



UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU

Seminarski rad:
Autonomno vozilo koje prati drugo vozilo

PREDMET:

- PRAKTIKUM ELEKTRONIKE -

Aladžuz Azra

Sarajevo, septembar 2023.g.



Sadržaj

1	Uvod	1
2	Postavka zadatka	2
1.	Blok struktura	2
2.	Proteus shema	4
3	Komponente	5
1.	PIC mikrokontroler	5
2.	Sharp 2Y0A02 senzor	7
3.	L298N Driver	10
4.	DC motor	12
4	Realizacija projektnog zadatka	14
5	Zaključak	16
6	Dodatak	17
1.	PCB dizajn	17
2.	Kod	19

Poglavlje 1



Uvod

Pojavom tehnologije autonomnih vozila, automobiliška industrija je doživjela je značajan pomak u posljednjih nekoliko godina. Jedan od značajnih aspekata ove tehnologije jeste razvoj autonomnih vozila koja su sposobna pratiti druga vozila na veoma učinkovit način. Autonomno praćenje vozila uz održavanje sigurne udaljenosti je koncept koji nudi poboljšan protok prometa, optimiziranu potrošnju goriva, povećanu sigurnost i učinkovitiju iskoristivost cesta. Dok prometne gužve i dalje pogađaju gradske centre i autoceste, potreba za inovativnim rješenjima postala je iznimno važna. Korištenjem najsavremenijih senzorskih tehnologija, algoritama umjetne inteligencije i pouzdanih komunikacijskih sistema nastoji se pronaći rješenje koje će sadržavati nevjerovatnu preciznost i koordinaciju, a znatno smanjiti prometne zastoje, povećati energetsku učinkovitost i smanjiti broj nesreća uzrokovanih ljudskom pogreškom.

Ovaj rad ima za cilj pružiti sveobuhvatan uvid u koncept autonomnog praćenja vozila, te nastoji doprinijeti evoluirajućem dijalogu o autonomnoj mobilnosti. Rad istražuje praktičnu realizaciju autonomnog vozila sposobnog slijediti drugo vozilo koristeći programski jezik C. Iskorištavanjem efikasnosti i fleksibilnosti jezika pokazano je kako se C može koristiti za razvoj složenih algoritama, integraciju senzora i sistema upravljanja potrebnih da bi autonomno vozilo tačno pratilo i slijedilo vodeće vozilo.

No, razvoj sistema autonomnog vozila koje slijedi drugo vozilo nije bez svojih složenosti. Neki od složenih zahtjeva koje je potrebno uzeti u obzir jesu obrada stvarnih podataka senzora u stvarnom vremenu, algoritmi donošenja odluka, komunikacija vozila s drugim vozilima i mehanizmi sigurnosti. Osiguranje sposobnosti vozila da održava sigurnu i optimalnu udaljenost od vodećeg vozila, reagira na nagle promjene i prilagodi se različitim scenarijima vožnje zahtijeva temeljito istraživanje softverskog dizajna i inženjeringu.

Fokus ovog rada proteže se izvan teoretskih okvira, obuhvaćajući praktične aspekte koji se javljaju pri pretvaranju konceptualnih algoritama u funkcionalni kod. Rad pruža uvid u proces razvoja autonomnog vozila, te doprinosi rastućem korpusu znanja koji oblikuje implementaciju tehnologija autonomnih vozila i njihov utjecaj na budućnost prijevoza.

Poglavlje 2

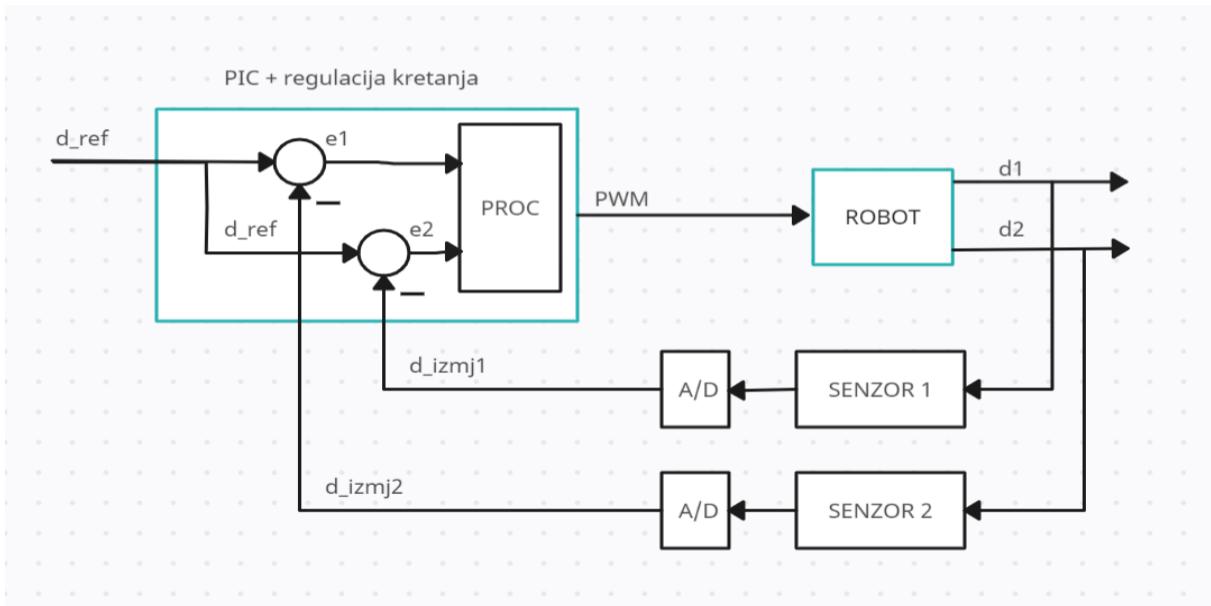


Postavka zadatka

U okviru projektnog zadatka potrebno je dizajnirati i implementirati autonomno vozilo koje je sposobno učinkovito pratiti i slijediti drugo vozilo. To zahtijeva implementaciju naprednih senzorskih sistema, sistema percepcije i kontrole kako bi se održavala sigurna udaljenost uz prilagođavanje promjenama brzine i smjera vodećeg vozila. Radi jednostavnije realizacije zadatka formirane su sheme koje daju uvid u način njegovog rješavanja.

