Projektiranje servera mobilnog robota

Prević, Matija

Undergraduate thesis / Završni rad

2010

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:284536

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-08**

Repository / Repozitorij:

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

Sveučilište u Zagrebu

Fakultet strojarstva i brodogradnje

ZAVRŠNI RAD

Voditelj rada:

prof. dr. sc. Mladen Crneković

Matija Prević

Zagreb, 2010.

Sveučilište u Zagrebu

Fakultet strojarstva i brodogradnje

ZAVRŠNI RAD

Matija Prević

Zagreb, 2010.

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: **MATIJA PREVIĆ** Mat. br.: 0035158367

Naslov: **PROJEKTIRANJE SERVERA MOBILNOG ROBOTA**

Opis zadatka:

Mobilni robot pogonjen s dva DC motora opremljen je infracrvenim daljinomjerima i pokretnom kamerom. Njegovo programiranje zahtijeva tzv. "low-level" vještine kojima se većina korisnika ne opterećuje, već želi programirati ponašanje robota na višoj razini. Zbog toga je potrebno osmisliti i napisati serverski program koji će ispunjavati naredbe više razine. Mobilni robot je povezan s "host" PC računalom preko bežične dvosmjerne serijske veze.
Potrebno je:

- Definirati i programirati skup serverskih naredbi
- Napisati programski paket za podršku serverskih naredbi na strani "host" računala
- Testirati softver s nekoliko algoritama ponašanja mobilnog robota

Zadatak zadan: 11. prosinca 2009.

Rok predaje rada: Prosinac 2010.

Zadatak zadao: Prof. dr. sc. Mladen Crneković

Predsjednik Povjerenstva: Prof. dr. sc. Dubravko Majetić

Roditelj me stvorio; učitelj me učinio čovjekom.

Inazo Nitobe, *Bushido - Kodeks Samuraja*

Izjava

Izjavljujem da sam izradio ovaj rad samostalno uz stručnu pomoć mentora.

M.P.

Zahvala

Zahvaljujem mentoru prof. dr. sc. Mladen Crnekoviću na korisnim savjetima, diskusijama, strpljenju i mentorstvu tijekom izrade ovoga rada.

Zahvaljujem se i obitelji, posebice bratu Luki, na strpljenju, požrtvovnosti i vjeri u mene, kako tijekom dosadašnjeg studija, tako i pri izradi ovog rada.

M.P.

Sažetak

Svrha ovog rada, osim edukacije, je poboljšati način upravljanja mobilnim robotom. Napravljen je serverski program koji pomoću bežične dvosmjerne serijske veze komunicira s robotom. Preko te aplikacije moguće je upravljati robotom teleoperacijski preko igraće palice ili uključiti autonomni rad.

Ključne riječi: mobilni robot, programiranje, server, delphi, C

Sadržaj

Popis slika

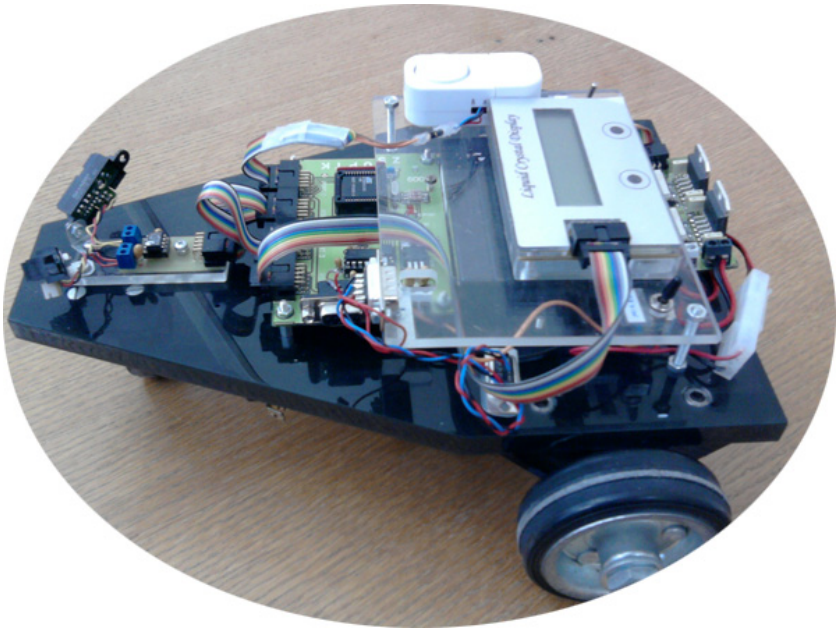
1

Uvod

Prednost mobilnih robota je upravo u mogućnosti kretanja u datom prostoru. Time nude fleksibilnu primjenu svojih sposobnosti u industriji, istraživanju novih prostora, vojsci, zabavi, i tako dalje. Najzanimljivija primjena mobilnih robota vjerojatno je ona vozila Spirit i Opportunity za istraživanje površine planeta Marsa, ili malog Upuaut-a Rudolfa Gatebrinka za istraživanje kanala Velike piramide u Gizi. U svakodnevnici tu su Helpmate roboti za transport hrane i lijekova unutar bolnica, te roboti za usisavanje. Kod nas su najpoznatija vozila za razminiravanje tvrtke DOK-ING d.o.o. Prema sustavu za pokretanje, mobilni se roboti dijele na one pokretane kotačima, nogama, bespilotne letjelice i podvodne robote. Daleko najpopularniji su mobilni roboti pokretani kotačima. Prema sustavu za navođenje mogu se podijeliti na teleoperacijske i autonomne mobilne robote. Može se reći da se teleoperacijski mobilni roboti češće koriste tamo gdje čovjek ne može ići, dok autonomni mobilni roboti dijele prostor s čovjekom i oslobađaju ga neke radnje.

U ovom se radu koristi već postojeći mobilni robot (slika 1.1) u vlasništvu fakulteta, pokretan kotačima u tipičnoj diferencijalnoj strukturi, navođen i teleoperacijski i autonomno. Na njega je bilo potrebno downloadati program napisan u C jeziku koji bi njime upravljao. Broj

algoritama ponašanja za autonomni rad koji se mogao spremiti na robota ovisio je o veličini memorije mikroprocesora robota. Mogućnost promjene algoritama ovisila je o broju tipki na daljinskom upravljaču. Također je postojala mogućnost teleoperacijskog upravljanja pomoću daljinskog upravljača, što je prilično grubo i nekontinuirano upravljanje. Cilj ovog rada je bilo dovesti to upravljanje na novu razinu. Prvi korak je prebaciti računalne operacije sa robota na osobno računalo koje ima mnogo više resursa i mogućnosti za potrebne operacije. Zatim se rješenja tih operacija trebaju prenjeti na robota. Za komunikaciju se koristi bežična bluetooth veza. Navedeno je dostatno za novi način autonomnog rada robota, no najveća prednost se ističe trećim korakom, a to je korištenje igraće palice (joystick) za fino kontinuirano teleoperacijsko upravljanje. Korisnik može birati kako želi upravljati robotom, te dobiva uvid u situaciju robota s infracrvenih daljinomjera, preko jednostavnog sučelja serverske aplikacije.



Slika 1.1: Korišteni mobilni robot

Rad je izložen u četiri dijela. Prvo su ukratko opisane hardverske komponente bitne za programiranje mobilnog robota. Slijedi softver korišten pri radu na završnom zadatku. Najbitniji dio je opis serverskog programa MobRob, a na kraju je opisan program na strani robota.

