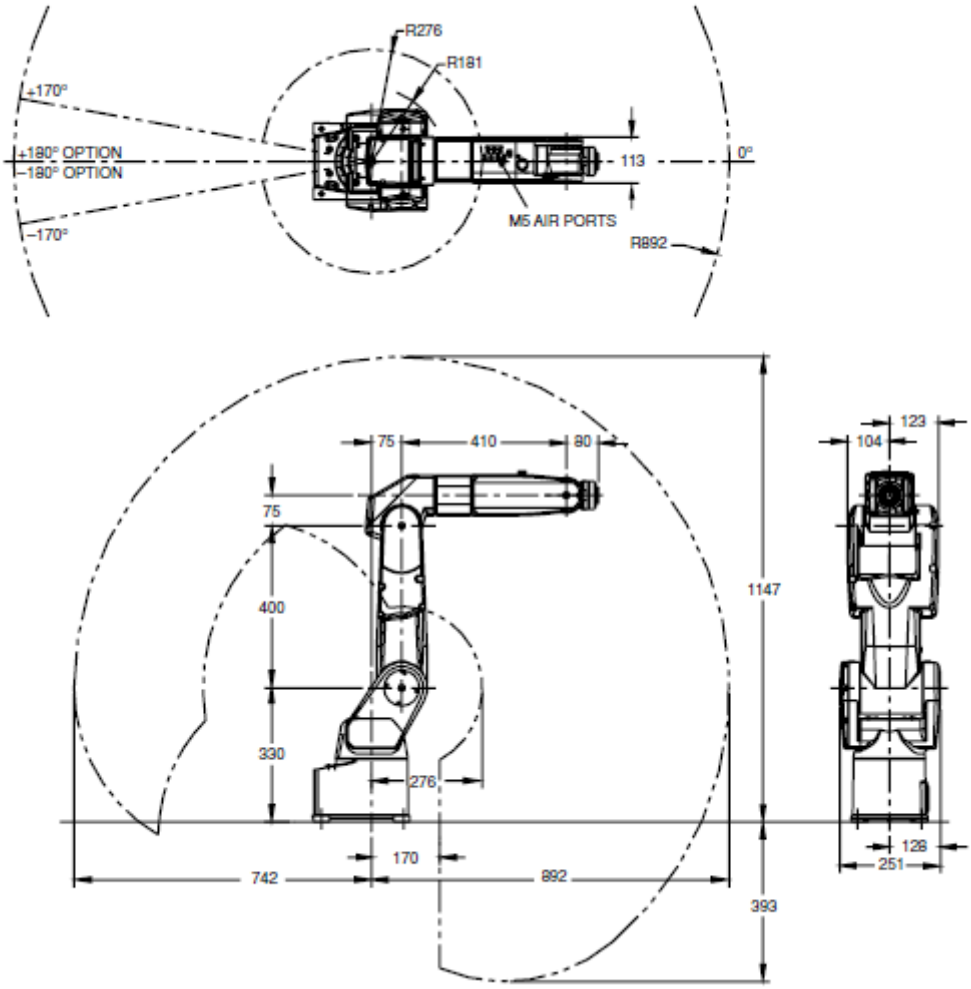
gdje su K_p, K_i i K_d pojačanja regulatora i veća su od 0. $u(T)$ je izlazna varijabla, e je greška sustava, e_1 je greška u prethodnom koraku, T je vrijeme diskretizacije koje je potrebno da se dobije novi podatak za regulaciju, a $q = q + e$, tj. q je akumulirana greška tijekom vremena regulacije. [4]

2 Opis sustava

U diplomskom radu su korištena dva FANUC robota: LR Mate 200iC/5L i M-10iA.

2.1 LR Mate 200iC [5]

LR Mate 200iC/5L serija mini-robota koja nudi najbolje performanse u klasi s laganim, učinkovitim, točnim i okretnim (LEAN – light, efficient, accurate and nimble) paketom. LR Mate 200iC se koristi za rukovanje materijalima, montažu, uzimanje i postavljenje, čišćenje dijelova, odlaganje materijala, testiranje, edukaciju i zabavu. Sastoji se od 6 stupnjeva slobode s ponovljivošću od ±0.03mm pod punim teretom i punoj brzini u cijelom području rada. Nosivost mu je 5kg. Komunikacija s drugim preko Ethernet standarda.



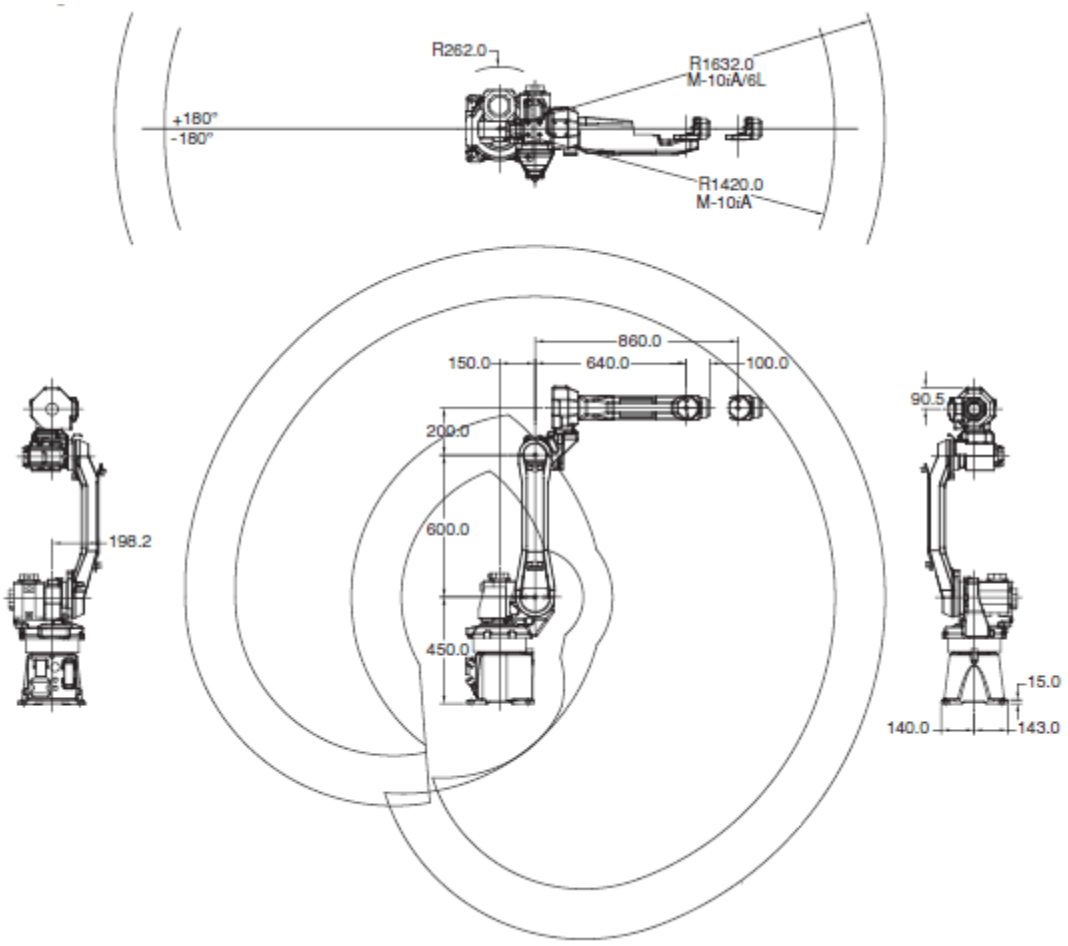
Slika 2: Dimenzije LR Mate 200iC/5L robota

Tabela 1: Specifikacije LR Mate 200iC/5L robota

Osi		6
Nosivost [kg]		5
Doseg [mm]		892
Ponovljivost [mm]		±0.03
radijus interferencije[mm]		181
Domet kretanja [°]	J1	340
	J2	230
	J3	373
	J4	380
	J5	240
	J6	720
Brzine kretanja [°/s]	J1	270
	J2	270
	J3	270
	J4	450
	J5	450
	J6	720
Moment zgloba [kgf*m]	J4	11.9 (1.21)
	J5	11.9 (1.21)
	J6	6.7 (0.68)
Inercija zgloba [kg*m^2]	J4	0.3
	J5	0.3
	J6	0.1
Mehaničke kočnice		J1, J2 & J3
Težina [kg]		26
Način postavljanja		Pod, strop, pod kutom, na zid
Temperatura montaže		0 do 45
Temperatura [°C]		
Vlažnost		Normalno : 75% ili manje
Vibracije m/s^2 [G]		0.5 ili manje
IP ocijena		IP 67

2.2 M-10iA [6]

M-10iA je 6-osni robot teži 130kg s nosivosti od 10kg i s najvećim momentom i inercijom u zglobu u svojoj klasi. Stavljanje kablova i cijevi za protok zraka je olakšano s 50 milimetarskim praznim prostorom kod gornje ruke robota, tako smanjujući trošak sustava i poboljšavajući pouzdanost. M-10iA može biti postavljen na pod, zid pod bilo kojim kutem ili na strop. M-10iA ima najveće brzine osi i najbolju ponovljivost u svojoj klasi čineći ga industrijskim standardom za male robote. Koristi se za rukovanje materijalima, montažu, uzimanje i postavljenje, pranje dijelova, odlaganje materijala, testiranja.



Slika 3: Dimenzije M-10iA robota

Tabela 2: Specifikacije M-10iA robota

Osi		6
Nosivost [kg]		10
Doseg [mm]		1420
Ponovljivost [mm]		±0.08
radijus interferencije[mm]		262
Domet kretanja [°]	J1	340/360
	J2	250
	J3	290 (445)
	J4	380
	J5	380
	J6	720
Brzine kretanja [°/s]	J1	210
	J2	190
	J3	210
	J4	400
	J5	400
	J6	600
Moment zgloba [kgf*m]	J4	22 (2.2)
	J5	22 (2.2)
	J6	9.8 (1.0)
Inercija zgloba [kg*m^2]	J4	0.63
	J5	0.63
	J6	0.15
Mehaničke kočnice		sve osi
Težina [kg]		130
Način postavljanja		Pod, strop, pod kutom, na zid
Temperatura montaže		0 do 45
Temperatura [°C]		
Vlažnost		Normalno : 75% ili manje
Vibracije m/s^2 [G]		4.9 ili manje
IP ocijena		IP 67/IP54

Slika 4: LR Mate 200iC/5L i M-10iA roboti

Slika 5: Sustav s dva robota u Laboratoriju za projektiranje izradbenih i montažnih sustava

3 Mjerenje odziva preko TCP/IP protokola i digitalnih signala

3.1 Mjerenje odziva preko TCP/IP protokola bez kretanja robota i pokretanja paralelnih programa

Zbog toga što se koristi diskretni PID regulator koji mora biti isprogramiran i zbog toga što se regulacija odvija preko računala čiji procesor ima određenu frekvenciju rada (250 Hz) i što se komunikacija odvija preko ethernet protokola koji također ne šalje podatke kontinuirano, potrebno je odrediti vrijeme diskretizacije koje je potrebno da *slave* robot primi novu koordinatu koju mu je poslao *master* robot.

To vrijeme će biti korišteno kao vrijeme diskretizacije u jednadžbi regulatora.

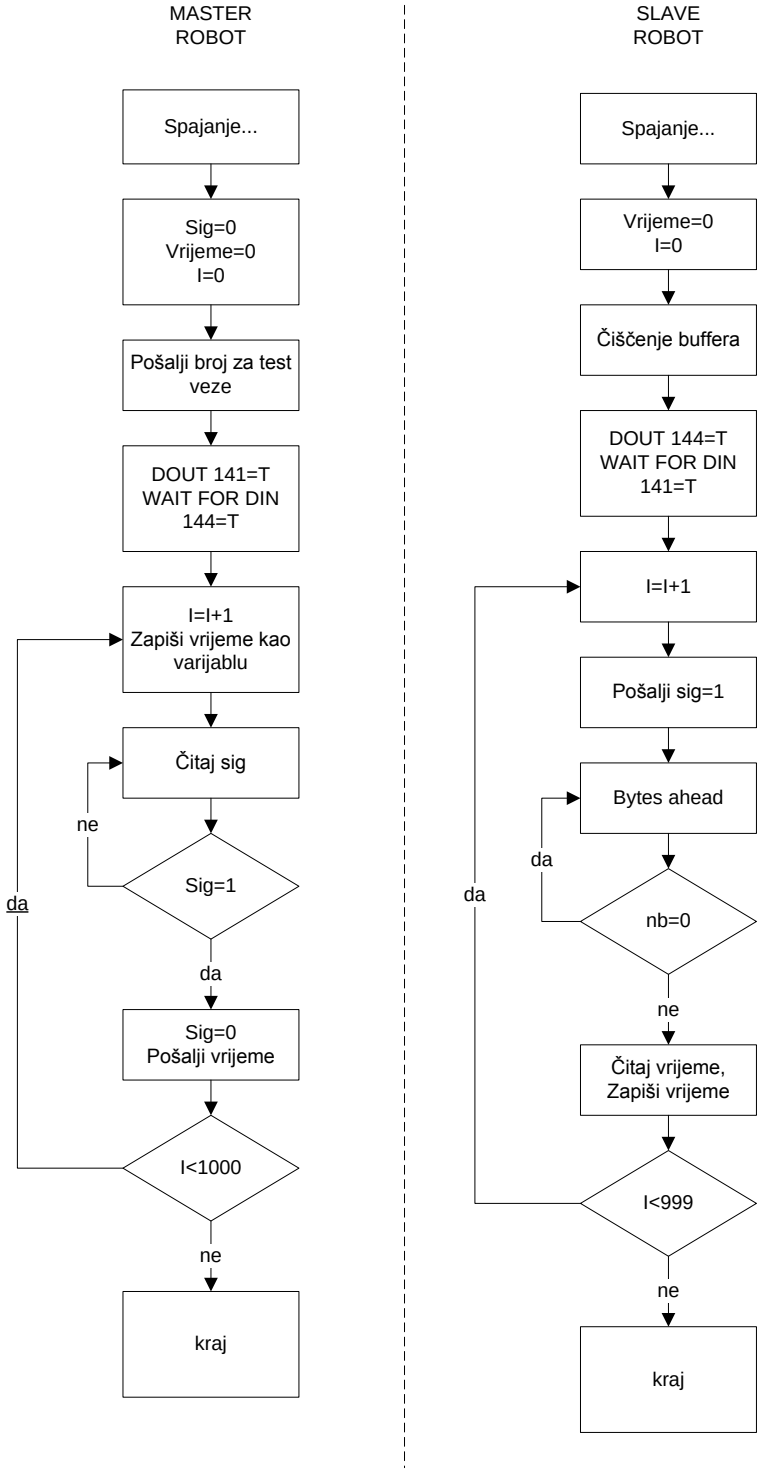
Da bi se odredilo to vrijeme korišten je unutarnji sat robota (varijabla $FAST_CLOCK koja počinje brojati od nule od trenutka kada se robot upali i nastavi brojati po 1 milisekundu za vrijeme trajanja rada robota). Jednostavno kontinuirano slanje sata *master* robota prema slave robotu i njegovo čitanje i zapisivanje tih podataka, ne bi bilo ispravno jer roboti imaju svoju međuspremnik (*eng. buffer*) od 256 bajtova, što znači da bi se kad bi se samo slao sat i ako proces primanja tog sata kod *slave* robota nije brži od programa slanja sata kod *master* robota, input buffer na *slave* robotu bi se punio sve dok *master* robot ne bi prestao slati svoje podatke. U takvom slučaju *slave* robot bi za regulaciju koristio puno starije podatke o poziciji i regulacija ne bi bila ispravna.

Da bi se to izbjeglo potrebno je *input buffer slave* robota uvijek imati praznim, tj. samo jedan podatak u njemu. To se postiglo jednostavnim polu-handshake protokolom. *Slave* i *master* roboti bi krenuli u svoju glavnu petlju u isto vrijeme. *Master* robot nakon toga ulazi u drugu petlju gdje čeka od *slave* robota signal za slanje svog sata. U isto vrijeme *slave* robot nakon što je poslao signal *master* robotu ulazi u petlju gdje kontinuirano provjerava svoj *buffer* da li ima više od 0 bitova u njemu. U trenutku kad *master* robot primi signal za slanje svog sata on resetira varijablu signala od *slave* robota i ponovno se vraća u petlju gdje čeka promjenu te varijable. *Slave* robot kada očita da se u *bufferu* pojavio neki podatak to vrijeme očita i zapiše ga i vraća se natrag u petlju. Proces na oba robota se ponavlja određen broj ciklusa zadanih brojačem. Diagram toka Slika 6. Graf kašnjenja signala Slika 7.

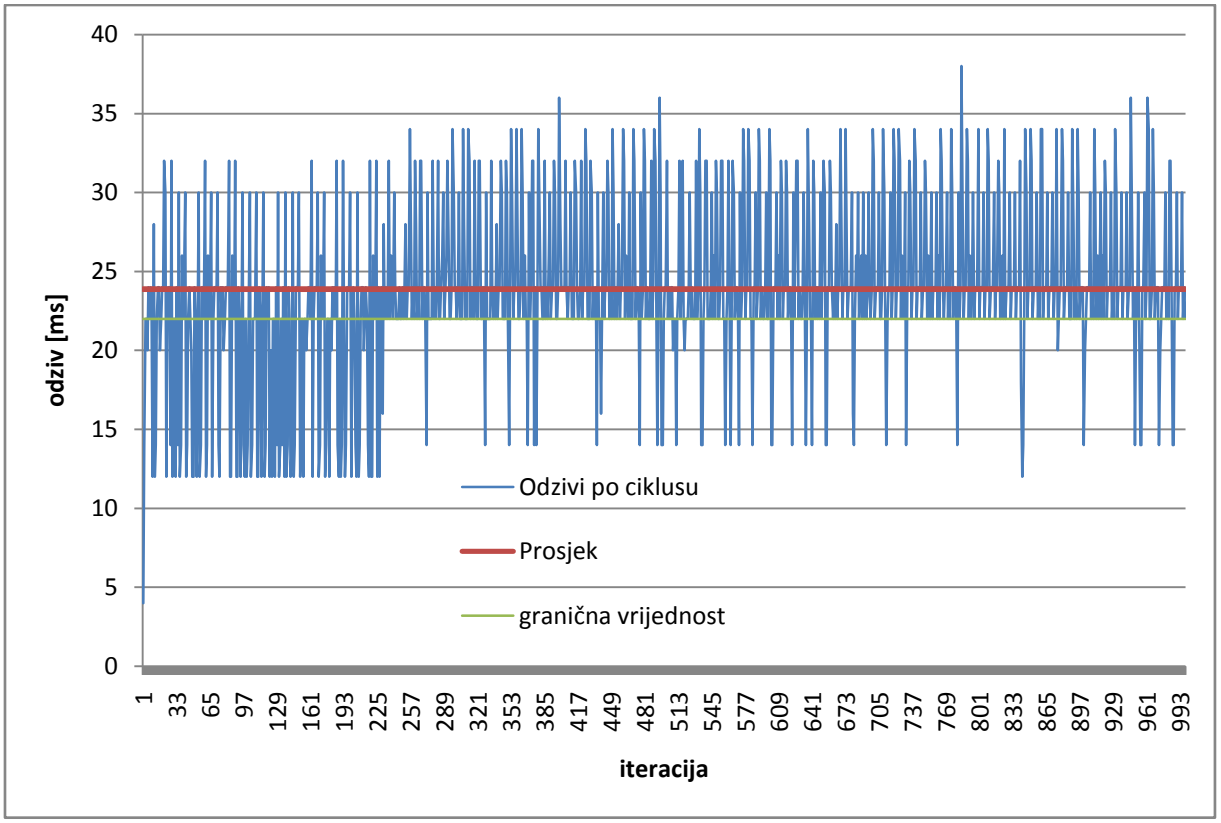
Pošto vrijeme koje je potrebno za primanje signala nije konstantno svaki ciklus već se stohastički mijenja, za vrijeme diskretizacije se uzima prosječna vrijednost svih vremena.

Potrebno je napomenuti da vrijeme diskretizacije ovako mjereno bez kretanja robota, regulacije i pozivanja drugih programa, puno manje nego kad su svi nabrojani faktori uključeni jer je potrebno određeno vrijeme robotu dodatne programe. Za točno vrijeme diskretizacije potrebno je pokrenuti testnu regulaciju u pokretu i iz nje očitati vrijeme kašnjenja.

U dodatcima A i B su dani kodovi robotskih programa u KAREL programskom jeziku za *slave* robot (brzSmjer) i za *master* robot (brzMmjer).



Slika 6: Diagram toka mjerenja odziva preko TCP/IP protokola bez kretanja robota



Slika 7: Odziv mjeren preko TCP/IP-a bez kretanje robota

Tabela 3: Rezultati mjerenja odziva preko TCP/IP-a bez kretanja robota

Min [ms]	Max [ms]	Prosjek [ms]
4	38	23.87375

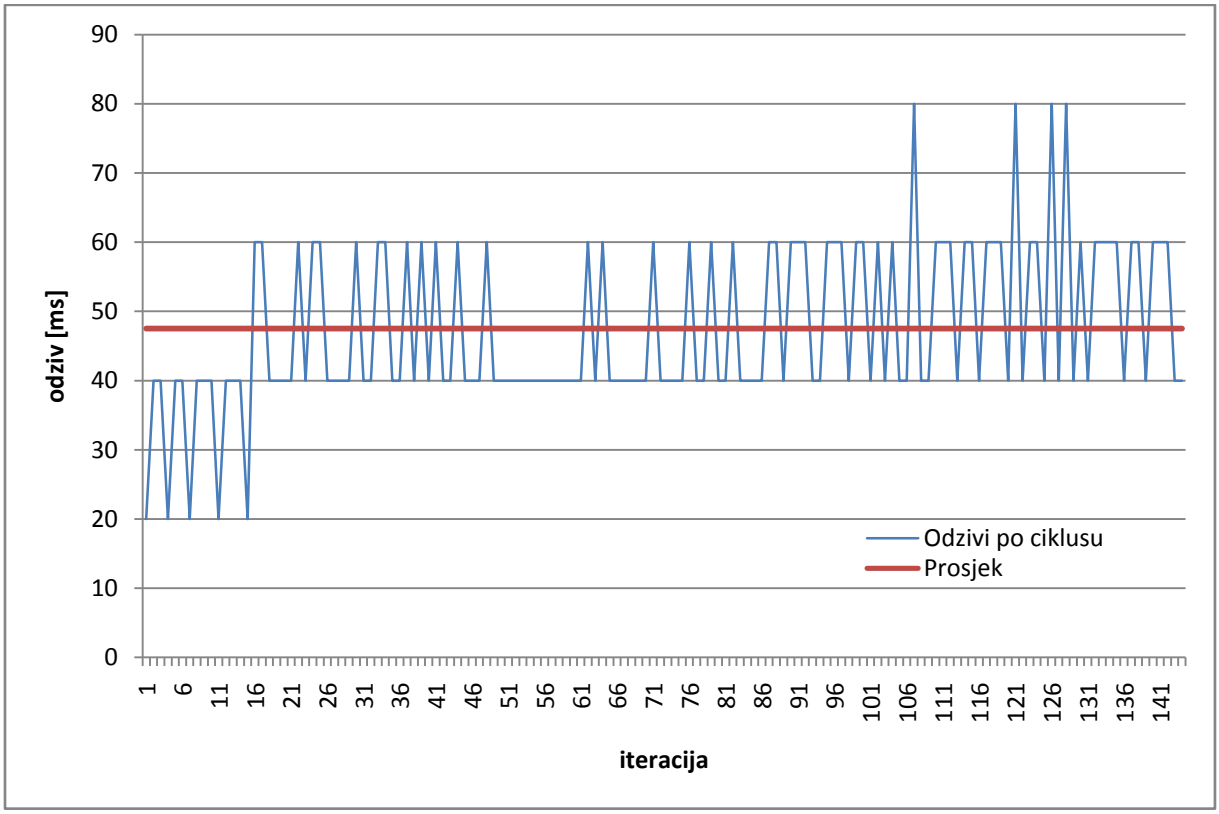
Kao što Slika 7 i Tabela 3 pokazuju odziv signala je izrazito stohastičan s minimalnom vrijednosti od 4 ms i maksimalnom od 38 ms i prosjekom od 23.9 ms. Stohastičnost signala nije konstantna tijekom cijelog procesa već na početku od prve do ~225 iteracije signal ima više vrijednosti ispod prosjeka, a nakon ~225 iteracije odziv se usporava i većina vrijednosti je iznad prosjeka. Može se također primijetiti da sustav na oko 22ms ima graničnu vrijednost oko koje oscilira nakon ~225 iteracije.

3.2 Mjerenje odziva preko TCP/IP protokola sa kretanjem robota i s pokrenutim paralelnim programima

Kao što je napomenuto u poglavlju 3.1 odziv sustava je drugačiji kada je robot u pokretu i kada uz kretnju vrši regulaciju. Slika 8 i Tabela 4 pokazuju rezultate odziva u tim uvjetima.

Kao što se vidi struktura odziva je slična kao i mjerenju bez kretanja (u početku odziv je mali a nakon određenje iteracije se poveća i ostane varirati između nekih vrijednosti) samo s puno većom srednjom vrijednosti od 47.5 i većom graničnom vrijednosti od 40ms. Minimalni odziv je 20 ms, maksimalni 80 ms.

U dodatcima E, F i G su dani kodovi robotskih programa u KAREL programskom jeziku za *slave* robot (brzS), regulator (REGULATOR) i za *master* robot (brzM) preko kojih se vršilo mjerenje.



