Programska aplikacija za decentralizirano upravljanje mobilnim robotima u formaciji

Dobrilović, Domagoj

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Domagoj Dobrilović

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Izv. prof. dr. sc. Andrej Jokić, dipl. ing. Domagoj Dobrilović

Zagreb, 2017.

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

Zahvaljujem mentoru izv. prof. dr. sc. Andreju Jokiću na stručnom vođenju, korisnim savjetima tijekom izrade rada.

Također zahvaljujem prof. dr. sc. Mladenu Crnekoviću na ukazanoj pomoći i savjetima tijekom izrade programske aplikacije, kao i asistentu Vladimiru Miliću na savjetima tijekom rada u laboratoriju.

Domagoj Dobrilović

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **Domagoj Dobrilović** Mat. br.: 0035183830

Naslov rada na hrvatskom jeziku: **Programska aplikacija za decentralizirano upravljanje mobilnim robotima u formaciji**

Naslov rada na engleskom jeziku: **Software Application for Decentralised Control of Mobile Robots in Formation**

Opis zadatka:

Na Katedri za strojarsku automatiku razvijeni su i proizvedeni edukativni mobilni roboti (3 robota). U laboratoriju Katedre nalazi se poligon za rad s ovim robotima. U ovom radu potrebno je osmisliti, razraditi i razviti modularne programske aplikacije za decentralizirano/distribuirano upravljanje mobilnih robota prilikom gibanja u formaciji. Postojeća oprema (senzori) omogućavaju vožnju u liniji, te će rad biti uže usmjeren na ovu konkretnu formaciju, dok je u samom radu potrebno prikladno se osvrnuti i na generalizaciju rezultata na općenitije formacije i veći (proizvoljan) broj mobilnih robota.

U radu je potrebno:

1) Iz pregleda literature sažeti osnovne pristupe upravljanju mobilnih robota u formaciji.
2) Razviti modularnu aplikaciju za upravljanje mobilnog robota u formaciji, u kojoj je korisniku na lagano dostupan način omogućena implementacija različitih upravljačkih algoritama.
3) Za vožnju u liniji, potrebno je sintetizirati i primijeniti u laboratoriju algoritme upravljanja i to: decentraliziran, distribuiran i centraliziran. Kroz niz eksperimenata s tri mobilna robota usporediti učinkovitost ovih algoritama.

Rad predati u pisanom i elektroničkom obliku, uz prikladne upute za korištenje razvijenih aplikacija.

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

Zadatak zadan: 11. svibnja 2017.

Rok predaje rada: 13. srpnja 2017.

Predviđeni datum obrane: 19., 20. i 21. srpnja 2017.

Zadatak zadao: Izv. prof. dr. sc. Andrej Jokić

Predsjednica Povjerenstva: Prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x_n	-	Pozicija vozila
u_n	%	Upravljana ulazna vrijednost mobilnog robota
d_n	-	Poremećaj koji djeluje na mobilnog robota
N	-	Broj članova formacije
v	m/s	Translacijska brzina robota
ω	°/s	Kutna brzina robota
r	m	Polumjer kotača robota
L	m	Razmak između kotača robota
ω_R	°/s	Kutna brzina desnog kotača
ω_L	°/s	Kutna brzina lijevog kotača
p_n	-	Pozicija mobilnog robota
K_p	-	Proporcionalno pojačanje
K_d	-	Derivacijsko pojačanje
f_n	-	Unaprijedno pojačanje pogreške razmaka
b_n	-	Unazadno pojačanje pogreške razmaka
k_n	-	Unazadno pojačanje razlike brzina
g_n	-	Pojačanje pogreške brzina
v_n	%	Brzina robota izražena u postocima
v_n^e	m/s	Pogreška brzine
δ_{ref}	m	Referentna udaljenost između robota
δ_x	m	Referentna udaljenost između prvog i drugog robota
δ_y	m	Referentna udaljenost između drugog i trećeg robota
$\delta_{1\text{-}2}$	m	Udaljenost izmjerena prednjim senzorom
$\delta_{2\text{-}3}$	m	Udaljenost izmjerena stražnjim senzorom

$\delta^{e}_{n-(n-1)}$	m	Relativna pogreška razmaka između robota
e(t)	m/s	Razlika željene i stvarne brzine robota

SAŽETAK

U ovome je radu prikazano i opisano decentralizirano upravljanje eMIR mobilnim robotima u linearnoj formaciji. Razvijena je programska aplikacija koja omogućuje upravljanje linearne formacije 3 mobilna robota, uz mogučnost primjene proizvoljnog upravljačkog algoritma. Razvijene su i aplikacije za svakog robota zasebno, s kojima se isključivo može postići decentralizirano upravljanje. Aplikacije su napisane pomoću programskog jezika Delphi, te je u radu dan kratak opis sučelja te upute za korištenje. U pregledu literature navedene su osnovne komponente i principi upravljanja linearnom formacijom mobilnih robota. Objašnjeni su, skicirani i implementirani u aplikaciju različiti algoritmi koji omogućuju upravljanje formacijom. Primjenom upravljačkih algoritama koji se aktiviraju unutar aplikacije, dobivene su informacije o gibanju svakog robota u formaciji, te su ti rezultati opisani i prikazani grafovima u zadnjem poglavlju rada.

Ključne riječi: eMIR, decentralizirano upravljanje, programiranje aplikacije, Delphi, formacija mobilnih robota.

SUMMARY

This thesis describes the experimental results of decentralized and distributed control of linear formantion of „eMIR“- eduational mobile robots. Software application is designed to enable implementation of different algorithms to control formation of 3 mobile robots. Additionally, several applications developed for controlling each robot seperately. Applications are coded using Delphi programming language. Short instructions and description of interfaces for using software are presented. In literature overview, major components and principles of the mobile robots linear platoon control are given. Several algorithms for formation control are explained, sketched and implemented on real-life platform. Results of the performed experiments are shown in graphs and described in the last chapter of the thesis.

Key words: eMIR, Decentralized control, Application programming, Delphi, formation of mobile robots

1 UVOD

Od samog konstruiranja prvih mobilnih robota promatraju se razni načini kako postaviti interakciju između robota i na koji način upravljati formacijom mobilnih robota. Razvojem metoda komunikacije i sustava za prepoznavanje okruženja robota sve se više promatraju različiti načini i primjene upravljanja robotskim formacijama. Zanimljiva su gibanja različitih formacija autonomnih mobilnih robota, kao primjerice automatizirani pogoni i skladišta koja koriste mobilne robote. Također su zanimljive i linearne formacije vozila ili autonomne kolone kamiona i automobila na autocestama, jer se takvom vožnjom u formaciji povećava protok vozila, sigurnost, te se smanjuje potrošnja goriva.

U sklopu ovoga rada, razvijena je aplikacija u kojoj su sintetizirani različiti algoritmi koji omogućuju upravljanje formacijom mobilnih robota. Aplikacijom se upravlja edukativnim mobilni robotima koji su proizvedeni i razvijeni na Katedri za strojarsku automatiku, Fakulteta strojarstva i brodogradnje i posjeduju opremu koja omogućuje vožnju robota u linearnoj formaciji. Kroz rad su navedene, ispitane i prikazane učinkovitosti različitih implementiranih algoritama, koji su lako dostupni korisniku kojemu se klikom omogućuje lagan odabir upravljanja unutar aplikacije.

2 PREGLED LITERATURE

U ovome radu promatra se formacija više vozila, koja se gibaju s relativno malim razmacima između njih, poput gibanja automobila u koloni. U gibanju u formaciji bitno je održavanje željenog razmaka između vozila, i to se rješava odgovarajućom (automatskom) kontrolom brzine vozila.

Ovakav način gibanja vozila predmet je istraživanja u mnogim radovima. Primjerice, vožnja u formacijama mogla bi povećati protok transportnih sustava, te znatno povećati sigurnost putovanja. Također protok i kapacitet sustava vozila može bitno ovisiti o načinu na kojim se održava željeni razmak između vozila unutar formacija.

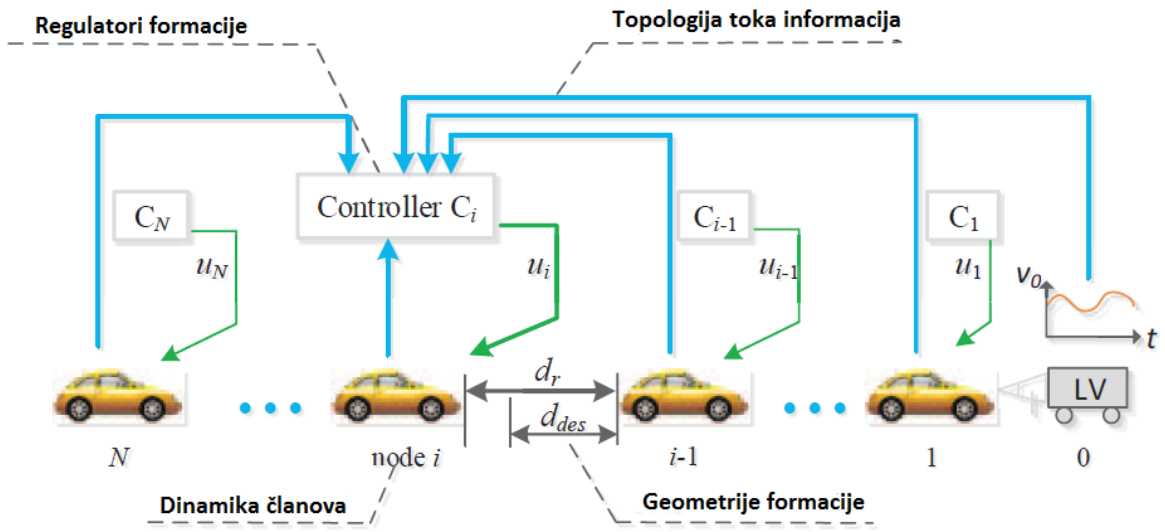
Zbog ekoloških i ekonomskih razloga u novije vrijeme istražuju se mogućnosti primjene različitih algoritama i načina vođenja automobila i kamiona u formaciji. Tako se na primjer može primijeniti vožnja kamiona u formaciji, u kojoj se pri manjim udaljenostima između vozila može znatno smanjiti otpor zraka, koji bitno utječe na gibanje vozila. Otpor zraka se znatno povećava pri visokim brzinama, te time u konačnici uzrokuje povećanje potrošnje goriva, prema tome vožnja u formaciji bi bila efektivnija pri komercijalnom prijevozu na dužim i brzim cestama. U nekim istraživanjima poput [1][2][3] autori se bave problemima vezanim uz smanjenje razmaka između vozila, odnosno smanjenja djelovanja otpora zraka na vozila u formaciji iza vodećeg vozila. Također u radovima se primjerice promatra i stabilnost niza vozila s uključenim ACC-om („adaptive cruise control“), sustavom koji prilagođava brzinu prilikom korištenja tempomata u vožnji. Razvojem raznih tehnologija u današnje vrijeme, prije svega razvojem komunikacije vozila s vozilom, te razvojem tehnologija za raspoznavanje okoline vozila, za očekivati je da će se vožnja kamiona u formacijama kao i autonomna vožnja automobila, u komercijalne svrhe ostvariti već u skorijoj budućnosti, a mnogo primjera ima već i danas.

Slika 1. Vožnja kamiona u formaciji [2]

U radovima poput [4], [5], [6] autori se fokusiraju na kontrolu i upravljanje formacije s obzirom na međudjelovanje vozila sa susjednim, odnosno najbližim vozilima, kao i kontrolu nad velikim brojem vozila u formaciji.

Prema [7], linearna formacija vozila, odnosno kolona vozila, može se prikazati kao kombinacija četiriju glavnih komponenti, prema kojima se mogu razvrstati radovi vezani uz ovu tematiku:

- dinamike članova– koja predstavlja ponašanje svakog vozila,
- topologije toka informacija – koja definira kako čvorovi izmjenjuju informacije,
- regulatori formacije – povratna veza provodi se određenim automatskim regulatorom i u pravilu se koriste distribuirani regulatori koji koriste informacije sa susjednim vozilima,
- geometrije formacije – kojom se određuju željeni razmaci između vozila unutar formacije.



Slika 2. Osnovne komponente vozila u formaciji: dinamika članova; topologija toka informacija; regulator; geometrija formacije [20]

Slika 2. prikazuje četiri glavne cjeline prema kojima se može razvrstati linearna formacija. Linearna formacija vozila se sastoji od vodećeg vozila (LV) označenog s bojem 0 i pratećih vozila koja su indeksirana od 1 do N. Vodeće vozilo, kao i ono zadnje u nizu (indeksirano s N+1) mogu biti fiktivni ili realni članovi.

2.1 Dinamika članova

Kada razmatramo dinamiku članova važno je naglasiti pojmove homogenosti članova unutar formacije. Kada svi članovi formacije imaju istu dinamiku, kaže se da je ta formacija homogena. U suprotnom kažemo da je nehomogena. Mnogi radovi se ograničavaju na homogene formacije, dok radovi koji proučavaju nehomogene sustave u pravilu više odgovaraju realnim slučajevima. Primjerice u radu [8] proučava se stabilnost heterogenog niza vozila, gdje se pogreške pozicija u formaciji ne razdjeljuju ravnomjerno duž niza, kao što je slučaj kod homogenog niza vozila. U radu su prikazane i analizirane stabilnost i propagacija pogreška razmaka koje se javljaju u formaciji proizvoljne duljine i proizvoljnog smještaja vozila različite dinamike. Također [9] je primjer rada u kojem se ispituje utjecaj heterogenosti i ne simetričnosti na granicu stabilnosti.

Kako se linearna formacija odnosi samo na jedan smjer kretanja, odnosno jednu dimenziju, tako se u pravilu upravljanje takvom formacijom, kao i sama dinamika određenih čvorova odnosi na uzdužan smjer kretanja.

U stvarnosti linearna dinamika članova vozila u formaciji je inherentno nelinearna. Ta je dinamika sastavljena od motora s unutarnjim izgaranjem, prijenosa snage, sistema kočenja, otpora zraka, trenja između podloge i kotača, gravitacijskog djelovanja, otpora kotrljanja i drugih. Neka istraživanja direktno koriste nelinearne modele linearne formacije vozila, ali njihova analiza učinkovitosti je znatno otežana danim topologijama toka informacija i razmacima u formaciji.[7]

Često korišteni modeli, su modeli koji se sastoje od :

- jednostrukog integratora,
- dvostrukog integratora,
- modela trećeg reda,

Model jednostrukog integratora je najjednostavniji slučaj, gdje upravljana veličina direktno utječe na brzinu mobilnog robota. Korištenjem ovog principa znatno se pojednostavljuje teoretska analiza, kao i sinteza regulatora. S druge strane, ovakav model odstupa od stvarne dinamike vozila i njime se ne može prikazati nestabilnost niza vozila, kao i efekt harmonike u formaciji [7]. Model jednostrukog integratora za *N* vozila dan je sljedećom jednadžbom:

$$\dot{x}_n = u_n + d_n, \quad n = 1, ..., \mathrm{N}, \tag{1}$$

gdje su u_i i d_i upravljani ulaz i poremećaj koji djeluju na *i*-to vozilo.

Bolji slučaj od navedenog modela je kada koristimo model s dvostrukim integratorom, odnosno model gdje upravljana veličina utječe na akceleraciju vozila:

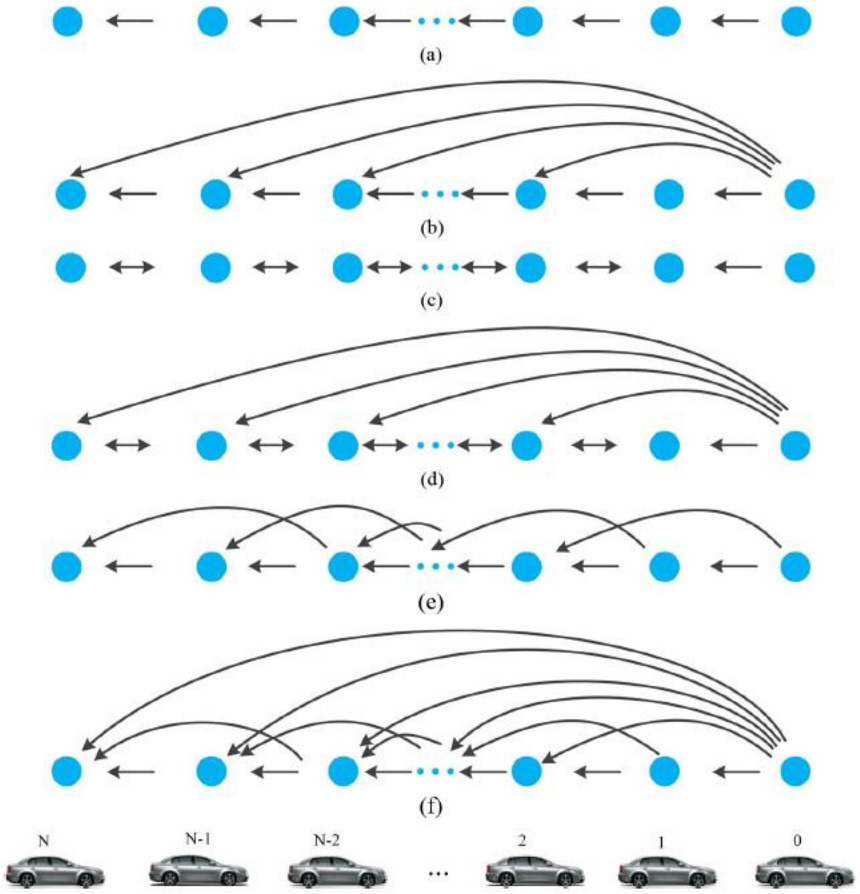
$$\ddot{x}_n = u_n + d_n, \quad n = 1, ..., \mathrm{N}, \tag{2}$$

Uz korištenje ovog modela dobiveni su rezultati za optimalnu kontrolu formacije [4], koherencije formacije vozila [10] ili računanja granice stabilnosti [9].

Gore navedeni modeli i dalje ne opisuju mnoga svojstva dinamike vozila kao kašnjenja zbog inercije motora vozila, utjecaja otpora zraka, utjecaj trenja i okretnog momenta kočenja, omjere prijenosa brzina, mase vozila i dr. Zato se u model uvodi još jedno stanje kako bi se bolje prikazalo ponašanje vozila. Time se dobivaju sustavi trećeg reda. Većina aproksimacija koriste tehniku linearizacija povratne veze ili tehniku kontrole donjeg sloja.[7] U ovakvom pristupu u pravilu se i dalje koristi niz pretpostavki, npr., vozilo se sastoji od krutih tijela, proklizavanja pneumatika je zanemarivo, konstantne temperature motora i sl.

2.2 Topologija toka informacija

Topologija toka informacija (IFT-„Information Flow Topology“) unutar linearne formacije vezana je uz način na koji svako vozilo dobiva informacije od vozila koje ga okružuju. ITF prikazuje kako informacija koju prima svaki lokalni regulator, utječe na ponašanje linearne formacije.



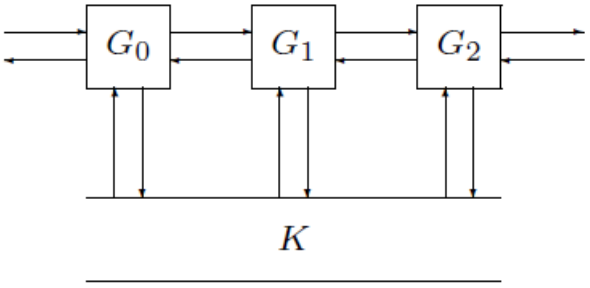
Slika 3. Topologija toka informacija [11]

Na slici 3. prikazani su neki primjeri ITF-a, poput formacija gdje vozila dobivaju informaciju od prethodnika (a) i dvosmjerno, informacijama od vozila ispred i straga (c). Takve se topologije formiranju korištenjem sustava senzora na samim vozilima. Razvojem tehnologija komunikacije između vozila moguće je dobiti informacije od vodećeg vozila u koloni, kao i ostalih članova formacije, a da nisu samo susjedna vozila tj. mobilni roboti. Tako postoje topologije koje koriste informacije od vodećeg vozila i prethodnika (b) ili od vodećeg vozila, prethodnika i sljedbenika (d). Unutar formacije mogu postojati veze s predvodećim vozilom i onim ispred njega (e), a ukoliko se nadoda veza i sa vodećim vozilom dobiva se topologija

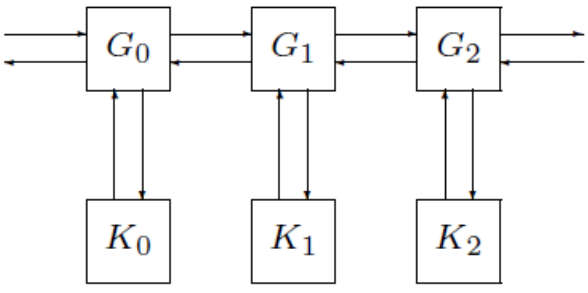
(f). Ovisno o mogućnostima članova formacije, mogu se formirati i drugačije topologije toka informacija.

2.3 Regulatori formacije

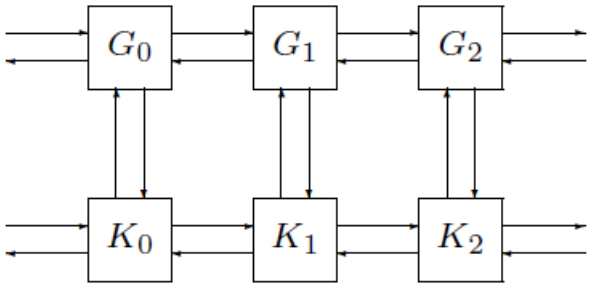
Upravljanje robotskim formacijama vrši se nekom vrstom automatskog regulatora. Na slikama dolje u tekstu, su prikazane različite upravljačke strukture. Centraliziranim regulatorom (slika 4.) koji prikuplja mjerenja od svih članova formacije, postiže se najbolje upravljanje. Međutim primjena centraliziranog regulatora je u praksi otežana radi potrebe intenzivne komunikacije, te otežanog rada s velikim brojem članova. Kada se svakim članom upravlja s lokalnim, vlastitim regulatorom, bez ikakvih dijeljenja informacija sa susjednim regulatorima, kažemo da je takvo upravljanje decentralizirano. Decentralizirano upravljanje koje je shematski prikazano na slici 5., je upravo suprotno od centraliziranog, pa tako i postiže najlošije rezultate. Distribuiranim upravljanjem koje je shematski prikazano na slici 6. linearna formacija se dijeli na niz podsistema koji su upravljani lokalnim regulatorima koji uz lokalna mjerenja surađuju i s određenim brojem drugih (obično susjednih) lokalnih regulatora. Ovakvim se pristupom mogu konstruirati različite vrste distribuiranog upravljanja, ovisno o vezama koje pojedini članovi formacije mogu uspostaviti.



Slika 4. Centralizirano upravljanje[12]



Slika 5. Potpuno decentralizirano upravljanje [12]



Slika 6. Distribuirano upravljanje [12]

Za linearnu formaciju, bitno je da je unutarnja stabilnost formacije osigurana, a to se može postići na par načina. Tako se za unutarnju stabilnost može koristiti globalni kao u [5] ili lokalni pristup. Kao globalnog pristupa sintetizira se centralizirani regulator za formaciju kao cjelokupni strukturirani sistem, na čiju sintezu topologija toka informacija nema previše utjecaja.

U pravilu se u radovima na temu linearne formacije vozila, odnosno u primjeni i teorijskoj analizi, koriste distribuirani načini upravljanja. Način, kao i poteškoće pri sintezi takvih regulatora postavlja topologija toka informacija, a samim time i učinkovitost gibanja vozila u formaciji. Tako način na koji je sintetiziran regulator ovisi od slučaja do slučaja.

U mnogim radovima poput [4,][13], predstavljaju se metode optimizacije, numerički ili analitički, te su također predložena neka optimizirana rješenja za pojačanja lokaliziranog sustava.

Neka istraživanja koriste klizni način upravljanja SMC („Sliding model control“) kako bi konstruirali stabilnu kolonu vozila. Primjer je u radu [14] gdje se koristi klizni način upravljanja za nelinarni regulator za adaptivni tempomat vozila (ACC) i [15] gdje se koristi za dobivanje stabilnosti homogene ili heterogene linearne formacije.

