Vođenje mobilnih robota u zadanom rasporedu

Jurendić, Domagoj

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Domagoj Jurendić

Zagreb, 2020.g.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori: Student:

Prof. dr. sc. Mladen Crneković, dipl. ing. Domagoj Jurendić

Zagreb, 2020.g.

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

Zahvaljujem se mentoru prof. dr. sc. Mladenu Crneković na ukazanoj stručnoj pomoći u izradi ovog završnog rada, kao i na iznimnoj ljubaznosti i strpljivosti u rješavanju problema, i upita.

Također zahvaljujem i svojoj obitelji za podršku i oslonac tijekom čitavog preddiplomskog studija.

Domagoj Jurendić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

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

ZAVRŠNI ZADATAK

Student: **DOMAGOJ JURENDIĆ** Mat. br.: 0035209079

Naslov rada na hrvatskom jeziku: **VOĐENJE MOBILNIH ROBOTA U ZADANOM RASPOREDU**

Naslov rada na engleskom jeziku: **CONTROL OF MOBILE ROBOTS IN FORMATION**

Opis zadatka:

Prvi korak u koordinaciji više mobilnih robota je vožnja u zadanom rasporedu. Održavanje formacije više robota preduvjet je za ostala složenija djelovanja (npr. zajedničko prenošenje istog predmeta).

Potrebno je pronaći komercijalne primjene mobilnih robota koji rade u zadanom rasporedu. Na eMiR mobilnim robotima ostvariti vožnju u formaciji za najmanje dva robota (jedan iza drugog, jedan pored drugog).

U radu je potrebno:

- definirati teoretske osnove vožnje mobilnih robota u formaciji
- predložiti algoritam vođenja za formaciju najmanje dva eMiR mobilna robota
- na poligonu eksperimentalno verificirati predloženi algoritam

Potrebno je navesti korištenu literaturu i ostale izvore informacija te eventualno dobivenu pomoć.

Zadatak zadan:
28. studenog 2019.

Zadatak zadao:
Prof.dr.sc. Mladen Crneković

Datum predaje rada:
1. rok: 21. veljače 2020.
2. rok (izvanredni): 1. srpnja 2020.
3. rok: 17. rujna 2020.

Predviđeni datumi obrane:
1. rok: 24.2. – 28.2.2020.
2. rok (izvanredni): 3.7.2020.
3. rok: 21.9. - 25.9.2020.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovom radu biti će prikazana izvedba algoritma za gibanje u formaciji dvaju eMIR edukativnih mobilnih robota razvijenih na Katedri za strojarsku automatiku. Problem formacijskog gibanja mobilnih robota nedovoljno je istraženo područje koje je uvelike limitirano mogućnošću izvedbe izvan zatvorenog poligona. Konstrukcija i raspored opreme(senzora) na robotima prilagođena je kako bi se omogućilo preciznije i bolje pozicioniranje jednog robota u odnosu na drugi.
U radu je razvijena i Android aplikacija u online programskom paketu MIT App Inventor, razvijenom od strane Google-a i MIT Media Lab-a, koja služi za upravljanje vodećeg robota pomoću virtualnog joystick upravljača, uz pomoć bluetooth veze.
Također stvorena je i programska biblioteka za upravljanje s dva ili više eMIR mobilnih robota putem serijske komunikacije u C++ programskom jeziku, sam algoritam stvoren je u Arduino Integrated Development Environment(Arduino IDE), koji se koristi za programiranje Arduino kompatibilnih mikrokontrolerskih pločica.
U radu su pokrivene i teorijske osnove ostvarivanja gibanja u formaciji, te su navedene i pobliže objašnjene neke od strategija ostvarivanja i upravljanja formacije mobilnih robota kako bi se ostvarila složena djelovanja grupe robota, koja bi omogućila čitav niz primjena u stvarnom svijetu, neka od kojih se već i koriste u svakodnevnom životu, a neka tek trebaju biti plasirana na tržište.

Ključne riječi: eMIR, programiranje, Arduino IDE, MIT App Inventor, mobilni roboti u formaciji

SUMMARY

This paper will describe the development of the algorithm for controlling the formation motion of two eMIR educational mobile robots developed at the Chair of Mechanical Engineering. The problem of formation movement of mobile robots is a undeveloped area of robotics greatly limited by the possibility of execution outside of an enclosed polygon. Construction and layout of the equipment(sensors) on the robots has been modified to enable better and more precise positioning of robots in respect to each other.

The paper will also show the developed Android app, made in a web based integrated development enviroment MIT App Inventor, developed by Google and MIT Media Lab, which will be used to control the leading robot using a virtual joystick controller, connected via bluetooth connection. Also a library has been developed for controlling two or more eMIR mobile robots using bluetooth serial communication, in C++ language, the algorithm for formation execution has been developed in Arudino Integrated Development Environment(Arduino IDE), which is used to programm Arduino compatible microcontroller boards.

The paper will aslo cover the theoretical basis of achieving the formation of mobile robots, and will list and describe some of the strategies of creating a formation to create complex motions of a group of robots, which will achieve a number of applications in real world situations, some of which are already implemented and used, and some that yet need to be developed.

Keywords: eMIR, programming, Arduino IDE, MIT App Inventor, mobile robots in formation

1 UVOD

Razvojem robotike kao i jednako bitnog područja senzorike, stvaraju se nove mogućnosti izvedbe raznih robotskih sustava, ali i otvaranja novih neistraženih područja na koje tek treba dati valjano rješenje. Od početka razvoja mobilnih robota razmišlja se i o njihovoj međusobnoj interakciji kao i interakciji s okolinom. Razvijaju se različiti algoritmi implementacije i upravljanja robotskim formacijama, u različitim okruženjima poput industrijskih pogona, automatiziranih skladišta, stvaraju se nove generacije pametnih kućanskih uređaja, u području autoindustrije razvijaju se sustavi za održavanje vozila u formaciji primjerice na autocestama, a sve u svrhu povećanja ekonomičnosti, sigurnosti, i smanjenja zamora ljudi uslijed teškog i monotonog rada.

U sklopu ovog rada razvijen je algoritam za održavanje linearne formacije dvaju eMIR edukacijskih robota, uz potreban popratni kod biblioteke za serijsku komunikaciju s robotom i upravljačem vodećeg robota, koji je ostvaren Android aplikacijom koja simulira joystick upravljač, te bluetooth vezom šalje potrebne podatke kontroleru, koji će obraditi informacije dobivene od očitanja senzora na robotima, te odrediti potrebne manipulacije robota kako bi se održala pravilna formacija robota.

U radu su također prikazane i razrađene osnovne teorije upravljanja formacijom grupe robota, među kojima je i strategija slijeđenja vođe, prikazan je princip ostvarivanja pojedinih strategija, uz osvrt na prednosti i nedostatke. Određivanje i praćenje trajektorije gibanja robota, zahtjevan je problem na koji su mnogi inženjeri dali svoje rješenje, no ne nalazimo idealno implementiran algoritam.

Ostvarivanje gibanja grupe robota u formaciji u određenoj specifičnoj okolini ideja je koju se sve češće razmatra, jer bi takva rješenja uvelike pojednostavila ljudski rad, kako u transportu i u radnoj okolini, tako i u sferi istraživanja okoliša, gdje se ljudi mogu naći u veoma teškim i po život opasnim situacijama, gdje bi grupa robota u potrebnoj formaciji mogla brzo i efikasno pregledati i istražiti veliko područje, pritom skupljajući potrebne informacije.

Primjer takve po život opasne situacije je posao razminiranja minski sumnjivih područja, od kojeg u čitavom svijetu stradaju stotine ljudi, što nažalost nije nepoznat slučaj i u našoj državi, ovakav tip posla idealan je motiv za istraživanje i razvoj gibanja robota u formaciji,

što bi omogućilo uklanjanje ljudi iz barem dijela opasnoga posla, ubrzalo bi postupak sanacije zemlje, te bi se time spriječile eventualne ljudske žrtve, jer ma koliko skup robotski sustav bio ljudski život nezamjenjiv je.

Slika 1. Robot za detekciju mina

2 FORMACIJA

Formacija mobilnih robota može se okarakterizirati kao grana robotskih sustava koja proučava koordinaciju grupe mobilnih robota koja će osigurati održavanje formacije željenog oblika. Ovakve formacije mobilnih robota mogu se iskoristiti za izvršavanje različitih zadataka, koje grupa robota može odraditi kvalitetnije i bolje, nego kada bi svaki robot radio kao pojedinac. Ti zadaci mogu biti raznoliki, pa možemo govoriti o transportu objekata koristeći mobilne robote koji će surađivati i gibati se u formaciji, ili zadaće poput istraživanja terena, nadzor okruženja i slično.

Sve ove zadaće mogu naći primjenu u današnjem svijetu, poput pretraživanja mjesta nesreća i prirodnih katastrofa, različitih primjena u medicinskim operacijama gdje služe kao pomoć medicinskom timu, zatim za razvoj mobilne senzorske mreže, za nadzor i mjerenje raznih veličina, poput temperatura mora, stupnja zagađenosti zraka itd.
Naravno primjena se nalazi i u užem području industrije gdje se formacije mobilnih robota mogu primijeniti u transportu različitih objekata, nezgrapnih oblika i masa, zatim i u automatiziranim skladištima gdje se gibanje u formaciji i detekcija kolizije implementiraju radi povećanja protočnosti transportnih kanala.

Temeljni problem predstavlja implementacija algoritama koji će omogućiti kontrolu grupe robota i njihovo održavanje u formaciji, te se u tom aspektu nailazi na nekoliko problema.
Najprije je potrebno utvrditi inicijalnu poziciju pojedinog robota, kako bi se utvrdilo početno stanje formacije, potom je potrebno odrediti putanju robota od inicijalne pozicije u formaciji do krajnje pozicije koja će zadovoljiti uvjete oblika formacije, a da se u tom procesu ostvari što stabilnija transformacija položaja jednog robota u odnosu na drugi.
U sklopu procesa treba predvidjeti i problem izbjegavanja prepreka i objekata koje algoritam također treba zadovoljiti.

Različiti su se lokalizacijski sustavi razvijali u svrhu pozicioniranja u lokalnom okruženju, a za tu primjenu koriste se različiti senzorski sustavi poput infra-crvenih daljinomjera, ultra zvučnih senzora, sonara i laserskih daljinomjera.

Različite metode određivanja putanja robota vrlo su važne komponente za formaciju više mobilnih robota, ove metode možemo podijeliti na globalne i lokalne, temeljeno na informacijama koje nam senzori pružaju. Tako će globalna metoda dati sigurnost u dostizanju određenog cilja odnosno pozicije, dok lokalne metode ne mogu sa sigurnošću ostvariti pozicioniranje u okolišu.

Ostvarenje grupnog gibanja robota možemo klasificirati kao:

1. Grupiranje
2. Kontrola formacije

Kontroliranje formacije grupe mobilnih robota znači stabiliziranje i održavanje željenog oblika formacije, poput formacije na donjoj slici.

Slika 2. Kontrola formacije grupe robota

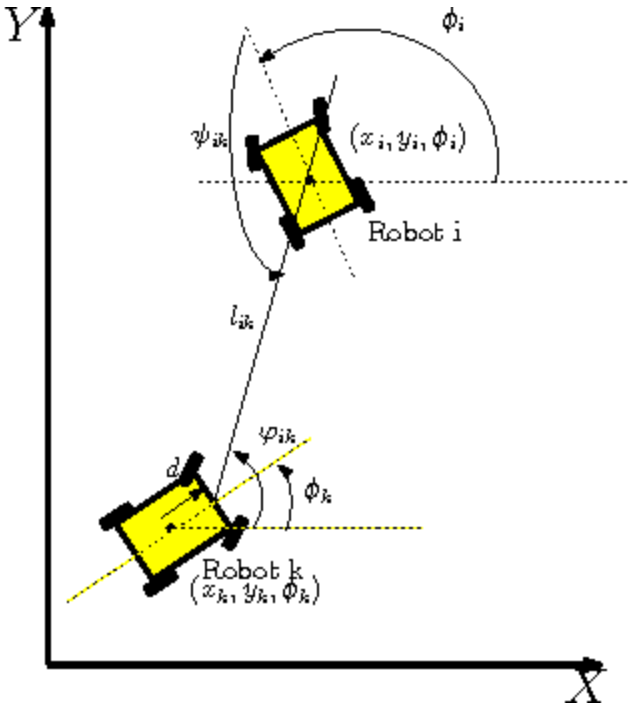
Citirajući nekoliko autora istraživača, kontrola gibanja u formaciji može se podijeliti na nekoliko pristupa:

1. Strategija praćenja voditelja[2]
2. Metoda virtualne strukture[3]
3. Metoda temeljena na ponašanju[3]

2.1 Strategija praćenja voditelja

Kod strategije praćenja voditelja robot koji vodi formaciju kreće se po unaprijed definiranoj putanji, ili kao što je u našem slučaju upravljan pomoću kontrolera po željenoj trajektoriji. Ostatak kolone ima zadatak održavati stabilnu formaciju s obzirom na vodećeg robota, različitim tehnikama praćenja trajektorije.