Slika 8: Odziv mjeren preko TCP/IP-a sa kretanjem robota

Tabela 4: Rezultati mjerenja odziva preko TCP/IP-a sa kretanjem robota

Min [ms]	Max [ms]	Prosjek [ms]
20	80	47.5

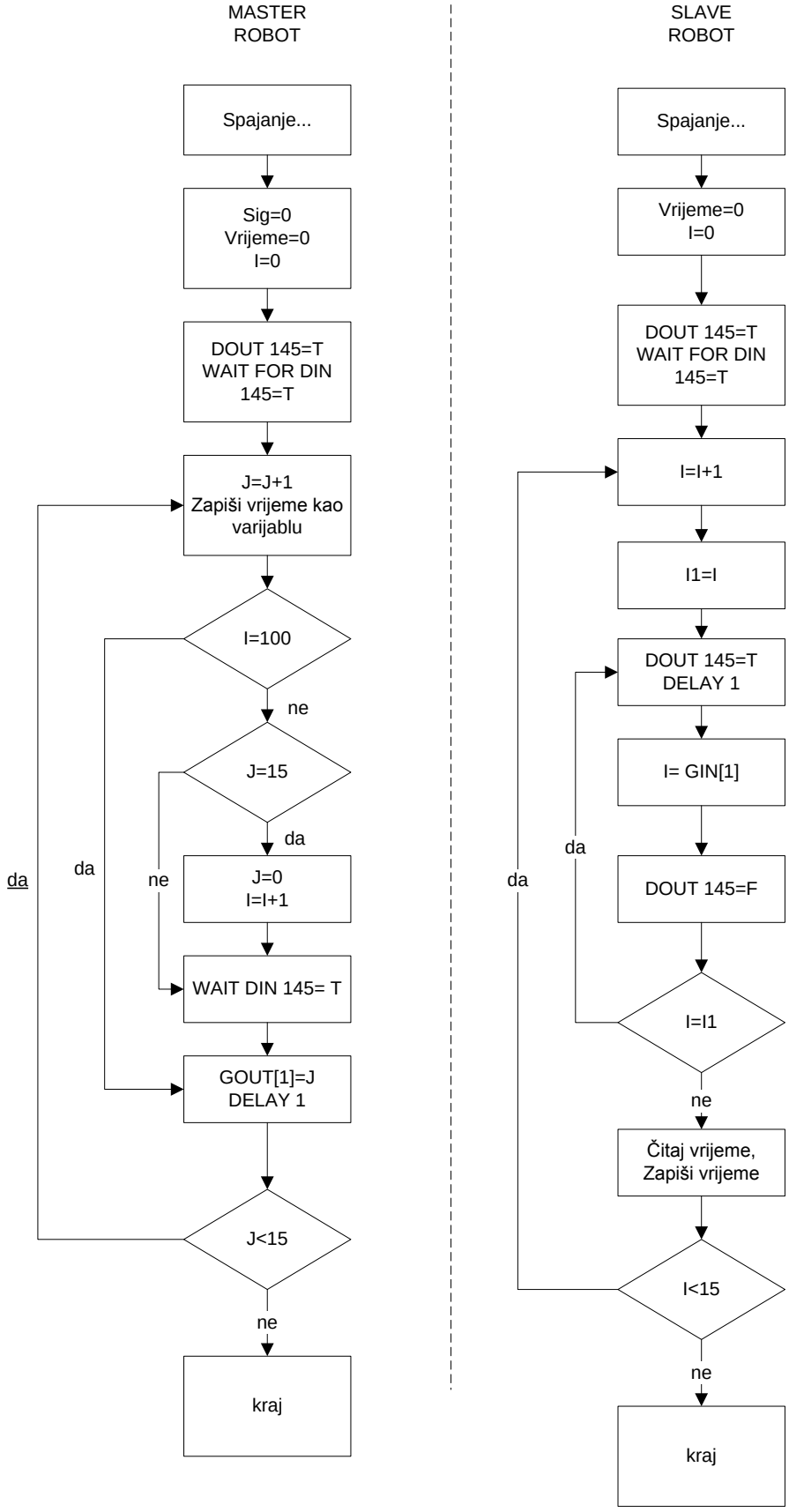
3.3 Mjerenje odziva preko digitalnih signala

Druga mogućnost slanja podataka između robota je preko digitalnih signala. Roboti između sebe imaju alocirano pet digitalnih signala (od 141 do 145). Ti signali se preko funkcije GOUT (*group out*) i GIN (*group in*) mogu koristiti za slanje brojeva koji se preko digitalnih signala šalju binarno. Zbog toga što ima pet signala najveći broj koji se može poslati je 31. Zbog toga što je potreban jedan signal od svakog robota za sinkronizaciju početka petlje i *handshakea* koji je potreban jer bi bez njega bilo potrebno pogađati čekanje koje bi master robot morao obaviti nakon slanja svakog signala jer u protivnom zbog brzine slanja i brzine procesiranja brojevi na slave robot ne bi stizali sekvencijalno. Zbog toga što drugih signala nije bilo raspoloživo zadnji digitalni signal 145 je uzet kao signal koji će se koristiti za sinkronizaciju kod pokretanja i za handshake. Tako najveći broj koji se može poslati preko digitalnih signala sad 15.

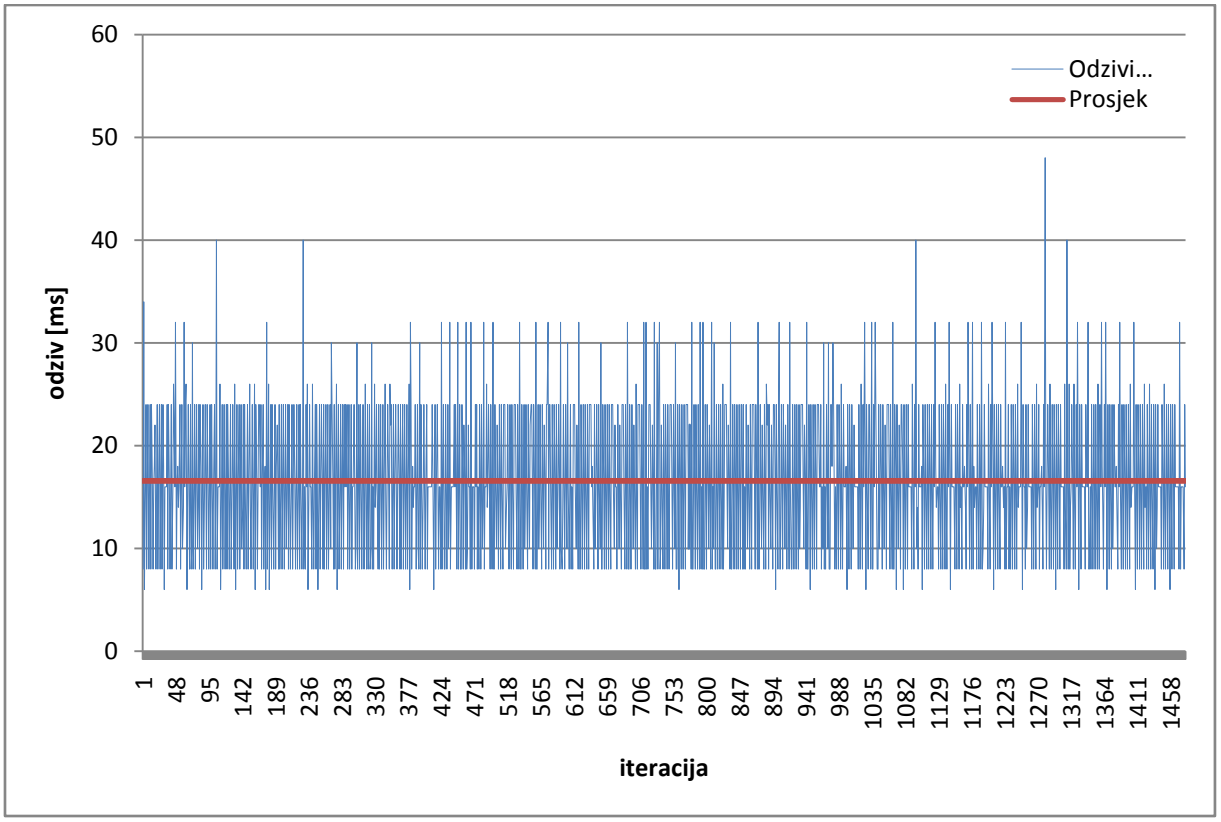
Slave robot upali digitalni signal 145 i čeka 1 milisekundu, u to vrijeme *master* robot čeka da signal 145 postane upaljen i u tom trenutku šalje novi broj na GOUT. *Slave* robot je za to vrijeme u petlji gdje provjera je li mu je GIN signal trenutno isti kao i koji je bio prethodni ciklus, ako nije zapisuje ga i vraća se na početak petlje. Zbog malog broja raspoloživih bitova napravljeno je da master robot svaki put kada dođe do najvećeg broja kojeg može poslati resetira brojač i počne brojati od nule, uz taj reset drugi brojač koji služi za određivanje broja ciklusa ponovnog slanja, tj. brojanja do 15, se poveća za jedan. Tako se dobije raspon ciklusa slanja od minimalnih koji bi bio 15 do 15*broj drugog brojača.

Diagram toka Slika 9. Graf kašnjenja signala Slika 7.

U dodatcima C i D su dani kodovi robotskih programa u KAREL programskom jeziku za *slave* robot (brzSDig) i za *master* robot (brzMDig).



Slika 9: Diagram toka mjerenja odziva preko TCP/IP protokola bez kretanja robota



Slika 10: Odziv mjeren preko digitalnih signala bez kretanje robota

Tabela 5:Rezultati mjerenja odziva preko digitalnih signala bez kretanja robota

Min [ms]	Max [ms]	Prosjek [ms]
6	48	16.55713

Iz Slika 10 i Tabela 5 se vidi da je signal stohastičan kroz sve iteracije za razliku od signala preko TCP/IP protokola. Iz Tabela 6 se vidi digitalni signali ima najmanju prosječnu vrijednost ali pri istim uvjetima testiranja ima veće oscilacije u signalu (34ms za TCP/IP, 42ms za digitalne signale). TCP/IP s kretanjem robota ima najveću prosječnu vrijednost uz najveće oscilacije signala.

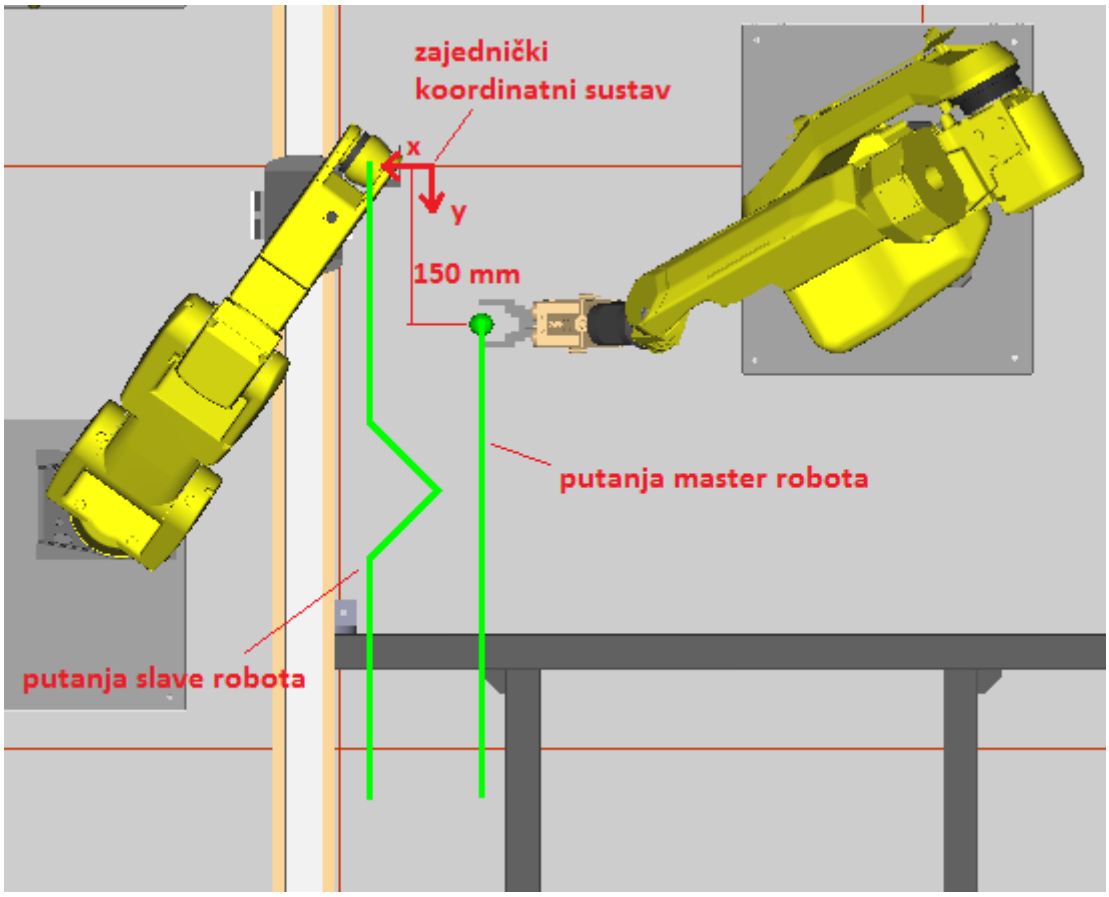
Tabela 6: Usporedba rezultata mjerenja odziva signala

	Min [ms]	Max [ms]	Prosjek [ms]
TCP/IP bez kretanja	4	38	23.87375
TCP/IP sa kretanjem	20	80	47.5
Digitalni signali	6	48	16.55713

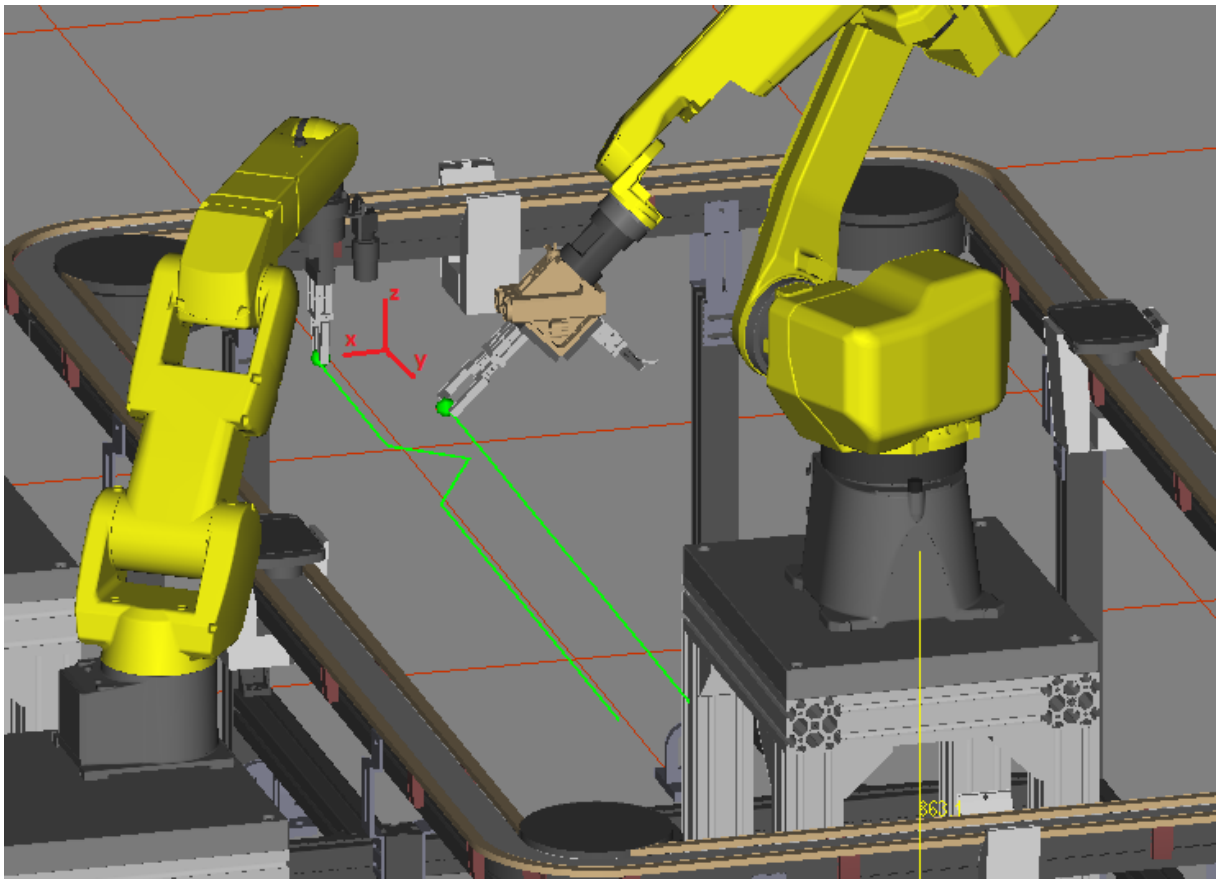
4 Praćenje robota [7]

4.1 Radni prostor

Svi testovi kretanja, mjerenja vremena odziva i regulacije su rađeni na LR Mate 200iC/5Lrobotu koji je služio kao *slave* robot (robot kojem se regulira brzina, koji sadrži regulator i koji izvršava kretanje spajanja) i FANUC M10iA robotu koji služi kao *master* robot (robot koji se kreće po jednoj osi konstantnom brzinom i šalje podatke o svojoj trenutnoj poziciji). Roboti se kreću putanjom od 1000 mm. *Master* robot kreće s odstupanjem od 150 mm po y osi od *slave* robota. Spajanje se vrši u x osi tako da se *slave* robot približava *master* robotu u negativnom smjeru x osi.



Slika 11: Tlocrt konfiguracije robota



Slika 12: Konfiguracija robota u prostoru

Za izvršenje spajanja u pokretu za što je potrebna određenja preciznost potreban je regulator koji će to osigurati. Za regulaciju su korištena tri upravljačka programa:prvi upravljački program se nalazi na *master* robotu i služi za određivanje točaka kretanja robota i sadrži protokol za komunikaciju i slanje podataka sa slave robotom (opis programa u 4.4, cijeli kod u dodatku E),. Drugi program služi samo za određivanje točaka kretanja slave robota, od početne do sljedeće i određivanje kretanja spajanja (opis programa u 4.4, cijeli kod u dodatku F), taj program poziva regulator. Treći upravljački program služi za regulaciju. On sadrži regulator i protokol za komunikaciju i slanje podataka s *master* robotom (opis programa u 4.5, cijeli kod u dodatku G), taj program samo zadaje brzinu *slave* robotu ne utječe na točke kretanja robota.

4.2 Kalibracija koordinatnih sustava robota i kalibracija vrha alata

Zbog lakšeg programiranja i određivanja greške regulacije potrebno je bilo staviti robote u isti koordinatni sustav. To se napravilo tako da je svakom robotu napravljen *USER FRAME* koji imali isto ishodište i orijentaciju koordinatnog sustava. Prije toga je bilo potrebno

odrediti i kalibrirati vrh alata svakog od robota zbog toga roboti imaju zamjenjive hvataljke od kojih svaka ima točku alata u drugoj poziciji.

4.2.1 Kalibracija vrha alata

Kalibracija vrha alata se izvodi tako da prvo se na robot se stavi odabrana hvataljka i s njom se prihvati alat za kalibraciju. Nakon toga se odabere proizvoljna točka u prostoru. Da bi imali fizičku reprezentaciju te točke koristi se alat sa šiljkom gdje vrh šiljka predstavlja tu točku u prostoru. Taj alat se pozicionira fizički da mu se osigura pozicija. Nakon toga s alatom za kalibraciju na hvataljki robota koji predstavlja vrh alata ručno dovodi u dodir s vrhom točke u prostoru. Postupak se ponavlja tri puta. Poželjno je da se točki u prostoru s alatom za kalibraciju približimo iz što različitijih smjerova i orijentacija robota da bi se osigurala preciznost kalibracije. Postupak kalibracije Slika 13.

Slika 13: Kalibracija vrha alata robota

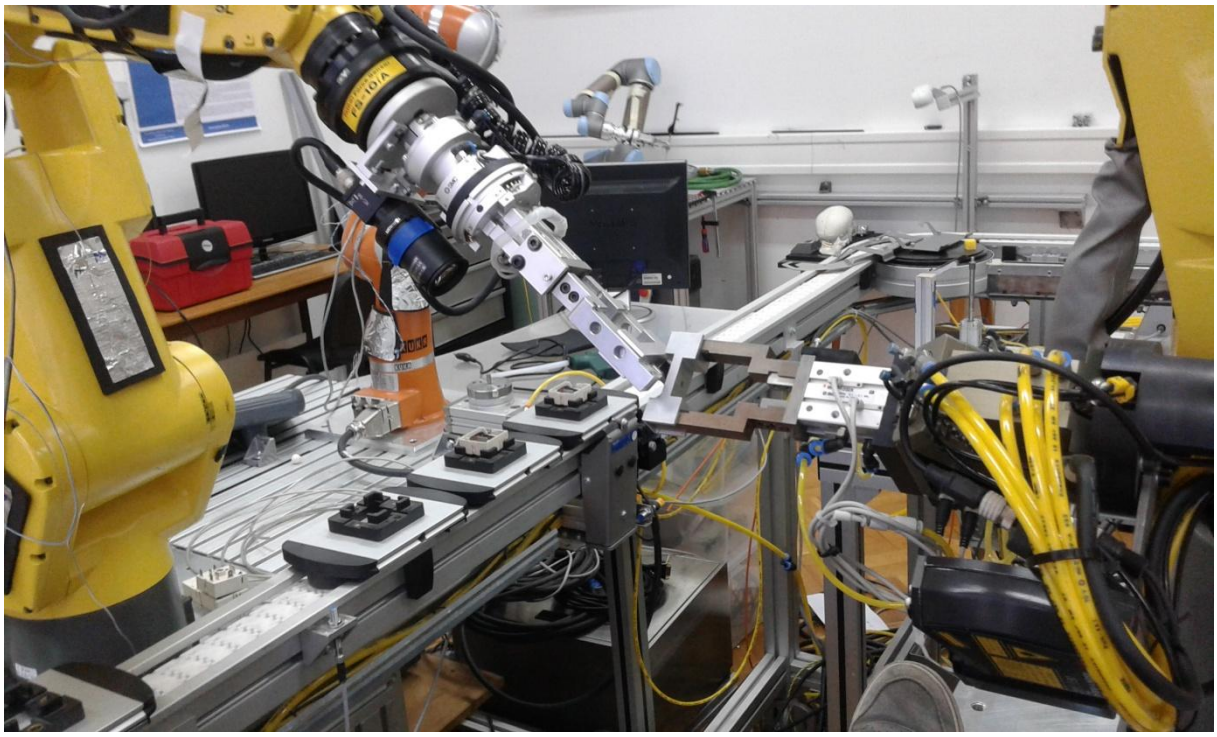
4.2.2 Kalibracija zajedničkih koordinatnih sustava

Nakon kalibracije vrha alata da bi se osiguralo da oba robota imaju isti koordinatni sustav, potrebno je napraviti poseban korisnički koordinatni (*USER FRAME*) sustav na svakom robotu i osigurati da ti koordinatni sustavi dijele isto ishodište i orijentaciju.

Određivanje korisničkog koordinatnog sustava je vrlo slično kalibraciji alata. Potrebna je točka u prostoru koja predstavlja ishodište i potrebno je robota pomaknuti u smjerovima x i

y osi da bi orijentirali sustav. Također *user frame* se može određivati ručnim unošenjem koordinata i kutova zakreta robota.

Da bi se olakšala kalibracija nije se koristila zasebna točka u prostoru kao kod kalibracije vrha alata već je za to korišten vrh alata *master* robota. Prvo se Master robotu odredio njegov user frame. Točka u prostru koja je ishodište odabrana je da bi se tijekom kretanja robot mogao slobodno kretati duž y osi bez kolizije s okolinom. Ta točka je ručno napisana u odnosu na njegov WORLD frame. Nakon toga u se odredili smjerovi y i x osi pomicanjem robota u željenim smjerovima i time je definiran *user frame master* robota. Nakon toga *master* robot se vraća u ishodište i stavlja u poziciju da bi se vrhovi alata *master* i *slave* robota mogli dotaknuti, tj. vrh alata master robota sada služi kao fizička točka u prostoru koja predstavlja ishodište za koordinatni sustav *slave* robota. *Slave* robot se dovodi sa svojim vrhom alata do vrha alata master robota i označi tu točku kao svoje ishodište, nakon toga se *master* robot pomiče u po x i y osi gdje također *slave* robot mora doći u doticaj s vrhom alata master robota. Zbog nesavršenosti postavljana robota u laboratoriju osi *master* i *slave* robota nisu paralelne i postoje mali kutovi između osi.



Slika 14: Smještanje robota u isti koordinatni sustav

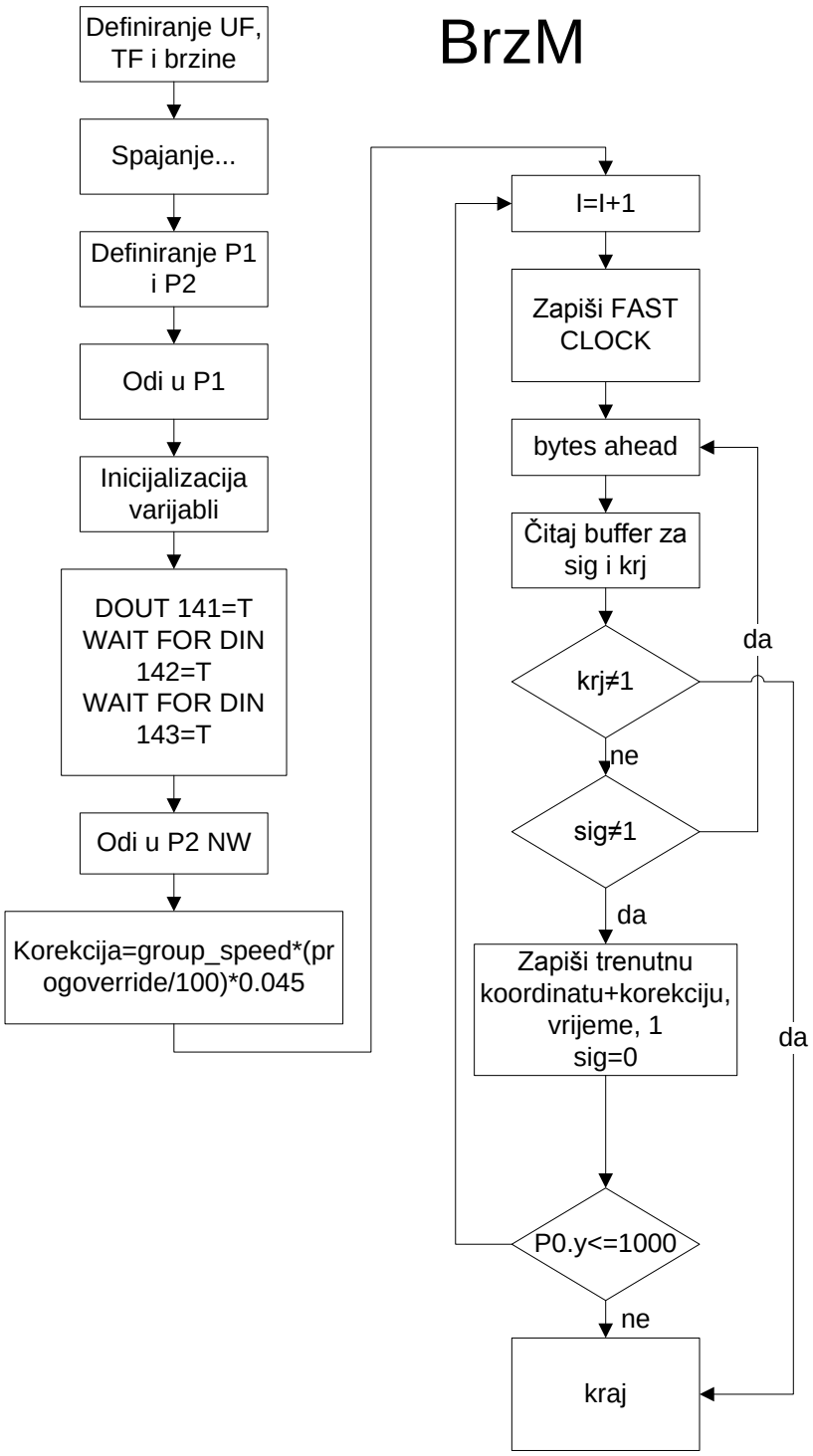
4.3 Program za kretanje master robota

Program *master* robota sadrži protokol za komunikaciju i za kretanje master robota. Prvo *master* robot definira *user* i *tool framove* i uspostavlja vezu sa *slave* robotom. Poslije toga definira početnu i krajnju točku kretanja i pomiče robota u početnu točku. Nakon dobivanja signala za sinkronizaciju od *slave* robota počinje petlju koja traje dok robot ne dođe do krajnje točke. U petlji robot prvo provjerava svoj *input buffer* sve dok ne dobije u njemu varijable od *slave* robota koje su signal za slanje trenutne pozicije i *krj* varijabla koja služi za zaustavljanje petlje u slučaju da *slave* robot završi prije sa svojim programom. Kad dobije signal za slanje *master* robot *slave* robotu šalje svoju trenutnu poziciju povećanu za faktor korekcije koji se dobije jednadžbom

$$\$GROUP[1].\$SPEED * \left(\frac{\$MCR\_GRP[1].\$PRGOVERRIDE}{100}\right) * T$$

gdje $GROUP[1].$SPEED je maksimalna brzina robota u mm/s, MCR_GRP[1].$PRGOVERRIDE je postotak od maksimalne brzine, a T vrijeme diskretizacije. Tako se dobije pozicija u kojoj će *master* robot biti nakon što je *slave* robot dobio taj podatak zbog kašnjenja komunikacije. Također *master* robot šalje i svoj sat ($FAST_CLOCK) preko kojeg se može nakon svakog pokretanja programa vidjeti vrijeme kašnjenja. Diagram toka Slika 15.

