Izrada jeftine mobilne platforme za razvoj navigacijskih algoritama

Barković, Luka

Sveučilište
Sjever

Završni rad br. 487/EL/2021

Izrada jeftine mobilne platforme za razvoj navigacijskih algoritama

Student

Luka Barković, 0336026296

Mentor

Miroslav Horvatić, dipl. ing.

Varaždin, srpanj 2021. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za elektrotehniku

STUDIJ preddiplomski stručni studij Elektrotehnika

PRISTUPNIK Luka Barković | JMBAG 0336026296

DATUM 02.07.2021. | KOLEGIJ Automatsko upravljanje

NASLOV RADA Izrada jeftine mobilne platforme za razvoj navigacijskih algoritama

NASLOV RADA NA ENGL. JEZIKU Construction of a low-cost mobile platform for the development of navigation algorithms

MENTOR Miroslav Horvatić, dipl. ing. | ZVANJE predavač

ČLANOVI POVJERENSTVA

1. mr. sc. Ivan Šumiga, dipl. ing., viši predavač
2. mr. sc. Matija Mikac, dipl. ing., viši predavač
3. Miroslav Horvatić, dipl. ing., predavač
4. dr. sc. Ladislav Havaš, docent, rezervni član
5.

Zadatak završnog rada

BROJ 487/EL/2021

OPIS

Potrebno je izraditi jeftinu mobilnu platformu koja će omogućiti razvoj navigacijskih algoritama. Za navigacijski i komunikacijski podsustav mobilne platforme koristiti LIDAR i Raspberry računalo. Upravljanje elektromotorima mobilne platforme realizirati Arduino mikrokontrolerskim sustavom. Mobilna platforma treba imati ugrađenu kameru i treba omogućiti ručno upravljanje s udaljenog računala.

U radu je potrebno:
- osmisliti jeftinu mobilnu platformu koja će omogućiti razvoj navigacijskih algoritama
- odabrati i opisati sklopovske komponente mobilne platforme
- osmisliti i realizirati programsku podršku koja će omogućiti rad mobilne platforme
- analizirati cijenu realizirane mobilne platforme
- ispitati i opisati rad mobilne platforme.

ZADATAK URUČEN 06.07.2021. | POTPIS MENTORA

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER

Predgovor

Ovaj završni rad izradio sam samostalno koristeći navedenu literaturu, znanje stečeno tijekom studiranja, pomoć drugih ljudi koji su se bavili sličnim problemima i pomoć mentora.

Posebnu zahvalu upućujem mentoru Miroslavu Horvatiću dipl. ing. koji je uvijek bio dostupan za bilo kakva pitanja te je vrlo kvalitetnim savjetima otklanjao većinu nedoumica i prepreka koje su se našle na putu i samim time smanjio vrijeme potrebno za izradu praktičnog dijela završnog rada.

Također, zahvaljujem se obitelji, djevojci i svim ostalima koji su mi pružali podršku i potporu za vrijeme studiranja i naravno tijekom izrade ovog završnog rada, te im se ujedno zahvaljujem i na strpljenju.

Sažetak

Završnim radom prikazani su zahtjevi koje treba zadovoljiti mobilna platforma, opisani su elementi korišteni za izradu završnog rada, prikazano je međusobno povezivanje i konfiguriranje pojedinih elemenata, upravljanje vozilom i opis gotove mobilne platforme koja je spremna za izradu navigacijskih algoritama. Nakon uvodnog dijela prikazuju se zadaci koje u konačnici mobilna platforma mora ispuniti. Navode se zahtjevi poput skeniranja prostora korištenjem LIDAR (Light Detection and Ranging) senzora, uspješno prenošenje podataka s Raspberry Pi pločice na Arduino korištenjem I2C protokola i slično. Detaljno se opisuje LIDAR senzor i ostali elementi kao što su tranzistorski most L298N, Raspberry Pi 3B+, Arduino Due i relejni modul. Prikazano je povezivanje navedenih elemenata i dane su sheme spajanja, a isto tako i programski kodovi koji pokazuju na koji način se implementira upravljanje vozilom i konfiguracija pojedinih elemenata. Na kraju su prikazani rezultati dobiveni razvojem mobilne platforme, npr. grafički prikaz mape prostora, snimak dobiven Raspberry Pi kamerom i uređaj s kojeg se vrši upravljanje korištenjem WiFi veze. Prikazana je obrada podataka dobivenih LIDAR senzorom, realizirana, korištenjem Python programskog jezika te nakon toga programiranje da bi se stvorila mapa prostora. Izvedeno je upravljanje mobilnom platformom korištenjem kursorskih tipki na tipkovnici udaljenog računala. Tablično su prikazane cijene pojedinih elemenata pri čemu se uočava da je uvjerljivo najskuplji dio mobilne platforme LIDAR senzor. Izračunati su ukupni troškovi izrade mobilne platforme.