Prednost je metode da je lako razumljiva i jednostavna za implementaciju, ne zahtjeva veliku procesorsku moć za izvedbu algoritma, no uz to postoje određeni nedostatci koji ukoliko se pravilno ne sagledaju mogu imati katastrofalne posljedice u vidu održavanja formacije robota. Jedan od nedostataka je potreba za centraliziranom kontrolom iz razloga što roboti pratioci koriste relativnu poziciju voditelja kao ulazne varijable, iz tog razloga metoda nije pogodna za veliki broj robota. Naravno tu je i problem ne postojanja povratne veze od strane pratitelja prema vodećem robotu što može stvoriti dodatne probleme. Na primjer ukoliko se vodeći robot giba prevelikom brzinom u usporedbi s pratiocima, ili ukoliko robot pratitelj naiđe na prepreku može se dogoditi gubitak formacije. Gledajući s druge strane mora se paziti također ukoliko dođe do zatajenja robota voditelja, da se poduzmu potrebni koraci kako bi se spriječio neželjen gubitak formacije i slično.



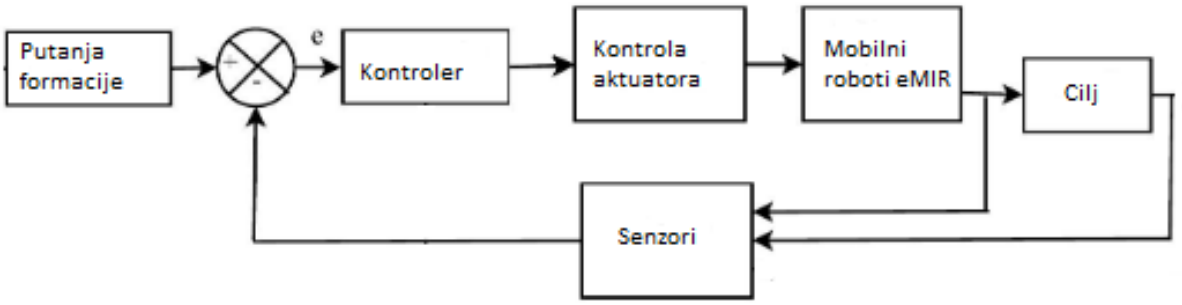
Slika 3. Strategija praćenja voditelja

Kao što je navedeno uz svoje prednosti metoda ima i svoje nedostatke koji moraju biti pravilno uzeti u obzir u procesu pisanja algoritma kako bi se što bolje spriječio gubitak formacije u vrijeme gibanja robota. Zato je potrebno dobro razraditi algoritam, koji će biti u mogućnosti uočiti problem i djelovati u cilju njegova otklanjanja.

2.2 Dizajn kontrolera

Zadatak kontrolera je pronaći vrijednosti translacijskih i rotacijskih brzina kotača robota, koje će dovesti do uklanjanja pogreške željene udaljenosti robota i njihove orijentacije. Na slici 4. prikazan je pojednostavljeni model jednog takvog kontrolera čija je zadaća održavati razinu pogreške udaljenosti i orijentacije na nuli. Svi roboti moraju se gibati u skladu s translacijskim brzinama *v* i rotacijskim brzinama ω određenim od strane kontrolera. U svakom trenutku računa se položaj robota pratioca obzirom na izmjereni položaj robota voditelja, koji se ažurira svakih ~250ms. Ukoliko imamo više pratioca, analogno navedenom za svaki slijedeći robot pratitelj računa se njegov položaj u odnosu na izmjereni položaj prethodnog robota pratitelja. Kako bi što

bolje smanjili pogrešku pozicioniranja stvorena je određena vrsta povratne veze kojom nastojimo reducirati odstupanje položaja robota na minimum.



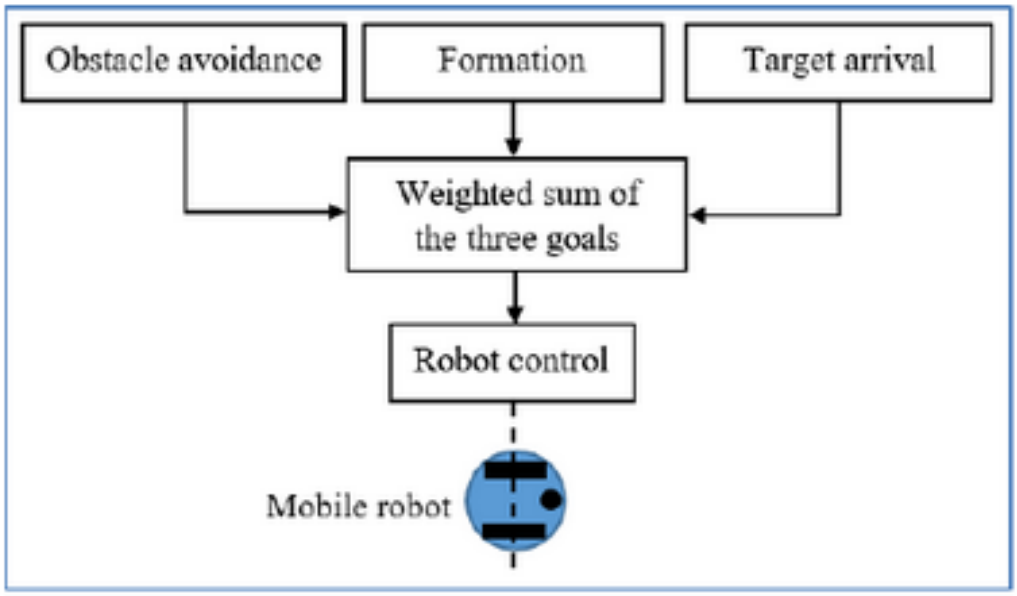
Slika 4. Kontroler formacije

Robot pratitelj konstantno prati voditelja, a onog trena kada parametar linearne udaljenosti i kutnog odstupanja padne na nulu, uspješno je izvršena regulacija položaja.

2.3 Metoda temeljena na ponašanju

Ova metoda zasniva se na decentraliziranom upravljanju koje omogućuje upravljanje jednog ili više robota u grupi. Treba nadodati da decentralizirana metoda upravljanja ne nudi planiranje i razumno rasuđivanje u odabiru načina reagiranja. To znači da ne postoji centralna jedinica koja će upravljati sustavom, tako da je metodu moguće implementirati s manje potrebe za komunikacijom, jer svaki pojedinačni robot upravlja vlastitim kontrolama pazeći pritom na nekoliko faktora, koji direktno utječu na odluku o slijedećem koraku u održavanju formacije. Dakle svaki robot odvaguje između nekoliko propisanih gibanja temeljem razmatranja relativne važnosti svakog faktora koji se prati.
Određivanje trajektorije gibanja robota pomoću ovakve metode generira se izračunavanjem srednje vrijednosti faktora za:

- Izbjegavanje prepreka
- Istraživanje formacije
- Prilaz cilju



Slika 5. Metoda temeljena na ponašanju[2]

U primjeni metode svako ponašanje robota sastoji se od navedenog skupa (koji može imati i mnog više elemenata) pod-ponašanja, od kojih je svako predstavljeno u obliku vektora sastavljenog od iznosa(amplitude) i smjera. Težinu/važnost svakog vektora možemo promijeniti mijenjanjem težinskih parametara.

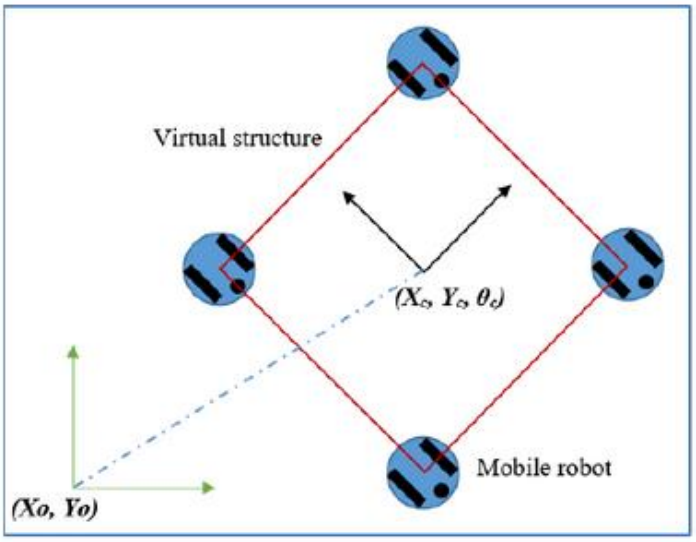
Tako će temeljeno na informacijama koje robot dobiva iz okoline, robot odabrati pravilno ponašanje, te će naredba za gibanje biti izvršena.

Ukupni vektor ponašanja koji je suma svih vektora pod-ponašanja može se dakle prikazati kao:

$$V_{cilj} = [f_1 \quad f_2 \quad f_3] * \begin{bmatrix} V_{izbjegavanje\_prepreka} \\ V_{formacija} \\ V_{prilaz\_cilju} \end{bmatrix}$$

2.4 Metoda virtualne strukture

Metoda virtualne strukture promatra formaciju robota kao jedinstvenu strukturu u kojoj svaki robot dobiva set upravljačkih naredbi za praćenje željene trajektorije, kako je prikazano na donjoj slici. Jedna od najvećih prednosti ove metode je jednostavan matematički prikaz translacije čitave strukture pomoću jednostavnih matematičkih relacija.
Također metoda omogućava jednostavno kombiniranje dobivenih ponašanja pomoću vektorskih operacija, i to tako da se svaka rotacija/ translacija virtualne strukture zapiše u obliku vektora, čija suma daje ukupno kombinirano gibanje strukture.



Slika 6. Virtualna struktura robota[2]

2.5 Ostale metode i strategije

2.5.1 Metoda razmještaja

Metoda kod koje roboti upravljaju razmještajem sebi susjednih robota, kako bi se ostvarila željena formacija. Formacija je određena željenim razmacima u odnosu na globalni koordinatni sustav, pod pretpostavkom da je svaki robot u mogućnosti mjeriti relativnu poziciju sebi susjednih robota, s obzirom na globalni koordinatni sustav.
Dakle uvjet za ostvarivane formacije ovom metodom je da robot mora biti u mogućnosti odrediti orijentaciju globalnog koordinatnog sustava kao i svoju poziciju u odnosu na taj sustav.

2.5.2 Metoda udaljenosti

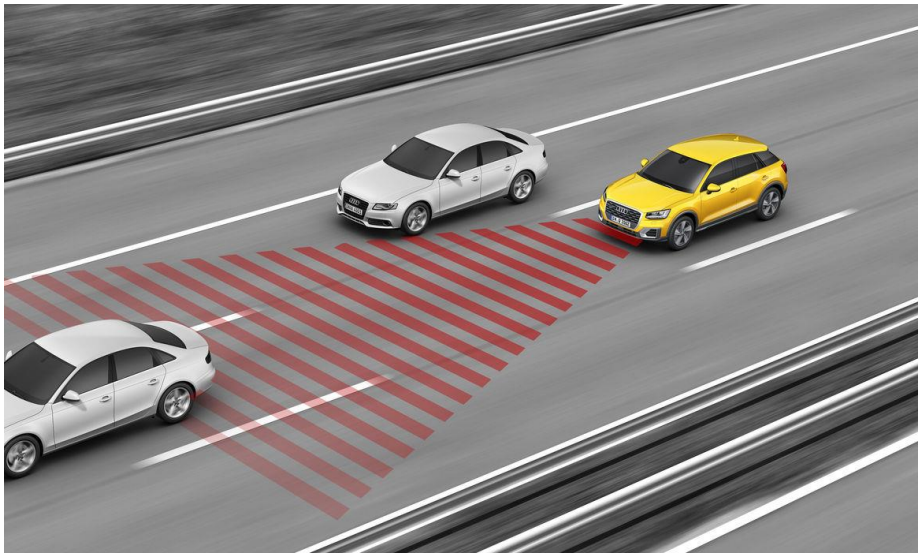
Da bi se ostvarila formacija, metoda upravlja unutarnjim udaljenostima robota. Određivanjem željenih udaljenosti definira se oblik formacije. Pretpostavka je da su roboti u mogućnosti mjeriti relativnu poziciju sebi susjednih robota s obzirom na svoje lokalne koordinatne sustave.
Treba napomenuti da pri ostvarivanju formacije ovom metodom, roboti nužno ne moraju biti pravilno raspoređeni jedan u odnosu na drugi.