U novije vrijeme, neki radovi istražuju modele za predviđanje dinamike u formaciji vozila (MPC metode –„Model Predictive Control“). U radu [16] korišten je MPC u kooperativne svrhe upravljanja tempomata vozila. Uz njih za poboljšavanje stabilnosti niza vozila koriste se i $H\infty$ metode upravljanja.[16]

2.4 Geometrije formacije

Formacija vozila može biti ostvarena tako da se upravlja:

- konstantnom udaljenošću,
- konstantnim vremenom održavanja razmaka,
- nelinearnom udaljenošću,

Kada se formacija osniva na održavanju konstantne udaljenosti, željena udaljenost između dvaju uzastopnih vozila, nezavisna je s obzirom na brzinu gibanja vozila, te se time može postići visoki prometni kapacitet. Pristupom konstantnog vremena održavanja razmaka, željeni odnos razmaka vozila mijenja se s brzinom vozila. Taj princip je više u skladu s ponašanjem vozača na cesti, ali ograničava prometni kapacitet koji bi se mogao ostvariti. Kod nelinearne udaljenosti, razmak između vozila je nelinearna funkcija brzine vozila koja ima potencijal za poboljšavanje stabilnosti toka prometa, kao i kapaciteta prometa, u odnosu na druga dva pristupa.[4] Više o takvom principu može se naći u [14] i [18] gdje se promatraju pristupi razmaka vozila s ACC-om s ciljem poboljšavanja stabilnosti toka prometa i stabilnosti niza vozila. U ovom radu, odnosno aplikaciji, koristi se pristup održavanja konstantne udaljenosti unutar linearne formacije.

2.5 Učinkovitost linearne formacije

U ovom dijelu rada će biti dan pregled najznačajnijih mjera učinkovitosti i stabilnosti linearne formacije vozila. Tako se formacija može promatrati u okviru stabilnosti niza vozila, granice stabilnosti i povezanosti, odnosno koherentnosti formacije.

2.5.1 Stabilnost niza vozila

U nizu vozila, pogreške u željenim relacijama između vozila se mogu pojačati kada propagiraju dalje kroz niz vozila, što u konačnici može izazvati nestabilnost pa i sudar vozila. Time nam je u cilju dizajnirati dobro upravljanje za danu geometrijsku formaciju i topologiju toka informacija kako bi dobili stabilan niz vozila.

Tako se primjerice u radu [19] pokazalo da nestabilnosti niza vozila s politikom konstantne udaljenosti rastu ukoliko vozilo dobiva informacije smo od prethodnika. Ukoliko se informacije dobivaju i od vodećeg vozila, problem je skalibilan jer su pojačanja grešaka poremećaja ravnomjerno vezane uz povećanje broja vozila. Također, rezultati su pokazali da

upravljanje regulatorom samo s vezama prethodnika i sljedbenika pati od temeljnih ograničenja izvedbe zatvorene petlje koja se ne može ublažiti odgovarajućim regulatorom. [20]

Neka od u literaturi predloženih rješenja za poboljšavanje stabilnosti niza vozila su:

- relaksiranjem krutosti formacije,
- korištenjem drugačijih kontrolera za različita vozila,
- proširivanjem topologije informacija,

Relaksiranje krutosti formacije može se vršiti uvođenjem dovoljnog vremena za održavanje razmaka ili nelinearnom politikom udaljenosti. Stabilnost se može poboljšati i podešavanjem pojačanja kontrolera u formaciji vozila [6].

U novijim radovima korištenjem kliznog načina upravljanja, $H\infty$ metodama, kao i prediktivnim metodama upravljanja, osigurava se stabilnost niza vozila, pri tome da svi oni koriste održavanje geometrije formacije s konstantnim vremenom održavanja razmaka ili koriste neku globalnu informaciju. [14] [15] [16]

2.5.2 Granica stabilnosti

Granica stabilnosti se koristi za opisivanje konvergencije brzine vozila unutar kolone vozila. Mnoga istraživanja proučavaju granicu stabilnosti na primjeru formacija s konstantnom udaljenošću između vozila, gdje je ona funkcija broja vozila u koloni, dinamike članova, informacije toka informacija i strukture kontrolera.[7]

U radovima [6] i [11] prikazano je kako se granica stabilnosti približava nuli u simetričnoj dvosmjernoj kontroli (s prethodnikom i sljedbenikom) kao $O(1/N^2)$ s velikim brojem vozila N. Ovo bi se asimptotsko ponašanje granice stabilnosti moglo poboljšati manjim podešavanjem, odnosno asimetrijom do $O(1/N)$, te time sustav postaje robustniji na manje greške prilikom povećanja broja vozila u koloni.

Korištenjem neusmjerene topologije u formaciji vozila granica stabilnosti ne konvergira k nuli povećavanjem broja vozila, ukoliko je veliki broj sljedbenih vozila vezan za informacije vodećeg. Time se proširivanjem topologije toka informacija formacija postiže skalibilnost. Također, još jedan način za poboljšavanje granice stabilnosti je korištenje asimetričnih kontrolera.[21]

2.5.3 Koherentnost formacije

Koherentnost formacije se može promatrati kao mjera koja kvantificira koliko se određena formacija ponaša blisko ponašanju krute rešetke ili krutog objekta. Koherentnost formacije je pojam različit od nestabilnosti niza mobilnih robota i često je nepovezan s njime. Problem koherentnosti formacije je najznačajniji upravo u slučaju linearne formacije vozila, odnosno jednodimenzionalne formacije.

Lokalna povratna veza može kao posljedicu imati stabilnost niza vozila, u smislu da učinci vanjskog poremećaja ne propagiraju s prostornim rastom formacije. Tako lokalno upravljanje jednodimenzionalnim nizom vozila može davati vrlo dobre rezultate na „mikroskopskoj" razini, jer će poremećaji između vozila biti dobro regulirani. Ali kada se u cijelosti promatra velika formacija vozila, pojavit će se neregulirani duljinski poremećaj- dugi prostorni valovi-nalik na „harmoniku" pokreta. Kako je formacija mikroskopski (lokalno) dobro regulirana, neće doći do sudara, ali će utjecati na učinak propusnosti vozila, jer ono ovisi o koherentnosti ili krutosti formacije. [10]

U [10] istražene su gornje asimptotske granice konerencije, s obzirom na broj vozila N u formaciji, te je naznačeno da topologija toka informacija ima značajniju ulogu naspram regulatora. U tablici 1. Su prikazane makroskopske učinkovitosti s obzirom na veličinu mreže M i dimenzije prostora d. Zapis se može izraziti kao i $M=N^d$. [10]

Mjere učinkovitosti koje ulaze u mjeru koherentnosti formacije su: lokalna greška, odnosno razlike između susjednih članova ili mobilnih robota; odstupanje od dugog raspona, koje daje odnos neslaganja između dva najudaljenija člana formacije; odstupanje od prosjeka, je mjera devijacije svakog člana od prosjeka svih.

Tablica 1. Gornje granične asimptotske vrijednosti formacije s obzirom na veličinu formacije *M* i prostorne dimenzije *d*

Makroskopske učinkovitosti	Prostorna dimenzija
M	*d=1*
log(M)	*d=2*
1	*d=3*

Iz navedenog se može zaključiti da kod jednodimenzionalne i dvodimenzionalne formacije lokalno upravljanje ne može regulirati poremećaje u velikim razmjerima (globalno), dok kod viših dimenzija to nije slučaj.

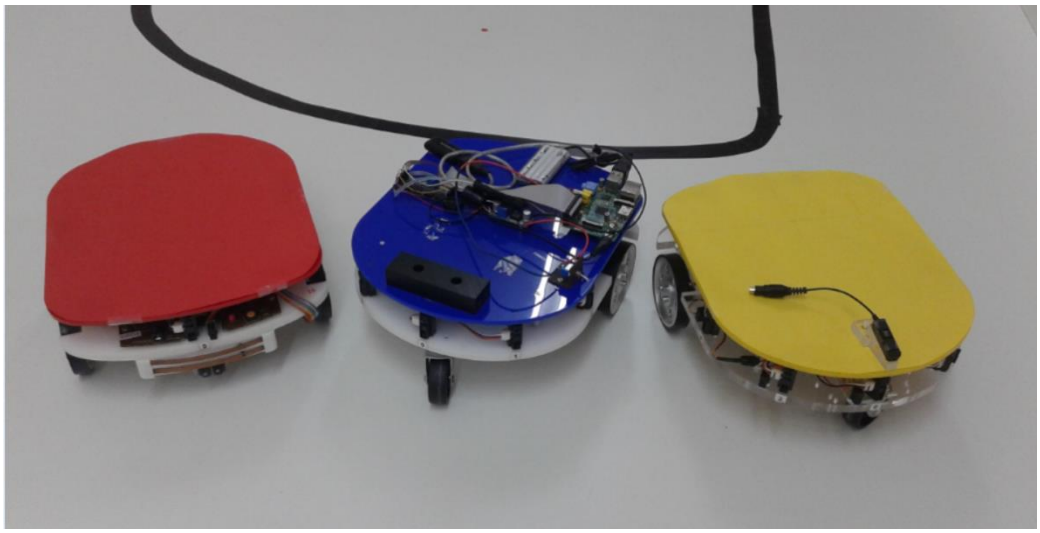
Radi poboljšavanja koherentnost formacije vozila, u radu [4] izvodi se optimalno lokalizirano upravljanje. Tako su u tablici 2. Prikazane vrijednosti za globalnu koherentnost formacije s obzirom da li je regulator optimalan, te da li je simetričan ili ne simetričan. Crvene vrijednosti su dobivene analitički a crne numerički i one vrijede za model s jednostrukim i dvostrukim integratorom.[4] U radu je prikazano da se odmicanjem od simetrije mogu postići bolji rezultati, te da najbolje rezultate daje optimalni lokalni regulator koji je ujedno ne simetričan i prostorno varirajući. Princip kojim se promatra ne-simetrično pojačanje, je princip gledanja unaprijed ($f_n = \alpha > 0,\ b_n = 0$), te je time i vidljivo da topologija toka informacija bitno utječe na koherenciju formacije. Formaciji se može dodati fiktivni pratitelj koji ima pristup svojoj globalnoj poziciji.

Tablica 2. Asimptotska ovisnost globalne učinkovitosti koherencije formacije Π_g o veličini formacije *N*, (ne)simetričnosti i optimalnom upravljanju.[4]

Regulator	Π_g
Jednoličan simetričan sa ili bez pratitelja	$O\left(\sqrt{N}\right)$
Jednoličan ne –nesimetričan	$O\left(\sqrt{N}\right)$
Optimalan simetričan bez pratitelja	$O\left(\sqrt{N}\right)$
Optimalan simetričan s pratiteljem	$O\left(\sqrt{N}\right)$
Optimalan ne-simetričan sa ili bez pratitelja	$O\left(\sqrt[4]{N}\right)$

3 OPIS ROBOTA

Na katedri za strojarsku automatiku Fakulteta strojarstva i brodogradnje konstruirana su i izgrađena 3 eMIR robota. Skraćenica eMIR označava "educational Mobile Intelligent Researcher". Roboti su napravljeni od PMMA (polimetilmetakrilat) ploča, gdje je donja ploča debljine 10 mm, a gornja debljine 6 mm koje zaštićuju ključne dijelove robota poput mikrokontrolera i senzora. Vanjske dimenzije robota su: duljina 300 mm, širina 250 mm i visina 110 mm. Robot je mase 3,5 kg i opremljen je standardnom punjivom baterijom 12 V/2 Ah , koja omogućuje autonomiju robota od 3 do 4 sata. Također roboti se mogu nadograđivati, kao što je i vidljivo na Slici 7. Žuti robot tako na sebi ima postavljenu kameru u boji rezolucije 720 x 576 te se može povezati sa osobnim računalom, iako prilikom toga jedinica za obradu slike koristi 640 x 480 VGA format s 32-bitnom rezolucijom. Crveni robot je nadograđen s uređajem koji mu omogućuje automatsko spajanje na punjač, kada se javi potreba za time.[22]



Slika 7. eMir Roboti

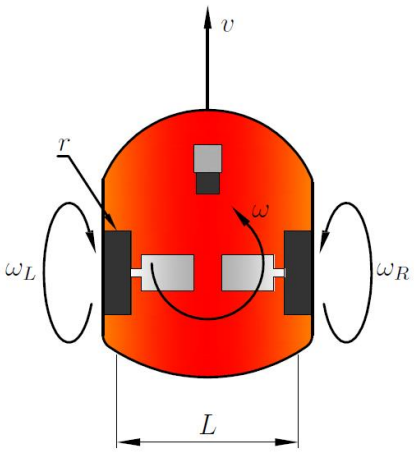
Kinematički model mobilnog robota dan je jednadžbama[22]:

$$v = \frac{r}{2}\left(\omega_R + \omega_L\right), \tag{3}$$

$$\omega = \frac{r}{L}\left(\omega_R - \omega_L\right), \tag{4}$$

,

gdje je v translacijska brzina, ω kutna brzina, ω_R i ω_L kutne brzine desnog i lijevog kotača, polumjer kotača r = 40 mm i razmak između kotača L = 240 mm, kao što je i prikazano na slici 2. Kada se u obzir uzme maksimalna brzina motora robota, dobiva se maksimalna translacijska brzina od oko 0,5 m/s i maksimalna kutna brzina od 240 °/s. Oba pogonska kotača izrađena su iz aluminijske legure i obložena su elastomernom oblogom, kako bi se povećalo trenje i smanjilo proklizavanje. [22]

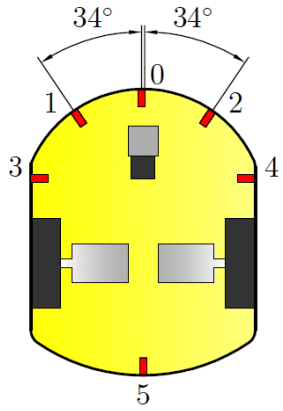


Slika 8. Osnovne dimenzije i gibanja [22]

Robot je pogonjen dvama istosmjernim motorima s reduktorima prijenosnih omjera 1:71. Motori su tipa IG-32M i razvijaju maksimalan moment od 0,32 Nm. Motori rade na naponu od 12 V iako mogu raditi i na 24 V, ali ovakvim smanjenim naponom nema potrebe za hlađenjem uz zadržavanje istih svojstava. Maksimalna jakost struje na bateriji 450 mA sa svim uključenim senzorima i kamerom.

3.1 Mjerni sustav

Mjerenje udaljenosti robota i okoline koja ga okružuje, vrši se pomoću 6 analognih infracrvenih senzora tipa GP2Y0A21 marke Sharp, koji se nalaze na određenim mjestima na robotu, kako je prikazano na Slici 3.



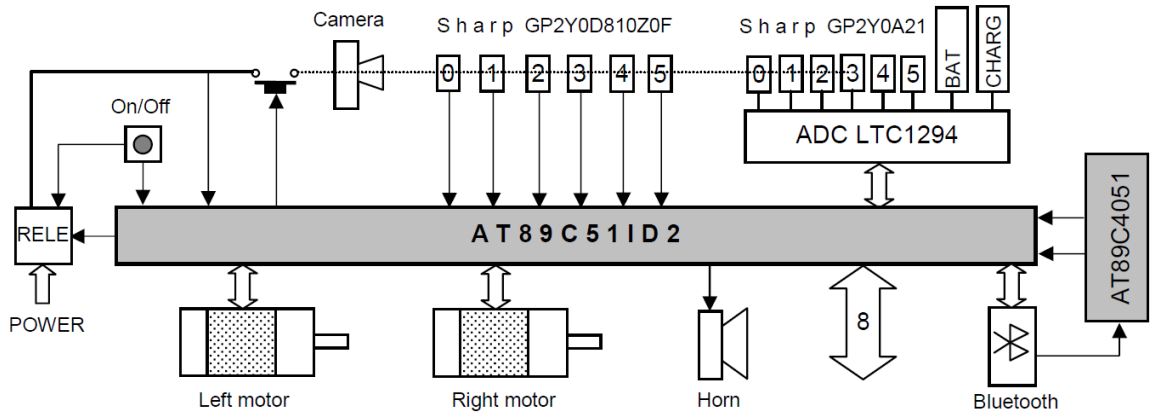
Slika 9. Mjerni sustav robota [1]

Pet senzora je namješteno da dobro percipiraju prostor ispred robota, dok je jedan senzor smješten sa stražnje strane. Senzori imaju raspon mjerenja od 10 cm do 80 cm, te mjere analogni signal koji se u analogno digitalnom pretvorniku pretvara u digitalni oblik, te se prosljeđuje mikrokontroleru. Mjerenje daljine, se vrši svakih 10 ms. Uz navedene senzore smješteni su i senzori GP2Y0D810Z0F tvrtke Sharp koji se uključuju ukoliko je zapreka na udaljenosti između 2 cm i 10 cm. Mjerenje brzine se vrši inkrementalnim enkoderima, smještenim uz vratilo motora. Ukupni broj impulsa po okretaju nakon multiplikacije signala iznosi 497, što daje rezoluciju od 0.72 ° po impulsu. Ukoliko se robot giba samo translacijski, ovo odgovara pomaku od 0,505 mm. Robot mjeri i napon baterije, koji se koristi u upravljačkom algoritmu robota, kako bi donio odluku o gašenju robota, kako ne bi nastala šteta na bateriji ili treba li robot staviti na punjenje. [22]

3.2 Upravljački sustav robota

Za upravljanje robota koristi se mikrokontroler AT89C51ID2 tvrtke Amtel, koji posjeduje 64 kB slobodne memorije i ima frekvenciju procesora od 1MHz. Shema upravljačkog sistema robota nalazi se na Slici 4. Kako mikrokontroler ne posjeduje A/D pretvorbu, koristi se vanjski serijski pretvarač ADC LTC1294. Uz glavni mikrokontroler koristi se i pomoćni

kontroler AT89C4051 za nadzor komunikacije.[22] Motorima se upravlja pomoću širinsko impulsne modulacije (PWM), a snaga se motorima dovodi preko integriranih krugova sa ugrađenim H-mostovima. [23]



Slika 10. Shema upravljačkog sustava robota [1]

Cijeli upravljački program zauzima 3,5 kB memorije, što čini samo 5% ukupne memorije mikrokontrolera. Upravljački program ima funkcije:

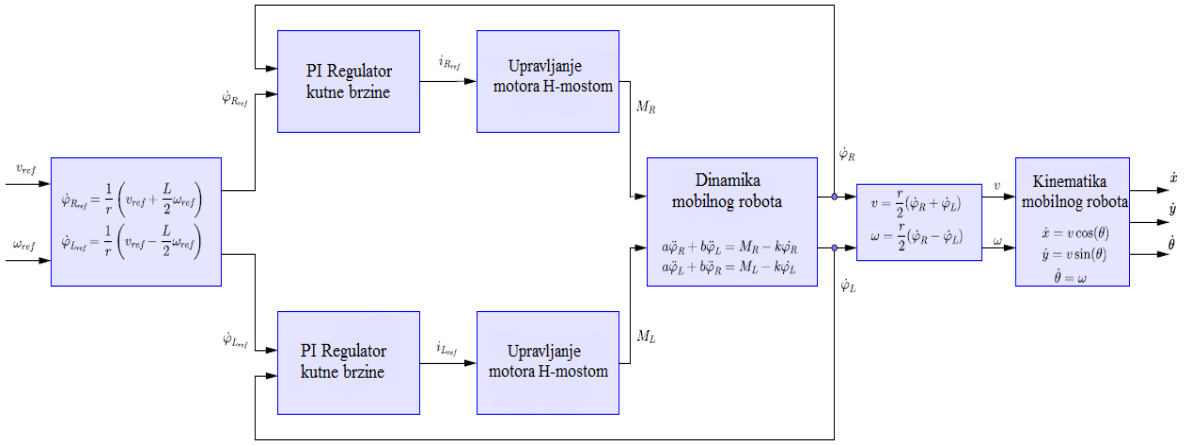
- kontrole, prekidanja i izvršavanja programa,
- mjerenja brzine motora svakih 260 ms,
- upravljanja brzine motora svakih 100 ms,
- očitavanja vrijednosti infracrvenog senzora svakih 10 ms,
- očitavanja stanja baterije svakih 100 ms,

Komuniciranje s mobilnim robotima vrši se bežičnom vezom, pomoću modula Sparfunk WRL-00582, koji omogućuje prijenos podataka brzinom 57600 bps, što zadovoljava interval prijema podataka od 0,1 s. Neke naredbe koje se šalju robotu prikazane su u sljedećoj tablici:

Tablica 3. Primjer naredbi za upravljanje e-MIR Robotom [22].

NN	Command	Description
0	`# 00 00 00 00 /`	Stop the robot and exit from a current task
1	`# 01 vv rr CS /`	Move the robot with translation velocity `vv` and angular velocity `rr` (–100 to +100%)
17	`# 11 PP 00 CS /`	Define returned information package `PP` (0, 1 or 2)
19	`# 13 tt 00 CS /`	Turn on internal horn for time `tt` tenths of seconds
255	`# FF 00 00 01 /`	Turn off the robot

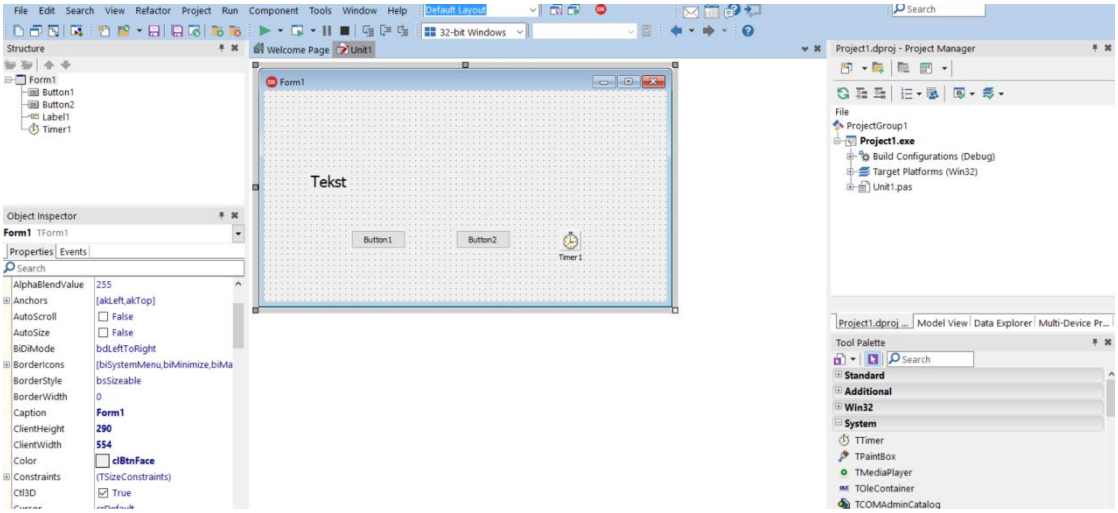
Svaka naredba treba započeti sa znakom „#“ i završiti sa znakom „/“. Između njih unose se informacije o modu rada, translacijskoj i rotacijskoj brzini i kontrolna vrijednost „CS“ (Check Sum) koja služi kao kontrolna vrijednost. Narede s računala se šalju svakih 4 do 10 puta u sekundi. Ukoliko se traže podaci sa senzora ili stanje baterije, nakon naredbe za njihovo uključivanje, robot računalu šalje skup podataka u kojemu se nalaze informacije o stanju daljinomjera, njegovoj vrijednosti, naponu baterije. Ukoliko se traži kinematika, upravljačka jedinica robota šalje podatke o trenutnoj translacijskoj i kutnoj brzini. Sve podatkovne vrijednosti šalju se u heksadekadskom brojevnom sustavu i omogućuju očitavanje stanja 3 do 5 puta u sekundi.[22] Kako se unutar upravljačke jedinice nalazi PI regulator kutne i translacijske brzine, korisniku izvana nije potreban pristup širinsko impulsnoj modulaciji i signalima s enkodera, već se robotu putem serijske veze šalju referentne kutne brzine ω i translacijske brzine v_t koje se zadaju u postocima.[23]



Slika 11. Struktura upravljanja diferencijalnog pogona mobilnog robota

4 OPIS PROGRAMSKOG SUČELJA

Za izradu ove aplikacije koristila se programska platforma RAD Studio XE5 tvrtke Embarcadero, koja je izvrstan alat za razvoj aplikacija za različite operacijske sustave, prvenstveno Windows, iOS i Android. Unutar RAD Studija primjenjuje se programski jezik Delphi, koji kao svoj programski prevoditelj koristi vlastito razvijeni Object Pascal, koji je modernizirana verzija programskog jezika Pascal. Delphi je objektno-orijentirani programski jezik koji omogućuje kreiranje vizualnog okruženja i komponenata za višestruko korištenje u raznim korisničkim programima.[22] Kao aplikacija za razvoj programa, Delphi predstavlja programiranje na visokoj razini i u praksi se koristi za razvoj sistemskih alata, igara, desktop, database i višeslojnih kompleksnih aplikacija. U Delphiju za izradu aplikacija, kao okosnica se koriste VLC („Visual Component Libary“), odnosno vizualne komponente objektno orijentiranog okvira, za razvoj vizualnog sučelja za Microsoft Windows. [23]



Slika 12. Primjer sučelja programa

U sučelju aplikacije u objektno orijentiranom programiranju, vizualno se postavljaju elementi, odnosno objekti sučelja. Primjer je prikazan na slici 12. Postavljene su tipke „Button1“ i „Button2“, kojima se mogu pridodati različita svojstva i događaji prilikom interakcije s tom tipkom na sučelju programa. Pridodavanjem događaja na objekt, poput klika na Button1, program Delphi stvara proceduru u kodu za taj događaj. Zatim, unutar te

procedure korisnik upisuje kod koji će se izvršiti prilikom klika na Button1, primjerice uključivanja objekta „Timer“.