Ključne riječi: Arduino, LIDAR senzor, mobilna platforma, navigacija, Python, Raspberry Pi

Summary

The final work presents the requirements that need to be met by the mobile platform, describes the elements used to create the final work, shows the interconnection and configuration of individual elements, vehicle management and a description of the finished mobile platform ready to create navigation algorithms. After the introductory part, the tasks that the mobile platform must ultimately fulfill are presented. Requirements such as space scanning using LIDAR (Light Detection and Ranging) sensors, successful data transfer from Raspberry Pi to Arduino using I2C protocol, and stuff like that. The LIDAR sensor and other elements such as the L298N transistor bridge, Raspberry Pi 3B +, Arduino Due and relay module are described in detail. The connection of the mentioned elements is shown and the connection diagrams are given, as well as the program codes that show how the vehicle management and the configuration of the individual elements are implemented. Finally, the results obtained with the development of the mobile platform are presented, such as a graphical representation of the space map, a snapshot obtained by a Raspberry Pi camera and a device from which mobile platform is controlled using the WiFi network. The processing of data obtained by the LIDAR sensor, realized, using the Python programming language and then programming to create a space map is shown. Control of the mobile platform was performed using the cursor keys on the keyboard of the remote computer. The prices of individual elements are tabulated, and it can be seen that the most expensive part of the mobile platform is the LIDAR sensor. The total cost of building a mobile platform was calculated.

Key words: Arduino, LIDAR sensor, mobile platform, navigation, Python, Raspberry Pi

Popis korištenih kratica

SLAM Simultaneous Localization And Mapping
Oznaka za obilježavanje skupine algoritama za autonomna vozila

LIDAR Light Detection and Ranging
Oznaka za obilježavanje senzora

SDA Serial Data Line

SCL Serial Clock Line

GND Ground

GPIO General Purpose Input/Output
Oznaka za obilježavanje ulaza/izlaza Raspberry Pi pločice

PWM Pulse-Width Modulation
Oznaka za obilježavanje pulsno širinske modulacije

NO Normal Open
Oznaka za obilježavanje kontakata releja koji se bez prisustva napona nalaze u otvorenom stanju

NC Normal Closed
Oznaka za obilježavanje kontakata releja koji se bez prisustva napona nalaze u zatvorenom stanju

Csv Comma-separated values
Ekstenzija datoteka koje imaju vrijednosti odvojene zarezom (.csv)

ToF Time-of-Flight
Oznaka za obilježavanje tipa LIDAR senzora

ROS Robot Operating System
Oznaka robotskog operacijskog sustava

UART Universal Asynchronous Receiver-Transmitter
Oznaka za obilježavanje portova koji rade kao univerzalni asinkroni prijemnik-predajnik

SADRŽAJ

1. UVOD

Sve veći razvoj tehnologije i moderniji načini prodaje te ljudska potreba za sve više informacija, zahtijevaju vrlo učinkovite tehničke sustave. Problem planiranja puta za autonomna vozila postepeno je privlačio pažnju ljudi te je postao hitni problem, kojeg je trebalo riješiti u procesu umjetne inteligencije i ekonomskog razvoja [1]. Obzirom da je upravljanje autonomnim vozilima još uvijek aktualna tema u mnogim državama svijeta, motivacije za ovaj rad nije nedostajalo. Spoznaja koliko se vodeći svjetski proizvođači automobila, poput Tesle, trude i ulažu u autonomna vozila, bila je glavni povod za odabir ove teme.

Pod pojmom planiranje puta misli se na traženje dijelova puta koji nemaju prepreke od početne do krajnje točke. Također se podrazumijeva i traženje najkraćeg puta do odredišta i minimalnog vremena koje je potrebno utrošiti da bi se došlo do tog istog odredišta. Izrada mobilne platforme koja će kasnije služiti za razvoj navigacijskih algoritama bit će detaljno prikazana u ovom radu.

Dosadašnji razvoj navigacijskih algoritama pokazao je odličan razvoj SLAM (Simultaneous Localization And Mapping) metoda za unutarnje primjene. Korištenjem SLAM-a moguće je brzo i precizno pregledati zgrade raznih veličina i složenosti za što bi u praksi bilo potrebno puno više vremena kad bi se pregled vršio ručno [3]. Mnoge implementacije SLAM metoda koriste LIDAR senzor jer korištenje tog senzora daje značajne prednosti poput velikog dometa, relativno malih dimenzija, velike preciznosti u slabo osvjetljenim okruženjima i mogućnosti da uhvati i najsitnije detalje na velikoj udaljenosti [4]. Zbog svih tih prednosti odlučeno je koristiti LIDAR senzor za izradu ovog završnog rada, iako se i dalje dosta često koriste druge vrste senzora poput ultrazvučnih, koji imaju znatno manju preciznost.