U dodatku E su dani kodovi robotskih programa u KAREL programskom jeziku za *master* robot (brzM).

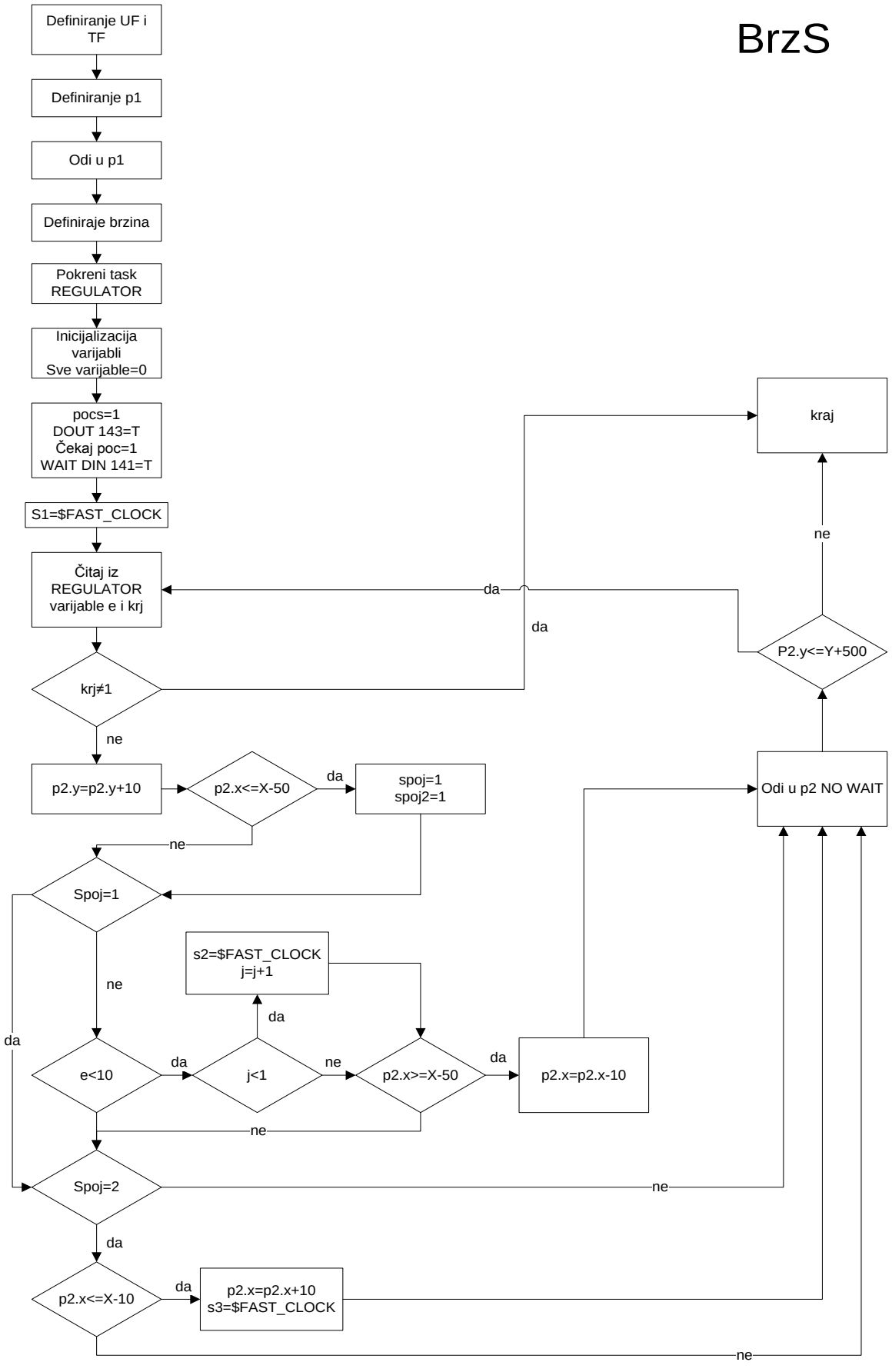


Slika 15: Diagram toka programa *master* robota

4.4 Program za kretanje *slave* robota

Program na *slave* robotu koji definira kretanje počinje definiranjem *user* i *tool frameova* robota. Nakon toga definira i pomiče robot u početnu točku u prostoru. Nakon toga kad primi signale za sinkronizaciju od ostalih programa ulazi u petlju. Dok ne prođe svoj zadani put od 1000mm program svaki ciklus iz programa za regulaciju uzima varijablu greške i *krj* varijablu (u slučaju da *master* robot završi svoju petlju prije *slave* robota *master* robot šalje signal preko te varijable da se petlja slave robota završi). Nakon toga definira sljedeću točku u y osi tako da se trenutnoj y koordinati zbroji 10 mm. Sljedeći uvjet je ako je x koordinata manja ili jednaka od početne x koordinate minus 50mm definiraju se varijable (spoj1 i spoj2) koje određuju hoće li robot po x osi radi spajanje ili odvajanje. Ako su obje varijable jednake nula robot će vršiti spajanje, ako su jednake 1, robot će vršiti odvajanje. U slučaju da su varijable jednake nuli i da je greška za vrijeme praćenja padne ispod 10mm *slave* robot se počne približavati *maste* robotu po x osi mijenjajući x koordinatu tako da trenutnoj x koordinati oduzima 10 mm sve dok se ne zadovolji da od početne x koordinate robot nije više ili jednako 50 mm udaljen. Nakon toga se varijable spoj1 i spoj2 mijenjaju i počinje odvajanje koje se odvija tako da se x koordinati dodaje 10 mm sve dok se ne vrati na početnu vrijednost. Nakon toga daje se naredba da robot krene u točku p2 bez čekanja (NO WAIT naredba kaže robotu u koju točku mora ić i nastavlja dalje s upravljačkim programom bez čekanja da robot prvo dođe u tu točku i onda tek nastavi sa programom). Diagram toka Slika 16.

U dodatku F su dani kodovi robotskih programa u KAREL programskom jeziku za *slave* robot (brzS).



Slika 16:Diagram toka programa za kretanje *slave* robota

4.5 Program za regulaciju

Program za regulaciju uz regulaciju služi i za komunikaciju i razmjenu podataka s *master* robotom. Nakon obavljenog protokola spajanja program čisti *input buffer* u slučaju da osigura da je sljedeći podatak u njemu s *master* robota. Nakon toga otvara .txt datoteku u kojoj zapisuje sve podatke pod funkcijom *rewrite* (*'rw'*) da izbriše sve podatke u njoj i zatvori je i nakon toga je opet otvara pod funkcijom *append* (*'ap'*). Nakon toga inicijaliziraju se varijable među njima i pojačanja regulatora. Poslije toga se čekaju signali za sinkronizaciju i počinje petlja koja traje dok robot nije prešao 1000mm u y osi. Program šalje signal preko mreže *master* robotu i nakon toga provjera svoj *input buffer* je li ima više od 0 bajtova u sebi. Ako ima to pročitaj koordinatu *master* robota i pročitaj varijablu *krj* (služi za prekid programa ako je *master* robot završio prije svoj program). Nakon toga se izračuna trenutna greška oduzimajući koordinatu *master* robota s trenutnom koordinatom *slave* robota. greška se zbroji sa zbrojem grešaka iz prethodnih ciklusa i nakon toga preko jednadžbi regulatora i dobiva se brzina *slave* robota u tom trenutku. Zbog fizičkih i programskih ograničenja robota ta brzina se mora ograničiti. Brzina ne smije preći preko 100 ili biti manja od 0.001 jer regulator regulira brzinu robota kao postotak maksimalne brzine koja je definirana u programu koji definira kretanje *slave* robota. Prava trenutna brzina robota u [*mm/s*] se dobije preko jednadžbe

$$\$GROUP[1].\$SPEED * \left(\frac{\$MCR\_GRP[1].\$PRGOVERRIDE}{100}\right)$$

gdje $GROUP[1].$SPEED je maksimalna brzina robota u mm/s, MCR_GRP[1].$PRGOVERRIDE je postotak od maksimalne brzine. Također se regulira i maksimalno ubrzanje pozitivno ili negativno koje je definirano na početku kao varijabla u mm/s^2.

Definira se preko jednadžbi:

ako je

$$\frac{v - v_1}{T} > a_{max}$$

onda je brzina jednaka:

$$v = a_{max} * T + v_1$$

ili ako je

$$\frac{v_1 - v}{T} < a_{max}$$

onda je brzina jednaka:

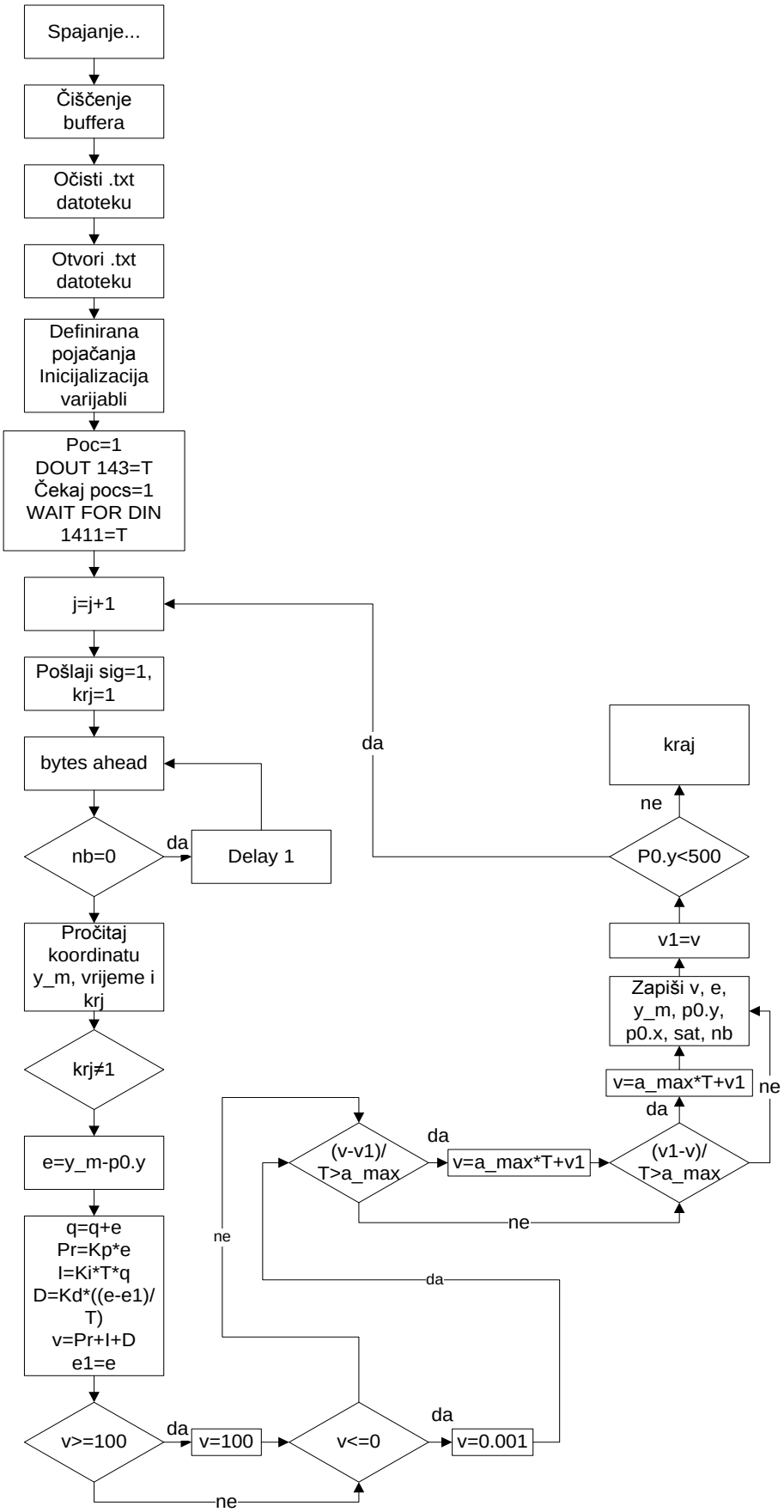
$$v = -a_{max} * T + v_1$$

Gdje je *T* vrijeme diskretizacije a_{max} maksimalno ubrzanje u obliku pogodnom za regulaciju jer kao što je navedeno robot ne gleda brzinu u mm/s već kao postotak maksimalne. Maksimalno ubrzanje u obliku za regulator dobija se preko jednadžbe:

$$a_{max}\,(\%) = \frac{a_{max}\left(\frac{mm}{s^2}\right)}{\frac{\$GROUP[1].\$SPEED}{\$MCR\_GRP[1].\$PRGOVERRIDE}}$$

Jednadžbe za ubrzanje su dobivene preko jednostavne diskretizacije diferencijalne jednadžbe $a = \frac{dv}{dt}$. Nakon toga se MCR_GRP[1].$PRGOVERRIDE sistemska varijabla definira kao izlazna brzina iz regulatora. Diagram toka Slika 17.

U dodatku F su dani kodovi robotskih programa u KAREL programskom jeziku za regulator (REGULATOR).



Slika 17: Diagram toka programa za regulaciju brzine

5 Regulacija

Testirali su se 3 različite vrste regulatora na nekoliko različitih brzina. Regulatori su testirani prvo na brzini *master* robota od 50 mm/s i onda se regulator koji se pokazao najboljim testirao na većim brzima (100, 150, 200, 300 mm/s). Sva mjerenja su rađena na putanji od 1000 mm uz maksimalnu brzinu *slave* robota od 400 mm/s. Pojačanja regulatora su tražena pokušajem i gledanjem prethodnog odziva sustava.

Ovisnost pojedinih pojačanja na odziv sustava su dana u sljdećoj tablici [8]:

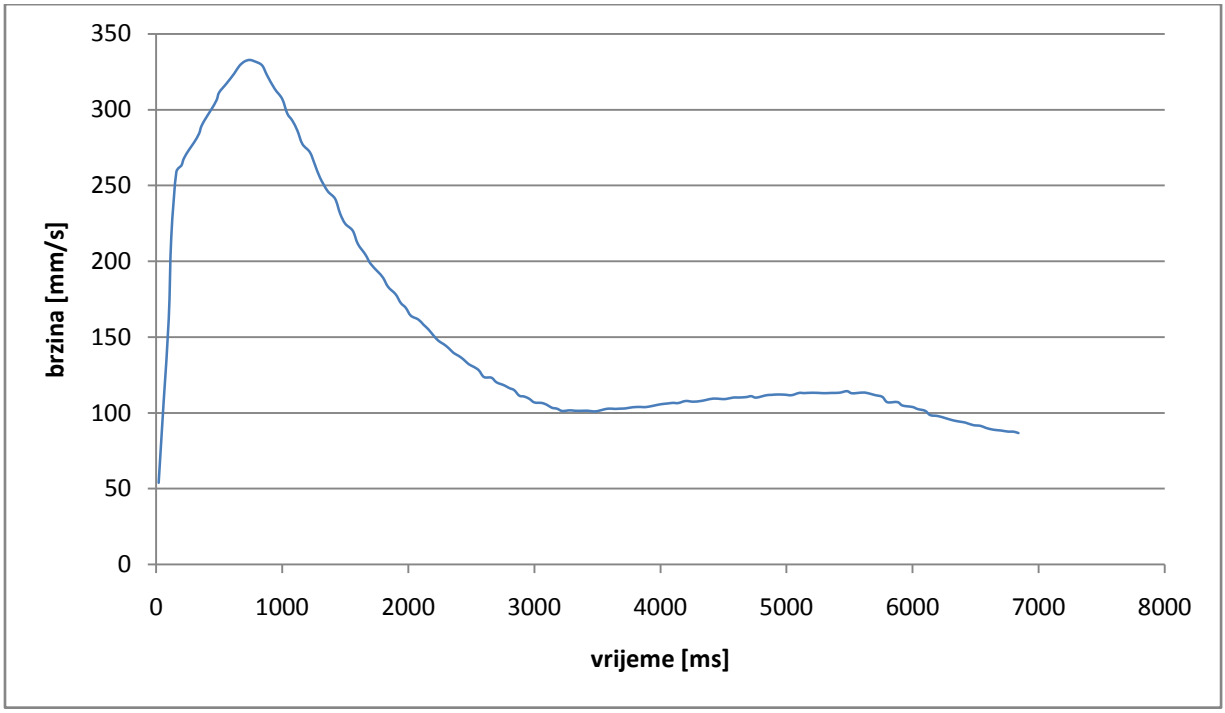
Tabela 7: Utjecaj nezavisnog pove canja parametara

Parametar	Brzina odziva	Prebačaj	Smirivanje	Reg. greška	Stabilnost
Kp	smanjuje	povećava	mala promjena	smanjuje	degradira
Ki	smanjuje	povećava	povećava	eliminira	degradira
Kd	mala promjena	smanjuje	smanjuje	nema utjecaj	povećava za mali **Kd**

5.1 PI regulator

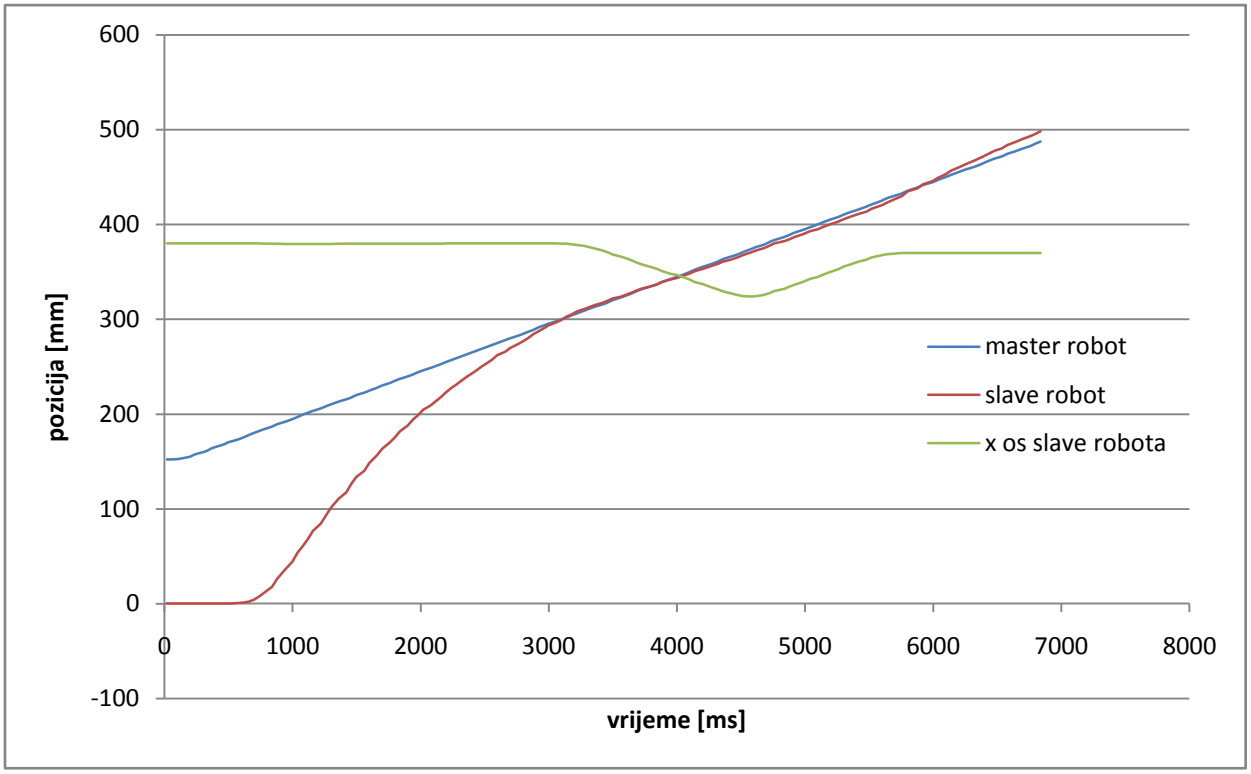
PI regulator nastaje iz PID regulatora tako kada mu pojačanje Kd je jednako nula.

Slijede grafovi brzine, greške i putanja robota (Slika 18, Slika 19, Slika 20).

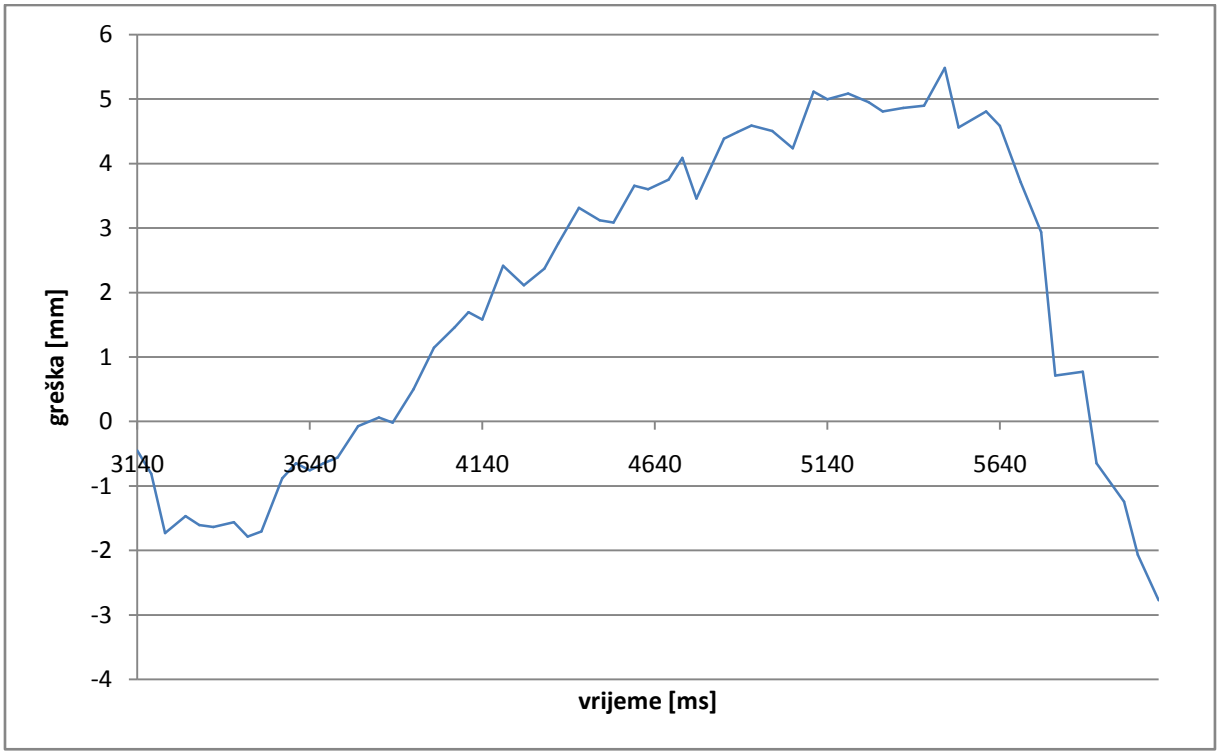


Slika 18: Brzina *slave* robota kod PI regulatora na 50 mm/s brzine *master* robota

Slika 19: Greška *slave* robota kod PI regulatora na 50 mm/s brzine *master* robota



Slika 20: Putanje *master* i *slave* robota kod PI regulatora po y osi i putanja *slave* robota po x osi za vrijeme spajanja pri brzini *master* robota od 50 mm/s



Slika 21: Greška PI regulatora u trenutku spajanja

Kao što se vidi za vrijeme spajanja (Slika 21) kada je preciznost najpotrebnija regulator ne može održati grešku konstantnom već je u laganom porastu. Nakon odvajanja regulator zbog promjene dinamike gibanja ne nastavi već se greška poveća i nastavi povećavati.