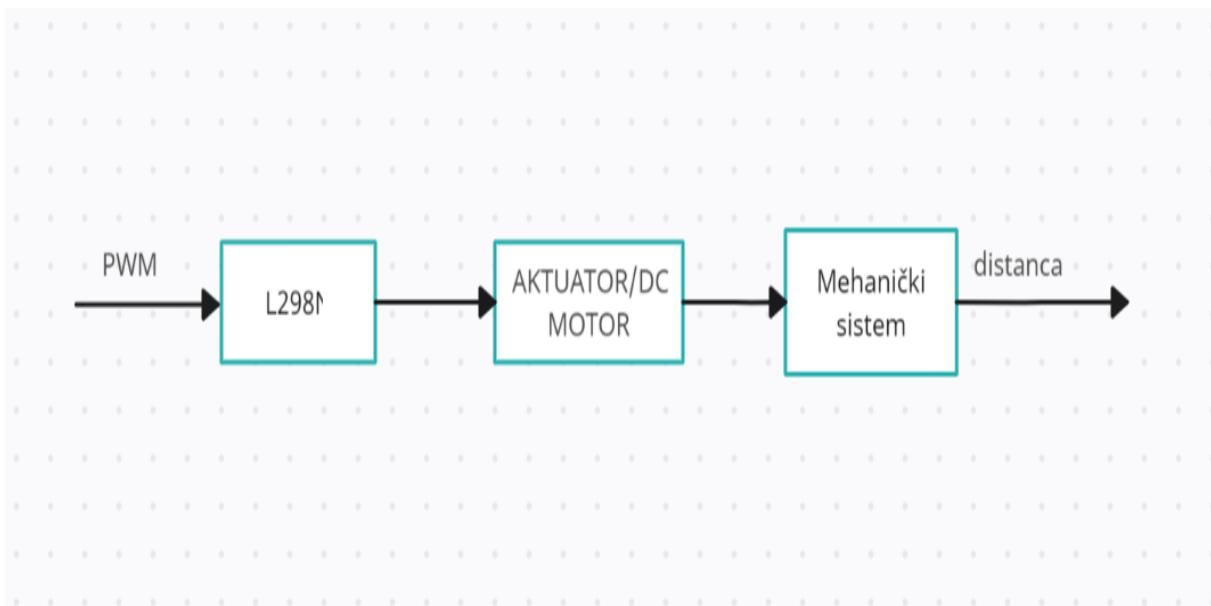
1. Blok struktura

Na slici 2.1 prikazana je blok struktura korištena prilikom realizacije autonomnog vozila. Sistem se sastoji od dva IC senzora udaljenosti. Ovi senzori rade tako da emituju signale prema okolini i zatim primaju reflektirane signale. Na osnovu vremena koje je potrebno da se signal vrati, senzori mogu izračunati udaljenost od prepreke. Svaki senzor daje izlazni napon koji varira u skladu s udaljenošću vozila ispred. Ovi izlazni naponi senzora ulaze u analogno-digitalni konverter (AD konverter) kako bi se digitalizirali i omogućila dalja obrada podataka. Ova digitalizacija omogućava precizno mjerjenje udaljenosti. Nakon što su naponi senzora digitalizirani, ulaze u regulator, koji u ovom slučaju predstavlja PIC mikrokontroler. PIC mikrokontroler je programiran tako da obrađuje ove podatke, napone pretvara u udaljenosti, i upoređuje ih s referentnim vrijednostima kako bi se obavila odgovarajuća regulacija. Na temelju ovog poređenja, mikrokontroler odlučuje kako prilagoditi brzinu i smjer motora. Mikrokontroler generira PWM signale kao izlaz. Ovi signali se šalju na poseban podsistem koji je na shemi predstavljen kao robot. PWM signali su digitalni signali koji omogućavaju preciznu kontrolu brzine i smjera motora.



Slika 2.1: Blok struktura

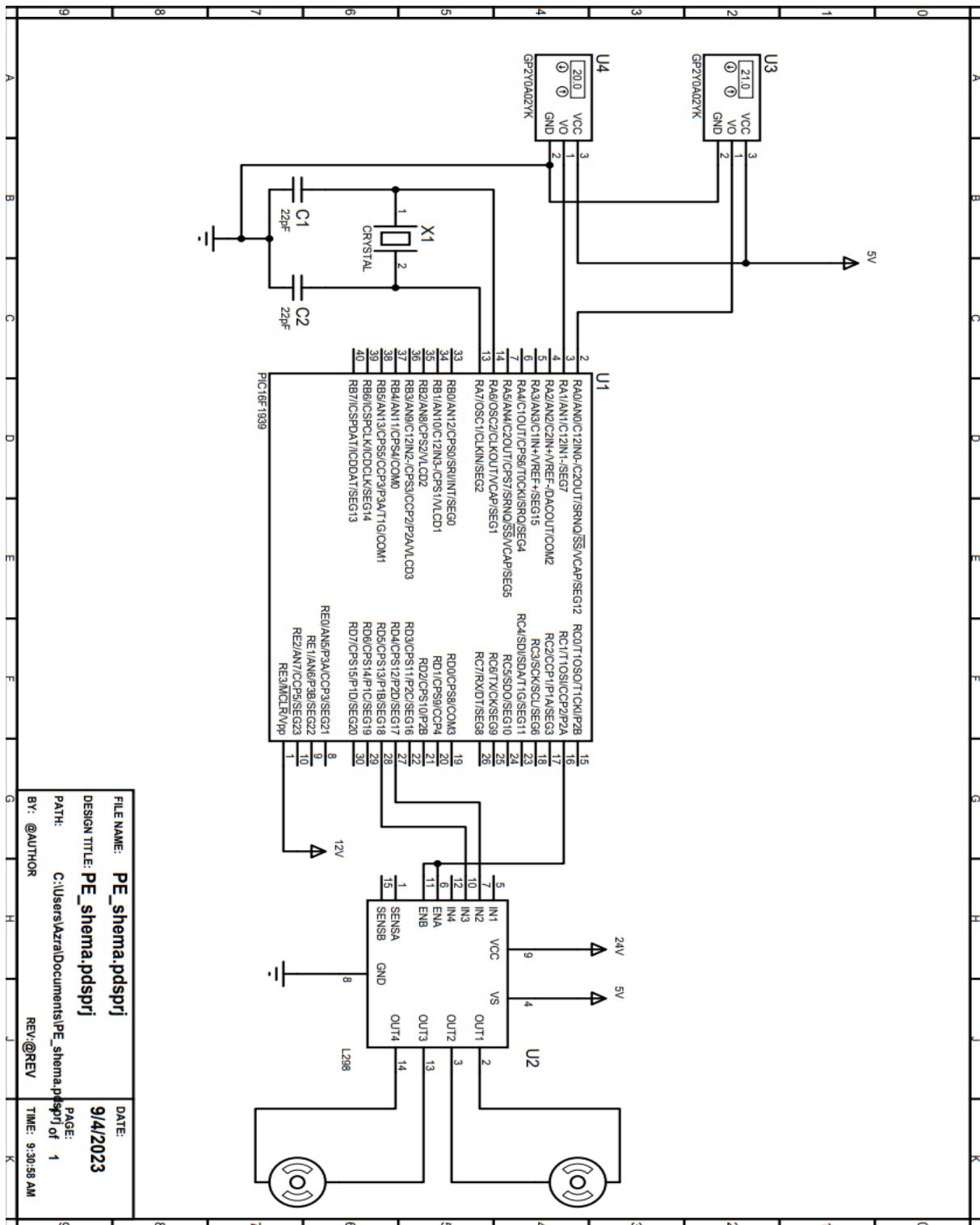
Ovisno o razlici između stvarnih udaljenosti od vozila ispred i referentnih vrijednosti, mikrokontroler će prilagoditi PWM signale za oba motora kako bi se vozilo kretalo prema željenom smjeru i brzini. Blok struktura podsistema robot data je na slici 2.2. Ovaj podsistem sastoji se od drivera, aktuatora odnosno DC motora i mehaničkog sistema. Driveri motora su elektronički uređaji koji primaju PWM signale od mikrokontrolera i koriste ih za kontrolu brzine i smjera motora. Ovi driveri omogućuju motorima da se kreću naprijed, nazad i da mijenjaju brzinu prema potrebi. Driveri zatim pokreću dva DC motora koji su fizički povezani s mehaničkim sistemom i zajedno sa njim čine vozilo. Ovisno o PWM signalima koje dobijaju od drivera, motori će se kretati u određenom smjeru i sa određenom brzinom, omogućavajući vozilu da se kreće na odgovarajući način.



Slika 2.2: Blok struktura podsistema Robot

2. Proteus shema

Proteus shema spoja data je na slici 2.3, a u narednim poglavljima je dat detaljan opis korištenih komponenti, kao i objašnjenje koda za regulaciju brzine i smjera motora.