Preostali sadržaj ovog završnog rada dan je u nastavku, a prikazan je sljedećim redoslijedom. U drugom poglavlju daje se detaljan opis zahtjeva koje treba zadovoljiti mobilna platforma koja se može koristiti za razvoj navigacijskih algoritama. U trećem poglavlju se detaljno opisuju sve komponente mobilne platforme koje su odabrane na temelju zahtjeva opisanih u drugom poglavlju. U četvrtom poglavlju opisuje se spajanje svih komponenata mobilne platforme i njihovo konfiguriranje. U petom poglavlju opisuje se korištenje mobilne platforme: upravljanje (koje se vrši s udaljenog računala) i mjerenje LIDAR-om, te spremanje i prikaz izmjerenih podataka. U šestom poglavlju tablično se prikazuju cijene pojedinih elemenata i ukupan trošak svih dijelova. Konačan cilj je izraditi

mobilnu platformu na kojoj se mogu razvijati navigacijski algoritmi pa se u tu svrhu prikazuje praktična implementacija i zajednički rad svih teoretski opisanih dijelova. Na kraju su dani zaključci koji proizlaze iz ovog završnog rada. Uz zaključak je naveden i osobni komentar, a uz to i problemi koji su se javljali tijekom izrade završnog rada.

2. OPIS ZAHTJEVA MOBILNE PLATFORME

Izrada ovog završnog rada sastoji se od nekoliko koraka koji će detaljno biti opisani u nastavku. Kako bi se u konačnici dobio kompletan završni rad, radio se svaki korak zasebno te se nakon toga krenulo na povezivanja svih koraka u jedan kompletan projekt. Između toga trebalo je uključiti i nekoliko dodatnih međukoraka poput uspostavljanja komunikacije između Raspberry Pi-a i Arduina, uspostavljanja komunikacije između LIDAR senzora i Raspberry Pi-a te postavljanja Raspberry Pi kamere. Jedan dio praktičnog dijela završnog rada je posvećen i modeliranju 3D dijelova koji su služili kao platforme na koje su se pričvrstile Raspberry Pi pločica i LIDAR senzor te Raspberry Pi kamera.

Prvotno se trebalo posvetiti LIDAR senzoru, a nakon što je bilo omogućeno skeniranje i zaprimanje podataka s LIDAR-a, trebalo je uspostaviti konekciju između samog LIDAR-a i Raspberry Pi pločice kako bi se podaci, koje je davao LIDAR, mogli prenijeti u Raspbian operacijski sustav i obraditi programima pisanim u Python programskom jeziku. To je ujedno predstavljalo prvi zahtjev koji je trebalo ostvariti da bi se moglo krenuti s daljnjim radom.

Komunikacija sa samim Raspbianom ostvarena je preko operacijskog sustava Windows instaliranog na udaljenom upravljačkom računalu, pri čemu se za komunikaciju koristio program *Remote Desktop Connection.* Nakon što su podaci dobiveni u Raspbian, moglo se iz tih podataka nacrtati mapu prostora korištenjem SLAM-a pa je to bio drugi zahtjev koji je trebalo ostvariti.

Kada je bilo ostvareno mapiranje trebala se ostvariti komunikacija između Raspberry Pi-a i Arduina korištenjem I2C načina komunikacije preko sabirnice. Nakon toga su se mogle slati upute Arduinu na koji način da vrši upravljanje motorima odnosno na koji način, u kojem trenutku i kojom brzinom u koju stranu treba uključiti elektromotore, za što je poslužio tranzistorski most L298N.

