Izbjegavanje sudara s preprekom pomoću ultrazvučnog senzora za maketu autonomnog vozila

Ćaleta, Mislav

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:186074

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IZBJEGAVANJE SUDARA S PREPREKOM POMOĆU ULTRAZVUČNOG SENZORA ZA MAKETU AUTONOMNOG VOZILA

Završni rad

Mislav Ćaleta

Osijek, 2021.

Sadržaj

1. UVOD 1

1.1. zadatak završnog rada 1

2. SKLOPOVLJE KORIŠTENO U RADU 1

2.1. Arduino UNO razvojna pločica 2

2.2. HC-SR04 ultrazvučni senzor 5

2.2.1. Brzina zvuka u zraku 5

2.2.2. Princip rada HC-SR04 5

2.3. Termistor 7

2.4. L298N DC motor driver 9

2.4.1. H-most 9

2.4.2. L298N modul 10

2.5. Arduino senzorski štit V5 12

2.6. Servomotor 14

2.7. Ostale komponente 16

3. PROGRAMSKI JEZIK KORIŠTEN U RADU (C) 17

3.1. Opće osnove programskog jezika C 17

3.1.1. Tipovi podataka, deklaracija i inicijalizacija varijabli 17

3.1.2. Petlje u C programskom jeziku 18

3.1.3. Naredbe grananja 22

3.1.4. Funkcije 24

3.1.5. Jednodimenzionalna i dvodimenzionalna polja 26

4. IZRADA ELEKTRIČNE SHEME I SKLOPOVLJA VOZILA 29

4.1. Izrada sustava za pokretanje i napajanje 29

4.2. Izrada senzorskog sustava 34

5. PROGRAMIRANJE MAKETE VOZILA 37

5.1. Postavljanje početnih varijabli i konstanti 37

5.2. Podešavanje priključaka i postavljanje početnih stanja 40

5.3. Funkcija za izračunavanje temperature zraka 41

5.4. Fukcije za izračun brzine zvuka i timeouta 42

5.5. Funkcije za kretanje vozila 43

5.6. Funkcija za mjerenje udaljenosti 45

5.7. Funkcija za pronalazak novog smjera kretanja nakon nailaska na prepreku 46

5.8. Glavna petlja programa, funkcija loop() 48

6. ZAKLJUČAK 49

LITERATURA 50

SAŽETAK 52

ABSTRACT 53

PRILOG 54

1. UVOD

U ovom radu opisuje se postupak izbjegavanja sudara s preprekom pomoću ultrazvuka za maketu autonomnog vozila, kao i izrada električne sheme i programiranje takve makete. Također su opisani alati koji se koriste pri izradi električne sheme i pri implementiranju algoritma. Kako bi se spriječio sudar vozila s preprekom na koju nailazi koristi se ultrazvučni senzor. Vozilo će se kretati pravocrtno i u određenim vremenskim intervalima ispred sebe ispustiti zvučni val. Zvučni će val zatim naići na prepreku i dogodit će se refleksija vala u smjeru njegovog izvorišta, što je u ovom slučaju maketa vozila. Pomoću mjerenja vremena koje je valu potrebno da se vrati i iz izračunate brzine zvuka iz mjerene temperature zraka, dolazi se do udaljenosti od prepreke do vozila. Zatim ako je udaljenost između vozila i prepreke premala, ono se zaustavlja i ultrazvučni senzor pomoću rotacijskog servomotora okreće u određenom kutu i ispušta zvučne valove u potrazi za dovoljno udaljenom preprekom prema kojoj se sigurno kretati. Ukoliko do toga ne dođe, samo vozilo se okreće i opet ponavlja pretragu pomoću rotiranja senzora. Kako bi se potrebne informacije obradile i kako bi vozilo izvelo ovakve operacije potreban je i mikroupravljač. Korišteni ATmega328P mikroupravljač nalazi se u sklopu Arduino UNO razvojne pločice. Pomoću Arduino UNO pločice upravlja se navedenim operacijama a za njegovo programiranje koristit će se C programski jezik.

1.1. zadatak završnog rada

Izraditi maketu vozila upravljanu pomoću odgovarajućeg mikroupravljača, koja će imati sposobnost izbjegavati sudar s preprekama. Za detekciju prepreka korisiti ultrazvučni senzor.

2. SKLOPOVLJE KORIŠTENO U RADU

U ovom poglavlju opisane su fizičke komponente korištene pri izradi električne sheme za maketu vozila.

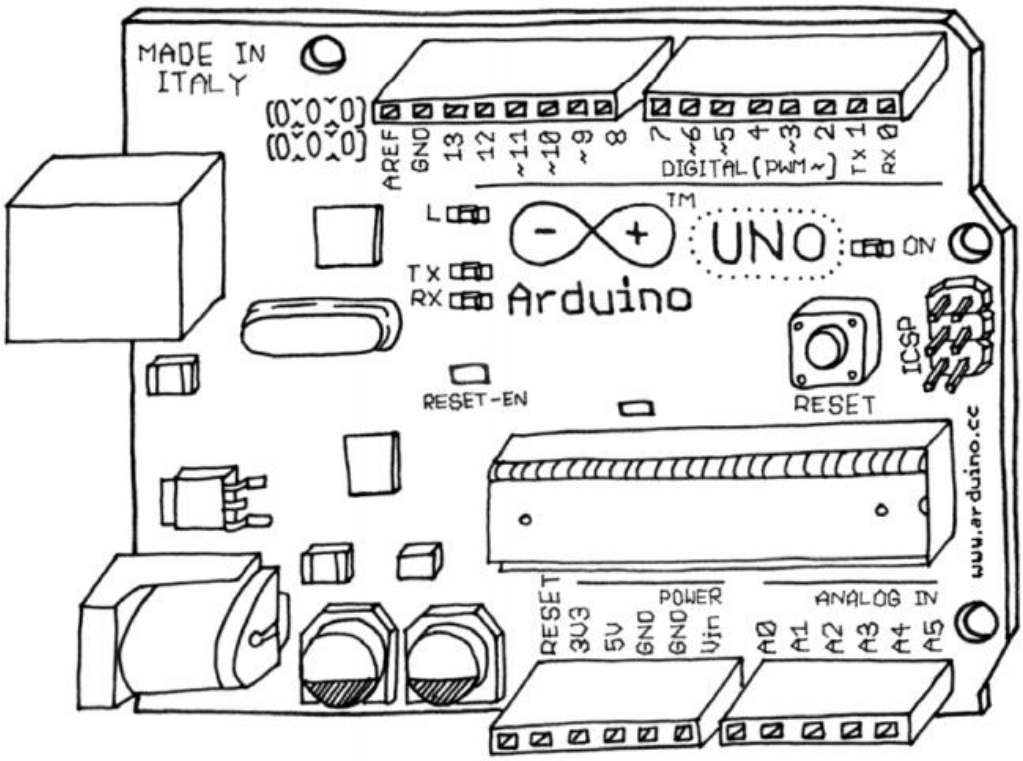
2.1. Arduino UNO razvojna pločica

U sklopu Arduino UNO platforme nalazi se mikroupravljač ATmega328P. Mikroupravljač je integrirani sklop unutar kojeg se nalaze razni djelovi koji čine mikroupravljač kao što je procesor, radna memorija, programska memorija itd. (slika 2.1.). Kako bi mikroupravljač izvodio željene operacije za njega treba napisati program. Pomoću ostatka Arduino platforme, kao što su ulazni i izlazni priključci i ostatak periferije, upravlja fizičkim komponentama koje su spojene na razne priključke i omogućeno je programiranje mikroupravljača pomoću računala. Prema [1, str. 15], „Sklopovlje koje se nalazi oko njega služi kako bi se mikroupravljač mogao pokrenuti i kako bismo u njega mogli ubaciti program koji pišemo na računalu. Komunikacija Arduino UNO pločice i računala odvija se preko USB priključka."

Slika 2.1. ATmega328P mikroupravljač [1]

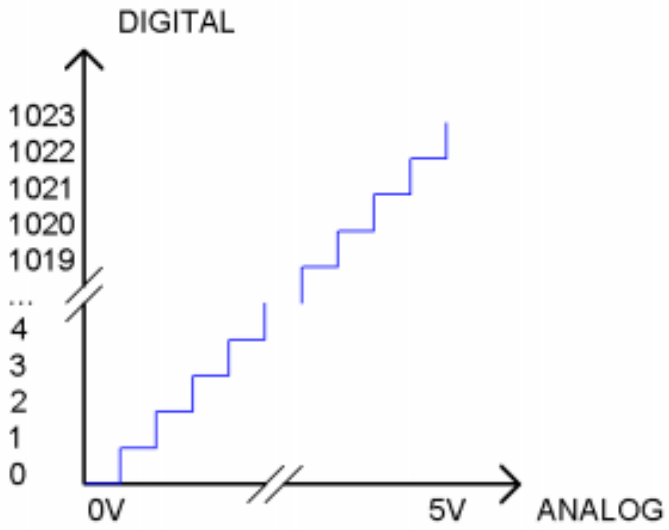
Prema [2, str. 1], Arduino je otvorena fizička računalna platforma zasnovana na jednostavnoj ulazno/izlaznoj ploči i razvojnom okruženju koje implementira procesni jezik. Budući da je arduino platforma otvoreni sustav, to ga čini idealnim alatom za izradu prototipova. Vrlo je jednostavan i fleksibilan za korištenje. Služi za opću primjenu i zato nije najbolji alat za specifične zadatke u industriji, ali rijetko može poslužiti i toj svrsi. Još jedna od prednosti Arduino razvojne pločice je sposobnost nadogradnje. Prema [3, str. 23], Jedna od velikih prednosti arduino sustava je njihova

sposobnost proširenja – to jest, vrlo je lako dodati nove funkcije u sklopovlju. Periferija Arduino UNO razvojne pločice sastoji se od nekoliko priključaka. (Slika 2.2.)



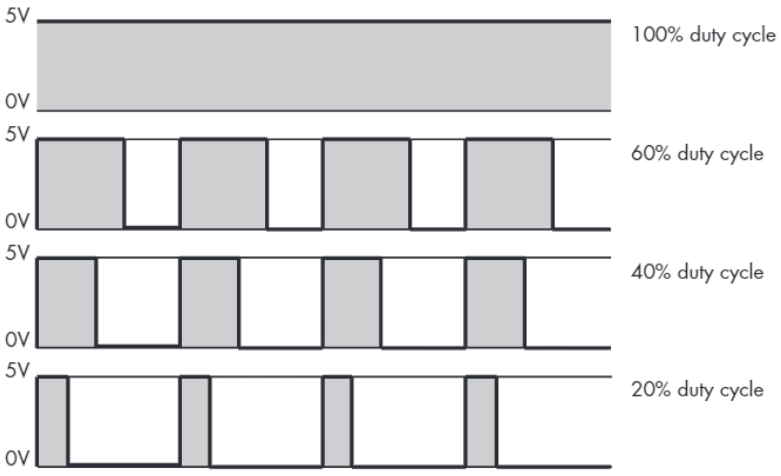
Slika 2.2. Arduino UNO razvojna pločica [2]

Postoji 14 digitalnih ulazno izlaznih priključaka, što znači da će takvi priključci davati ili očitavati samo vrijednosti visoko ili nisko. Ti priključci su označeni s brojevima od 0 do 13 i mogu služiti kao izlaz ili kao ulaz ovisno o tome kako smo ih definirali u programu. Visoku vrijednost na priključku predstavlja 5V dok će nisku vrijednost predstavljati 0V. Također svaki digitalni priključak ima i ugrađeni pull-up otpornik. Iduća skupina priključaka na pločici su analogni ulazni priključci koji su označeni s A0 do A5. Oni očitavaju analognu vrijednost napona i konvertiraju je u broj između 0 do 1023 koristeći analogno-digitalni 10 bitni pretvarač. Analogno-digitalni pretvarač pretvara analogne vrijednosti u cijele brojeve prema slici 2.3.



Slika 2.3. Ovisnost vrijednosti pretvarača o analognom naponu

Treća skupina priključaka su digitalni priključci koji se mogu reprogramirati kako bi proizveli analogni napon. To su priključci 3, 5, 6, 9, 10 i 11. Takav napon radi se putem modulacije širine impulsa. Što znači da će analogni napon na izlazu ovisiti o postotku perioda signala kojega zauzima visoka razina (eng. duty cycle). Što veći postotak perioda zauzima visoka razina, to će veći biti analogni napon na izlazu (Slika 2.4.).



Slika 2.4. Modulacija širine pulsa [3]

2.2. HC-SR04 ultrazvučni senzor

2.2.1. Brzina zvuka u zraku

Zvuk je longitudalni val. Prema [4, str. 108.], „Zvuk je najvažniji primjer longitudalnog mehaničkog vala. U širem smislu zvuk obuhvaća sve longitudalne elastične valove u čvrstim tijelima, tekućinama i plinovima.” Za ovaj projekt računa se udaljenost od izvora zvučnog vala do objekta odbijanja. Kako bi se došlo do te udaljenosti trebamo podatak o brzini zvuka u plinu koji okružuje vozilo. Taj plin bit će zrak, pa se ta brzina može izračunati prema izrazu

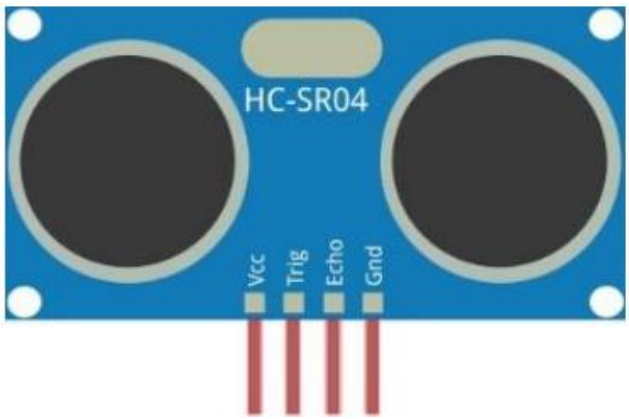
$$v = \sqrt{x\frac{RT}{M}} \tag{2-1}$$

gdje je R = 8,314 J/mol K, plinska konstanta, T apsolutna temperatura u kelvinima, x adijabatski koeficijent koji za zrak iznosi 1.4, a M molarna masa plina koja za suhi zrak iznosi0.0289644 kg/mol.

2.2.2. Princip rada HC-SR04

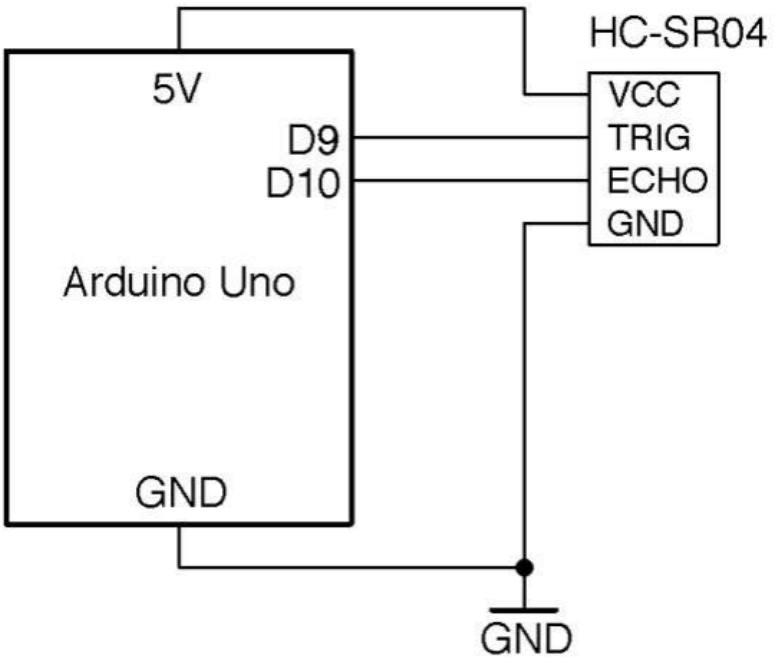
Jedna od glavnih komponenti fizičke makete vozila koja se izrađuje u ovom radu je HC-SR04 ultrazvučni senzor. HC-SR04 za ovaj projekt sasvim je dovoljan, ali za veće udaljenosti treba potražiti drugačije senzore. Prema [5, str. 710.], Za udaljenosit između 4 inča (10 cm) i 6 stopa (2 m) koristite HC-SR04 modul. Pomoću njega ranije spomenuti mikroupravljač dobit će podatke iz okoline potrebne za prepoznavanje prepreka. Funkcionira tako što ispušta ultrazvuk frekvencije 40kHz. Zbog te frekvencije ispuštenog vala znamo da taj zvuk možemo nazivati ultrazvukom. Prema [4, str. 124.], „Mehanički valovi frekvencije veće od 20 kHz pripadaju području ultrazvuka.” Kada ultrazvuk naiđe na prepreku on se odbija i vraća prema senzoru. Ako uzmemo u obzir brzinu zvuka, koju ćemo izračunati po izrazu (2-1), možemo izračunati udaljenost od prepreke do senzora, odnosno vozila. Priključci koji se nalaze na HC-SR04 modulu su VCC, TRIG, ECHO, GND (slika 2.5.). VCC priključak se priključuje na +5V kao napon napajanja, dok se GND priključuje na uzemljenje. Kako bi se generirao ultrazuvčni val treba postaviti TRIG (trigger) priključak na visoku vrijednost koja će trajati 10 µs. U isto vrijeme kada se generira ultrazvučni val, na ECHO priključku postavlja se

visoka vrijednost i traje sve dok se val ne vrati ili prođe određeno vrijeme. Na taj način mjerenjem vremena trajanja visokog napona na ECHO priključku, ustvari mjerimo približno vrijeme koje je ultrazvučnom valu trebalo da prijeđe put od senzora do prepreke i nazad. Iz ovoga možemo zaključiti da dijeljenjem tog vremena s dva, dobijemo vrijeme koje je ultrazvučnom valu potrebno da prijeđe put od senzora do prepreke.