2

Komponente i struktura

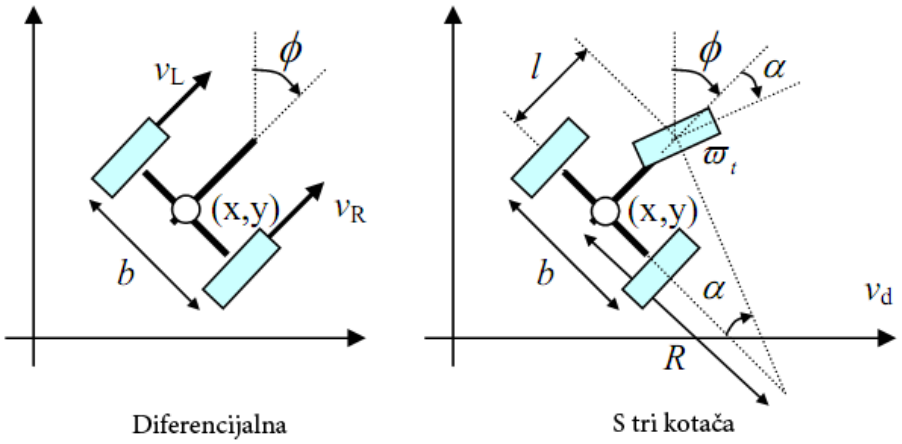
Za potrebe ovoga rada korišteni su mobilni robot, netbook s MobRob aplikacijom, RS-232 USB na serijsku vezu adapter, 2 bluetooth adaptera, te igraća palica.

2.1 Mobilni robot

Dijelovi mobilnog robota bitni pri programiranju servera MobRob su sustav za pokretanje, mikroprocesor, infracrveni daljinomjeri i LCD ekran.

2.1.1 Sustav za pokretanje

Kao što je spomenuto u uvodu, korišten je robot s kotačima takozvane diferencijalne strukture. Ta struktura je prikazana na slici 2.1 uz drugu najčešću strukturu kod mobilnih robota ovog tipa.



Slika 2.1: Dvije tipične strukture mobilnih robota pokretanih kotačima

Za taj tip mobilnog robota kinematička jednadžba glasi:

$$v = \frac{v_R + v_L}{2} = \frac{\omega_R + \omega_L}{4} d \tag{2.1}$$

$$\omega = \frac{v_R + v_L}{b} = \frac{\omega_R + \omega_L}{2b} d \tag{2.2}$$

gdje je b udaljenost između pogonskih kotača, d je promjer pogonskih kotača, v i ω su odgovarajuće linearne i kutne brzine mobilnog robota, a ω_L i ω_R su kutne brzine lijevog i desnog kotača. Za upravljanje mobilnim robotom važne su upravo varijable ω_L i ω_R, te se iz (2.1) i (2.2) dobiva:

$$\omega_L = \frac{2v - b\omega}{d} \tag{2.3}$$

$$\omega_R = \frac{2v + b\omega}{d} \tag{2.4}$$

Kada se iz željenih brzina robota v i ω dobiju kutne brzine kotača ω_L i ω_R, još je potrebno prilagoditi njihov iznos vrijednosti impulsno širinske modulacije drivera motora, što je prikazano jednadžbama 2.5 i 2.6. Ta se vrijednost kreće između 0 i 255, gdje je 0 maksimalna brzina. Konačno se vrijednosti PWM šalju robotu u paketu bežično preko serijske veze.

$$PWM_L = (1 - \frac{\omega_L}{\omega_{max}}) * 255 \tag{2.5}$$

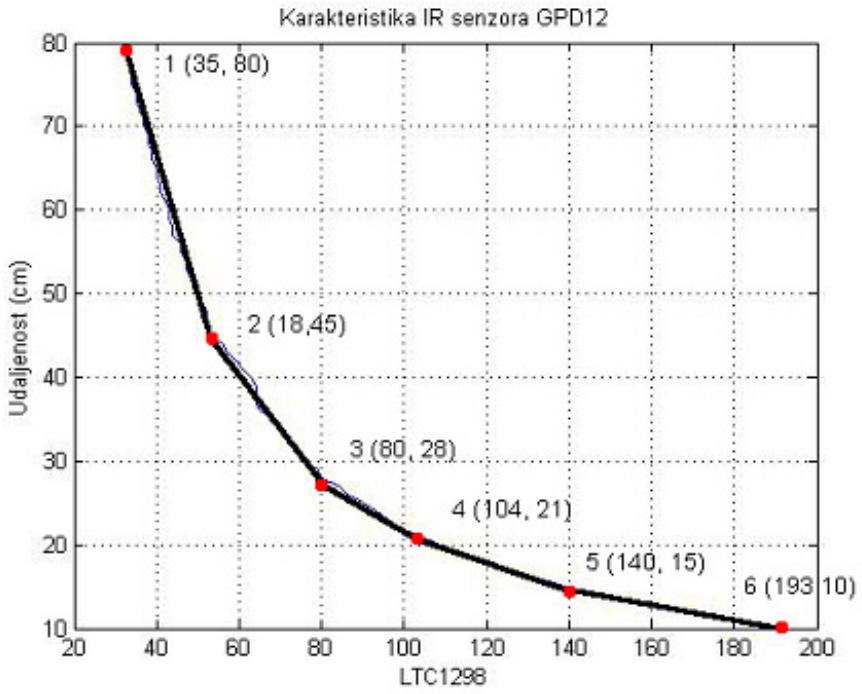
$$PWM_R = (1 - \frac{\omega_R}{\omega_{max}}) * 255 \tag{2.6}$$

2.1.2 Mikroprocesor

Ugrađen je mikroprocesor tvrtke Atmel AT89C51ID2 s kojim smo se susreli i radili u sklopu redovne nastave na usmjerenju za mehatroniku i robotiku. Mentor Mladen Crneković napisao je dodatnu biblioteku kako bi se u C jeziku moglo komunicirati preko serijske veze, s infracrvenim daljinomjerima, driverima motora i LCD ekranom.

2.1.3 Infracrveni daljinomjeri

Kako bi se omogućio autonomni način rada mobilnog robota potrebno je da ima osjetila - senzore. Korišteni robot sadrži samo GP2D12 infracrvene daljinomjere tvrtke Sharp na svom prednjem kraju. Jedan je postavljen blago ulijevo, a drugi udesno. Pouzdano područje rada daljinomjera je između 15 cm i 80 cm. Kalibraciju senzora je izvršio Josip Kos u svojem završnom radu [2]. Linearizirani karakteristiku senzora je prikazao na slici 2.2. Na apscisi je prikazana funkcija $LTC1298()$ koja daje vrijednost senzora, a na ordinati približna udaljenost od zapreke u cm pri toj vrijednosti.



Slika 2.2: Linearizirana karakteristika senzora

2.1.4 LCD ekran

Na ekran se ispisuje smjer kretanja robota, te vrijednost funkcije *LTC*1298() koja vraća očitanja sa infracrvenih daljinomjera. Svrha ekrana je više u provjeri i ispravljanju programa, nego u samoj uporabi od strane korisnika MobRob sučelja. Ekran može prikazati sveukupno 32 znaka u dva reda.

2.2 Računalo i komponente

Korišten je Toshibin netbook NB100 za podržavanje serverskog programa MobRob. Oba USB utora su iskorištena za igraću palicu i RS-232 USB na serijsku vezu adapter. Potonji se mora postaviti u točno određen USB port (COM8) da bi aplikacija MobRob radila[1]. Na njega se postavlja drugi adapter, s RS-232 na bluetooth, podešen na 9600 impulsa po sekundi (baud rate). Identično podešen drugi bluetooth adapter se postavlja na mobilnog robota u za to već predviđen serijski utor. Komponente su prikazane na slici 2.3.