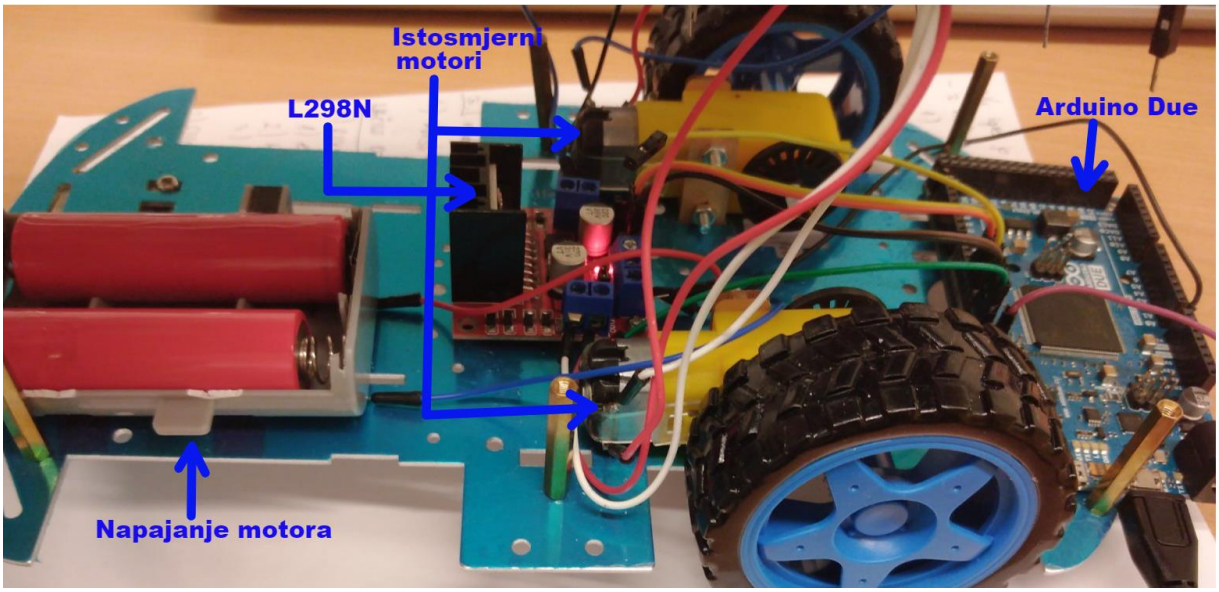
Nakon što se pokazalo da Arduino sigurno i pouzdano odrađuje svoj dio posla, dodan je i relejni modul koji je isključivao i uključivao rad motora u ovisnosti o tome što je bilo napisano u programu. Također, na prednji dio mobilne platforme postavljena je kamera koja je na bilo kojem uređaju, koji ima pristup preko WiFi veze morala u realnom vremenu prikazivati put kojim se kreće platforma.

Još jedan zahtjev, koji platforma mora ispuniti, bio je da omogućuje upravljanje korištenjem kursorskih tipki koje se nalaze na tipkovnici udaljenog prijenosnog računala. Konkretno, mora se detektirati koja tipka na tipkovnici prijenosnog računala je pritisnuta i

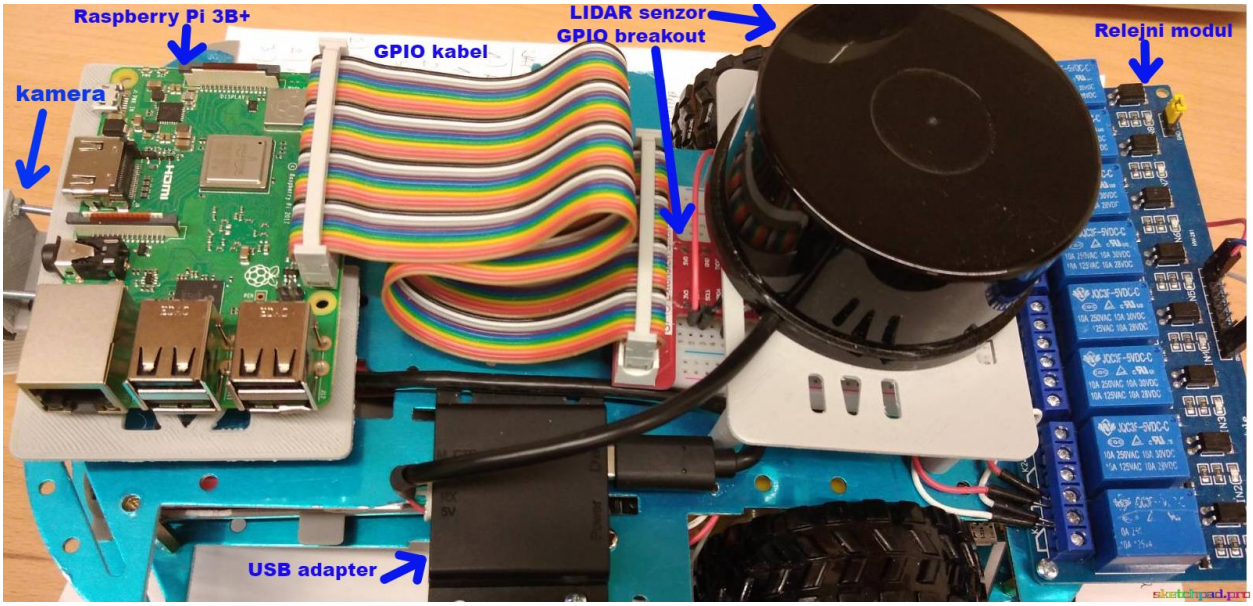
ta informacija se mora prenijeti u Raspberry Pi korištenjem WiFi veze i zatim obraditi u Python programskom jeziku. Na kraju se mobilna platforma mora kretati u smjeru koji označava kursorska tipka na prijenosnom računalu.

Ključan zahtjev koji se u konačnici mora ispuniti jest da mobilna platforma bude spremna za razvijanje navigacijskih algoritama, a da bi to bio slučaj potrebno je ispuniti sve prethodno definirane zahtjeve.