Slika 2.5. HC-SR04 ultrazvučni senzor

Želi li se postići željeno ponašanje, odnosno želi li se mijenjati razina napona za TRIG priključak i mjeriti veličina napona s ECHO priključka, treba napraviti elektronsku konfiguraciju prema slici 2.6. gdje bi se pomoću D9 priključka upravljalo ispuštanjem ultrazvučnog vala.



Slika 2.6. Povezivanje HC-SR04 na arduino [5]

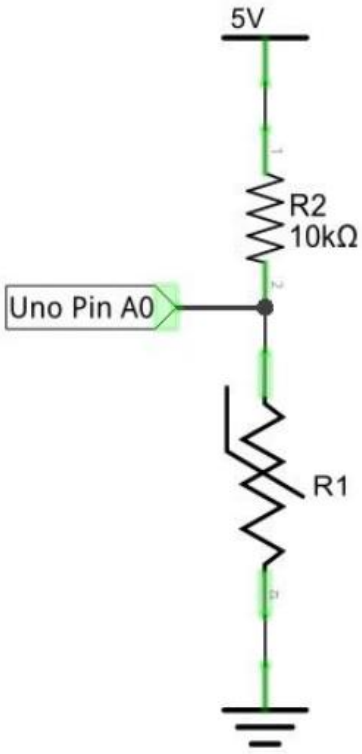
2.3. Termistor

Prema [6, str. 53.], „Termistori su poluvodički otpornici s negativnim temperaturnim koeficijentom (porastom temperature smanjuje im se otpor). Promjenom temperature poluvodiča mijenja se i koncentracija slobodnih nosilaca, pa prema tome i njegova provodnost, odnosno porastom temperature raste koncentracija nosilaca naboja i provodnost poluvodiča, a njegov otpor opada.” U ovom radu termistor ima ulogu temperaturnog senzora, odnosno pomoću njega dolazimo do temperature okoline koju ćemo uvrštavati u izraz (2-1) kako bi došli do brzine zvuka. Do temperature možemo doći koristeći izraz

$$Rt = R \cdot e^{B \cdot (\frac{1}{T_2} - \frac{1}{T_1})} \qquad (2\text{-}2)$$

gdje je *Rt* otpor termistora na temperaturi *T2*, *R* otpor termistora na temperaturi *T1*, *B* termalni indeks, *T1* i *T2* temperature u kelvinima. Parametri koji se koriste za termistor u ovom radu su *R* = 10kΩ, *B* = 3950, *T1* = 25K. Do otpora *Rt* dolazi se tako što se mjeri trenutni pad napona na otporniku i iz njega

se može doći do otpora, a zatim se pomoću otpora dolazi do temperature *T2* koja predstavlja temperaturu na kojoj se u trenutku mjerenja nalazi termistor, odnosno temperaturu okoline. Termistor kakav se koristi u radu može se vidjeti na slici 2.8., a najjednostavnija konfiguracija za mjerenje napona koristeći Arduino UNO, kakva se koristi i u ovom radu, vidi se na slici 2.7.



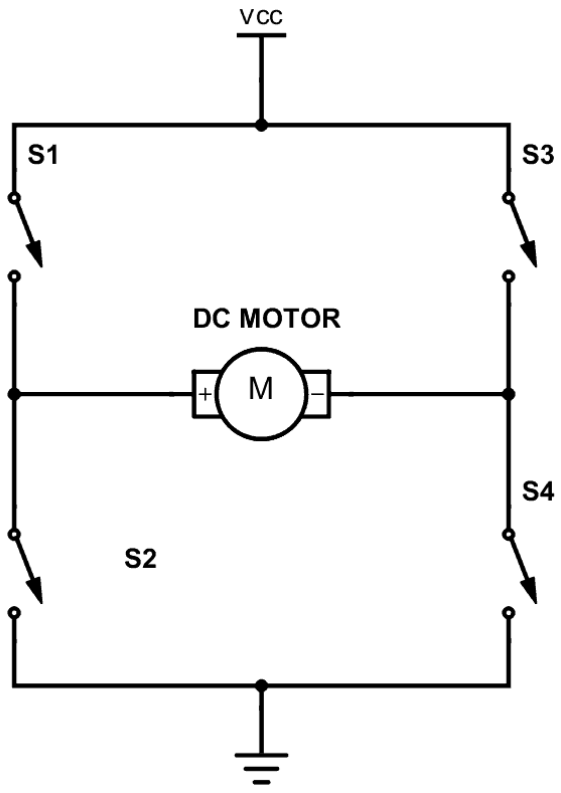
Slika 2.7. Spoj termistora za mjerenje pada napona

Slika 2.8. Primjer termistora

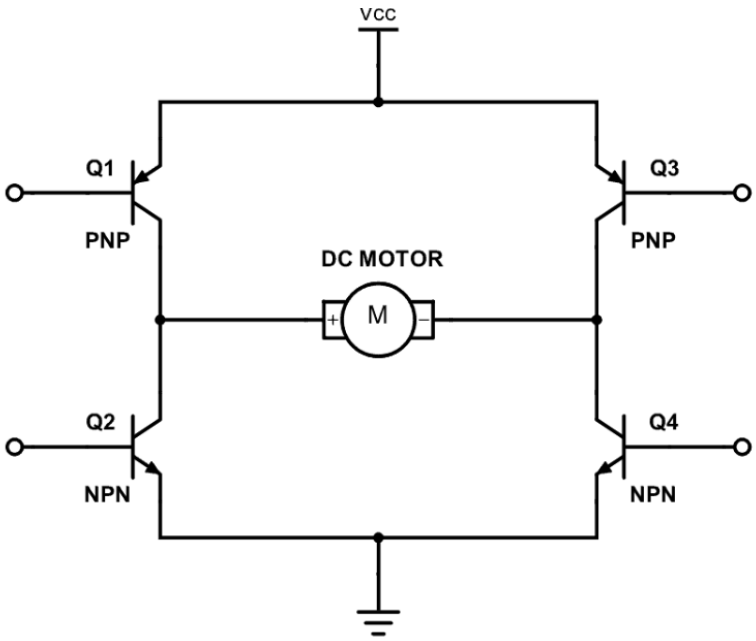
2.4. L298N DC motor driver

2.4.1. H-most

Prema [7], H-most je jednostavni sklop koji dopušta upravljanje smjera DC motora prema naprijed ili nazad. Pomoću H-mosta koji se nalazi u L298N komponenti upravlja se DC motorima koji će okretati kotače vozila. Prema [8], L298N pokretač motora je upravljač koji koristi H-most kako bi se lagano upravljalo brzinom i smjerom 2 motora. DC motor se okreće u smjeru ovisno o tome kako se spoje njegovi polovi. Prema slici 2.9. može se vidjeti pojednostavljeni prikaz H-mosta, iako se on obično realizira koristeći tranzistore kao na slici 2.10.. Zatvorimo li na sklopu prikazanom na slici 2.9. sklopku 2 i 3 motor će se vrtiti u jednom smjeru, a otvorimo li sklopke 2 i 3 i zatvorimo 1 i 4 on se počinje vrtiti u drugom smjeru. Na taj način upravljat će se i smjerom okretanja motora koji upravljaju različitim kotačima na vozilu a time i njegovo kretanje.



Slika 2.9. spoj H-most [7]

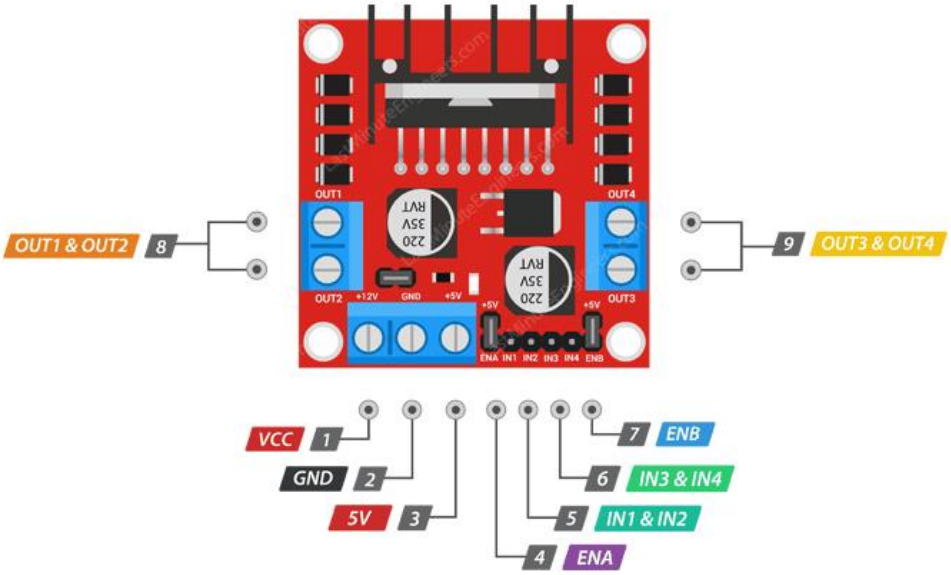


Slika 2.10. spoj H-most pomoću tranzistora [7]

2.4.2. L298N modul

Prema [9], Jedan od najjednostavnijih i najjeftinijih načina za upravljanje DC motorom je sučelje L298N Motor Driver s Arduinom. Može upravljati brzinom i smjerom okretanja dva DC motora. L298N modul koristi se za upravljanje dva DC motora, ali u ovom radu upravljat ćemo s četiri motora tako što ćemo dva motora priključiti kao jedan, što je pokazano u poglavlju gdje je opisano sastavljanje fizičke strukture vozila. Kako bi imali potpunu kontrolu nad kretanjem vozila u ovom projektu, moramo upravljati brzinom i smjerom motora. Brzinom motora upravljamo modulacijom širine pulsa koja je objašnjena u prvom potpoglavlju ovog poglavlja. Što je veći postotak perioda visoki napon, to će analogni napon biti veći i brzina okretaja motora veća. Smjerom okretanja motora upravljamo pomoću H-mosta čiji je princip rada objašnjen iznad. Prema slici 2.11. može se vidjeti izgled L298N modula. Najveći crni dio modula koji se nalazi na njegovom vrhu je L298N što je ustvari dvokanalni H-most po kojem je modul dobio ime. Ostatak modula je perfierija pomoću koje upravljamo njegovim radom i koja nam olakšava rad i nudi gotove mogućnosti kao što je na primjer napajanje. Napajanje modula izvedeno je s tri priključka na slici 2.11. označenima s 1 (VCC), 2 (GND), i 3 (5V). Također

se direktno iznad VCC i GND priključka nalazi premosnik. Svrha premosnika je uključivanje ili isključivanje 78M05 5V regulatora. VCC priključak može primiti od 5V do 35V i služi kao napajanje za motor. GND priključak je zajedničko uzemljenje, a 5V priključak služi kao napajanje za logičke sklopove unutar L298N modula. Ako nismo pomaknuli 5V-EN premosnik, onda je 78M05 regulator uključen i sve napajanje se obavlja kroz VCC priključak. Onda se 5V priključak koristi kao izlaz od 5V i 0.5A i može služiti za napajanje Arduina. Ako je 5V-EN premosnik pomaknut, regulator je isključen i trebamo na 5V priključak, koji sada ima ulogu ulaza, dovesti 5V napona tako što ga spojimo na 5V priključak na Arduino pločici. Ako je napajanje za motor veće od 12V treba maknuti premosnik s modula kako se regulator ne bi oštetio. U slučaju da je napajanje manje od 12V premosnik se može ostaviti na mjestu ali nikako ne treba koristiti 5V priključak kao napajanje. Kada koristimo VCC napajanje za motore treba uzeti u obzir unutarnji pad napona L298N modula koji iznosi 2V. Odnosno za napajanje motora koji maksimalnom brzinom rade na određenom naponu uvijek treba, želimo li da imaju mogućnost okretanja maksimalnom brzinom, dovesti napon 2V veći nego što je maksimalni napon motora. Izlazni priključci za motore na slici 2.11. označeni su brojevima 8 (OUT1 i OUT2) i 9 (OUT3 i OUT4). Na ta dva priključka mogu se spojiti DC motori koji rade na naponima od 5V do 35V, a maksimalna izlazna struja za motore može biti i do 2A. Postoje dvije vrste priključaka koje nam dopuštaju da upravljamo brzinom okretaja i smjerom okretanja motora. Ti priključci se na slici 2.11. mogu vidjeti desno od napajanja i označeni su kao 7 (ENB), 6 (IN3 i IN4), 5 (IN1 i IN2) i 4 (ENA). Pomoću priključaka ENA i ENB upravlja se brzinom motora. Ako su priključci na visokoj razini motor se vrti, a ako su na niskoj razini on staje, ali koristeći tehniku modulacije širine pulsa može se upravljati brzinom u određenom intervalu. Na ta dva priključka također se nalazi premosnik i pomoću njega se može odrediti hoće li se motor vrtjeti ili ne, odnosno, ako se premosnik nalazi na priključcima motori se vrte maksnimalnom brzinom, a želimo li upravljati brzinom onda premosnik možemo ukloniti. Koristeći priključke za promijenu smjera, a to su IN3, IN4 i IN1, IN2, možemo odrediti hoće li se motor vrtiti u jednom ili drugom smjeru. Oni u stvari upravljaju tranzistorima unutar H-most sklopa koji se nalazi u integriranom sklopu modula. Za svaki motor određena su dva priključka preko kojih će se upravljati njegovim smjerom. IN1 i IN2 upravljaju smjerom motora čije je napajanje označeno s OUT1 i OUT2, dok IN3 i IN4 priključci upravljaju smjerom motora čije je napajanje označeno s OUT3 i OUT4. Smjerom okretanja motora može se upravljati tako što se primjeni visoka razina ili niska razina na priključke. Prema tablici 2.1. se može vidjeti primjer kako se koriste priključci za upravljanje smjerom.



Slika 2.11. L298N modul [9]

Tablica 2.1. Način korištenja in1 i in2 priključaka [9]

In1	In2	smjer
0	0	Motor isključen
1	0	Naprijed
0	1	Nazad
1	1	Motor isključen

Motor će biti isključen kada na in1 i in2 odjednom imamo nisku razinu ili visoku razinu zato što nema nikakve razlike potencijala.

2.5. Arduino senzorski štit V5

Prema [10], Arduino senzorski štit je pločica koja se koristi za povezivanje senzora, servomotora, LCD-ova s arduino pločicom bez potrebe za lemljenjem. Postoje dvije verzije Arduino senzorskog štita V4 i V5, a u ovom projektu koristi se V5 verzija. Priključci na Arduino senzorskom štitu

raspoređeni su u dvije glavne grupe, kao i kod arduino pločice, u analogne i digitalne priključke. Za svaki digitalni arduino ulazno / izlazni priključak izvedeni su i posebni priključci za 5V i uzemljenje. Tako da koristeći ovaj štit možemo svakoj komponenti koja je priključena na jedan digitalni ulazno / izlazni priključak pridružiti i vlastito uzemljenje i visoku razinu od 5V. Kao i digitalni priključci, analogni priključci su također raspoređeni u trojke koje se sastoje od uzemljena, priključka za 5V i samog ulaznog analognog priključka preko kojeg možemo vršiti mjerenja. Prema tablici 2.2. može se vidjeti raspored digitalnih priključaka na štitu, a prema tablici 2.3. može se vidjeti raspored analognih priključaka.