2.5.3 Metoda pozicija

Kod ove metode nužan preduvjet je da svaki robot ima mogućnost određivanja vlastite pozicije u odnosu na globalni koordinatni sustav. Oblik formacije biti će tada definiran pozicijama i orijentacijom robota unutar koordinatnog sustava

3. PRIMJENA

Kao što je vidljivo iz prethodnih poglavlja ostvarenje gibanja mobilnih robota u formaciji kompleksan je problem sam po sebi, dodamo li u tu temu još i potrebu izbjegavanja mogućih prepreka, stacionarnih ili dinamičkih, utjecaje okoline, uzrokovane smetnje u signalima, problem preciznog pozicioniranja u globalnom sustavu, možemo zaključiti da je primjena mobilnih robota s gibanjem u formaciji veoma zahtjevan problem za izvedbu u realnom svijetu.
Zato je danas izvan nekih specifičnih lokalnih okruženja teško naći mobilne robote u primjeni, no razvojem tehnologija vidimo da se i ti problemi postepeno savladavaju. Najbolje je to vidljivo u autoindustriji i razvoju autonomnih i polu autonomnih automobila i kamiona, koje možemo smatrati i jednom vrstom mobilnih robota. Tako je već danas veliki broj auto proizvođača u osnovnu opremu automobila uvrstio i takozvani ACC sustav(Adaptive cruise control), to jest sustav adaptivnog tempomata koji vozilima omogućuje održavanje linearne formacije u gibanju na autocesti.



Slika 7. ACC sustav

3.1 ACC sustavi

Sustav adaptivnog tempomata(Adaptive cruise control), sustav je koji se ugrađuje u cestovna, osobna i kamionska vozila, za automatsku prilagodbu brzine vozila kojom se održava sigurna udaljenost između vozila u prometu. Kontrola brzine vrši se temeljem informacija dobivenih od senzorskih sustava sačinjenih uglavnom od radarskih ili laserskih senzora, ili vizijskih sustava koji omogućavaju vozilu smanjenje brzine pa i kočenje u slučaju kad upravljačka jedinica sustava detektira vozilo ispred. Isto tako sustav će i ubrzati vozilo ukoliko situacija u prometu to dozvoljava.

Ovi sustavi predstavljaju ključnu komponentu buduće generacije inteligentnih vozila. Osiguravaju sigurnost sudionika u prometu, povećavaju udobnost i lakoću putovanja, ali i povećavaju kapacitet propusnosti ceste, održavajući optimalan razmak između vozila, te tako smanjuju utjecaj greške vozača.

Vozila ovim sustavom spadaju u prvu razinu kategorija autonomije automobila, kako je definirano od strane SAE International(Udruženje inženjera motornih vozila).

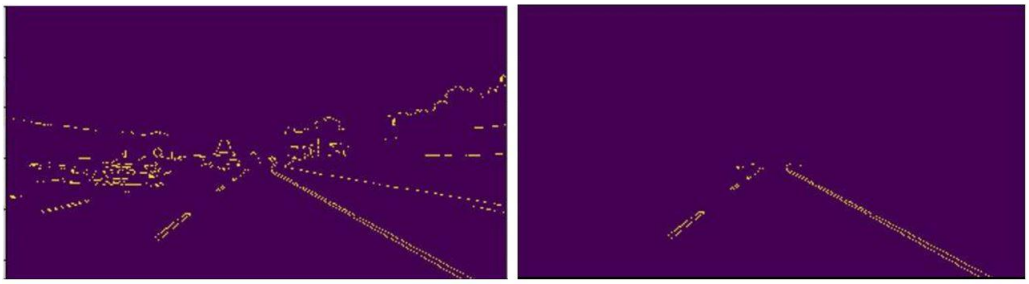
ACC sustave dijelimo na nekoliko podvrsta:

- Pomoćni sustavi - sustavi s radarskim senzorima, koji su često integrirani sa sustavima detekcije sudara, koji će vozača upozoravati o mogućim opasnostima i/ili pružiti pomoć u kočenju
- Više senzorski sustavi – Sustavi koji pomoću više senzora mogu objediniti podatke u cilju poboljšanja sigurnosti i općenitog iskustva vožnje. Na primjer GPS podatci mogu obavjestiti sustav o nadolazećem izlasku s autoceste, dok će u tom slučaju vizulani sustav kamera detektirati ponašanje vozila ispred, poput kočenja, ili davanja žmigavca. Ovo bi omogućilo sustavu da zaključi kako upaljen žmigavac vozila ispred znači kako nema potrebe za usporavanjem vozila, jer će vodeći automobil sići s autoceste.
- Sustavi s predviđanjem – Ovi sustavi prilagođavali bi brzinu vozila temeljem predviđanja o ponašanju vozila ispred. Ovo bi omogućilo raniju postepeniju prilagodbu brzine vozilo, poput na primjer slučaja predviđanja prelaska vozila ispred u drugu kolničku traku.

Slika 8. Radarski senzor

3.2 LDWS sustavi

Također nerijetka su pojava i sustavi LDWS(Lane departure warning system) sustavi za održavanje vozila u voznoj traci, koji čak i kontroliraju kut zakreta letve volana održavajući pravilnu trajektoriju vozila. Sustav je konstruiran da bi minimizirao broj nezgoda, rješavanjem jednog od najčešćih uzroka nesreća, a to su greška vozača, umor, i slično.
Sustav detekcije kolničke trake/linije kao jedan od podsustava, koristi se da bi pomoću Hough-ove transformacije za obradu slika i računalnu viziju, te Canny edge detectora(operatora za detekciju bilo kakvih vrsta rubova i linija na slikama.), obrađivao slike prednje kamere automobila u realnom vremenu.



Slika 9. Canny detekcija rubova i Houghova transformacija

Očigledno jedan od glavnih i najviše korištenih senzora jest kamera, koja se najčešće ugrađuje iza vjetrobranskog stakla, ili iza središnjeg retrovizora vozila. No uz dodatak kameri, koriste se i Laserski senzori, montirani najčešće sprijeda, te infracrveni senzori, koji se nalaze ili iza vjetrobranskog stakla ili čak ispod vozila.

LDWS sustave možemo slično ACC sustavima podijeliti u tri grupe:

- Sustavi upozorenja – Upozoravaju vozača o odlasku iz kolničke trake vizualnim, zvučnim, ili vibracijskim signalima(Lane departure warning DPW)
- Sustavi upozoravanja s automatskim djelovanjem – ukoliko vozač ne reagira na upozorenje sustava, poduzet će se automatski koraci kako bi automobil ostao u traci (Lane keeping assist, LKA/LKS)
- Sustavi pomoćnog upravljanja – pomažu vozaču u upravljanju volanom vozila, centrirajući vozilo na sredinu trake. Upozoravaju vozača o potrebi preuzimanja kontrole u slučaju zahtjevne situacije(Lane centering assist, LCA)

Sustavi još uvijek teško funkcioniraju u situacijama gdje ne postoje vidljive oznake na kolniku.

Tako na primjer nailazimo na potpuno autonomne kamione tvrtke Scania, koji osim svoje autonomije rade u kolaboraciji jedni s drugima, održavajući vožnju u formaciji upravo pomoću ACC i LDWS sustava. Još uvijek kamioni funkcioniraju samo u zatvorenom okruženju, što zbog zakonskih razloga što zbog eventualnih nedostataka u sustavima, no vremenom bi se mogli naći i na javnim cestama. cilj je ovakve kamione osposobiti najprije za rad u zatvorenim okruženjima izvan javnih cesta, poput kamenoloma, rudnika i sličnih lokacija.

Slika 10. Autonomni kamioni u formaciji

3.3 Poljoprivreda

Kako ne bi dobili sliku o ograničenosti primjene algoritama za gibanje u formaciji, valja napomenuti i navesti i slučajeve izvan sfere transporta i automobilizma. Naime, jedan od problema poljoprivredne proizvodnje je smanjene broja radne snage, kako raste stupanj obrazovanja stanovništva u razvijenim zemljama, dok istovremeno raste svjetska potražnja za poljoprivrednim proizvodima.

Iz tog razloga brojni inženjeri uložili su napore kako bi stvorili robote za obavljanje raznih poljoprivrednih zadaća, na dva osnovna principa:

- Strojevi sa vizijskim sustavima, odometrima, akcelerometrima i sl.
- Strojevi sa sustavima temeljenima na satelitskoj GPS navigaciji

Greška u točnosti kinematičkih GPS sustava u stvarnom vremenu (RTK GPS), danas je svega 1 do 2cm na 10km. Često se za ovu zadaću implementiraju algoritmi brisanja prostora za sustave s više robota. Jedina razlika između zadaće poljoprivrednog stroja, i algoritma brisanja prostora je ta da poljoprivredni stroj mora svaki komad zemlje obraditi samo jednom, u minimalno potrebnom vremenu.

Često je potrebno koordinirati dva ili više robota kako bi se povećala efikasnost obavljanja posla. Primjerice, robotski kombajn obrađujući polje, uz sebe mora imati pratnju transportnih pretovarnih robota, koji će voziti produkte sjetve na za to predviđene lokacije, i omogućiti kombajnu neometan rad bez potrebe za stajanjem.

Drugi primjer bio bi suradnja dvaju robotskih traktora, od kojih bi vodeći obavljao radnju primjerice oranja, dok bi sljedeći traktor obavljao zadaću sadnje ili gnojenja zemljišta.

Uglavnom se istraživači orijentiraju na master-slave algoritme za obavljanje poljoprivrednih zadataka, gdje se sustav uglavnom bazira na GOTO i FOLLOW naredbama. Master će upravljati slave jedinicom, i određivati joj na kojoj udaljenosti i pod kojim kutom mora pratiti glavnu jedinicu u paralelnoj ili linearnoj formaciji.

Dakako, primjena dvaju robota može imati i svoje negativne posljedice na sustav. Ukoliko nije pravilno izveden, sustav bi se mogao nepotrebno zakomplicirati umjesto njegovog pojednostavljenja u odnosu na sustave s jednim robotskim strojem. Sustavi s više robota, kako je već i napomenuto u prethodnom poglavlju nailaze na probleme koji ne postoje u sustavima s jednim robotom, poput komunikacije između robota, praćenja i održavanja optimalne formacije

za obradu zemlje, ali i specifičnih problema za ovaj slučaj, poput načina suradnje u zaokretu stroja prilikom dolaska do ruba odrađenog zemljišta.

Slika 11. Autonomni poljoprivredni stroj

3.4 Automatizirani skladišni prostori

Moraju se spomenuti i sve češći i razvijeniji sustavi automatiziranih skladišnih prostora, koji koriste veliki broj transportnih robota različitih oblika i primjena. Tvrtke poput Amazona, ulažu velika financijska sredstva u automatizaciju svojih postrojenja, gdje se svakodnevno procesira milijuni paketa. Zato je potreban iznimno velik broj transportnih robota, koji moraju funkcionirati u istom radnom prostoru, izbjegavajući kolizije jedni s drugima. Naravno implementiraju se i algoritmi gibanja u formacijama, radi povećanja protočnosti robota transportnim putevima.

Naravno treba napomenuti kako potpuno automatizirana skladišta koja su u mogućnosti raditi s različitim vrstama paketa još uvijek nisu razvijena, jer za svaku novu radnju potrebno je reprogramirati robota, što je dug i skup proces, a razvoj umjetne inteligencije koja bi omogućila autonomno učenje robota nije dosegla tu razinu razvijenosti.