Kako bi se dobila jasnija slika gdje se što nalazi i kako izgledaju dijelovi koji su korišteni u izradi završnog rada, na slikama 2.1 i 2.2 prikazan je konačan izgled mobilne platforme s označenim dijelovima koji su detaljno opisani u trećem poglavlju.



Slika 2.1 Donji dio platforme s korištenim dijelovima



Slika 2.2 Gornji dio platforme s korištenim dijelovima

3. OPIS DIJELOVA MOBILNE PLATFORME

3.1. LIDAR senzor

Da bi autonomna vozila mogla biti pouzdana i sigurna za upotrebu, moraju sadržavati veoma dobre senzore koji to omogućuju. Senzori su uređaji koji imaju doticaj s vanjskim svijetom i od njega prikupljaju informacije, a upravljanje autonomnim vozilima se vrši u ovisnosti o podacima sa senzora.

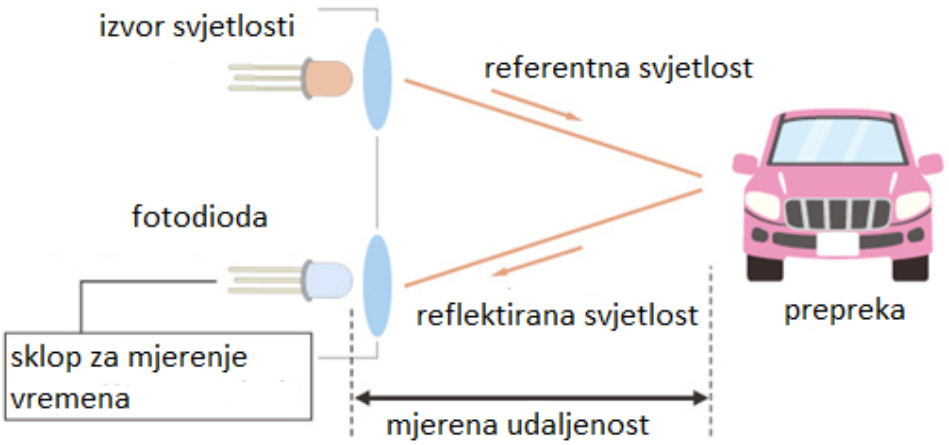
U potpuno autonomnim vozilima nema vozača pa su upravo senzori, ključni uređaji zaduženi za detektiranje prepreka na putu. Postoje mnoge vrste senzora, no danas su u autonomnim vozilima najzastupljeniji LIDAR senzori. Nije trebalo dugo da se LIDAR senzori počnu više razvijati i masivnije proizvoditi pa tako danas postoje mnoge vrste LIDAR senzora. No, princip je kod svih ostao isti, ili vrlo sličan. Ono što je karakteristično za LIDAR senzor jest da generira mikrometarski val svjetlosti na temelju kojeg skenira okolinu. Često se unutar autonomnih vozila koriste i kamere koje vrlo lako mogu detektirati prepreke u normalnim vremenskim uvjetima, no kod loših vremenskih uvjeta njihova preciznost je znatno smanjena. S druge strane, LIDAR senzor može raditi normalno i raspoznati najsitnije detalje u ekstremnim vremenskim uvjetima, a novije verzije mogu razlikovati vremenske uvjete, kretanje, brzinu kretanja te smjer [5]. LIDAR senzor prima reflektiranu lasersku zraku te dobiva sve informacije preko lasera, a obzirom na veliku brzinu skeniranja može vozilu dati dovoljno vremena za reakciju.

Ovisno o principu na kojem se temelji njihov rad, postoji više tipova LIDAR senzora. Najčešće korišteni tip LIDAR senzora u sustavima upravljanja autonomnim vozilima je sasvim sigurno ToF (Time-of-Flight) tip koji će detaljno biti predstavljen u ovom potpoglavlju. Zanimljivo je spomenuti da postoje mnoge tvrtke koje proučavaju i proizvode LIDAR senzore, a neke od njih su: Velodyne Lidar [23], Quanergy [24], Robosense [25] itd. Neke vrste LIDAR senzora prikazane su na slici 3.1.

Slika 3.1 Neke vrste LIDAR senzora [7]

3.1.1. Princip rada ToF metode LIDAR senzora

ToF metoda mjerenja udaljenosti koristi impulsni laser koji generira laserske zrake pojedinačno ili kontinuirano do prepreka. Kada započne generiranje laserskih zraka, automatski se uključuje i brojač vremena koji je ugrađen u senzor. ToF LIDAR senzor radi na principu mjerenja vremena koje je potrebno da laserska zraka stigne do prepreke i da se nakon refleksije od prepreke vrati do prijemnika. Princip rada prikazan je na slici 3.2. Da bi se laserske zrake generirale treba postojati izvor laserske svjetlosti koji se nalazi u predajniku LIDAR-a i fotoelektrični senzor u prijemniku laserske svjetlosti koji prima reflektiranu svjetlost.