Slika 2.3: Proteus shema

Poglavlje 3



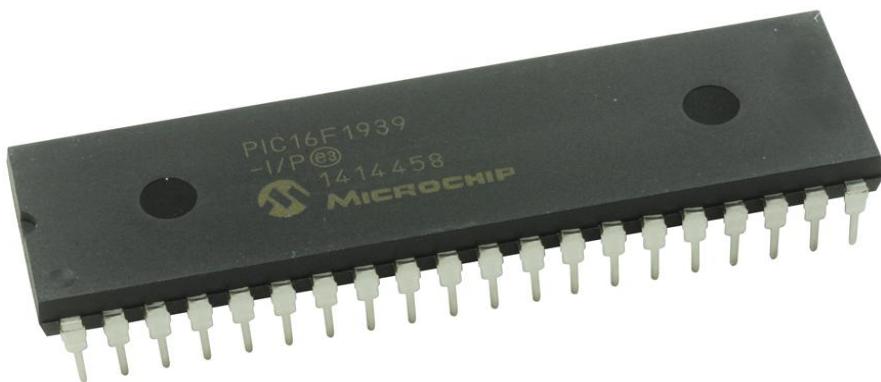
Komponente

1. PIC mikrokontroler

Mikrokontroler predstavlja osnovnu jedinicu sistema, te obuhvata algoritam za upravljanje ostvaren u vidu pisanog programa. Poseban tip mikrokontrolara jeste PIC. PIC (Peripheral Interface Controller) je kontroler perifernog sučelja i predstavlja porodicu mikrokontrolera razvijenu od strane tvrtke Microchip Technology.

Što se tiče arhitekture PIC mikrokontrolera, imaju odvojene memorije za programsku i podatkovnu memoriju čime je omogućen brz pristup instrukcijama i podacima. Također, obično imaju reprogramabilnu flash memoriju za pohranu programskih kodova za jednostavno ažuriranje i modifikaciju. Mnogi PIC mikrokontroleri dizajnirani su s naglaskom na nisku potrošnju energije, što ih čini prikladnim za aplikacije s baterijskim napajanjem. Dolaze s širokim rasponom ugrađenih periferija kao što su tajmeri, brojači, analogno-digitalni pretvarači (ADC), digitalno-analogni pretvarači (DAC), UART, SPI i I2C, čime je poboljšana njihova sposobnost za komunikaciju s vanjskim svijetom.

Porodica PIC mikrokontrolera nudi različite serije i modele s različitim karakteristikama i sposobnostima, prilagođavajući se različitim zahtjevima aplikacija. PIC kontroler korišten za projektni zadatak jeste PIC16F1939, slika 3.1, mikrokontroler iz PIC16 porodice sa 8-bitnim RISC CPU jezgrom visokih performansi.

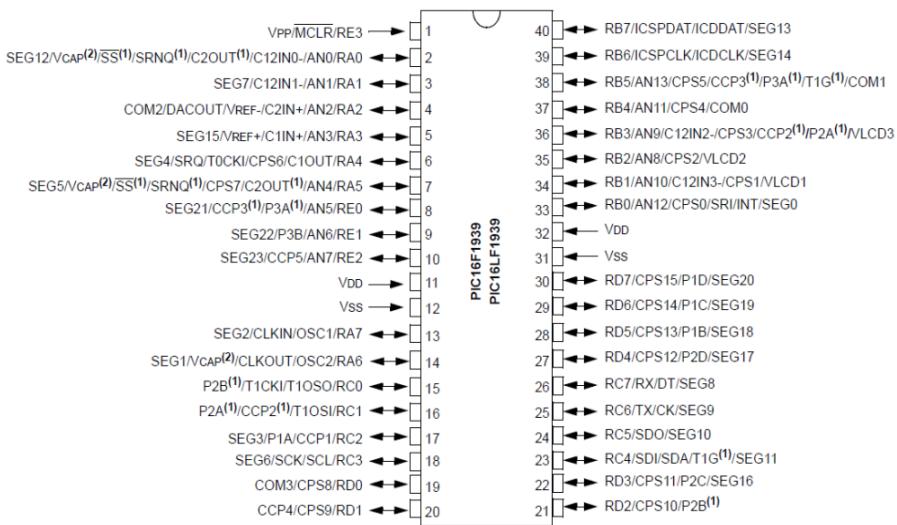


Slika 3.1: PIC16F1939 mikrokontroler

PIC16F1939 kontroler ima 49 instrukcija od kojih su sve jednociklusne osim instrukcija grananja koje su dvociklusne, a frekvencija sata iznosi do 32 MHz. Prelazak na bilo koji drugi

kontroler iz Microchipove porodice je vrlo jednostavan s obzirom na kompatibilnost ovog i ostalih kontrolera u smislu instrukcijskog seta i arhitekture. PIC16F1939 dolazi opremljen raznim ugrađenim perifernim uređajima što ga čini pogodnim za aplikacije na jednom čipu, a raspored pinova mu je dat na slici 3.2. Neke od ovih periferija uključuju:

- analogno-digitalni konverter (ADC),
- digitalno-analogni konverter (DAC),
- module za impulsno-širinsku modulaciju (PWM),
- tajmere i brojače,
- serijska komunikacijska sučelja (UART, SPI, I2C),
- CCP module,
- ECCP module.



Slika 3.2: Raspored pinova PIC16F1939 mikrokontrolera

Za programiranje PIC mikrokontrolera često su u upotrebi programski jezik C ili asembler. Za pisanje programskog koda koji implementira autonomno vozilo koje slijedi drugo vozilo, radi jednostavnosti, korišten je C programski jezik. Ono što Microchip nudi jeste besplatan softverski alat MPLAB X IDE, koji omogućava programiranje, simulaciju i debagovanje PIC mikrokontrolera. MPLAB X IDE je integrirano razvojno okruženje dizajnirano da radi na više operativnih sistema, uključujući Windows, macOS i Linux, što ga čini dostupnim širokom spektru programera bez obzira na preferiranu platformu. Posjeduje simulator koji omogućava testiranje i otklanjanje grešaka u kodu bez potrebe za stvarnim hardverom. Ovo je posebno korisno za rane faze razvoja i situacije u kojima je pristup hardveru ograničen.

U okviru projekta, napisan je C programski kod upravo korištenjem MPLAB X IDE razvojnog alata. Asembliranjem programa kreira se nekoliko fajlova, uključujući onaj sa ekstenzijom .hex koji se upisuje u mikrokontroler. Nakon uspješnog asembliranja programa obavlja se njegovo upisivanje u fizički sistem, PIC16F1939, pomoću programatora. Za potrebe provjere ispravnosti koda korišten je programator realiziran na pločici razvojnog sistema zajedno sa softverom PICPgm.

2. Sharp 2Y0A02 senzor

Sharp 2Y0A02 senzor, slika 3.3., je infracrveni senzor za mjerjenje udaljenosti proizведен od strane tvrtke Sharp Corporation. Namijenjen je aplikacijama za mjerjenje blizine i udaljenosti te koristi infracrvenu svjetlost kako bi odredio udaljenost između senzora i objekta. Ovaj tip senzora često se koristi u robotici, automatizaciji te drugim primjenama gdje su precizna mjerena udaljenosti bitna. Infracrveni senzori udaljenosti emituju snop infracrvene svjetlosti i detektuju signal reflektovan od objekta. Procjena udaljenosti do objekta obavlja se na temelju rekonstruiranog signala i pretpostavke da se infracrvena svjetlost reflektirala od objekta. Emitovani infracrveni signal se kodira zbog mogućnosti postojanja drugih izvora infracrvene svjetlosti, te se detektovani signali ignoriraju dok se ne detektuje onaj odgovarajućeg talasnog oblika.



Slika 3.3: Sharp infracrveni senzor

Neke od ključnih karakteristika koje ovaj senzor čine pogodnim za razne primjene su:

- analogni izlaz: analogni izlaz napona odgovara izmjerenoj udaljenosti i raste ili opada ovisno o detektiranoj udaljenosti,
- kompaktna veličina i jednostavno korištenje
- brz odziv

- niska cijena

Postoje i neka ograničenja koja treba uzeti u obzir prilikom rada sa ovim senzorom, poput utjecaja vanjskih faktora (npr. osvjetljenje, reflektirajuće površine), ograničen opseg mjerena i tačnost mjerena koja može varirati ovisno o reflektivnosti objekta, njegovom obliku i površinskim karakteristikama.