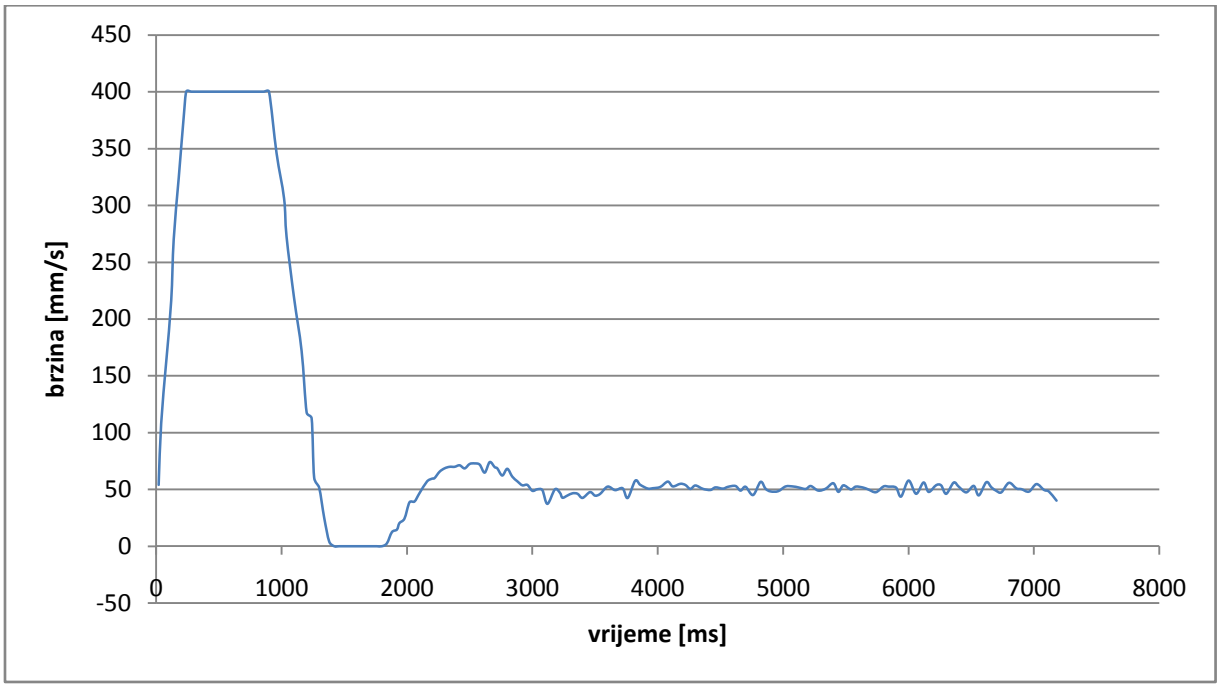
Tabela 8: Mjerenja greške u trenutku spajanja PI regulatora

Min [mm]	Max [mm]	Prosjek [mm]
-2.77765	5.47995	1.818189483

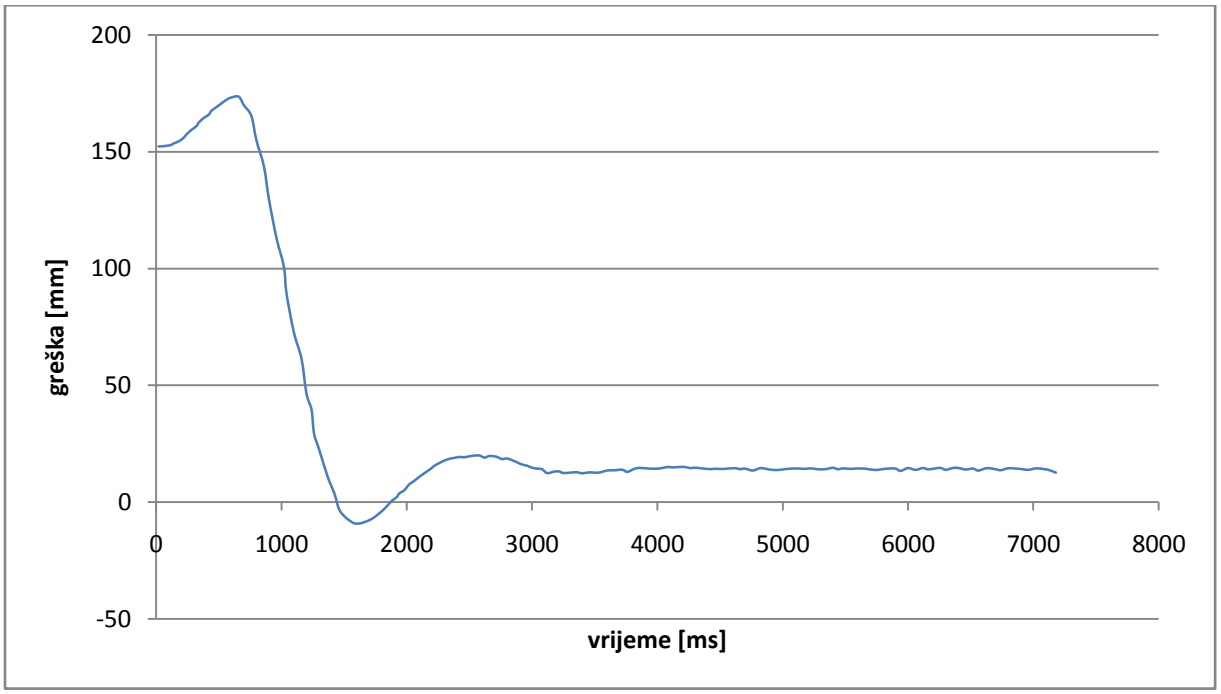
5.2 PD regulator

PD regulator nastaje iz PID regulatora tako kada mu pojačanje Ki je jednako nula.

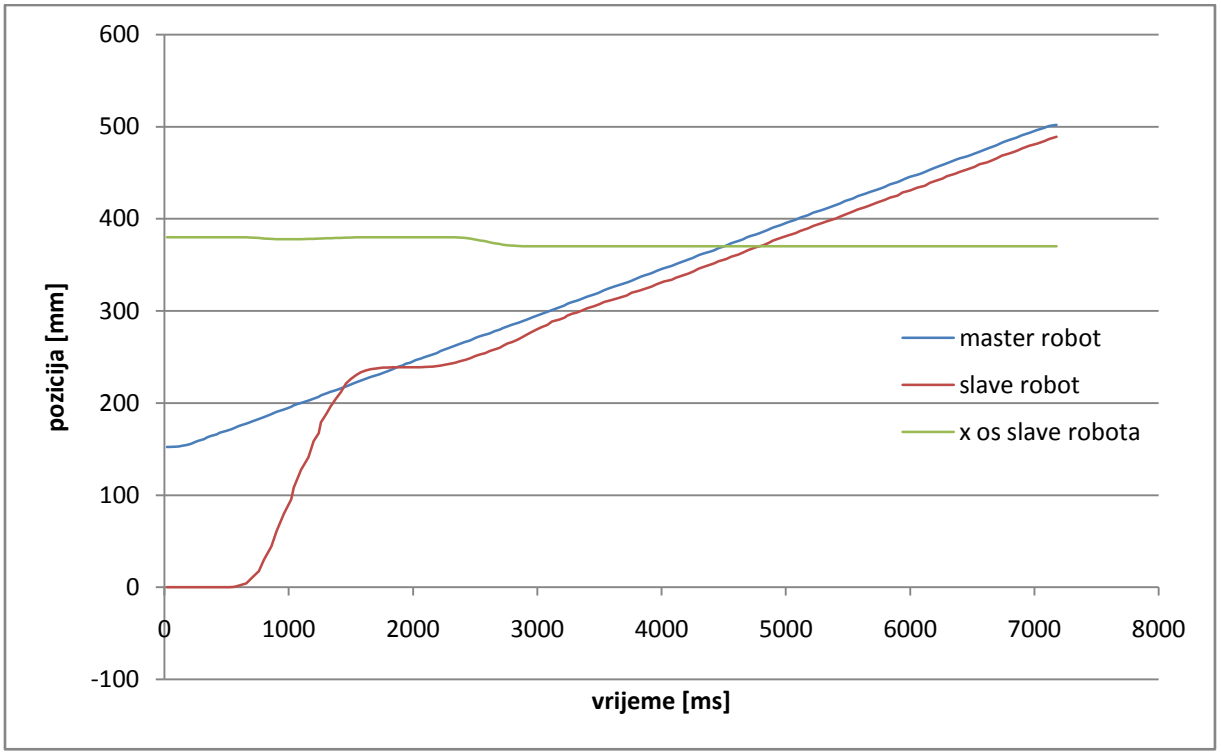
Slijede grafovi brzine, greške i putanja robota (Slika 22, Slika 23, Slika 25).



Slika 22: Brzina slave robota kod PD regulatora na 50 mm/s brzine master robota



Slika 23: Greška slave robota kod PD regulatora na 50 mm/s brzine master robota



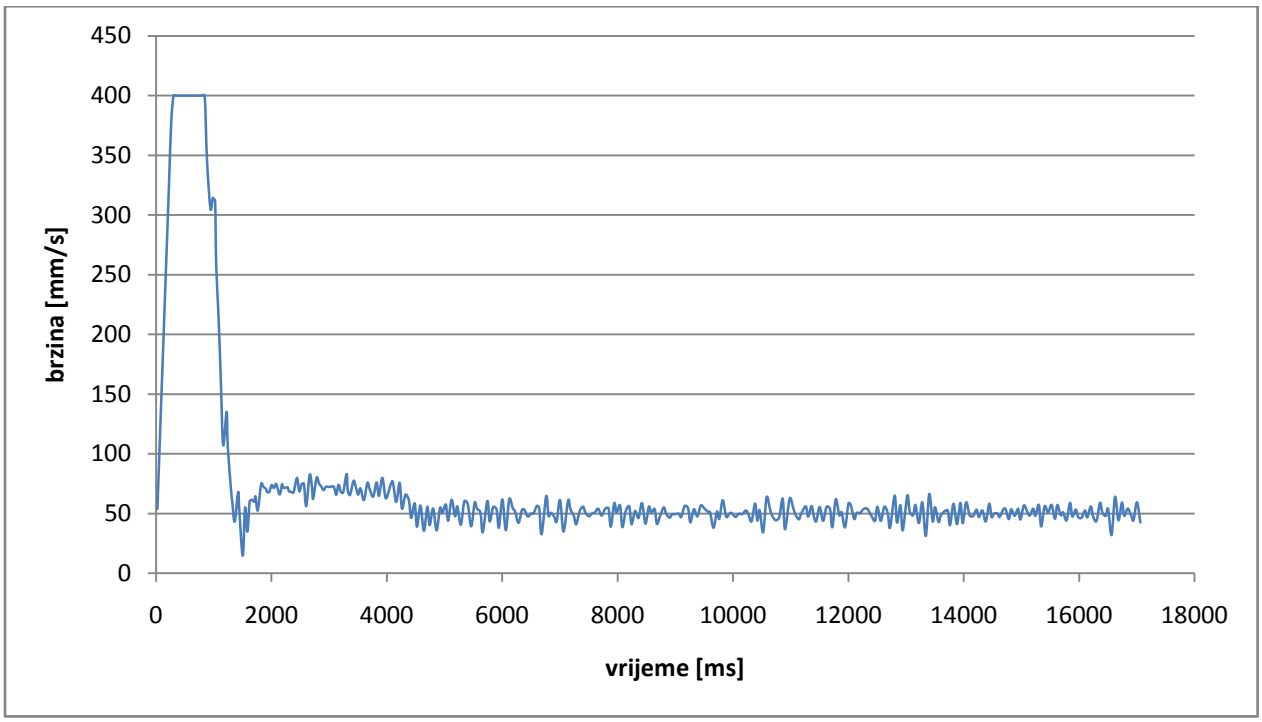
Slika 24: Putanje *master* i *slave* robota kod PD regulatora po y osi i putanja *slave* robota po x osi za vrijeme spajanja pri brzini *master* robota od 50 mm/s

Kao što se vidi (Slika 22, Slika 23, Slika 24) *slave* robot uz prebačaj odlično prati *master* robot , ali zbog nedostatka integralnog člana ima regulacijsko odstupanje koje je preveliko da bi robot ikad krenuo u spajanje. *Slave* robot kreće u spajanje kada mu je greška kontinuirano manja od 10mm. U trenutku prebačaja sustav ima grešku manju do 10 ali ne dovoljno dugo da bi *slave* robot izvršio spajanje. Regulacijsko odstupanje od trenutka smirivanja je 13.955mm

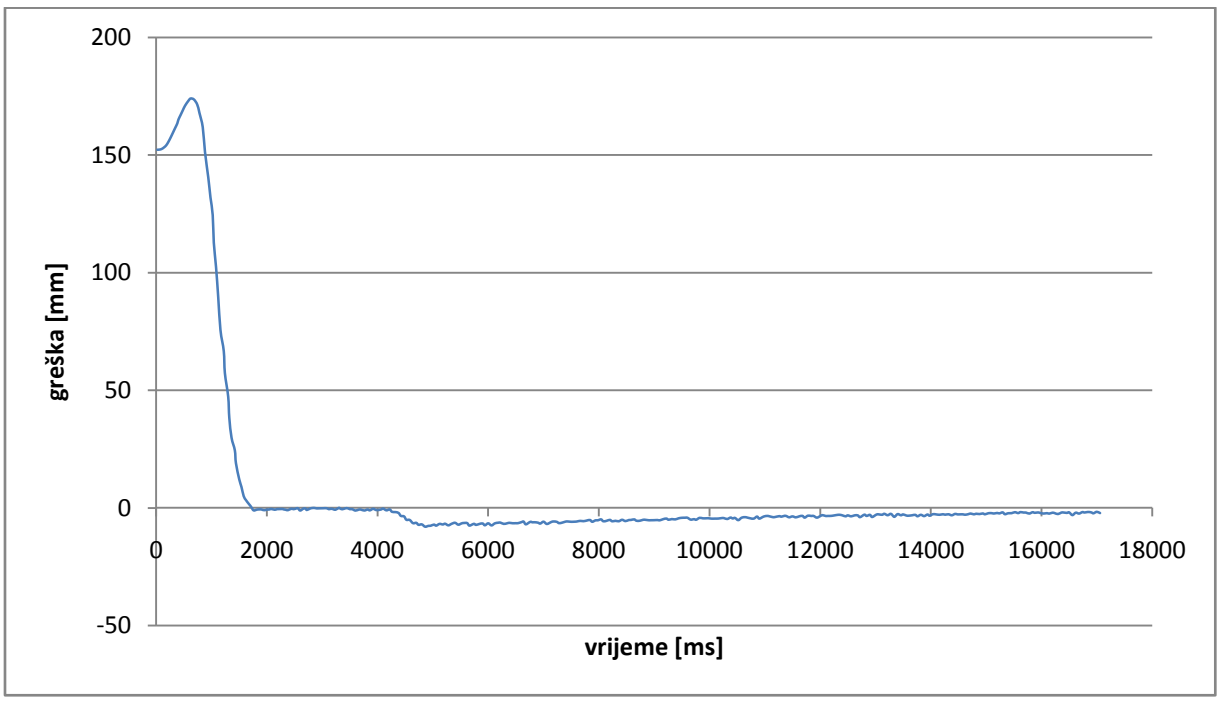
5.3 PID regulator

PID regulator kako mu ime govori sadrži sva tri pojačanja tj. ima proporcionalno, integralno i derivacijsko djelovanje.

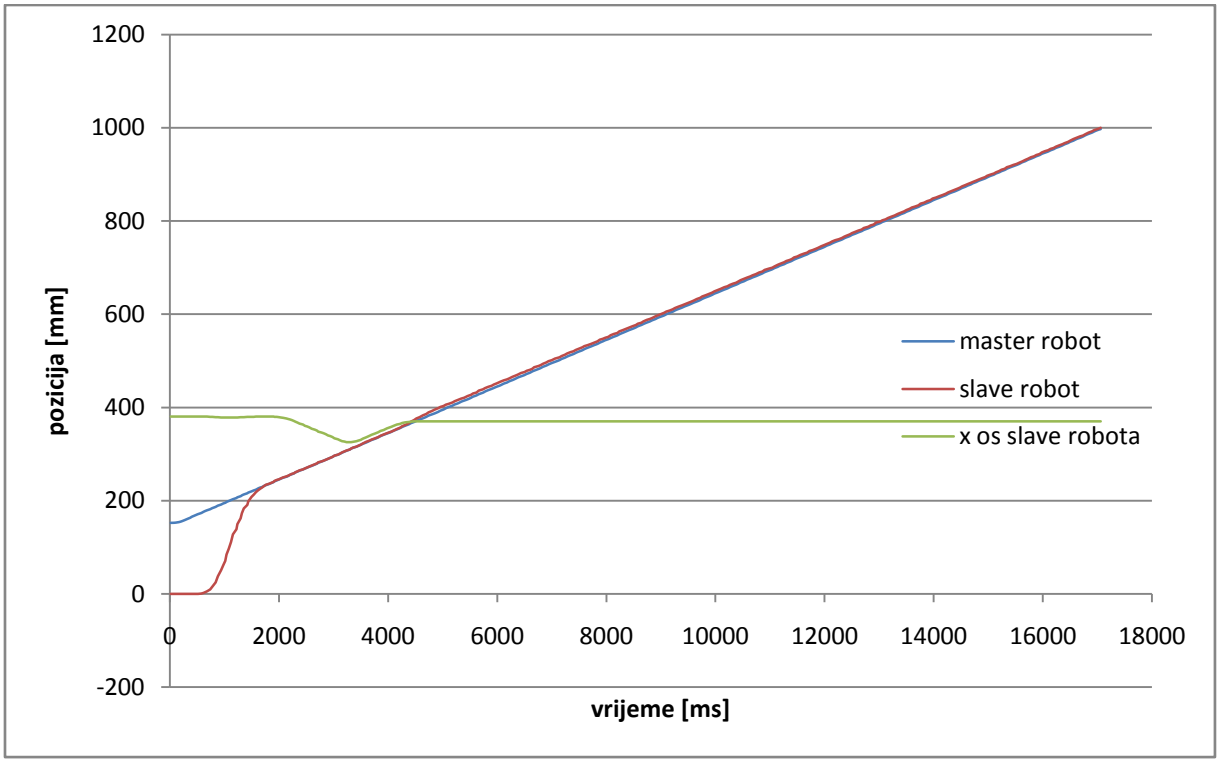
Slijede grafovi brzine, greške i putanja robota (Slika 25, Slika 26, Slika 27).



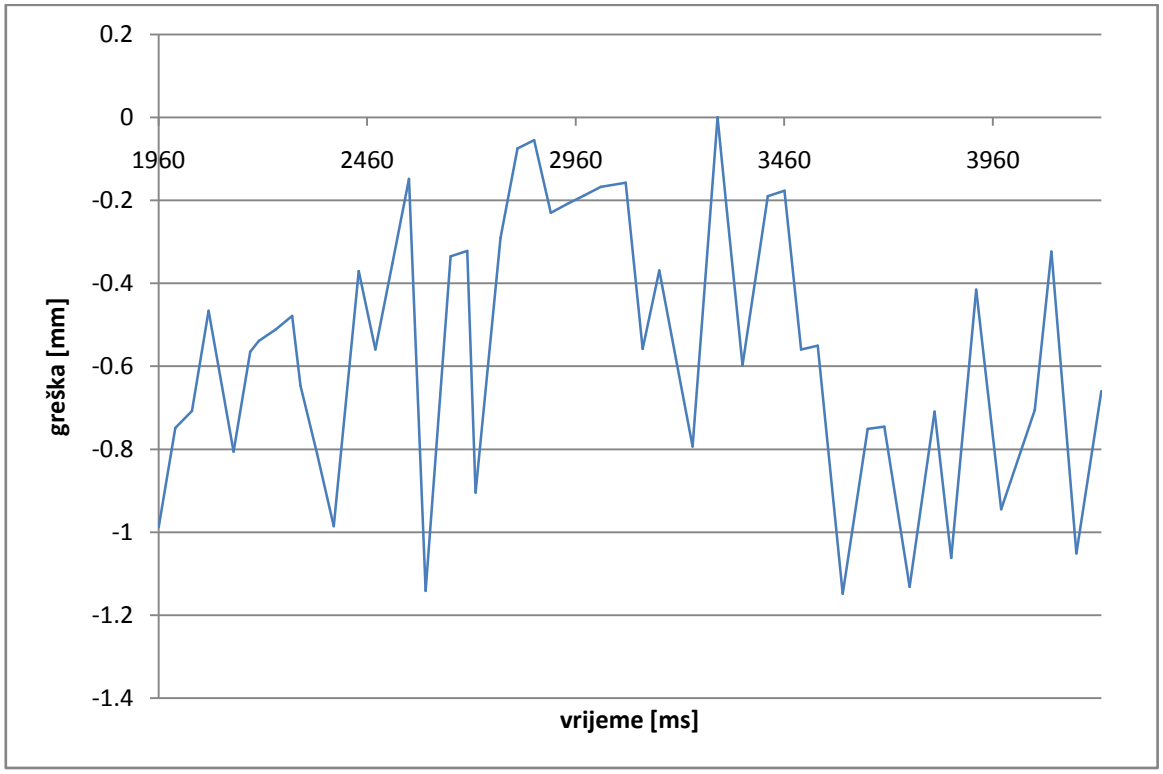
Slika 25: Brzina slave robota kod PD regulatora na 50 mm/s brzine master robota



Slika 26: Greška slave robota kod PID regulatora na 50 mm/s brzine master robota



Slika 27: Putanje *master* i *slave* robota kod PID regulatora po y osi i putanja *slave* robota po x osi za vrijeme spajanja pri brzini master robota od 50 mm/s



Slika 28: Greška PID regulatora u trenutku spajanja

Kao što se vidi PID regulator nudi najbolje od tri ponuđena regulatora. Greška za vrijeme spajanja je blizu nule i konstantna i nakon odvajanja regulator smanjuje regulacijsko odstupanje. Jedina mana je što dinamički član stvara šumove na izlazu iz regulatora, tj. brzini.

Pošto se PID regulator pokazao najboljim s njim se testiralo ponašanje na većim brzinama kretanja master robota.

Tabela 9: Mjerenja greške u trenutku spajanja PID regulatora

Min [mm]	Max [mm]	Prosjek [mm]
-1.14902	0.00003	-0.548961765

Iz Tabela 10 i Slika 21 iSlika 28 se vidi da PI regulator ima veće oscilacije za vrijeme spajanja (8.2576 mm) usporedno s PID regulatorom (1.14905 mm) i također se vidi da PID regulator oscilira oko svoje srednje vrijednosti dok PI regulator iako nema velike oscilacije greška mu je u porastu tijekom spajanja. PD regulator zbog nedostatka integralnog člana ima preveliku kontinuiranu grešku od 13.955mm i samim time ne može početi spajanje.

Tabela 10: Usporedba grešaka za vrijeme spajanja PI i PID

	Min [mm]	Max [mm]	Prosjek [mm]
PI regulator	-2.77765	5.47995	1.818189483
PID regulator	-1.14902	0.00003	-0.548961765

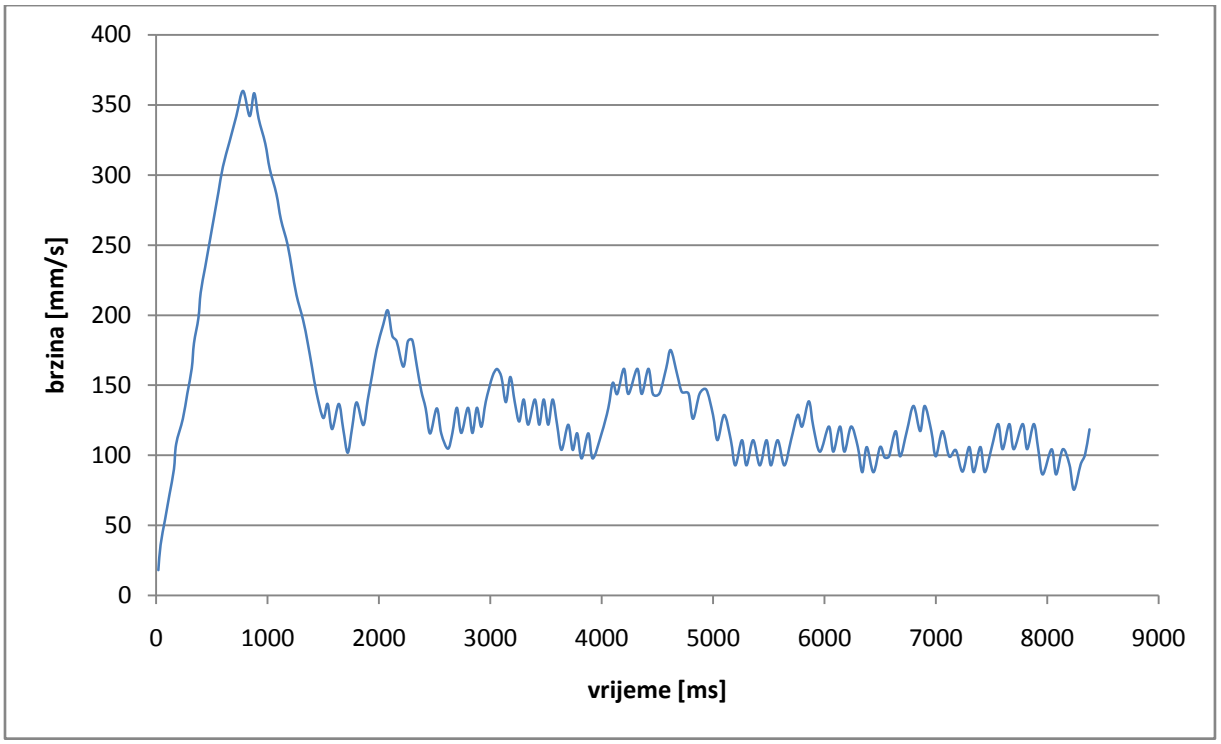
5.4 Testiranje PID regulatora na većim brzinama

Regulator je testiran nakon prvog testa gdje *master* robot ima brzinu od 50 mm/s, na brzinama *master* robota od 100, 150, 200 i 300 mm/s

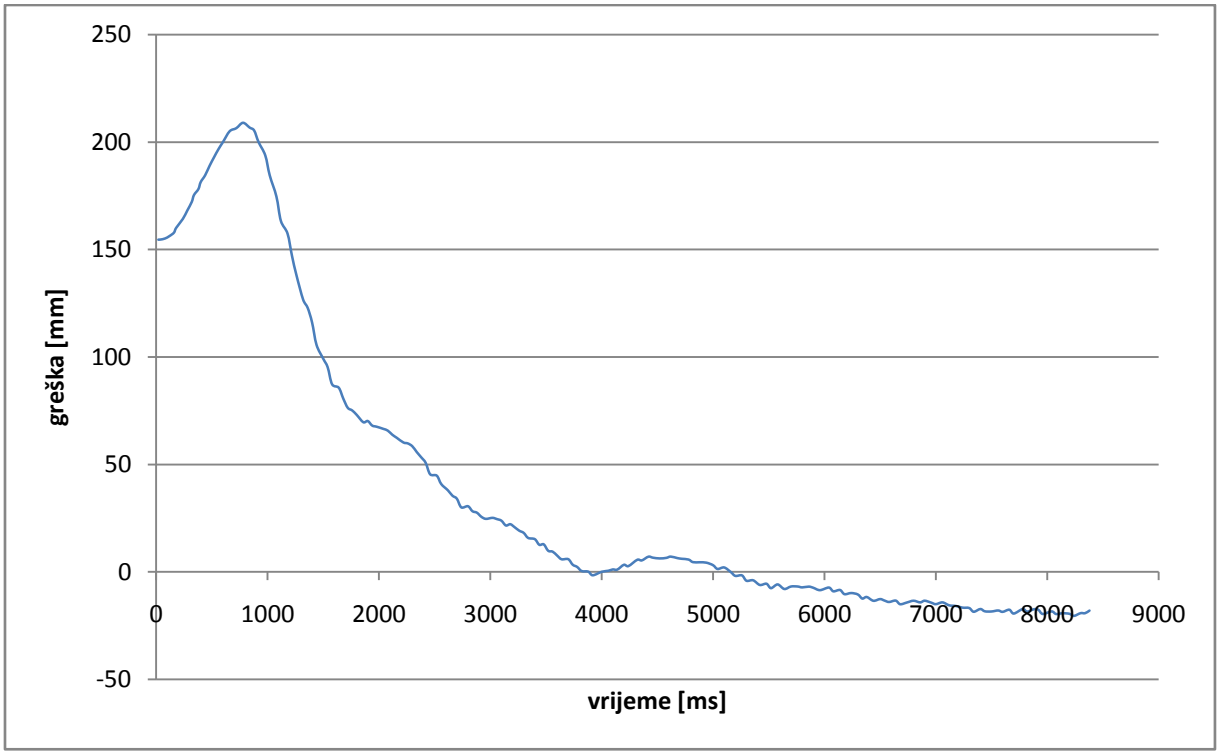
S promjenom brzine mijenja se i dinamika sustava s čijom promjenom je potrebno zadavati druge parametre regulatora, ali i dolazi do izražaja prespora komunikacija robota koje je 0.45 ms, tj. ~22.2Hz što znači da što je veća brzina *slave* robot pređe veću udaljenost prije nego dobije podatak o novoj koordinati *master* robota. S time se povećavaju oscilacije u sustavu i smanjuje se broj podataka koje robot ima na raspolaganju za regulaciju.