Slika 3.2 Princip rada ToF tipa LIDAR senzora [8]

Sklop za mjerenje vremena zapravo radi na način da izračunava razliku vremena koje je potrebno da laserska zraka dođe od predajnika do prijemnika. Da bi se mogla odrediti udaljenost do prepreke mora se poznavati broj generiranih impulsa laserske zrake *n*, a taj podatak je senzoru jako dobro poznat. Udaljenost se izračunava kako je prikazano sljedećim izrazom:

$$d = \frac{c \cdot t}{2} = \frac{c}{2 \cdot f} \cdot n \qquad (1)$$

Pri čemu je: d – udaljenost od prepreke (mm)

c – brzina svjetlosti (km/s)

n – broj generiranih impulsa

f – frekvencija sklopa za mjerenje vremena (Hz)

t - vrijeme potrebno da laserska zraka dođe od predajnika do prijemnika (s)

Princip rada zasniva se na tome da senzor uključuje laser pomoću kojeg generira laserske zrake, pri čemu se pokreće brojač impulsa. Brojač broji impulse kako bi na temelju toga mogao proračunati udaljenost. Kada se zraka svjetlosti odbije od prepreke, difuzno prolazi kroz zrak te dolazi do prijemnika. Na prijemniku se laserska zraka pretvara u električni signal pomoću nekog od fotoelektričnih detektora, a nakon toga se pojačava preko pojačala. Signal s pojačala ujedno služi i kao informacija brojaču da je signal došao do odredišta. Kada su prikupljeni navedeni podaci, mikroupravljač u LIDAR-u prema formuli (1) računa udaljenost do objekta.

ToF metoda je danas vrlo raširena i ne koristi se samo u LIDAR senzorima već nalazi svoju primjenu u mnogim drugim granama. Neki primjeri koji to dokazuju su upotreba u vojne svrhe gdje se često prate usmjereni vojni projektili, također se često koristi i u topografskim mjerenjima, mjerenjima udaljenosti u svemiru itd.

3.1.2. Vrste LIDAR senzora s kontinuiranim skeniranjem

LIDAR senzori koji vrše kontinuirano skeniranje okoline sastavljeni su od motora, koji služi za vrtnju glave za skeniranje okoline, zatim od laserske diode, koja generira lasersku zraku, prijemnika reflektirane laserske zrake te mikroprocesora. Dijele se u senzore koji skeniraju okolinu pomoću jedne laserske zrake te na senzore koji istovremeno skeniraju okolinu s više laserskih zraka. Vojni radari predstavljaju klasičan primjer korištenja principa

LIDAR senzora jer rade na principu odašiljanja mikrovalova, kojima onda mjere vrijeme potrebno da stignu od predajnika do prijemnika, na temelju čega računaju udaljenost kao i LIDAR senzor. Razlika između LIDAR-a i radara je u tome da LIDAR emitira i prima lasersku zraku, dok radar emitira i prima radiovalove. Poput LIDAR-a i vojni radari mogu skenirati okolinu oko svih 360 °.

3.1.3. LIDAR senzori s jednostrukom zrakom svjetlosti

Glavna obilježja senzora koji odašilju samo jednu zraku svjetlosti su: visoka frekvencija, velik domet, jedan prijemnik, velika rezolucija kuta i mala težina. Dosta često se koriste u autonomnim vozilima gdje vrlo precizno mogu odrediti prepreke koje se nalaze ispred vozila. Naravno, prepreke koje se nalaze s bočnih strana vozila, teže uočavaju, pa se u tu svrhu obično ugrađuje više takvih senzora koji mogu odrediti položaj vozila i njegovu širinu. Ovakav tip senzora je precizniji, ali i jeftiniji od senzora s višestrukim zrakama svjetlosti. Na slici 3.3. prikazan je primjer jednog LIDAR senzora s jednostrukom zrakom.

Slika 3.3 LIDAR senzor s jednostrukom zrakom svjetlosti [9]

3.1.4. LIDAR senzor s višestrukim zrakama svjetlosti

Ovaj tip LIDAR senzora istovremeno generira više zraka svjetlosti što mu omogućava praćenje više prepreka odjednom. Sadrži zrcala koja rotiraju zajedno s motorom

pa se rotacijom tih zrcala i refleksijom laserskih zraka od zrcala omogućava stvaranje višestrukih zraka. Na taj način omogućava stvaranje 3D slike okoline, što dodatno poboljšava rezultate SLAM metode. Princip rada ovakvih senzora je prilično složen, no opet se zasniva na istim pravilima. Senzor prvo generira višestruke laserske zrake na prethodno opisan način, nakon toga te zrake dolaze do prepreka od kojih se reflektiraju prema LIDAR prijamniku, nakon čega se vrši uzorkovanje podataka. Zatim se podaci trebaju obraditi u mikroupravljaču pri čemu se pažnja obraća na to koji podaci su ispravni, a koji nisu odnosno koji predstavljaju prepreke, a koji ne. Ako se, na primjer, koristi senzor koji generira 8 laserskih zraka istovremeno, tada, nakon uzorkovanja najmanje 4 podatka moraju biti jednaka, kako bi LIDAR zaključio da se radi o prepreci.