Slika 2.3: Korišteno računalo i potrebne komponente

[1]Nije bilo potrebe za prilagodbom jer je svrha rada isključivo edukativna

3

Korišten softver

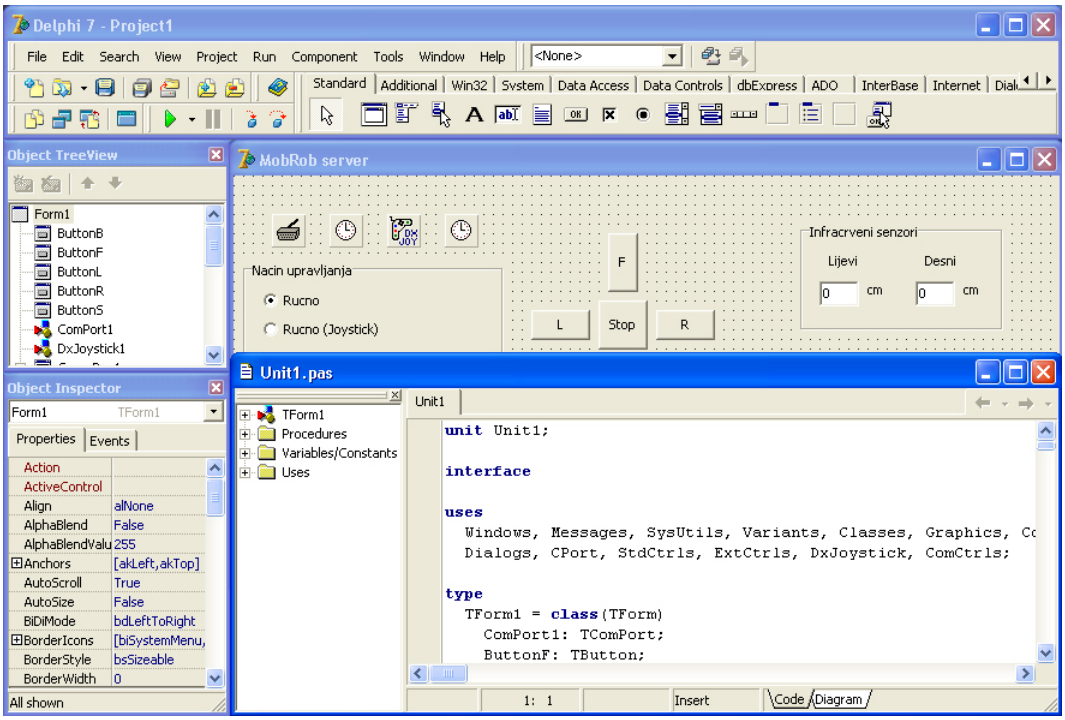
Pri projektiranju servera za mobilnog robota bilo je potrebno više programskih paketa. Potreban je programski paket kojim se može napraviti server za bežičnu komunikaciju s robotom za Windows operativni sustav. Zatim je potreban programski paket za programiranje mikrokontrolera robota koji će primati naredbe sa servera i slati povratne informacije. I na kraju je potreban programski paket koji će taj program prebaciti na robota. Ta 3 korištena alata su opisana u ovome poglavlju.

3.1 Delphi

Za programiranje samog servera korišten je Borlandov programski paket Delphi 7 Personal Edition za Windows operativne sustave. Delphi programski jezik, znan i kao Objektni Pascal, temeljen je na proširenom objektno orijentiranom Pascal programskom jeziku. Delphi 7 je jedna od najuspješnijih Borlandovih integriranih razvojnih okolina[1] zbog svoje stabilnosti, brzine i malih zahtjeva. Kao i uobičajeno svaka integrirana razvojna okolina, sastoji se od source kod editora, kompajlera, debuggera, uz dodatak grafičkog "drag and drop" dizajniranja

[1] IDE - Integrated Development Environment

forme s paletom komponenata, i tako dalje. Sučelje je prikazano na slici 3.1. Usprkos tome što je izdan "davne" 2002., u ovom radu je izabran jer je to zadnja verzija koja ima mogućnost korištenja bez aktivacije za komercijalnu upotrebu.

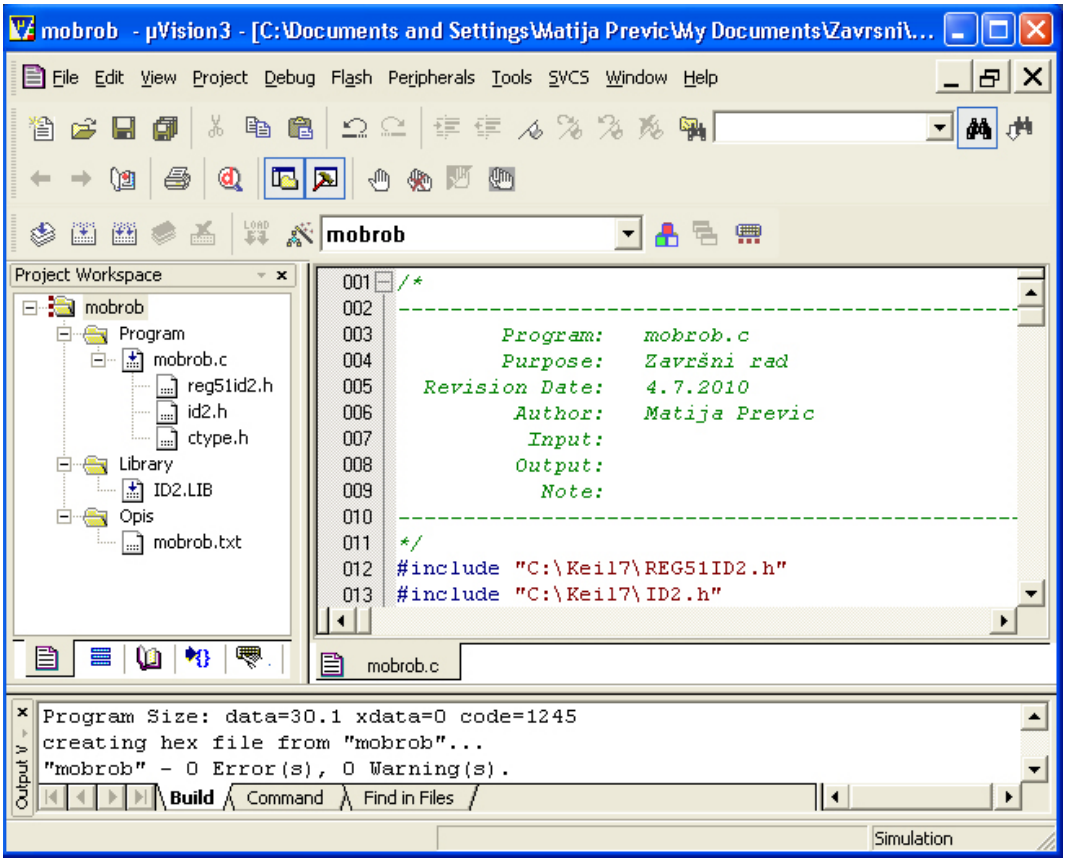


Slika 3.1: Sučelje programskog paketa Delphi 7

Osim standardnih Borlandovih biblioteka, korištene su još CPortLib 3.10 Dejana Crnile za komunikaciju servera sa serijskom RS232 vezom i DirectX Joystick komponenta od WinSofta za komunikaciju servera sa igračom palicom koristeći DirectX 8. Potonja nije besplatna i košta 20 $, no funkcionira dovoljno dobro i kao trial verzija, te nije bilo potrebe za ulaženje u trošak.