Nakon brzine od 50 mm/s dobivanje parametara regulacije koji dali odziv imalo sličan odzivu regulatora na 50 mm/s se pokazalo zahtjevno bez dinamičkog i kinematičkog modela robota matematički pronađenih pojačanja. Također se pojavio problem maksimalne brzine na kojoj *slave* robot može dostići *master* robota.

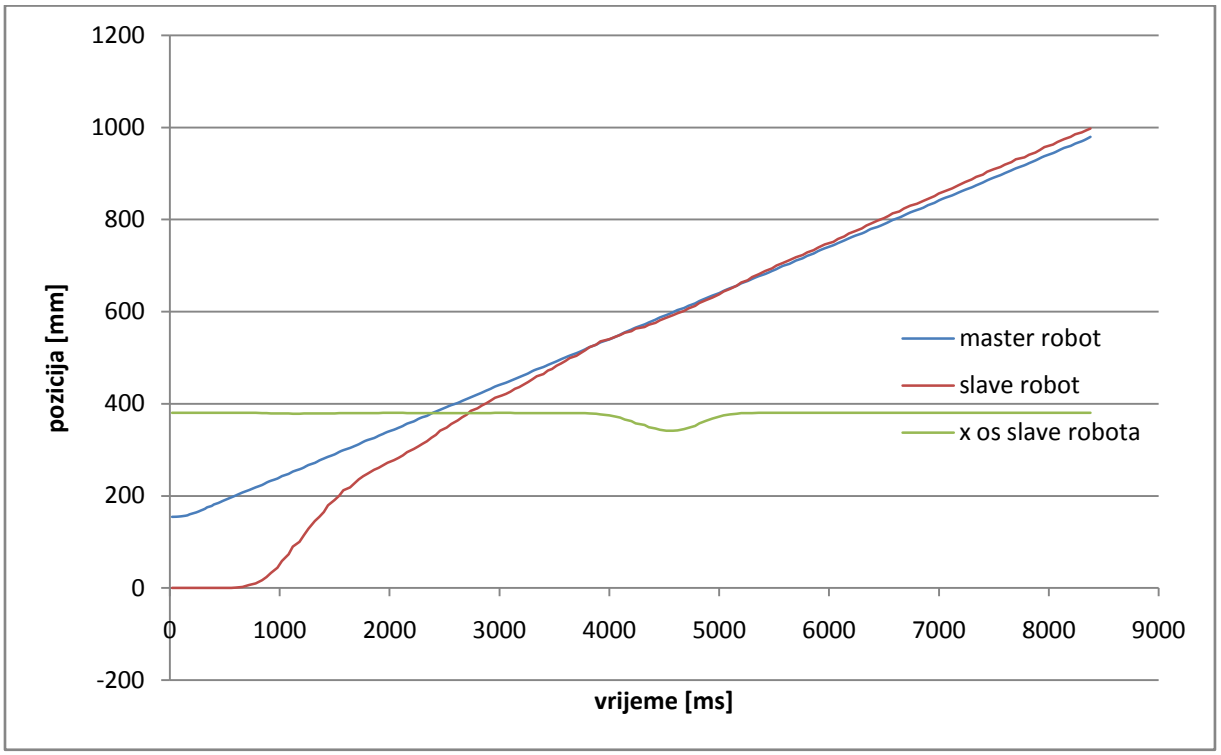
5.4.1 100 mm/s



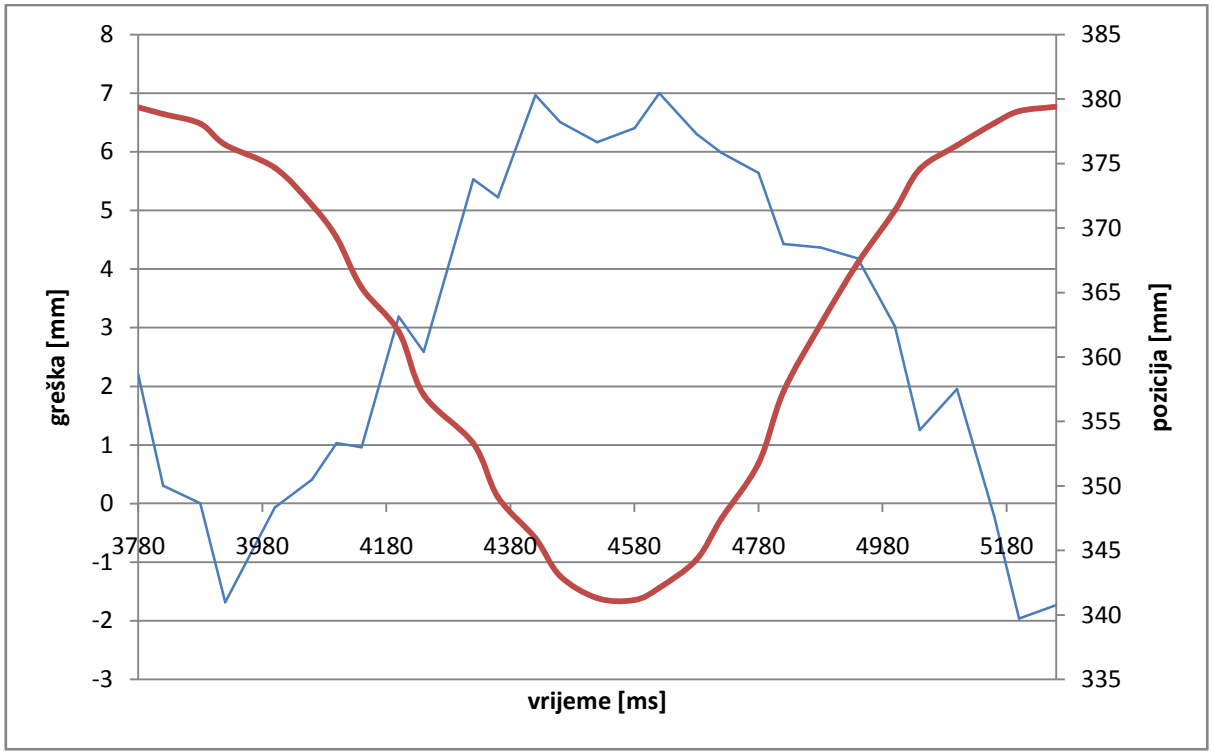
Slika 29: Brzina slave robota kod PD regulatora na 100 mm/s brzine master robota



Slika 30: Greška slave robota kod PID regulatora na 100 mm/s brzine master robota



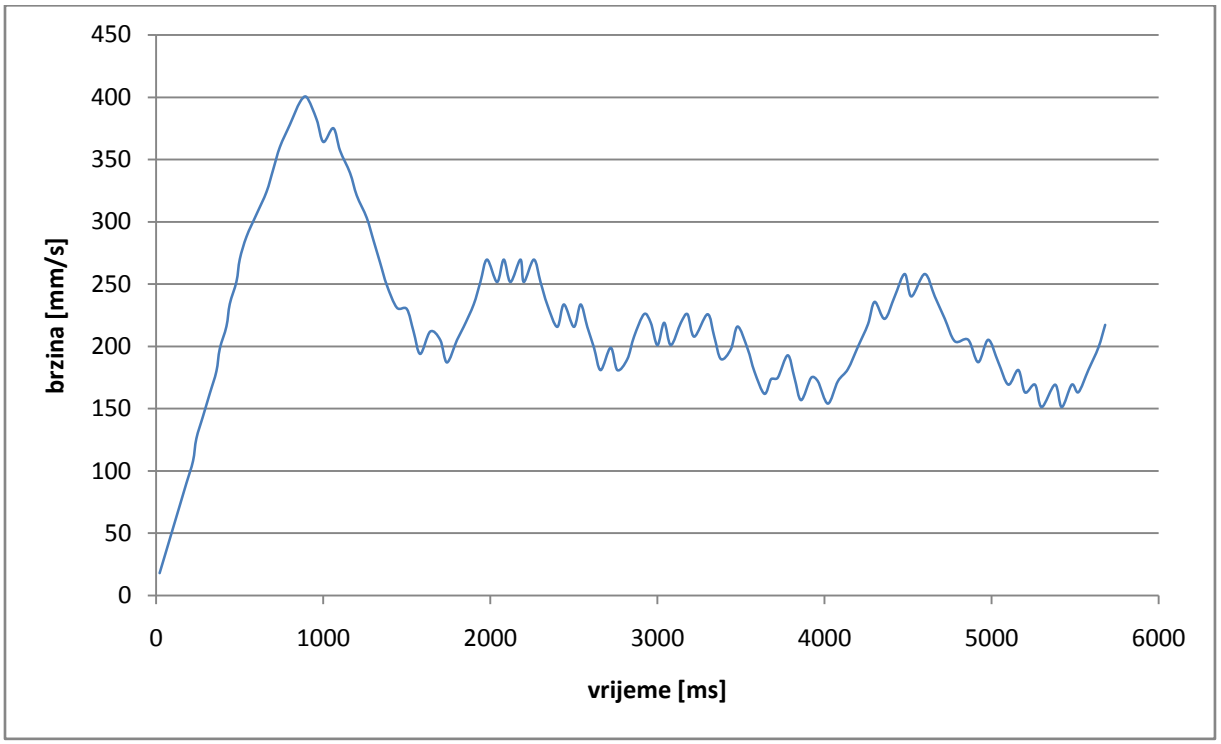
Slika 31: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 100 mm/s



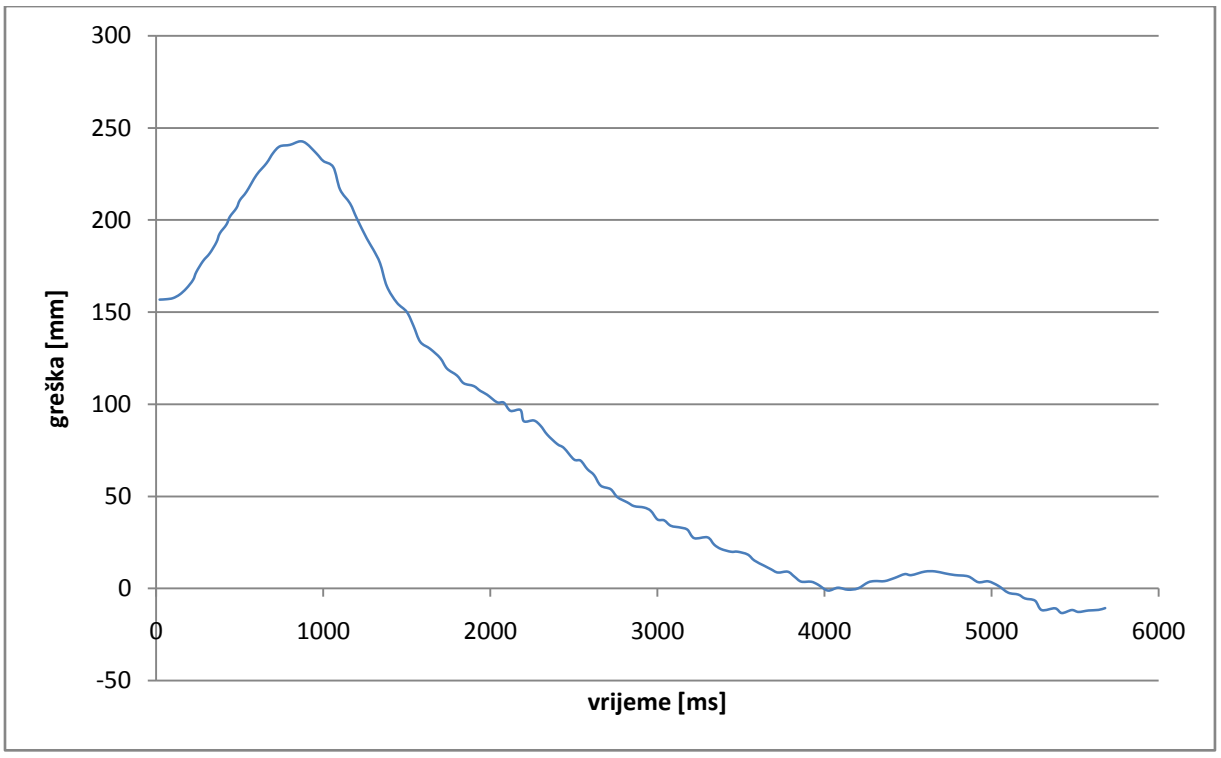
Slika 32: Greška za vrijeme spajanja na 100 mm/s

Kao što se vidi na Slika 30Slika 32Slika 33 regulator pri brzini od 100 mm/s ne može zadržati grešku stabilnom u trenutku spajanja greška raste ~-1.5 mm do 7mm i nakon toga se opet spušta prijašnju razinu i nastavlja opadati. Slika 29 pokazuje da se pojavljuju oscilacije u brzini robota koje osciliraju oko zadane vrijednosti od 100 mm/s.

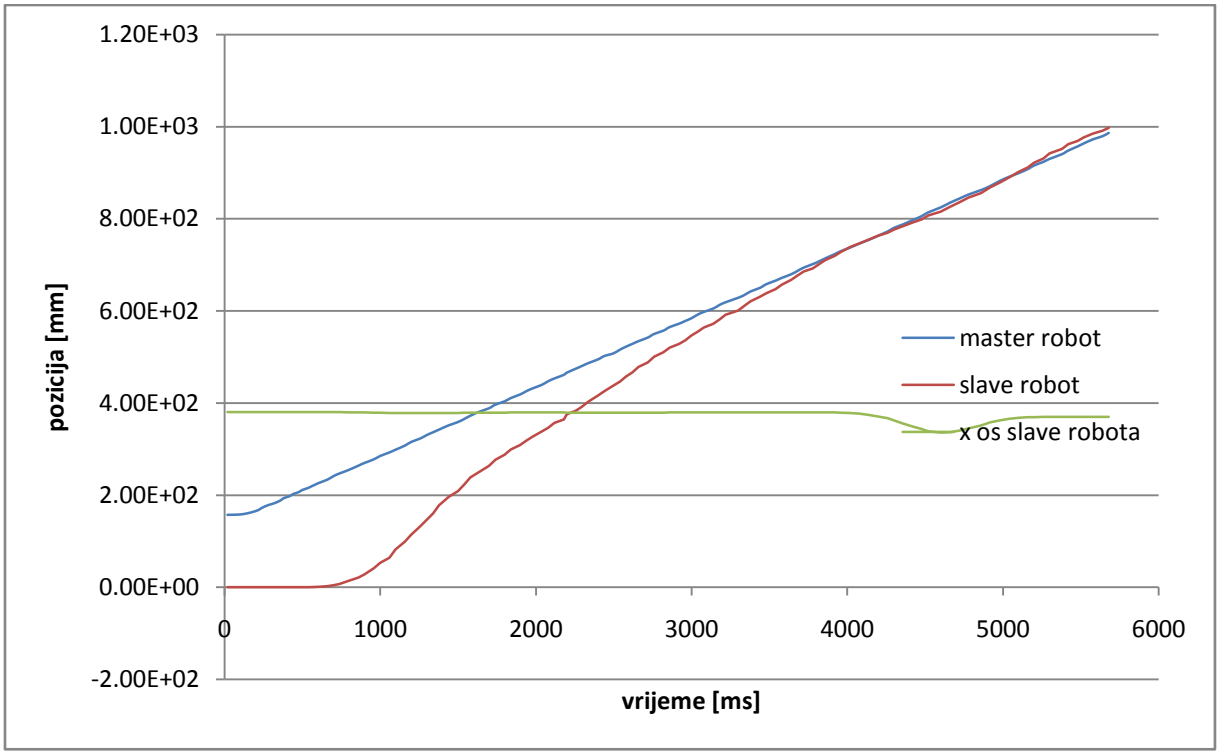
5.4.2 150 mm/s



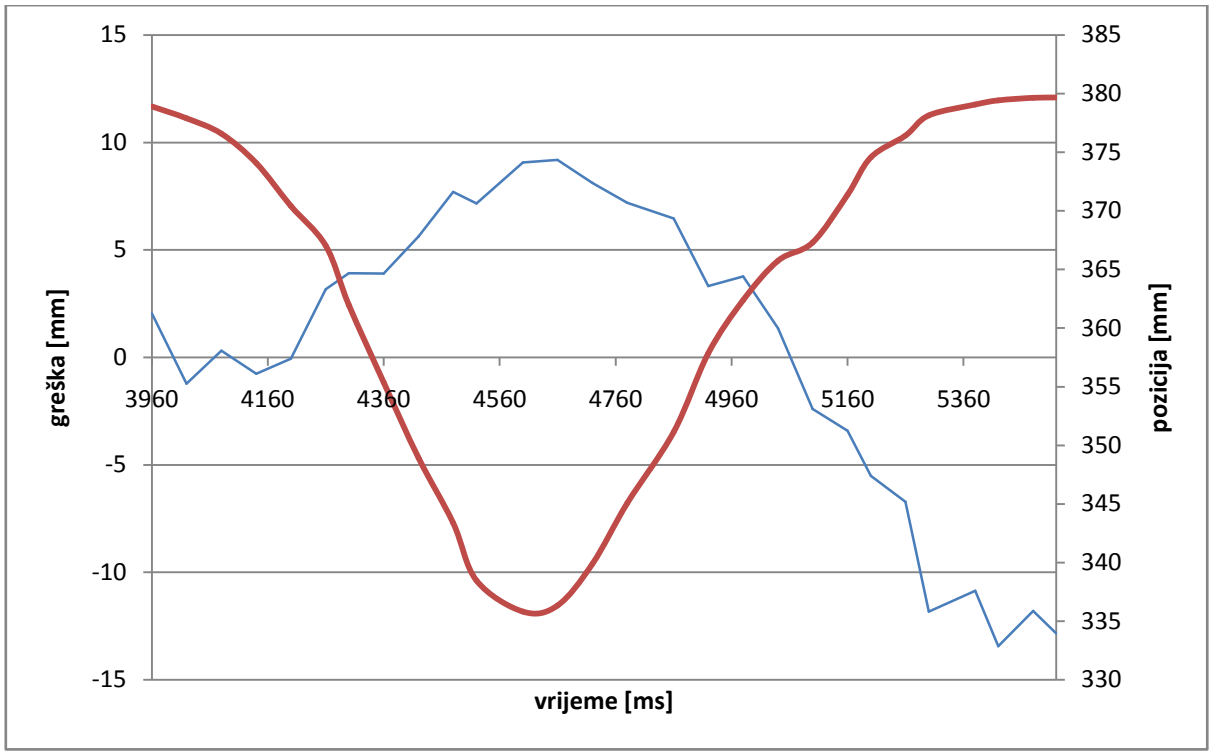
Slika 33: Brzina slave robota kod PD regulatora na 150 mm/s brzine master robota



Slika 34: Greška slave robota kod PID regulatora na 150 mm/s brzine master robota



Slika 35: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 150 mm/s

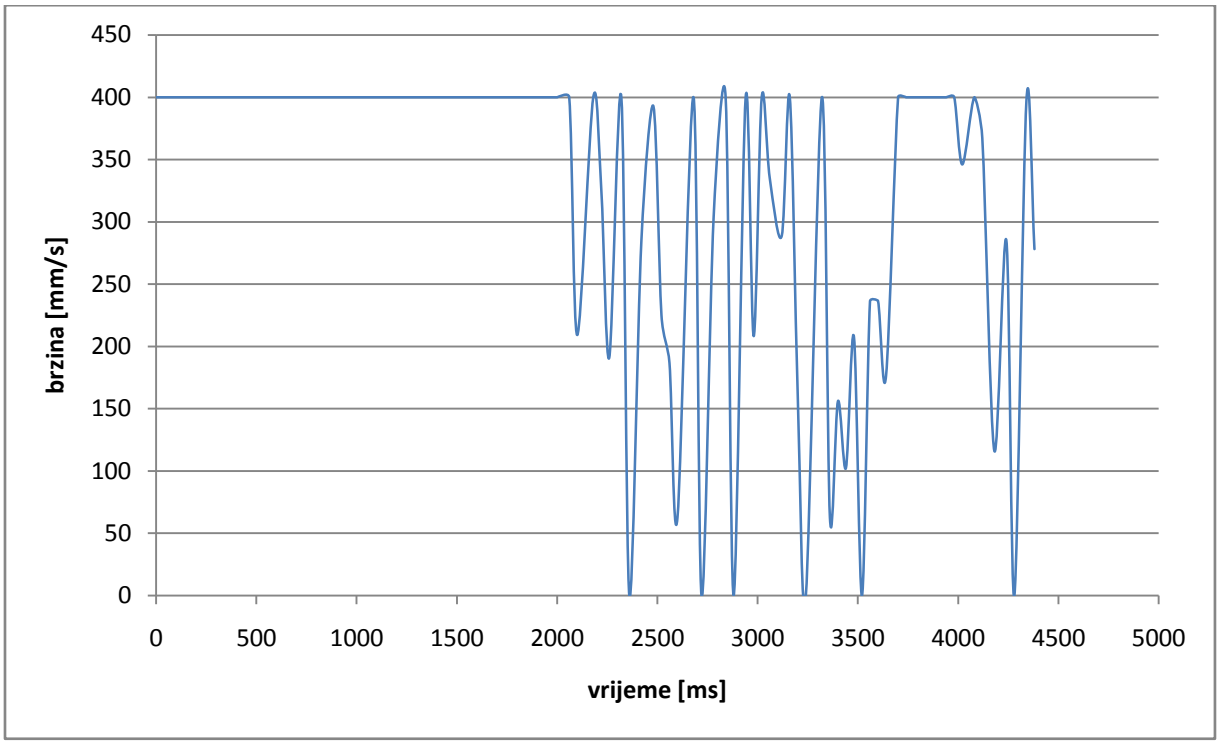


Slika 36: Greška za vrijeme spajanja na 150 mm/s

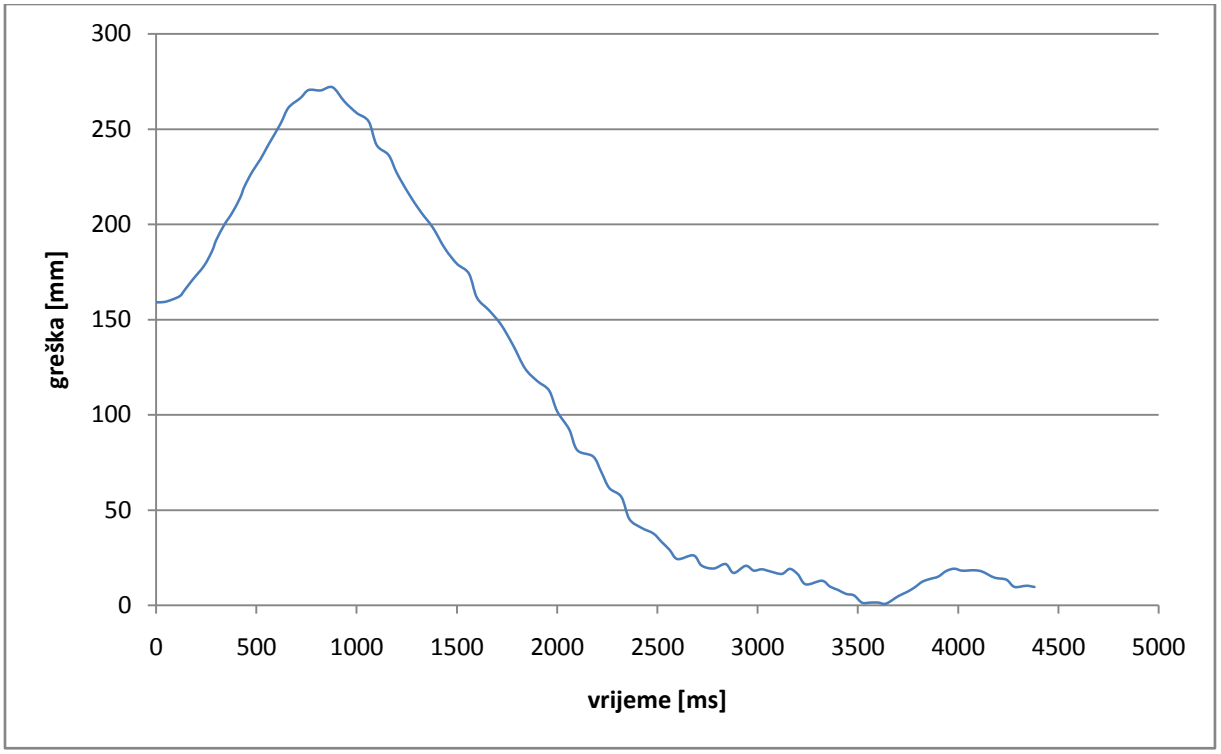
Kao i na prijašnjoj brzini od 100 mm/s Slika 34 iSlika 36 greška regulatora u trenutku spajanja raste s ~0 mm na ~10mm a nakon prestanka spajanja opada na razinu na kojoj bila da se dinamika sustava nije promijenila tj. da nije bilo izvršeno spajanje. Uspoređujući putanje

robota (Slika 31 i Slika 35) vidi se da robotu treba puno duže da dođe do trenutka spajanja. Uspoređujući brzine (Slika 29 i Slika 33) vide se puno veće oscilacije pri brzini od 150 mm/s.