Slika 3.4 LIDAR s višestrukim zrakama svjetlosti [9]

3.1.5. Radni mehanizam LIDAR senzora TG15

Tip LIDAR senzora koji se koristi u ovom radu je TG15. Princip rada je u potpunosti identičan principu rada opisanom u potpoglavlju 3.1.1., jer LIDAR senzor koristi ToF metodu.

Radni mehanizam YDLIDARA serije TG15 ima 4 načina rada:

1. **Mirujući način** – kad se priključi na napajanje motor LIDARA se ne vrti te samim time LIDAR ne mjeri udaljenost i laser je ugašen.
2. **Skeniranje** – kada je u ovom načinu rada, LIDAR se okreće i vrši kontinuirano uzorkovanje okoline i grafički prikazuje okolinu u realnom vremenu na zaslonu računala.

3. **Stop način** – u ovom načinu rada LIDAR primarno radi, no u međuvremenu dođe do greške pa sustav sam gasi LIDAR te javlja grešku (motor se ne vrti, laser ne mjeri udaljenost, …).
4. **Način zaštite od isključenja** – u ovom načinu LIDAR mora kontinuirano primati informacije o skeniranju, odnosno o mjerenju udaljenosti, u vremenu od 3 sekunde. Ukoliko ne primi informaciju o udaljenosti koju mjeri, sustav će automatski ugasiti LIDAR. Takav način rada nije inicijalno uključen.

Slika 3.5 YDLIDAR TG15 [6]

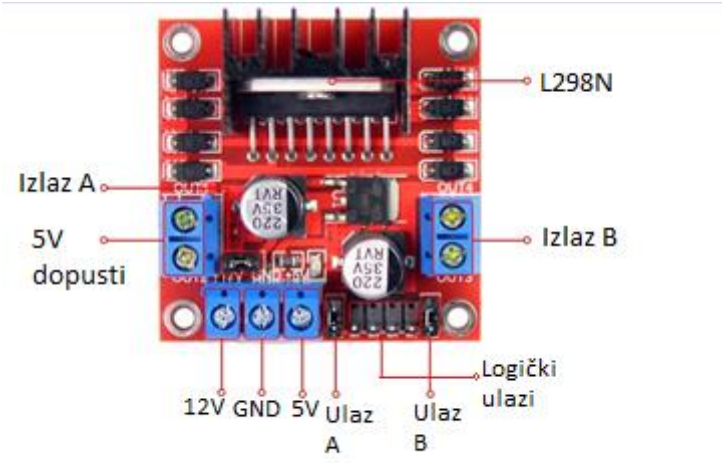
Glavne karakteristike LIDAR senzora tipa TG15 navedene su u nastavku:

- Frekvencija dometa = 20 kHz
- Frekvencija skeniranja = 5 - 15 Hz
- Domet = 0.05 – 15 m
- Kut skeniranja = 360°
- Rezolucija kuta = 0.09° - 0.22°
- Veličina laserske zrake = ϕ70 · 35 mm

3.2. Tranzistorski most L298N

Tranzistorski most L298N, koji je sastavljen od MOSFET tranzistora, omogućuje rad u sva četiri kvadranta što znači da korištenjem PWM (Pulse-Width Modulation) pulsno širinske modulacije može upravljati brzinom vrtnje, ali omogućuje isto tako i promjenu smjera vrtnje. Istovremeno je moguće spojiti dva motora, pa se u ovom završnom radu koriste dva istosmjerna motora. Most je mogao osigurati napajanje za motore od 5 V do 35