Tablica 2.2. raspored digitalnih priključaka na Arduino štitu V5 [10]

G		Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd
V		Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc
S	Aref	Gnd	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tablica 2.3. raspored analognih priključaka na Arduino štitu V5 [10]

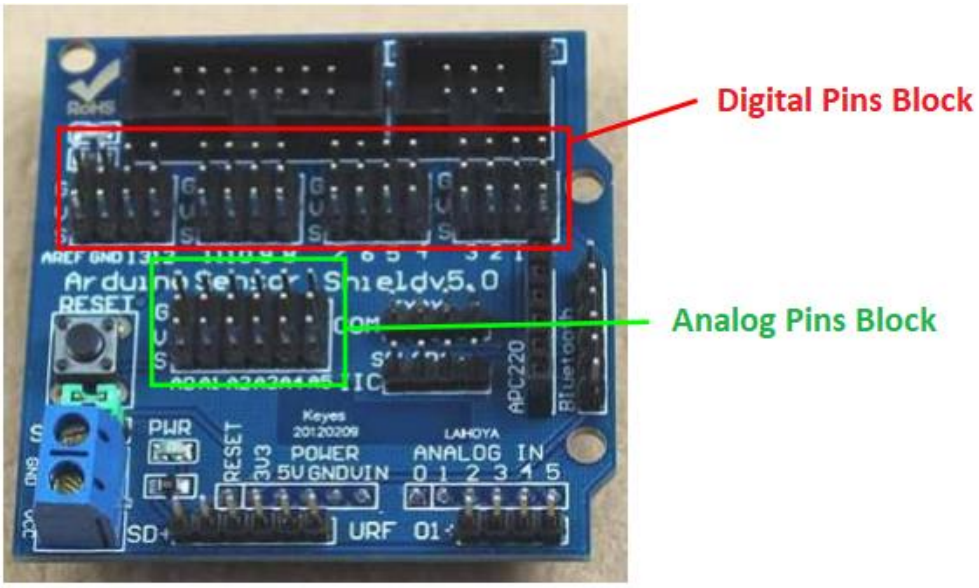
G	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd
V	Vcc	Vcc	Vcc	Vcc	Vcc	Vcc
S	A0	A1	A2	A3	A4	A5

Arduino štit V5 dolazi i u uz druga razna poboljšanja i izvode specijalizirane za određene senzore, a samo neka od tih poboljšanja su:

- Kompatibilnost s Arduino UNO i Mega pločicama
- Sučelje za RB URF v1.1 ultrazvučni senzor

- Sadrži sučelje za komunikaciju s modulom za SD karticu
- Sadrži sučelje za komunikaciju s Bluetooth modulom
- Sadrži sučelje za upravljanje servomotorom

Primjer Arduino senzorskog štita V5 zajedno s istaknutim analgonim i digitalnim priključcima može se vidjeti prema slici 2.12.

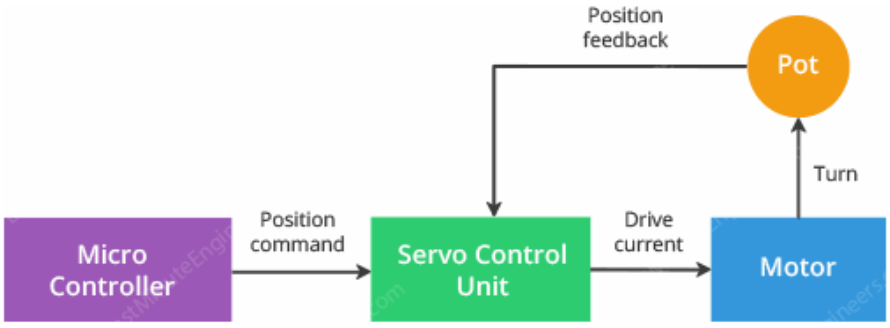


Slika 2.12. Arduino senzorski štit V5 s pripadajućim analognim i digitalnim priključcima [10]

2.6. Servomotor

U ovom radu servomotor se koristi za rotiranje ultrazvučnog modula koji će biti pričvršćen na servomotor. Na taj će način servomotor omogućiti pretragu okoline u cilju pronalaska puta bez prepreke. Prema [11, str. 80.], DC motori služe kao odlični pogonski motori, ali nisu idealni za precizni rad zato što ne dolazi do nikakve povratne veze. Drugim rječima bez korištenja nekakvog vanjskog enkodera nikad nećemo znati apsolutnu poziciju DC motora. Servomotori ili servoi, za razliku su jedinstveni po tome što im se naredi da se rotiraju na određeni kutni položaj i oni ostaju ondje dok im se ne kaže da se rotiraju na drugi položaj. Servo je ustvari sustav s povratnom vezom koji se koristi povratnim signalom kako bi podesio svoju brzinu i smjer u cilju dolaska do željenog položaja (slika 2.13.). Obično servomotor može rotirati svoju osovinu između 0 stupnjeva i 180

stupnjeva, a to je slučaj i za servomotor koji se koristi u ovom radu. Servomotor koji koristimo ima tri priključka. To su VCC ili 5V priključak, GND priključak i signal priključak. Ti priključci mogu se vidjeti na slici 2.14. i istim redom su označeni s 2, 1 i 3. VCC priključak treba se spojiti na 5V Arduino pločice dok se GND priključak treba spojiti na uzemljenje. Signal priključak povezuje se na jedan od izlazno / ulaznih digitalnih priključaka Arduino pločice, i kutom servomotora se preko tog priključka upravlja putem modulacije širine pulsa. Uglavnom taj signal treba biti frekvencije 50Hz. Ako je signal te frekvencije i puls je na visokoj razini 1ms onda će se servo nalaziti na kutu 0 stupnjeva. Ako je puls na visokoj razini 1.5ms, servo će se nalaziti na kutu od 90 stupnjeva, a za visoki puls od 2ms nalazit će se na 180 stupnjeva. Iako je to direktan način na koji se može upravljati položajem servomotora, u ovom projektu koriti se biblioteka koja će se pobrinuti za modulaciju širine pulsa.



Slika 2.13. Servo, sustav s povratnom vezom [9]

Slika 2.14. Servomotor i njegovi priključci [9]

2.7. Ostale komponente

U ovom poglavlju navedene su ostale komponente koje se koriste u radu i kratko je objašnjena njihova svrha.Te komponente su:

- 4 DC motora
- 2 Litij-ionske baterije
- Djelovi za sastavljanje plastičnog kućišta
- Žice s plastičnom izolacijom za povezivanje komponenti

4 DC motora koristit će se u kombinaciji s L298N modulom kao što je objašnjeno, a cijeli sustav napaja se iz Litij-ionskih baterija. Cijeli električni sustav bit će pričvršćen na plastično kučište koje će imati dvije platforme. Na gornjoj platfromi nalazit će se Arduino uno pločica sa senzorom i baterijama, dok će se na donjoj platformi nalaziti pogonske komponente kao što je motor i L298N modul.

3. PROGRAMSKI JEZIK KORIŠTEN U RADU (C)

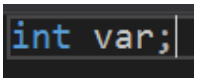
U ovom poglavlju bit će objašnjene neke osnove programskog jezika C koji se koristi u ovom radu za programirnaje makete vozila.

3.1. Opće osnove programskog jezika C

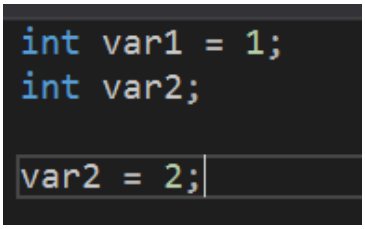
U ovom potpoglavlju bit će ukratko objašnjen samo osnovni dio programskog jezika C. Što znači da se ne ulazi u opisivnaje kompliciranijih pojmova koji nisu potrebni za razumijevanje ovog rada i specifičnu primjenu jezika u Arduino okruženju, već se pristupa objašnjenju jezika kao jezika opće namjene. Također vrlo jednostavni pojmovi, kao na primjer varijabla, neće biti objašnjeni, već će se samo opisati način na koji se koriste u programskom jeziku C.

3.1.1. Tipovi podataka, deklaracija i inicijalizacija varijabli

Prema, [12, str. 9.], U C-u, sve varijable moraju se deklarirati prije nego što se koriste, obično na početku funkcije prije bilo koje izvršne naredbe. Varijable u C-u kao i u ostalim programskim jezicima služe za lakši pristup podacima na određenoj memorijskoj adresi. Pomoću deklaracije varijable određujemo njena svojstva. Deklaracija varijable sastoji se od tipa podatka, koji govori kojeg tipa podatka će biti deklarirana varijabla, i od popisa imena varijabli. Može biti jedno ime ili ih možemo navesti više, odvajajući ih zarezom. Deklaracija varijable može se vidjeti prema slici 3.1. Želimo li varijablu ujedno i inicijalizirati, onda koristimo znak jednakosti koji u ovom kontekstu predstavlja znak pridurživanja vrijednosti. Također, varijabli ne mormao pridružiti vrijednost prilikom njezine deklaracije, već u idućim linijama koda programa možemo navesti njeno ime i znakom pridruživanja joj pridružiti vrijednost. Pridružimo li vrijednost na ovaj način varijabli koja već ima svoju zadanu vrijednost, onda se prošla vrijednost briše i varijabla poprima novopridruženu vrijednost. Inicijaliziranje varijable kao i naknadno pridruživanje vrijednosti može se vidjeti prema slici 3.2.



Slika 3.1. deklaracija varijable tipa int



Slika 3.2. inicijalizacija varijable i naknadno pridruživanje vrijednosti

Postoji nekoliko osnovnih tipova podataka u C programskom jeziku. Ti tipovi podataka su:

- char
- int
- float
- double

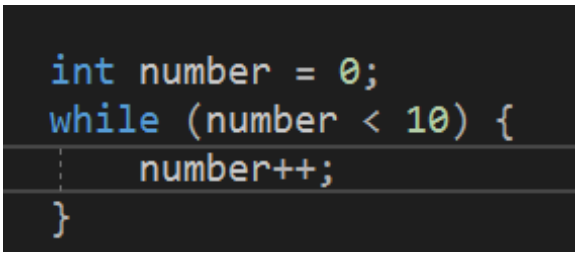
Char ima jedan bajt i ima mogućnost držanja jednog znaka. Int ili Integer predstavlja cijeli broj i njegova veličina ovisi koliko cijeli broj zauzima memorije na određenom računalu. Float i double predstavljaju decimalne brojeve, ali double je duplo precizniji od float tipa. Uz osnovne tipove postoje i određena proširenja. Na int tip mogu se primjeniti short i long proširenja. Gdje short predstavlja manju verziju int tipa podatka dok log predstavlja veću verziju int tipa podatka. Short je obično veličine 16 bita, a long 32 bita, iako svaki kompajler sam određuje veličinu koja je pogodna za sklopovlje koje računalo koristi. Proširenje signed ili unsigned može se koristiti na char tipu ili integer tipu. Unsigned varijable uvijek su pozitivne ili 0, odnosno nemaju predznak, dok signed varijable mogu imati i predznak, odnosno mogu imati negativne vrijednosti uz nulu i pozitivne vrijednosti. Također postoji long double tip koji proširuje decimalnu preciznost.

3.1.2. Petlje u C programskom jeziku

Prema, [13 str. 29.], Petlje su posebna vrsta upravljačkih naredbi. Upravljačka naredba odlučuje hoće li se kod pokrenuti, a petlja odlučuje koliko puta će se komad koda pokrenuti. Vidimo da je petlja vrlo koristan dio svakog programskog jezika zato što se često moraju obavljati zadatci koji se ponavljaju veliki broj puta. Budući da petlja odlučuje koliko puta će se neki određeni dio koda u programu pokrenuti, to znači da se također taj dio koda može pokrenuti nula puta, odnosno ne mora se uopće pokrenuti ovisno o uvjetu. U programskom jeziku C mogu se koristiti tri vrste petlji, a to su:

- while
- do while
- for

Prema, [13 str. 29.], „Najjednostavnija vrsta petlje u C-u je while petlja. While petlja pokreće kod iznova i iznova sve dok određeni uvjet ostaje istinit.” While petlja je petlja s provjerom uvjeta na ulasku što znaći da će se prvo provjeriti određeni uvjet, a onda ako je uvjet istinit, pokrenuti kod koji se nalazi unutar while petlje. Petlja do while funkcionira na isti način kao while petlja, ali ta petlja ima provjeru uvjeta na izlasku, što znači da će se prvo pokrenuti kod unutar petlje, a onda će se provjeriti je li određeni uvjet istinit. Iz tog se može zaključiti da će se kod unutar do while petlje uvijek pokrenuti bar jednom. For petlja se često može koristiti isto kao while petlja, ali razlika je što se while petlja koristi kada ne znamo koliki broj iteracija imamo, a u for petlji znamo koliki će nam broj iteracija biti potreban za neki zadatak. Također sve tri petlje imaju pomalo drugačije tijelo. U do while petlji i while petlji samo moramo postaviti uvijet čija će se istinitost provjeravati u svakoj iteraciji, a u for petlji moramo incijalizirati polaznu točku, postaviti uvjet i odrediti veličinu koraka u svakoj iteraciji petlje. Primjer while, do while i for petlje može se vidjeti na slici 3.3., 3.4. i 3.5., dok se njihovi dijagrami tokova mogu vidjeti na slikama 3.6., 3.7., 3.8.



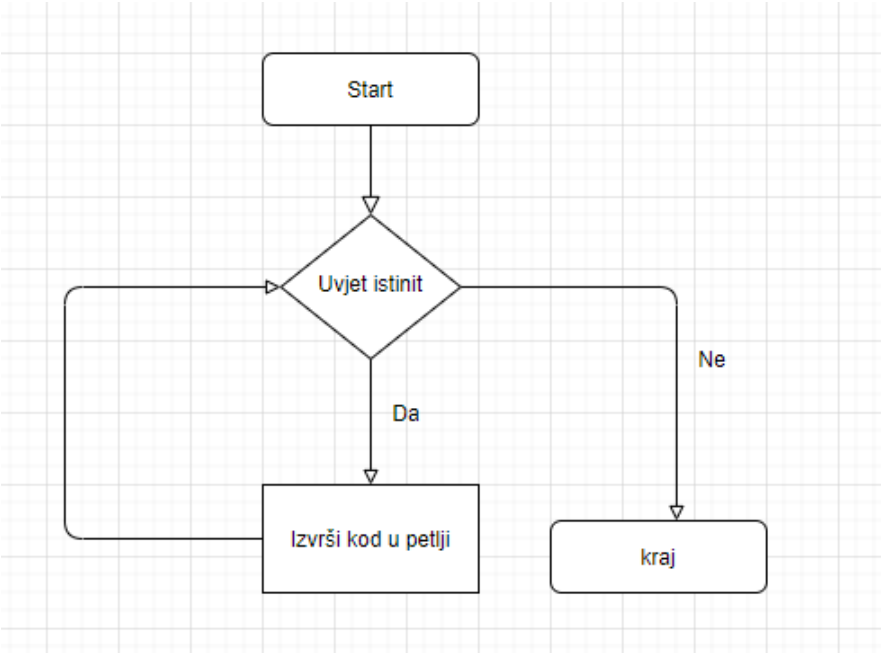
Slika 3.3. primjer while petlje

```
int number = 0;
do {
    number++;
}
while (number < 10);
```

Slika 3.4. primjer do while petlje

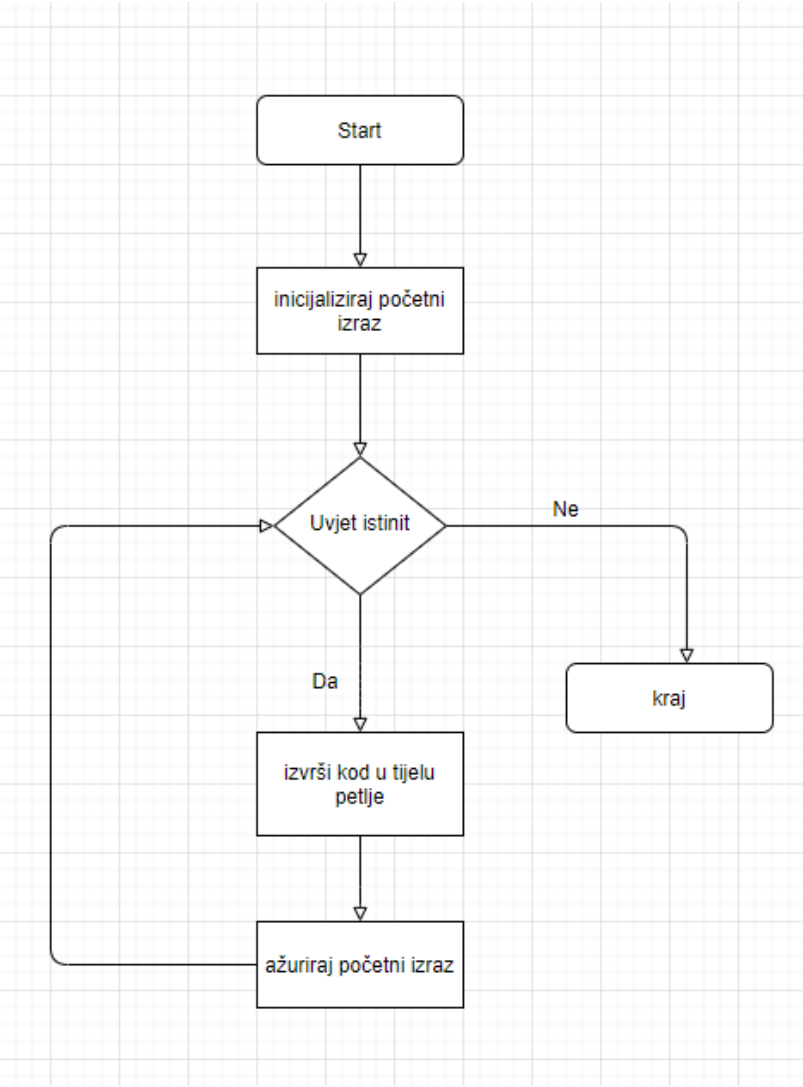
```
int number = 0;
for (int i = 0; i < 10; i++) {
    number++;
}
```