3.2 μVision

Za programiranje mikrokontrolera mobilnog robota korišten je programski paket μVision 3 V3.80 od Keil Software, Inc. Ova integrirana razvojna okolina je brza i jednostavna za upotrebu. Nudi rukovanje projektima, editiranje source koda, kompajliranje i debuggiranje. Koristi C51 kompajler V7.01 za C programski jezik. Sučelje je prikazano na slici 3.2.



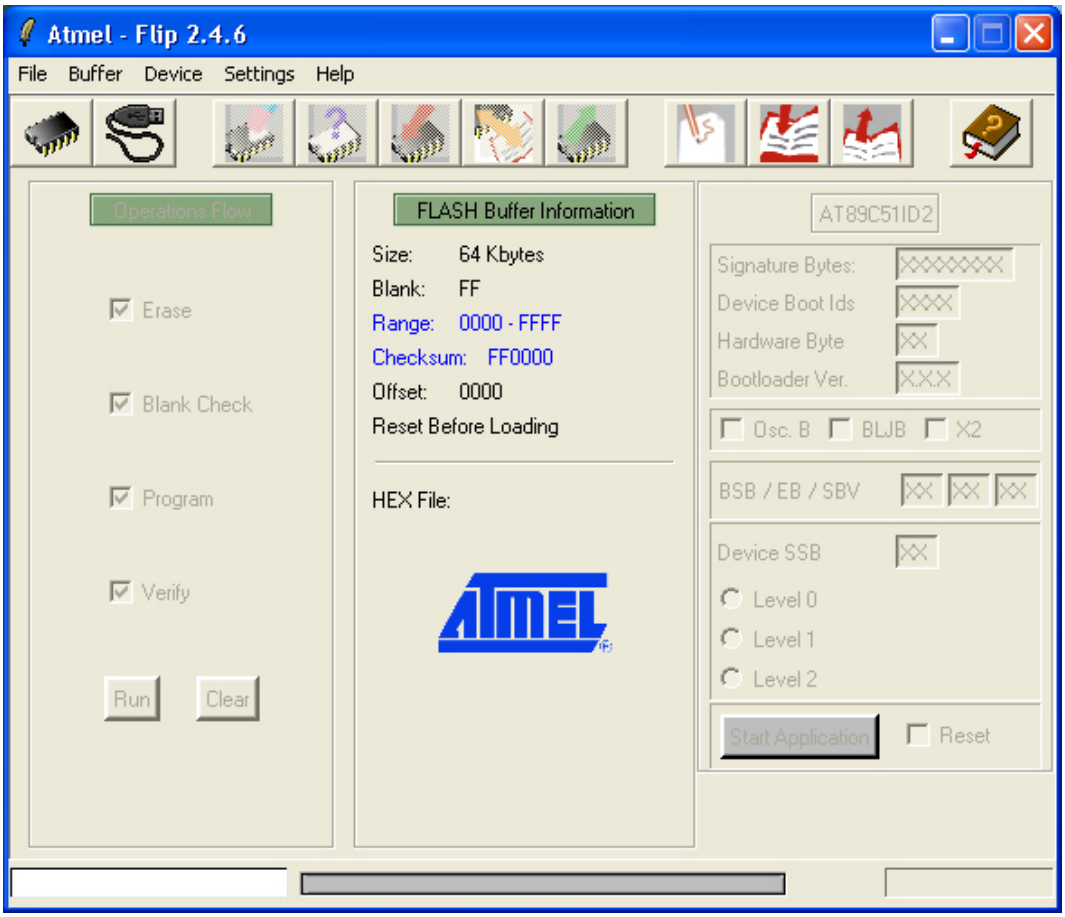
Slika 3.2: Sučelje programskog paketa μVision 3

Osim standardnih C biblioteka korištene su IDE2.lib biblioteka i IDE2.h header datoteka profesora Mladena Crnekovića, te REG51ID2.h header datoteka. U C jeziku header datoteke sadrže deklaracije klasa, podrutine, varijable i ostale identifikatore, a jedan od razloga je zato da bi se mogli koristiti iz više korisnikovih source datoteka. Datoteke IDE2.* su korištene za komuniciranje mikrokontrolera sa serijskom RS232 vezom, LCD-om, motorima i infra crvenim senzorima. Datoteka REG51ID2.h služi Keil kompajleru za definiranje registra s posebnim funkcijama[2]. Ti se registri koriste za upravljanje timerima, counterima, serijskim U/I-ima, U/I portovima i periferijom.

[2]SFR - Special Function Register

3.3 Flip

Za preuzimanje *.hex programa napravljenog u μVisionu na Atmelov AT89C51ID2 mikroprocesor mobilnog robota korišten je Atmelov programski paket Flip[3] 2.4.6, čije je sučelje prikazano na slici 3.3.



Slika 3.3: Sučelje programskog paketa Flip 2.4.6

[3]FLexible In-system Programmer

4

Server MobRob

Glavno programiranje robota je pripalo serverskoj aplikaciji MobRob na Windows operativnom sustavu. Njen je cilj omogućiti teleoperacijsko i autonomno navođenje robota preko bežične dvosmjerne serijske veze, uz korištenje naredbi više razine. Potonje znači da se korisnik ne mora zamarati programiranjem robota već upravlja ponašanjem robota preko ponuđenih serverskih naredbi.

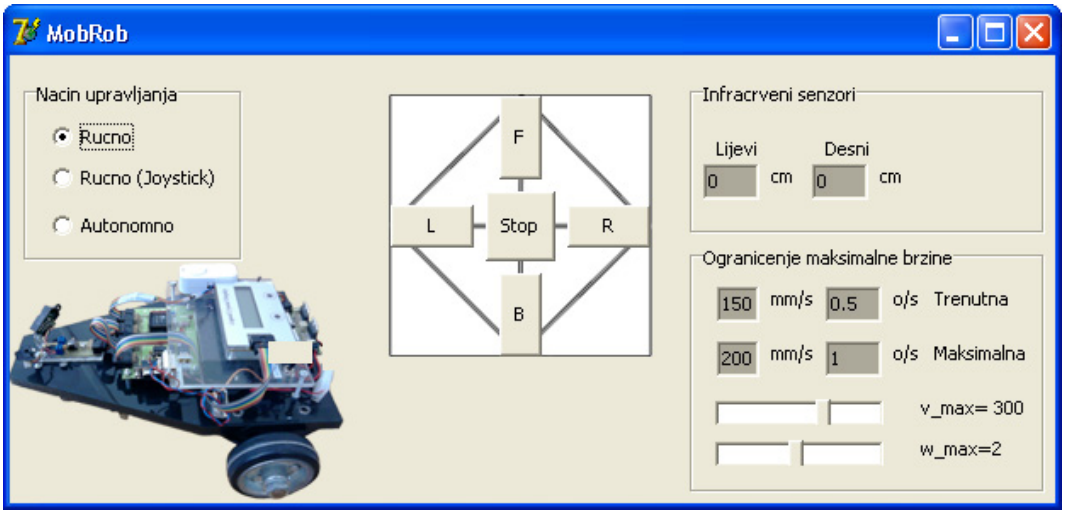
4.1 Općenito

Aplikacija MobRob podržava tri načina upravljanja:

1. Ručno
2. Ručno pomoću igraće palice
3. Autonomno

Treba napomenuti da se u niti jednom načinu upravljanja robot ne kreće "savršeno po liniji" što je posljedica u nemogućnosti reproduciranja dva potpuno identična motora. To bi

se moglo dovoljno dobro rješiti uporabom neke vrste regulacije pri upravljanju. U slučaju da se dogodi prekid serijske veze, mobilni je robot programiran da stane i čeka dok se veza ponovo ne uspostavi. U svim se načinima rada cijelo vrijeme dobivaju informacije o stanju infracrvenih daljinomjera, trenutnoj brzini mobilnog robota, te postavljenom ograničenju maksimalne brzine. Te su informativne kućice prepoznatljive po sivoj podlozi oko teksta, za razliku od bijele podloge koja označava varijable koje je moguće promijeniti.