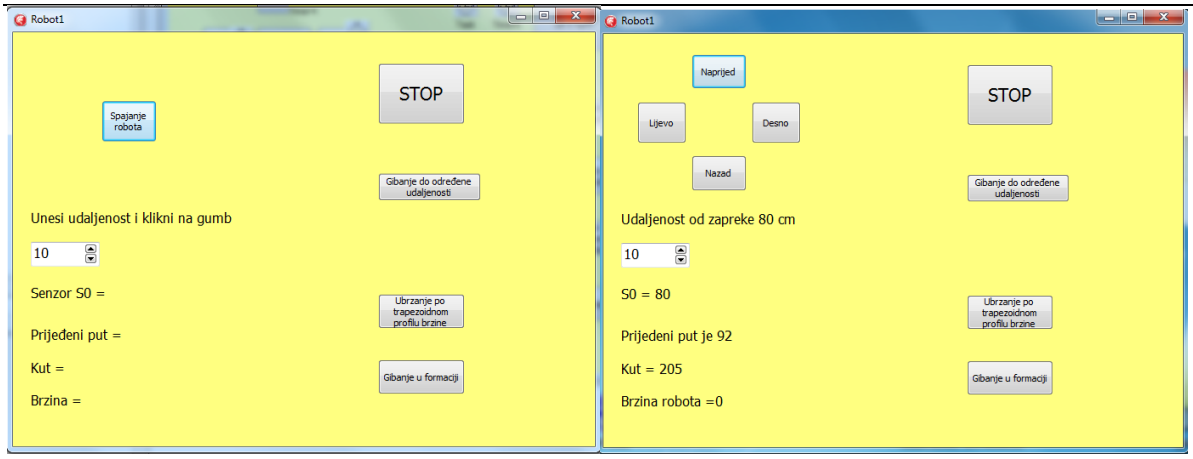
U ovom radu, priložene su tri aplikacije za decentralizirano upravljanje svakog robota zasebno, te aplikacija za decentralizirano i distribuirano upravljanje linearne formacije tri robota. Aplikacije se pokreću kao .exe datoteke, koje su dobivene debugiranjem dizajnirane programske aplikacije u Delphiju. U prilogu se nalazi kod programa za upravljanje linearnom formacijom tri eMIR mobilna robota, kao i CD-sa svim programskim aplikacijama i dodatnim datotekama.

4.1 Programsko sučelje aplikacije za jednog robota

Za decentralizirano upravljanje, napravljena je aplikacija za upravljanje jednog robota, preko koje se ne dobivaju niti dijele informacije s drugim robotima. Sve tri aplikacije programirane su na sličan način, odnosno kod im je relativno sličan. Aplikacije se razlikuju jedino bojom sučelja, koja ovisi o boji određenog eMIR robota, i drugačijom naredbom spajanja robota preko Bluetooth serijske komunikacije.

Tako je prije svega potrebno da računalo ima mogućnost slanja i primanja informacija pomoću Bluetooth veze. Ukoliko je takva veza omogućena potrebno je uključiti Bluethoot uređaj i spojiti se s robotom. Prilikom spajanja računalo pridodaje komunikacijski priključak (COM Port –„Communication Port“) koji mora biti jedinstven za svakog robota. Unutar same aplikacije koriste se i izvorna datoteka Unit_eMIR.pas profesora M. Crnekovića. Unit_eMIR.pas sadrži skup procedura kojima se robotu zadaju naredbe spajanja, gašenja, translacije, rotacije i drugih. Kako se te naredbe zadaju samo jednom robotu, unutar te datoteke postoji niz naredbi kojima se određuje s kojim se komunikacijskim priključkom komunicira s robotom. Za najlakše spajanje, potrebno je stvoriti .cfg datoteku u kojoj se nalaze redni brojevi komunikacijskih uređaja pridodani imenu robota, te ju je potrebno smjestiti glavni lokalni disk (C:). Primjer te datoteke nalazi se na CD-u, u prilogu.

Unutar programskog sučelja, koja je prikazana na slici 13. Postoji tipka za spajanje robota te tipke za određene načine gibanja. Uz njih se nalaze oznake s tekstom koje odmah pri uključivanju robota prikazuju udaljenost do zapreke ili drugog robota, prijeđeni put, kut i brzinu robota. Nakon spajanja robota nestaje tipka za spajanje i umjesto nje se javljaju tipke za upravljanje robota u različitim smjerovima, njegovom translacijom ili rotacijom.



Slika 13. Sučelje programa prije i nakon uključivanja žutog robota

Unutar samog sučelja postoje tipke koje pružaju mogućnost različitog gibanja mobilnog robota. Pritiskom na određenu tipku, uključuje se algoritam za gibanje navedeno na toj tipki. Mobilni robot se može postaviti na vodeće mjesto u formaciji, gdje mu se zadaje referentno gibanje ili ga se može postaviti na mjesta iza vodećeg, te se tada giba kao sljedbeni mobilni robot.

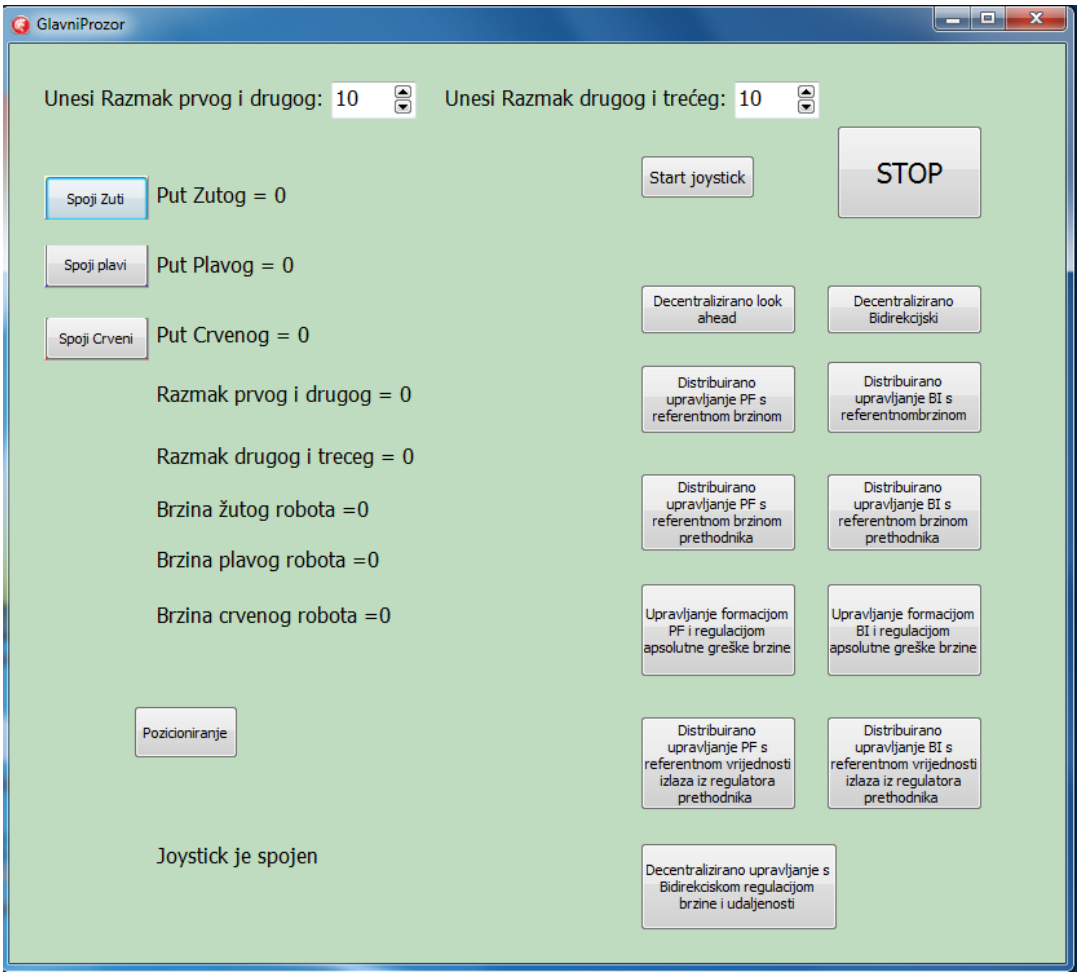
4.2 Programsko sučelje aplikacije za upravljanje kolone robota

Aplikacija za sva upravljanje sa tri eMIR mobilna robota programirana je na sličan način kao i za jednog robota. U ovom slučaju korištena je izvorna datoteka Unit_3eMIR.pas profesora M. Crnekovića, koja sadrži iste naredbe za upravljanje kao i Unit_eMIR.pas, ali za sva tri robota. Prilikom ispisivanja naredbi, kako bi se znalo kojem su robotu dodijeljene naredbe, dodaje im se predznak boje robota. Time se informacije šalju i primaju preko onog komunikacijskog priključka koji je pridodan za zadani robot, odnosno predznak boje robota.
Kako se ovom aplikacijom upravljaju roboti u linearnoj formaciji, tj. koloni vozila, potrebno ih je postaviti u liniju i tako da je žuti mobilni robot na prvoj poziciji, na poziciji vođe kojem se zadaje referentno gibanje i zatim iza njega plavi, te crveni na kraju kolone.
Kao u programskom sučelju za jednog robota, postoje tipke za spajanje robota. Pojavom boje robota na mjestu tipke za spajanje, dana nam je informacija da je robot spojen. Sučelje programa je prilikom samog pokretanja prikazana na slici 14., dok slika 15. prikazuje aplikaciju u toku izvršavanja određenog načina upravljanja kolone mobilnih robota. U sučelju aplikacije će stajati i obavijest da li je joystick spojen s računalom. Žuti robot, koji je u ovoj

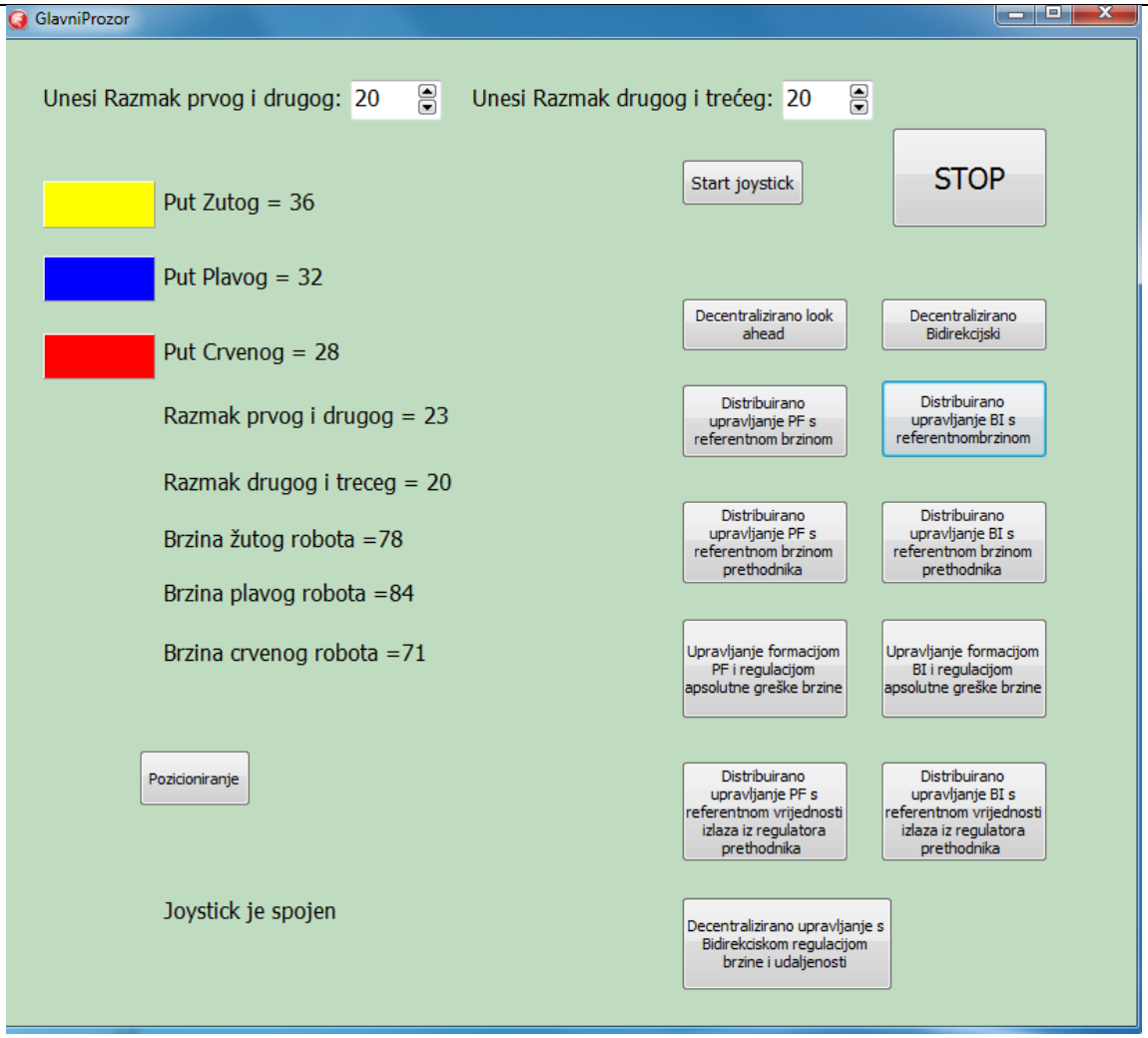
aplikaciji postavljen kao vodeći, može se translacijski upravljati i s joystickom, ukoliko je odabrana tipka za vođenje formacije joystickom. U sučelju aplikacije na slici 14. Mogu se pratiti određene informacije o gibanju, stanju robota, i razmaka između njih. U ovoj aplikaciji postoji više mogućih odabira algoritama upravljanja, čiji je naziv naveden na naslovu tipke, te je svaki od njih razrađen u eksperimentalnom dijelu rada s dobivenim vrijednostima.

Prije svega potrebno je podesiti željene razmake između vozila. Iako su namješteni na minimalnu početnu vrijednost od 10 cm, mogu se podesiti na višu vrijednost, što je i poželjno. Prilikom mjerenja podataka za eksperimentalni dio rada, vrijednosti referentne udaljenosti između mobilnih robota u formaciji bile su podešene na udaljenost 20 cm.

Ukoliko je potrebno poravnavanje, potrebno je upisati željeni razmak i kliknuti na tipku „poravnavanje“, sve dok se ne postignu željeni razmaci u formaciji. Tipka STOP, omogućuje zaustavljanje svih vrsta gibanja, ukoliko se želi zaustaviti gibanje formacije u određenom trenutku ili uslijed opasnosti od sudara, odnosno nestabilnosti.



Slika 14. Programsko sučelje za gibanje 3 robota u koloni

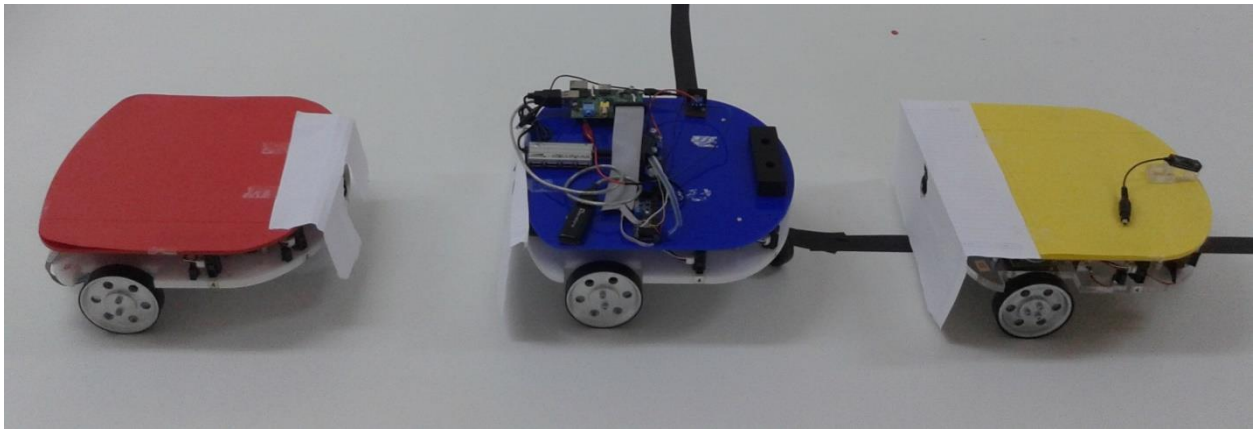


Slika 15. Programsko sučelje za gibanje 3 robota u koloni, tokom zadanog gibanja

Kada je klikom odabran način gibanja formacije, uključuje se algoritam koji u posebnoj programskoj petlji, svakih 100 ms zapisuje podatke u tekstualnu datoteku Record.txt. Klikom na tipku STOP, prekida se zapisivanje podataka. Tekstualna datoteka Record.txt. nalazi na istome mjestu gdje i sama aplikacija, te se iz nje dobivaju željeni brojčani podaci o poslanim i primljenim informacijama. Vrijednosti dobivene iz te datoteke korištene su za prikazivanje grafova u eksperimentalnom dijelu ovoga rada.

5 UPRAVLJANJE ROBOTIMA U LINEARNOJ FORMACIJI

U ovoj Aplikaciji promatrati će se različiti algoritmi i načini upravljanja niza mobilnih robota, odnosno kolona mobilnih robota. Vod (eng. „platoon“), ili kolona, je s gledišta upravljanja jednodimenzionalna mreža dinamičkih sistema, u kojoj mobilni roboti dijele informacije sa susjednim robotima i tako izvedenim regulatorima koji im omogućuju globalnu koordinaciju. [4] Slika 16. Prikazuje kolonu mobilnih robota sa žutim robotom na vodećoj poziciji koji će voditi gibanje formacije, po referentnom gibanju koje mu je zadano u aplikaciji. Također, na robote su nalijepljeni papiri s otvorima za senzore, kako bi se infracrvenim senzorima omogućilo što preciznije mjerenje međusobne udaljenosti. Bitno je napomenuti da senzori navedeni u opisu robota, iako imaju mjerno područje od 10 do 80 cm, najtočnije mjere razmak od oko 20 cm. Približavanjem gornjoj granici mjernog područja, očitanja senzora značajno osciliraju oko određene vrijednosti. Ukoliko se zapreka ispred senzora nalazi na udaljenosti od oko 10 cm, postoji mogućnost da senzor izlazi iz mjernog područja, te daje vrijednost 0 cm.



Slika 16. eMir Roboti u linearnoj formaciji

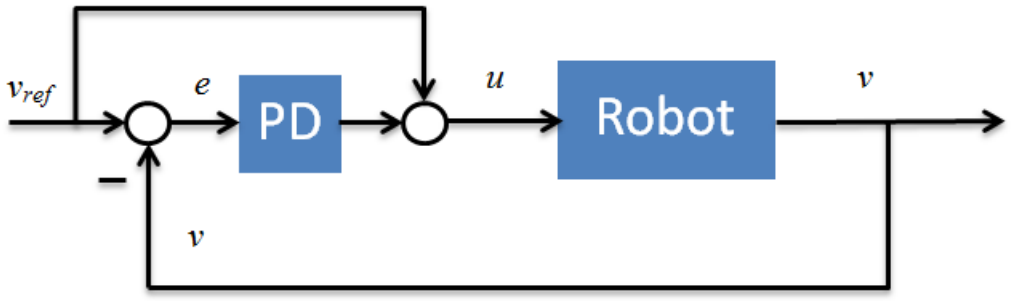
Poligon na kojemu se provode istraživanja vezana uz eMIR robote, nalazi se u laboratoriju Katedre za strojarsku automatiku. Dimenzija laboratrija je 4 m x 2 m, s dovoljnom visinom za lagano ručno postavljanje robota u formaciju.

Za upravljanje mobilnih robota u formaciji možemo koristiti decentralizirano i distribuirano upravljanje, te će se ti načini izvesti i eksperimentalno provjeriti na opisanim robotima. Kako bi se rezultati mogli lakše uspoređivati, potrebno je napraviti jednak referentni model gibanja.

5.1 Decentralizirano upravljanje kolonom mobilnih robota

Kako je objašnjeno u ranijim poglavljima rada, za decentralizirano upravljanje kolone mobilnih robota mogu se koristiti aplikacije za svakog robota zasebno. Robote treba postaviti u linearnu formaciju i pokrenuti sve tri aplikacije. Zatim je u aplikacijama robota koji su sljedbenici potrebno kliknuti tipku za gibanje u formaciji, te je u aplikaciji vodećeg robota potrebno pokrenuti gibanje po profilu brzine. Time plavi i crveni robot slijede vodećeg žutog, koristeći se samo informacijama dobivenim preko senzora. Također, u aplikaciji za upravljanje svih robota primijenjen je isti algoritam, te će se ona koristiti radi lakšeg ispisivanja podataka.

Prvo se definira upravljanje brzine vodećeg vozila po nekom profilu brzina. Tako je u ovoj slici dano upravljanje vodećeg robota (Slika 17.), gdje se brzina robota regulira PD regulatorom. Referentna ulazna vrijednost je zadana postotkom brzine, odnosno ubrzanjem do 80% maksimalne brzine robota, te se na toj brzini robot zadržava nekoliko sekundi. Nakon toga slijedi usporavanje do zaustavljanja, istim vremenom promjene brzine kao i kod ubrzavanja. Primjer regulacije robota po takvom trapezoidnom profilu brzine prikazan je na slici 18, gdje je referentna brzina prikazana zelenom, brzina koju robot očitava crnom i prijeđeni put crvenom bojom.