Dva Sharp 2Y0A02 senzora korištena u toku izrade projekta mogu prepoznati objekte na udaljenostima od 20 cm do 150 cm, te imaju nelinearnu karakteristiku. Za oba senzora određena je karakteristika podešavati udaljenost objekta koji reflektira infracrveni signal od senzora u opsegu od 10 cm do 150 cm, i sa korakom 10 cm. Izmjerene vrijednosti, date u tabeli 3.1, iskorištene su za određivanje parametara statičke karakteristike korištenjem programskog paketa MATLAB.

Udaljenost [cm]	Napon prvog senzora [V]	Napon drugog senzora [V]
10	2.88	2.76
20	2.55	2.45
30	1.96	1.86
40	1.49	1.42
50	1.18	1.15
60	0.98	0.94
70	0.85	0.82
80	0.71	0.70
90	0.65	0.62
100	0.57	0.54
110	0.50	0.47
120	0.45	0.40
130	0.40	0.34
140	0.37	0.34
150	0.34	0.30

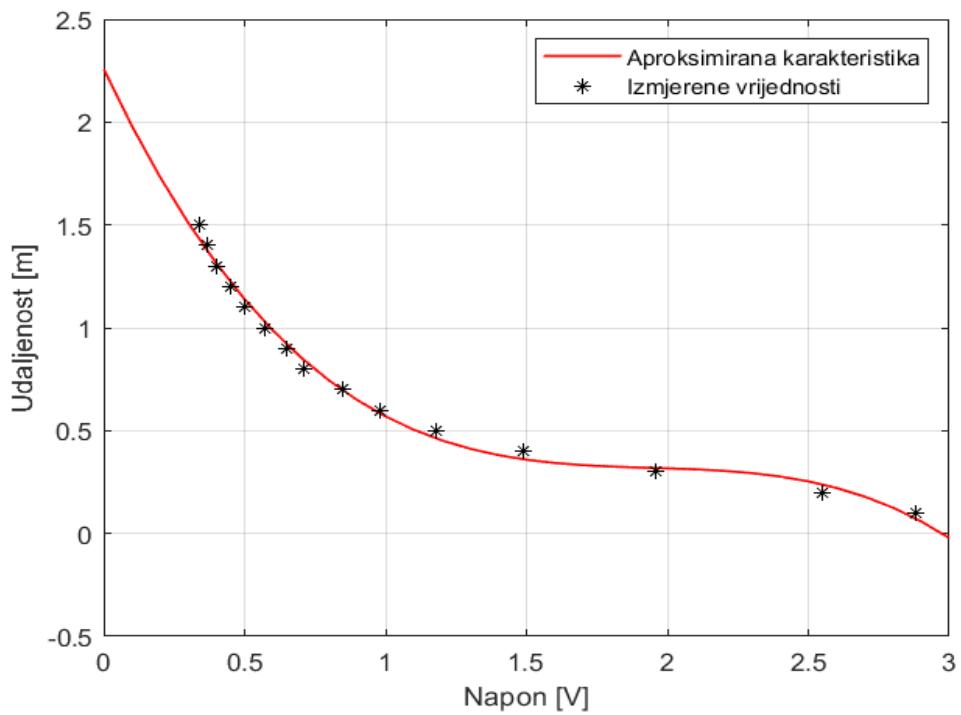
Tabela 3.1: Izmjerene vrijednosti napona za dva korištena senzora

Karakteristike senzora pretpostavljene su u obliku polinoma trećeg reda i određene korištenjem Curve Fitting Tool-a:

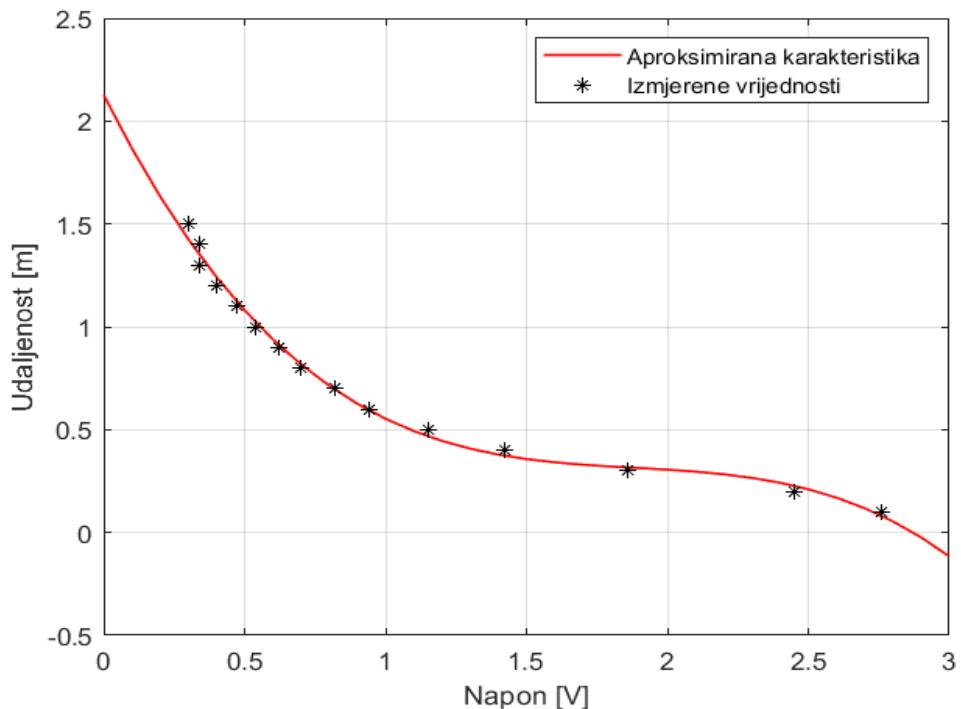
$$f(x) = -0.2539 \cdot x^3 + 1.48 \cdot x^2 - 2.915 \cdot x + 2.258 \quad (3.1)$$

$$f(x) = -0.2502 \cdot x^3 + 1.414 \cdot x^2 - 2.738 \cdot x + 2.127 \quad (3.2)$$

Aproksimirane karakteristike zajedno sa izmjerenim vrijednostima date su na slikama 3.4 i 3.5.



Slika 3.4: Karakteristika prvog senzora



Slika 3.5: Karakteristika drugog senzora

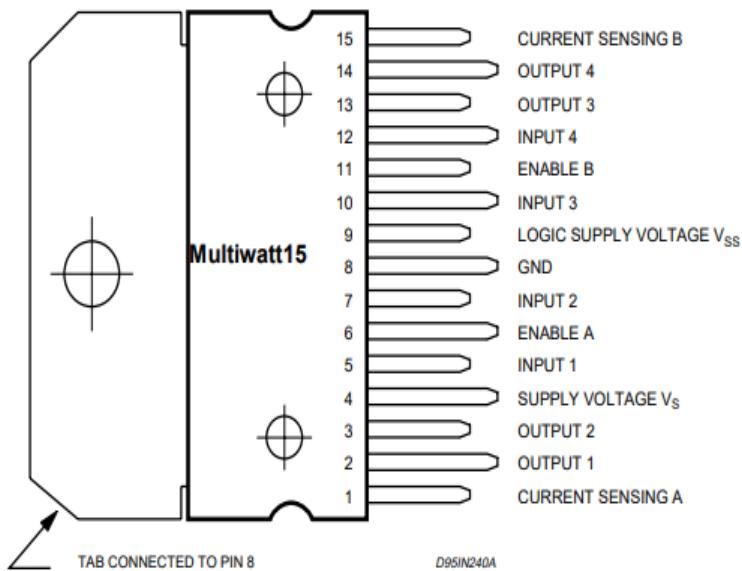
3. L298N Driver

Brzinom istosmjernog (DC) motora moguće je jednostavno upravljati kontrolom ulaznog napona u motor, a najčešći način za to jeste korištenje PWM signala. PWM ili širinsko impulsna modulacija je tehnika kojom se podešava prosječna vrijednost izlaznog napona. Ova tehnika se temelji na principu promjene širine impulsa signala, dok se period impulsa održava konstantnim. Širina impulsa predstavlja omjer vremena tokom kojeg je signal u visokom stanju (logička 1) u odnosu na ukupno trajanje perioda impulsa. Prosječna vrijednost napona ovisi o radnom ciklusu (duty cycle), odnosno o vremenu u kojem je signal uključen u odnosu na vrijeme u kojem je signal isključen u jednom periodu. Na ovaj način moguće je povećavati i smanjivati brzinu DC motora. Za kontrolu smjera rotacije motora, potrebno je samo obrnuti smjer struje kroz motor. Ovdje je najčešće prisutno korištenje H-mosta. Krug H-mosta sadrži četiri prekidačka elementa, obično tranzistora, s motorom u središtu. Istovremenim aktiviranjem dva prekidača obavlja se promijena smjera struje, a time i smjera vrtnje motora.