Slika 12. Idejna zamisao automatizacije skladišta

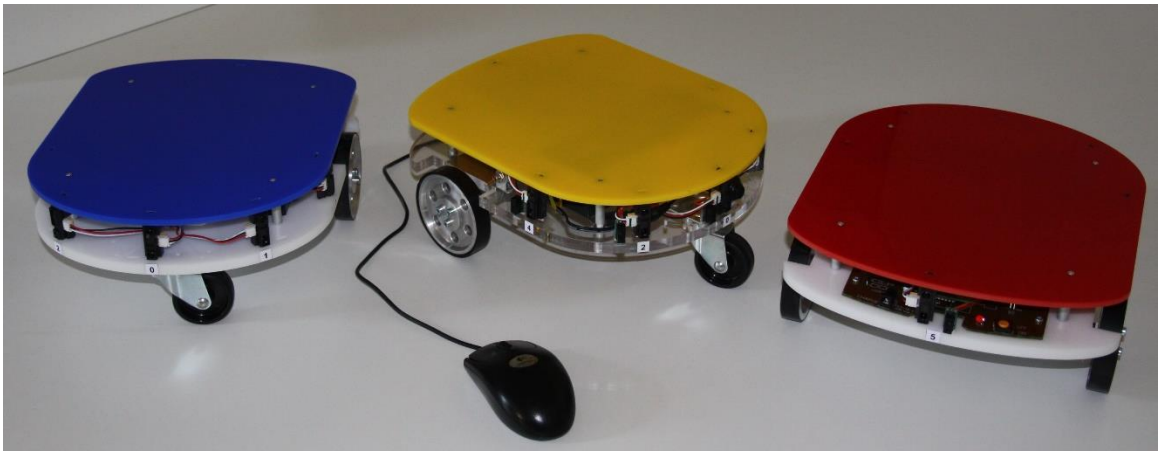
4. eMIR MOBILNI ROBOT

4.1 Opis robota

Na Katedri za strojarsku automatiku u sklopu LIPS laboratorija(Laboratory for inteligent production systems), nalaze se tri robota pod nazivom eMIR koji označava „educational Mobile Intelligent Researcher“.
Roboti su vanjskih dimenzija:

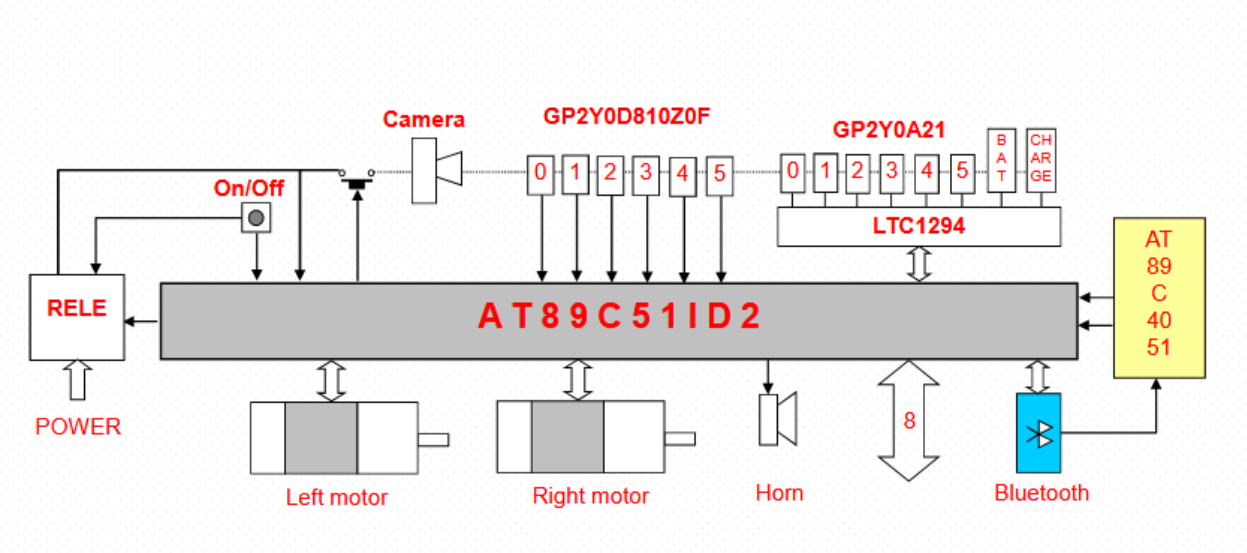
- Duljina 300mm
- Širina 250mm

Kučište robota izrađeno je od dvije PMMA ploče(polimetilmetakrilat), i to tako da je donja ploča debljine 10mm, a gornja ploča debljine 6mm. Tri robota razlikuju se po boji gornje ploče, pa tako imamo eMIR_Red, eMIR_Blue te eMIR_Yellow. Robot sa svom opremom(mikrokontroler, senzori, motori, baterija…) teži 3.5kg.
Opremljen je baterijom od 12V i 2.2Ah, koja će robotu omogućiti tri do četiri sata autonomije ukoliko u obzir uzmemo da je maksimalna potrošnja robota s uključenim senzorima i kamerama oko 400mA.
Naravno roboti su konstrukcijski pogodni za daljna nadograđivanja, pa je tako na donjoj slici vidljivo da je na žutom robotu ugrađena prednja kamera u boji rezolucije 720*576 koja se može povezati s osobnim računalom.Crveni robot je primjerice nadograđen uređajem za automatsko spajanje na punjač kada se za to javi potreba. Ispod se nalazi slika koja prikazuje robote u originalnom ne izmijenjenom stanju.



Slika 13. eMIR roboti[1]

Roboti sadrže vlastite mikrokontrolere AT89C51ID2 kojim upravljaju motorima, komuniciraju s kontrolerom, pale/gase senzore i kamere, pale i gase zvučne signale. Slika ispod prikazuje strukturu robota.



Slika 14. Kontrolna struktura eMIR robota[1]

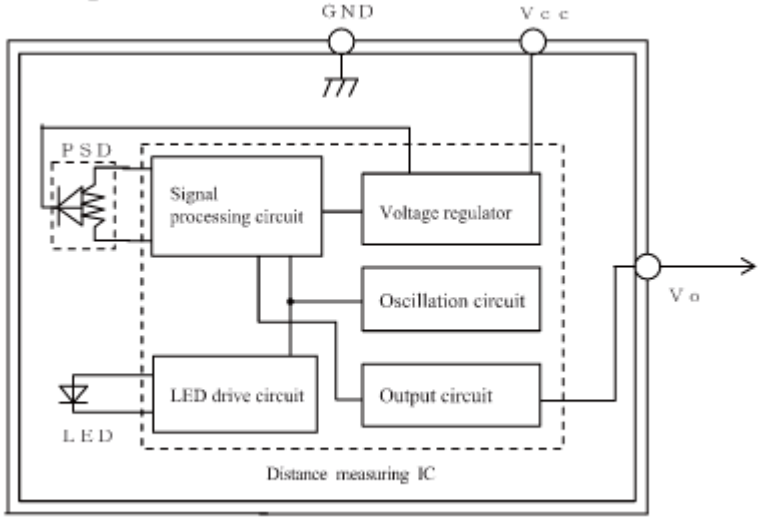
4.2 Oprema robota

4.2.1 Infracrveni senzori

Roboti su opremljeni sa šest komada analognih infracrvenih daljinomjera. Senzori su marke Sharp pod oznakom GP2Y0A21YK0F, spojeni su na analogne ulaze mikrokontrolera na robotu, koji obrađuje signale i pretvara ih u egzaktnu vrijednost udaljenosti izraženu u centimetrima. Područje rada senzora je od 10 do 80 cm, no njihova nelinearna karakteristika pruža veću preciznost na manjim udaljenostima.

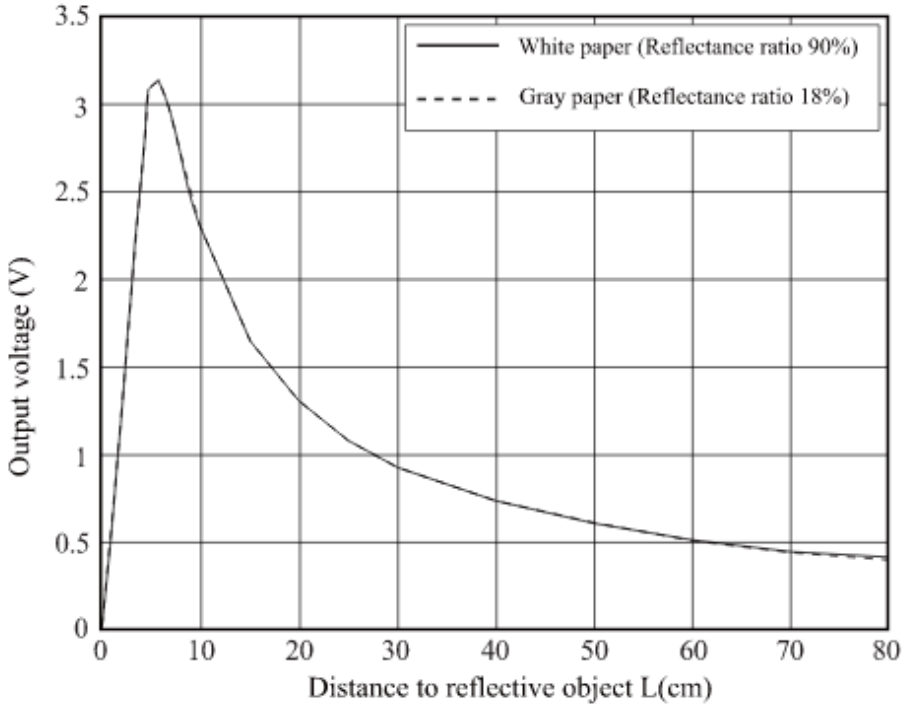
Slika 15. Sharp IR daljinomjer[1]

Senzor se sastoji od integriranog sklopa PSD-a(position sensitive detector), IRD(infrared emitting diode), te skopa za procesiranje signala, radnog napona 4.5 do 5V.



Slika 16. Blok dijagram Sharp daljinomjera

Mjerna karakteristike senzora prikazana je na donjoj slici, gdje vidimo da za područja od 10~30cm imamo poprilično dobru razlučivost signala, dok s porastom udaljenosti ulazimo u strmiji dio karakteristike te razlučivost pada, a time i smetnje u signalu.



Slika 17. Mjerna karakteristika Sharp daljinomjera

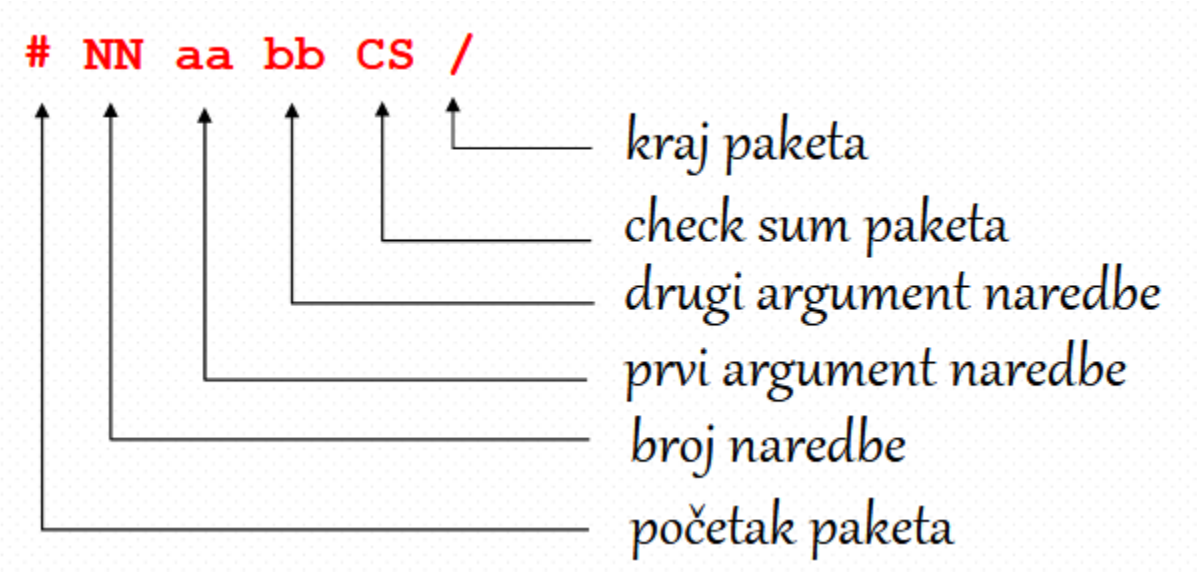
U izvedbi algoritma za gibanje u formaciji dva eMIR edukacijska robota korištena su dva IR senzora koji se nalaze na prednjem dijelu pratećeg robota. Vodeći voditelj na svojoj stražnjoj strani ima postavljenu bijelu plohu s koje se odbija infracrvena svjetlost senzora.

4.2.2 Bluetooth komunikacija

Svaki eMIR robot komunicira s kontrolerom pomoću bluetooth serijske komunikacije, kojom prima pakete naredbi, te vraća pakete očitanja stanja senzora i drugih informacija o robotu.
U robot je ugrađen bluetooth modul WRL-00582 proizvođača SparkFun. Komunikacija je podešena za rad pri brzini od 57600bps(bita u sekundi), no modul je u mogućnosti obavljati komunikaciju pri brzinama od 2400bps pa sve do 115200bps. Navedeni modul odlikuje iznimno dobar domet od 106m na otvorenom naspram običnih komercijalnih modula.
Bluetooth komunikacija pruža nam siguran i točan prijenos podataka uz velike brzine komuniciranja.

Slika 18. WRL-00582 bluetooth modul

Kao što je rečeno, naredbe robotu šalju se bluetooth komunikacijom u obliku paketa točno definiranog oblika koji sadrži oznaku početka naredbe(#), zatim broj naredbe s popratnim argumentima naredbe zapisanim u heksadekadskom sustavu, check sum paketa za provjeru ispravnosti pročitane narebe, te na kraju oznaku za kraj paketa(/).