Slika 17. PD regulator brzine robota

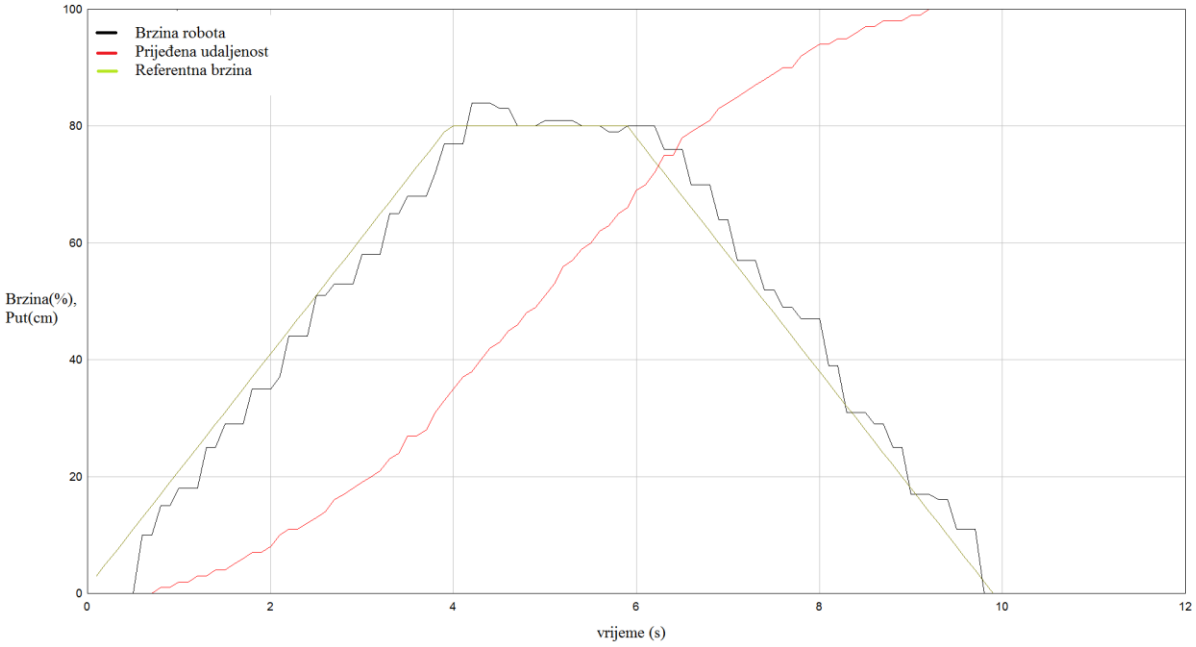
Izlaz iz regulatora, odnosno ulazna vrijednost koja se predaje robotu, računa se prema:

$$u = (K_p \bullet e(t)) + \left(K_d \bullet \frac{d}{dt} e(t) \right) + v_{ref}, \tag{5}$$

odnosno

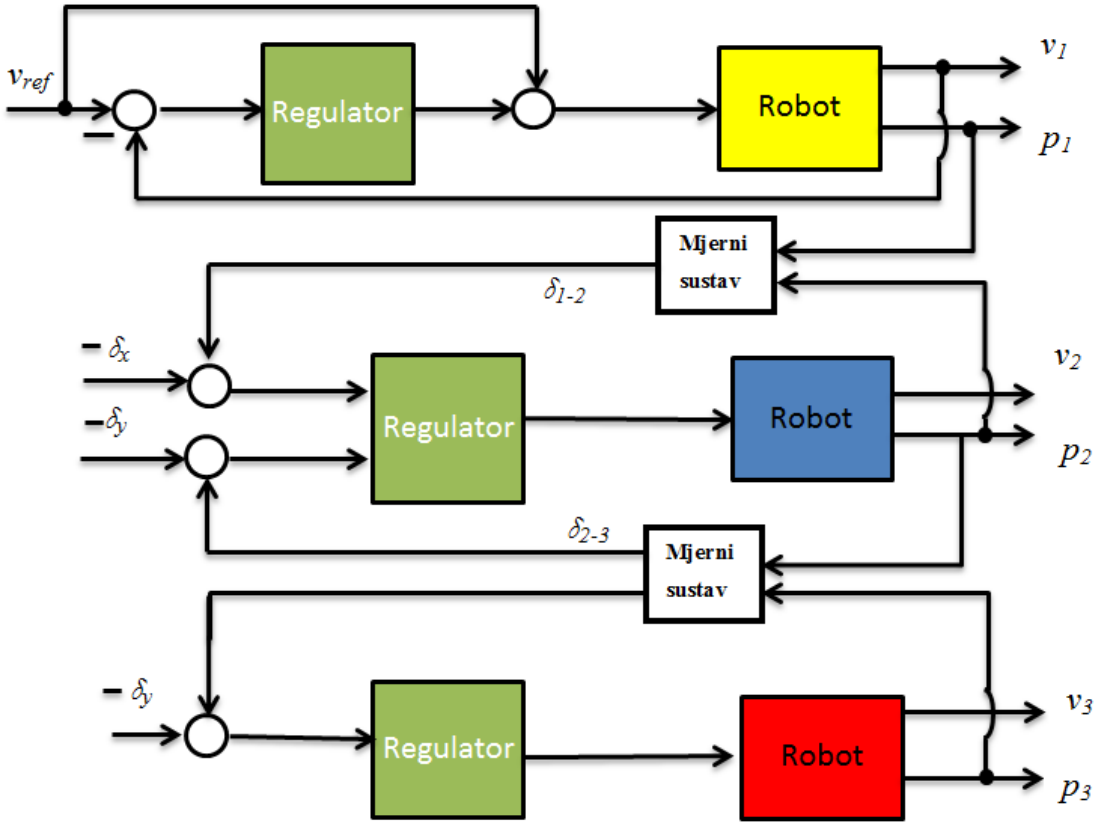
$$u = (K_p \bullet e) + \left(K_d \bullet \frac{e - e_{prošli}}{t_{iteracije}} \right) + v_{ref}, \tag{6}$$

gdje su K_p i K_d, proporcionalna i derivacijska pojačanja PD regulatora koja djeluju na pogrešku e referentne brzine robota v_{ref} i stvarne brzine robota v. Derivacijsko djelovanje u diskretnom sustavu, računa se kao razlika sadašnje i prijašnje pogreške u vremenu iteracije, odnosno u slučaju ove aplikacije, između izvršavanja petlji računalnog koda.



Slika 18. Gibanje robota po trapezoidnom profilu brzine.

Na dijagramu je prikazano reguliranje translacijske brzine robota po trapezoidnom profilu brzine, pojačanjima $K_p = 0{,}45$ i $K_d = 0{,}03$. Isprekidani skokovi brzine robota rezultat su relativno dugog vremenskog razmaka (260 ms) kojim robot šalje podatak o svojoj brzini. Na grafu je prikazan i prijeđeni put mobilnog robota od 0 do 100 cm unutar 10 sekundi.
Struktura i zakon takvog upravljanja prikazani su na slici 19. I formulama koje slijede.



Slika 19. Struktura decentraliziranog upravljanja robota u koloni.

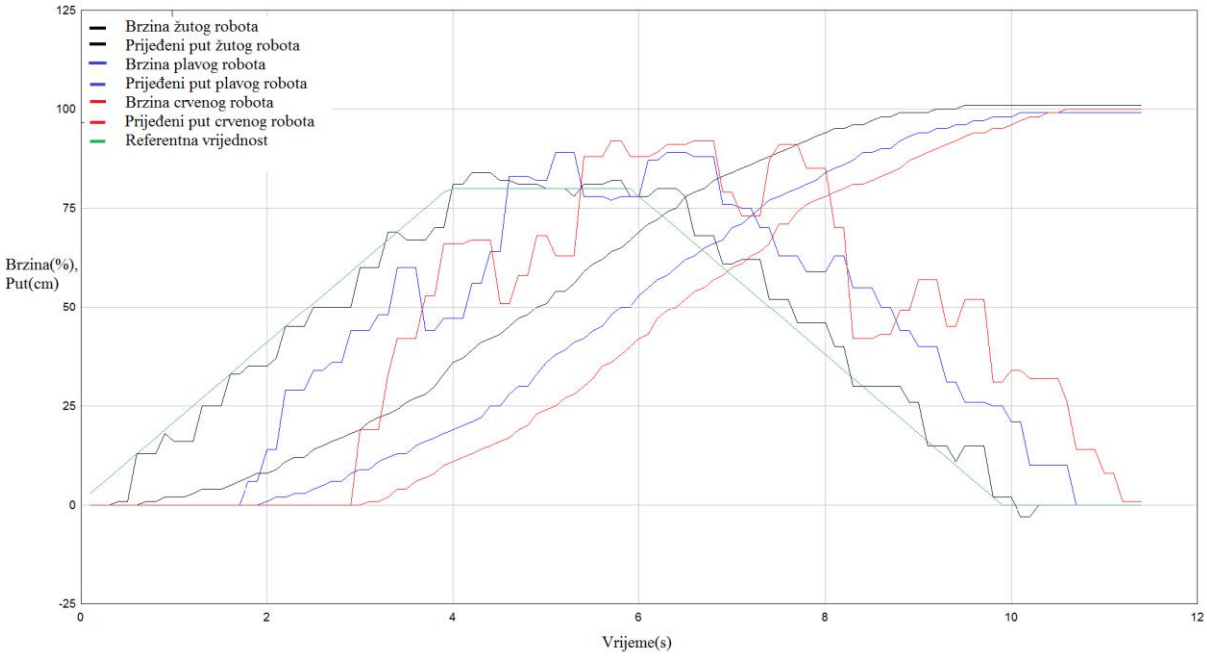
U decentraliziranom slučaju prikazanom na slici 19., na vodeću poziciju je postavljen žuti robot kojem je zadano referentno gibanje, kao što je prethodno prikazano u radu. Sljedbeni roboti upravljani su lokalnim regulatorom prema:

$$u_2 = f_1\left(\delta_{1-2} - \delta_x\right) + b_1\left(-\delta_{2-3} + \delta_y\right), \tag{7}$$

$$u_3 = f_2\left(\delta_{1-2} - \delta_y\right), \tag{8}$$

gdje su u osnovnom nesimetričnom slučaju, odnosno u strategiji gledanja unaprijed pojačanja $f_n \neq 0$, a pojačanja $b_n = 0$. δ_x i δ_y su željene udaljenosti između prvog i drugog, odnosno drugog i trećeg mobilnog robota, a δ_{1-2} i δ_{2-3} stvarne udaljenosti očitanja senzora s prednje i stražnje strane robota kojima su u programskom kodu dodijeljeni indeksi „S0“ i „S5“. Implementacijom ovakvog načina upravljanja unutar aplikacije, i primjenom na eMIR robotima, dobiveni rezultati su prikazani na grafu brzina i puta gibanja robota u koloni prema referentnom trapezoidnom profilu brzina (slika 20). Radi boljeg raspoznavanja, prijeđena udaljenost i brzina žutog robota, u vremenu su označene crnom bojom, dok su brzine i putevi

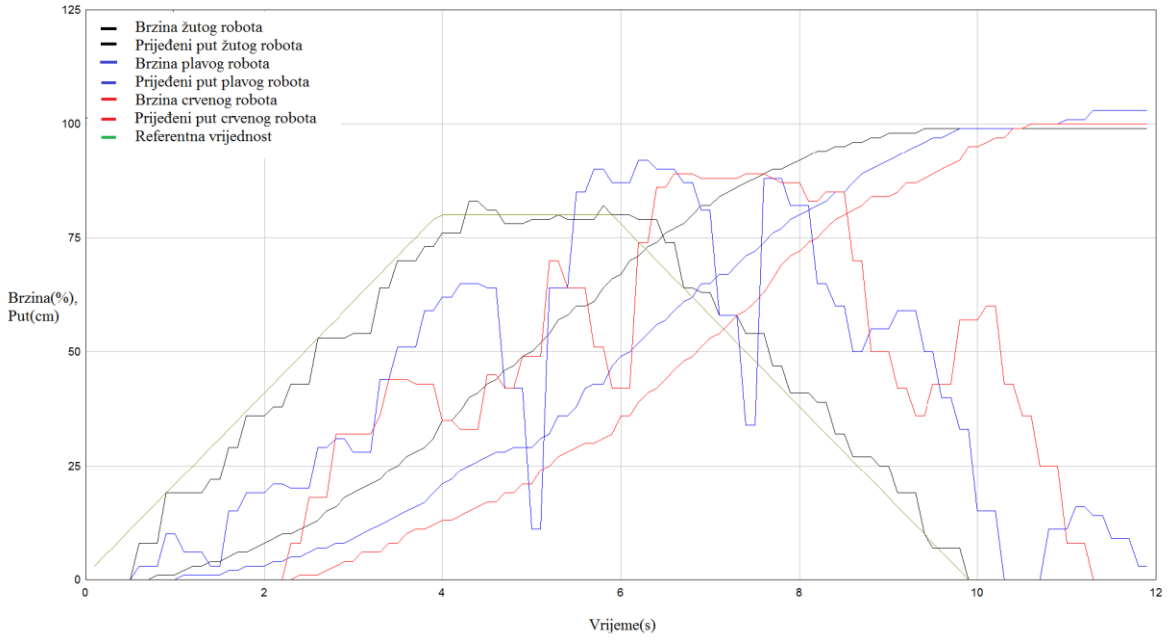
plavog i crvenog eMIR robota označeni plavom i crvenom. Referentna vrijednost brzine, izražena u postocima, prikazana je zelenom bojom.



Slika 20. Prikaz decentraliziranog upravljanja formacijom robota s uključenim prednjim senzorom

Na grafu (slika 20.) prikazano je kako sljedbeni roboti prate vodećeg po danom profilu brzina. Korištenjem informacije samo s prednjeg senzora i pojačanjima $f_1 = f_2 = 3{,}6$, vidljivo je blago kašnjenje sljedbenih robota, te kako oni pomoću proporcionalnog regulatora smanjuju pogrešku udaljenosti između sebe i prethodnika. Dobivene vrijednosti su očekivane zbog nedostatka informacija o referentnoj brzi gibanja kolone robota, odnosno nedostatka komunikacijskih veza s ostalim članovima formacije.

Također moguće je dodati i informaciju sa stražnjeg senzora robota, s pojačanjem $b \neq 0$, te time dobiti bolju raspodjelu pogreške udaljenosti između mobilnih robota. Kako je vidljivo iz grafa na slici 21., decentraliziranom metodom sa simetričnim pojačanjima ($f_n = b_n$), bez informacije o brzini vodećeg vozila, rezultati su nešto lošiji. U ovom slučaju plavi eMIR robot (linije plave boje na grafu), koji je smješten u sredinu kolone, pokušava ujednačiti udaljenosti između sebe i susjednih robota, pa su tako vidljiva nagla usporavanja i ubrzavanja tokom gibanja. Rezultat se može znatno poboljšati dodavanjem „feed-forward“ veze referentne brzine, što će se i pokazati u distribuiranom slučaju.



Slika 21. Prikaz decentraliziranog upravljanja formacijom robota s uključenim prednjim i stražnjim senzorom

5.2 Distribuirano upravljanje mobilnih robota u linearnoj formaciji modelom jednostrukog integratora

U ovom poglavlju koristi se aplikacija za upravljanje 3 eMIR mobilna robota u koloni. U ovome slučaju, navedena formacija robota prikazana je sljedećim modelom:

$$\dot{p}_n = d_n + u_n, \quad n \in \{1, ..., \mathrm{N}\}, \tag{9}$$

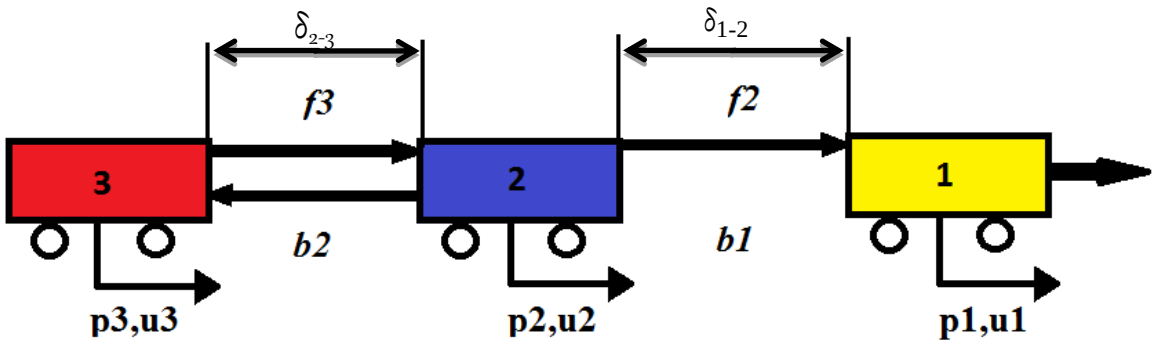
gdje je p_n pozicija n-tog vozila i d_n poremećaj, a u_n kontrolirana ulazna veličina koja direktno utječe na brzinu mobilnog robota. Željena pozicija *n*-tog mobilnog robota zadana je kao:

$$p_{d,n} = v_d t + n\delta, \tag{10}$$

gdje su v_d željena brzina gibanja kolone, a δ željeni razmak između dvaju susjednih vozila. Sva vozila u formaciji moraju imati pristup željenom razmaku i željenoj brzini gibanja formacije. Tako se mogu sintetizirati lokalizirani regulatori koji koriste greške relativnih pozicija između susjednih vozila:

$$u_n = -f_n\left(p_n - p_{n-1} - \delta\right) - b_n\left(p_n - p_{n+1} + \delta\right) + v_{ref}, \tag{11}$$

gdje f_n i b_n označavaju unaprijedno i unazadno pojačanje povratnih veza po poziciji *n*-tog vozila, a v_{ref} je referentna vrijednost brzine. Na slici 22. Ilustrirane su pozicije jednodimenzionalne formacije robota i odnosi udaljenosti između njih.



Slika 22. Jendnodimenzionalna formacija robota

U ovom radu, korištenjem aplikacije za upravljanje sva tri eMIR robota, koriste se modeli jednostrukog i dvostrukog integratora, u kojemu se povratna informacija o razmaku između robota dobiva pomoću infracrvenih duljinskih senzora. Tako se u modelu s jednostrukim integratorom dobiva:

$$u_n = -f_n\left(\delta_{n-(n-1)} - \delta_{ref}\right) - b_n\left(-\delta_{n-(n+1)} + \delta_{ref}\right) + v_{nref}\,, \tag{12}$$

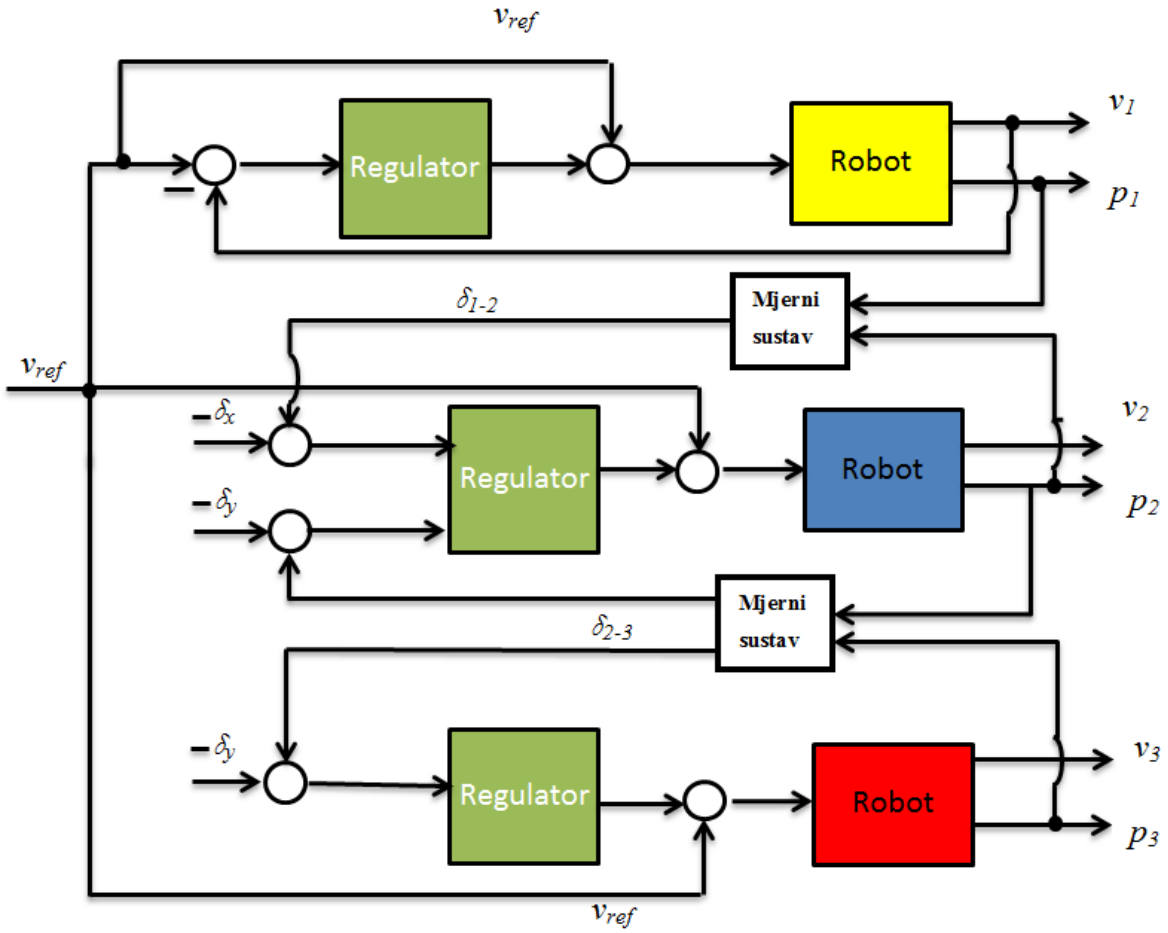
gdje su $\delta_{n-(n-1)}$ i $\delta_{n-(n+1)}$ udaljenosti izmjerene senzorima s prednje i stražnje strane robota i gdje je δ_{ref} referentna vrijednost razmaka između robota, a v_{nref} referentna brzina zadana *n*-tom mobilnom robotu. U matričnom obliku ovaj se distribuirani zakon upravljanja može prikazati kao:

$$u_n = -K\begin{pmatrix}\delta^e\\ v^e\end{pmatrix} + v_{nref}\,, \tag{13}$$

gdje δ^e označavaju pogreške razmaka između robota, v^e pogreške brzine vodećeg robota i u_n vektora reguliranih ulaza. Tako je primjerice $u_n = \begin{bmatrix}u_1 & u_2 & u_3\end{bmatrix}^T$. K je matrica u kojoj su definirana pojačanja danih vektora, te za različite načine reguliranja ima drugačije vrijednosti. U slučaju distribuiranog upravljanja tri mobilna robota u koloni, pod topologijom toka informacija, gdje se informacije o udaljenosti susjednih mobilnih robota dobivaju sa senzora, te je poznato referentno gibanje, zakon vođenja formacije se može izraziti kao:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = - \begin{pmatrix} 0 & 0 & g_1 \\ f_2 & b_2 & 0 \\ 0 & f_3 & 0 \end{pmatrix} * \begin{pmatrix} \delta^e_{1-2} \\ \delta^e_{2-3} \\ v^e_1 \end{pmatrix} + \begin{pmatrix} v_{ref\,1} \\ v_{ref\,2} \\ v_{ref\,3} \end{pmatrix}, \tag{14}$$

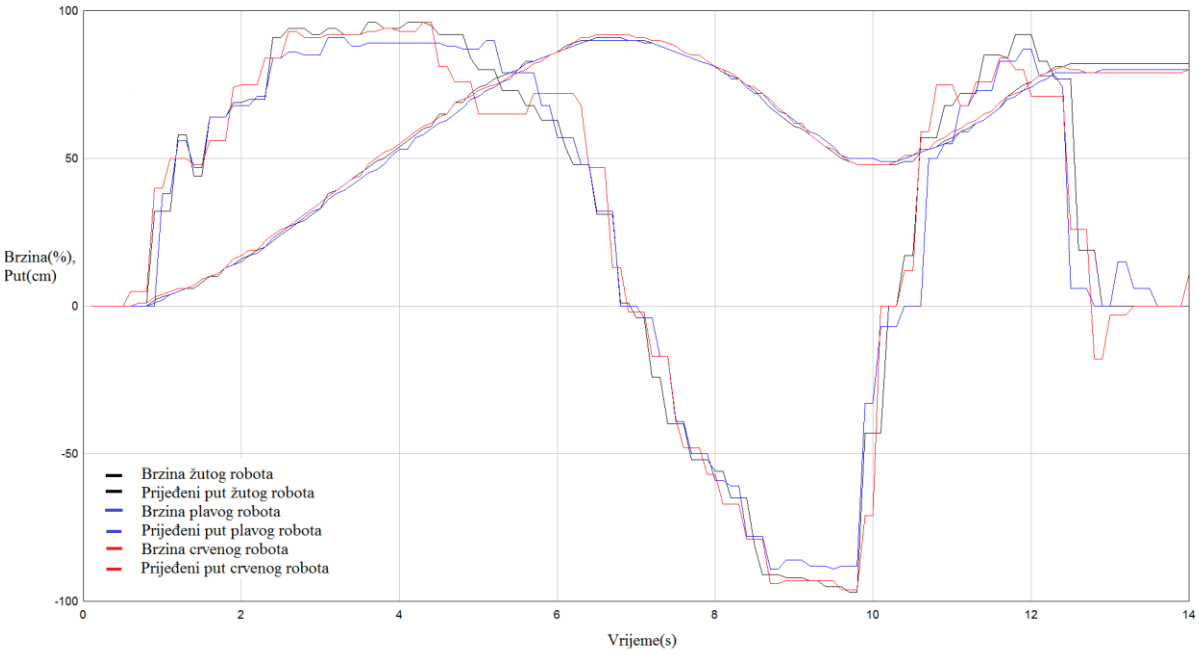
Shema navedenog upravljanja prikazana je na slici 23. Kako je navedeno u tekstu, svi lokalni kontrolirani ulazi dijele podatak o željenoj referentnoj brzini i sve su vrijednosti u vektoru referentnih brzina jednake za sve regulatore. Referentna brzina je zadana trapezoidnom promjenom brzine, prethodno navedenom u tekstu, pri čemu vodeći robot regulira svoju brzinu P regulatorom, dok je referentna brzina sljedbenim robotima unaprijedno dodana.