Kombiniranjem dvije metode, PWM i H most, moguće je u potpunosti upravljati DC motorom, kontrolom i brzine i smjera vrtnje. Integrirani krug koji se koristi za kontrolu i pogon istosmjernih (DC) motora u različitim aplikacijama jeste L298N. Široko se koristi u robotici, automatizaciji i elektroničkim projektima gdje je potrebna precizna kontrola motora. L298N sadrži dva H mosta, čime je omogućeno neovisno upravljanje s dva motora, te podržava dva načina kontrole, naprijed i unatrag. Raspored pinova prikazan je na slici 3.6. Za svaki motor postoje dva upravljačka ulaza, jedan omogućavajući ulaz i dva izlaza. L298N zahtijeva dva ulaza za napajanje:

- napon logičkog napajanja (Vss) koji se obično povezuje s niskonaponskim izvorom kako bi se napajala unutarnja logika upravljanja,
- napon motornog napajanja (Vs) koji je povezan je s izvorom višeg napona (od 5V do 35V) kako bi se napajali motori.

Za kontrolu L298N potrebno je osigurati odgovarajuće logičke signale na ulaznim pinovima (INPUT1 i INPUT2 za jedan motor, INPUT3 i INPUT4 za drugi motor) na temelju željenog smjera motora. Dodatno, moguće je prilagoditi razinu napona na ulaznom pinu (ENABLE A i ENABLE B) za kontrolu brzine motora. Za promjenjivu brzinu može se koristiti PWM. Također, L298N ima ugrađene otpornike za detekciju struje koji omogućavaju praćenje struje koju crpe motori. To može biti korisno za sprečavanje preopterećenja i zaštitu motora.



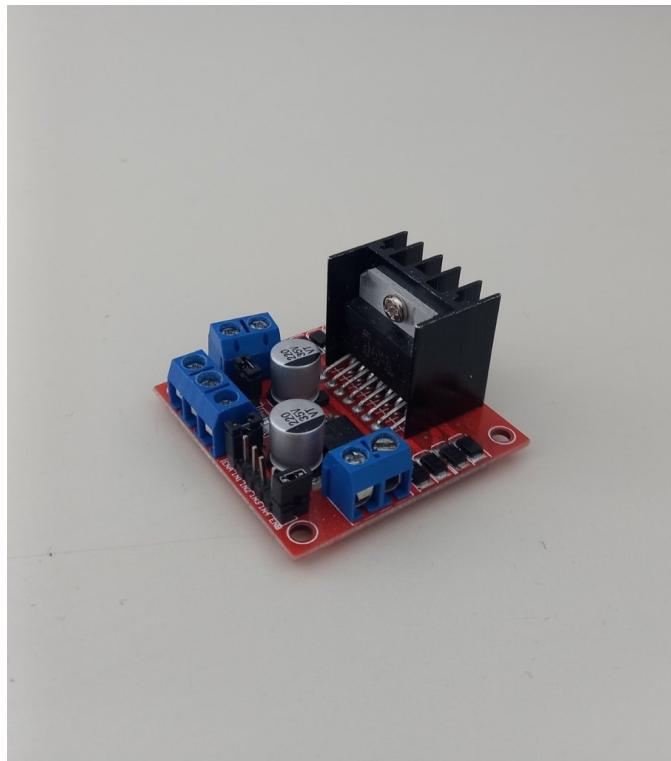
Slika 3.6: Raspored pinova L298N drivera

Na slici 3.7 prikazan je L298N modul s Arduinom za pogon istosmjernog motora. Modul ima 11 pinova čiji je raspored dat na slici 3.8. Ovaj modul ima dva pina za ulazno napajanje: VS i VSS. VS pin napaja unutarnji H-most koji pokreće motore i prihvata napon u rasponu od 5V do 12V. VSS se koristi za napajanje logičkog sklopa unutar L298N i kreće se između 5V i 7V. Na izlazne kanale OUT1 i OUT2, te OUT3 i OUT4, moguće je spojiti dva 5V-12V DC motora. Svaki kanal za DC motor može obezbijediti do 2A. Pinovi ENA i ENB koriste se za uključivanje i isključivanje motora, te kontrolu njihove brzine. Modul ima po dva pina za kontrolu smjera motora. Pinovi IN1 i IN2 kontroliraju smjer vrtnje prvog motora, dok IN3 i IN4 kontroliraju smjer vrtnje drugog motora. U tabeli 3.2 prikazane su različite kombinacije za ove pinove kao i rezultati koje daju.

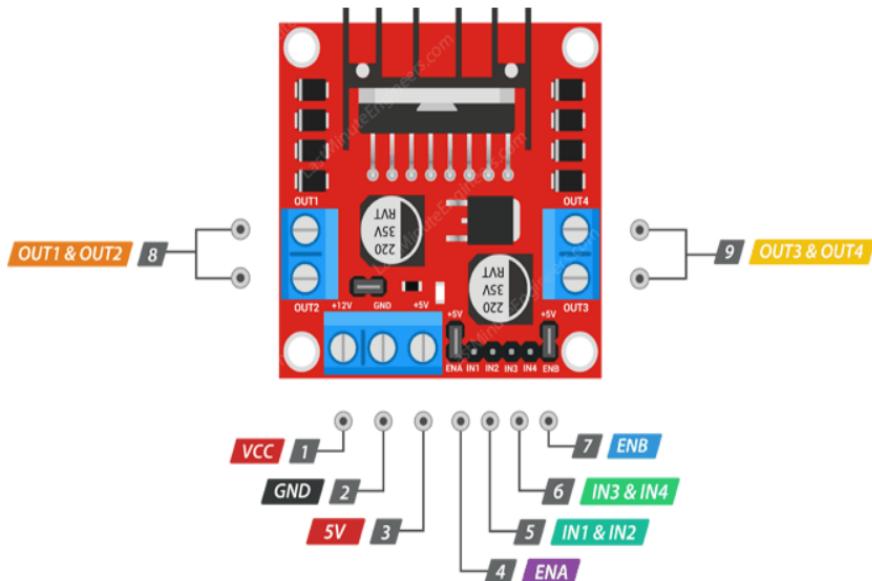
IN1	IN2	Smjer rotacije motora
Low(0)	Low(0)	Motor isključen
High(1)	Low(0)	Naprijed
Low(0)	High(1)	Nazad
High(1)	High(1)	Motor isključen

Tabela 3.2: Stanje motora za različite kombinacije

L298N ima pad napona od približno 2V zbog činjenice da unutar internog sklopa tranzistori imaju pad napona od približno 1 V. S obzirom da H-most zahtijeva da struja prođe kroz dva tranzistora, ukupan pad napona iznosi 2 V. Imajući ovo u vidu, ako se na pin za napajanje motora spoji 12V, motori će dobiti približno 10V. To znači da se oni neće vrtjeti punom brzinom ukoliko zahtijevaju napajanje od upravo 12V. Kako bi motor radio maksimalnom brzinom, napajanje motora mora imati napon koji je malo viši (oko 2V) od stvarnog zahtjeva za naponom motora.