Slika 3.5. primjer for petlje



Slika 3.6. dijagram toka while petlje

Slika 3.7. dijagram toka do while petlje



Slika 3.8. dijagram toka for petlje

3.1.3. Naredbe grananja

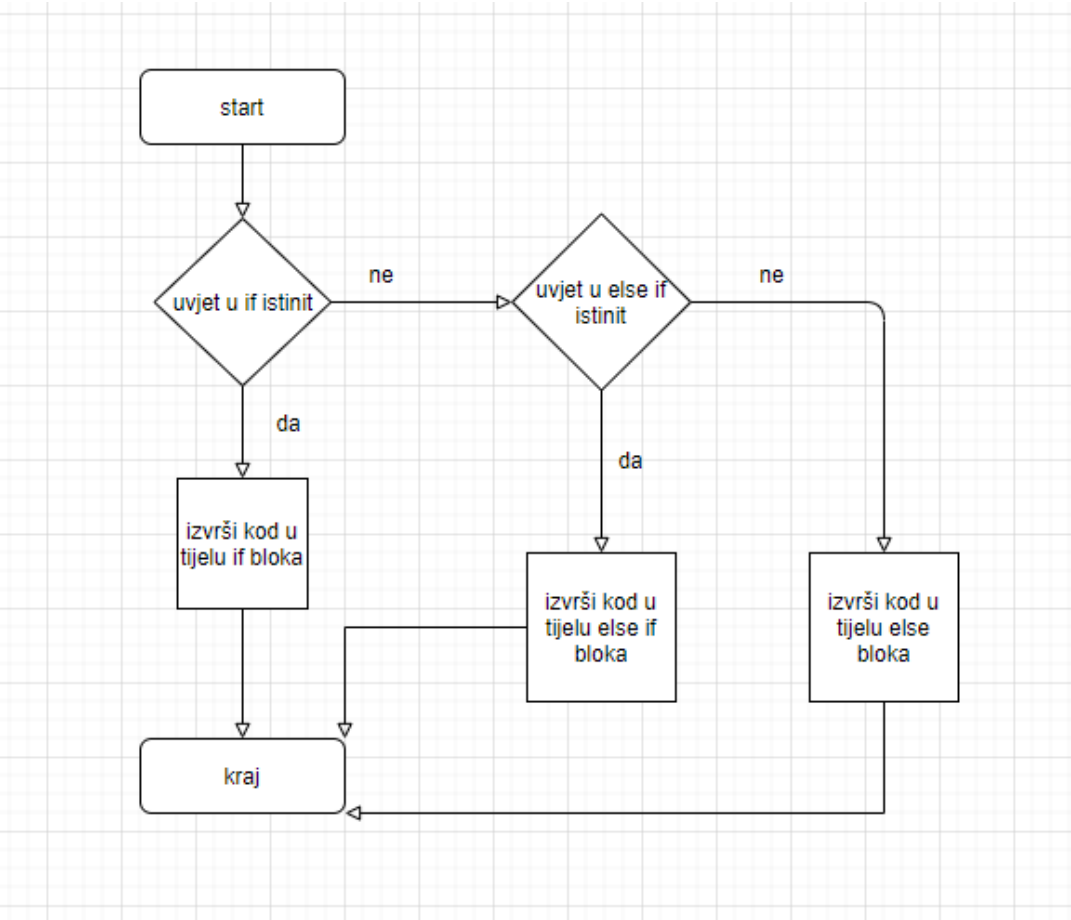
Naredbe grananja su način na koji našem programu možemo omogućiti donošenje različitih odluka pomoću istinitosti nekih tvrdnji. Naredbe grananja koje postoje u C programskom jeziku su:

- if
- else
- else if

If naredbu grananja koristimo kada određeni dio koda želimo izvršiti samo ako je neki zadani uvjet istinit. To može biti blok koda ili može biti samo jedna linija. Uz if mogu se vezati i druge dvije naredbe grananja ili još if izraza. Ako u lanac takvih izraza postavimo samo if izraze, provjerit će se svaki uvjet koji se nalazi unutar zagrada koje stoje uz if naredbu grananja. Ako je svaki od tih uvjeta istinit, izvršit će se svaki blok koda if izraza. Ako u takav „lanac" postavimo else if naredbu grananja i ako uvjet prije uz if izraz bude zadovoljen onda se ne provjerava uvjet uz else if izraz. Ako uvjet uz prijašnji if izraz ne bude zadovoljen, onda se tek provjerava uvjet uz else if izraz i ako je istinit izvršava se kod koji se nalazi uz njega. Treću naredbu grananja, else, postavljamo ako želimo omogućiti našem „lancu" sigurno izvršavanje bar nekog koda, odnosno, kod else bloka izvršit će se ako nijedan od prijašnjih uvjeta nije zadovoljen. Ako je neki od prijašnjih uvjeta zadovoljen, kod uz else izraz se neće izvršiti. Iz objašnjenoga se jasno može uočiti koliku važnost će ovakvo donošenje odluka imati u ovom radu, ali i koliku važnost ima u većini programa koji se ne bave potpuno trivijalnim problemima. Na slici 3.9. može se vidjeti primjer programa u kojem se koriste sve tri naredbe grananja, a prema slici 3.10. može se vidjeti dijagram toka if – else if – else.

```
int age = 16;

if (age < 14) {
    printf("premladi ste za vlak smrti");
}
else if (age >= 14 && age < 18) {
    printf("morate ići na dječju vožnju");
}
else {
    printf("uživajte u vožnji");
}
```

Slika 3.9. primjer korištenja naredbi grananja



Slika 3.10. dijagram toka if- else if – else

3.1.4. Funkcije

Prema, [14 str. 114.], Sve instrukcije u C programu nalaze se unutar funkcije. Svaka funkcija obavlja određeni zadatak. Posebno ime funkcije je main(): funkcija s ovim imenom je prva koja se pokreće kada program započne. Svrha funkcije je odvajanje poslova. Odnosno mogu se odvojiti dijelovi koda koji obavljaju određene poslove i onda bez ponovnog pisanja cijelog koda samo se pozvati na taj kod kada je potreban. Funkcije tako čine cijeli kod programa puno urednijim, ali čine i daljnje programiranje puno lakšim i smislenijim. Deklaracija funkcije ili prototip funkcije u C jeziku sastoji se od povratnog tipa funkcije koji određuje što će funkcija vratiti, to može biti i tip void koji nam govori da funckija ne vraća ništa. Iduće što se nalazi u potpisu funkcije je njeno ime koje u C jeziku mora biti jedinstveno. Uz ime nalaze se zagrade u kojima se nalaze parametri, što su ustvari lokalne

varijable funkcije, u koje se spremaju predani argumenti pri pozivu funkcije. Može ih biti više, jedan ili ih ni ne mora biti. U definiciji funkcije piše se kod u tijelu koji će izvršavati željeni posao. Sve varijable koje se deklariraju unutar funkcije su implicitno lokalne i automatske varijable, iako se eksplicitno može naglasiti drugačije. Deklarira li se varijabla izvan funkcije ona će biti globalna i statička varijabla. Pojam lokalna varijabla predstavlja varijablu kojoj se može pristupiti samo iz radnog opsega funkcije u kojoj je deklarirana, što predstavlja tijelo funkcije, dok se globalnim varijablama može pristupiti iz bilo koje funkcije. Ako je varijabla automatska znači da se memorija za nju zauzima prilikom pokretanja funkcije u kojoj je deklarirana, a u trenutku završetka izvršavanja funkcije ta memorija se oslobađa. Statičke varijable postoje od početka izvršavanja programa do kraja njegovog izvršavanja neovisno o nekoj funkciji. Kada se u programskom jeziku C pišu funkcije mora se funkciju ili deklarirati, to jest napisati njen prototip prije poziva funkcije, a zatim se može pisati definicija funkcije nakon poziva, ili se mora odmah definirati prije prvog poziva kako bi poziv funkcije bio valjan. Na slici 3.10. može se vidjeti deklaracija funkcije, poziv funkcije i definicija funkcije.

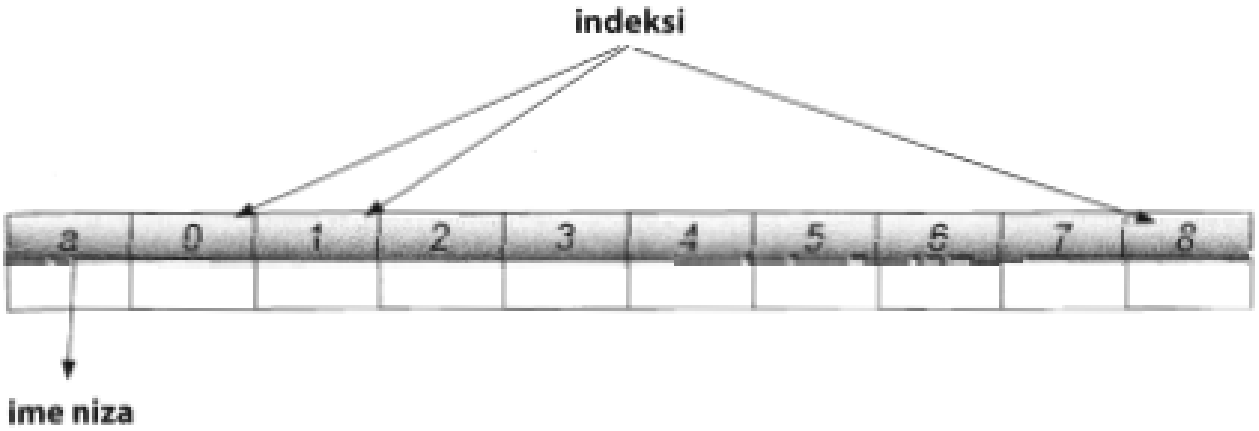
```c
int zbroji(int brojPrvi, int brojDrugi); // prototip funkcije

int main() {
    int zbroj = zbroji(2, 3); // poziv funkcije
    return 0;
}

int zbroji(int brojPrvi, int brojDrugi) { // definicija funkcije
    int zbroj;
    zbroj = brojPrvi + brojDrugi;
    return zbroj;
}
```

Slika 3.10. prototip, poziv i definicija funkcije

Funkcija zbroji sa slike 3.10. vrlo je jednostavna i vratit će zbroj dva broja koji su predani pri pozivu funkcije. Taj zbroj također će vratiti na mjesto poziva, i rezultat će biti spremljen u varijablu zbroj. Obje varijable zbroj su različite varijable iako imaju isto ime i može im se pristupiti samo iz radnih opsega funkcija u kojima su deklarirane. Što znači da su te varijable lokalne varijable.

3.1.5. Jednodimenzionalna i dvodimenzionalna polja

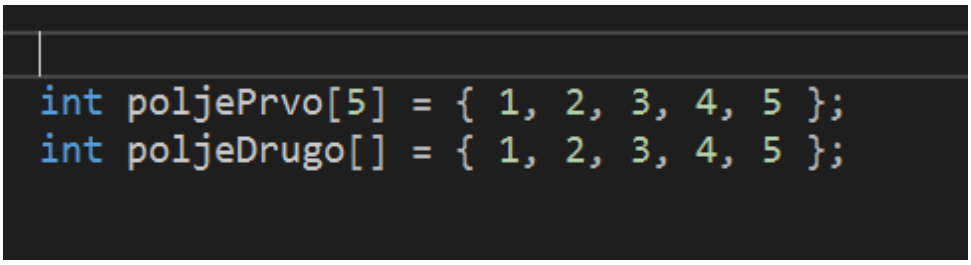
Prema, [15 str. 114.], „Preuzeti su iz matematike, jer omogućavaju pristup većem broju podataka uporabom jednog simboličnog imena. Definiraju se kao ograničen, uređen skup elemenata istog tipa. Od jedne do druge vrijednosti u nizu kreće se pomoću indeksa. Indeksi su nenegativne cjelobrojne vrijednosti, a određuju mjesto u nizu na kojem se nalazimo.“ Prvi indeks koji onzačava početak polja je nula, a ne jedan. Razlogu tomu je način na koji se dolazi do adresa elemenata unutar polja. Prema slici 3.11. može se vidjeti prikaz jednodimenzionalnog polja i indeksa koji pripadaju svakom elementu polja.



Slika 3.11. prikaz jednodimenzionalnog polja [15]

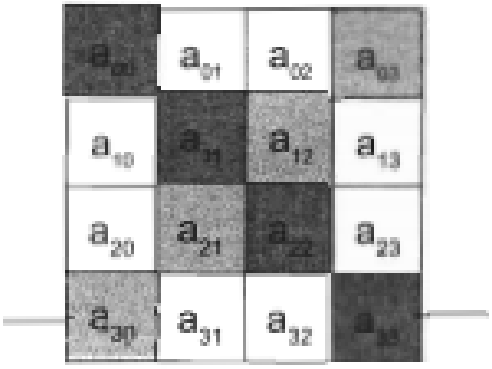
Naziv polja je memorijska adresa prvog elementa polja. Na slici 3.12. mogu se vidjeti dva načina deklaracije jednodimenzionalnog polja i njegovo inicijaliziranje korištenjem inicijalizatora ,koji se sastoji od vitičastih zagrada, kada je navedena veličina polja i kada nije navedena veličina polja unutar

uglatih zagrada. Navede li se pri deklaraciji veličina polja, kao na slici 3.12., i u inicijalizatoru se napiše manje elemenata nego što je određeno za polje prilikom deklaracije. Tada se ostatak mjesta unutar polja popunjava nulama. Pri deklaraciji polja ne mora se navesti njegova veličina, ali to vrijedi samo za slučaj kada se u istom trenutku i inicijalizira polje. Tada će prevoditelj prepoznati koliko elemenata se nalazi unutar inicijalizatora i to će biti veličina polja.



Slika 3.12. deklaracije jednodimenzionalnih polja

Oba polja sa slike 3.12. sastoje se od 5 elemenata, ali u drugom polju nije navedena veličinu nego ju je prevoditelj sam odredio iz broja elemenata unutar vitičastih zagrada. Dvodimenzionalni niz je baš kao i jednodimenzionalni niz uređen i ograničen skup podataka koji moraju biti istog tipa. Razlika je ta što se u dvodimenzionalnom nizu do elemenata dolazi pomoću dva indeksa. Prvi indeks predstavlja oznaku reda unutar dvodimenzionalnog polja, ako se zamisli kao matrica ili tablica, a drugi indeks onda predstavlja oznaku stupca unutar te matrice. Na slici 3.13. može se vidjeti prikaz dvodimenzionalnog polja u obliku matrice i indeksi svakog elementa, a na slici 3.14. može se vidjeti primjer deklaracije dvodimenzionalnog polja i njegova inicijalizacija. Dvodimenzionalno polje deklarira se na sličan način kao jednodimenzionalno polje, ali koriste se dvije uglate zagrade za deklaraciju. U prvu uglatu zagradu upisuje se broj redova koje će sadržavati dvodimenzionalno polje, a u drugu uglatu zagradu upisuje se broj stupaca koje će sadržavati dvodimenzionalno polje. Pri pridruživanju vrijednosti dvodimenzionalnom polju, koriste se vitičaste zagrade kao i kod jednodimenzionalnog polja, ali svakom redu posebno se pridružuju vrijednosti tako što se piše vitičasta zagrada unutar glavnih vitičastih zagrada posebno za svaki red. Ako se jednom redu do kraja ne pridruže vrijednosti, kao i kod jednodimenzionalnog polja, ostatak elemenata u redu se popunjava nulama.