Slika 23. Struktura distribuiranog upravljanja robota s podacima o udaljenosti prethodnika i referentnoj brzini

U aplikaciji je dana mogućnost upravljanja vodećeg žutog robota s joystickom, ukoliko je on spojen s računalom. Postavljanjem robota u liniju i klikom na tipku „Start joystick“ pokreće se distribuirano gibanje mobilnih robota u koloni. Uključivanjem tog načina gibanja referentna brzina svih mobilnih robota je $v_{ref}=v_{joystick}$, pri tome korisnik određuje referentnu

brzinu postotkom nagiba joysticka, od 100% do -100%. Primjer gibanja kolone mobilnih robota prikazan je na slici 24. Boje linija koje predstavljanju određeni robot, iste su kao i kod decentraliziranog gibanja. Osim referentne brzine mobilni roboti dobivaju informacije o udaljenosti prethodnika, s unaprijednim pojačanjima $f_n=4$.

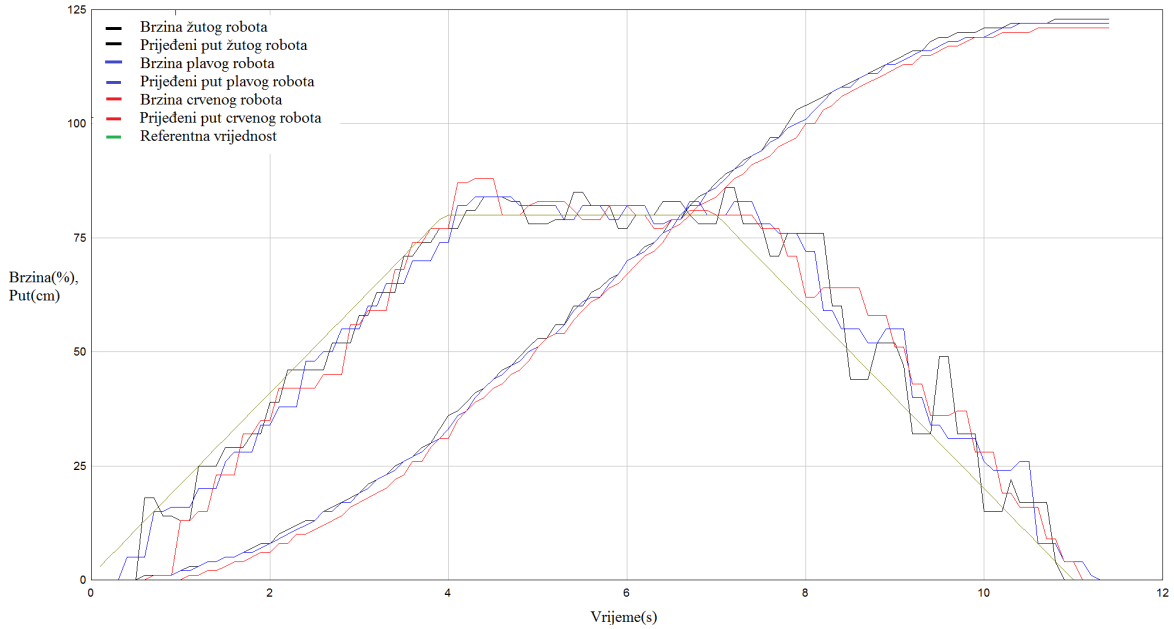


Slika 24. Gibanje kolone mobilnih robota vođene joystickom

Rezultati prikazani na grafikonu gibanja kolone, prikazuju mjesta kada dolazi do nekih odstupanja, odnosno pogrešaka u formaciji. Na mjestima s naglom promjenom brzine dolazi do prebačaja vrijednosti brzine. Ukoliko se nagibom joysticka zadaje maksimalna referentna brzina, brzina mobilnog robota varira zbog inercije robota i napona baterije, odnosno istrošenosti baterije robota. U ovom primjeru je vidljivo kako plavi robot razvija manju maksimalnu brzinu gibanja u odnosu na ostale robote. Provjerom napona, očitano je da je napon u bateriji plavog robota pao na 11.8 V, dok je napon žutog i plavog robota iznosio 12.2 V. Napon baterije utječe na iznos maksimalne brzine, a poželjno je da su baterije mobilnih robota jednako pune, odnosno da je napon na bateriji isti.

Kako bi se bolje usporedili rezultati, na slici 25. Prikazano je distribuirano upravljanje, gdje svaki član ima podatak o referentnoj brzini koja je trapezoidnog profila, te informaciju o udaljenosti prethodnika. Vidljivo je da se takvom komunikacijom ostvaruju vrlo dobri rezultati gibanja formacije. Time je postignuta bolja koherentnost i stabilnost formacije.

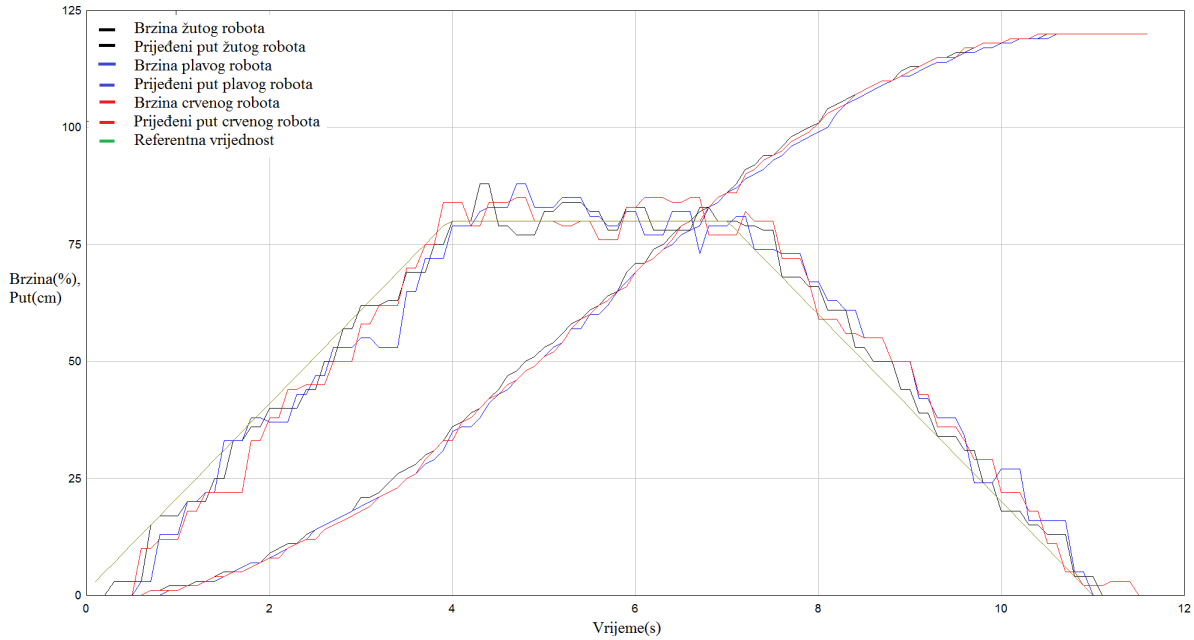
Vrijednosti pojačanja $f_2 = 3{,}6$ i $f_3 = 3{,}4$, koje se primjenjuju u ovom načinu upravljanja, podešene su slično kao u [4], u kojem su izračunate optimalne vrijednosti simetričnih kao i nesimetričnih pojačanja za veću kolonu vozila. Također je prikazano da se udaljavanjem od vodećeg vozila u koloni, moraju blago smanjivati pojačanja kako bi se postiglo bolje vođenje formacije.



Slika 25. Graf brzine i puta kolone robota, u kojoj se lokalno regulira udaljenost u odnosu na prethodnika, te je poznata referentna brzina gibanja kolone

Kada lokalni regulator ima *feedforward* vezu referentne brzine gibanja, istovremeno se pokreću svi mobilni roboti u formaciji, tako da nema kašnjenja sljedbenih robota. Razmak mobilnih robota se regulira samo prednjim senzorima (slika 25.) ili dvosmjerno prednjim i stražnjim senzorom (slika 26.). U slučaju dvosmjernog korištenja informacija, u kojem su primijenjena pojačanja $f_2 = b_n = 2.8$ i $f_3 = 2.6$, odnosno kada se regulira i pogreška udaljenosti sa sljedbenikom, iz grafa je vidljivo kako se tim pristupom postiže bolja regulacija pogrešaka razmaka između robota u koloni.

Usporedbom grafova sa slika 25. I 26., distribuiranog upravljanja mobilnih robota s informacijom o željenoj referentnoj brzini kolone, s grafovima decentraliziranog gibanja na slikama 20. I 21., vidljivo je da se postižu znatno bolji rezultati. Koherentnost formacije mobilnih robota je u slučaju poznavanja referentne brzine gibanja znatno bolja, odnosno vidljivo je očuvanje željenog razmaka između robota.

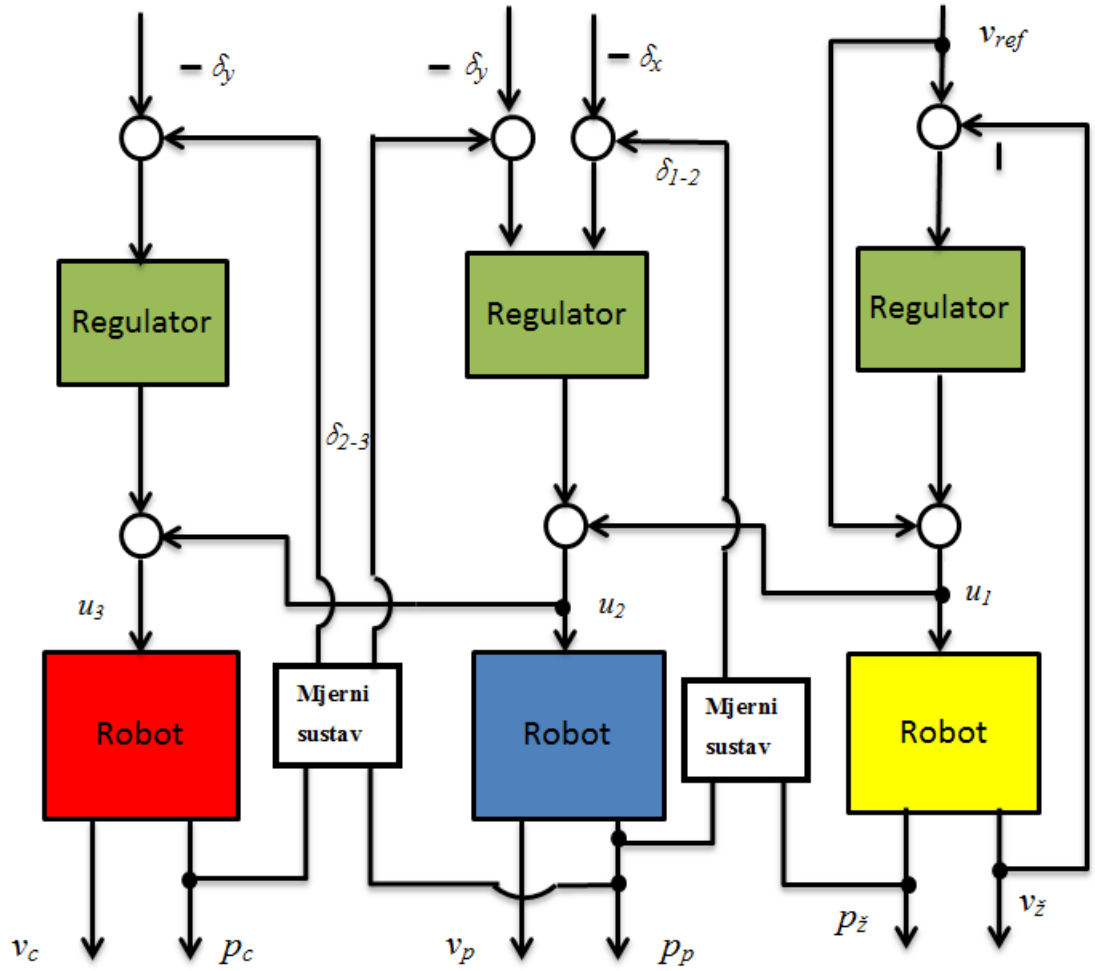


Slika 26. Graf brzine i puta kolone robota, u kojoj se lokalno regulira udaljenost u odnosu na prethodnika i sljedbenika, te je poznata referentna brzina gibanja kolone

Na grafikonu na slici 26., vidljivo je značajnije poboljšanje koherencije formacije mobilnih robota u slučaju korištenja informacije sa senzora ispred i iza i poznavanja referentne brzine gibanja kolone, naspram decentraliziranog slučaja (slika 21.).

5.3 Distribuirano upravljanje kolone mobilnih robota s informacijom o ulaznoj vrijednosti brzine prethodnika

Sljedeći način kojim se može upravljati kolonom mobilnih robota prikazan je na slici 27. I opisan je na sljedećoj stranici.

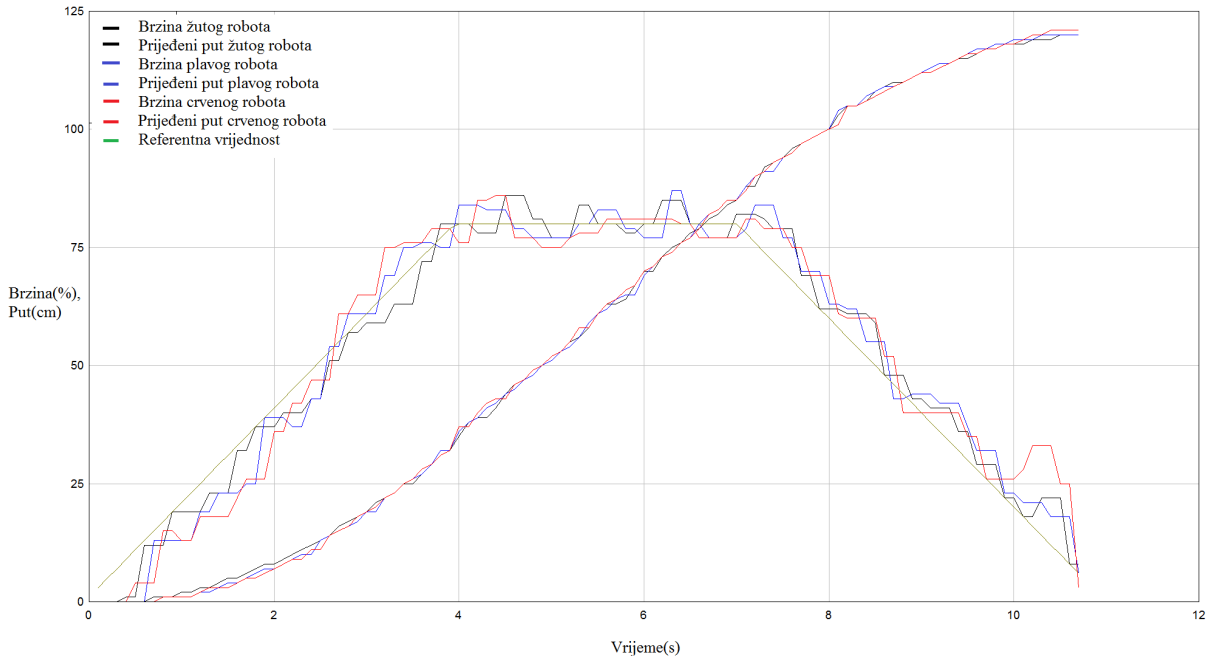


Slika 27. Struktura distribuiranog upravljanja kolone mobilnih robota s poznavanjem regulirane ulazne vrijednosti brzine prethodnika

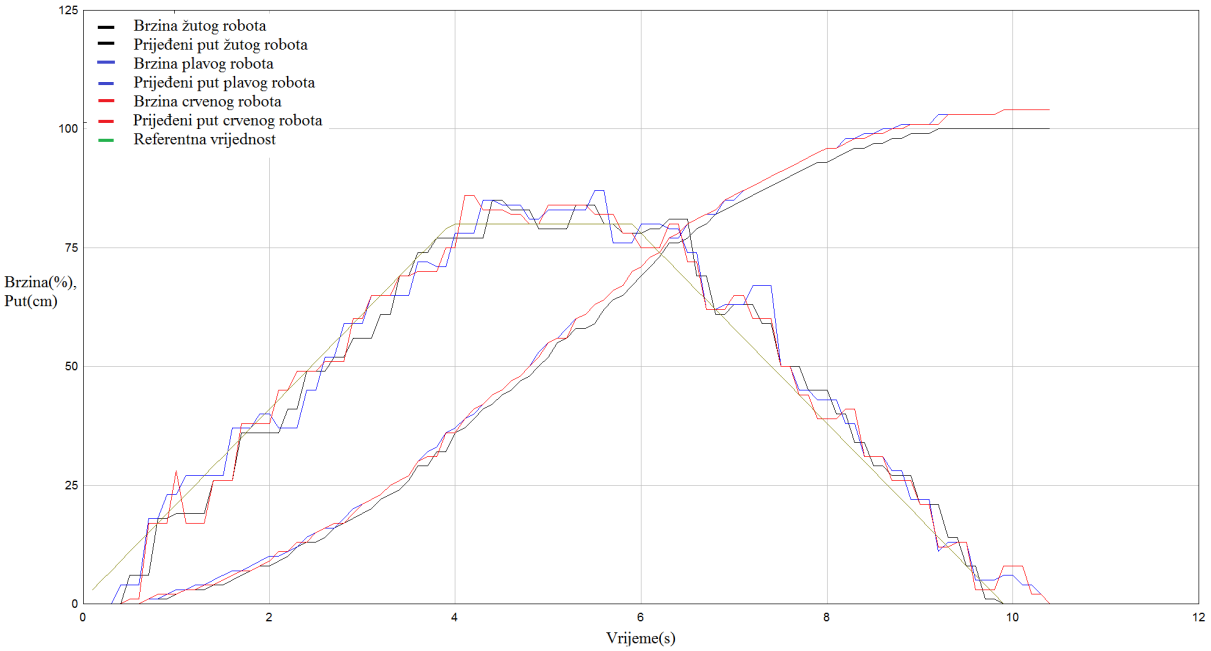
U ovom slučaju koristi se model s jednostrukim integratorom (9), kako je objašnjen u prethodnom upravljačkom algoritmu, u kojemu se mobilni roboti reguliraju prema udaljenosti između susjednih robota, te imaju podatak o željenoj brzini. U ovome slučaju je razlika što je unaprijedna referentna vrijednost koja se šalje mobilnom robotu, ulazna vrijednost kojom je reguliran prethodni mobilni robot. Takva se regulacija se može zapisati kao:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = -\begin{pmatrix} 0 & 0 & g_1 \\ f_2 & b_2 & 0 \\ 0 & f_3 & 0 \end{pmatrix} * \begin{pmatrix} \delta^e_{1-2} \\ \delta^e_{2-3} \\ v^e_1 \end{pmatrix} + \begin{pmatrix} v_{ref} \\ u_1 \\ u_2 \end{pmatrix}, \tag{15}$$

U aplikaciji dana pojačanja iznose $f_2 = b_2 = 3{,}2$ i $f_3 = 3$, te su odabirom gibanja za ovu shemu upravljanja dobiveni rezultati, koji su prikazani na slikama 28. I 29., ovisno o uključenim senzorima.



Slika 28. Graf brzine i puta kolone robota, s regulacijom udaljenosti u odnosu na prethodnika i vezom regulirane vrijednosti brzine prethodnika



Slika 29. Graf brzine i puta kolone robota, s regulacijom udaljenosti u odnosu na prethodnika i sljedbenika, i vezom regulirane vrijednosti brzine prethodnika

Odabirom ovog načina gibanja dobiveni su rezultati slični kao i u slučaju kada je mobilnom robotu dana informacija o gibanju vodećeg, gdje se postiže dobra stabilnost i koherentnost

formacije robota. Na slici 29. Vidljivo je kako zadnji, crveni robot odstupa od ukupne udaljenosti za 4 cm, koju imaju žuti i plavi robot, što nije bio slučaj kada se razmak regulirao samo pomoću informacija sa prednjih senzora.

5.4 Distribuirano upravljanje mobilnih robota u linearnoj formaciji korištenjem modela dvostrukog integratora

Kako bi se sustav bolje opisao, aplikacijom ćemo provjeriti i upravljanje linearnom formacijom na sljedećem modelu:

$$\ddot{p}_n = d_n + u_n,\ n \in \{1,...,\mathrm{N}\}, \tag{16}$$

Ovdje se radi o slučaju s dvostrukim integratorom, gdje su svi elementi u jednadžbi identični, kao i u prethodnom modelu (9), s jednostrukim integratorom. U ovom slučaju razmatraju se lokalizirani regulatori koji primaju informacije o relativnim pogreškama pozicije, odnosno razmaka između susjednih vozila, te informaciju o apsolutnim pogreškama brzine.

Pogreške razmaka robota su dobivene razlikom dobivene vrijednosti s infracrvenih senzora i referentne vrijednosti udaljenosti između susjednih robota. Vrijednost koju prima svaki robot može se izraziti kao:

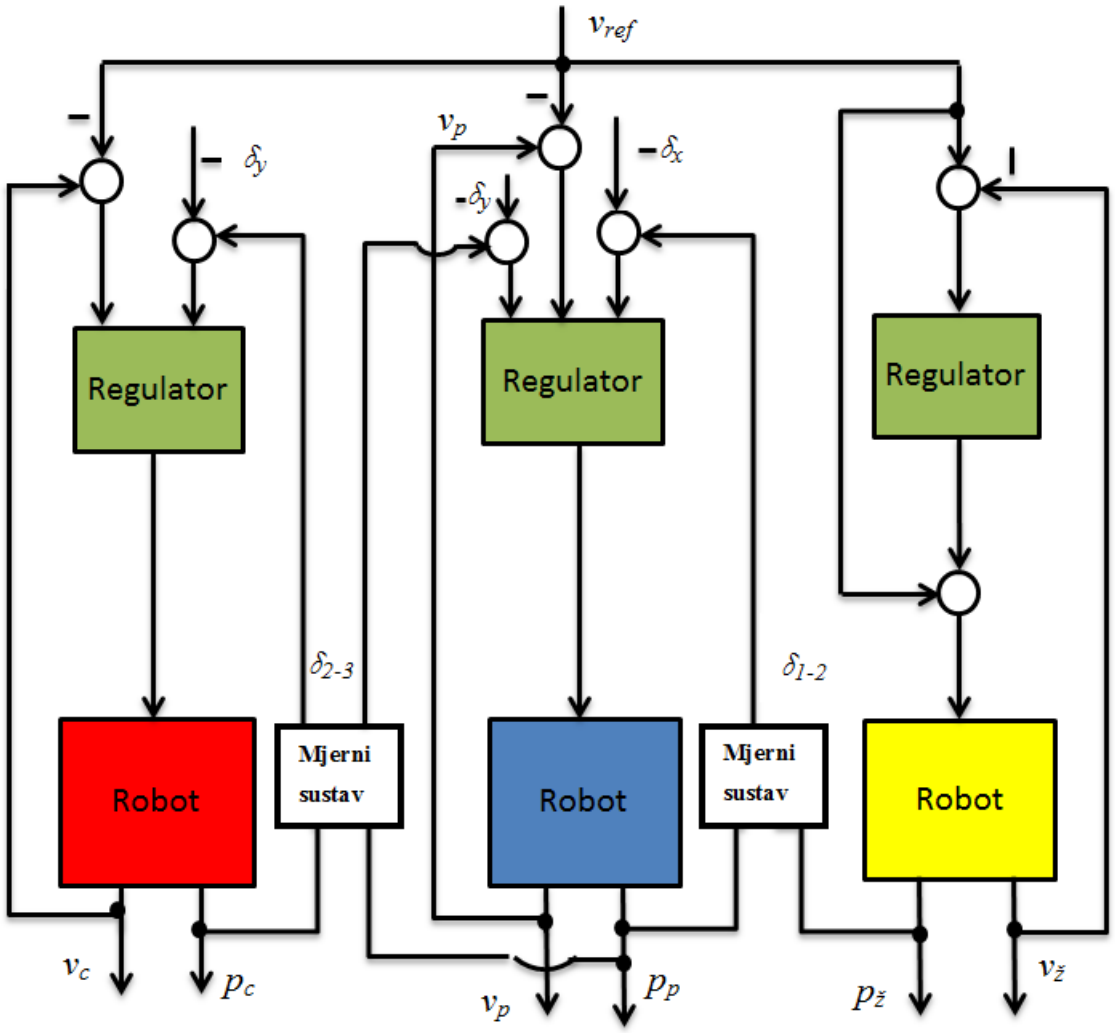
$$u_n = -f_n\left(\delta_{n-(n-1)} - \delta_x\right) - b_n\left(-\delta_{n-(n+1)} + \delta_y\right) - g_n(\dot{p}_n - v_{ref}), \tag{17}$$

Pri tome da su f_n i b_n unaprijedno i unazadno pojačanje pogreške pozicije, a g_n pojačanje pogreške brzine. U matričnom obliku ovaj zakon vođenja formacije se može zapisati kao:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = -\begin{pmatrix} 0 & 0 & g_1 & 0 & 0 \\ f_2 & b_2 & 0 & g_2 & 0 \\ 0 & f_3 & 0 & 0 & g_3 \end{pmatrix} \begin{pmatrix} \delta^e_{1-2} \\ \delta^e_{2-3} \\ v^e_1 \\ v^e_2 \\ v^e_3 \end{pmatrix}, \tag{18}$$

gdje δ^e_{1-2} i δ^e_{2-3} relativne pogreške udaljenosti između prvog i drugog, te drugog i trećeg robota, dok su v^e apsolutne pogreške brzine.