Slika 3.7: L298N driver



Slika 3.8: Pinout modula

4. DC motor

DC motor je električni uređaj koji pretvara istosmjernu (DC) električnu energiju u mehaničku. Djeluje pomoću interakcije između magnetnog polja i vodiča kojim teče struja, obično namota žice. DC motori se koriste u raznim primjenama, kao što su vozila, robotika, industrijski strojevi i kućanski aparati. Nude preciznu kontrolu nad brzinom i smjerom, što ih čini svestranim

za mnoge zadatke.

DC motori se često koriste za različite funkcije unutar vozila, poput pokretanja električnih prozora, brisača, ventilatora za hlađenje, te u nekim slučajevima i za pogon u električnim vozilima i hibridnim vozilima. U kontekstu implementacije autonomnog vozila koje slijedi drugo vozilo, korištenje istosmjernih motora nudi praktično rješenje. Ovi DC motori mogu se koristiti za precizno upravljanje kretanjem i upravljanjem vozila. Prilagodbom brzine pojedinačnih motora, vozilom se može jednostavno upravljati i održavati sigurnu udaljenost od vozila ispred. Jednostavnost i upravljivost istosmjernih motora čine ih prikladnima za ovu primjenu, te je omogućena prilagodba u stvarnom vremenu kao i odgovarajući odziv na promjene u brzini i smjeru vodećeg vozila.

Za potrebe ovog projekta, korišten je IG32 24VDC 074 RPM motor, slika 3.9, odnosno dva ova motora, za pokretanje dva prednja točka vozila. Ovaj motor sa brušenim permanentnim magnetom je dizajniran da radi sa naponom od 24V i rotira brzinom od 74 RPM. Navedene vrijednosti su one kod kojih motor radi s najvećom učinkovitošću, a može isporučiti i više od svojih nazivnih vrijednosti.



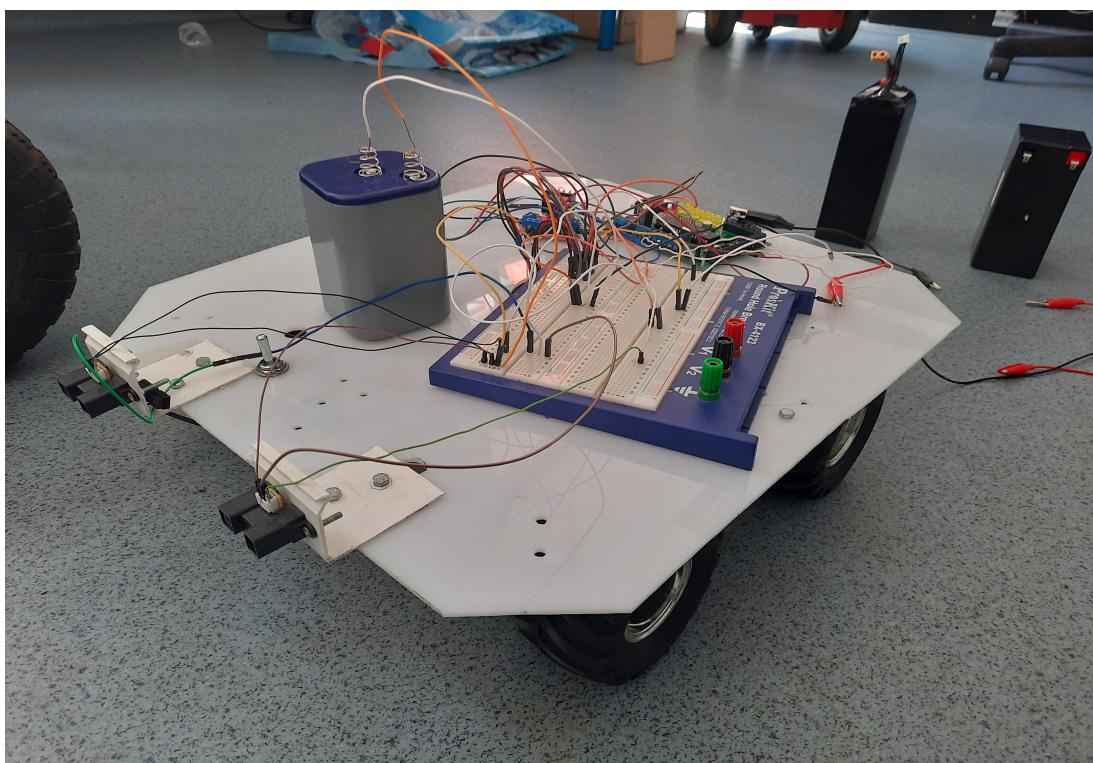
Slika 3.9: DC motor

Poglavlje 4



Realizacija projektnog zadatka

Sama izrada projekta obuhvata nekoliko bitnih koraka i uključuje hardverski i softverski dio. Početna faza uključivala je testiranje funkcionalnosti dva korištena DC motora. Potvrđeno je da oba motora ispravno rade, sa promjenom brzine u zavisnosti od napajanja, i to u oba smjera, naprijed i natrag. Ovaj je korak osigurao uspostavljanje temelja za kontrolirano kretanje. Dalje je slijedilo određivanje karakteristika IC senzora udaljenosti što je objašnjeno u jednom od prethodnih poglavlja. Ovaj korak je imao za cilj pojednostaviti integraciju podataka senzora u sistem upravljanja. Nakon što su testirane sve komponente i dobivene karakteristike i podaci potrebni za pisanje programskog koda, pristupilo se integraciji komponenti. Senzori, motori i driver za njihovo pokretanje i upravljanje, spojeni su na odgovarajući način, korištenjem dostupnih datasheetova, zajedno sa PIC mikrokontrolerom. Ova faza je uključivala ožičenje i osiguravanje spremnosti fizičkih komponenti za interakciju s kontrolnim sistemom. Konačan izgled sistema sa svim komponentama prikazan je na slici 4.1.



Slika 4.1: Autonomno vozilo

Što se tiče softverskog dijela zadatka, kod za PIC mikrokontroler napisan je u programskom jeziku C koristeći MPLAB X IDE. Sam kod se sastoji iz nekoliko dijelova te sadrži više funkcija. Prvi dio postavlja konfiguraciju mikrokontrolera, poput osnovnih postavki oscilatora, tajmera i slično. Zatim su definisane konstante: maksimalni PWM duty cycle, željena udaljenost od drugog vozila, prag udaljenosti, maksimalna i minimalna udaljenost od senzora, te prototipovi funkcija koje će se koristiti kasnije u kodu. Napisane su odvojene funkcije za inicijalizaciju AD konverzije, inicijalizaciju pinova i inicijalizaciju PWM modula. U ovim funkcijama konfigurisani su pinovi AN0 i AN1 kao analogni ulazi za uzimanje podataka sa senzora, RD4 i RD5 kao digitalni izlazi za upravljanje uključenjem/isključenjem motora i RC1 kao PWM izlaz. Ove tri funkcije za inicijalizaciju pozivaju se unutar main funkcije. Također, napisane su funkcije za čitanje napona sa senzora i njegovu konverziju u udaljenost korištenjem karakteristike za lijevi i desni senzor. Unutar main funkcije pokreće se beskonačna while petlja unutar koje se kontinuirano, pokretanjem AD konverzije, čitaju vrijednosti sa senzora, konvertuju u udaljenosti i zatim porede sa referentnim vrijednostima. Na osnovu ovih poređenja, korištenjem if uslova, pozivaju se funkcije sa upravljanje smjerom i brzinom motora. Ovo su funkcije Direct_Movement(), Turn_Right() i Turn_Left() i one postavljaju brzinu i smjer rotacije motora u zavisnosti od udaljenosti senzora sa desne i lijeve strane vozila. Unutar ovih funkcije poziva se funkcija Speed() koja postavlja brzinu motora.