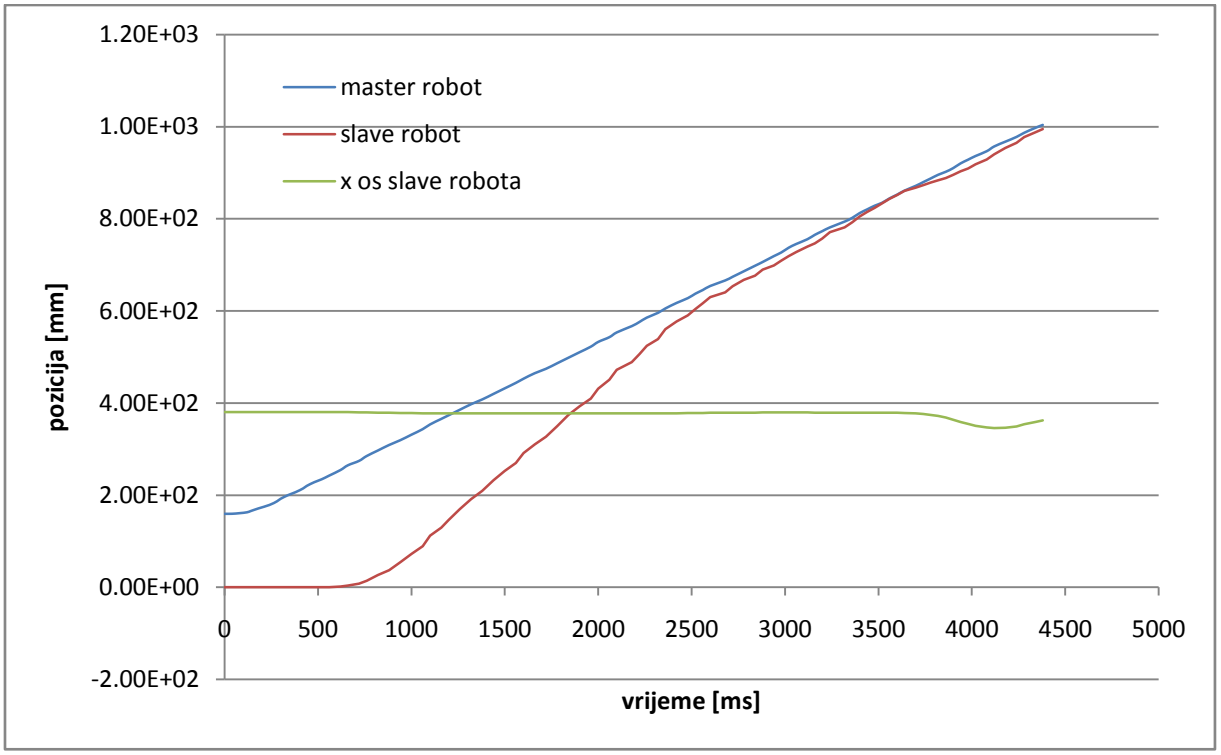
5.4.3 200 mm/s



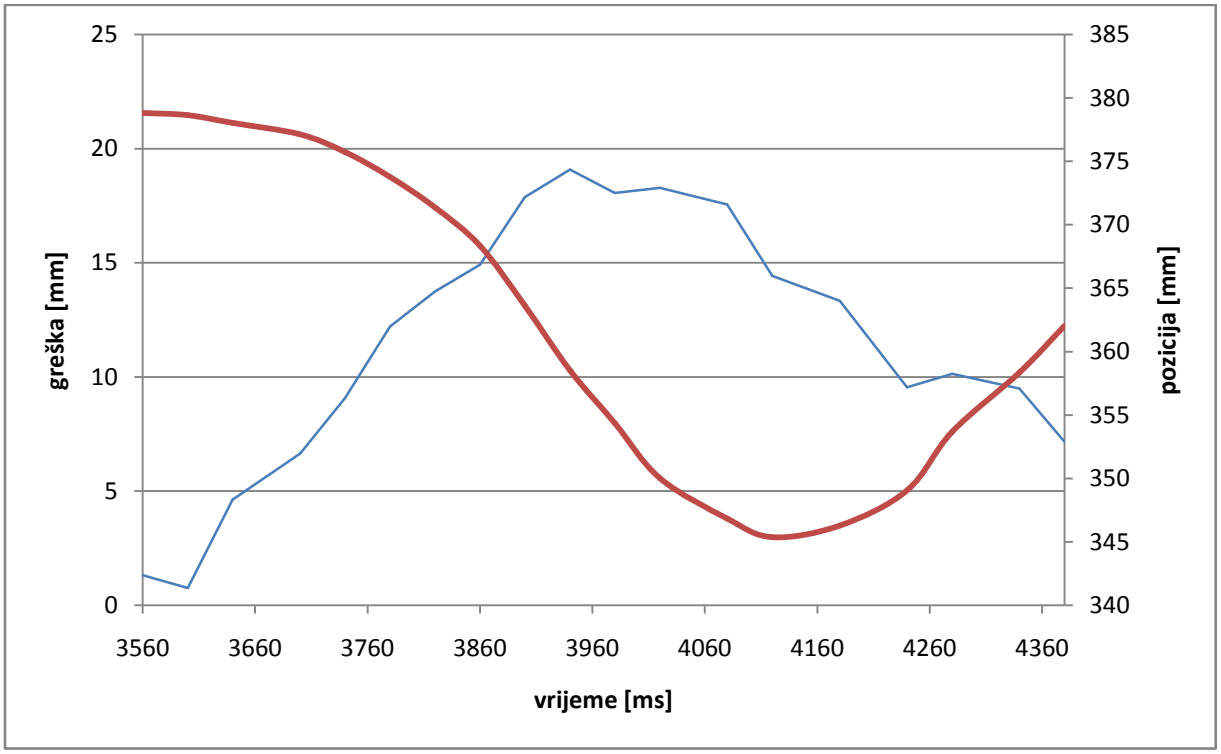
Slika 37: Brzina slave robota kod PD regulatora na 200 mm/s brzine master robota



Slika 38: Greška slave robota kod PID regulatora na 200 mm/s brzine master robota



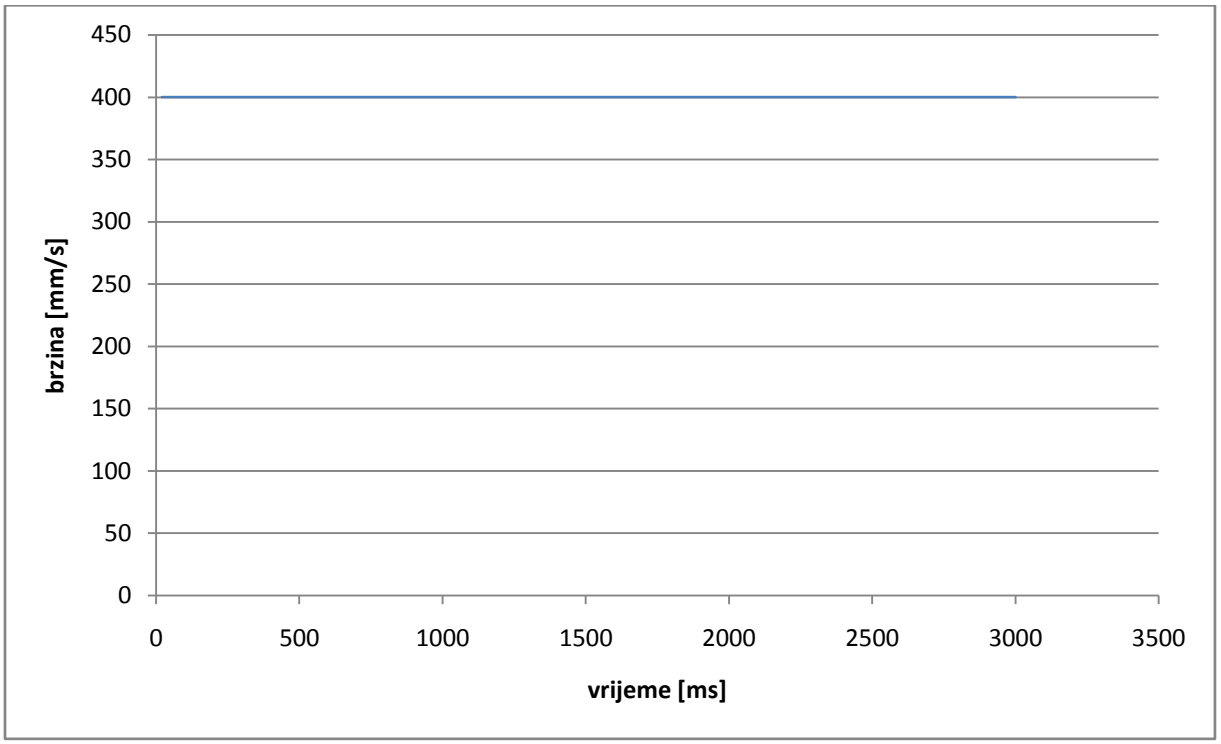
Slika 39: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 200 mm/s



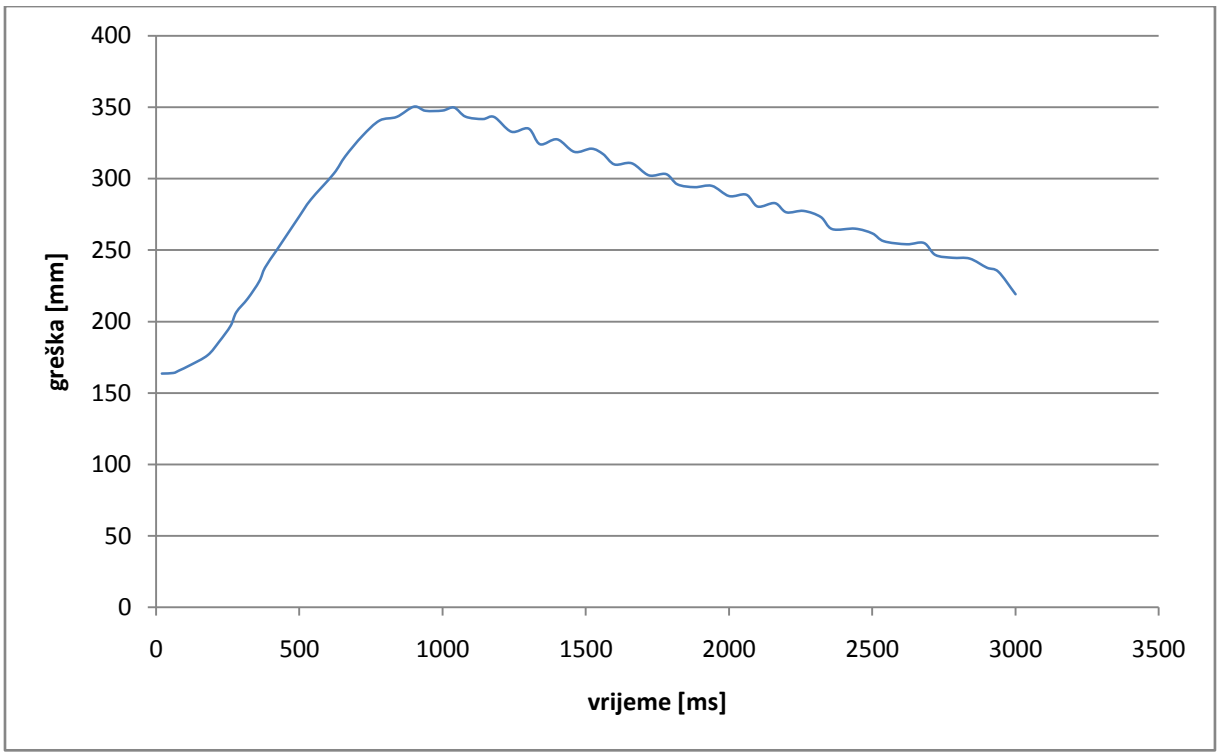
Slika 40: Greška za vrijeme spajanja na 200 mm/s

Iz Slika 39 i Slika 40 se vidi da spajanje počinje tek na kraju putanje po y osi i da se ne dovrši do kraja. Slika 37 pokazuje da *slave* robot se giba svojom maksimalnom brzinom većinu puta i kad sustigne *master* robota pojave se velike oscilacije u brzini.

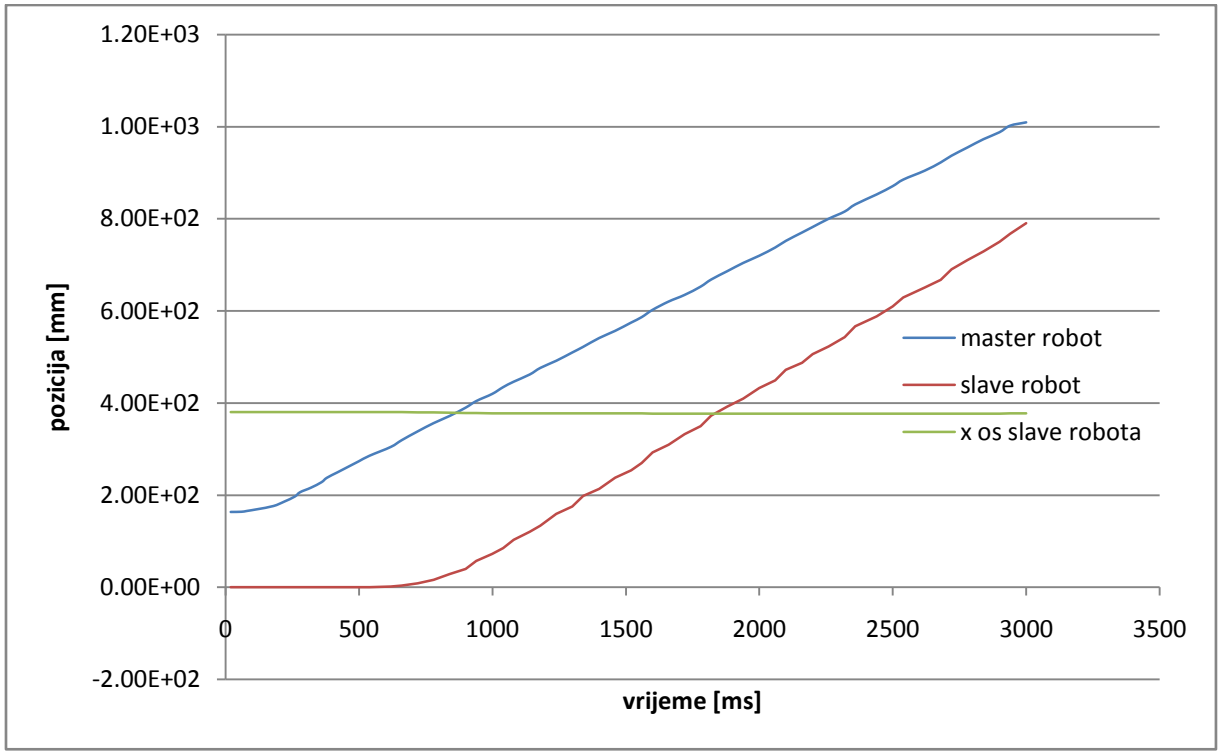
5.4.4 300 mm/s



Slika 41: Brzina slave robota kod PD regulatora na 300 mm/s brzine master robota



Slika 42: Greška slave robota kod PID regulatora na 300 mm/s brzine master robota



Slika 43: Putanje master i slave robota kod PID regulatora po y osi i putanja slave robota po x osi za vrijeme spajanja pri brzini master robota od 300 mm/s

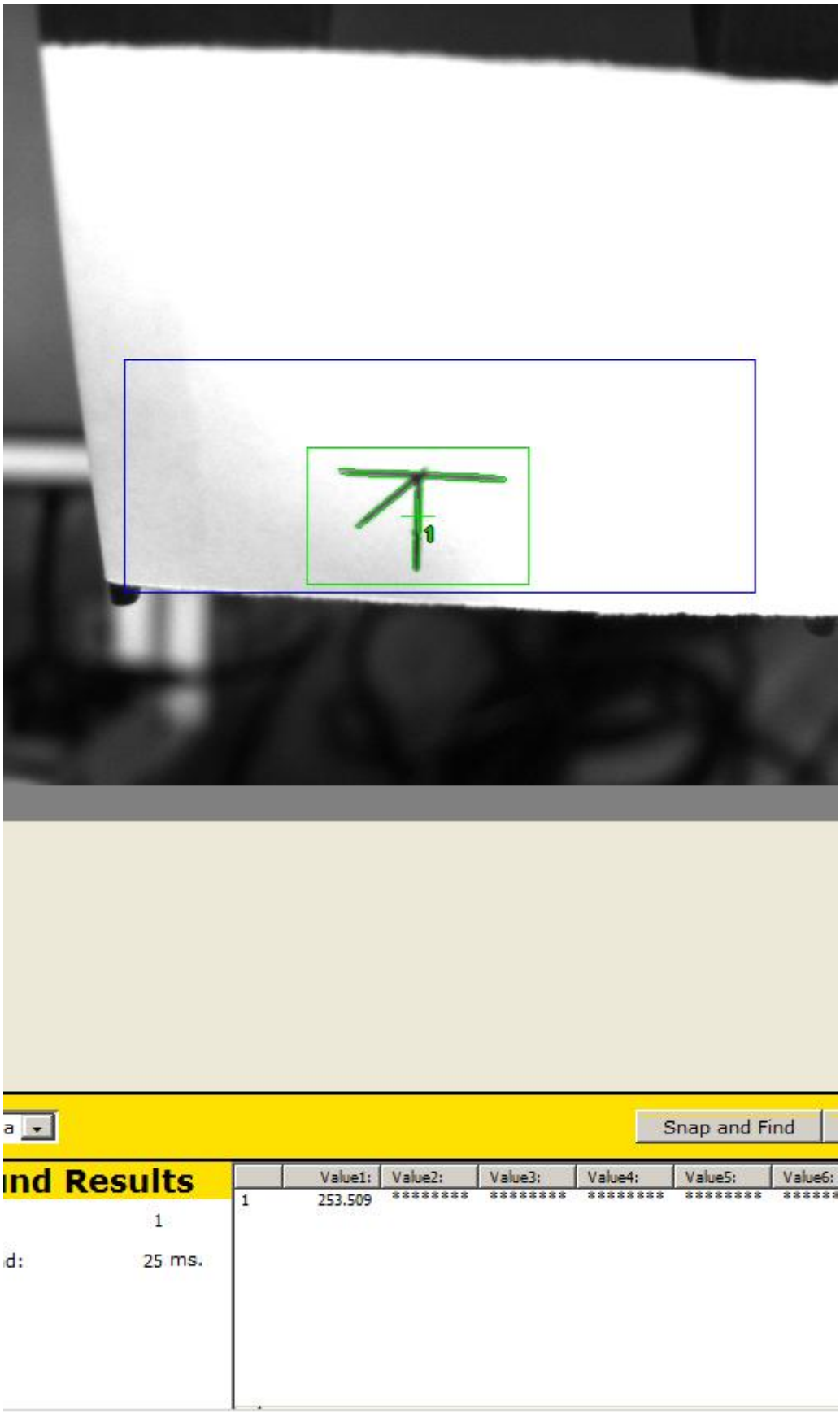
Na brzini od 300 mm/s se vidi da *slave* robot jednostavno više ne može stići *master* robota u zadanom području rada (Slika 43). Ako *slave* robot ima maksimalnu brzinu od 400 mm/s, a *master* robot od 300 mm/s, *master* robot pređe put od 850 mm za vrijeme od 2.83 sekunde. a *slave* robot put od 1000mm prođe za 3 sekunde (2.5sekundi putovanje + 0.5 sekundi kašnjenja kretanja) vidi se da *slave* robot ne može stići *master* robota jer *master* robot završi svoje kretanje prije nego slave robot može ga dostići pri kretanju konstantno pri maksimalnoj brzini (Slika 41).

6 Provjera sa vizijskim sustavom

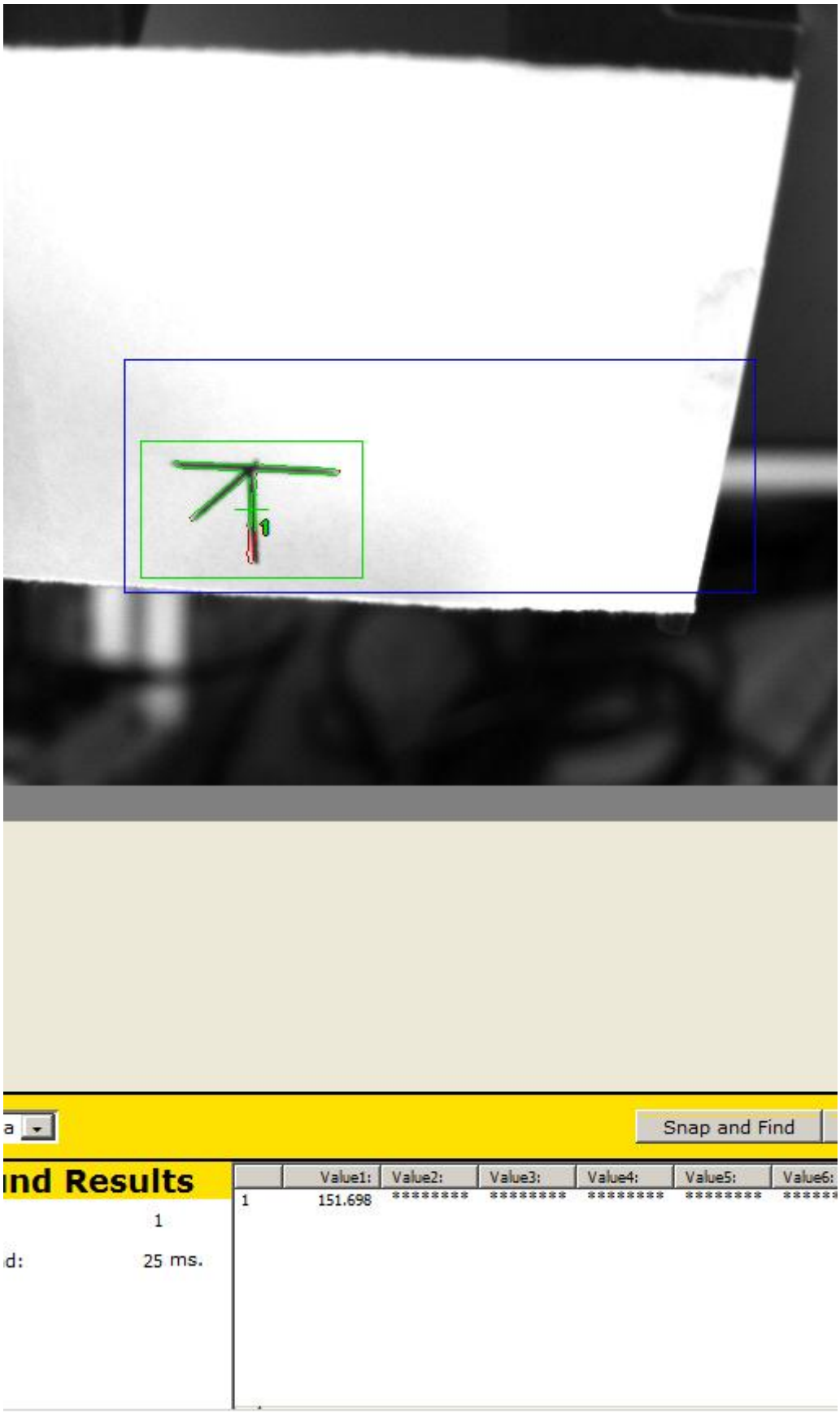
Svi podatci do sad su dobiveni iz robota preko njihovih enkodera i kinematskih modela. Zbog toga što roboti nisu savršeno kruti, enkoderi točni i kalibracije nisu savršeno napravljene, nastaju greške zbog kojih podatci iz robota ne odgovaraju realnosti tj. imaju odstupanja pa je zbog toga potrebno je bilo napraviti usporedbu s nezavisnim sustavom mjerenja. To se napravilo preko vizijskog sustava.

Dok su roboti u pokretu i u trenutku spajanja kamera traži *pattern* i njegovo težište. Centru tog težišta daje vrijednost stupca slike u pikselima kojima se vrijednost kreće od 0-512 jer je rezolucija slike 512x480 piksela.

Prije toga se roboti pozicioniraju tako da oboje imaju iste y koordinate, uzme se slika *patterna* u tom trenutku i očita se vrijednost stupca. Nakon toga se jedan od robota pomakne za određenu vrijednost i uzme se druga slika i očita se vrijednost stupca. U ovom slučaju roboti su poravnati i vrijednost stupca je bila 253.509 piksela (Slika 44). Nakon toga se *slave* robot pomaknuo za 15 mm i vrijednost stupca u tom trenutku je iznosila 151.698 piksela (Slika 45).



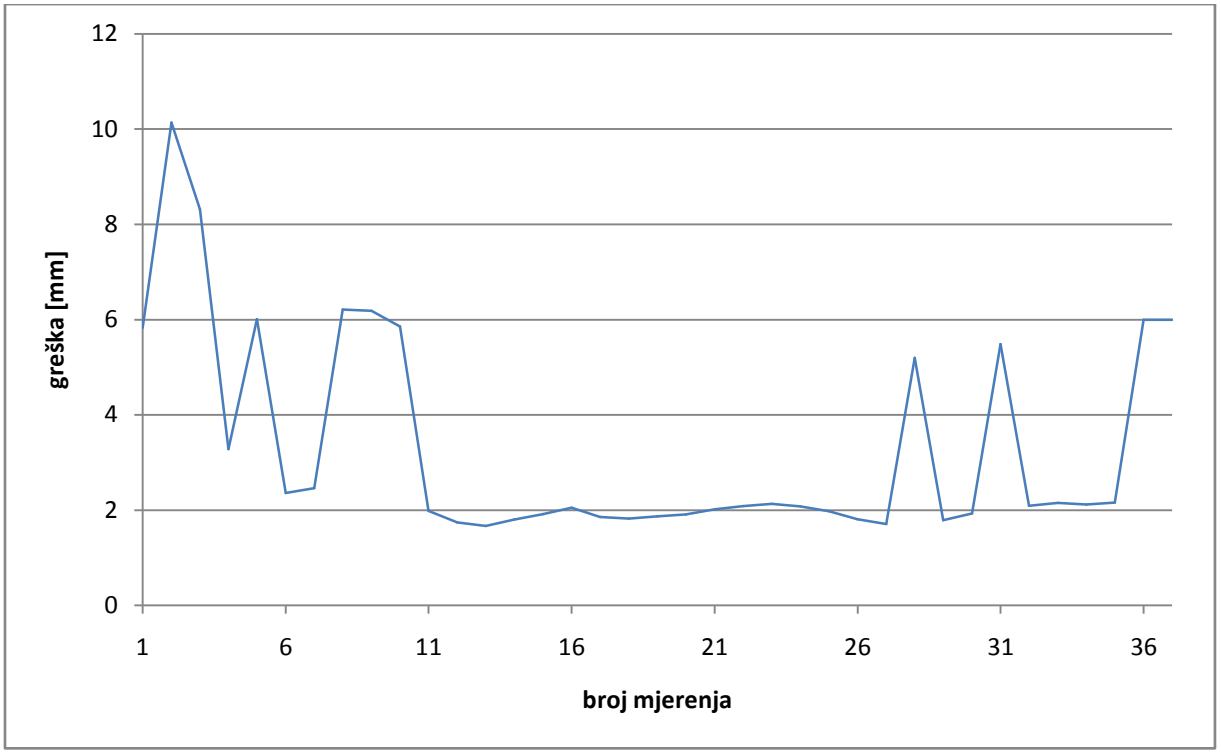
Slika 44: Očitanje prve vrijednosti



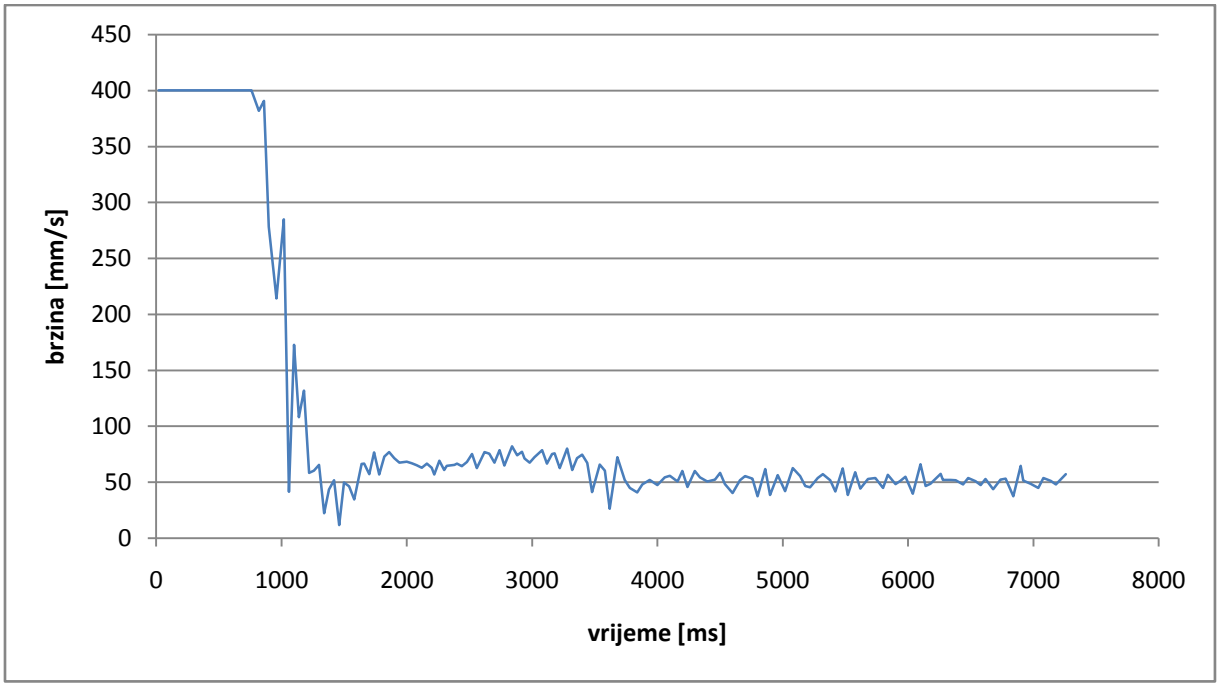
Slika 45: Očitanje druge vrijednosti

Oduzimanjem ta dva broja se dobije da 15mm je jednako 101.811 piksela. Dijeljenem tog broja s 15 dobijemo vrijednost jednog milimetra u pikselima i iznosi 6.787 piksel. Nakon toga od očitane vrijednosti koje smo dobili iz kamere oduzimamo vrijednost stupca za vrijeme kada su roboti bili poravnati (282.206) i taj rezultat dijelimo sa 6.791 za vrijednost u milimetrima.