Slika 19. Oblik paketa naredbe robotu[1]

U donjoj tablici prikazane su neke naredbe robotu:

NN	Paket	Opis
0	# 00 00 00 /	Zaustavi robota i prekini tekući posao
1	# 01 vv rr CS /	Postavi robota na zadanu brzinu vv i rr(-100 do 100%)
4	# 04 tt 00 CS /	Uključi sirenu tt desetinki sekunde
5	# 05 aa 00 CS /	Uključi(01)/isključi(00) senzore i kameru
255	# FF 00 00 01 /	Isključi robota

Tablica 1. Naredbe robotu

Povratni paket informacija od robota dolazi u drugačijem obliku. Oznaka početka povratnog paketa je (*), a za kraj paketa je(/). Unutar paketa nalazi se velik broj informacija o očitanjima senzora, trenutnoj brzini i rotaciji robota, naponu baterije , modu rada i ostalim informacijama, također zapisanima u heksadekadskom zapisu.

Paket povratnih informacija izgledati će tako na slijedeći način:

*aa bb cc dd ee ff uu vv rr gg hh mm jj kk CS /

Gdje pojedini dijelovi paketa označavaju:

- *- početak povratnog paketa
- aa…ff -izmjerene udaljenosti daljinomjera u cm
- uu -napon baterije ili napon punjenja
- vv -translacijska brzina robota(-100 do 100%)
- rr -rotacijska brzina robota(-100 do 100%)
- gg -PWM lijevog motora(-100 od 100%)
- hh -PWM desnog motora(-100 do 100%)
- mm -mod rada
- jj -stanje digitalnih ulaza
- kk -rezerva
- CS -check SUM paketa
- / -kraj paketa

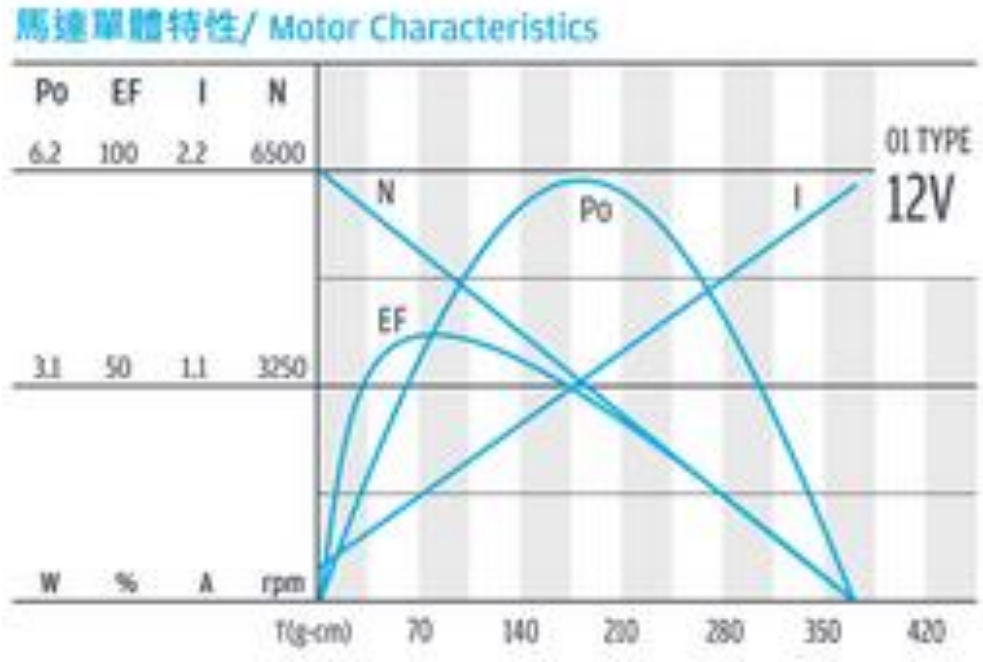
Ukoliko robot 1 sekundu ne primi nikakvu naredbu, zaustaviti će se, a ukoliko 120 sekundi ne primi nikakvu naredbu isključiti će se. Ukoliko napon baterije robota padne ispod 10.5V robot će se ugasiti radi produženja vijeka baterije, ali i osiguranja pouzdanog rada.

4.2.3 Motori i enkoderi

U robot su ugrađena dva motora s kotačima IG-32GM, istosmjerni motor s četkicama. Motor ima integriran planetarni reduktor prijenosnog omjera 71. Nazivni radni napon motora je 24V, no u sklopu eMIR robota koristi se na radnom naponu od 12V, ujedno i naponu baterije. Na motoru nalazi se dvokanalni enkoder koji daje 7 impulsa po okretaju motora. Uzmemo li u obzir prijenosni omjer planetarnog reduktora, možemo zaključiti da će enkoder dati 497 impulsa po jednom okretaju kotača robota.
U radu bez opterećenja motori troše ispod 150mA , te pri naponu od 12V daju nazivnu snagu od 4.22W.

Slika 20. Motor IG-32GM

Slika ispod prikazuje karakteristiku motora IG-32GM pri radnom naponu od 12V, možemo vidjeti da je maksimalan broj okretaja neopterećenog motora 6500 o/min.



Slika 21. Karakteristika motora IG-32GM na 12V

馬達單體型式/ Motor Data

定格電壓 Rated volt (V)	定格扭力 Rated torque (g-cm)	定格轉數 Rated speed (rpm)	定格電流 Rated current (mA)	無負荷轉數 No load speed (rpm)	無負荷電流 No load current (mA)	定格出力 Rated output (W)	重量 Weight (g)
12	78	5290	≤ 530	6500	≤ 150	4.22	73.0
24	74	5250	≤ 250	6500	≤ 85	3.97	73.0

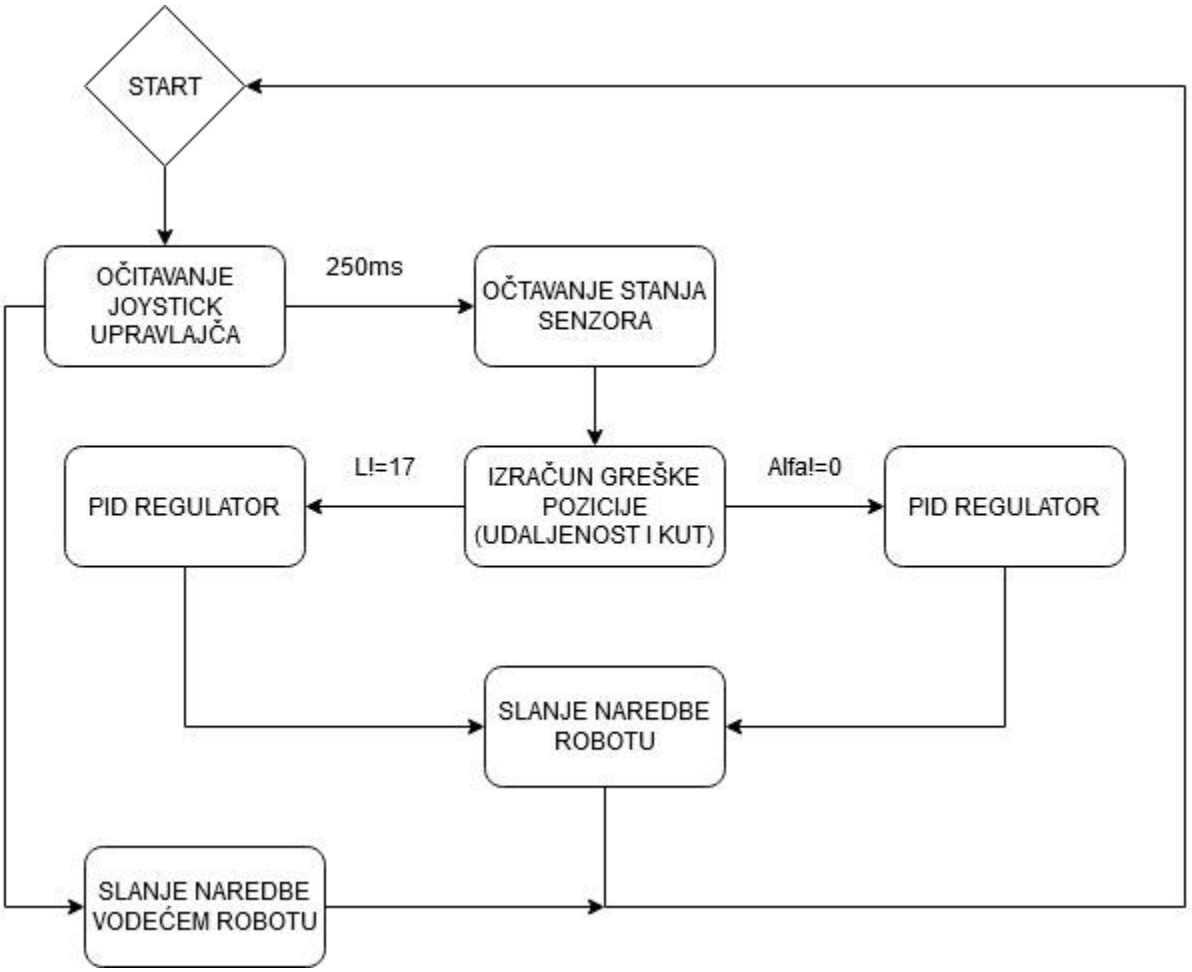
Slika 22. Podaci motora IG-32GM

Kako motori ne troše veliku snagu, za njihovo upogonjenje ugrađen je Driver LMD18200, s maksimalnom strujom od 3A. Regulacija brzine kotača odvija se u mikrokontroleru robota, brzina kotača mjeri se svakih ~260ms, dok se postavna veličina računa svakih ~100ms. Za regulaciju implementiran je PI regulator konstante proporcionalnosti K_p=1, i integratora=±1, za e≠0.

5. ALGORITAM

U skladu s gore navedenim teorijskim osnovama ostvarivanja gibanja grupe robota u formaciji, dolazimo do prijedloga izvođenja takvog gibanja. U ovom poglavlju pokriti ćemo hardwersko i softwersko rješenje za ostvarivanje zadatka gibanja dvaju eMIR edukacijskih robota, jedan iza drugoga, ili jedan pored drugoga.

Algoritam praćenja pisan je u skladu s strategijom praćenja voditelja koja je opisana u 2. poglavlju ovog rada. Slika ispod prikazuje dijagram osnovnog algoritma za gibanje robota u formaciji.



Slika 23. Algoritam

5.1 Korištena oprema

5.1.1. Arduino Mega2560

Za centralnu upravljačku jedinicu odabrana je arduino mikrokontrolerska pločica Arduino MEGA 2560, zasnovana na ATmega 2560 mikrokontroleru. Pločica je odabrana iz razloga što je sposobna upravljati kompliciranijim zadatcima, iz razloga što ima dovoljno veliku memoriju, za pohranu programa i varijabli, sa 16MHz kristalom postiže dovoljno dobru procesorsku brzinu, uz to kompaktna je i jednostavna za koristiti, sadrži veliki broj ulazno izlaznih priključaka. No ono što je bilo jedno od najvažnijih stvari u ovom projektu koje ova Arduino pločica nudi su čak četiri harverski izvedena porta za UART serijsku komunikaciju, te 5 I^2C port serijske komunikacije. Ovo nam je omogućilo jednostavno izvođenje bluetooth komunikacije s robotima i android aplikacijom za upravljanje vodećim robotom.
Pločica ima slijedeće specifikacije:

Microcontroller	**ATmega2560**
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

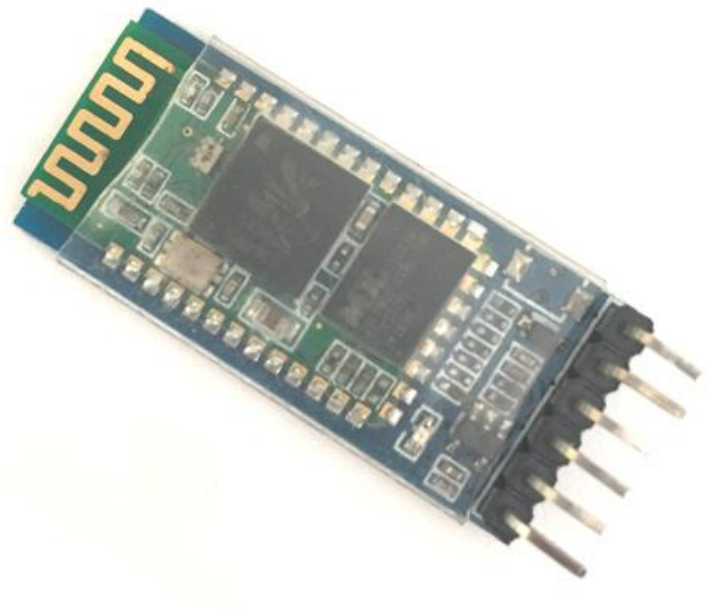
Tablica 2. Karakteristike Arduino Mega2560

Pločicu je moguće napajati usb kabelom pomoću računala, ili DC adaperom ili baterijom, što pruža mobilnost i lakoću korištenja pločice. Prilikom upload-a programa na mikrokontroler,

vidimo da uz sve potrebne programske biblioteke zauzimamo nešto manje od 10% programske memorije, memorijske lokacije globalnih varijabli zauzimaju oko 15% dinamičke memorije

5.1.2. HC-05 Bluetooth modul

Bluetooth bežična komunikacija koristi kratko-valne radio valove visoke frekvencije od 2.4 do 2.485GHz. Odabrani moduli za ovu primjenu jesu bluetooth HC-05 konfigurabilni moduli. Odlika ovih modula je da imaju mogućnost rada ili u Master ili u Slave načinu, također korištenjem specijalnog seta AT naredbi moguće je konfigurirati i sve ostale postavke modula poput brzine serijske komunikacije, naziva modula, bluetooth adrese na koju se modul spaja pri uključenju i mnogo drugih postavki.