Prije prelaska na fizičku implementaciju, obavljena je simulacija unutar MPLAB X IDE okruženja. Ovaj korak je omogućio provjeru logike i funkcionalnosti koda bez potrebe za njegovim pokretanjem na stvarnom hardveru. Nakon uspješne simulacije pristupilo se programiranju PIC mikrokontrolera i testiranju sistema, što je također uspješno realizirano. Jedna poteškoća prilikom hardverske realizacije bilo je napajanje sistema. Zbog težine baterija, slika 4.2, vozilo se kretalo otežano, te je bilo potrebno skloniti baterije sa platforme, tj. vozila. Nakon što se prevazišao ovaj problem testiranje sistema je obavljeno uspješno.



Slika 4.2: Napajanje sistema

Poglavlje 5



Zaključak

Glavni cilj ovog projekta je sticanje naprednog znanja iz područja upravljanja kretanjem, ugradbenog programiranja i sistema upravljanja. Fokus jeste na razvoju funkcionalnog sistema koji djeluje u zatvorenom prostoru. Jedan od ključnih aspekata projekta je postizanje autonomne funkcije kretanja, što znači da sistem može samostalno donositi odluke o svom kretanju, bez ljudske kontrole, koristeći senzore i algoritme za donošenje odluka. Da bi se postigla autonomna funkcija kretanja, pristupilo se programiranju mikrokontrolera. Ovdje je uključeno korištenje mikrokontrolera za upravljanje motorima i senzorima za detekciju okoline. Kompletan sistem predstavljen je blok strukturom unutar koje su navedeni i podsistemi, te je na ovaj način olakšanja konačna realizacija sklopa.

U toku izrade autonomnog vozila koje prati drugo vozilo, došlo je do dva ključna problema. Prvi problem se odnosio na težinu baterija, budući da su tri baterije rezultirale prevelikom težinom vozila i povećanim trenjem, što je ometalo njegovo kretanje. Kako bi se prevazišao ovaj problem i postigla bolja pokretljivost, dva lijeva i dva desna motora su serijski spojena. Drugi problem bili su loši konektori za senzore. Ovaj problem se jednostavno rješava lemljenjem, što će omogućiti pouzdanu i stabilnu komunikaciju između senzora i sistema upravljanja vozilom. Autonomno vozilo koje prati drugo vozilo predstavlja izazovan i perspektivan domen razvoja autonomne tehnologije. Izazovi poput upravljanja težinom, poboljšanja pokretljivosti i pouzdane komunikacije senzora ključni su faktori za uspjeh ovakvih projekata. Ova tehnologija ima potencijal za primjenu u različitim industrijama, uključujući transport, logistiku i mnoge druge, pružajući potencijalne koristi u efikasnosti i bezbjednosti.

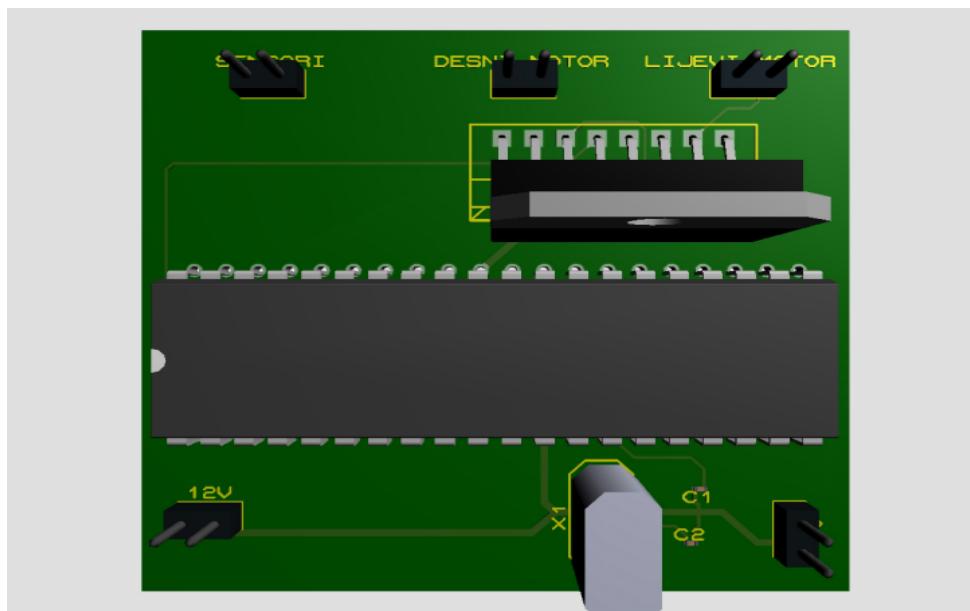
Poglavlje 6



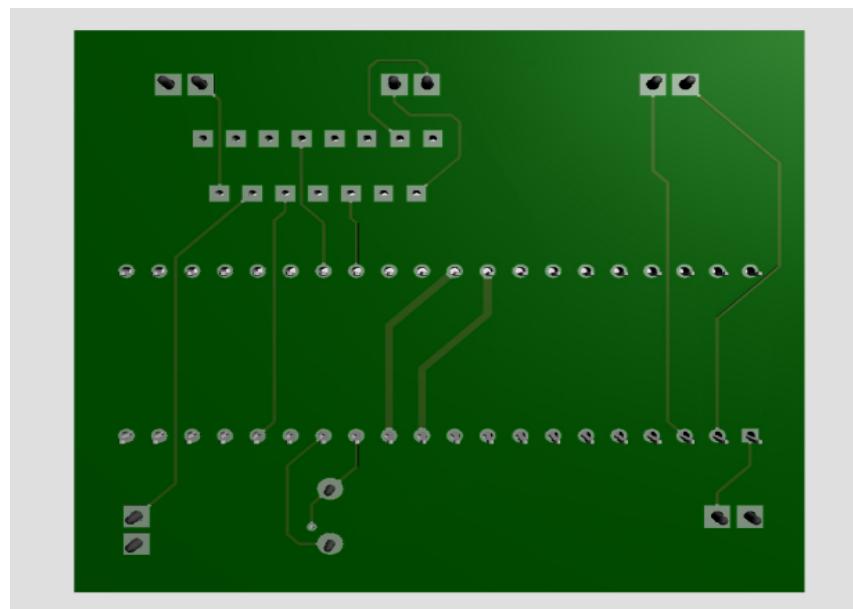
Dodatak

U ovom dijelu rada naveden je upotrebljeni kod. Kod sadrži kratke komentare koji pojašnjavaju ključne dijelove rada programa. Također dat je i pcb dizajn. Na slikama 6.1, 6.2, 6.3 i 6.4 date su pcb sheme.