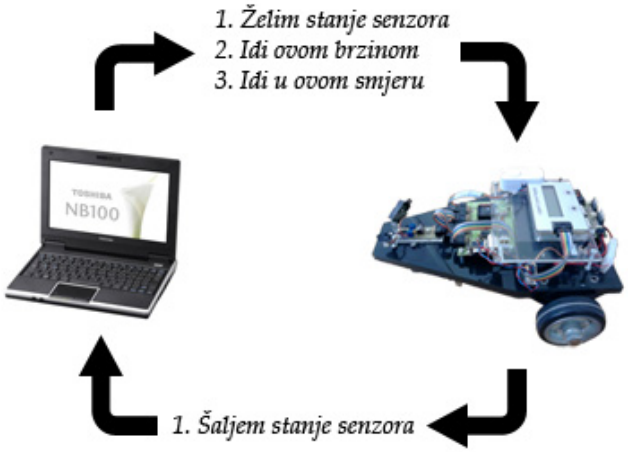


Slika 4.1: Aplikacija MobRob u ručnom načinu rada

4.1.1 Infracrveni daljinomjeri

Udaljenost između robota i prepreke se cijelo vrijeme računa u pozadini i prikazuje u aplikaciji MobRob. Zasebno se prikazuju približne udaljenosti prepreke od lijevog i desnog daljinomjera. Raspon prikaza udaljenosti nalazi se između 15 cm i 80 cm, dakle unutar pouzdanog dijela karakteristike senzora. Ako se robot nalazi dalje od 80 cm od prepreke umjesto brojke prikazat će se simbol ">>>". Ako se nalazi bliže od 15 cm, udaljenost će se povećavati umjesto smanjivati. Do te greške dolazi zbog karakteristike senzora, gdje napon počinje padati umjesto da nastavi rasti daljnjim približavanjem predmeta. Ta greška postaje problem tek u autonomnom načinu rada, gdje udaljenost koju senzori mjere više nije samo informativna. Zbog toga je, pri pokretanju aplikacije, zadana minimalna vrijednost na koju robot smije prići predmetu 25 cm, a ne 15 cm. Pokazalo se tijekom rada da je najbolje držati udaljenost od 30

cm. Razlika dobivenog rezultata sa senzora poradi različite boje i osvjetljenja predmeta kojem se robot približava može se zanemariti, a izraženija je što je predmet dalje.



Slika 4.2: Razmjena paketa između serverske strane i mobilnog robota

4.1.2 Slanje paketa mobilnom robotu

Kao što je već više puta spomenuto, aplikacija MobRob šalje preko bežične dvosmjerne serijske veze pakete mobilnom robotu. U tim paketima se nalazi uputa o ponašanju. Tri su tipa upute:

1. Traži vrijednost infracrvenog daljinomjera 10 puta u sekundi
2. Pošalji potrebni smjer kretanja mobilnog robota
3. Pošalji odgovarajući PWM za postizanje željene brzine

Prve dvije upute su u formatu znaka (character), a treća je kodirani paket oblika $\#PWM_L PWM_R *$, gdje # i $*$ označavaju početak i kraj paketa, a PWM_L i PWM_R heksadecimalni broj koji će se prevesti u decimalni na drugoj strani. Taj broj označava vrijednost impulsno širinske modulacije koja predstavlja odgovarajuću vrijednost traženih kutnih brzina kotača, a dobiva se

pomoću jednadžbi 2.5 i 2.6. Jedini paket koji se dobiva s druge strane veze je stanje infracrvenog daljinomjera, a pobliže je opisan u narednom dijelu rada. Aplikacija MobRob taj paket prevodi u približnu udaljenost u cm. Na slici 4.2 je prikazana sva razmjena podataka između MobRob aplikacije i mobilnog robota.

4.2 Ručno upravljanje

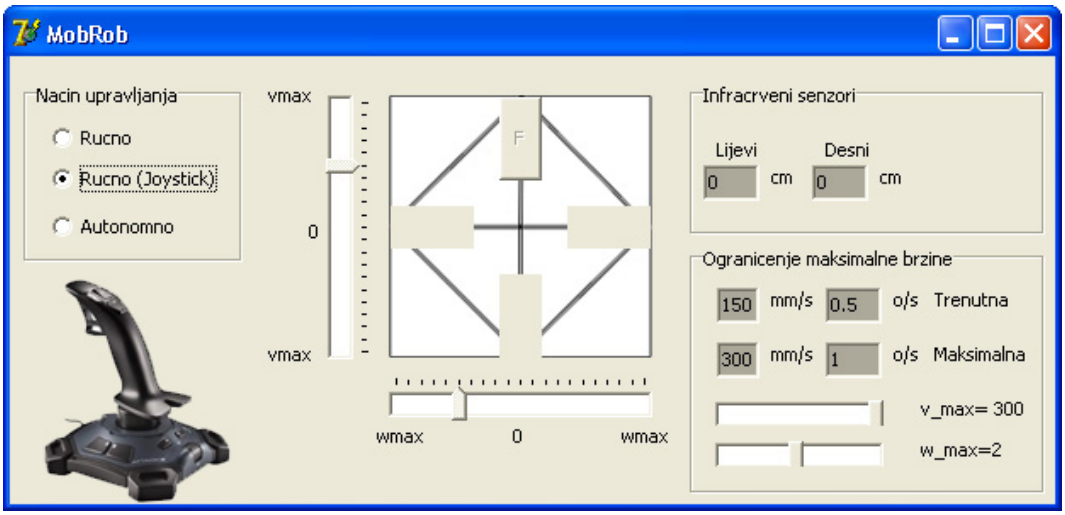
Prvi način je samo u slučaju nužde ako ne bi bilo mogućnosti za korištenje igraće palice (joysticka), jer je upravljanje nezgrapno i sporo. Sučelje je prikazano na slici 4.1. Robot ima 4 smjera kretanja i zaustavljanje (Stop). Može se kretati s oba kotača naprijed (F) ili nazad (B), te međusobno obrnutim smjerom kretanja kotača u dvije kombinacije - lijevo (L) i desno (R). Kod kretanja lijevo i desno brzine oba kotača su jednake te će to rezultirati okretanjem robota na mjestu. Drugim riječima linearna brzina robota je nula, dok kutna nije.

4.2.1 Ograničenje maksimalne brzine

Brzine oba kotača su jednake, a mijenjaju se tako da se ograniče maksimalna linearna i kutna brzina mobilnog robota pomoću pomičnih traka. Ponovnim pritiskom na željeni smjer kretanja, robot koristi novo izabranu brzinu. Maksimalna brzina kojom se robot može gibati po pravcu je 300 mm/s. Ta je brzina predstavljena varijablom v_{max}. Robot se može okretati na mjestu maksimalnom kutnom brzinom od 2 o/s, koja je predstavljena varijablom ω_{max}. Pri pokretanju aplikacije zadane su maksimalne brzine 300 mm/s za linearnu i 1 o/s za kutnu brzinu. Pomoću dvije pomične trake korisnik može podešavati maksimalne dozvoljene brzine robota v_{max} i ω_{max}. Za linearnu se brzinu raspon kreće između 11 mm/s i 300 mm/s, a za kutnu između 0.1 o/s i 2 o/s. Veličina koraka je 3 mm/s, odnosno 0.1 o/s. Općenito brzine treba uzeti samo informativno.