Slika 24. HC-05 modul

Kao što je spomenuto, dva su načina rada ovog modula. Prvi način je podatkovni, u kojem modul ostvaruje komunikaciju u dva smjera (slanje/primanje) s drugim bluetooth modulom. Drugi način rada je AT komandni način rada koji se koristi za konfiguraciju postavki modula. Drugi način potreban je samo kod prve inicijalizacije modula, i aktivira se na različite načine ovisno o proizvođaču modula. Jedan od načina je držanje tipkala na samom modulu pri uključivanju modula, tada će signalna lampica dužim periodičkim blinkanjem signalizirati da je modul u AT načinu rada. Tada smo u mogućnosti komunicirati s modulom preko serijskog porta pomoću seta

AT naredbi. Po završetku konfiguracije modula dovoljno ga je resetirati, te se modul vraća u podatkovni režim rada. Donja tablica prikazuje opis funkcije pinova na HC-05 modulu.

Pin Number	Pin Name	Description
1	Enable / Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default it is in Data mode
2	Vcc	Powers the module. Connect to +5V Supply voltage
3	Ground	Ground pin of module, connect to system ground.
4	TX – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RX – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
6	State	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module • Blink once in 2 sec: Module has entered Command Mode • Repeated Blinking: Waiting for connection in Data Mode • Blink twice in 1 sec: Connection successful in Data Mode
8	Button	Used to control the Key/Enable pin to toggle between Data and command Mode

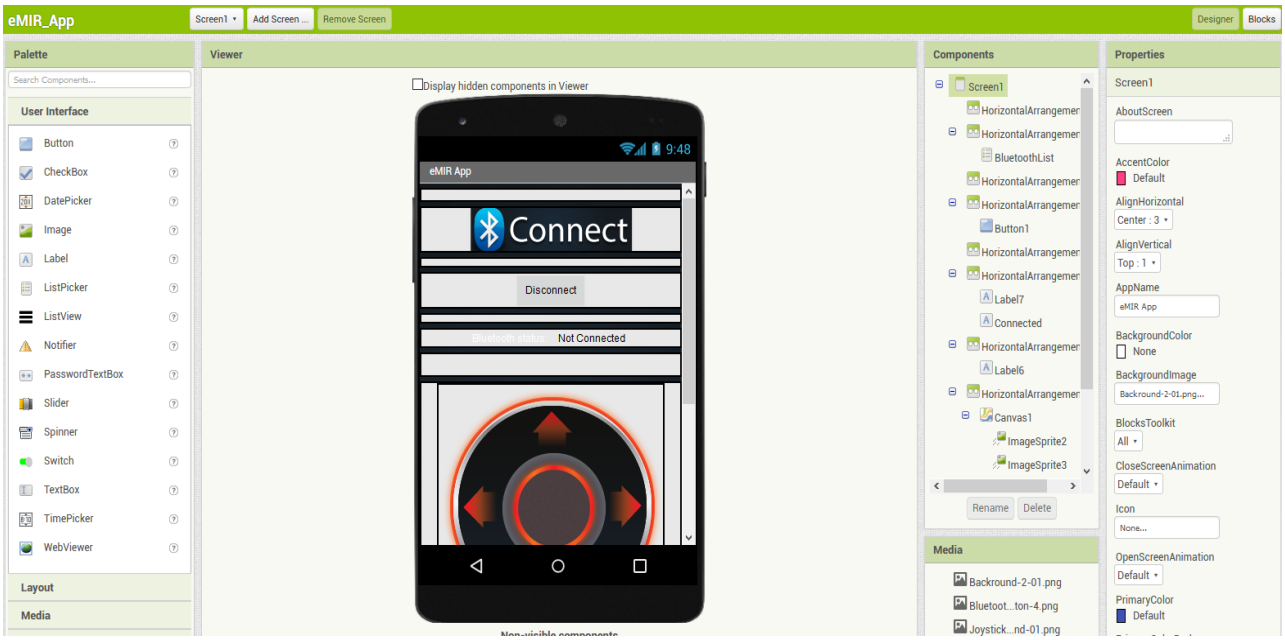
Tablica 3. HC-05 pinovi

Za izvedbu ovog projekta korištena su tri HC-05 modula, koje spajamo na Arduino pločicu. Dva modula koriste se za ostvarivanje komunikacije, tj. slanje naredbi i čitanje podataka dobivenih od dva eMIR robota, treći modul koristi se za spajanje s mobilnim uređajem koji ima instaliranu eMIR aplikaciju za upravljanje vodećim robotom.

5.1.3. eMIR App

Ova aplikacija izrađena je u svrhu zamjene fizičkog upravljača kojim bi se kontrolirao vodeći robot u formaciji. Aplikacija uspostavlja bluetooth vezu s gore navedenim modulom. Aplikacija se sastoji od virtualnog joystick upravljača, i njena jedina funkcija je očitavanje položaja upravljača u X-Y ravnini te slanje informacije o položaju na Arduino kontroler, koji dalje obrađuje te podatke i prilagođava ih u oblik paketa naredbi za eMIR edukacijske robote. Alat za izradu aplikacije je MIT App Inventor, online paket pogodan za stvaranje malih aplikacija jednostavnih funkcija. Alat je razvijen najprije od strane Googl-a, a zatim je preuzet od

MIT Media Lab-a. Alat koristi jednostavno GUI(Graphic User Interface) sučelje koje korisniku omogućava jednostavan „drag and drop“ način rada.



Slika 25. Grafičko sučelje MIT App Inventora

Završni proizvod je jednostavna aplikacija, koja nakon spajanja na bluetooth šalje informacije o položaju joysticka.

Slika 26. eMIR App

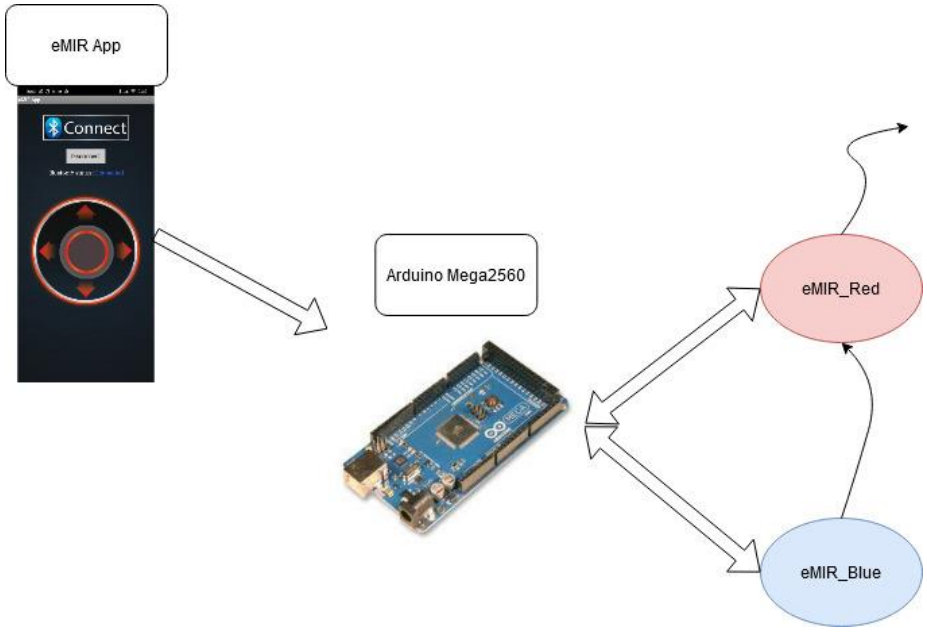
5.2. Centralizirano upravljanje

Za izvedbu zadatka, odlučeno je primijeniti koncept centraliziranog upravljanja, koje kao što je gore navedeno ima specifičnost da se sva komunikacija izvodi preko jedne centralne jednice, u našem slučaju, Arduino Mega2560 pločice.

Tako aplikacija za upravljanje vodećeg robota, komunicira direktno s centralnom jedinicom, koja potom vodećem robotu šalje prikladne naredbe za gibanje zadanom trajektorijom.

Također, i vodeći i prateći robot komuniciraju direktno s centralnom jedinicom, dobivajući od nje potrebne podatke, kao i vraćajući joj povratne pakete poruka o očitanjima.

Ovaj princip odabran je zbog bržeg i jednostavnijeg programiranja, i prilagođavanja algoritma, iz razloga što nije potrebno mijenjati nikakve parametre, ni na robotima, ni na eMIR aplikaciji, nego je dovoljno promijeniti programski kod na Arduino pločici.



Slika 27. Centralizirano upravljanje

5.3. Program

U ovom poglavlju proći ćemo dijelove programa za ostvarivanje gibanja grupe robota u formaciji. U prilogu rada, biti će dodana i programska biblioteka Robot.h za komunikaciju s dva ili više eMIR edukacijskih robota te za komunikaciju s aplikacijiom eMIR App.

Za početak potrebno je konfigurirati potrebne varijable koje ćemo koristiti u programu, te definirati programske biblioteke potrebne za izvođenje programa:

```
#include "Robot.h"
bool comStatus=true;
int X,Y,X1,Y1,X2,Y2;
//Bluetooth connection states
int state1=22, state2=23, state3=26;
//Senzori
int s10,s11,s12,s13,s14,s15;
int s20,s21,s22,s23,s24,s25,s2speed,s2mode;
int n=0;
//regulator variables
int avg_dist, ref_dist=17, reg_value_distance;
int ref_angle=0, reg_value_angle, eIntega=0, ePreva=0;;
int pGain=2, eInteg=0, ePrev=0;
float iGain=0.1, dGain=0.6;
int speed2, step2=0;
//Formation values
int min_a=-5, max_a=5;
float L=14, diff;
double alfa, var;
int Alfa;
```

Slijedeće dvije linije koda definiraju robote koje ćemo koristiti pridavajući pojedinim alijasima identifikacijski broj(1 ili 2), koje koriste programskoj biblioteci Robot.h kako bi se znalo koja naredba se šalje kojem robotu:

```
Robot robot_red(1);
Robot robot_blue(2);
```

Slijedeći dio koda, odvija se samo jednom pri pokretanju mikrokontrolera, i služi za inicijalizaciju ulaznih pinova, „state1“, „state2“ i „state3“ koji nam služe za provjeru povezanosti bluetooth modula. Program će ući u while petlju te čekati povratnu informaciju od bluetooth modula. Potom inicijaliziramo Timer4 mikrokontrolera koji će nam služiti za periodičku

aktivaciju dijela programa za komunikaciju s robotom i izračunavanje postavnih veličina regulatora frekvencijom od 5 Hz:

```
void setup() {
  pinMode(state1, INPUT);
  pinMode(state2, INPUT);
  pinMode(state3, INPUT);
  //Wait for bluetooth to connect
  while(state1!=HIGH && state2!=HIGH  && state3!=HIGH){
    delay(10);
 }
 //Start communication
  Serial.begin(57600);
  Serial1.begin(57600);
  Serial2.begin(57600);
  Serial3.begin(57600);

  //Setup timer interrupt
  //Using timer 4
  cli();//stop interrupts
  //set timer4 interrupt at 1Hz
  TCCR4A = 0;// set entire TCCR1A register to 0
  TCCR4B = 0;// same for TCCR1B
  TCNT4  = 0;//initialize counter value to 0
  // set compare match register for 5hz increments
  OCR4A = 50000;//= 16000000 / (64 * 5) - 1 (must be <65536)
  // turn on CTC mode
  TCCR4B |= (1 << WGM42);
  // Set CS12 and CS10 bits for 64 prescaler
  TCCR4B |= (0 << CS42) | (1<<CS41) | (1 << CS40);
  // enable timer compare interrupt
  TIMSK4 |= (1 << OCIE4A);

  sei();//allow interrupts
```

Također u ovaj dio koda dodajemo i set dviju naredbi za svakog robota, koje će upaliti senzore pojedinog robota, i poslati naredbu za slanje povratnih informacija od robota.

```
  robot_red.sensors_on();
  robot_red.send_info();
  delay(10);
  robot_blue.sensors_on();
  robot_blue.send_info();
  delay(10);

}
```