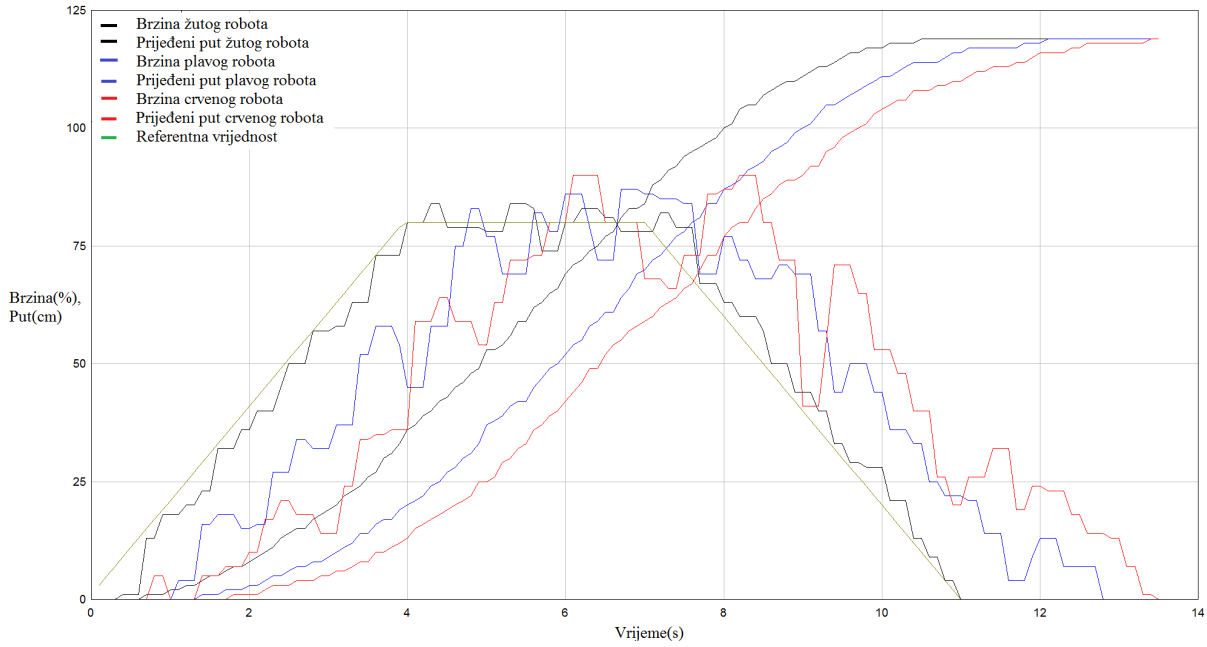
Na slici 30. Prikazana je struktura upravljanja linearne formacije mobilnih robota, gdje se mobilni roboti reguliraju prema udaljenosti od prethodnika, te prema referentnoj brzini gibanja formacije.



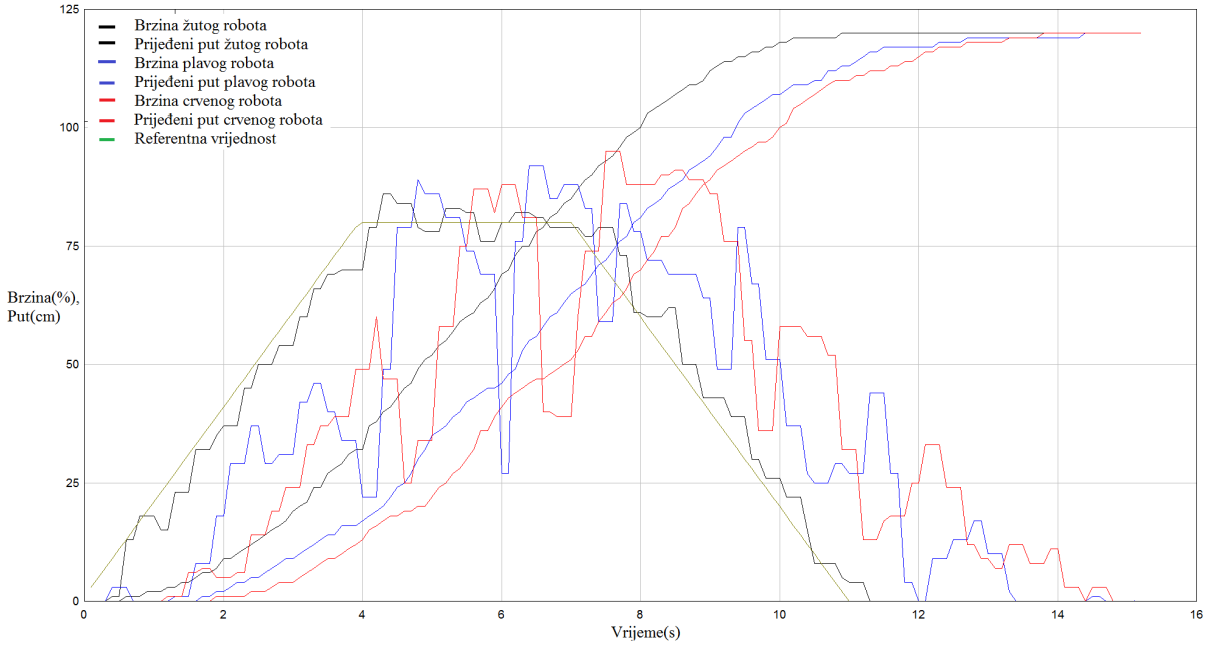
Slika 30. Struktura upravljanja formacije mobilnih robota, s regulacijom pogreške pozicije i apsolutne pogreške brzine

U programskoj aplikaciji, kao i za ostale algoritme, postoje tipke koje omogućuju upravljanje formacijom mobilnih robota ovim algoritmom. Pozicije se reguliraju samo pomoću prednjeg senzora ili obostrano korištenjem prednjeg i stražnjeg senzora. Kako je navedeno u pregledu literature, sustav s dvostrukim integratorom bolje opisuje ponašanje vozila, primjerice u koloni vozila na cesti, odnosno mobilnog robota u koloni, te su tako vidljiva kašnjenja i veće pogreške u formaciji. Ovakvim upravljanjem, bitnu ulogu imaju pojačanja regulatora, jer pravilnim odabirom regulatora bitno se poboljšava upravljanje formacije. U mnogim radovima autori se bave načinima kako poboljšati upravljanje ovakvim sustavom, te predlažu više metoda za optimalno sintetiziranje regulatora. Odabirom ovog algoritma u programskoj aplikaciji su uzeta sljedeća pojačanja sljedbenih robota: $f_2= b_2=3,8$, $f_3=3,6$ i $g_2=g_3=0,25$. U

grafu prikazanom na slici 31., robot prima informacije s prednjeg senzora o razmaku između prethodnika, dok u grafu na slici 32. Robot prima informacije bidirekcijski s oba senzora.



Slika 31. Upravljanje kolonom mobilnih robota s lokalnom regulacijom apsolutne pogreške brzine i udaljenosti do prethodnika



Slika 32. Upravljanje kolonom mobilnih robota s lokalnom regulacijom apsolutne pogreške brzine i udaljenosti do prethodnika i sljedbenika

Ovim modelom iz prikazanih grafova je vidljivo kako gibanje formacije nalikuje na „harmoniku“ gibanja. Na slici 31. Prikazano je kako se ovim modelom postižu rezultati slični

kao i kod decentraliziranog gibanja. Danim pojačanjima proporcionalno se regulira pogreška koja se smanjivanjem razmaka smanjuje, ali kako vodeći robot ubrzava i usporava dolazi do naglih skokova ili propada brzine. U ovome slučaju, bolja regulacija postignuta je s regulacijom razmaka robota samo između prethodnika, te je vidljivo da se u bidirekcijskom slučaju roboti točno pozicioniraju gotovo 2 *s* kasnije.

5.5 Distribuirano upravljanje mobilnih robota u linearnoj formaciji s regulacijom razmaka i brzine od susjednih mobilnih robota

U aplikaciji se može odabrati i metoda upravljanja prema kojoj se regulira razmak između susjednih mobilnih robota te pogreška pozicije s obzirom na izlaz iz regulatora prethodnog robota. Koristi se model s dvostrukim integratorom (16). U ovom se slučaju pogreška brzine sljedbenih robota računa kao razlika vrijednosti brzine robota i regulirane vrijednosti brzine prethodnog robota:

$$v_n^e = \dot{p} - u_{n-1}\,, \tag{19}$$

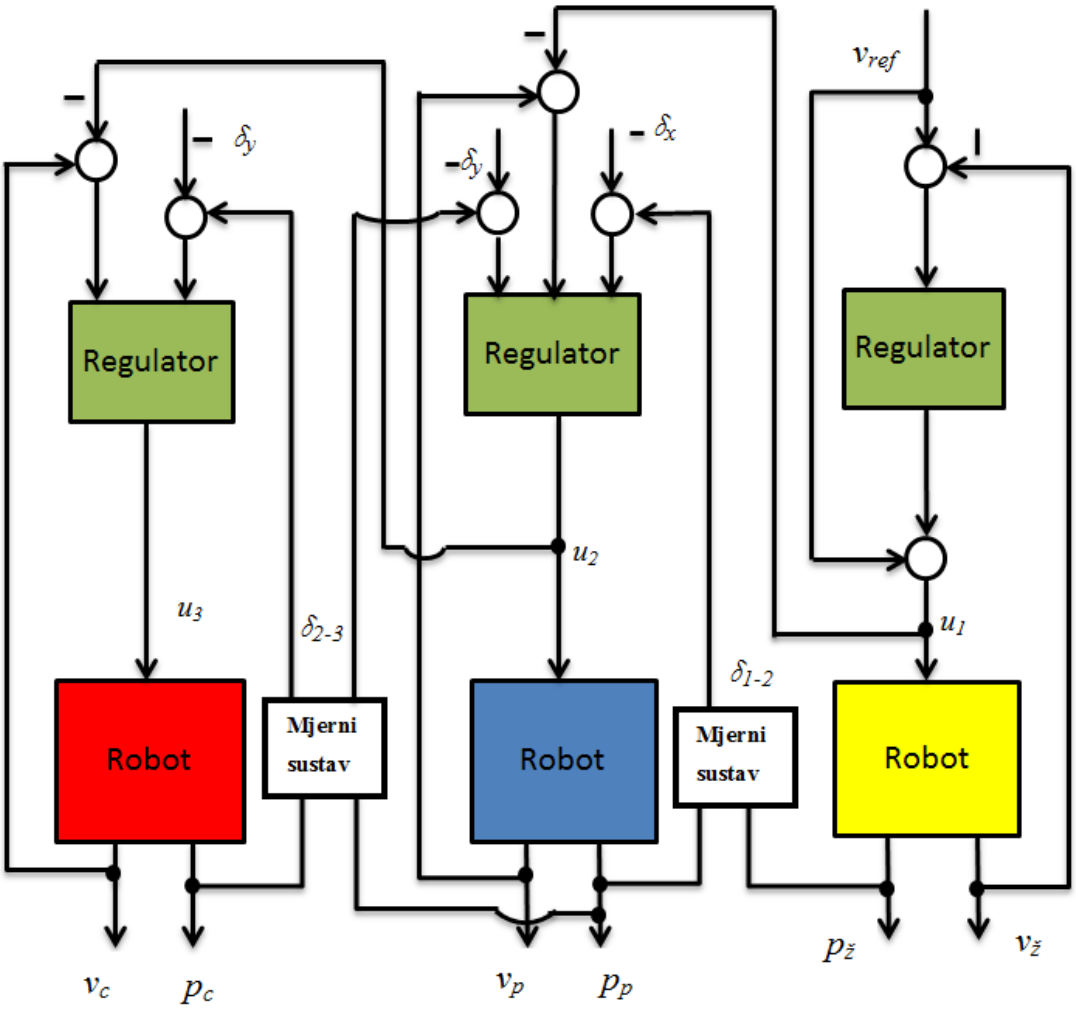
te se time zakon regulacije može zapisati kao:

$$u_n = -f_n\left(\delta_{n-(n-1)} - \delta_x\right) - b_n\left(-\delta_{n-(n+1)} + \delta_y\right) - g_n(\dot{p}_n - u_{n-1})\,, \tag{20}$$

Što se u matričnom obliku može prikazati:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = -\begin{pmatrix} 0 & 0 & g_1 & 0 & 0 \\ f_2 & b_2 & 0 & g_2 & 0 \\ 0 & f_3 & 0 & 0 & g_3 \end{pmatrix} \begin{pmatrix} \delta_{1-2}^e \\ \delta_{2-3}^e \\ v_1^0 \\ v_2^e \\ v_3^e \end{pmatrix}, \tag{21}$$

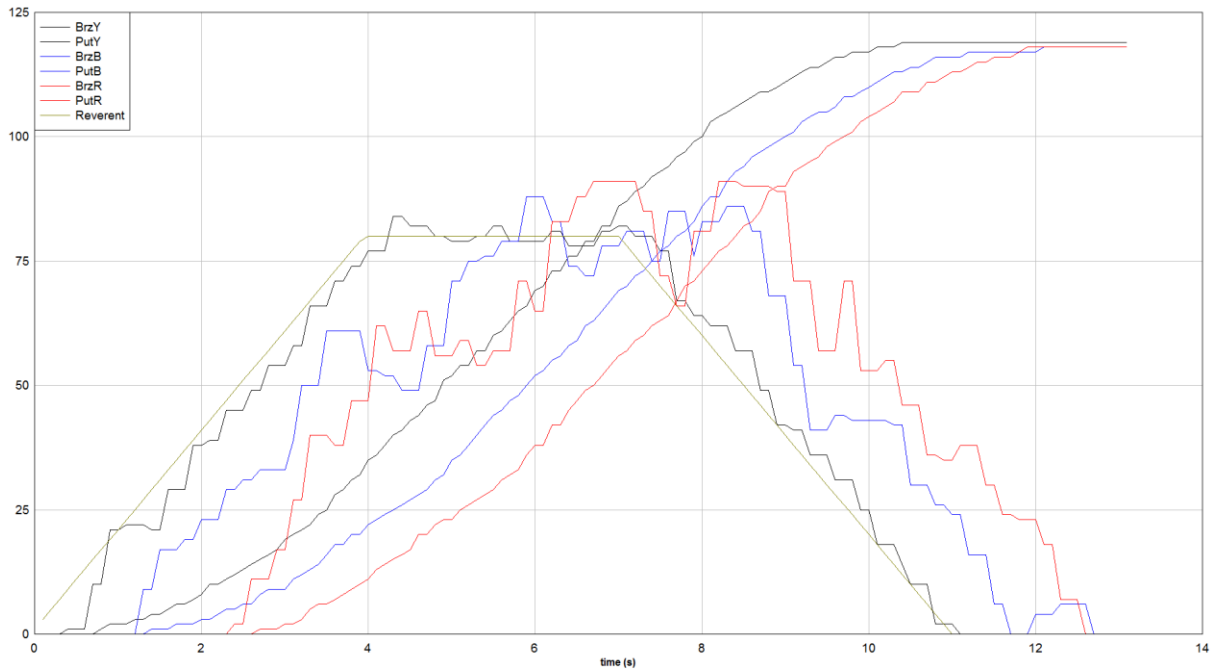
pri čemu se brzina vodećeg vozila v_1^0 regulirana P regulatorom, kao i u prijašnjim algoritmima, prema referentnom profilu brzine. Robot iza vodećeg regulira svoju brzinu prema ulaznoj vrijednosti prvog robota, te se izlazna vrijednost njegovog regulatora šalje zadnjem robotu, koji prema njoj regulira svoju brzinu. Struktura takvog upravljanja prikazana je na slici 33. U ovom se slučaju također može odabrati reguliranje formacije robota ovisno o uključenim senzorima.



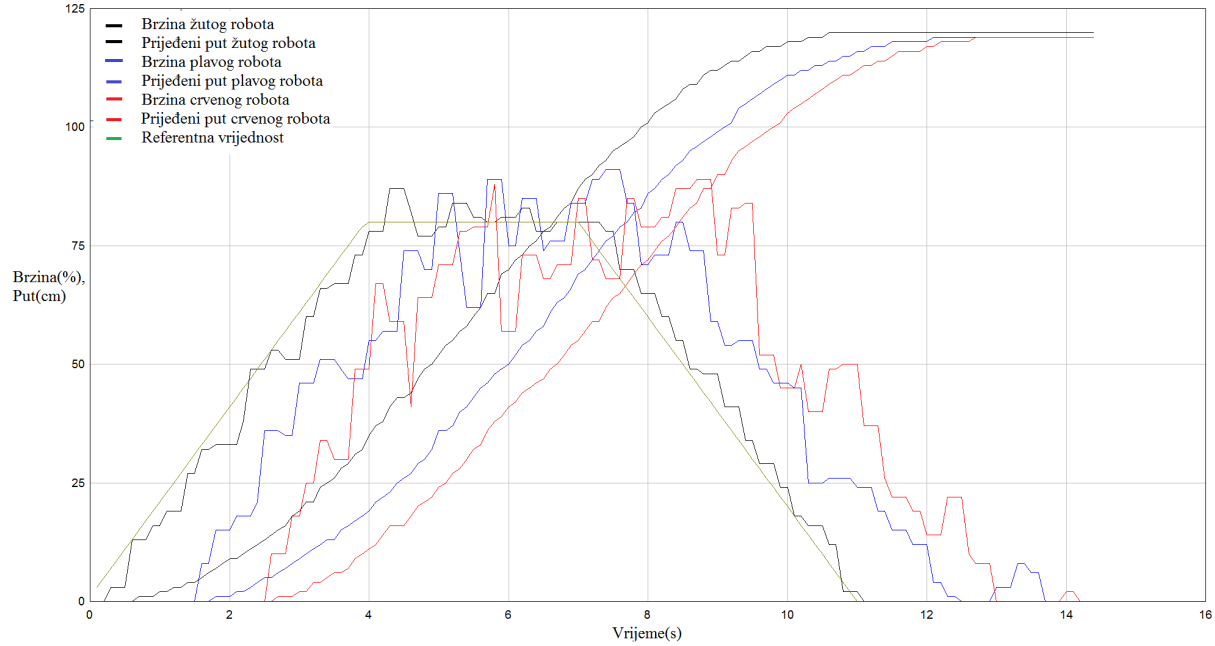
Slika 33. Decentralizirano upravljanje kolonom mobilnih robota s regulacijom razmaka između robota i vezom s reguliranom vrijednošću brzine prethodnika

Izborom ovih načina upravljanja linearnom formacijom mobilnih robota dobiveni su grafovi na slikama 34. I 35., pri čemu su vrijednosti pojačanja iznosila $f_2=b_2=3$, $f_3=2,8$ i $g_2=g_3=0,1$.

Tako je na prvoj slici prikazano vođenje formacije s regulacijom pogreške udaljenosti s obzirom na prethodnika ($b_2=0$), dok je na drugoj slici uključen i stražnji senzor, čime se reguliraju razmaci i prethodnika i sljedbenika. Kako je vidljivo iz grafova, ovom se metodom postiže bolje upravljanje formacijom, nego samo metodom reguliranja pogreške razmaka između prethodnih robota u formaciji s regulacijom apsolutne pogreške brzine, kao što je i u prethodnom slučaju.

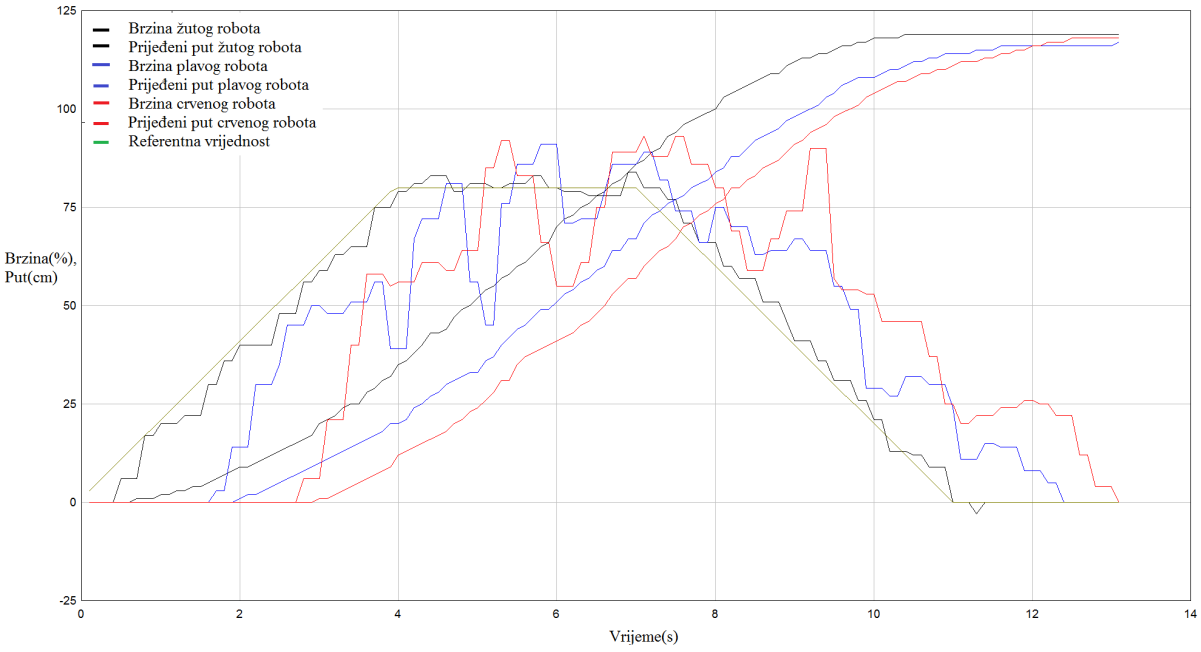


Slika 34. Distribuirano upravljanje kolonom mobilnih robota s regulacijom pogreške razmaka prethodnika i regulacijom brzine s referentnom ulaznom vrijednošću izlaza regulatora prethodnika



Slika 35. Distribuirano upravljanje kolonom mobilnih robota s dvosmjernom regulacijom pogreške udaljenosti robota i regulacijom brzine s referentnom ulaznom vrijednošću izlaza regulatora prethodnika; $f_2=b_2=3$, $f_3=2,8$ i $g_2=g_3=0,1$.

Učinkovitost ovog načina upravljanja linearnom formacijom bitno ovisi o vrijednostima pojačanja lokalnog regulatora mobilnog robota. Radi usporedbe, na slici 36. Prikazan je graf brzine i puta linearne formacije, gdje su vrijednosti pojačanja $f_2=b_2=4$, $f_3=3,8$ i $g_2=g_3=0,3$. Vidljivo je kako se s ovim pojačanjima bolje regulira razmak u usporedbi s grafom na slici 35., ali dolazi do većih oscilacija brzina plavog i crvenog robota.