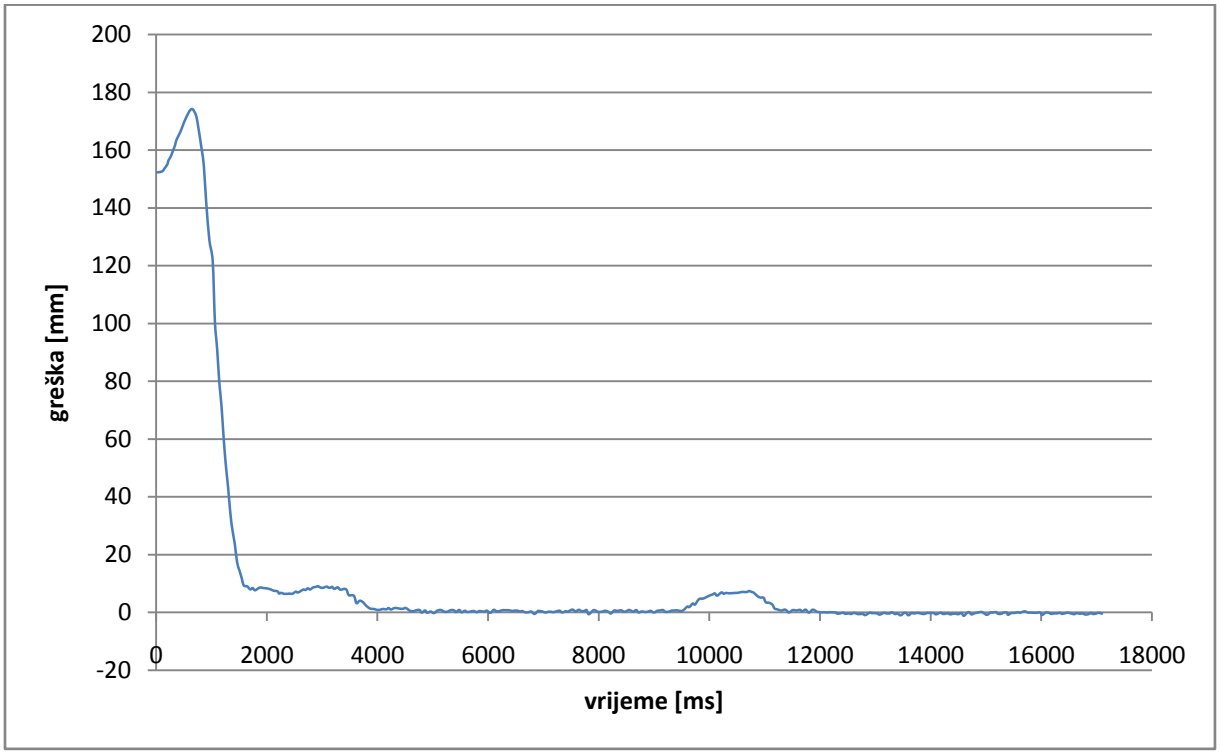
Rezultati mjerenja i usporedba rezultata:



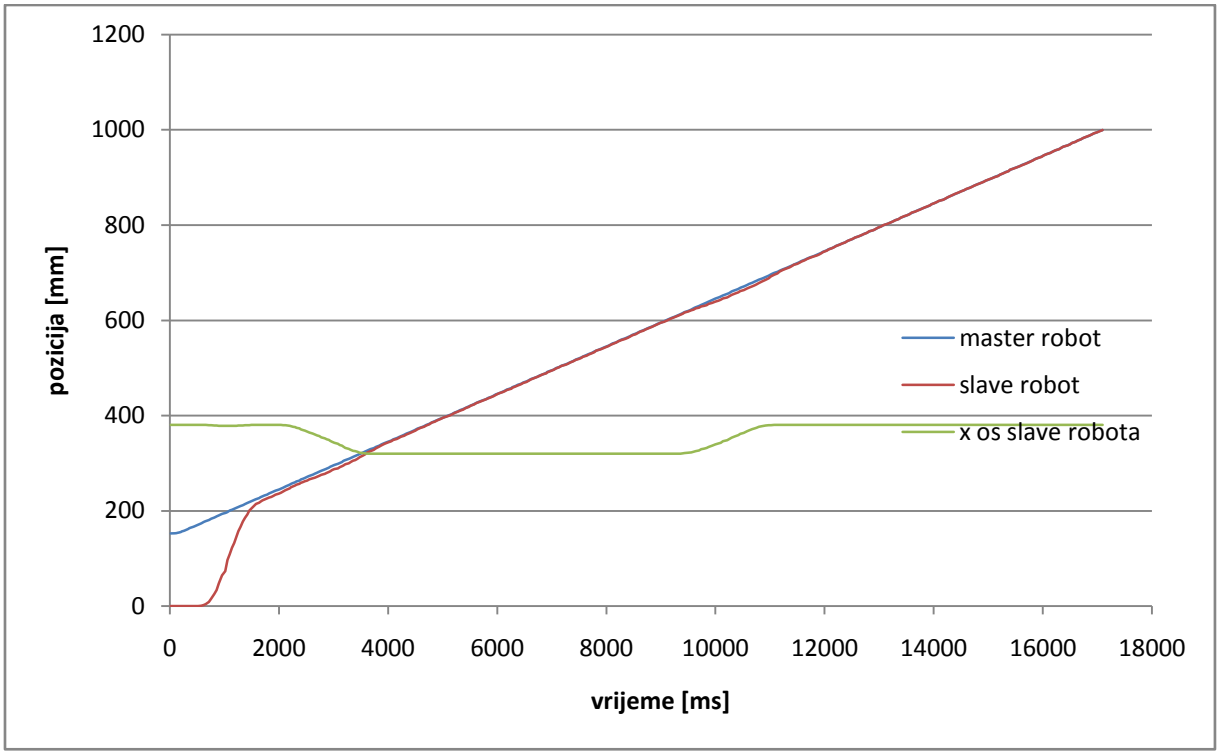
Slika 46: Greška sustava dobivena preko vizijskog procesa



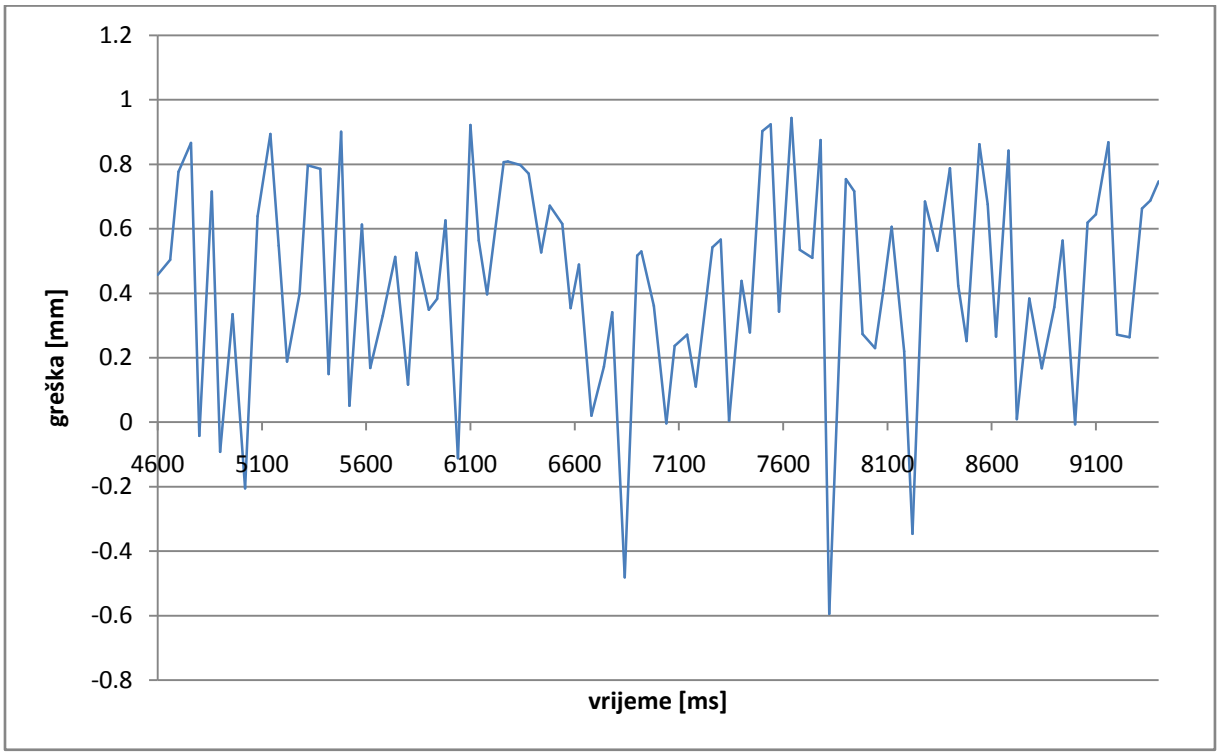
Slika 47: Brzina robota



Slika 48: Greška robota



Slika 49: Putanja *slave* i *master* robota



Slika 50: Greška za vrijeme spajanja –podatci iz robota

Tabela 11: Usporedba rezultata vizijskog sutava i podataka iz robota

	Min [mm]	Max [mm]	Prosjek [mm]
Robot	-0.59442	0.94311	0.44404
Vizija	1.66352	10.1371	3.34774

Kao što se vidi iz Tabela 11 i Slika 46 i Slika 50 iz robota se dobiju podatci o puno većoj točnosti s manjim oscilacijama nego iz podataka iz vizijskog sustava.

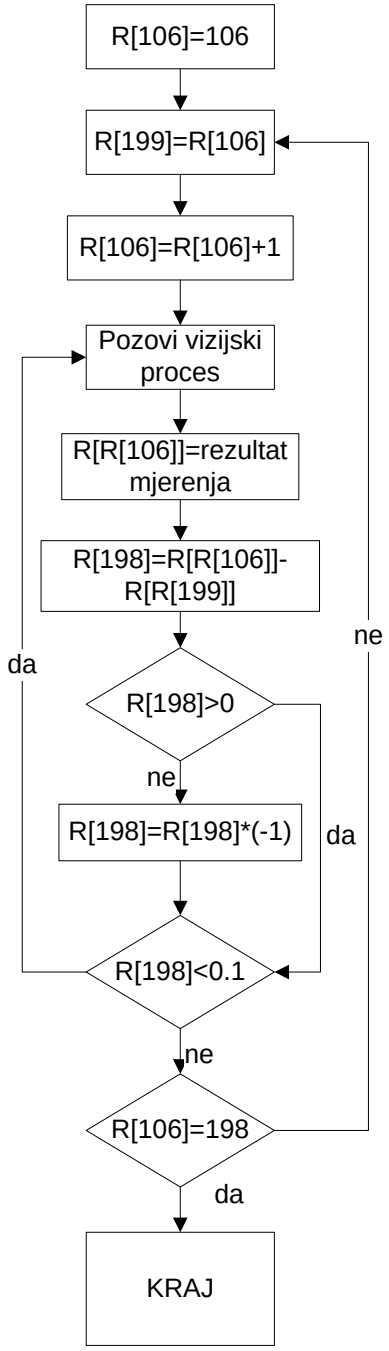
6.1 TP program za mjerenje greške

Slave robot zbog svojih mogućnosti nije mogao uz gibanje, regulaciju pokrenuti dodatan program vizije pa je zbog toga korišten treći robot na kojeg se priključila kamera *slave* robota.

Za taj robot je napravljen TP program koji poziva vizijski proces, očitava vrijednost i ako je vrijednost različita od prethodne zapisuje je u registar tako da svaka nova vrijednost je zapisana u svoj registar. Zbog nesavršenosti snimanja dok bi roboti bili stacionarni svako novo mjerenje bi bilo različito od prethodnog za par decimala. Da bi se spriječilo nepotrebno punjenje registra postavljen je uvjet koji zahtjeva da sljedeći i prethodni broj budu različiti bar za 0.1 da bi se to smatralo novom vrijednosti.

Problem ovakve postave je što se ne može dobiti u kojem trenutku je kamera snimila i izmjerila grešku nego se može samo dobiti usporedba s brojevima iz robota. Za ovaj test je promijenjena putanja robota da umjesto kad dosegne krajnju točku spajanja u x koordinati odmah krene s odvajanjem, robot se nastavi kretati još 300mm u smjeru osi y prije početka odvajanja. To je napravljeno da se osigura da kamera snima samo za vrijeme spajanja – da bi površina slike koju program mora pretraživati bila što manja i time ubrzala snimanje i da se dobije što više podataka.

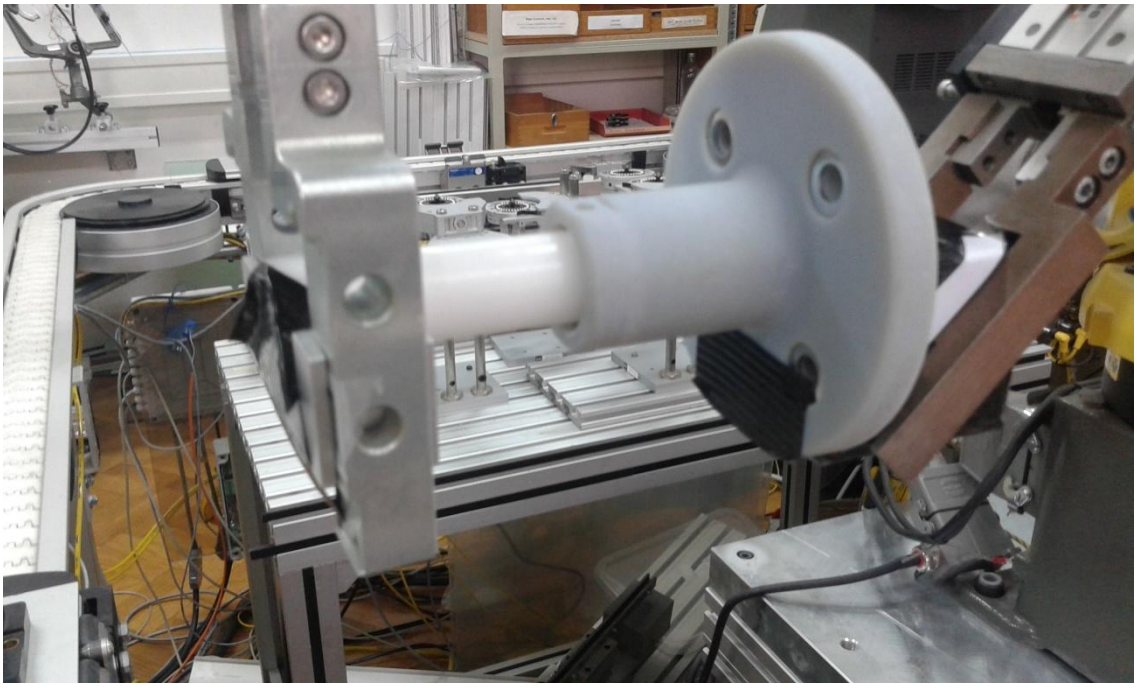
Cijeli TP program se nalazi u dodatku H.



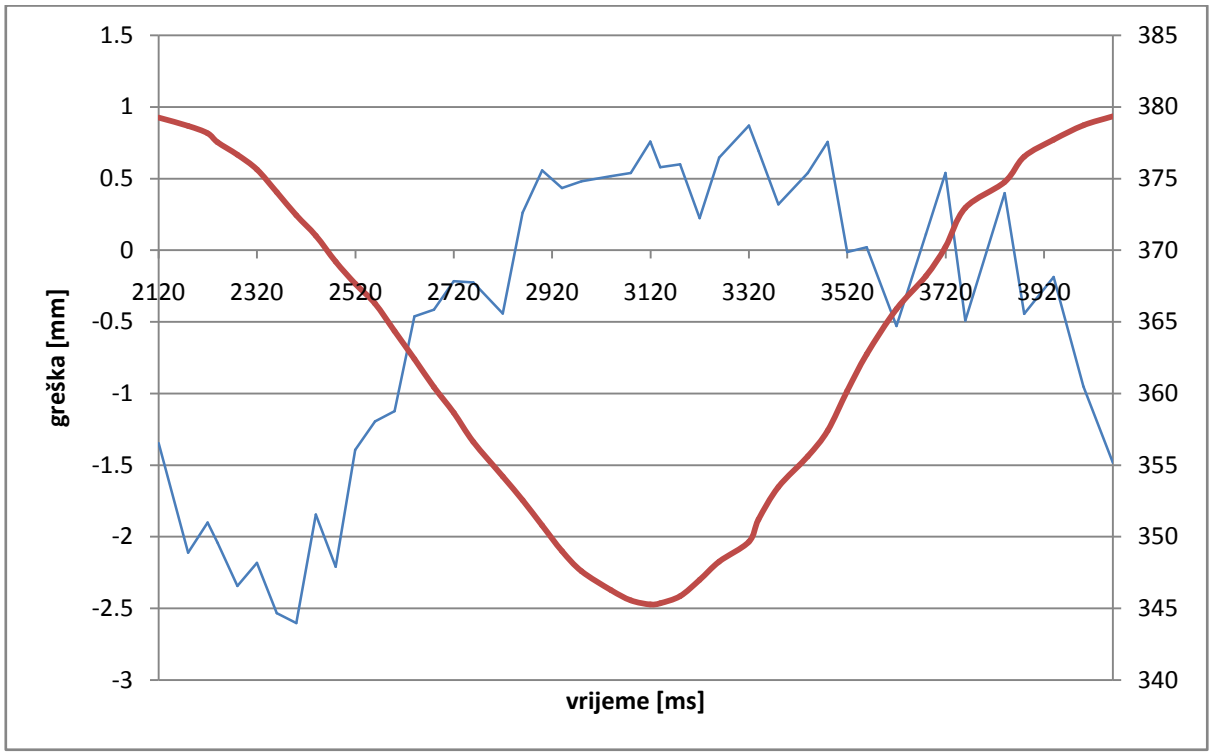
Slika 51: Diagram toga TP programa

7 Testiranje spajanja

Test spajanja izvršen je sa cilindričnim predmetom promjera ϕ20.8 mm dužine 73.6mm koji je pričvrščen na *slave* robota u drugi cilindrični predmet vanjskog promjera ϕ35 i unutarnjeg promjera ϕ23.5mm. Tolerancija kod spajanja je 2.7mm.



Slika 52: Predmeta za spajanje



Slika 53: Greška u odnosu na poziciju x osi za vrijeme spajanja

Iz Slika 53 se vidi da je greška kad *slave* robot s predmetom počne ulaziti u greška regulatora je oko 0.5mm što je dovoljno da uz tolerancu od 2.7 mm se izvrši spajanje bez kolizije. Korišten je PID regulator kao u poglavlju 32 na brzini od *master* robota od 50 mm/s.

8 Zaključak

U radu je pokazano eksperimentalno praćenje i spajanje u pokretu s dva robota koja komuniciraju preko *Ethernet* protokola. Ovakav sustav bi ubrzao klasičnu montažu s pokretnom trakom gdje se prilikom montaže traka zaustavlja dok se montaža obavlja. Pri velikim serijama i mala ušteda vremena pri montaži jednog sklopa bi kroz više smjena i dana i godina značila velike uštede.

Regulacija preko *Ethernet* protokola se pokazala složenom zbog brzine veze između robota, *input buffera slave* robota, koji prilikom dobivanja podataka od *master* robota ukoliko mu proces obrade tih podataka nije brz kao i kao i vrijeme dobivanja istih u svoj *buffer* sprema podatke dok se ne obrade i tako sprječava *real-time* obradu podataka koja je potrebna za regulaciju, i promjene dinamike robota s promjenom brzine gibanja, što se vidjelo tako da pri različitim brzinama kretanja robota isti parametri pojačanja ne daju iste rezultate sustava,i dodatnog gibanja u x osi zbog spajanja, koje zahtjeva veću brzinu robota jer se ono sada kreće umjesto u jednoj osi s brzinom istom kao i kod *master* robota u dvije osi i rezultantna brzina robota je time veća što također dovodi do promjene dinamike sustava. Pri većim brzinama robot nije imao dosta podataka jer su dolazili sa sporom frekvencijom (22.2 Hz) pa je regulacija bila skoro pa nemoguća. Minimalna frekvencija potrebna za ovaj sustav robota je 100 Hz a naravno poželjno je da bude i veća.

PID regulatori su primjenjivi u različitim vrstama problema i često njihove performanse su zadovoljavajuće ali generalno ne pružaju optimalnu kontrolu. Glavni problem s PID regulatorom je sustav povratne veze i konstantni parametri i bez matematičkog modela sustava njegovo djelovanje je reaktivno i radi na kompromisima. PID regulator je najbolji regulator u procesima gdje nemamo modela sustava i gdje unutarnje stanje sustava temeljimo na ulaznim i izlaznim parametrima.

PID regulatori su za ovakav tehnički zahtjev gdje ima dosta varijabli sustava (brzine robota, početna udaljenost robota od robota, osi/ravnine kretanja, rotacije robota) gdje pri promjeni jednog od njih regulator ne reagira isto, gdje je sustav dosta složen za opisati matematički (roboti sa 6 stupnjeva slobode gibanja) i gdje u primjeni u realnosti na montažnoj liniji pri svakoj promjeni uvjeta spajanja bi trebalo ponovo namještati parametre pojačanja koji

nemaju matematički model po kojem se namještavaju već se namještavaju iskustveno i na temelju izlaznih podataka nisu primjereni u ovakvom obliku.

Poboljšanje ovog sustava bi zahtijevalo uvođenje brže komunikacije među robotima, matematički model cjelokupnog sustava koji bi nalazio parametre regulacije odmah pri promjeni varijabli sustava ili nalaženje načina regulacije koji nije preko PID regulatora. Također pokušati i s načinom regulacije i praćenja koji ne zahtijeva komunikaciju dva robota osim možda početne za sinkronizaciju kretanja, npr. korištenjem senzora sile.

Prilozi

I. CD-R disc

Literatura

1. *ISO Standard 8373:1994, Manipulating Industrial Robots – Vocabulary.*

2. titan fsb. [Mrežno] [Citirano: 17. 11 2015.]
http://titan.fsb.hr/~zkunica/nastava/pms/roboti_manip.pdf.

3. PID CONTROL. [Mrežno] [Citirano: 17. 11 2015.]
http://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf.

4. codeproject. [Mrežno] [Citirano: 17. 11 2015.]
http://www.codeproject.com/Articles/36459/PID-process-control-a-Cruise-Control-example.

5. Fanuc robotics. [Mrežno] [Citirano: 17. 11 2015.]
http://www.fanucrobotics.com/cmsmedia/datasheets/LR%20Mate%20200iC%20Series_10.pdf.

6. Fanuc robotics. [Mrežno] [Citirano: 17. 11 2015.]
http://www.fanucrobotics.com/cmsmedia/datasheets/M-10iA%20Series_13.pdf.

7. *FANUC Robotics SYSTEM R-30iA Controller KAREL Reference Manual.* s.l. : FANUC Robotics America, Inc., 2007.

8. saba.kntu.ac.ir. [Mrežno] [Citirano: 17. 11 2015.]
http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf.