Glavna petlja programa služi isključivo za čitanje stanja pozicije upravljača, te periodičko pokretanje pod-programa za komunikaiciju s robotima ukoliko je podignuta comStatus zastavica:

```
void loop() {
  while(Serial1.available()>0){
  robot_red.read_controller(&X,&Y);
  }
  X1=X/2;
  X1=constrain(X1,-10,10);
  Y1=Y;
  //X2=X;
  Y2=Y1;

  if(comStatus==true){
    execute_communication();
  }
}
```

Prekidna rutina timera 4, koja se pokreće svaki put kada TCNT4 registar postigne vrijednost comparatorskog registra OCR1A, ima funkciju podizanja zastavice comStatus:

```
ISR(TIMER4_COMPA_vect){//timer4 interrupt 5Hz
 comStatus=true;
}
```

Slijedeći potprogram najprije će očitati stanja senzora na robotima, spremiti ih u odgovarajuće proračunske varijable, neke od kojih će ispisati na Serial Monitoru kako bi se mogle pratiti vrijednosti varijabli. Potom će iz očitanih vrijednosti izračunati udaljenost između dva robota „avg_dist“, te kut pod kojim se jedan robot nalazi u odnosu na drugi „Alfa“. Potom se pozivaju potprogrami PID regulatora za linearnu udaljenost „PID_dist“, te regulatora za kompenzaciju greške kutne pozicije „PID_angle“. Vrijednosti regulatora pribrajaju se potom postavnim vrijednostima translacijske i rotacijske brzine, te se na poslijetku šalju naredbe robotima:

```
void execute_communication(){
  if(n<10){
    robot_blue.send_info();
    n++;}

  while(Serial2.available()>0){
    robot_red.read_data();
  }


  while(Serial3.available()>0){
    robot_blue.read_data();
 }
  s20=robot_blue.aa;
  s21=robot_blue.bb;
  s22=robot_blue.cc;//cc-jedan iza drugog, ee-jedan pored drugog
  s2mode=robot_blue.mm;
```

```
  Serial.print("S20=");
  Serial.println(s20);
  Serial.print("s21=");
  Serial.println(s21);
  Serial.print("s22=");
  Serial.println(s22);


  avg_dist=(s21+s22)/2;
  reg_value_distance=PID_dist(avg_dist, ref_dist);

  speed2=Y2+reg_value_distance;
  Serial.print("Distance regulator:")
  Serial.println(reg_value_distance);

  speed2=constrain(speed2,-100,100);
  Serial.print("Robot 2 speed:")
  Serial.println(speed2);

  if(s22>s21){diff=s22-s21; var=(diff)/L;}
  else if(s22<s21){diff=s21-s22; var=-(diff)/L;}
  alfa=asin(var);
  alfa=alfa*(180/PI);
  Alfa=alfa;
  if(s22>65){Alfa=50;}
  if(s21>65){Alfa=-50;}
  Serial.print("alfa=");
  Serial.println(Alfa);


  if(Alfa>min_a && Alfa<max_a){Alfa=0;}
  reg_value_angle=PID_angle(Alfa, ref_angle);
  Serial.print("Angle regulator")
  Serial.println(reg_value_angle);


  robot_red.send_cmd_move(Y1,X1);
  Serial.print("distance=");
  Serial.println(avg_dist);

  //Compensate for distance
  if(avg_dist!=17 && s22<65 && s21<65){
  robot_blue.send_cmd_move(speed2,0);
  }
  //Then compensate angle
  else if(s2mode!=2){
  robot_blue.send_cmd_move(0,(reg_value_angle)/2);
  }


  comStatus=false;
}
```

Potprogrami PID regulatora za kompenzaciju linearne i kutne pogreške s konstantama regulatora „pGain“, „iGain“ i „dGain“:

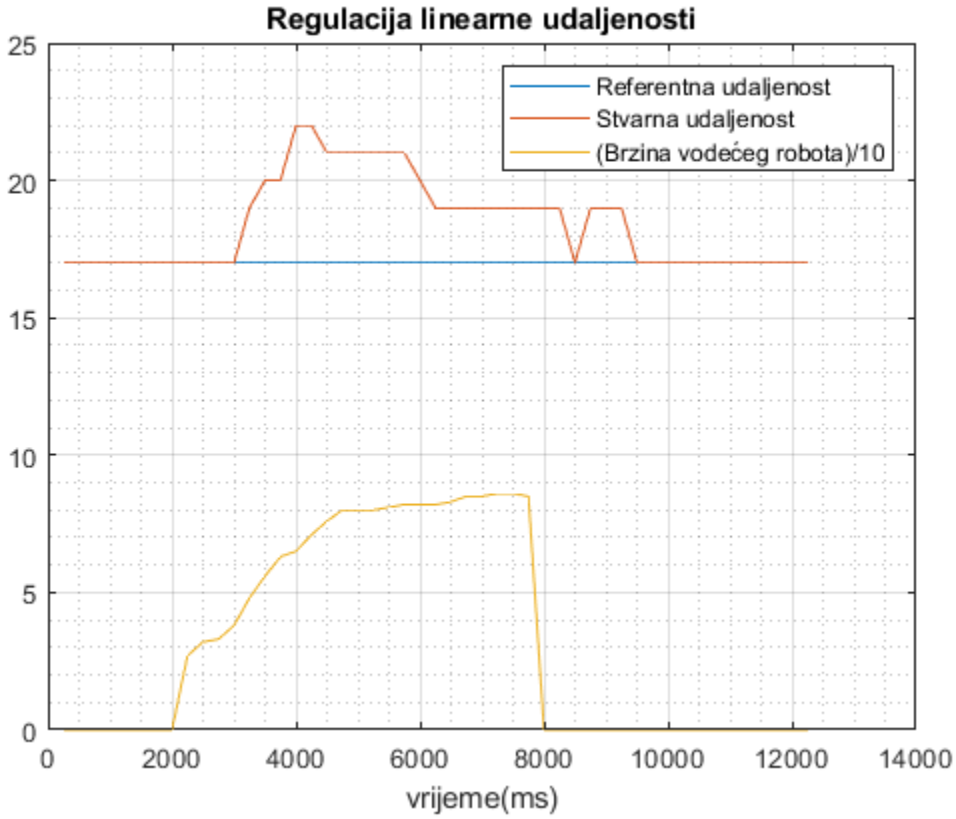
```
//distance PID regulator
int PID_dist(unsigned int reg_value, int cur_value){
  int error;
  int pid_value;
  error=reg_value-cur_value;
  pid_value=(pGain*error)+(iGain*eInteg)+(dGain*(error-ePrev));
  pid_value=constrain(pid_value,-50,50);
  eInteg+=error;
  ePrev=error;
  if(avg_dist<19 && avg_dist>15){pid_value=0;
  eInteg=0;
  ePrev=0;
  avg_dist=17;}

  return pid_value;

}

//Angle PID regulator
int PID_angle(unsigned int reg_valuea, int cur_valuea){
  int errora;
  int pid_valuea;
  errora=reg_valuea-cur_valuea;
  pid_valuea=(pGain*errora)+(iGain*eIntega)+(dGain*(errora-ePreva));
  pid_valuea=constrain(pid_valuea,-50,50);
  eIntega+=errora;
  ePreva=errora;
  if(Alfa>min_a && Alfa<max_a){pid_valuea=0;
  eIntega=0;
  ePreva=0;
  }

  return pid_valuea;

}
```

5.4. Rezultati regulacije

Gore navedeni PID regulatori za regulaciju linearne udaljenosti i pogreške kuta formacije daju zadovoljavajuće rezultate regulacije, uzmemo li u obzir vrstu senzora koja daje očitanje(infracrveni daljinomjeri). Naime, pri očitavanju udaljenosti između dva robota, dolazi do oscilatornih mjerenja, jer u stacionarnom stanju senzori ne daju savršeno točno mjerenje, već dolazi do oscilacija. Iz tog razloga za reguliranje linearne udaljenosti, računa se prosječna udaljenost, izmjerena na dva senzora, kako bi se smanjile oscilacije očitanja.

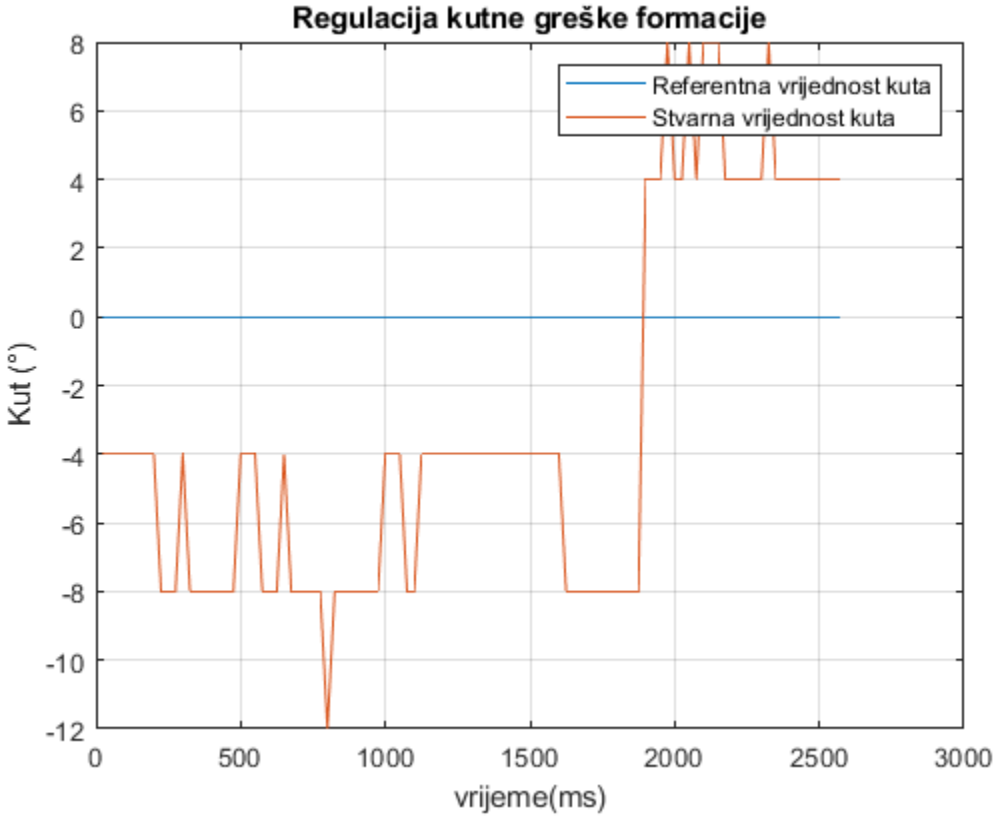
S druge strane, pri regulaciji i računanju kuta između dva robota, dolazi do velikih odstupanja iznosa kuta prilikom računanja upravo zbog ovakvih oscilatornih mjerenja. Iz tog razloga, pri regulaciji greške kuta formacije u programu je uvedeno tolerancijsko polje kuta od ±4 stupnja, kako bi se otklonilo oscilatorno djelovanje regulatora.

Prikaz regulacije za oba parametra nalazi se na slikama ispod:



Slika 28. Regulacija linearne udaljenosti

Kao što je vidljivo na grafu iznad, referentna udaljenost između robota iznosi 17cm, dok je naravno referentna vrijednost kuta formacije 0 stupnjeva, kao što je vidljivo na slijedećem prikazu regulacije kuta. Također treba primjetiti šiljast graf stvarne vrijednosti kuta, koji nastaje upravo zbog gore spomenutih oscilatornih očitanja infracrvenih senzora. Također je jasno vidljiv i pojas tolerancije.



Slika 29. Regulacija kutne greške formacije

6. ZAKLJUČAK

PID regulacija linearne udaljenosti i pogreške kutne pozicije formacije nudi dobro ponašanje robota pri gibanju u formaciji jedan iza drugoga, ili jedan pored drugoga. Regulatori nude brzu eliminaciju greške regulacije, a podešavanjem faktora regulatora može se postići ujednačena promjena brzine robota. Greške koje nastaju u održavanju formacije najprije su vezane uz infracrvene daljinomjere koji ipak u svojim očitanjima daju poprilično oscilirajuće vrijednosti, što nam sprječava precizno postizanje položaja robota te smo primorani koristiti srednje vrijednosti očitanja, kao i uvesti prag regulacije koji će otkloniti pretjerane oscilacije u regulaciji. Također problemi se javljaju pri pokušaju ostvarivanja kompleksnijih trajektorija gibanja vodećeg robota, poput naglog skretanja u stranu, gdje prateći robot, što zbog svoje konstrukcije, što zbog senzora koje smo koristili, u pokušaju praćenja postavljene trajektorije izgubi vezu s vodećim robotom, te dolazi do raspada formacije.