Slika 3.12. prikaz dvodimenzionalnog polja [15]

```
int dvodimenzionalnoPolje[2][3] = { {1, 2, 4}, {3, 6} };
```

Slika 3.13. deklaracija i inicijalizacija dvodimenzionalnog polja

Na slici 3.13. treći element drugog reda unutar dvodimenzionalnog polja bit će nula zato što se nisu do kraja naveli svi elementi reda.

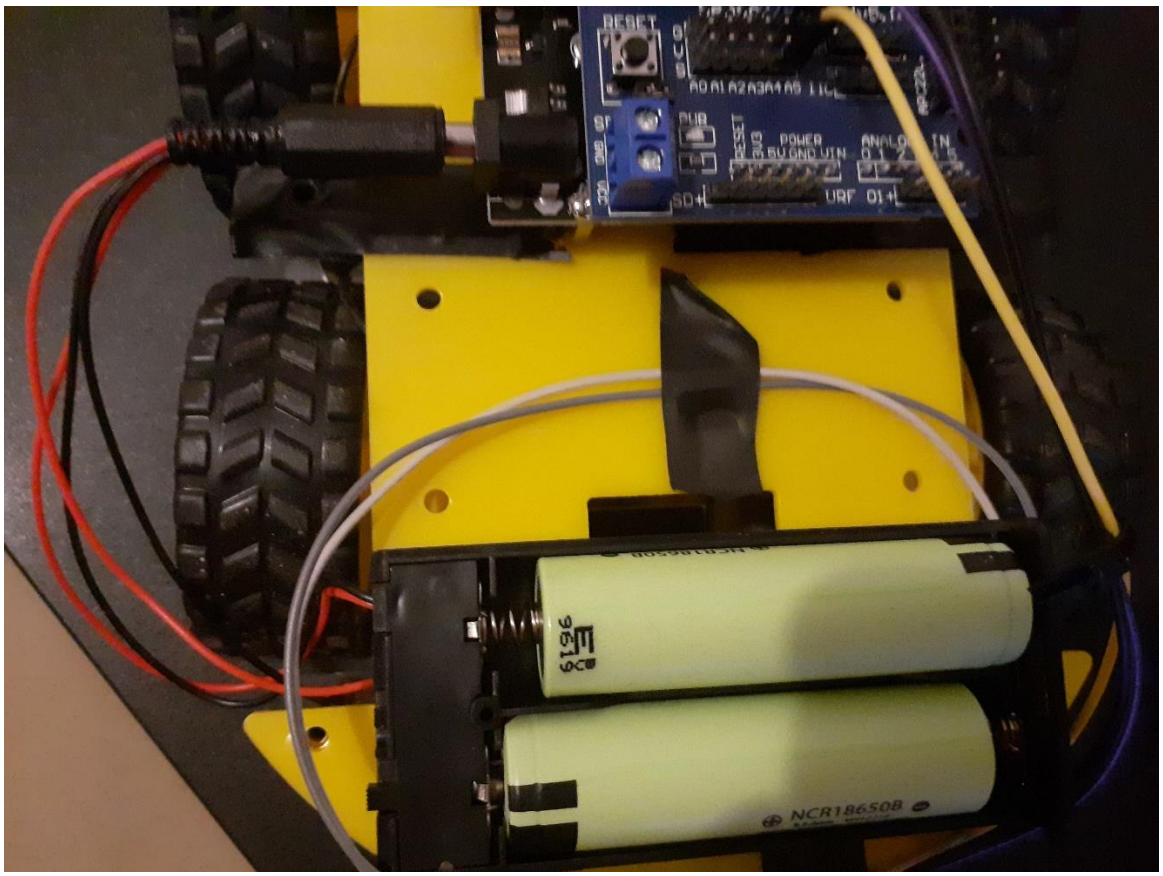
4. IZRADA ELEKTRIČNE SHEME I SKLOPOVLJA VOZILA

U ovom poglavlju govori se o spajanju električne sheme vozila, koja će se nalaziti na dvije platforme vozila. U fizičkoj maketi koristi se LAFVIN razvojna pločica umjesto Arduino razvojne pločice. Između te dvije pločice ne postoje razlike i LAFVIN je napravljen potpuno po uzoru na Arduino UNO. Programira se i koristi na isti način. Svi priključci su isti i imaju iste funkcionalnosti. Električna shema može se razdvojiti u dva glavna dijela. Prvi podsustav u sklopu cijele električne sheme je sustav za pokretanje i napajanje vozila u koji ulaze motori, L298N modul i baterije koje se koriste za napajanje cijelog vozila. U drugi podsustav, senzorski sustav, ulaze ultrazvučni senzor, termistor i servomotor koji će okretati ultrazvučni senzor. Ova podijela na podsustave napravljena je zbog preglednosti električnih shema, ali na fizičkoj maketi vozila sve se nalazi spojeno na jednu LAFVIN razvojnu pločicu s koje se upravlja cijelim sustavom. Pri spajanju električnih komponenti prvo se kreće od izrade sustava za pokretanje, a zatim se spaja i senzorski sustav

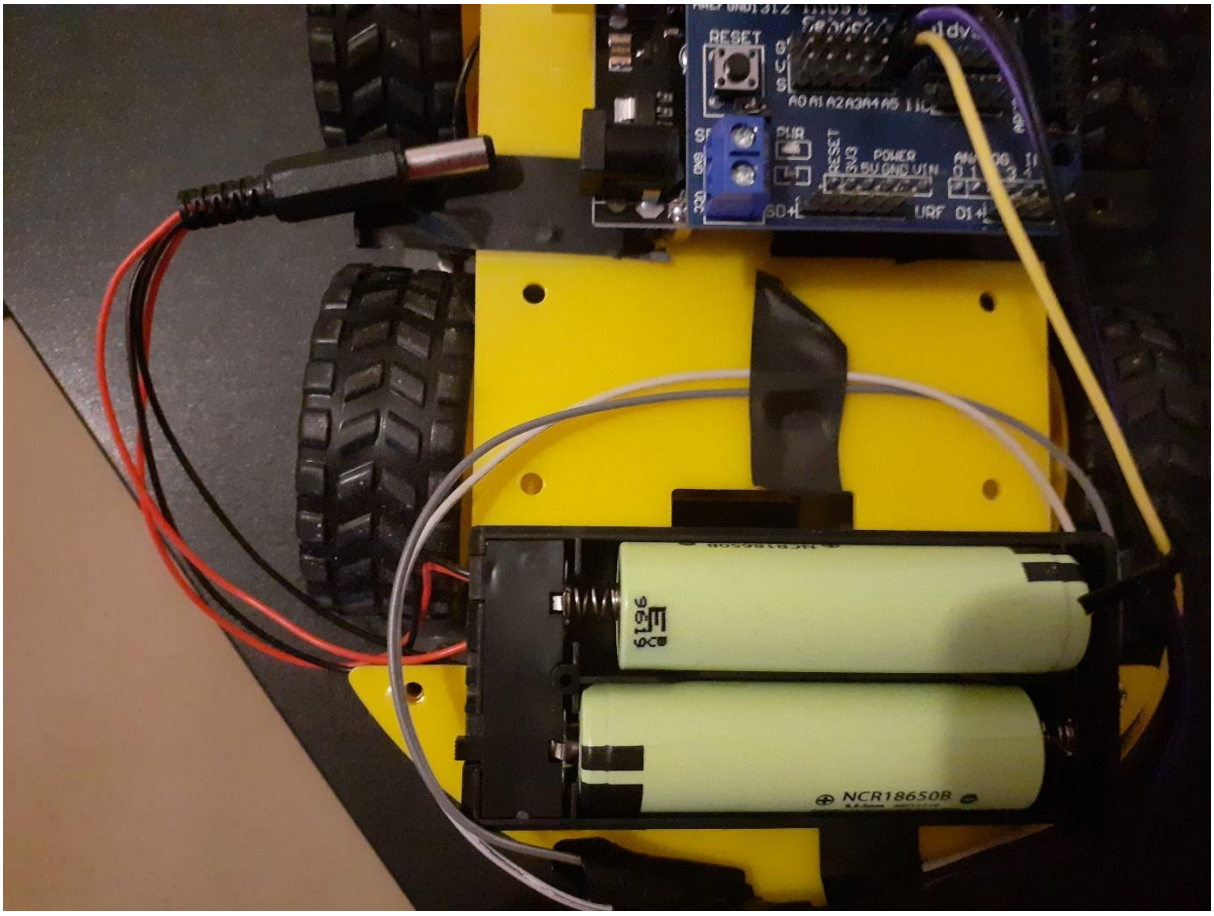
4.1. Izrada sustava za pokretanje i napajanje

Pri izradi sustava za pokretanje koriste se opisane komponente. Četiri motora spajaju se na priključke za motore koji se nalaze s lijeve i desne strane L298N modual. Svaki motor ima svoju negativnu i pozitivnu stranu. Kako bi se u isto vrijeme upravljalo s oba kotača s jedne strane, spajaju se negativna strana i pozitivna strana svakog motora zajedno. Odnosno, negativne strane motora spajaju se na OUT1 i OUT4. To su izlazi čijim se smjerom upravlja pomoću IN1 i IN4 priključaka L298N modula. Pozitivne strane motora spajaju se na OUT2 i OUT3 priključke kojima se upravlja pomoću IN2 i IN3 ulaznih priključaka. Iz ovog se može vidjeti zašto će se oba motora okretati u istim smjerovima. Žice se na opisana mjesta stavljaju i učvršćuju zatezanjem vijaka koji se nalaze na izlaznim priključcima OUT1, OUT2, OUT3 i OUT4 modula. Za napajanje L298N modula i ostatka sustava koriste se NCR18659B baterije. Pozitivni pol baterija spaja se na ulazni priključak modula označen s 12V, a negativni pol baterija spaja se na GND priključak modula. Premosnik nije potrebno micati budući da svaka od baterija ima 3.7 V, odnosno ukupno će dvije korištene baterije imati 7.4 V što ne prelazi granicu od 12 V na kojoj se uklanja premosnik. Žice se na mjestu učvršćuju na isti način kao i žice

motora na izlaznim priključcima, zatezanjem vijaka. Na 5V priključak ne spaja se ništa. Ulazni priključci pomoću kojih se upravlja motorima spajaju se na digitalne priključke LAFVIn razvojne pločice, odnosno Arduino senzorskog štita V5. Digitalni D2 priključak arduina povezuje se na IN4 priključak L298N modula. D4 povezuje se na IN3, D5 na IN2, a D2 digitalni priključak povezuje se na IN1 priključak. Za upravljanje brzinom motora koriste se ENA i ENB priključci na L298N. Modulacijom širine pulsa u ovom projektu brzinom motora upravlja se pomoću D3 i D6 priključaka na LAFVIN pločici. D3 priključak povezan je na ENB stranu, dok je D6 priključak povezan na ENA stranu. Preko ENB upravlja se brzinom motora na lijevoj strani vozila, a preko ENA upravlja se brzinom na desnoj strani vozila. Napajanje LAFVIN razvojne pločice može se ostvariti na dva načina. Prvi način se izvodi spajanjem negativnog pola baterija na GND pločice i pozitivnog pola na VIN pločice. Drugi način koji se koristi i u ovom radu je spajanje baterija kroz kružni priključak za napajanje na Lafvin pločici. Ovaj načina napajanja prikazan je na slikama 4.1. i 4.2. Na prvoj slici vidi se uključeno a na drugoj isključeno napajanje. Oba načina su ista i ne dolazi do nikakvih značajnih razlika pri njihovoj izmjeni.

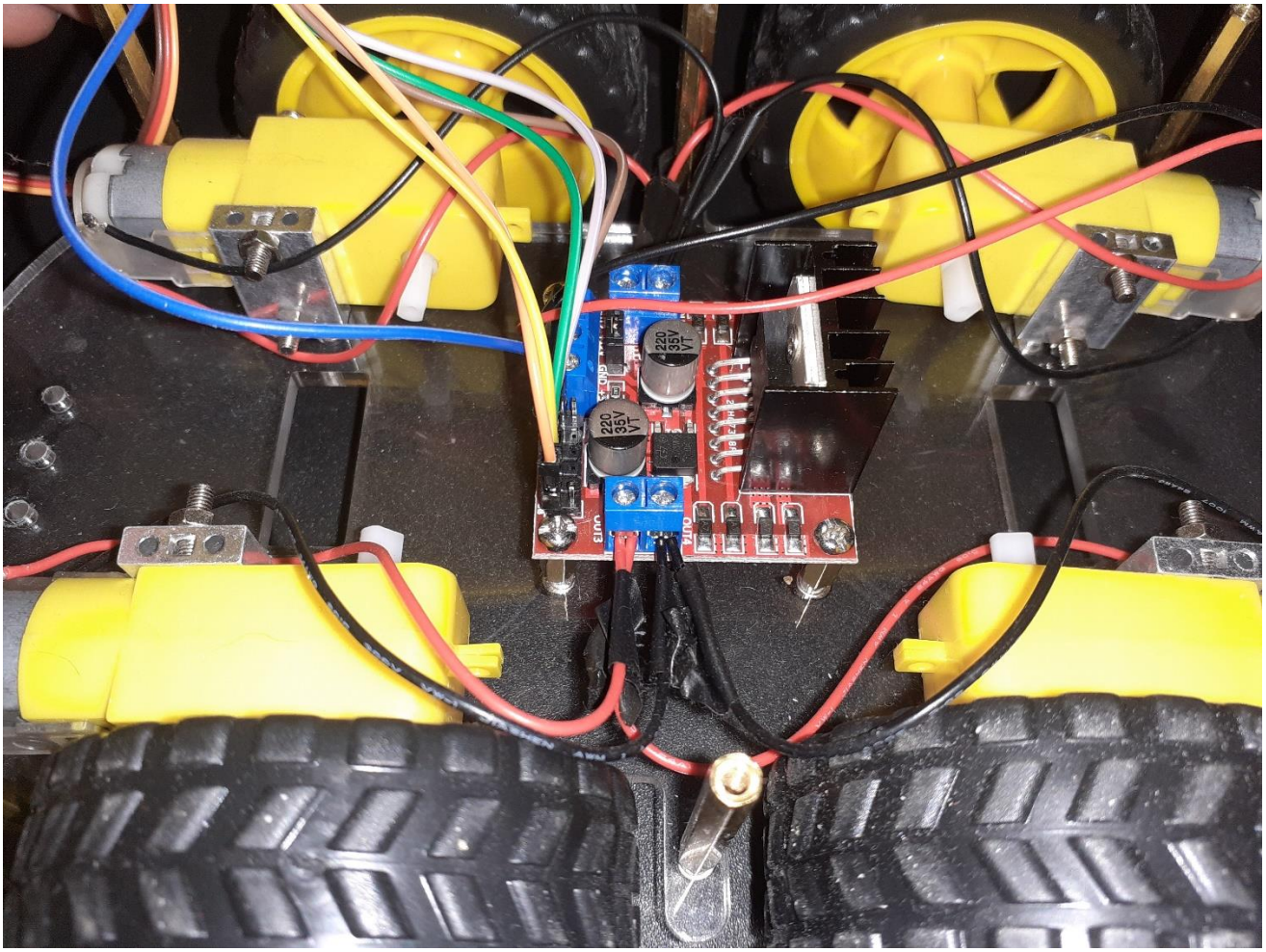


Slika 4.1. uključeno napajanje LAFVIN pločice



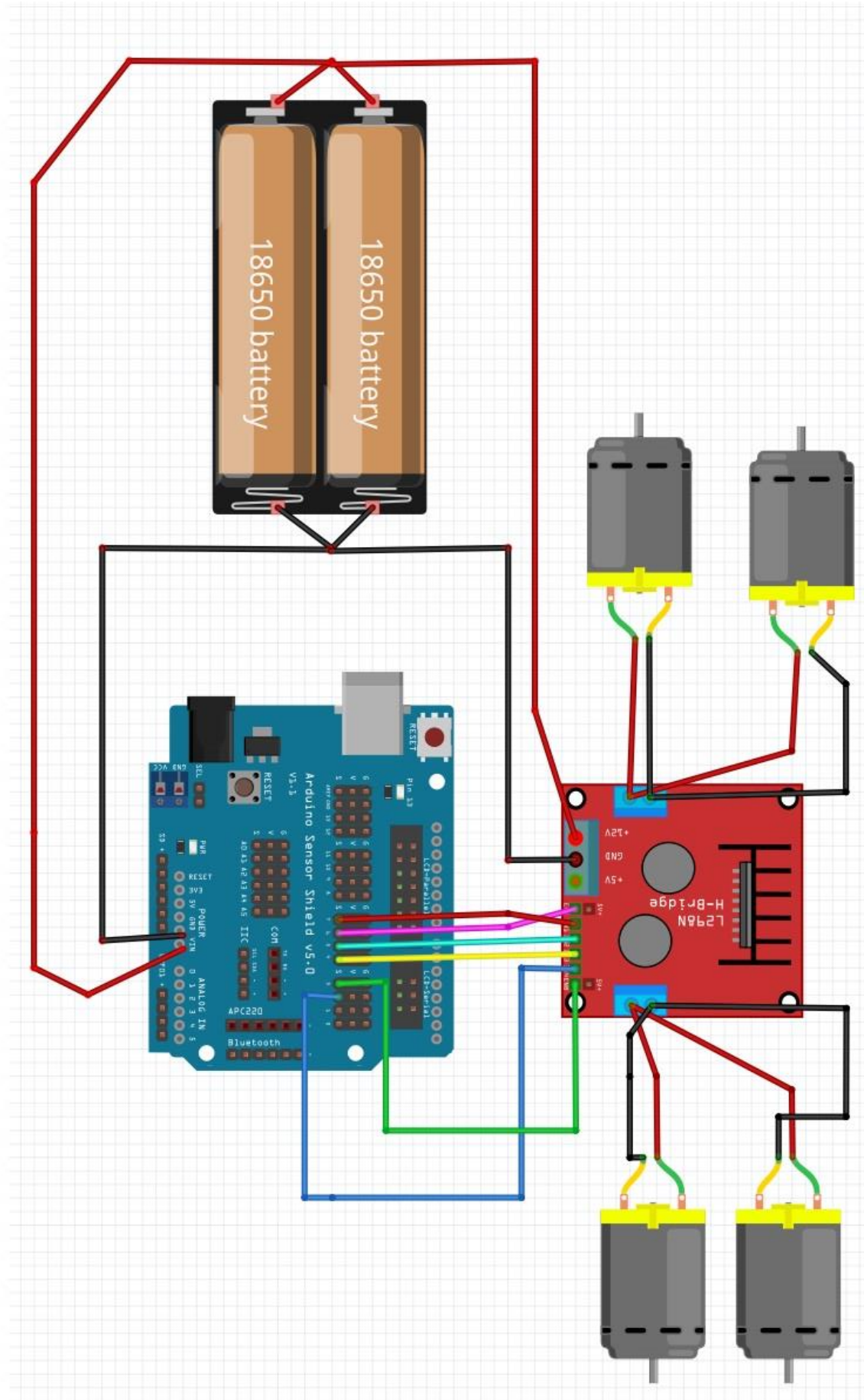
Slika 4.2. isključeno napajanje LAFVIN pločice