4.3 Ručno upravljanje igraćom palicom

Drugi način rada je osnova teleoperacijskog upravljanja pomoću MobRob aplikacije i prikazan je na slici 4.3. Dovoljno je da je bilo koja igrača palica ili "gamepad" spojena na neki od USB utora na računalu na kojem se nalazi MobRob aplikacija kako bi korisnik mogao preko nje upravljati mobilnim robotom. To je omogućeno spomenutom komponentom za Delphi u 3. poglavlju.



Slika 4.3: Aplikacija MobRob u ručnom načinu rada pomoću igraće palice

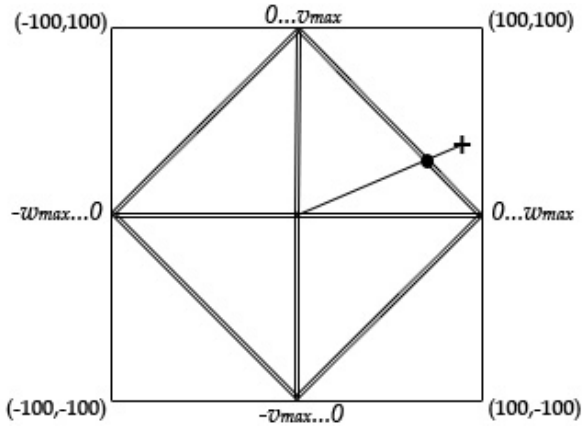
U sučelju se promijenio samo središnji dio, gdje se umjesto tipaka za 4 smjera kretanja i zaustavljanje nalaze informativne trake o veličini i smjeru linearne i kutne brzine. Tipke se isto tako pojavljuju, no ovog puta kao pasivni elementi informativnog karaktera o približnom smjeru kretanja robota. Korisnik pomicanjem palice prema naprijed i nazad zadaje linearnu brzinu, a pomicanjem prema lijevo i desno kutnu brzinu mobilnog robota. No sada robot ima još jedan način kretanja a to je kada je palica usmjerena ukoso, odnosno kada niti linearna niti kutna brzina nisu blizu nuli. Tada robot skreće lijevo ili desno, bilo da se kreće naprijed ili nazad. Za ručno upravljanje igraćom palicom se preporuča ostaviti ograničenje maksimalne linearne brzine na zadanoj vrijednosti od 300 mm/s. Korisnik pomicanjem palice mijenja trenutnu brzinu koja će biti jednaka maksimalnoj kada je koordinata palice jednaka 100% u jednom od smjerova. Na slici 4.3 je prikazan položaj palice gdje se robot kreće prema napri-

jed na pola maksimalne linearne brzine i skreće ulijevo na pola maksimalne kutne brzine od postavljenog. Sigurnosna kočnica se nalazi na prvom tipkalu, što je na igraćoj palici uobičajeno tipka za kažiprst.

Brojač zadužen za izvršavanje ovog načina upravljanja aktivira se svakih 200 ms. Pri učitavanju koordinata sa igraće palice potrebno ih je prilagoditi skali s X i Y koordinatama od -100 do 100, radi lakšeg računanja:

```
1 // Ucitavanje pozicije igrace palice
2 Xkoord:=Round((DxJoystick1.PositionX-32767)/327.67); // cini skalu od -100 do 100
3 Ykoord:=Round((32767-DxJoystick1.PositionY)/327.67); // -||-
4
5 Ykabs := Abs(Ykoord); // apsolutna vrijednost koordinate
6 Xkabs := Abs(Xkoord); // -||-
```

Koordinata X reprezentira linearnu brzinu v, a koordinata Y kutnu brzinu ω. No tu dolazi do problema. Kada bi zbroj X i Y koordinata bio iznad 100 (100%), to bi prema jednadžbama (2.3) i (2.4) dalo veće kutne brzine kotača robota od mogućeg. Zbog toga je potrebno ograničiti područje koordinata igraće palice kako bi u svakoj točki zbroj dviju kooordinata bio najviše 100. Na slici 4.4 je prikazano to ograničenje područja koordinata s primjerom kada se palica



Slika 4.4: Ograničenje područja koordinata igraće palice

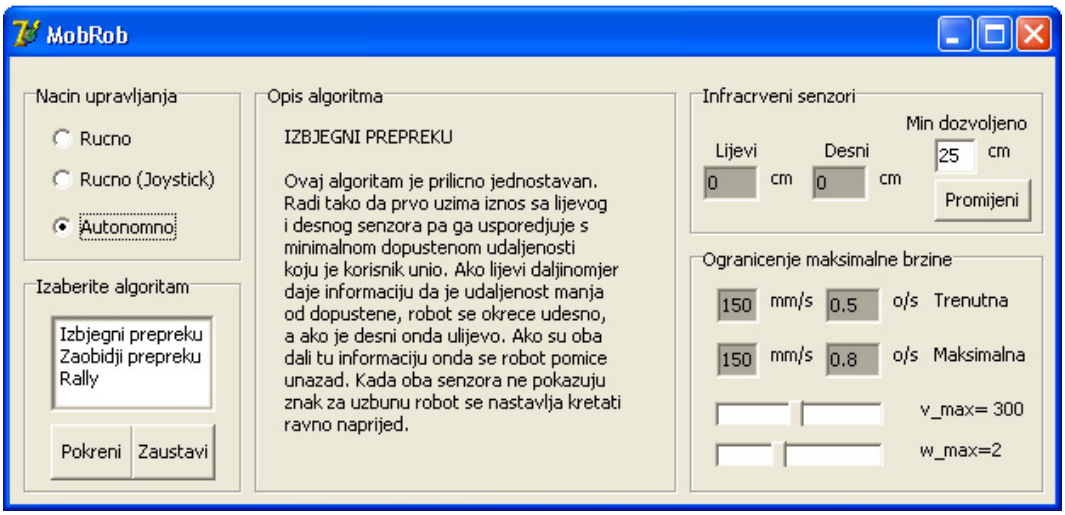
nalazi izvan područja (crni križ na slici). Ta se pozicija tretira kao da je palica pozicionirana na sjecištu granične dužine i pravca koji spaja poziciju palice s ishodištem (crna točka na slici). U programu to izgleda:

```
1  Xp:=Round((100*Xkoord)/(Ykoord+Xkoord));
2  Yp:=Round((100*Ykoord)/(Ykoord+Xkoord));
3  if (Pitagora(Xkabs,Ykabs)>Pitagora(Xp,Yp)) then
4      begin
5      Xkabs := Abs(Xp);
6      Ykabs := Abs(Yp);
7      end;
8  v := Round(vmax*Ykabs/100);
9  w := wmax*Xkabs/100;
10 // kutne brzine kotaca
11 wL:=(2*v+w*bias)/d;
12 wR:=(2*v-w*bias)/d;
13 smjerkretanja:='F';
```

gdje funkcija $Pitagora(a, b)$ daje hipotenuzu pravokutnog trokuta s katetama a i b.