Ovaj problem najprije bi se mogao riješiti korištenjem druge vrste senzora, na primjer kamere, ili LIDAR sustava s očitanjem od 360^{o}, jer u ovakvim situacijama roboti moraju izvesti manevre koje nije moguće kontrolirati samo s dva infracrvena senzora.

LITERATURA

[1]- Prof.dr.sc. Mladen Crneković- educational Mobile Intelligent Researcher

[2]- Rashid, Abdulmuttalib & Abdulrazaaq, Bayader. (2019). A Survey of Multi-mobile Robot Formation Control. International Journal of Computer Applications. 181. 12-16. 10.5120/ijca2019918651.

[3] Madhevan, Sreekuman, Tracking Algorithm Using Leader Follower Approach for Multi Robots. International Conference on Design And Manafacturing, IConDM 2013.

PRILOZI

I. CD-R disc
II. Programski kod biblioteke „Robot.h“

Header file programske biblioteke „Robot.h“:

```
/*
 * Robot.h- Library for serial bluetooth comunication
 * with eMIR mobile Robots
 */

 #ifndef Robot_h
 #define Robot_h

 #include "Arduino.h"


 class Robot
 {
  public:
  Robot(int robot_id);
  void send_cmd_move(int v, int r);
  void send_cmd_translate(int path, int sp);
  void send_cmd_rotate(int angle, int sp);
  void set_min_distance(int D);
  void set_max_speed(int Max_speed);
  void set_max_rotation(int Max_rotation);
  void set_azimut();
  void cmd_stop();
  void reset_counters();
  void set_digital_output(int num, int state);
  void beep(int t);
  void send_info();
  void no_send_info();
  void sensors_on();
  void sensors_off();
  void turn_off();
  void read_controller(int* xAxis, int* yAxis);
  void read_data();
  int xAxis;
  int yAxis;
  int CS;
  int aa, bb, cc, dd, ee, ff, uu, vv, rr, gg, hh, mm, jj, kk, pp, ss, CSC, q,
p;
  int Speed, Rotation, LeftPWM, RightPWM, Path, Angle, WorkMod, Azimut;
  char SX;
  int _robot_id;
  int ID;
  String CMD;



  private:
```

```
  uint8_t zero=00;
  void byte_to_str(char* buff, int val);
  char nibble_to_hex(int nibble);
  void send_command(char command[]);
  int CheckSum(int Sum);
  char _robot_cmd;
  char _cmd_start='#';
  char _cmd_end='/';
  char Buffer[];
  int _CS;
  int _v;
  int x=0;
  int y=0;
  String _aa, _bb, _cc, _dd, _ee, _ff, _uu, _vv, _rr, _gg, _hh, _mm, _jj,
_kk, _pp, _ss, _CSC, _q, _p;
  byte index;
  char inChar;
  char _CMD[36];
  bool started=false;
  bool ended=false;
  #define SOM '*'
  #define EOM '/'
  };
 #endif
```

Programski kod biblioteke „Robot.h“:

```
/*
 * Robot.cpp Library for serial bluetooth comunication
 * with eMIR mobile Robots
 * Serial2 robot number one(red)
 * Serial3 robot number two(blue)
 */
 #include <stdlib.h>
 #include <string.h>
 #include "Arduino.h"
 #include "Robot.h"

 Robot::Robot(int robot_id)
 {
  ID=robot_id;
  }

 void Robot::send_cmd_move(int v, int r)
 {
  char _robot_cmd[10]="";
  _robot_cmd[0]=_cmd_start;
  byte_to_str(&_robot_cmd[1], 01);
  byte_to_str(&_robot_cmd[3], v);
  byte_to_str(&_robot_cmd[5], r);
  CS=CheckSum(0x01+v+r);
  byte_to_str(&_robot_cmd[7], CS);
  _robot_cmd[9]=_cmd_end;

  send_command(_robot_cmd);
```

```
}


void Robot::send_cmd_translate(int path, int sp)
{
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 02);
 byte_to_str(&_robot_cmd[3], path);
 byte_to_str(&_robot_cmd[5], sp);
 CS=CheckSum(0x02+path+sp);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
 }

void Robot::cmd_stop()
{
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 00);
 byte_to_str(&_robot_cmd[3], 00);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x00);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
 }

void Robot::send_cmd_rotate(int angle, int sp){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 03);
 byte_to_str(&_robot_cmd[3], (angle/2));
 byte_to_str(&_robot_cmd[5], sp);
 CS=CheckSum(0x03+(angle/2)+sp);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
}

void Robot::set_min_distance(int D){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 11);
 byte_to_str(&_robot_cmd[3], D);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x11+D);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
}

void Robot::set_max_speed(int Max_speed){
```

```
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 12);
 byte_to_str(&_robot_cmd[3], Max_speed);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x12+Max_speed);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
}

void Robot::set_max_rotation(int Max_rotation){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 13);
 byte_to_str(&_robot_cmd[3], Max_rotation);
 byte_to_str(&_robot_cmd[5], zero);
 CS=CheckSum(0x13+Max_rotation);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
}

void Robot::set_azimut(){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 20);
 byte_to_str(&_robot_cmd[3], 00);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x14);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
 }

 void Robot::reset_counters(){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 21);
 byte_to_str(&_robot_cmd[3], 00);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x15);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
 }

void Robot::set_digital_output(int num, int state){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 06);
 byte_to_str(&_robot_cmd[3], num);
 byte_to_str(&_robot_cmd[5], state);
 CS=CheckSum(0x06+num+state);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
```

```
}

void Robot::beep(int t)
{
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 04);
 byte_to_str(&_robot_cmd[3], t);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x04+t);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
}

void Robot::send_info(){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 16);
 byte_to_str(&_robot_cmd[3], 01);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x10+0x01);
 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);

}

 void Robot::no_send_info(){
   char _robot_cmd[10]="";
   _robot_cmd[0]=_cmd_start;
   byte_to_str(&_robot_cmd[1], 16);
   byte_to_str(&_robot_cmd[3], 00);
   byte_to_str(&_robot_cmd[5], 00);
   CS=CheckSum(0x10+0x00);
   byte_to_str(&_robot_cmd[7], CS);
   _robot_cmd[9]=_cmd_end;
   send_command(_robot_cmd);
}

void Robot::sensors_on(){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 05);
 byte_to_str(&_robot_cmd[3], 01);
 byte_to_str(&_robot_cmd[5], 00);
 CS=CheckSum(0x05+0x01);

 byte_to_str(&_robot_cmd[7], CS);
 _robot_cmd[9]=_cmd_end;
 send_command(_robot_cmd);
}

void Robot::sensors_off(){
 char _robot_cmd[10]="";
 _robot_cmd[0]=_cmd_start;
 byte_to_str(&_robot_cmd[1], 05);
```

```
  byte_to_str(&_robot_cmd[3], 00);
  byte_to_str(&_robot_cmd[5], 00);
  CS=CheckSum(0x05);
  byte_to_str(&_robot_cmd[7], CS);
  _robot_cmd[9]=_cmd_end;
  send_command(_robot_cmd);

 }

 void Robot::turn_off(){
  char _robot_cmd[10]="";
  _robot_cmd[0]=_cmd_start;
  byte_to_str(&_robot_cmd[1], 255);
  byte_to_str(&_robot_cmd[3], 00);
  byte_to_str(&_robot_cmd[5], 00);
  CS=CheckSum(0xFF);
  byte_to_str(&_robot_cmd[7], CS);
  _robot_cmd[9]=_cmd_end;
  send_command(_robot_cmd);
  }


 int Robot::CheckSum(int SUM){
  _CS=lowByte(256-lowByte(SUM));
  return _CS;
  }

  char Robot::nibble_to_hex(int nibble) {  // convert a 4-bit nibble to a
hexadecimal character
  nibble &= 0xF;
  return nibble > 9 ? nibble - 10 + 'A' : nibble + '0';
  }

  void Robot::byte_to_str(char* buff, int val) {  // convert an 8-bit byte to
a string of 2 hexadecimal characters
  buff[0] = nibble_to_hex(val >> 4);
  buff[1] = nibble_to_hex(val);
  }

  void Robot::send_command(char command[]){

  if(ID==1){
    for(int i=0; i<10;i++)
      Serial2.write(command[i]);}
  if(ID==2){
    for(int i=0; i<10;i++){
      Serial3.write(command[i]);}
    }

  }

  void Robot::read_controller(int* xAxis, int* yAxis){
  while (Serial1.available()>=2){
    x=Serial1.read();
    delay(10);
    y=Serial1.read();
    if(x>60 & x<220){
```

```
      *xAxis=map(x,220,60,-100,100);
      }
    if(y>60 & y<220){
      *yAxis=map(y,220,60,-100,100);
      }

  }
  return;
  }

  void Robot::read_data(){
  if(ID==1){
  while(Serial2.available()>0){
    char inChar=Serial2.read();
    if(inChar==SOM){
      index=0;
      _CMD[index]='\0';
      started=true;
    }
    else if(inChar==EOM){
      ended=true;
      break;
    }
    else{
      if(index<35){
        _CMD[index]=inChar;
        index++;
        _CMD[index]='\0';
      }
    }
  }
  }
  //Reading data from robot two
  if(ID==2){
  while(Serial3.available()>0){
    char inChar=Serial3.read();
    if(inChar==SOM){
      index=0;
      _CMD[index]='\0';
      started=true;
    }
    else if(inChar==EOM){
      ended=true;
      break;
    }
    else{
      if(index<35){
        _CMD[index]=inChar;
        index++;
        _CMD[index]='\0';
      }
    }
  }
  }

  if(started && ended){
    started=false;
```

```
ended=false;
index=0;
CMD=_CMD;
_CMD[index]='\0';
_aa=CMD.substring(0,2);
_aa.toCharArray(Buffer, 3);
aa=strtol(Buffer, NULL,16);

_bb=CMD.substring(2,4);
_bb.toCharArray(Buffer, 3);
bb=strtol(Buffer, NULL,16);

_cc=CMD.substring(4,6);
_cc.toCharArray(Buffer, 3);
cc=strtol(Buffer, NULL,16);

_dd=CMD.substring(6,8);
_dd.toCharArray(Buffer, 3);
dd=strtol(Buffer, NULL,16);

_ee=CMD.substring(8,10);
_ee.toCharArray(Buffer, 3);
ee=strtol(Buffer, NULL,16);

_ff=CMD.substring(10,12);
_ff.toCharArray(Buffer, 3);
ff=strtol(Buffer, NULL,16);

_uu=CMD.substring(12,14);
_uu.toCharArray(Buffer, 3);
uu=strtol(Buffer, NULL,16);

_vv=CMD.substring(14,16);
_vv.toCharArray(Buffer, 3);
vv=strtol(Buffer, NULL,16);

_rr=CMD.substring(16,18);
_rr.toCharArray(Buffer, 3);
rr=strtol(Buffer, NULL,16);

_gg=CMD.substring(18,20);
_gg.toCharArray(Buffer, 3);
gg=strtol(Buffer, NULL,16);

_hh=CMD.substring(20,22);
_hh.toCharArray(Buffer, 3);
hh=strtol(Buffer, NULL,16);

_mm=CMD.substring(22,24);  //mode 2--translate //mode 3--rotate
_mm.toCharArray(Buffer, 3);
mm=strtol(Buffer, NULL,16);

_jj=CMD.substring(24,26);
_jj.toCharArray(Buffer, 3);
jj=strtol(Buffer, NULL,16);

_kk=CMD.substring(26,28);
```

```
    _kk.toCharArray(Buffer, 3);
    kk=strtol(Buffer, NULL,16);

    _pp=CMD.substring(28,30);
    _pp.toCharArray(Buffer, 3);
    pp=strtol(Buffer, NULL,16);

    _ss=CMD.substring(30,32);
    _ss.toCharArray(Buffer, 3);
    ss=strtol(Buffer, NULL,16);

    _CSC=CMD.substring(32,34);
    _CSC.toCharArray(Buffer, 3);
    CSC=strtol(Buffer, NULL,16);

    if(vv<128){Robot::Speed=vv;}
    else{Robot::Speed=vv-256;}
    if(rr<128){Robot::Rotation=rr;}
    else{Robot::Rotation=rr-256;}
    if(gg<128){Robot::LeftPWM=gg;}
    else{Robot::LeftPWM=gg-256;}
    if(hh<128){Robot::RightPWM=hh;}
    else{Robot::RightPWM=hh-256;}
    if(pp<128){Robot::Path=round(pp*1.03);}
    else{Robot::Path=round((pp-256)*1.03);}
    if(ss<128){Robot::Angle=round(ss*2.14);}
    else{Robot::Angle=round((ss-256)*2.14);}
    Robot::WorkMod=mm;
    Robot::Azimut=2*kk;

  }
  }
```