Preporučeni napon za napajanje pločice je 7 do 12 V, baterije korištene u ovom radu pružaju napon od 7.4 V što je dovoljno za napajanje. Granični napon može ići i minimalno do 6 V, a maksimalno do 20 V, ali takvo korištenje nije preporučeno. Na slici 4.3. može se vidjeti sustav za pokretanje na fizičkoj maketi vozila, a na slici 4.4. cijela električna shema koja je opisana u ovom potpoglavlju. Slika sheme izrađena je u programu FRITZING. Način napajanja na shemi drugačije je prikazan zbog ograničenja alata, ali u fizičkoj maketi vozila to ne mijenja ništa osim izgleda. Ovakav način napajanja korišten je zbog jednostavnosti i preglednosti. Također je puno lakše popraviti kvar, ako se napajanje odvoji od drugih priključaka za upravljanje i ako se za njega koristi samo jedan ulaz.



Slika 4.3. sustav za pokretanje

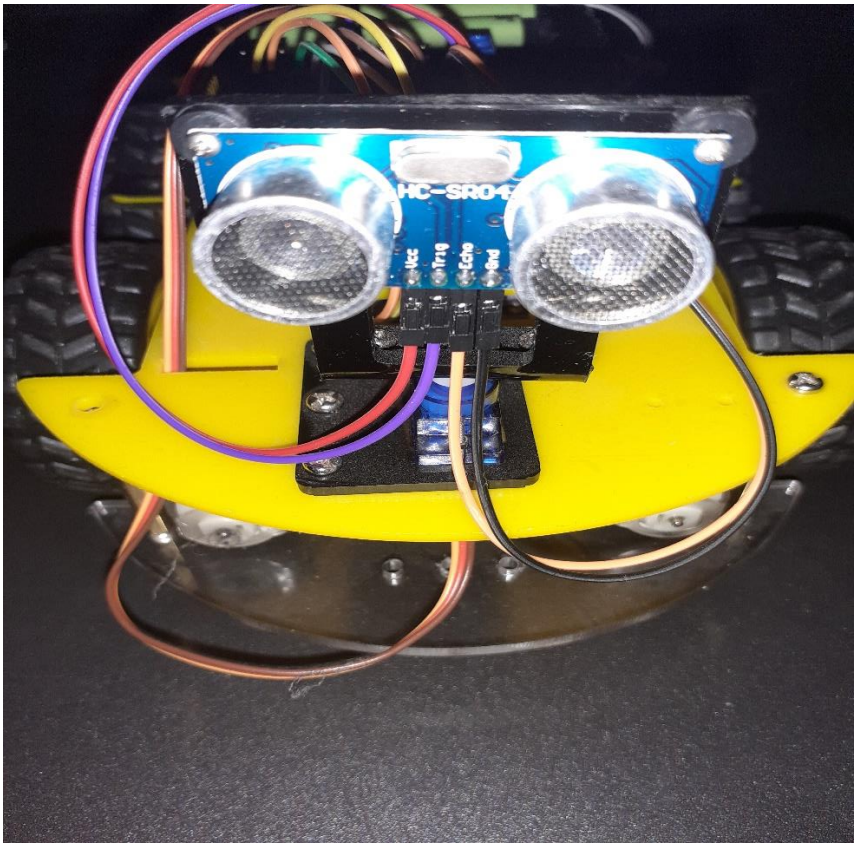
Kao što je opisano na početku potpoglavlja, na slici 4.3. jasno se vide negativne strane koje predstavljaju crne žice motora i pozitivne strane koje predstavljaju crvene žice motora. Te žice ulaze u plave priključke koji su ustvari izlazi iz modula L298N. Na motorima se mogu vidjeti kotači koji su jednostavno spojeni. Sam modul je učvršćen vijcima u podnožje vozila, baš kao i motori. Najveći problem pri ovakvom rasporedu motora i žica predstavlja zapinjanje žica za motor i kotač i njihovo čupanje iz izlaznih priključaka za upravljanje brzinom.


18650 battery
18650 battery
Arduino Sensor Shield v5.0
V1.1
RESET
PWR
POWER
ANALOG IN
IIC
APC220
Bluetooth
L298N
H-Bridge
+12V
GND
+5V

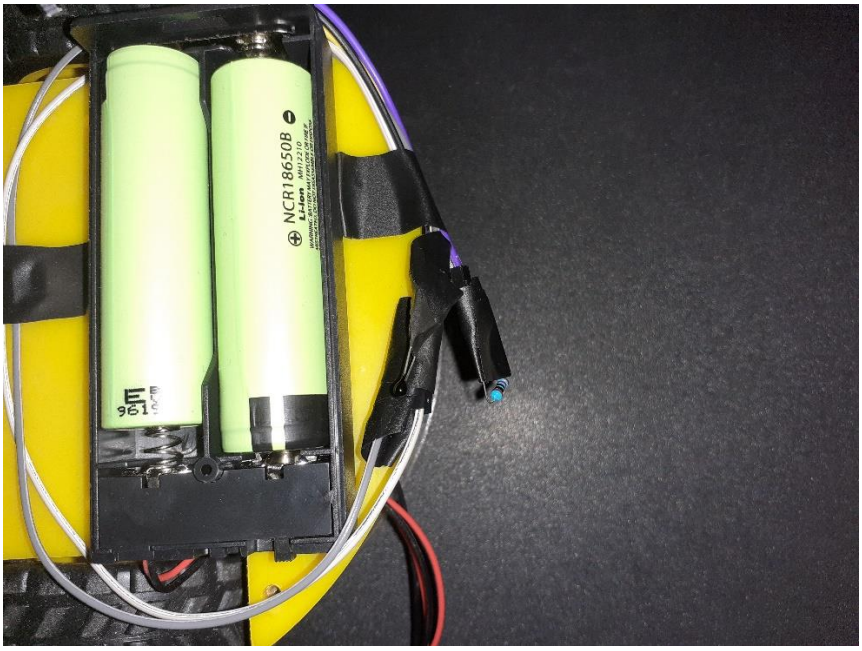
Slika 4.4. Električna shema potsustava za napajanje i pokretanje

4.2. Izrada senzorskog sustava

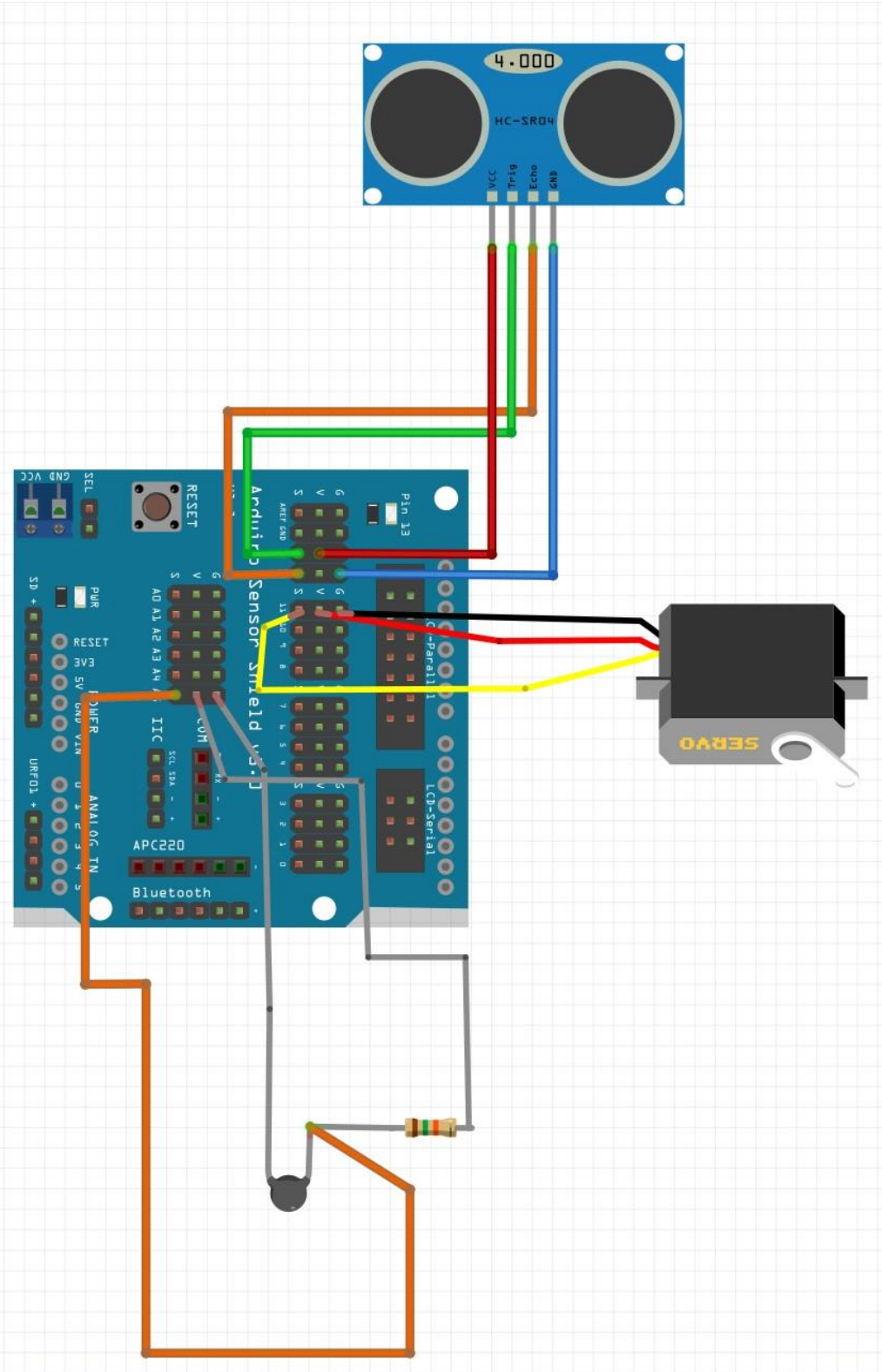
Pri izradi senzorskog sustava, koriste se komponente: ultrazvučni senzor HC-SR04, servomotor, termistor i obični otpornik od 10 kΩ. HC-SR04 ultrazvučni senzor, postavlja se na plastični držač koji se vijicima učvršćuje za servomotor a senzor se učvršćuje za držač. Tako se lagano može upravljati usmjerenjem ultrazvučnog senzora pomoću servomotora. Signalni priključak servomotora u ovom radu priključen je na digitalni 11 priključak Arduino senzorskog štita V5. Pomoću njega se upravlja smjerom okretanja od 0 do 180 stupnjeva. GND i VCC priključci su povezani s VCC i GND priključcima na Arduino senzorskom štitu pod oznakom 11. Echo priključak, na kojem mjerimo trajanje visokog stanja, od HC-SR04 modula, spaja se na digitalni 12 priključak na senzorskom štitu, također se pod oznakom 12 spaja i GND s HC-SR04. Trig priključak, pomoću kojeg se aktivira slanje ultrazvučnog vala i aktivira visoko stanje na Echo priključku, povezuje se na digitalni 13 priključak štita, pod čijom oznakom se također povezuje VCC s modula HC-SR04. Pri spajanju termistora i otpornika, termistor se spaja na GND od senzorskog štita odnosno LAFVIN pločice, a otpornik se spaja na VCC pločice. Između njih se izvodi žica pomoću koje se mjeri napon koji se u trenutku mjerenja nalazi na termistoru. Ta žica se spaja na analogni A5 priključak koji je uvijek samo ulazni priključak i kroz njega očitavamo ADC vrijednost. Ultrazvučni senzor zajedno sa servomotorom postavljen je kroz utor na plastičnom kućištu vozila. Postavljen je na prednju stranu vozila, na gornju platformu, budući da se konstantno mora očitavati vrijednost koja predstavlja udaljenost od prepreke ispred vozila. Termistor i otpornik na kućište su pričvrščeni ljepljivom trakom i postavljeni su na stražnji dio vozila zato što njihov položaj ne predstavlja bitnu stavku u izradi makete vozila. Pomoću ovako spojenog sustava lako se može izračunati pad napona na termistoru iz ADC mjerene vrijednosti. Iz napona se može doći do struje iz koje se na kraju može dobiti trenutni otpor termistora. Pomoću njega se lagano dolazi do temperature okoliša koja se u ovom radu koristi za izračun brzine zvuka. Opisani ultrazvučni senzor i servo mogu se vidjeti u sklopu makete vozila na slici 4.5., a termistor na slici 4.6. Cijela električna shema ovog potsustava vidi se na slici 4.7.



Slika 4.5. ultrazvučni senzor na maketi vozila



Slika 4.6. termistor na maketi vozila



Slika 4.7. Električna shema senzorskog potsustava

Na električnoj shemi senzorskog sustava, koji je prikazan slikom 4.7., ultrazvučni senzor nije postavljen na servomotor, ali na fizičkoj maketi vozila modul HC-SR04 i servomotor su spojeni.

5. PROGRAMIRANJE MAKETE VOZILA

U ovom poglavlju opisan je algoritam po kojemu radi vozilo i također je opisan i prikazan kod programa. Kao što je već napisano u poglavlju 3, maketa vozila programira se u programskom jeziku C. Prvo su napisane funkcije za kretanje, zatim funkcije za mjerenje temperature i izračunavanje brzine zvuka, a onda su napisane funkcije za mjerenje udaljenosti i pronalazak novog smjer zbog nailaska na prepreku. Algoritam po kojem funkcionira program vrlo je jednostavan. Vozilo će se kretati naprijed i pri nailasku na prepreku zaustavit će se i odmaknuti unazad za kratku udaljenost. Nakon toga vozilo će okrenuti ultrazvučni senzor koji je pričvršćen na servomotor. Prvo desno i ispustiti ultrazvučni val, zatim ispred sebe kako bi se provjerilo je li došlo do pomicanja prepreke i na kraju se okreće lijevo i ispušta se ultrazvučni val. Pri svakom ispuštanju ultrazvučnog vala, mjeri se udaljenosti i na kraju se uzima najveća udaljenost od svih izmjerenih i ona koja je veća od 15 cm, što predstavlja minimalnu udaljenost od prepreke. Vozilo se okreće u smjeru najveće udaljenosti i nastavlja se kretati u tom smjeru do iduće prepreke.