4.4 Autonomno upravljanje

Treći način rada je autonomno navođenje mobilnog robota gdje korisnik bira neke od ponuđenih algoritama ponašanja mobilnog robota. Sučelje je prikazano na slici 4.5.

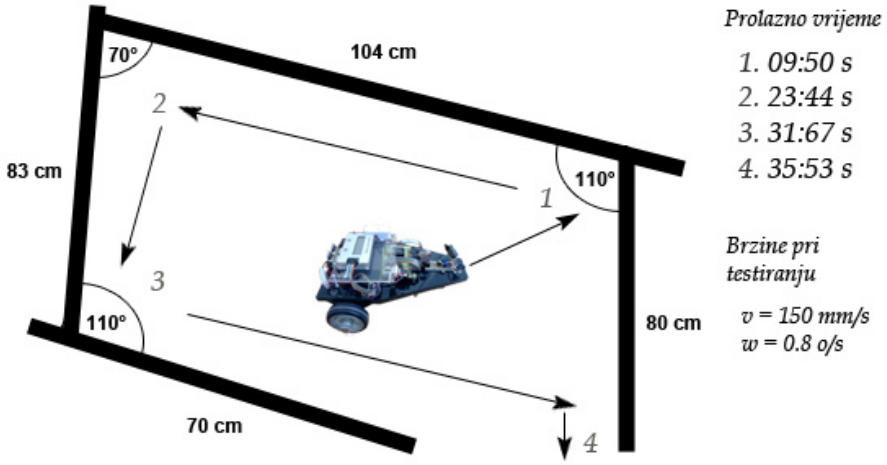


Slika 4.5: Aplikacija MobRob u autonomnom načinu rada

Novi dio sučelja je izbornik algoritma. Korisnik može izabrati željeni algoritam. Opis algoritma će se prikazati u središnjem dijelu sučelja. Po želji se algoritam može pokrenuti i zaustaviti. Ako korisnik promijeni način upravljanja, algoritam će se odmah prestati izvršavati i robot će stati. Brzina se podešava ograničavanjem maksimalne brzine, kao i u ručnom načinu upravljanja. Postoji mogućnost podešavanja minimalne dozvoljene udaljenosti na koju robot smije priči predmetu. Brojači zaduženi za izvršavanje algoritama aktiviraju ih svakih 100 ms.

4.4.1 Izbjegni prepreku

Prvi algoritam rješava problem izlaska robota iz labirinta. Na slici 4.6 je prikazano testiranje algoritma. Robot je postavljen na sredinu labirinta i početno usmjeren tako da bi se osiguralo da će "zaglaviti" u svakom kutu labirinta. Trebalo mu je 10 sekundi da savlada prvi tupi kut, 13 sekundi za šiljasti kut, 8 sekundi za drugi tupi kut i još 4 sekunde da izađe van. Sveukupno 35 sekundi. Robot se gibao linearnom brzinom od 150 mm/s i kutnom brzinom od 0.8 o/s.



Slika 4.6: Prikaz testiranja algoritma "Izbjegni prepreku"

Ovaj algoritam je prilično jednostavan. Radi tako što prvo uzima iznos sa lijevog i desnog senzora pa ga uspoređuje s minimalnom dopuštenom udaljenosti koju je korisnik unio. Ako lijevi daljinomjer daje informaciju da je udaljenost manja od dopuštene, robot se okreće udesno, a ako je desni dao tu informaciju onda se robot okreće ulijevo. Ako su oba daljinomjera dala

tu informaciju onda se robot pomiče unazad. Kada oba senzora ne pokazuju znak za uzbunu robot se nastavlja kretati ravno naprijed.

Početno je ograničenje maksimalne brzine, što je ujedno i trenutna brzina, zadano na vrijednost od 150 mm/s linearne i 0.8 o/s kutne brzine. Veća linearna brzina od 150 mm/s nije preporučljiva, dok je kutnu brzinu najbolje držati između 0.8 o/s i 1 o/s. U slučaju šiljastih kuteva ovaj će se algoritam po dosada opisanom ponašanju zaglaviti. Kretati će se lijevo desno na mjestu pri kutnoj brzini većoj od 0.5 m/s ili naprijed nazad ako se kreće manjom brzinom od 0.5 m/s. Za brzine iznad 1.5 o/s robot ne stigne dovoljno brzo reagirati kako bi izbjegao sudar s preprekom pod šiljastim kutom[1]. Slučaj zaglavljenja rješava funkcija *Jel_Zapeo*() koja, ako nije bio dovoljan broj naredbi za kretanje naprijed u određenom vremenu, malo udaljava robota i zakreće ga u smjeru u kojem trenutno daljinomjer očitava veću udaljenost od drugog daljinomjera. Za kutne brzine manje od 0.5 m/s funkcija *Jel_Zapeo*() se aktivira i ako je bio prevelik broj naredaba za natrag unutar određenog vremena. Svi parametri korišteni u funkciji *Jel_Zapeo*() su određeni empirijski. Općenito se robot bolje izbavlja iz manjih kuteva pri manjoj brzini a iz većih pri većoj brzini.

4.4.2 Rally

Drugi se algoritam, za razliku od prvoga, bazira na povjerenju da će robot uvijek imati prostora za skrenuti, odnosno da nema slijepe ulice. Kao što i ime govori napravljen je za uređenu stazu omeđenu zidovima. Robot vozi puno brže nego u prvom algoritmu. Također može skretati kao i u upravljanju igraćom palicom, a ne samo ići naprijed-nazad i okretati se lijevo-desno kao u ručnom upravljanju i prvom algoritmu.

Pri radu i ovaj algoritam prvo provjerava stanje infracrvenih daljinomjera. Ako je udaljenost do zidova veća od 80 cm, tada kreće naprijed maksimalnom brzinom. Kada lijevi ili desni daljinomjer osjeti prepreku na udaljenosti manjoj od 80 cm, postepeno smanjuje linearnu brzinu kako udaljenost pada i lagano skreće kutnom brzinom u suprotnu stranu. Kutna

[1]To se neće dogoditi ako je minimalna udaljenost na koju robot smije prići prepreci podešena na veći broj.

brzina se povećava kako je zid sve bliži. Ako se zapreka nađe unutar 30 cm od senzora, tada će robot prestati ići naprijed, brzo će se okretati u suprotnu stranu dok oba senzora ne budu pokazivala više od 30 cm, te nastaviti ravno s blagim skretanjem dok se ne odvoji od zida. Kućica za minimalnu dozvoljenu udaljenost kojoj robot smije prići u ovom algoritmu služi kako bi se početak okretanja robota po želji postavio na veću udaljenost.

5

Program na strani robota

Prethodno je opisan rad serverskog programa na osobnom računalu, i njegova komunikacija s robotom. Osim toga potreban je program na robotu koji će stvarno i izvršiti očekivanja sa servera.