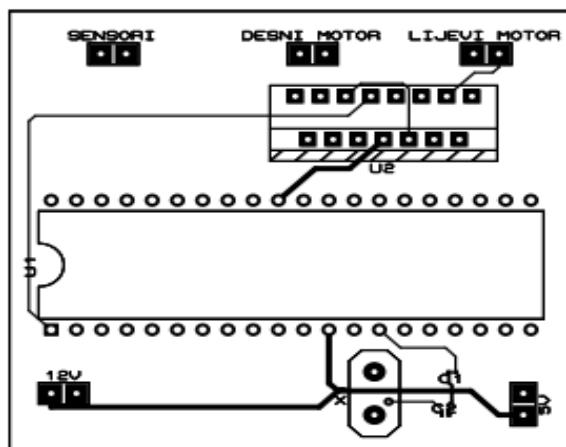
1. PCB dizajn



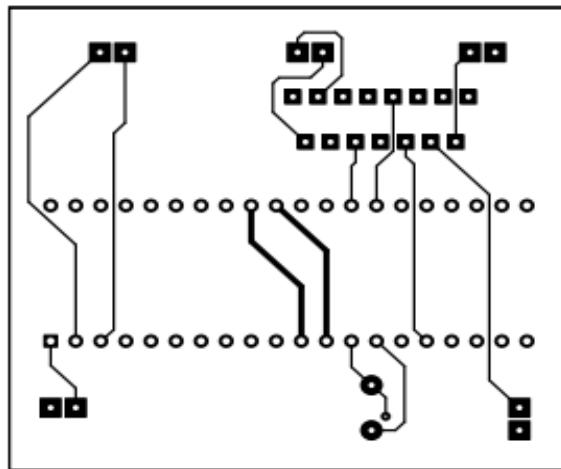
Slika 6.1: PCB 3D izgled gornja strana



Slika 6.2: PCB 3D izgled donja strana



Slika 6.3: PCB dizajn gornja strana



Slika 6.4: PCB dizajn donja strana

2. Kod

```

1 #include <xc.h>
2 #pragma config FOSC = HS, WDTE = OFF, PWRTE = OFF, MCLRE = ON,
   CP = OFF, CPD = OFF, BOREN = OFF, CLKOUTEN = OFF
3 #pragma config IESO = OFF, FCMEN = OFF, WRT = OFF, VCAPEN = OFF
   , PLLEN = OFF, STVREN = OFF, LVP = OFF
4 #define _XTAL_FREQ 8000000
5 // Konstante
6 #define PWM_MAX 255           // Max PWM duty cycle
7 #define DESIRED_DISTANCE 40    // Zeljena udaljenost od vozila
   (0.4 m)
8 #define DISTANCE_THRESHOLD 10 // Prag
9 #define MAX_DISTANCE 130      // Max udaljenost od senzora
   (1.3 m)
10 #define MIN_DISTANCE 20
11 // Prototipovi funkcija
12 void PWM_setup();
13 void Speed(int speed);
14 void Direct_Movement(int distance);
15 void Turn_Right(int distance);
16 void Turn_Left(int distance);
17 // Inicijalizacija AD konverzije
18 void setupAD() {
19     TRISA = 0xEF;
20     ANSELA = 0xFF;
21     ADCON1bits.ADFM = 0;
22     ADCON1bits.ADCS2 = 1;
23     ADCON1bits.ADCS1 = 1;

```

```

24 ADCON1bits.ADCS0 = 1;
25 ADCON1bits.ADNREF = 0;
26 ADCON1bits.ADREF1 = 0;
27 ADCON1bits.ADREF0 = 0;
28 }
29 void setup() {
30     TRISD = 0x00; // PORTD as output
31     ANSELD = 0x00;
32     PORTD = 0x00;
33     TRISC = 0x00; // PORTC as output
34     PORTC = 0x00;
35 }
36 // Citanje vrijednosti sa senzora
37 int Read_Value_From_Sensor(int channel)
38 {
39     if (channel == 0) {
40         ADCON0bits.ADON = 1;
41         ADCON0bits.CHS4 = 0;
42         ADCON0bits.CHS3 = 0;
43         ADCON0bits.CHS2 = 0;
44         ADCON0bits.CHS1 = 0;
45         ADCON0bits.CHS0 = 0;
46     }
47     else {
48         ADCON0bits.ADON = 1;
49         ADCON0bits.CHS4 = 0;
50         ADCON0bits.CHS3 = 0;
51         ADCON0bits.CHS2 = 0;
52         ADCON0bits.CHS1 = 0;
53         ADCON0bits.CHS0 = 1;
54     }
55     ADCON0bits.ADGO = 1;
56     while (ADCON0bits.ADGO);
57     unsigned int volt = ADRESH;
58     int ones = (volt * 5.0 / 255);
59     int decimals = (int)(100 * (5.0 * volt/ 255 - ones));
60     int Voltage = ones * 1000 + decimals * 10; //u mV
61     return Voltage;
62 }
63 // Proracun udaljenosti
64 int Calculate_Distance(int Voltage, int sensor_channel)
65 {
66     int distance = Voltage;
67     if(sensor_channel == 1) {
68         // lijevi senzor
69         distance = -0.2539 * 100 * Voltage*Voltage*Voltage
/1000000000 + 1.48 * 100 * Voltage*Voltage /1000000 -2.915 *
100* Voltage /1000 + 2.258 * 100;

```

```

70     }
71     else {
72         // desni senzor
73         distance = -0.2502 * 100 * Voltage*Voltage*Voltage
74             /1000000000 + 1.414 * 100 * Voltage*Voltage /1000000 -2.738
75             * 100 * Voltage/1000 + 2.127 * 100;
76     }
77     return distance;
78 }
79
80 void main() {
81     setupAD();
82     setup();
83     PWM_setup();
84     while (1) {
85         int sensor1Channel = 0; // Lijevi senzor
86         int sensor2Channel = 1; // desni senzor
87         // Uzimanje vrijednosti napona sa senzora
88         int Voltage_Sensor1 = Read_Value_From_Sensor(
89         sensor1Channel);
90         __delay_ms(10);
91         int Voltage_Sensor2 = Read_Value_From_Sensor(
92         sensor2Channel);
93         // Racunanje udaljenosti
94         int distance_Sensor1 = Calculate_Distance(
95         Voltage_Sensor1, 1);
96         int distance_Sensor2 = Calculate_Distance(
97         Voltage_Sensor2, 2);
98         // Podesavanje brzine i smjera rotacije motora
99         if (distance_Sensor2 > distance_Sensor1 +
100             DISTANCE_THRESHOLD) {           // skreni lijevo
101             Turn_Left(distance_Sensor1);
102         }
103         else if (distance_Sensor2 < distance_Sensor1 -
104             DISTANCE_THRESHOLD) {        // skreni desno
105             Turn_Right(distance_Sensor2);
106         }
107         else if ( (distance_Sensor2 > DESIRED_DISTANCE +
108             DISTANCE_THRESHOLD && distance_Sensor1 > DESIRED_DISTANCE +
109             DISTANCE_THRESHOLD) ||
110             (distance_Sensor2 < DESIRED_DISTANCE -
111             DISTANCE_THRESHOLD && distance_Sensor1 < DESIRED_DISTANCE -
112             DISTANCE_THRESHOLD) ) {
113             Direct_Movement((distance_Sensor1 +
114                 distance_Sensor2)/2); //prilagodi brzinu ako je van zeljene
115                 udaljenosti
116         }

```

```

103         else if (distance_Sensor2 < MIN_DISTANCE &&
104             distance_Sensor1 < MIN_DISTANCE) {
105             Speed(0);
106         }
107         else {
108             PORTD = 0x30;
109         }
110     }
111 }
112
113 void PWM_setup()
114 {
115     PR2 = 0xFF;
116     CCP2M3 = 1;
117     CCP2M2 = 1;
118     // PWM duty cycle
119     CCPR2L = 0;
120     // Timer2
121     T2CKPS0 = 1; // prescaler 16
122     T2CKPS1 = 1;
123     TMR2ON = 1;
124 }
125
126 void Direct_Movement(int distance)
127 {
128     int speed = (PWM_MAX * distance) / MAX_DISTANCE;
129     PORTDbits.RD4 = 1;
130     PORTDbits.RD5 = 1;
131
132     // Postavi brzinu
133     Speed(speed);
134 }
135
136 void Turn_Right(int distance)
137 {
138     int speed = (PWM_MAX * distance) / MAX_DISTANCE;
139     PORTDbits.RD4 = 0;
140     PORTDbits.RD5 = 1;
141
142     Speed(speed);
143 }
144
145 void Turn_Left(int distance)
146 {
147     int speed = (PWM_MAX * distance) / MAX_DISTANCE;
148     PORTDbits.RD4 = 1;
149     PORTDbits.RD5 = 0;

```

```
150
151     Speed(speed);
152 }
153
154 void Speed(int speed)
155 {
156     // ogranicenje brzine
157     if(speed > PWM_MAX)
158         speed = PWM_MAX;
159     else if(speed < 0)
160         speed = 0;
161
162     // PWM duty cycle
163     CCPR2L = speed;
164 }
```