5.1. Postavljanje početnih varijabli i konstanti

U početku programa deklarirane su i inicijalizirane varijable koje će se koristiti kroz ostatak programa također je na engleskom jeziku pojašnjena svrha određenih skupina varijabli. Prvo se uključuje biblioteka Servo.h, pomoću koje se olakšano upravlja servomotorom za pomicanje ultrazvučnog senzora. Deklarira se objekt klase Servo pomoću čijih će se metoda to upravljanje postići. Ovo je jedini element objektno orijentiranog jezika C++ u kodu ovog programa. Također je potrebno

definirati konstantu koja pretstavlja maksimalnu udaljenost na koju može putovati ultrazvučni val. Ta konstanta se koristi u funkciji za izračun maksimalnog vremena čekanja na povratak ultrazvučnog vala. Nakon toga deklarirana je i inicijalizirana varijabla isMoving koja poprima samo vrijednost nula ili jedan. Iz nje algoritam zaključuje je li vozilo trenutno u stanju gibanja ili ga treba zaustaviti. Skupina varijabli pomoću kojih se upravlja smjerom okretaja motora sadrži in1, in2, in3 i in4 varijable. Te varijable koriste se u funkcijama koje upravljaju kretanjem vozila i dane su im vrijednosti sedam, pet, četiri i dva koje predstavljaju digitalne priključke na senzorskom štitu. Dvije varijable koje se koriste za upravljanje brzinom okretaja motora su ena i enb varijable koje se koriste za modulaciju širine pulsa. Vrijednosti koje sadrže te varijable su šest i tri i one predstavljaju priključke na senzorskom štitu. Šest i tri su vrijednosti tih varijabli iz razloga što samo neki priključci imaju mogućnost modulacije širinom pulsa, a šest i tri su jedni od tih priključaka. Očitavanje vrijednosti s ultrazvučnog senzora i određivanje kada će se ispustiti ultrazvučni val obavlja se koristeći varijable echo i trig. Echo predstavlja echo priključak modula dok trig predstavlja trig priključak modula. Varijabli echo pridružena je vrijednost dvanaest, a varijabli trig vrijednost trinaest, baš kao što je prikazano i u električnoj shemi senzorskog sustava. Varijabla koja predstavlja signal priključak servomotora s kojim se određuje njegov položaj je servoPin varijabla i pridružen joj je digitalni priključak štita s oznakom jedanaest. To upravljanje će se vršiti pozivanjem metode na objektu servo Servo klase, koja se naziva attach(). Predaje joj se vrijednost priključka koji označava signalnu liniju servomotora. U ovom slučaju predaje joj se varijabla servoPin. Parametri termistora, koji su potrebni za izračun temperature okoline u kojoj se nalazi vozilo, postavljeni su u četiri varijable. Varijabla thermistorPin, čija je vrijednost oznaka ulaznog analognog priključka na senzorskom štitu A5, koristi se za očitavanje napona koji se u trenutku mjerenja nalazi na termistoru. Točnije, očitava se ADC vrijednost koja se onda jednostavnim izračunom pretvara u približni napon koji se nalazi na termistoru. Varijabla B drži vrijednost termalnog indeksa termistora čija vrijednost treba ostati uvijek ista. T1 predstavlja temperaturu u kelvinima na kojoj termistor ima nominalni otpor. Vrijednost nominalnog otpora spremljena je u varijablu pod nazivom R. Vrijednosti koje se koriste za izračunavanje brzine zvuka u zraku oko vozila, uz trenutnu temperaturu zraka koja je izmjerena na termistoru, spremaju se u varijable G, X, i M. U varijablu G spremljena je vrijednost plinske konstante, u varijabli X nalazi se vrijednost koja predstavlja adijabatski koeficijent zraka, a u varijabli M nalazi se molarna masa suhog zraka. Sve opisane varijable koriste se u jednoj ili više funkcija koje se nalaze u nastavku programa. Opisane varijable mogu se vidjeti na slici 5.1.

```
#include <Servo.h>

#define MAX_DISTANCE 200 //maximum distance for hc-sr04 in cm

int isMoving = 1;
Servo servo;

//l298n direction control pins
int in1 = 7;
int in2 = 5;
int in3 = 4;
int in4 = 2;

//l298n speed control pins
int ena = 6;
int enb = 3;

//hc-sr04 control pins
int echo = 12;
int trig = 13;

//servo signal pin
int servoPin = 11;

//thermistor parameters
int thermistorPin = A5;
int B = 3950; //thermal index
int T1 = 298.15; //nominal ressistance temperature in kelvins
int R = 10000; //nominal ressistance

//speed of sound parameters
float G = 8.314; //gas constant
float x = 1.4; //adiabatic coefficient
float M = 0.0289644; //molar mass of dry air
```

Slika 5.1. varijable i konstante programa

5.2. Podešavanje priključaka i postavljanje početnih stanja

U programiranju u Arduino okruženju funkcija setup() koristi se za postavljanje početnih stanja. Ona se poziva na samom početku programa samo jednom. Koristeći funkciju pinMode(), postavljamo priključak na izlazni ili ulazni. Priključke in1, in2, in3 kao i ena, enb i trig postavljamo na izlazne priključke zbog njihove funkcije, a echo postavljamo na ulazni zbog očitavanja visoke ili niske vrijednosti. Na početku izvršavanja programa na ena i enb priključke ispisujemo vrijednost za modulaciju širine pulsa, koja iznosi 100 od maksimalno 255. Što znači da se vozilo neće kretati maksimalnom brzinom. To je napravljeno zbog lakšeg zaustavljanja pri nailasku na prepreku. Na servo objeku poziva se metoda attach() čija je funkcionalnost objašnjena i kao argument predaje joj se varijabla koja sadrži vrijednost signalnog priključka servomotora, a nakon toga metoda write() s argumentom 90. To je učinjeno kako bi se ultrazvučni senzor okrenuo u smjeru kretanja vozila, odnosno kako bi se postavio u središnji položaj servomotora. Opisane postavke mogu se vidjeti na slici 5.2.

```
void setup() {
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);

  analogWrite(ena, 100);
  analogWrite(enb, 100);

  servo.attach(servoPin);
  servo.write(90);
}
```

Slika 5.2. funkcija setup() i početne postavke

5.3. Funkcija za izračunavanje temperature zraka

Funckija za izračunavanje temperature zraka koristi se kako bi se preciznije odredila brzina zvuka. Brzina zvuka u zraku neće se puno razlikovati s promijenom temperature, ali ovako se ipak koristi točniji podatak. Kako bi se došlo do temperature, prvo se mora doći do otpora termistora na toj temperaturi koja se traži. Prvi korak je mjerenje ADC vrijednosti na termistoru. Zatim, kako bi se došlo do napona, ukupni napon koji daje LAFVIN pločica, što je 5 V, dijeli se s 1024 kako bi se dobilo koliko napona odgovara jednoj jedinici ADC vrijednosti. Zatim se množi s izmjerenom ADC vrijednošću i dobiva se približni napon na termistoru. Nakon što se izračunao napon, računa se struja u grani termistora i otpornika. Struja se računa tako što se napon na otporniku koji nije termistor podjeli s njegovim otporom što je 10 kΩ. Na kraju se dolazi do otpora dijeljenjem napona na termistoru sa strujom u grani. Kada se izračunao otpor, temperatura se računa kao što je prikazano na slici 5.3. koristeći izraz (2-2), a cijela opisana funkcija može se vidjeti na slici 5.4.

$$Rt = R \cdot e^{B\left(\frac{1}{T2}-\frac{1}{T1}\right)}$$

$$\frac{Rt}{R} = e^{B\left(\frac{1}{T2}-\frac{1}{T1}\right)}$$

$$\ln\left(\frac{Rt}{R}\right) = B\left(\frac{1}{T2}-\frac{1}{T1}\right)$$

$$\frac{\ln\left(\frac{Rt}{R}\right)}{B} = \frac{1}{T2}-\frac{1}{T1}$$

$$\frac{T1 \cdot \ln\left(\frac{Rt}{R}\right) + B}{B \cdot T1} = \frac{1}{T2}$$

$$T2 = \frac{B \cdot T1}{T1 \cdot \ln\left(\frac{Rt}{R}\right) + B}$$

Slika 5.3. izračun temperature termistora

Na slici 5.3. Rt predstavlja otpor termistora na temperaturi T2, R otpor na temperaturi T1, a B je toplinski indeks.

```
//function to calculate temperature of enviroment in Kelvins
float calculateTemperature() {
  int adcValue = analogRead(thermistorPin);
  float thermistorVoltage = adcValue * 5.0 / 1024.0;
  float current = (5.0 - thermistorVoltage) / 10000.0;
  float thermistorRessistance = thermistorVoltage / current;
  float tempInK = (float(B) * T1) / (T1 * log(thermistorRessistance / R) + B);
  return tempInK;
}
```

Slika 5.4. funkcija za izračunavanje temperature termistora u kelvinima

Funckija calculateTemperature() na kraju vraća vrijednost temperature u kelvinima.

5.4. Fukcije za izračun brzine zvuka i timeouta

Brzina zvuka u zraku oko vozila računa se po izrazu (2-1). Temperatura koja se uvrštava u taj izraz dobiva se pozivanjem funkcije calculateTemperature() koja vraća temperaturu u kelvinima. Nakon izračuna, funkcija calculateSpeedOfSound vraća izračunatu brzinu zvuka koja se koristi u funkciji calculateTimeout(), ali i u nastavku programa. Funkcija calculateTimeout() izračunava koliko dugo treba čekati na povratak zvučnog vala nakon njegovog stvaranja i vraća tu vrijednost. TimeOut vrijednost računa se tako što se maksimalna udaljenost u centimetrima množi s dva, zato što zvučni val treba preći tu udaljenost jednom do prepreke, a drugi puta ju treba prijeći kada se vraća od mjesta prepreke nakon odbijanja od te iste prepreke. Zatim se ta vrijednost dijeli sa 100 kako bi se centimetri pretvorili u metre. Kada se izračuna ukupna udaljenost do prepreke i nazad od prepreke, onda se dijeli s brzinom zvuka, koja se izračunala u funkciji calculateSpeedOfSound. Nakon dijeljenja s brzinom zvuka sve se množi s 1000000 zato što je dobivena vrijednost u sekundama, a za kasniju upotrebu u funkciji koja se koristi za mjerenje visoke vrijednosti na echo priključku, potrebna je vrijednost u mikrosekundama. Opisane funkcije mogu se vidjeti na slici 5.5.

```
//calculate speed of sound in current enviroment in meters per second
float calculateSpeedOfSound() {
  float speedOfSound = sqrt(x * G * calculateTemperature() / M);
  return speedOfSound;
}

//calculate hc-sr04 timeout (time to wait for the wave to return)
float calculateTimeout() {
  float timeOut = 2.0 * MAX_DISTANCE / 100 / calculateSpeedOfSound() * 1000000;
  return timeOut;
}
```

Slika 5.5. funkcija za izračunavanje brzine zvuka i vremena čekanja na povratak zvučnog vala

5.5. Funkcije za kretanje vozila

Napisano je pet funkcija za kretanje vozila. Funkcija moveForward() pokreće vozilo prema naprijed tako što se na out1 i out3 daju visoke vrijednosti odnosno 5 V napona, dok se na out2 i out4 daju niske vrijednosti napona. Na desnoj strani visoka se vrijednost daje na negativnoj strani motora, dok se na lijevoj daje na pozitivnoj strani motora. Funkcija moveBackwards() radi suprotno od funkcije moveForward(). Funkcija turnRight() pokreće vozilo udesno, tako što se i na desnoj i na lijevoj strani na pozitivni pol motora daje visoka vrijednost napona. Funkcija turnLeft() pokreće vozilo ulijevo i daje visoku vrijednost i s desne i s lijeve strane na negativne polove motora. Funkcijom stopMovement() zaustavlja se kretanje vozila, tako što se zaustavlja okretanje motora. Okretanje motora se zaustavlja zato što se na lijevoj i desnoj strani vozila, na negativne i pozitivne polove motora dovodi niska vrijednost napona. Na taj način razlika potencijala je nula, odnosno nema napona između pozitivnog i negativnog pola motora, pa se motor neće kretati. Kombinirajući svih pet funkcija za kretanje, u kasnijem kodu programa, lagano se upravlja smjerom vozila bez pisanja dodatnog koda za kretanje. Sve opisane funkcije za kretanje, mogu se vidjeti na slici 5.6.

```
//movement functions
void moveForward() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void moveBackwards() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void turnRight() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void turnLeft() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void stopMovement() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

Slika 5.6. funkcije za kretanje vozila

5.6. Funkcija za mjerenje udaljenosti

Funkcija za mjerenje udaljenosti, getDistance(), koristi se za izračunavanje udaljenosti prepreke od vozila. Kako bi se došlo do udaljenosti. Prvo se pomoću funkcije digitalWirte() na trig priključak ispisuje visoka vrijednost i zatim se čeka 10 mikrosekundi, zato što je toliko minimalno potrebno kako bi se aktiviralo stvaranje ultrazvučnog vala. Nakon toga se na echo priključku hc-sr04 modula pojavljuje visoka vrijednost. Pomoću funkcije pulseIn() mjeri se trajanje visoke vrijednosti na echo priključku u mikrosekundama. Treći argument unutar funkcije pulseIn() je vrijeme čekanja na povratak vala nakon njegovog stvarnaja. U ovom slučaju to vrijeme se izračunava korištenjem funkcije calculateTimeout(). Ako se val nije vratio u tom vremenu funkcija pulseIn() vraća vrijednost nula. Ako se to dogodi funkcija getDistance() vratit će vrijednost 9999.0 zato što se pretpostavlja da je to vrlo daleka udaljenost od prepreke ako se val nije vratio. Inače, funkcija nastavlja izračunavanje udaljenosti tako što se brzina zvuka, dobivena pozivom funkcije calculateSpeedOfSound(), množi s vremenom putovanja zvuka, koje je podijeljeno s dva budući da je to izmjereno vrijeme koje je potrebno zvučnom valu do prepreke i nazad. Nakon toga, sve se dijeli s 10000 kako bi se dobila udaljenost u centimetrima. Točnije, prvo se dijeli s 1000000 kako bi dobile sekunde, a onda se množi sa 100 kako bi se dobili centimetri, što se svodi na dijeljenje s 10000. Na kraju funkcija vraća udaljenost od prepreke. Opisana funkcija može se vidjeti na slici 5.7.

```
//function to get current distance from obstacle in cm
float getDistance() {
  float distance;
  float timeOfTravel;
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  timeOfTravel = pulseIn(echo, HIGH, calculateTimeout());
  if (timeOfTravel == 0) {
    return 9999.0;
  }
  distance = calculateSpeedOfSound() * timeOfTravel / 2 / 10000;
  return distance;
}
```

Slika 5.7. funkcija za mjerenje udaljenosti do prepreke

5.7. Funkcija za pronalazak novog smjera kretanja nakon nailaska na prepreku

Kada vozilo naiđe na prepreku ispred sebe onda se poziva funkcija findNewDirection(), koja će pronaći novi smjer u kojem se vozilo treba kretati. Prvo funkcija okreće servo na nula stupnjeva, odnosno okreće se ultrazvučni senzor na 0 stupnjeva. To je desno od vozila. Ispušta se ultrazvučni val desno od vozila i očekuje njegov povratak. Udaljenost koja je izmjerena se sprema u varijablu currentDistance. Ako desno ima mjesta, udaljenost koja je izmjerena se onda sprema u varijablu distance i čuva se stupanj servomotora pod kojim je izmjerena ta udaljenost. Ako nema mjesta, ne događa se ništa. Zatim se servomotor okreće naprijed, na 90 stupnjeva. Opet se izvršava ispuštanje ultrazvučnog vala, kako bi se provjerilo nalazili li se prepreka još ispred vozila. Ako ispred vozila ima mjesta i udaljenost je veća od prošle udaljenosti ona se sprema u varijablu gdje je spremljena prošla najveća udaljenost i čuva se kut pod kojim se treba nalaziti servomotor. Ako nema mjesta, ne događa se ništa kao i u prošloj iteraciji. Zadani položaj servomotora je 180 stupnjeva, tako se servomotor okreće lijevo od vozila i opet obavlja mjerenje udaljenosti. Ako ima mjesta i ako je udaljenost veća od prošle, prepisat će prošlu kao i stupnjevi. Ako mjesta nema, opet se neće dogoditi ništa. Između svake iteracije događa se odgađanje programa na 500 milisekundi, iako je dovoljno odgoditi samo 29 milisekudni što je najmanji broj milisekudni između ispuštanja ultrazvučnog vala, zbog vidljivosti i se postavlja na 500. Nakon potpune pretrage servomotor se opet vraća u kut od 90 stupnjeva kako bi bio spreman za kretanje vozila prema naprijed i mjerenje udaljenosti ispred njega. U nastavku funkcije provjerava se ima li varijabla currentDirection vrijednost -1. Ako ima, onda će funkcija vratiti vrijednost 0. Current direction postavljen je na -1 u početku funkcije, a ako se pronašao bilo koji smjer kretanja, postavlja se na kut servomotora pod kojim je pronađen slobodni smjer kretanja. Zatim se izvodi provjera svakog od kuteva, 0, 90 i 180 stupnjeva. Ako je kut 0 stupnjeva, vozilo se određeno vrijeme koje je dovoljno da zaobiđe prepreku okreće desno, ako je kut 90 stupnjeva, nastavlja se kretati naprijed, a ako je kut 180 stupnjeva, okreće se lijevo kako bi izbjeglo prepreku ispred njega. Na kraju funkcija vraća vrijednost 1 ako se funkcija uspijela izvršiti do kraja, što znači da varijabla currentDirection nije zadržala svoju početnu vrijednost koja iznosi -1. Opisana funkcija prikazana je na slici 5.8.