5.1 Program

Nakon iniciranja serijske veze, infracrvenog porta i LCD ekrana, te zaustavljanja motora robota, program ulazi u glavnu petlju. Prvo uzima znak sa serijske veze:

```
1 tipka=GetKey();
2 tipka=toupper(tipka); // pretvori mala slova u velika
```

Nadalje provjerava je li taj znak onaj koji određuje u kojem smjeru će se motori robota pokretati, a ako jest onda mijenja smjer motora. Primjera radi prikazan je dio koda kada motor treba stati:

```
1 else if (tipka=='S')
2 {
3     RelePort=MotStop; // Zaustavi motore robota
```

```
4       ClearLine(1); // Obrisi prvi red na LCD-u
5       MoveCursor(1,1); // Pocni pisat od prvog reda, prva kolona
6       WriteString("Smjer: stani");
7   }
```

Nakon toga provjerava je li možda uhvaćen znak '#':

```
1   if (tipka=='#')
2       {
3       RxBuffer[0]=tipka;
4       for(i = 1; i < 6;i++)
5           {
6           RxBuffer[i]=GetKey();
7           }
8       if (RxBuffer[5]=='*')
9           {
10          PWML=GetByte(1);
11          PWMR=GetByte(3);
12          RunPWM(1);
13          RunPWM(2);
14          SetPWM(1,PWML);     // 255-0 (gdje je 0 trazeni iznos)
15          SetPWM(2,PWMR);
16          }
17      }
```

Ako jest onda program ulazi u drugu petlju koja će primiti još 5 znakova prije nego se prekine. Prva 4 znaka su heksadekadski kod PWM vrijednosti koja se treba dovesti na motore robota i spremljeni su u za to određen niz. Ako je peti znak '$*$' to znači da je niz ispravan i predaje se funkciji *GetByte*() koja će pretvoriti heksadekadske brojke u dekadske. Te se vrijednosti postavljaju funkcijom $SetPWM()$ na drivere motora. Nakon toga se provjerava je li uhvaćen znak za dobivanje informacija o stanju infracrvenih daljinomjera, a ako jest pokreće se funkcija *IRsenzor*(). Na kraju se, prije vraćanja na početak petlje, postavlja uhvaćeni znak na 'S', kako bi se motori robota zaustavili ako nikakva naredba ne dođe serijskom vezom.

5.2 Funkcije

U programu su korištene standardne funkcije C biblioteke i Atmelovog mikroprocesora. Osim toga korištene su i neke funkcije mentora Mladena Crnekovića za komuniciranje s robotovim komponentama poput LCD ekrana, drivera motora, infracrvenih daljinomjera i serijske veze. Funkcije *IRsenzor*() i *GetByte*() su napisane unutar programa robota.

5.2.1 ID2.h

U header datoteci ID2.h mentora Mladena Crnekovića nalaze se slijedeće korištene funkcije:

```
1  /*
2  ---------------------------------------------------------------
3                Program:   ID2.h
4                Purpose:   Prototypes for C51ID2.lib
5          Revision Date:   01-06-2009
6                 Author:   Mladen Crnekovic
7                  Input:   Procedures from UTIL, KBD & LCD library
8                 Output:   ID2.lib
9                   Note:
10 ---------------------------------------------------------------
11 */
12 // Funkcije za komunikaciju s LCD ekranom
13 extern void  InitLCD();
14 extern void  ClearLCD();
15 extern void  ClearLine(unsigned char L);
16 extern void  HideCursor();
17 extern void  MoveCursor(unsigned char RED, unsigned char KOL);
18 extern void  WriteChar(unsigned char X);
19 extern void  WriteString(unsigned char *STR);
20 extern void  WriteInt(unsigned int I, unsigned char D);
21 // Funkcije za komunikaciju sa serijskom vezom
22 extern void  InitRS232();
23 extern void  PrintChar(unsigned char c);
24 extern void  PrintString(unsigned char *STR);
25 extern void  PrintInt(unsigned int I, unsigned char D);
26 extern unsigned char GetKey();
27 // Funkcije za komunikaciju s motorima
28 extern void  SetPWM(unsigned char N, unsigned char x);
29 extern void  RunPWM(unsigned char N);
```

```
30 extern void  StopPWM(unsigned char N);
31 // Funkcija za komunikaciju s infracrvenim daljinomjerima
32 extern unsigned int LTC1298(unsigned char Channel);
```

5.2.2 Infracrveni daljinomjeri

Kada je zaprimljen zahtjev za stanjem infracrvenih daljinomjera, izvršava se funkcija IRsenzor(). Ona koristi funkciju $LTC1298()$ s argumentom 0 ili 1. Ako je dani argument 0, funkcija vraća stanje desnog senzora, a ako je 1 tada vraća stanje lijevog senzora. Dobivenim podacima se uklanjaju 4 znamenke najmanjeg značaja (least significant digit) čime se olakšava teret na serijskoj vezi. Nakon toga se ispisuje na serijsku vezu paket oblika $Lx_L * Rx_D*$, gdje su x_L i x_D vrijednosti lijevog i desnog senzora, L označuje nadolazeći početak vrijednosti lijevog senzora, a R nadolazeći početak vrijednosti desnog senzora, dok * označava kraj. Stanje senzora se ispisuje na LCD u formatu koji daje funkcija $LTC1298()$, a ne u cm.

5.2.3 Pretvorba heksadekadskog koda u dekadsku vrijednost PWM-a motora

Da bi se odkodirala naredba sa servera, korištena je funkcija *GetByte*(). U paketu je poslana naredba prvo za lijevi pa za desni motor. Dakle, potrebno je u argument GetByte funkcije dati broj 1 za odkodiranu vrijednost lijevog motora i 3 za odkodiranu vrijednost desnog motora.

```
1 unsigned char GetByte(unsigned char index)
2 {
3 unsigned char b, HiChar, LoChar;
4 code unsigned char DD[23] = {0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,0,0,10,11,12,13,14,15};
5       HiChar=RxBuffer[index]; LoChar=RxBuffer[index+1];
6       b=16*DD[HiChar-48] + DD[LoChar-48];
7       return(b);
8 }
```

6

Zaključak

Ovim je radom postignut osnovni cilj da se način upravljanja mobilnim robotom približi korisniku. Naravno uvijek postoji mjesta za poboljšanjem. Tako bi se, na primjer, moglo dodati više algoritama. Dodavanjem opcije izbora porta na koji će se spojiti serijska veza bi omogućilo korištenje aplikacije MobRob na bilo kojem računalu. Povećanjem skale koordinata s igraće palice dobila bi se bolja finoća teleoperacijskog upravljanja. Podvožje robota je na prednjoj strani uže a na stražnjoj šire, što uz daljinomjere sprijeda dovodi do povremenog zapinjanja robota za predmete pri skretanju. Zato bi za bolji rad bilo dobro ugraditi još infracrvenih daljinomjera kako bi se pokrila cijela robotova okolina. Također bi se prednji daljinomjeri mogli prebaciti na sredinu robota, čime bi se rješio problem kada je prepreka bliže od 15 cm. Smatram da osim navedenog nije potrebno omogućiti korisniku više opcija i informacija s postojećim mogućnostima robota. No, kada bi se iskoristila kamera to bi otvorilo nove mogućnosti. Unutar programerskog koda bi se također dala napraviti poboljšanja koja sam uočio tek kada sam isprogramirao cjelinu. Na primjer, kod na serverskoj strani bi se dosta pojednostavio, ako bi se serijskom vezom slala brzina, a PWM preračunavao na robotovoj strani. Sve u svemu zadovoljan sam naučenim i postignutim tijekom ovog završnog rada, kao i tijekom dosadašnjeg studija.

Literatura

[1] Siegwart R., Nourbakhsh I.R., *Introduction to autonomous mobile robots*, MIT, 2004.

[2] Kos J., *Završni rad - Mobilni robot s infracrvenim daljinomjerima*, FSB, 2009.

[3] Borland Software Corporation, *Delphi 6 Developer's Guide*. SAD, 2001.

[4] Keil Software, Inc, *µVision IDE Overview*. http://www.keil.com/uvision/, 4.7.2010.

[5] Keil Software, Inc, *Special Function Registers*. http://www.keil.com/support/man/docs/c51/c51_le_sfrs.htm, 4.7.2010.