DODATAK A

```
PROGRAM brzSmjer
%NOLOCKGROUP
%NOPAUSE = ERROR + COMMAND + TPENABLE
VAR
 STATUS,i,i2:INTEGER
 pod:FILE
 rand,x,y,z,w,p,r,j,e,v,a,b,d,rez,rez2,m,del:REAL
 ENTRY,iv,c,nb,Msat,Msat2,Msat3,Msat4:INTEGER
 SALJI, PRIMI: FILE
 rf:BOOLEAN
 p1,p2,p0,frm, frm2 : XYZWPR
 config_var:CONFIG
 s:STRING[100]
   ROUTINE OPEN_FILE_ (FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
   ROUTINE CLOSE_FILE_(FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2

BEGIN
      DOUT [144]=FALSE

--*********************************************************
       SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$OPER',0,STATUS) ;
       SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STATE',0,STATUS) ;
DELAY 20

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$COMMENT','SOUND',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$PROTOCOL','SM',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$REPERRS','FALSE',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$TIMEOUT',9999,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$PWRD_TIMEOUT',0,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$SERVER_PORT',1110,STATUS) ;


SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STRT_PATH','192.168.123.24'
,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$PATH','192.168.123.24',STAT
US) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$remote','192.168.123.24',ST
ATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STRT_REMOTE','192.168.123.2
4',STATUS) ;


      --pokreni ponovo etag 7 kada si napravio promjenu i uspostavi
konekciju

     DELAY 10 ;
SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$OPER',3,STATUS);
      SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STATE',3,STATUS) ;
```

```
      DELAY 10 ;
      reconn_::

      DELAY 500;  CLOSE_FILE_(PRIMI,'C7:') ; DELAY 100;
      OPEN_FILE_(PRIMI,'C7:');

-----------------------ČISTI .txt FILE-----------------------

      OPEN FILE pod ('rw','POD.TXT')
      WRITE pod (CR)
      CLOSE FILE pod

-----------------------INICIJALIAZACIJA VARIJABLI-----------------
------

      I=0
      Msat=0

      OPEN FILE pod ('ap','POD.TXT')

-----------------------ZA ČIŠĆENJE BUFFERA-----------------------

      ovdje_::
      READ PRIMI(Msat2);
      BYTES_AHEAD(PRIMI,nb,STATUS)
      IF nb>0 THEN; GOTO ovdje_ ; ELSE GOTO ovdje2_
      ENDIF
      ovdje2_::

-----------------------SINKRONIZACIJA POČETKA--------------------
--

      DOUT [144]=TRUE
      WAIT FOR DIN [141]=TRUE
      WRITE('pocetak petlje',CR)

-----------------------PETLJA-----------------------
-----------------------------------------------------------------

WHILE I<999 DO --1 MANJE OD WHILE PETLJE U MASTERU

      I=I+1

      ------------ČITAJ SAT-----------------

      WRITE PRIMI (1,CR) ---ŠALJI SIGNAL MASTERU

      pon_::

      BYTES_AHEAD(PRIMI,nb,STATUS)
      IF STATUS<>0 THEN; GOTO reconn_;ENDIF

      IF nb=0 THEN; DELAY 1; GOTO pon_; --AKO BROJ BITOVA U BUFFERU
JE 0 ONDA DELAY 1 I PONOVO
      else
      READ PRIMI(Msat); --ČITAJ PODATAK OD MASTERA
```

```
      ENDIF

      WRITE pod(Msat,nb,CR); --ZAPIŠI PODATKE NA KARTICU
(FASTCLOCK,BROJ BITOVA, NOVI RED)

      IF I MOD 10=0 THEN; --AKO JE I DJELJIV SA 10 ZAPIŠI PODATKE NA
EKRANU
                  WRITE(I,Msat,nb,CR);
      ENDIF

      --DELAY 1

ENDWHILE

      CLOSE FILE pod --ZATVORI TXT DATOTEKU

      CLOSE_FILE_(PRIMI,'C7:') --ZATVORI VEZU

END brzSmjer
```

DODATAK B

```
PROGRAM brzMmjer
%NOLOCKGROUP
%NOPAUSE = ERROR + COMMAND + TPENABLE
VAR
ENTRY, STATUS,NB,iv,t0,t1,t,I,K,Msat,Msat2,rez,sat: INTEGER
SALJI, PRIMI, pod: FILE
rf:BOOLEAN
S,NIZ:STRING[100]
w,p,r,X,Y,Z,X_REG,Y_REG,Z_REG,W_REG,P_REG,R_REG,TPIN_REG,IPAD_ENB,po
m,vr,rez2,m,sig:REAL
KOORD: ARRAY[70] OF STRING[100]
SX,SY,SZ,SW,SP,SR,L_STR,S_STR,SLANJE:STRING[120]
P0,P1,P2,frm,frm2:XYZWPR
CONFIG_VAR:CONFIG

ROUTINE OPEN_FILE_ (FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
ROUTINE CLOSE_FILE_(FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2

BEGIN

WRITE('RONNA',CR)

SET_REAL_REG(15,0.1,STATUS)      --STATUS VEZE - U DRUGIM PROGRAMIMA
SE TEK POČINJE RADITI KADA
               --OVAJ STATUS BUDE 1.1 TJ. KADA SE USPOSTAVI VEZA SA
SERVEROM PC

     DOUT [141]=FALSE

--*********************************************************

       SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',0,STATUS) ;
       SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',0,STATUS) ;
DELAY 20

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$COMMENT','SOUND',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PROTOCOL','SM',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$REPERRS','FALSE',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$TIMEOUT',9999,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PWRD_TIMEOUT',0,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$SERVER_PORT',1110,STATUS) ;


SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_PATH','192.168.123.24'
,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_REMOTE','192.168.123.2
4',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PATH','192.168.123.24',STAT
US) ;
```

```
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$remote','192.168.123.24',ST
ATUS) ;

          --pokreni ponovo tag 8 kada si napravio promjenu i
uspostavi konekciju

      DELAY 10 ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',3,STATUS);
      SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',3,STATUS);
      reconn_::

      WRITE('tag 2',CR)

      DELAY 500;  CLOSE_FILE_(SALJI,'S8:') ; DELAY 100;
      OPEN_FILE_(SALJI,'S8:');

      WRITE('tag 3',CR)

      IF STATUS<>0 THEN; GOTO reconn_;ENDIF
      WRITE SALJI(123456789,CR) --ZA PROVJERU VEZE

------------------------INICIJALIAZACIJA VARIJABLI------------------
------

      I=0
      sig=0
      Msat=0

------------------------SINKRONIZACIJA POČETKA----------------------
--

      DOUT [141]=TRUE
      WAIT FOR DIN [144]=TRUE
      WRITE('pocetak petlje',CR)

------------------------PETLJA--------------------------------------
---------------------------------------------
-------------------------------------------------------------------
---------------------------------------------------

WHILE I<1000 DO

        I=I+1

        Msat=$FAST_CLOCK

      -------ZAPISUJ SAT-------

      ponovi_::

      --DELAY 1

      BYTES_AHEAD(SALJI,nb,STATUS)
      IF STATUS<>0 THEN; GOTO reconn_;ENDIF
```

```
      READ SALJI (sig); --ČEKAJ SIGNAL OD SLAVEA DA MU POŠALJEŠ
PODATAK

      IF sig<>1 THEN GOTO ponovi_; --AKO JE SIGNAL RAZLIČIT OD 1
PONOVO
      ELSE
      sig=0; WRITE SALJI (Msat,CR); --AKO JE SIGNAL 0 POŠALJI MU
PODATAK
      ENDIF

      WRITE (Msat,CR) --ZAPIŠI NA EKRAN TAJ PODATAK

ENDWHILE

      CLOSE_FILE_(SALJI,'S8:') --ZATVORI VEZU

END brzMmjer
```

DOCATAK C

```
PROGRAM brzSDig

VAR
ENTRY, STATUS, I,I1,tm,J: INTEGER
SALJI, PRIMI, pod: FILE

BEGIN

      DOUT [145]=FALSE

       -------ČISTI .txt FILE-------
      OPEN FILE pod ('rw','POD.TXT')
      WRITE pod (CR)
      CLOSE FILE pod

      -------OTVORI .txt FILE-------
      OPEN FILE pod ('ap','POD.TXT')


      I=0
      I1=0
      J=0
      tm=0

      DOUT [145]=TRUE
      WAIT FOR DIN [145]=TRUE
      DOUT [145]=FALSE

      WRITE ('petlja',CR)
------------------------PETLJA-------------------------------------
-------------------------------------------
-------------------------------------------------------------------
-------------------------------------------------

WHILE I<15 DO

      I1=I

      tu_::

      --PULSE DOUT[145] FOR 1 -- presporo

      DOUT [145]=TRUE

      DELAY 1

      I=GIN[1]

      DOUT [145]=FALSE

      IF I=I1 THEN

            GOTO tu_

      else
            tm=$FAST_CLOCK
```

```
            WRITE pod (I,tm,CR)
            WRITE(I,CR)
      ENDIF
ENDWHILE
CLOSE FILE pod
DOUT [145]=FALSE
tu2_::
WRITE ('stop',CR)
END brzSDig
```

DODATAK D

```
PROGRAM brzMDig

--%NOLOCKGROUP
--%NOPAUSE = ERROR + COMMAND + TPENABLE
VAR
ENTRY, STATUS, I,J: INTEGER
--I: INTEGER
--SALJI, PRIMI, pod: FILE
--rf:BOOLEAN
--S,NIZ:STRING[100]
--
w,p,r,X,Y,Z,X_REG,Y_REG,Z_REG,W_REG,P_REG,R_REG,TPIN_REG,IPAD_ENB,po
m,vr,rez2,m,sig:REAL
--KOORD: ARRAY[70] OF STRING[100]
--SX,SY,SZ,SW,SP,SR,L_STR,S_STR,SLANJE:STRING[120]
--P0,P1,P2,frm,frm2:XYZWPR
--CONFIG_VAR:CONFIG

BEGIN;

      DOUT [145]=FALSE
      GOUT[1]=0

      I=0
      J=0

      DOUT [145]=TRUE
      WAIT FOR DIN [145]=TRUE
      DOUT [145]=FALSE

------------------------PETLJA---------------------------------------
-----------------------------------------

WHILE J<15 DO

      J=J+1

      IF I=100 THEN

            GOTO tu_
      ENDIF

      IF J=15
            THEN J=0
            I=I+1
      ENDIF

      tu_::

      WAIT FOR DIN [145]=TRUE

      GOUT[1]=J

      WRITE (J,CR)

      DELAY 1
```

```
ENDWHILE

DOUT [145]=FALSE

END brzMDig
```

DODATAK E

```
PROGRAM brzM
%NOLOCKGROUP
%NOPAUSE = ERROR + COMMAND + TPENABLE
VAR
ENTRY, STATUS,NB,iv,t0,t1,t,I,K,Msat,Msat2,rez,sat,y_k: INTEGER
SALJI, PRIMI, pod: FILE
rf:BOOLEAN
S,NIZ:STRING[100]
w,p,r,X,Y,Z,X_REG,Y_REG,Z_REG,W_REG,P_REG,R_REG,TPIN_REG,IPAD_ENB,po
m,vr,kor,rez2,m,sig,krj,y_m:REAL
KOORD: ARRAY[70] OF STRING[100]
SX,SY,SZ,SW,SP,SR,L_STR,S_STR,SLANJE:STRING[120]
p0,P1,P2,frm,frm2,frm3,frm4:XYZWPR
CONFIG_VAR:CONFIG

ROUTINE OPEN_FILE_ (FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
ROUTINE CLOSE_FILE_(FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2

BEGIN
 CNV_STR_CONF('nut000', config_var, STATUS)
      X=0;Y=0;Z=0;W=0;P=0;R=0;

      --frm3=POS(500,-400,0,0,0,-90,config_var)--UF (500 mm radnog
kretanja)
      frm3=POS(0,-400,0,0,0,-90,config_var)--UF (1000 mm radnog
kretanja)

      frm4=POS(-8,1.340,369,-179.349,0.71,-3.159,config_var)--TF

      $MCR_GRP[1].$PRGOVERRIDE=100
      $GROUP[1].$UFRAME = frm3--$MNUFRAME[1,8];
      $GROUP[1].$UTOOL = frm4--$MNUTOOL[1,8];
      $GROUP[1].$MOTYPE=LINEAR;
      $GROUP[1].$SPEED=400; --brzina do pocetne tocke programa
      $GROUP[1].$TERMTYPE=FINE;

SET_REAL_REG(15,0.1,STATUS)      --STATUS VEZE - U DRUGIM PROGRAMIMA
SE TEK POČINJE RADITI KADA
               --OVAJ STATUS BUDE 1.1 TJ. KADA SE USPOSTAVI VEZA SA
SERVEROM PC

       DOUT [141]=FALSE


--*********************************************************

       SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',0,STATUS) ;
       SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',0,STATUS) ;
DELAY 20

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$COMMENT','SOUND',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PROTOCOL','SM',STATUS) ;
```

```
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$REPERRS','FALSE',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$TIMEOUT',9999,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PWRD_TIMEOUT',0,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$SERVER_PORT',1110,STATUS) ;


SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_PATH','192.168.123.24'
,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STRT_REMOTE','192.168.123.2
4',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$PATH','192.168.123.24',STAT
US) ;

SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$remote','192.168.123.24',ST
ATUS) ;

          --pokreni ponovo tag 8 kada si napravio promjenu i
uspostavi konekciju

      DELAY 10 ;
SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$OPER',3,STATUS);
      SET_VAR(entry,'*SYSTEM*','$HOSTS_CFG[8].$STATE',3,STATUS);
      reconn_::

      DELAY 500;  CLOSE_FILE_(SALJI,'S8:') ; DELAY 100;
      OPEN_FILE_(SALJI,'S8:');

      IF STATUS<>0 THEN; GOTO reconn_;ENDIF
      WRITE SALJI(123456789,CR) --ZA PROVJERU VEZE

      $MCR_GRP[1].$PRGOVERRIDE=100


      X=275;Y=0;Z=0;W=0;P=-45;R=0;

      P1= POS(x,y+150,z,w,p,r, config_var)
      P2= POS(x,y+1000.1,z,w,p,r, config_var)

      MOVE TO P1

      $GROUP[1].$SPEED=400 --daljnja maks. brzina u programu
      $MCR_GRP[1].$PRGOVERRIDE=12.5
      I=0
      sig=0
      krj=1

      DOUT [141]=TRUE
      WAIT FOR DIN [143]=TRUE
      WAIT FOR DIN [142]=TRUE
```

```
      WRITE('pocetak petlje',CR)

      MOVE TO p2 NOWAIT
      p0=CURPOS(0,0)

      kor=$GROUP[1].$SPEED*($MCR_GRP[1].$PRGOVERRIDE/100)*0.045

------------------------PETLJA---------------------------------------
-------------------------------------------
      WHILE ABS(p0.y)<=y+(1000) DO
      I=I+1
      Msat=$FAST_CLOCK;

      -------ZAPISUJ KOORDINATU-------ETHERNET
      ponovi2_::

      BYTES_AHEAD(SALJI,nb,STATUS)
      IF STATUS<>0 THEN; GOTO reconn_;ENDIF
      READ SALJI (sig,krj);
      IF krj<>1 THEN GOTO kraj_;
      else
      IF sig<>1 THEN GOTO ponovi2_;
      ELSE
      sig=0; p0=CURPOS(0,0); WRITE SALJI (p0.y+kor,Msat,1,CR); ---
WRITE('p0.y=',p0.y::3::2,nb,CR);
      ENDIF
      ENDIF

      ENDWHILE

      WRITE SALJI (p0.y,Msat,2,CR)

kraj_::

      CLOSE_FILE_(SALJI,'S8:')

END brzM
```

DODATAK F

```
PROGRAM brzS
%NOPAUSE = ERROR + COMMAND + TPENABLE
%NOLOCKGROUP

VAR
STATUS:INTEGER
prog_indx :INTEGER
 pod:FILE

rand,x,y,z,w,p,r,e,del,a,a_max,poc,q,g,l,h,v,v1,T,e1,e2,x_s,Pr,i,D,K
p,Ki,Kd,y_m,y_s,Msat,Msat2,diff,diff1,rez,krj,ve:REAL
 s1,s2,s3,s4,vp,vs,vps,vks,spoj,spoj2,pocs:REAL
 ENTRY,iv,c,nb,j:INTEGER
 SALJI, PRIMI: FILE
 rf:BOOLEAN
 p1,p2,p0,p3,frm, frm2,frm3,frm4 : XYZWPR
 config_var:CONFIG

ante :STRING [8]

-- s :STRING[100]

   ROUTINE OPEN_FILE_ (FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
   ROUTINE CLOSE_FILE_(FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2

BEGIN
      CNV_STR_CONF('nut000', config_var, STATUS)
      X=0;Y=0;Z=0;W=0;P=0;R=0;

      --frm3=POS(721.977,48.960,114.923,179.681,-
180.058,1.837,config_var)---UF (500 mm radnog kretanja) - novi !!!
      frm3=POS(721.977,548.960,114.923,179.681,-
180.058,1.837,config_var)---UF (1000 mm radnog kretanja) - novi !!!
      --SLAVE ROBOT MORA IMATI NEKE KUTOVE MEĐU OSIMA ZBOG
NESAVRŠENOSTI!!!

      frm4=POS(1.169,-0.6,337.851,0,0,0,config_var)--TF - velika
hvataljka
      --frm4=POS(0.475,-0.915,305.091,0,0,0,config_var)--TF - mala
hvataljka

WRITE('v1',CR)

      $MCR_GRP[1].$PRGOVERRIDE=100
      $GROUP[1].$UFRAME = frm3--$MNUFRAME[1,8];
      $GROUP[1].$UTOOL = frm4--$MNUTOOL[1,8];
      $GROUP[1].$MOTYPE=LINEAR;
      $GROUP[1].$SPEED=400; --brzina do pocetne tocke programa
      $GROUP[1].$TERMTYPE=NODECEL;

      DOUT [142]=FALSE

      X=380;Y=0;Z=0;W=180;P=0;R=183; --drugi Z za test spajanja

      p1= POS(x,y,z,w,p,r, config_var)
```

```
      $MCR_GRP[1].$PRGOVERRIDE=100
      MOVE TO P1 --kretanje u pocetnu toccku programa
      $GROUP[1].$SPEED=400 --daljnja maks. brzina u programu
      $MCR_GRP[1].$PRGOVERRIDE=100

      WRITE('pocetak petlje',CR)

      p0=CURPOS(0,0)

      RUN_TASK('REGULATOR',0,TRUE,TRUE,1,STATUS)--POKRENI PROGRAM
REGULATOR

      poc=0
      pocs=0
      krj=1
      j=0
      spoj=0
      spoj2=0
      vp=0
      vs=0
      vps=0
      vks=0
      g=1
      h=1

      pocs=1
      DOUT [142]=TRUE

      tu2_::
      GET_VAR(entry,'REGULATOR','poc',poc,STATUS)
      IF poc=1 THEN GOTO tu_ else GOTO tu2_
      ENDIF
      tu_::

      WAIT FOR DIN [141]=TRUE

      p2=CURPOS(0,0)

      WRITE('pocetak petlje slave',CR)

------------------------PETLJA-------------------------------------
-----------------------------------------------------------------
-----------------
-----------------------------------------------------------------
-----------------------------------------------------------------
----------------------
      s1=$FAST_CLOCK;

WHILE ABS(p2.y)<=Y+1000 DO

      GET_VAR(entry,'regulator','e',e,STATUS) --UZMI VARIJABLU e IZ
regulator.kl
      GET_VAR(entry,'regulator','krj',krj,STATUS) --UZMI VARIJABLU
krj IZ regulator.kl

      IF krj<>1 THEN GOTO kraj_; ENDIF
```

```
      WRITE(e,CR);

      p2.y=p2.y+10

      IF p2.x<=X-50 THEN spoj=1; spoj2=1;
      ENDIF

----------------------SPAJANJE-----------------------
-----------------------------------------------------------------

      IF spoj=1 THEN
            GOTO odvoji_
      else

      IF e<10 THEN
            WHILE j<1 DO s2=$FAST_CLOCK; j=j+1; ENDWHILE -- petlja za
zapisivanje kada je spajanje počelo
      IF p2.x>=X-50 THEN
            p2.x=p2.x-10
      ENDIF
      ENDIF
      ENDIF

      odvoji_::

----------------------ODVAJANJE----------------------
-----------------------------------------------------------------

      IF spoj2=1 THEN

      IF p2.x<=X-10 THEN
            p2.x=p2.x+10
      s3=$FAST_CLOCK;

      ENDIF
            else GOTO odvoji2_
      ENDIF

      odvoji2_::

-----------------------------------------------------
-----------------------------------------------------

      MOVE TO p2 NOWAIT

ENDWHILE

      s4=$FAST_CLOCK;

      vp=(2*(s4-s1))/1000
      vs=(2*(s3-s2))/1000
      vps=(2*(s2-s1))/1000
      vks=(2*(s4-s3))/1000
      WRITE('vp=',vp::3::2,CR,'vs=',vs::3::2,CR,'vps=',vps::3::2,CR,'
vks=',vks::3::2,CR)
```

```
  kraj_::
      CLOSE_FILE_(PRIMI,'C7:')
END brzS
```

DODATAK G

```
PROGRAM REGULATOR
%NOLOCKGROUP
%NOPAUSE = ERROR + COMMAND + TPENABLE
VAR
STATUS,register_no,int_value:INTEGER

 pod:FILE
 ds,no,real_value:REAL

rand,x,y,z,w,p,r,e,del,a,a_max,poc,q,g,l,h,v,v1,T,e1,e2,x_s,Pr,I,D,K
p,Ki,Kd,y_m,y_m1,y_s,Msat,Msat2,diff,rez,krj,ve,ru:REAL
 rk,pocs,kor,a_maxiso:REAL
 ENTRY,iv,c,nb,j:INTEGER
 SALJI, PRIMI: FILE
 rf,real_flag:BOOLEAN
 p1,p2,p0,p3,frm, frm2 : XYZWPR
 config_var:CONFIG
 s:STRING[100]
   ROUTINE OPEN_FILE_ (FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2
   ROUTINE CLOSE_FILE_(FILE_ : FILE; TAG_ : STRING) FROM LIB_FIL2

BEGIN


     DOUT [143]=FALSE
     poc=0

------------------------ZA SPAJANJE-----------------------
----------------------------------------------------------

--**********************************************************
       SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$OPER',0,STATUS) ;
       SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STATE',0,STATUS) ;
DELAY 20

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$COMMENT','SOUND',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$PROTOCOL','SM',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$REPERRS','FALSE',STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$TIMEOUT',9999,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$PWRD_TIMEOUT',0,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$SERVER_PORT',1110,STATUS) ;


SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STRT_PATH','192.168.123.24'
,STATUS) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$PATH','192.168.123.24',STAT
US) ;

SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$remote','192.168.123.24',ST
ATUS) ;
```

```
SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STRT_REMOTE','192.168.123.2
4',STATUS) ;

      --pokreni ponovo etag 7 kada si napravio promjenu i uspostavi
konekciju

      DELAY 10 ;
SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$OPER',3,STATUS);
      SET_VAR(entry,'*SYSTEM*','$HOSTC_CFG[7].$STATE',3,STATUS) ;
      DELAY 10 ;
      reconn_::

      DELAY 500;  CLOSE_FILE_(PRIMI,'C7:') ; DELAY 100;
      OPEN_FILE_(PRIMI,'C7:');

      -------ZA ČIŠĆENJE BUFFERA-------
      ovdje_::
      READ PRIMI(c);
      BYTES_AHEAD(PRIMI,nb,STATUS)
      IF nb>0 THEN; GOTO ovdje_ ; ELSE GOTO ovdje2_
      ENDIF
      ovdje2_::

------------------------------------------------------------
------------------------ZA SPAJANJE-------------------------

      -------ČISTI .txt FILE-------
      OPEN FILE pod ('rw','POD.TXT')
      WRITE pod (CR)
      CLOSE FILE pod

      -------OTVORI .txt FILE--------
      OPEN FILE pod ('ap','POD.TXT')

      -------INICIJALIZACIJA VARIJABLI-------

      --PI REGULATOR--50m/s - 500mm
      --Kp=0.4
      --Ki=0.09
      --Kd=0

      --PD REGULATOR--
      --Kp=0.9
      --Ki=0
      --Kd=0.05

      --PID REGULATOR--50m/s - 1000mm
      Kp=0.7
      Ki=0.09
      Kd=0.15

      --PID REGULATOR--150mm/s - 1000mm
      --Kp=3--1 3
      --Ki=0.13 --0.15
      --Kd=1     --0.8
```

```
      T=0.045

      a_maxiso=10000 --mm/s^2    30    200   400   10000
      a_max=a_maxiso/($GROUP[1].$SPEED/$MCR_GRP[1].$PRGOVERRIDE) --
jer nije definirana preko preko
                                                          --mm/s^2 već
kao % ukupne


      q=0
      e=150
      e1=150
      e2=0
      --v=0.001
      v1=0
      j=0
      Msat=0

      Msat2=0
      y_m=150 --POČETNA GREŠKA - RAZMAK IZMEĐU SLAVEA I MASTERA
      poc=0
      krj=1
      ru=0

      poc=1
      DOUT [143]=TRUE

      tu2_::
      GET_VAR(entry,'BrzS','pocs',pocs,STATUS)
      IF pocs=1 THEN GOTO tu_ else GOTO tu2_
      ENDIF
      tu_::

      WAIT FOR DIN [141]=TRUE

------------------------PETLJA-------------------------------------
-------------------------------------
-------------------------------------------------------------------
---------------------------------------------

      p0=CURPOS(0,0)

WHILE p0.y<1000 DO

      j=j+1

  -------ČITAJ KOORDINATE-------ETHERNET
      WRITE PRIMI (1,1,CR)

      pon_::

      BYTES_AHEAD(PRIMI,nb,STATUS) --čitaj input buffer
      IF STATUS<>0 THEN; GOTO reconn_;ENDIF
```

```
      IF nb=0 THEN DELAY 1; --ako je broj bitova u input bufferu = 0
delay 1ms
            GOTO pon_;
      ELSE
            READ PRIMI(y_m,Msat,krj);
            IF krj<>1 THEN
                  GOTO kraj_; --za gašenje ak je M brži
            ENDIF
      ENDIF

      p0=CURPOS(0,0)

       e=y_m-p0.y;

      -------REGULATOR-------

      q=q+e --za zbrajanje greške

      Pr=Kp*e
      I=Ki*T*q
      D=Kd*((e-e1)/T)

      v=Pr+I+D

      --Pr=(Kp+(Ki*T/2)+(Kd/T));
      --I=(-Kp+(Ki*T/2)-((2*Kd)/T));
      --D=Kd/T;
      --v=v1+Pr*e+I*e1+D*e2;

      --e2=e1

      e1=e

      -------OGRANIČENJE BRZINE I UBRZANJA-------

      a=((v-v1)/T)

      IF v>=100 THEN v=100
      ENDIF
      IF v<=0 THEN v=0.001
      ENDIF
      IF (v-v1)/T>a_max THEN v=a_max*T+v1
      ENDIF
      IF (v1-v)/T>a_max THEN v=-a_max*T+v1
      ENDIF

      rez=(2*(Msat-Msat2))/1000

      WRITE pod (v::5::5,e::5::5,y_m,p0.y::5::5,p0.x,Msat,nb,CR)

      DELAY 1

      v1=v

      $MCR_GRP[1].$PRGOVERRIDE=v
```

```
ENDWHILE

      WRITE PRIMI (1,2,CR)
      krj=2;
      kraj_::

      CLOSE FILE pod

      CLOSE_FILE_(PRIMI,'C7:')

END REGULATOR
```

DODATAK H

```
 1:  R[106]=106    ;
 2:  LBL[2] ;
 3:  R[199]=R[106]    ;
 4:  R[106]=R[106]+1    ;
 5:  LBL[4] ;
 6:  VISION RUN_FIND 'ANTE'    ;
 7:  VISION GET_OFFSET 'ANTE' VR[1] JMP LBL[1] ;
 8:  LBL[1] ;
 9:  R[R[106]]=VR[1].MES[1] ;
10:  R[198]=R[R[106]]-R[R[199]]    ;
11:  IF R[198]>0,JMP LBL[5] ;
12:  R[198]=R[198]*(-1)    ;
13:  LBL[5] ;
14:  IF R[198]<1,JMP LBL[4] ;
15:  IF R[106]=198,JMP LBL[3] ;
16:  JMP LBL[2] ;
17:  LBL[3] ;
```