```
//function to find new direction to move in
int findNewDirection() {
  int pos;
  int currentDistance;
  int distance = 0;
  int currentDirection = -1;

  for (pos = 0; pos <= 180; pos += 90) {
    servo.write(pos);
    delay(500);
    currentDistance = getDistance();
    if (currentDistance > distance && currentDistance > 15) {
      distance = currentDistance;
      currentDirection = pos;
    }
  }
  servo.write(90);

  if (currentDirection == -1) {
    return 0;
  }

  if (currentDirection == 0) {
    turnRight();
    delay(1500);
  }
  else if (currentDirection == 90) {
    moveForward();
  }
  else {
    turnLeft();
    delay(1500);
  }

  return 1;

}
```

Slika 5.8. funkcija za pronalazak novog smjera

5.8. Glavna petlja programa, funkcija loop()

U funkciji loop() nalazi se glavni tok programa. Funkcija loop() tokom izvršavanja programa stalno se ponavlja kao beskonačna petlja. Unutar funkcije prvo se provjerava je li isMoving varijabla istinita. Ako je, vozilo se kreće naprijed, dok ne naiđe na prepreku koja je udaljena manje ili 15 centimetara. Ako se dogodi takav slučaj, vozilo se kreće unazad 600 milisekundi i onda zaustavlja svoje kretanje. Nakon toga vozilo traži novi smjer, pozivanjem funkcije findNewDirection. Vraćena vrijednost sprema se u directionFound varijablu. Ako directionFound ima vrijednost 0 vozilo se okreće lijevo dovoljno dugo da dođe do smjera kojeg nije bilo moguće pretražiti iz prošlog položaja. Zatim ponavlja pretragu. Ako smjer nije pronađen isMoving se postavlja na 0. Inače, ako isMoving nije 1, trajno se zaustavlja kretanje vozila. Loop() funkcija vidi se na slici 5.9.

```
void loop() {
  if (isMoving) {
    moveForward();
    if (getDistance() <= 15) {
      moveBackwards();
      delay(600);
      stopMovement();
      int directionFound;
      directionFound = findNewDirection();
      if (!directionFound) {
        turnLeft();
        delay(2300);
        directionFound = findNewDirection();
        if (!directionFound) {
            isMoving = 0;
        }
      }
    }
  }
  else {
    stopMovement();
  }

  delay(40);
}
```

Slika 5.9. Glavna funkcija programa

6. ZAKLJUČAK

U radu je opisano spajanje, sastavljanje i programiranje makete vozila. Cilj vozila je sigurno kretanje uz izbjegavanje prepreka na koje naiđe. Vozilo je uspješno realizirano na dvije platforme. Donja platforma na kojoj se nalazi sustav za pokretanje i gornja platforma na kojoj se nalazi senzorski sustav i sustav za napajanje cijelog vozila. Na gornjoj platformi vozila također se nalazi LAFVIN razvojna pločica na koju je spojen Arduino senzorski štit V5. LAFVIN je potpuno isti kao i Arduino razvojna pločica i programira se na isti način koristeći C jezik. Za programiranje ovog vozila nisu potrebni kompleksni pojmovi iz korištenog jezika. Kako bi program za upravljanje vozilom bio što pregledniji, kod programa je razdvojen u nekoliko funkcija. Koriste se funkcije za izračunavanje temperature zraka, za izračunavanje brzine zvuka i za izračunvanje vremena čekanja ultrazvučnog modula nakon slanja ultrazvučnog vala. Funkcije za kretanje vozila podijeljene su na pet funkcija, za kretanje naprijed, nazad, lijevo, desno i za zaustavljanje. Glavne funkcije u ovom radu su funkcije za mjerenje udaljenosti od prepreke na koju nailazi vozilo i funkcija za pronalazak novog smjera nakon što vozilo naiđe na prepreku ispred sebe. Cijeli tok programa nalazi se u loop() funkciji koja se bez prestanka poziva. Na kraju, spajanjem opisanog programa i sklopovlja, vozilo je uspješno izrađeno i funkcionalnost vozila koja je bila cilj ovog rada je postignuta.

LITERATURA

[1] P, Zenzerović, Arduino kroz jednostavne primjere, Hrvatska zajednica tehničke kulture, Zagreb, 2014.

[2] M, Banzi, Getting Started With Arduino 2nd Edition, Make: Books, Sebastopol, 2011.

[3] J, Boxall, Arduino Workshop, William Pollock, San Francisco, 2013.

[4] V, P, Henč-Bartolič, Kulišić, Valovi i optika, Školska knjiga, Zagreb, 2004.

[5] S, Monk, Electronics Coockbook, O'Reilly Media, Inc., Sebastopol, 2017.

[6] T, Švedek, Poluvodičke komponente i osnovni sklopovi Svezak I. Poluvodičke komponente, Graphis d.o.o., Zagreb, 2001.

[7] Ø.N., Dahl, What Is an H-Bridge?, build electronic circuits, 2018., dostupno na: https://www.build-electronic-circuits.com/h-bridge/ [03.06.2021.]

[8] R, Chan, how to use L298N Motor Driver, Arduino Project Hub, 2020., dostupno na: https://create.arduino.cc/projecthub/ryanchan/how-to-use-the-l298n-motor-driver-b124c5 [03.06.2021.]

[9] Anoniman autor, Interface L298N DC Motor Driver Module with Arduino, Last Minute ENGINEERS, 2021., dostupno na: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/ [05.06.2021.]

[10] J, Wilson, Introduction to Arduino Sensor Shield, THE ENGINEERING PROJECTS, 2020., dostupno na: https://www.theengineeringprojects.com/2020/10/introduction-to-arduino-sensor-shield.html [08.06.2021.]

[11] J, Blum, Exploring Arduino: Tools and Techniques for Engineering Wizardry, John Wiley & Sons, Inc., Indianapolis, 2013.

[12] B.W., D.M., Kernighan, Ritchie, The C programming language second edition, AT&T Bell Laboratories, New Jersey, 1988.

[13] D, D, Griffiths, Griffiths, Head First C, O'Reilly Media, Inc., Sebastopol, 2012.

[14] P, T, Prinz, Crawford, C in a nutshell, O'Reilly Meadia, Inc., Sebastopol, 2006.

[15] T, V, Stranjak, Tomić, C jezik, Školska knjiga, Zagreb, 2005.

SAŽETAK

U ovom radu opisana je izrada i programiranje fizičke makete autonomnog vozila koje će izbjegavati prepreke koje se nađu na putu vozila. Za takvo izbjegavanje koristi se ultrazvučni senzor koji se nalazi na modulu hc-sr04. Električni sustav vozila podijeljen je u dva potsustava. Sustav za kretanje i senzorski sustav koji se koristi za snalaženje u prostoru i mjerenje temperature zbog izračunavnja brzine zvuka. Za mjerenje temperature koristi se termistor. Program vozila podijeljen je u nekoliko funkcija: Funkcija za izračunavanje temperature, brzine zvuka, vremena čekanja povratka ultrazvučnog vala, funkcije za kretanje vozila, funkcija za mjerenje udaljenosti koja se koristi u funkciji za pronalazak novog smjera nakon nailaska na prepreku i glavna petlja programa unutar funkcije loop(). Funkcije u kojima se nalazi glavni dio ovog rada su funkcije za pronalazak novog smjera kretanja i funkcija za mjerenje udaljenosti od prepreke.

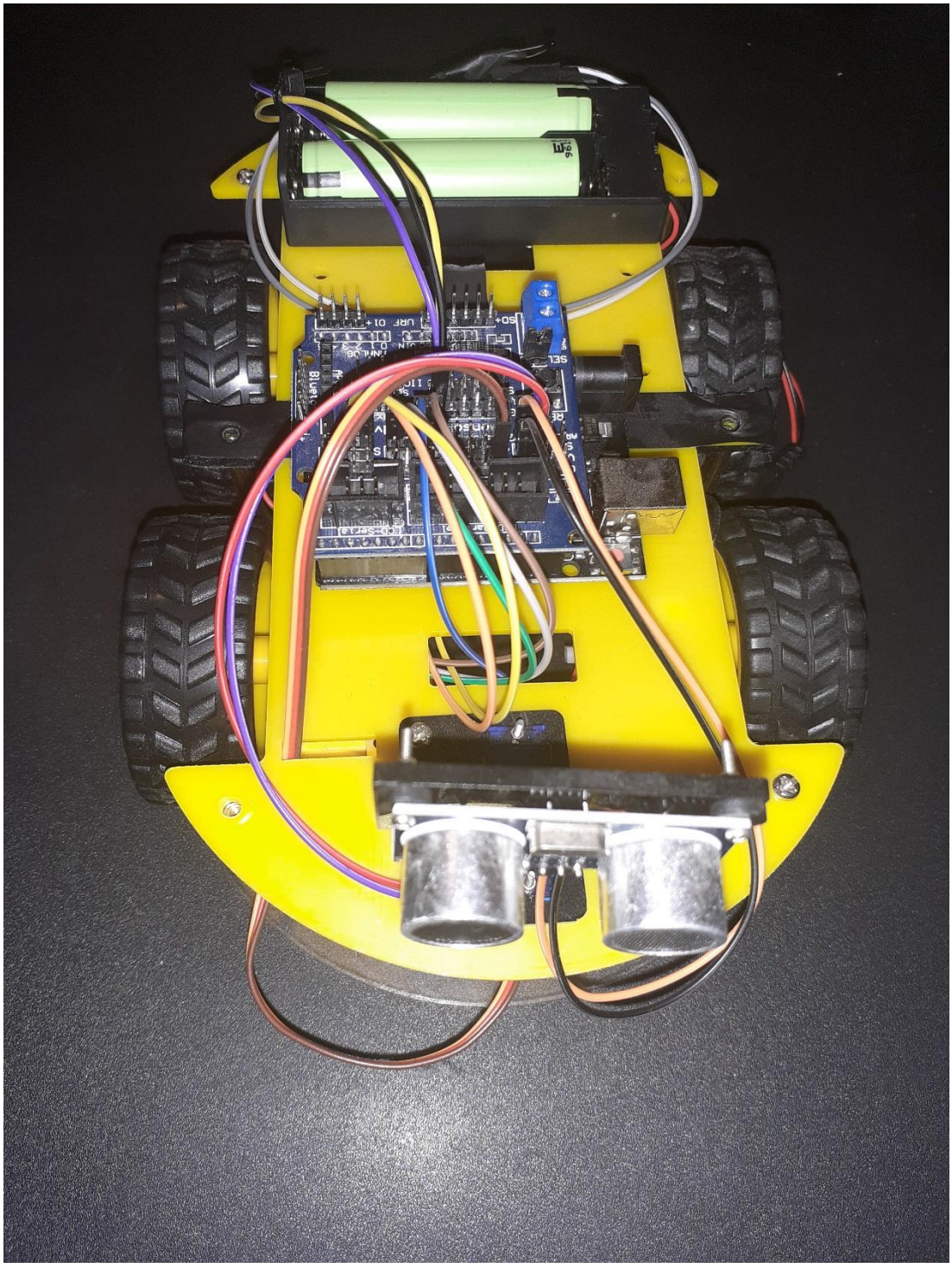
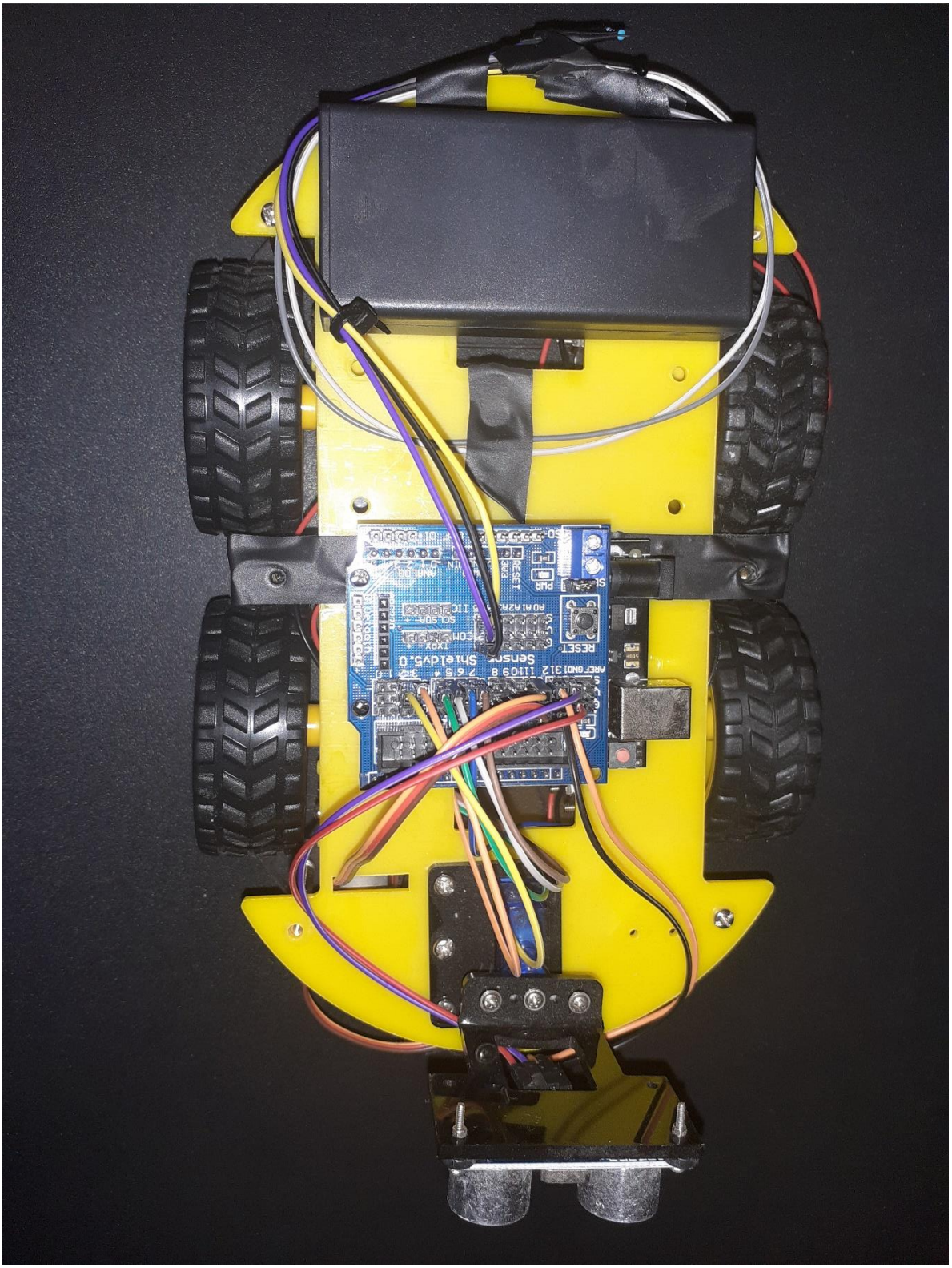
Ključne riječi: autonomno, vozilo , hc-sr04, ultrazvuk, termistor, funkcija

ABSTRACT

Title: Collision avoidance using sonar for an autonomous vehicle model

The focus In this work, construction and programming of a physcial model of autonomous vehicle with obstacle avoidance ability, is described. Obstacle avoidance is achieved using an ultrasoinc sensor that is located on the hc-sr04 module. The electrical system of the vehicle is divided into two subsystems. The system for movement and the sensor system that is used to find the right direciton in space and to measure the environment temperature for speed of sound calculation. A thermistor is used to measure the temperature. The program of the vehicle is divided into sveral functions: a function which calculates the temperature, speed of sound and time to wait for the ultrasonic wave to return, functions for vehicle movement, a function for measuring the distance that is used in the function which searches for a new direction after confronting an obstacle and the main loop of the program inside the loop() function. The functions that represent the main part of this work are the functions for finding a new direction of movement and a function that measures the distance from the obstacle.

Keywords: autonomous, vehicle, hc-sr04, ultrasonic, thermistor, function

PRILOG

U prilogu se nalaze slike potpuno realiziranog vozila.