Slika 36. Distribuirano upravljanje kolonom mobilnih robota s dvosmjernom regulacijom pogreške udaljenosti robota i regulacijom brzine s referentnom ulaznom vrijednošću izlaza regulatora prethodnika; $f_2=b_2=4$, $f_3=3,8$ i $g_2=g_3=0,3$

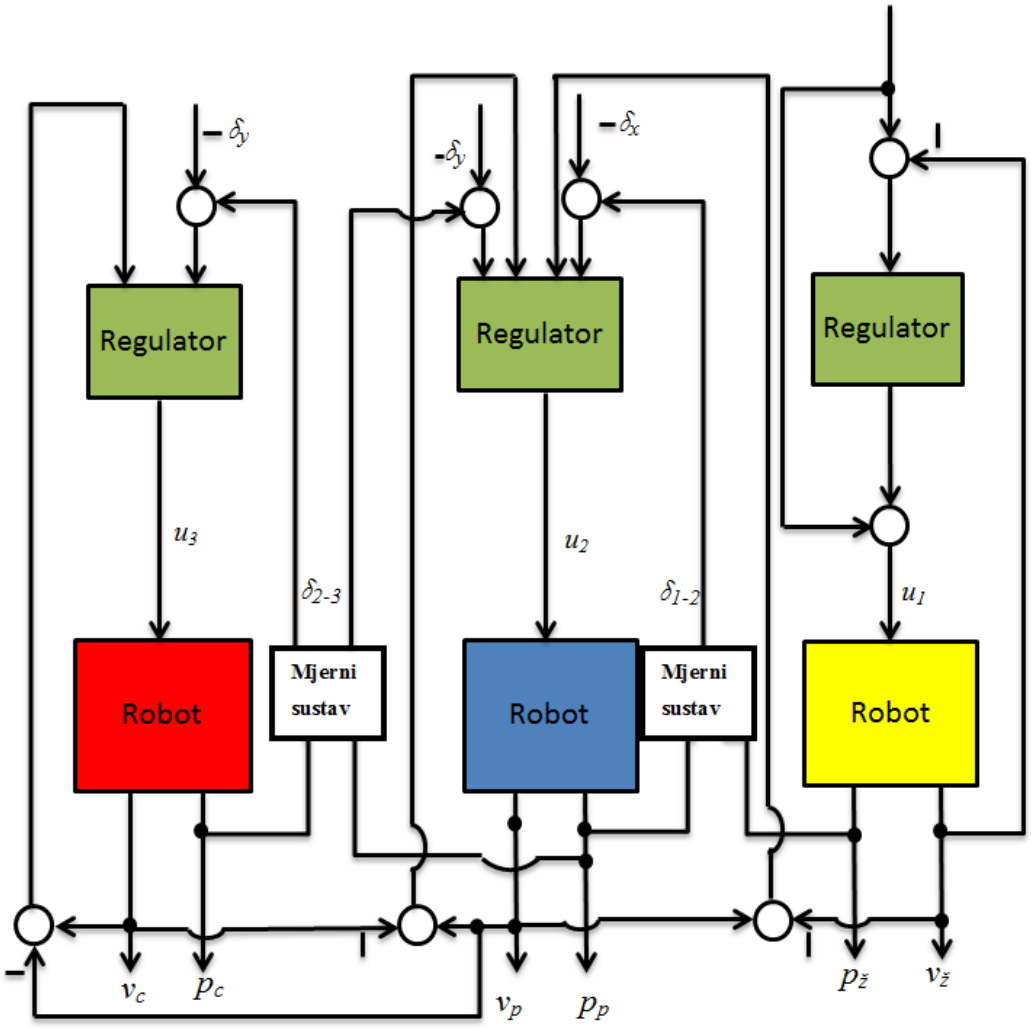
5.6 Decentralizirano upravljanje linearnom formacijom mobilnih robota s bidirekcijskom regulacijom brzine i udaljenosti

U ovom zadnjem slučaju promatrat će se decentralizirano upravljanje s bidirekcionalnom vezom između mobilnih robota u formaciji. Ulazne vrijednosti koje prima regulator su odstupanja od referentne udaljenosti između robota i razlika brzina u odnosu na susjedne robote. Time bi se regulirane ulazne vrijednosti mogle zapisati kao:

$$u_n = -f_n\left(\delta_{n-(n-1)} - \delta_x\right) - b_n\left(-\delta_{n-(n+1)} + \delta_y\right) - g_n\left(v_n - v_{n-1}\right) - k_n\left(v_n - v_{n+1}\right), \quad (22)$$

gdje su f_n i b_n unaprijedna i unazadna pojačanja pogrešaka razmaka između robota ispred i iza, a g_n i k_n unaprijedna i unazadna pojačanja pogreške brzine robota v_n u odnosu na prethodnika v_{n-1} i sljedbenika v_{n+1}. Shema ovakvog načina upravljanja prikazana je na slici

37. U ovoj programskoj aplikaciji, koriste se simetrična pojačanja, te će se vodeći mobilni robot kretati po referentnom profilu brzine.



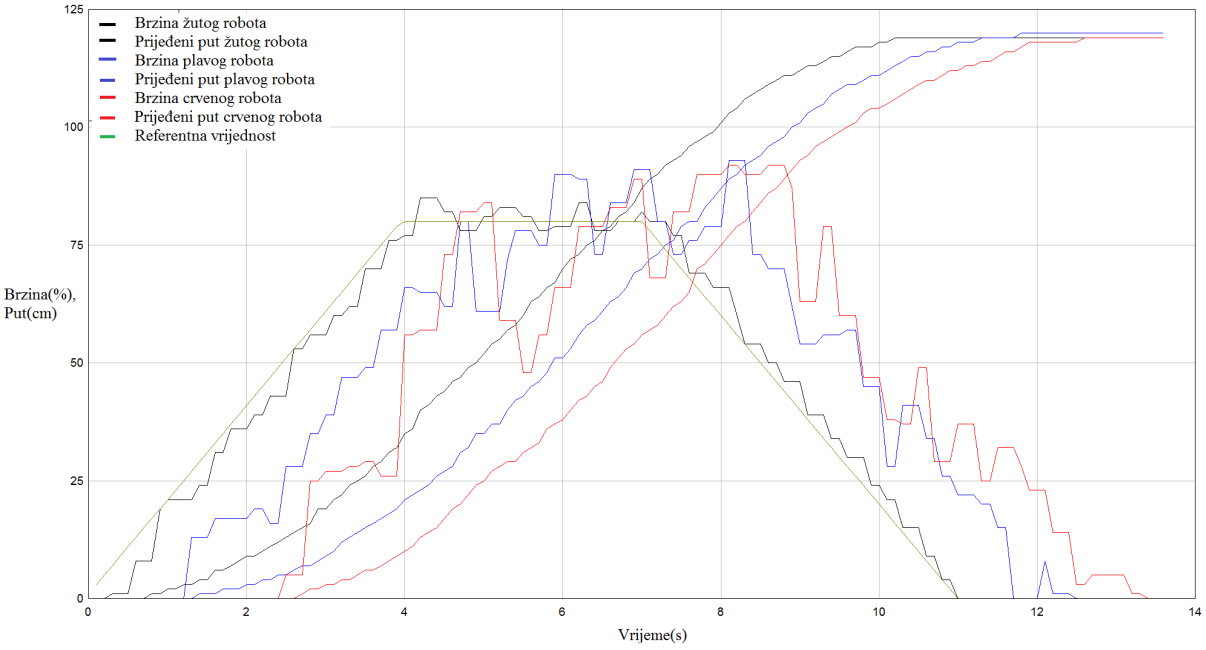
Slika 37. Struktura decentraliziranog bidirekcijskog upravljanja s regulacijom brzine i razmaka između mobilnih robota

Vrijednosti koeficijenata primijenjenih u aplikaciji izračunate su kao u [9], odnosno :

$$f_n = b_n = 1 + 0,2\left(2\pi(\mathrm{N}-\mathrm{n})/\mathrm{N}\right), \tag{23}$$

$$g_n = k_n = 0,5 + 0,2\left(2\pi(\mathrm{N}-\mathrm{n})/\mathrm{N}\right), \tag{24}$$

gdje je N broj robota u formaciji. Tim načinom upravljanja dobiveni rezultati brzine gibanja i prijeđenog puta prikazani su u grafu na slici 38. U njemu se vidi kako zadnji, crveni mobilni robot više puta smanjuje, tj. regulira svoju brzinu u odnosu na brzinu i razmak prema plavom robotu.



Slika 38. Graf brzine i puta decentraliziranog bidirekcijskog upravljanja formacije robota s regulacijom brzine i razmaka između mobilnih robota

6 ZAKLJUČAK

U ovom su radu prikazani neki od različitih pristupa s kojima se može upravljati linearnom formacijom robota. Za svaki navedeni algoritam napisan je kod u aplikaciji za linearno upravljanje eMIR mobilnih robota, koji se može vidjeti u prilogu rada. Rezultati dobiveni primjenom programske aplikacije opisuju različita ponašanja formacije robota, ovisno o odabranom algoritmu upravljanja, te je iz dobivenih grafova moguće usporediti učinkovitosti gibanja različitih načina upravljanja formacijom. Iz eksperimentalnog dijela može se zaključiti kako se najbolje upravljanje postiže korištenjem modela s jednostrukim integratorom gdje se regulira razmak između mobilnih robota i gdje je robotima poznata referentna brzina gibanja (koja im se unaprijedno dodaje). Također, vrlo dobri rezultati postižu se kada se na sličan način sljedbenom robotu umjesto referentne brzine dodaje izlazna vrijednost lokalnog regulatora prethodnog robota. Promatranjem ostalih pristupa vožnje u formaciji, vidljivo je kašnjenje sljedbenih robota za vodećim robotom, pri tome da se bidirekcijskim upravljanjem, s uključenim senzorima u oba smjera, postiže nešto lošije upravljanje formacijom, zbog ujednačavanja razmaka između prethodnika i sljedbenika. U tom slučaju, ovisno o lokalnim regulatorima, robotima u koloni je pri zaustavljanju potrebno i više od 2 sekunde kako bi se točno pozicionirali.

Prilikom niza eksperimenata zaključeno je da ponovljivost i točnost rezultata bitno ovise o stanju baterija mobilnih robota. Rezultati mogu bitno varirati kada se rade mjerenja s punim i praznim baterijama, te je stoga poželjno koristiti mobilne robote s istim naponom u bateriji. Uz to, na ponašanje kolone mobilnih robota, prilikom odabira većih vrijednosti pojačanja, bitno utječu početni uvjeti, odnosno početni razmaci između robota. Ponašanje robota bitno ovisi i o podacima dobivenih sa senzora, koji znaju često varirati, što je i vidljivo pri samom uključivanju robota. Stoga se pojačanja podešavaju kako bi što bolje regulirala pogrešku razmaka, te kako ne bi dolazilo do nestabilnosti.

LITERATURA

[1] S. van de Hoef, K. H. Johansson, D. V. Dimarogonas : Fuel-Optimal Centralized Coordination of Truck Platooning Based on Shortest Paths, American Control Conference. Chicago, str. 2740-3745 SAD, 2015.

[2] A. Alam:Fuel-efficient heavy-duty vehicle platooning, doktorski rad, KTH, Stockholm, 2014.

[3] S. Tsugawa, S. Kato, K. Aoki :An Automated Truck Platoon for Energy Saving, IEEE/RSJ konferencija o inteligentnim robotskim sistemima, str. 4109-4114, San Francisco, 2011.

[4] F. Lin, M. Fardad, M. R. Jovanović: Opimal Control of vehicular Formations With Nearest Neighbour Interactions, IEEE Trans. Intell. Transp. Syst. Vol.57 NO 9., 2012, str: 2203-2218.

[5] G. Guo, W. Yue: Autonomous Platoon Control Allowing Range-Limited Sensors, IEE transactions on vehicular technology, vol. 61. No. 7., 2012., str. 2901-2912

[6] P. Barooah, P. G. Mehta, J. P. Hespanha: Mistuning-based control design to improve closed-loop stability margin of vehicular platoons, IEEE Trans. Autom. Control, 2009, Vol 54, str.2100 -2113

[7] S.E. Li, Y. Zheng, K. Li, J Wang: An Overview of Vehicular Platoon Control under the Four-Component Framework, IEEE Intelligent Vehicles Symposium, Seoul, str 286- 291.

[8] E. Shaw, J. Hedrick: String stability analysis for heterogeneous vehicle strings, New York, 2007, str.3118 -3125.

[9] H. Hao, P. Barooah: Control of large 1D networks of double integrator agents: role of heterogeneity and asymmetry on stability margin, IEEE Conference, 2010, Atlanta str. 7395-7400.

[10] B. Bamieh, M. R. Jovanović, P. Mitra, S Patterson: Coherence in Large-Scale Networks: Dimension-Dependent Limitations of Loal Feedback. IEEE Trans. Intell. Transp. Syst. Vol.57 NO 9., 2012, str. 2235-2249

[11] Y. Zheng, S. Li, J. Wang, D. Cao, K. Li: Stability and Scalability of Homogeneous Vehicular Platoon: Study on Influence of Information Flow Topologies, IEEE Trans.

Intell. Transp. Syst.Vol. 17, no. 1, 2015.

[12] M. Jovanović: Design of structured optimal feedback gains for interconnected systems, Minnesota, 2010.

[13] S. Stanković, M. Stanojević, D. Šiljak: Decentralized overlapping control of a platoon of vehicles, IEEE Trans. Contr. Syst. Technol.,Vol. 8, no. 5, 2000, str.816 -831.

[14] J. Zhou, H. Peng: Range policy of adaptive cruise control vehicle for improved flow stability and string stability, IEEE Trans. Intell. Transp. Syst. Vol. 6, no. 2, 2005., str.229 -237.

[15] L. Xiao, F. Cao: Practical string stability of platoon of adaptive cruise control vehicles, IEEE Trans. Intell. Transp. Syst., Vol 12, no. 4, 2010, str. 1184 -1194.

[16] R. Kianfar, B. Augusto, A. Ebadighajari: Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge, IEEE Trans. Intell. Transp. Syst., Vol 13,no. 3, 2012 str. 994-1007.

[17] G. Guo, W. Yue, "Hierarchical platoon control with heterogeneous information feedback",IET Control Theory Appl., 2011, pp.1766 -1781.

[18] K. Santhanakrishnan, R. Rajamani: On spacing policies for highway vehicle automation, IEEE Trans. Intell. Transp. Syst.,Vol. 4,no. 4, 2003, str.198 -204

[19] B. Bamieh, M. R. Jovanović, P. Mitra, S Patterson: Coherence in Large-Scale Networks: Dimension-Depandent Limitations of Loal Feedback. IEEE Trans. Intell. Transp. Syst. Vol.57 NO 9., 2012, str: 2235-2249

[20] P. Barooah, J.P. Hespanha: Error amplification and disturbance propagation in vehicle strings with decentralized linear control, Proc. IEEE Conf. Decision Control, 2005, str. 4964-4969.

[21] Y.Zheng, S.Eben Li, K. Li, L-Y. Wang: Stability Margin Improvement of Vehicular Platoon Considering Undirected Topology and Asymmetric Control, IEEE Trans. Intell. Transp. Syst.,Vol 24, no.4, 2016. Str 1253-1265..

[22] Laboratorijski postav, Mobilni roboti, Fakultet strojarstva i brodogradnje

[23] M.Lukas: Diplomski rad, 2015

[24] internetski izvor: https://www.embarcadero.com/products/delphi 18.6.2017.

[25] Internetski izvor: izvor: https://en.wikipedia.org/wiki/Delphi_(programming_language) 18.6.2017.

PRILOZI

I. CD-R disc

II. Kod programa

Kod programa aplikacije za upravljanje kolonom od 3 eMIR robota:

```
////////////////////////////////////////////////////////////////////////////////////////////////
unit SviRoboti;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
  Vcl.Samples.Spin, Joystick;

Procedure StopRecording;
Procedure StartRecording(S : string; VarMask : string);

type
  TGlavniProzor = class(TForm)

    btnZuti: TButton;
    btnCrveni: TButton;
    btnPlavi: TButton;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    seRazmak: TSpinEdit;
    Label2: TLabel;
    btnStop: TButton;
    btnStartjoystick: TButton;
    Joystick: TJoystick;
    Timer1: TTimer;
    lblStatus: TLabel;
    Tjoystick: TTimer;
    btnPozicioniranje: TButton;
    Pozicija: TTimer;
    Gibanje: TButton;
    GibanjeCent: TTimer;
```

```
  btnDistribBIsvodecomvelicinom: TButton;
  gibanje2: TTimer;
  lblPutZuti: TLabel;
  lblPutPlavog: TLabel;
  lblPutCrvenog: TLabel;
  lblRazmak12: TLabel;
  lblRazmak23: TLabel;
  TJtimer2: TTimer;
  btnGibanjesvodecim: TButton;
  Gibsvod: TTimer;
  btnDecentraliziranoPF: TButton;
  DecentraliziranoPF: TTimer;
  DecentraliziranoBD: TTimer;
  btnDecentraliziranoBD: TButton;
  Label1: TLabel;
  seRazmak23: TSpinEdit;
  Button1: TButton;
  Timer2: TTimer;
  Button2: TButton;
  Timer3: TTimer;
  Timer4: TTimer;
  Button3: TButton;
  Button4: TButton;
  Button5: TButton;
  Timer5: TTimer;
  Timer6: TTimer;
  Button6: TButton;
  Timer7: TTimer;
  lblbrzinacrveni: TLabel;
  lblbrzinaplavi: TLabel;
  lblbrzinazuti: TLabel;
  procedure btnZutiClick(Sender: TObject);
  procedure btnCrveniClick(Sender: TObject);
  procedure btnPlaviClick(Sender: TObject);
  procedure btnStopClick(Sender: TObject);
  procedure btnStartjoystickClick(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure TjoystickTimer(Sender: TObject);
  procedure PozicijaTimer(Sender: TObject);
  procedure btnPozicioniranjeClick(Sender: TObject);
  procedure GibanjeClick(Sender: TObject);
  procedure GibanjeCentTimer(Sender: TObject);
  procedure btnDistribBIsvodecomvelicinomClick(Sender: TObject);
  procedure gibanje2Timer(Sender: TObject);
  procedure TJtimer2Timer(Sender: TObject);
  procedure btnGibanjesvodecimClick(Sender: TObject);
```

```
    procedure GibsvodTimer(Sender: TObject);
    procedure btnDecentraliziranoPFClick(Sender: TObject);
    procedure DecentraliziranoPFTimer(Sender: TObject);
    procedure btnDecentraliziranoBDClick(Sender: TObject);
    procedure DecentraliziranoBDTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Timer4Timer(Sender: TObject);
    procedure Timer5Timer(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Timer6Timer(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Timer7Timer(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

const DeadJoystickPosition= 3;

var
  GlavniProzor    : TGlavniProzor;
  Stop            : Boolean;
  CommandTimeOut  : word;
  WaitTime        : word;
  JoystickConnected:boolean;
  Pozz      : boolean;
  TaskM     : boolean;
  Referenca : integer;
  TimerA        : word;
  TaskTime      : Word;
  CommandTime   : word;
  RecordTime    : word;
  RecFile       : TextFile;
  MASK          : string;
  Brzina        :integer;
  aa            :integer;
  zz            :integer;
  kutz          :integer;
  kutp          :integer;
  kutc          :integer;
```

```
  ed            :integer;
implementation

{$R *.dfm}
uses
Unit_3eMIR;

///-----------------------------------------------------------------------------------------
 Procedure Wait(tt : word);
begin
    WaitTime:=tt;
    WHILE WaitTime > 0 DO Application.ProcessMessages;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnCrveniClick(Sender: TObject);
begin
     RedRobot.Connect;
    if RedRobot.Connected then btnCrveni.Visible:=False;
    Panel2.ParentColor:=False;
end;
procedure TGlavniProzor.btnDecentraliziranoBDClick(Sender: TObject);
begin
       YellowRobot.ResetCounters;
       BlueRobot.ResetCounters;
       RedRobot.ResetCounters;
       Wait(5);
       Brzina:=1;
       aa:=0;
       Kutz:=0;
       Kutp:=0;
       kutc:=0;
       ed:=0;
       DecentraliziranoBD.Enabled:=true;
       StartRecording('Decentralizirano upravljanje s bidirekcijskim tokom informacija',
'1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnDecentraliziranoPFClick(Sender: TObject);
begin
       YellowRobot.ResetCounters;
       BlueRobot.ResetCounters;
       RedRobot.ResetCounters;
       Wait(5);
       Brzina:=1;
       aa:=0;
       Kutz:=0;
       Kutp:=0;
```

```
      kutc:=0;
      ed:=0;
      DecentraliziranoPF.Enabled:=true;
      StartRecording('Decentralizirano upravljanje s look ahead strategijom',
'1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnDistribBIsvodecomvelicinomClick(Sender: TObject);
begin
      YellowRobot.ResetCounters;
      BlueRobot.ResetCounters;
      RedRobot.ResetCounters;
      Brzina:=1;
      aa:=0;
      Kutz:=0;
      Kutp:=0;
      kutc:=0;
      Wait(5);
    gibanje2.Enabled:=true;
    StartRecording('BI s referentnom brzinom', '1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnGibanjesvodecimClick(Sender: TObject);
begin
       YellowRobot.ResetCounters;
      BlueRobot.ResetCounters;
      RedRobot.ResetCounters;
      Wait(5);
      Brzina:=1;
      aa:=0;
      Kutz:=0;
      Kutp:=0;
      kutc:=0;
    Gibsvod.Enabled:=true;
    StartRecording('PF s referentnom brzinom regulatora sljedbenika', '1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnPlaviClick(Sender: TObject);
begin
   BlueRobot.Connect;
   if BlueRobot.Connected then btnplavi.Visible:=False;
   Panel3.ParentColor:=False;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnZutiClick(Sender: TObject);
begin
   YellowRobot.Connect;
```

```
    if yellowRobot.Connected then btnZuti.Visible:=False;
    Panel1.ParentColor:=False;
end;
procedure TGlavniProzor.Button1Click(Sender: TObject);
begin
       YellowRobot.ResetCounters;
       BlueRobot.ResetCounters;
       RedRobot.ResetCounters;
       Wait(5);
       Brzina:=1;
       aa:=0;
       Kutz:=0;
       Kutp:=0;
       kutc:=0;
       Timer2.Enabled:=true;
       StartRecording('PF s informacijom o razmaku prethodnika te referentnom brzinom izlaza iz
regulatora prethodnika', '1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Button2Click(Sender: TObject);
begin
  YellowRobot.ResetCounters;
       BlueRobot.ResetCounters;
       RedRobot.ResetCounters;
       Wait(5);
       Brzina:=1;
       aa:=0;
       Kutz:=0;
       Kutp:=0;
       kutc:=0;
       Timer2.Enabled:=true;
       StartRecording('Decentralizirano upravljanje s Bidirekciskom regulacijom brzine i udaljenosti',
'1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Button3Click(Sender: TObject);
begin
YellowRobot.ResetCounters;
       BlueRobot.ResetCounters;
       RedRobot.ResetCounters;
       Wait(5);
       Brzina:=1;
       aa:=0;
       Kutz:=0;
       Kutp:=0;
       kutc:=0;
    timer4.Enabled:=true;
```

```
    StartRecording('PF s referentnom brzinom regulatora sljedbenika', '1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Button4Click(Sender: TObject);
begin
      YellowRobot.ResetCounters;
      BlueRobot.ResetCounters;
      RedRobot.ResetCounters;
      Wait(5);
      Brzina:=1;
      aa:=0;
      Kutz:=0;
      Kutp:=0;
      kutc:=0;
    timer5.Enabled:=true;
    StartRecording('PF s regulacijom udaljenosti i apsolutne greške brzine', '1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Button5Click(Sender: TObject);
begin
      YellowRobot.ResetCounters;
      BlueRobot.ResetCounters;
      RedRobot.ResetCounters;
      Wait(5);
      Brzina:=1;
      aa:=0;
      Kutz:=0;
      Kutp:=0;
      kutc:=0;
    timer6.Enabled:=true;
    StartRecording('BI s regulacijom udaljenosti i apsolutne greške brzine', '1100011000110001000');
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Button6Click(Sender: TObject);
begin
      YellowRobot.ResetCounters;
      BlueRobot.ResetCounters;
      RedRobot.ResetCounters;
      Wait(5);
      Brzina:=1;
      aa:=0;
      Kutz:=0;
      Kutp:=0;
      kutc:=0;
      Timer2.Enabled:=true;
      StartRecording('BI s informacijom o razmaku prethodnika te referentnom brzinom izlaza iz
regulatora prethodnika', '1100011000110001000');
```

```
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.DecentraliziranoBDTimer(Sender: TObject);
var
Razmak12,Razmak23,e,y :integer;
b,s,Reg:double;
begin
  TaskM:=True;
    Razmak12:=SeRazmak.Value;
    Razmak23:=seRazmak.Value;
    StopTask:=false;
    if stopTask then  exit;
     Brzina:=Brzina+2;
     inc(aa);
     if (aa >=60) then Brzina:=Brzina-4;
     if (Brzina > 80) then  Brzina:=80;
     if (Brzina<0) then Brzina:=0;

     e:=(Brzina-YellowRobot.Speed);
     Reg:=((0.45*e)+(0.03*(e-ed))+Brzina);
     ed:=e;
     y:=Round(reg);
     if (y > 80)  then    y:=80;
     if (y < 4) and (y > -4) then  y:=0;
     YellowRobot.Move(y,-kutz);
     kutz:=Yellowrobot.Angle;
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
       b:=(3.6*(BlueRobot.S0-razmak12))+(3.6*(-Bluerobot.S5+razmak23));
       if (b > 90)  then    b:=90;
       if (b < 4) and (b > -4) then   b:=0;
       BlueRobot.Move(Round(b),-kutp);
       kutp:=bluerobot.Angle;
          s:=(3.6*(Redrobot.S0-Razmak23));
          if (s>90) then  s:=90;
          if (s < 4) and (s > -4) then    s:=0;
          RedRobot.Move(Round(s),-kutc);
          Kutc:=Redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.DecentraliziranoPFTimer(Sender: TObject);
var
Razmak12,Razmak23,e :integer;
b,s,Reg:double;
begin
  TaskM:=True;
    Razmak12:=SeRazmak.Value;
    Razmak23:=seRazmak.Value;
```

```
    StopTask:=false;
    if stopTask then  exit;
     Brzina:=Brzina+2;
     inc(aa);
     if (aa >=60) then Brzina:=Brzina-4;
     if (Brzina > 80) then  Brzina:=80;

     e:=(Brzina-YellowRobot.Speed);
     Reg:=((0.45*e)+(0.03*(e-ed))+Brzina);
     ed:=e;
     if brzina>0 then YellowRobot.Move(Round(reg),-kutz);
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
     kutz:=Yellowrobot.Angle;
       b:=(3.6*(BlueRobot.S0-razmak12));
       if (b > 90)  then    b:=90;
       if (b < 4) and (b > -4) then   b:=0;
       BlueRobot.Move(Round(b),-kutp);
       kutp:=bluerobot.Angle;
         s:=(3.6*(Redrobot.S0-Razmak23));
         if (s>90) then  s:=90;
         if (s < 4) and (s > -4) then    s:=0;
         RedRobot.Move(Round(s),-kutc);
         Kutc:=Redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnPozicioniranjeClick(Sender: TObject);
begin
    Pozicija.Enabled:=True;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnStartjoystickClick(Sender: TObject);
begin
      Pozz:=false;
      YellowRobot.ResetCounters;
      RedRobot.ResetCounters;
      blueRobot.ResetCounters;
      Wait(5);
      StartRecording('gibanje1','1100011000110000000');
      Timer1.Enabled:=true;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.btnStopClick(Sender: TObject);
begin
      Stop:=True;
      Timer1.Enabled:=False;
      Pozz:=true;
      GibanjeCent.Enabled:=false;
```

```
      Pozicija.Enabled:=false;
      Gibanje2.Enabled:=False;
      Gibsvod.Enabled:=false;
      DecentraliziranoPF.Enabled:=false;
      DecentraliziranoBD.Enabled:=false;
      Timer2.Enabled:=false;
      Timer3.Enabled:=false;
      timer4.Enabled:=false;
      timer5.Enabled:=false;
      timer6.Enabled:=false;
      timer7.Enabled:=false;
      StopRecording;
end;

procedure TGlavniProzor.GibanjeClick(Sender: TObject);
begin
       YellowRobot.ResetCounters;
       BlueRobot.ResetCounters;
       RedRobot.ResetCounters;
       Brzina:=1;
       aa:=0;
       Kutz:=0;
       Kutp:=0;
       kutc:=0;
       Wait(5);
       GibanjeCent.Enabled:=True;
       StartRecording('PF s referentnom brzinom','1100011000110001100')
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.GibsvodTimer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy:integer;
b,s,Regy:double;
begin
  TaskM:=True;
    Razmak12:=seRazmak.Value;
    Razmak23:=seRazmak23.Value;
    StopTask:=false;
    if stopTask then  exit;
     Brzina:=Brzina+2;
     inc(aa);
     if (aa >=60) then Brzina:=Brzina-4;
     if (Brzina > 80) then  Brzina:=80;
     if (Brzina<0) then Brzina:=0;

      ey:=(Brzina-YellowRobot.Speed);
      RegY:=((0.45*ey)+{(0.03*(ey-edy))}+Brzina);
```

```
      edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
     kutz:=yellowrobot.angle;
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
       b:=(3.2*(BlueRobot.S0-razmak12))+ regy;
       if (b > 90)  then    b:=90;
       if (b < 5) and (b > -5) then   b:=0;
       BlueRobot.Move(Round(b),-kutp);
       kutp:=bluerobot.Angle;
         s:=(3*(Redrobot.S0-Razmak23)) + b;
         if (s>90) then  s:=90;
         if (s < 5) and (s > -5) then    s:=0;
         RedRobot.Move(Round(s),-kutc);
         kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.FormCreate(Sender: TObject);
var
X: word;
begin
    Joystick.Active:=True;
    JoystickConnected:=True;
    try
      x:= Joystick.PositionX;
      except
       joystickConnected:=False;
    end;
    if joystickConnected then lblStatus.Caption:='Joystick je spojen'
    else lblStatus.Caption:='Joystick nije spojen';
    if joystickConnected then Tjoystick.Enabled:=true;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.gibanje2Timer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
C,B,RegY,RegB,RegR : double;
begin
     TaskM:=true;
     if (aa <70)then Brzina:=Brzina+2;
     aa:=aa+1;
     Razmak12:=seRazmak.Value;
     Razmak23:=seRazmak23.Value;
     if (aa >=70) then Brzina:=Brzina-2;
     if (Brzina > 80) then  Brzina:=80;
     if (Brzina<0) then Brzina:=0;

      ey:=(Brzina-YellowRobot.Speed);
```

```
      RegY:=((0.45*ey){+(0.03*(ey-edy))}+Brzina);
      edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
     kutz:=Yellowrobot.Angle;
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
       B:=(2.8*(BlueRobot.S0-Razmak12))+(-2.8*(BlueRobot.S5+Razmak23))+ brzina;
       if (B > 90)  then    B:=90;
       if (B < 4) and (B > -4) then   B:=0;
       BlueRobot.Move(Round(B),-kutp);
       kutp:=bluerobot.Angle;
         C:=(2.6*(RedRobot.S0 - razmak23)) + brzina;
         if (C > 90)  then    C:=90;
         if (C < 4) and (C > -4) then   C:=0;
         RedRobot.Move(Round(C),-kutc);
         Kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.GibanjeCentTimer(Sender: TObject);
Var
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
s,b,z,RegY,RegB,RegR : double;
begin
     TaskM:=true;
     if (aa <70)then Brzina:=Brzina+2;
     aa:=aa+1;
     Razmak12:=seRazmak.Value;
     Razmak23:=seRazmak23.Value;
     if (aa >=70) then Brzina:=Brzina-2;
     if (Brzina > 80) then  Brzina:=80;
     if (Brzina<0) then Brzina:=0;

     ey:=(Brzina-YellowRobot.Speed);
     RegY:=((0.45*ey)+{(0.02*(ey-edy))}+Brzina);
     edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
     kutz:=Yellowrobot.Angle;
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
      b:=(3.6*(BlueRobot.S0-razmak12))+Brzina;
      if (b > 100)  then    b:=100;
      if (b < 5) and (b > -5) then   b:=0;
      BlueRobot.Move(round(b),-kutp);
      kutp:=bluerobot.Angle;
       s:=(3.4*(Redrobot.S0-Razmak23)) + b;
       if (s>100) then  s:=100;
       if (s < 5) and (s > -5) then    s:=0;
       RedRobot.Move(Round(s),-kutc);
       Kutc:=redrobot.Angle;
```

```
end;
///-----------------------------------------------------------------------------------------
  ///// Pozicioniranje
procedure TGlavniProzor.PozicijaTimer(Sender: TObject);
var
Razmak12,Razmak23,y,b,r:integer;
begin
     pozz:=false;
     Razmak12:=SeRazmak.Value;
     Razmak23:=seRazmak23.Value;
     if BlueRobot.Connected  and YellowRobot.Connected  and RedRobot.Connected then
     Begin
       b:=(3*(BlueRobot.S0-razmak12));
       if (b > 80)  then    b:=80;
     if (b < 3) and (b > -3) then   b:=0;
      BlueRobot.Move(b,0);
      R:=3*(RedRobot.S0 - Razmak23);
      if (r > 80)  then    r:=80;
     if (r < 3) and (r > -3) then   r:=0;
     RedRobot.Move(r,0);
     End;
     if  (Abs(BlueRobot.S0-razmak12) < 2)  and  (Abs(RedRobot.S0 - Razmak23) < 2) then
     Pozicija.Enabled:=False;
end;
 ///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Timer1Timer(Sender: TObject);
var
Razmak12,Rotacija:integer;
Brzina,b,s:double;
begin
     Razmak12:=SeRazmak.Value;
    if BlueRobot.Connected  and YellowRobot.Connected  and RedRobot.Connected
                        {and (YellowRobot.S0>20) and (YellowRobot.S2>20)
                        and (RedRobot.S2>20)   and (BlueRobot.S2>20)}  then
     Begin
     if joButton1 IN Joystick.PressedButtons  then
      begin
        Brzina := Round((32767-Joystick.PositionY)/327.67);   // -100 .. +100 %
        Rotacija:= Round((32767-Joystick.PositionX)/327.67);   // -100 .. +100 %
        if Abs(Brzina)    <= DeadJoystickPosition then Brzina:=0;
        if Abs(Rotacija) <= DeadJoystickPosition then Rotacija:=0;
        end
       else
        begin Brzina:=0;  Rotacija:=0; end;
        YellowRobot.Move(Round(Brzina),0);
     b:=(4*(BlueRobot.S0-razmak12)) {- (4*(-BlueRobot.S5 + Razmak))} + Brzina;
     if (b > 100)  then    b:=100;
```

```
     if (b < 5) and (b > -5) then   b:=0;
      BlueRobot.Move(Round(b),0);
    s:=(4*(Redrobot.S0-Razmak12)) + Brzina;
    if (s>100) then  s:=100;
    if (s < 5) and (s > -5) then    s:=0;
      RedRobot.Move(Round(s),0);
      end;
end;
///-----------------------------------------------------------------------------------------
 procedure TGlavniProzor.Timer2Timer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
C,B,RegY,RegB,RegR : double;
begin
     TaskM:=true;
     if (aa <70)then Brzina:=Brzina+2;
     aa:=aa+1;
     Razmak12:=seRazmak.Value;
     Razmak23:=seRazmak23.Value;
     if (aa >=70) then Brzina:=Brzina-2;
     if (Brzina > 80) then  Brzina:=80;
     if (Brzina<0) then Brzina:=0;

      ey:=(Brzina-YellowRobot.Speed);
      RegY:=((0.45*ey){+(0.03*(ey-edy))}+Brzina);
      edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
     Kutz:=YellowRobot.Angle;
       B:=(3*(BlueRobot.S0-Razmak12))-(0.1*(blueRobot.Speed-regy));
       if (B > 90)  then    B:=90;
       if (B < 4) and (B > -4) then   B:=0;
       BlueRobot.Move(Round(B),-kutp);
       kutp:=BlueRobot.Angle;
         C:=(2.8*(RedRobot.S0 - razmak23))-(0.1*(BlueRobot.Speed-b));
         if (C > 90)  then    C:=90;
         if (C < 4) and (C > -4) then   C:=0;
         RedRobot.Move(Round(C),-kutc);
         kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Timer3Timer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
C,B,RegY,RegB,RegR : double;
begin
     TaskM:=true;
```

```
    if (aa <70)then Brzina:=Brzina+2;
    aa:=aa+1;
    Razmak12:=seRazmak.Value;
    Razmak23:=seRazmak23.Value;
    if (aa >=70) then Brzina:=Brzina-2;
    if (Brzina > 80) then  Brzina:=80;
    if (Brzina<0) then Brzina:=0;

     ey:=(Brzina-YellowRobot.Speed);
     RegY:=((0.45*ey){+(0.03*(ey-edy))}+Brzina);
     edy:=ey;
    if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
    if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
    Kutz:=YellowRobot.Angle;
      B:=(1.7*(BlueRobot.S0-Razmak12))+(1.7*(-BlueRobot.S5+Razmak23))-
(0.58*(blueRobot.Speed-YellowRobot.Speed))-(0.58*(blueRobot.Speed-redrobot.Speed));
      if (B > 90)  then    B:=90;
      if (B < 4) and (B > -4) then   B:=0;
      BlueRobot.Move(Round(B),-kutp);
      kutp:=BlueRobot.Angle;
        C:=(1*(RedRobot.S0 - razmak23))-(1*(BlueRobot.Speed-b));
        if (C > 90)  then    C:=90;
        if (C < 4) and (C > -4) then   C:=0;
        RedRobot.Move(Round(C),-kutc);
        kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------
procedure TGlavniProzor.Timer4Timer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy:integer;
b,s,Regy:double;
begin
  TaskM:=True;
    Razmak12:=seRazmak.Value;
    Razmak23:=seRazmak23.Value;
    StopTask:=false;
    if stopTask then  exit;
     Brzina:=Brzina+2;
     inc(aa);
     if (aa >=60) then Brzina:=Brzina-4;
     if (Brzina > 80) then  Brzina:=80;
     if (Brzina<0) then Brzina:=0;

      ey:=(Brzina-YellowRobot.Speed);
      RegY:=((0.45*ey)+{(0.03*(ey-edy))}+Brzina);
      edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
```

```
    kutz:=yellowrobot.angle;
    if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
     b:=(3.2*(BlueRobot.S0-razmak12))+ (3.2*(-bluerobot.S5+razmak23))+ regy;
     if (b > 90)  then    b:=90;
     if (b < 5) and (b > -5) then   b:=0;
     BlueRobot.Move(Round(b),-kutp);
     kutp:=bluerobot.Angle;
      s:=(3*(Redrobot.S0-Razmak23)) + b;
      if (s>90) then  s:=90;
      if (s < 5) and (s > -5) then    s:=0;
      RedRobot.Move(Round(s),-kutc);
      kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Timer5Timer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
C,B,RegY,RegB,RegR : double;
begin
    TaskM:=true;
    if (aa <70)then Brzina:=Brzina+2;
    aa:=aa+1;
    Razmak12:=seRazmak.Value;
    Razmak23:=seRazmak23.Value;
    if (aa >=70) then Brzina:=Brzina-2;
    if (Brzina > 80) then  Brzina:=80;
    if (Brzina<0) then Brzina:=0;
     ey:=(Brzina-YellowRobot.Speed);
     RegY:=((0.45*ey){+(0.03*(ey-edy))}+Brzina);
     edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
     kutz:=Yellowrobot.Angle;
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
      B:=(3*(BlueRobot.S0-Razmak12))-(0.2*(blueRobot.Speed-Brzina));
      if (B > 90)  then    B:=90;
      if (B < 4) and (B > -4) then   B:=0;
      BlueRobot.Move(Round(B),-kutp);
      kutp:=bluerobot.Angle;
       C:=(2.9*(RedRobot.S0 - razmak23))-(0.25*(RedRobot.Speed-Brzina));
       if (C. > 90)  then    C:=90;
       if (C < 4) and (C > -4) then   C:=0;
       RedRobot.Move(Round(C),-kutc);
       Kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Timer6Timer(Sender: TObject);
var
```

```
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
C,B,RegY,RegB,RegR : double;
begin
    TaskM:=true;
    if (aa <70)then Brzina:=Brzina+2;
    aa:=aa+1;
    Razmak12:=seRazmak.Value;
    Razmak23:=seRazmak23.Value;
    if (aa >=70) then Brzina:=Brzina-2;
    if (Brzina > 80) then  Brzina:=80;
    if (Brzina<0) then Brzina:=0;

     ey:=(Brzina-YellowRobot.Speed);
     RegY:=((0.45*ey){+(0.03*(ey-edy))}+Brzina);
     edy:=ey;
    if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
    kutz:=Yellowrobot.Angle;
    if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
      B:=(3*(BlueRobot.S0-Razmak12))+(3*(-BlueRobot.S5+razmak23))- (0.2*(BlueRobot.Speed-
Brzina));
       if (B > 90)  then    B:=90;
       if (B < 4) and (B > -4) then   B:=0;
       BlueRobot.Move(Round(B),-kutp);
       kutp:=bluerobot.Angle;
         C:=(2.9*(RedRobot.S0 - razmak23))-(0.2*(RedRobot.Speed-Brzina));
         if (C > 90)  then    C:=90;
         if (C < 4) and (C > -4) then   C:=0;
         RedRobot.Move(Round(C),-kutc);
         Kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.Timer7Timer(Sender: TObject);
var
Razmak12,Razmak23,ey,edy,eb,edb,er,edr:integer;
C,B,RegY,RegB,RegR : double;
begin
    TaskM:=true;
    if (aa <70)then Brzina:=Brzina+2;
    aa:=aa+1;
    Razmak12:=seRazmak.Value;
    Razmak23:=seRazmak23.Value;
    if (aa >=70) then Brzina:=Brzina-2;
    if (Brzina > 80) then  Brzina:=80;
    if (Brzina<0) then Brzina:=0;

     ey:=(Brzina-YellowRobot.Speed);
     RegY:=((0.45*ey){+(0.03*(ey-edy))}+Brzina);
```

```
      edy:=ey;
     if brzina>0 then YellowRobot.Move(Round(regy),-kutz);
     if (aa>60) and (Brzina<=0) then  yellowrobot.Move(0,0);
     Kutz:=YellowRobot.Angle;

       B:=(3*(BlueRobot.S0-Razmak12))+(3*(-BlueRobot.S5+Razmak23))-(0.1*(blueRobot.Speed-
regy));
       if (B > 90)  then    B:=90;
       if (B < 4) and (B > -4) then   B:=0;
       BlueRobot.Move(Round(B),-kutp);
       kutp:=BlueRobot.Angle;

         C:=(2.8*(RedRobot.S0 - razmak23))-(0.1*(BlueRobot.Speed-b));
         if (C > 90)  then    C:=90;
         if (C < 4) and (C > -4) then   C:=0;
         RedRobot.Move(Round(C),-kutc);
         kutc:=redrobot.Angle;
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.TjoystickTimer(Sender: TObject);
Var  SetSpeed, SetRotation : integer;
begin
    if NOT yellowRobot.Connected then Exit;
    if TaskM then Exit;
    if joButton1 IN Joystick.PressedButtons  then
      begin
        SetSpeed   := Round((32767-Joystick.PositionY)/327.67);   // -100 .. +100 %
        SetRotation:= Round((32767-Joystick.PositionX)/327.67);   // -100 .. +100 %
        if Abs(SetSpeed)    <= DeadJoystickPosition then SetSpeed:=0;
        if Abs(SetRotation) <= DeadJoystickPosition then SetRotation:=0;
      end
      else
      begin
    SetSpeed:=0;  SetRotation:=0;
      end;
    YellowRobot.Move(SetSpeed, 0);
end;
///-----------------------------------------------------------------------------------------
procedure TGlavniProzor.TJtimer2Timer(Sender: TObject);
  Var  t : real;
begin
    if CommandTimeOut > 0 then Dec(CommandTimeOut);
    if WaitTime       > 0 then Dec(WaitTime);
    if TimerA         > 0 then Dec(TimerA);
   // CommandTimeOutLabel.Caption:= IntToStr(CommandTimeOut DIV 10);
    if TaskM then Inc(TaskTime);
    Inc(CommandTime);
```

```
    if BlueRobot.Recording then
     begin
        Inc(RecordTime);
        t:=RecordTime/10;
        Write(RecFile, '      ',t:4:1);
        if MASK[ 1] = '1' then Write(RecFile, YellowRobot.Speed:10);
        if MASK[ 2] = '1' then Write(RecFile, YellowRobot.Path:10);
        if MASK[ 3] = '1' then Write(RecFile, YellowRobot.Rotation:10);
        if MASK[ 4] = '1' then Write(RecFile, YellowRobot.Angle:10);

        if MASK[ 6] = '1' then Write(RecFile, BlueRobot.Speed:10);
        if MASK[ 7] = '1' then Write(RecFile, BlueRobot.Path:10);
        if MASK[ 8] = '1' then Write(RecFile, BlueRobot.Rotation:10);
        if MASK[ 9] = '1' then Write(RecFile, BlueRobot.Angle:10);

        if MASK[11] = '1' then Write(RecFile, RedRobot.Speed:10);
        if MASK[12] = '1' then Write(RecFile, RedRobot.Path:10);
        if MASK[13] = '1' then Write(RecFile, RedRobot.Rotation:10);
        if MASK[14] = '1' then Write(RecFile, RedRobot.Angle:10);

        if MASK[16] = '1' then Write(RecFile, Brzina:10);
        if MASK[17] = '1' then Write(RecFile, BlueRobot.S0:10);
        if MASK[18] = '1' then Write(RecFile, BlueRobot.S5:10);
        if MASK[19] = '1' then Write(RecFile, RedRobot.S0:10);
        Writeln(RecFile);
     end;
  lblPutZuti.Caption:='Put Zutog = ' + Inttostr(YellowRobot.Path);
  lblPutCrvenog.Caption:='Put Crvenog = ' + Inttostr(Redrobot.Path);
  lblPutPlavog.Caption:='Put Plavog = ' + inttostr(BlueRobot.Path);
  lblRazmak12.Caption:='Razmak prvog i drugog = '+Inttostr(BlueRobot.S0);
  lblRazmak23.Caption:='Razmak drugog i treceg = '+inttostr(RedRobot.S0);
  lblBrzinazuti.Caption:='Brzina žutog robota ='+inttostr(yellowrobot.Speed);
  lblBrzinaplavi.Caption:='Brzina plavog robota ='+inttostr(bluerobot.Speed);
  lblBrzinacrveni.Caption:='Brzina crvenog robota ='+inttostr(redrobot.Speed);
end;
//---------------------------------------------------------------------------
Procedure StartRecording(S : string; VarMask : string);
begin
    if BlueRobot.Recording then Exit;     // inaèe I/O error 103
    RecordTime:=0;
    MASK:=VarMask;  while Length(MASK) < 19 do MASK:=MASK+'0';
    AssignFile(RecFile,'Record.txt'); Rewrite(RecFile);
    Writeln(RecFile, S);
    Write(RecFile,'  time (s)');
    if MASK[ 1] = '1' then Write(RecFile,'      BrzY');
    if MASK[ 2] = '1' then Write(RecFile,'      PutY');
    if MASK[ 3] = '1' then Write(RecFile,'      RotY');
```

```
    if MASK[ 4] = '1' then Write(RecFile,'      KutY');

    if MASK[ 6] = '1' then Write(RecFile,'      BrzB');
    if MASK[ 7] = '1' then Write(RecFile,'      PutB');
    if MASK[ 8] = '1' then Write(RecFile,'      RotB');
    if MASK[ 9] = '1' then Write(RecFile,'      KutB');

    if MASK[11] = '1' then Write(RecFile,'      BrzR');
    if MASK[12] = '1' then Write(RecFile,'      PutR');
    if MASK[13] = '1' then Write(RecFile,'      RotR');
    if MASK[14] = '1' then Write(RecFile,'      KutR');

    if MASK[16] = '1' then Write(RecFile,'   Reverent');
    if MASK[17] = '1' then Write(RecFile,'   PlaviS0');
    if MASK[18] = '1' then Write(RecFile,'   PlaviS5');
    if MASK[19] = '1' then Write(RecFile,'  CrveniS0');
    Writeln(RecFile);
    BlueRobot.Recording:= True;
end;
///-----------------------------------------------------------------------------------------
Procedure StopRecording;
begin
    if BlueRobot.Recording then CloseFile(RecFile);
    BlueRobot.Recording:= False;
end;
end.
```