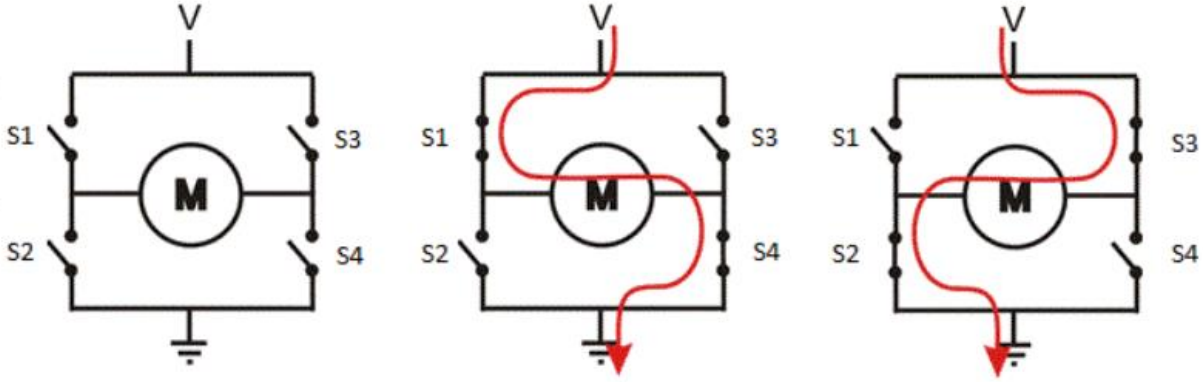
V pri čemu se u ovom radu koristilo 8.4 V na ulazu mosta. Obzirom da postoje padovi napona zbog MOSFET tranzistora, na motore je došao napon od približno 6 V što je ujedno i maksimalan napon kojim su motori mogli biti napajani. Pri 6 V broj okretaja u minuti, bez tereta na vratilu elektromotora, iznosio je 12 000 rpm, a motor je trošio struju od 170 mA. Za napajanje su se koristile dvije punjive Litijeve baterije, svaka od 4.2 V. Izgled mosta L298N s označenim pinovima prikazan je na slici 3.6.



Slika 3.6 H-most L298N [14]

3.2.1. Princip rada mosta L298N

L298N radi na principu koji je prikazan na slici 3.7. Potrebno je paziti da se istovremeno ne uključe sklopke S1 i S2 ili S3 i S4 jer se na taj način izazove kratki spoj, što dovodi do uništenja mosta. Radi se o elektroničkim sklopkama koje se sastoje od tranzistora i njemu antiparalelno spojene diode, no na slici 3.7 taj spoj prikazan je pomoću jedne sklopke. Ako se želi osigurati motorski rad u prvom kvadrantu, moraju voditi sklopke S1 i S4. Tada se elektromotori vrte u smjeru kazaljke na satu. Ako se želi osigurati motorski rad u trećem kvadrantu, tada, moraju voditi sklopke S2 i S3 pa se elektromotori vrte u suprotnom smjeru od kazaljke na satu.

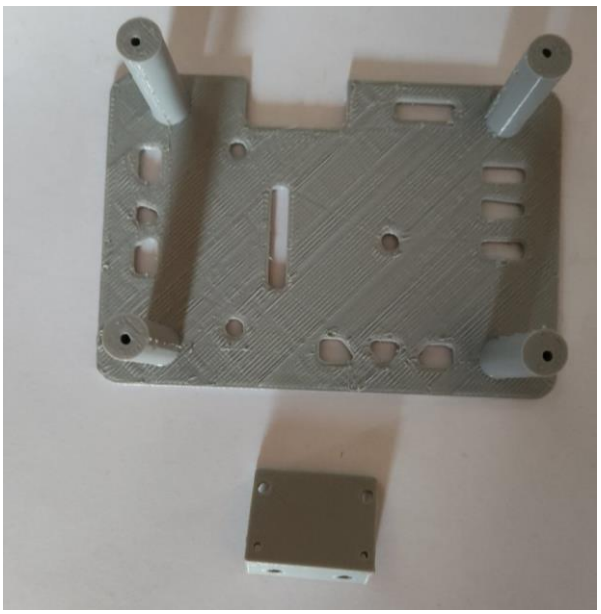


Slika 3.7 Princip rada mosta L298N [14]

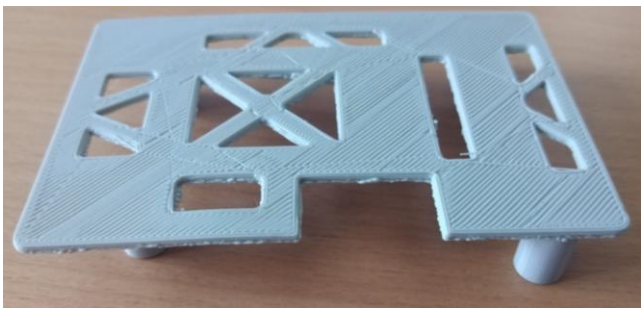
3.3. 3D printanje postolja za Raspberry Pi, LIDAR senzor i kameru

Još jedan problem koji je bilo potrebno riješiti prilikom izrade završnog rada bila je izrada postolja za Raspberry Pi pločicu, LIDAR senzor te Raspberry Pi kameru. Postolja su omogućila lakšu montažu navedenih elemenata na mobilnu platformu. Treba imati na umu da LIDAR skenira okolinu tako da ga je trebalo postaviti na višu poziciju od ostalih elemenata pa je postolje poslužilo kao idealno rješenje.

Modeliranje postolja vršilo se u programu *Solidworks*, dok se završna priprema prije printanja odvijala u programu *Cura Slicer*. Izgled gotovog postolja prikazan je na slikama 3.8 i 3.9.



Slika 3.8 Postolje za LIDAR i kameru



Slika 3.9 Postolje za Raspberry Pi

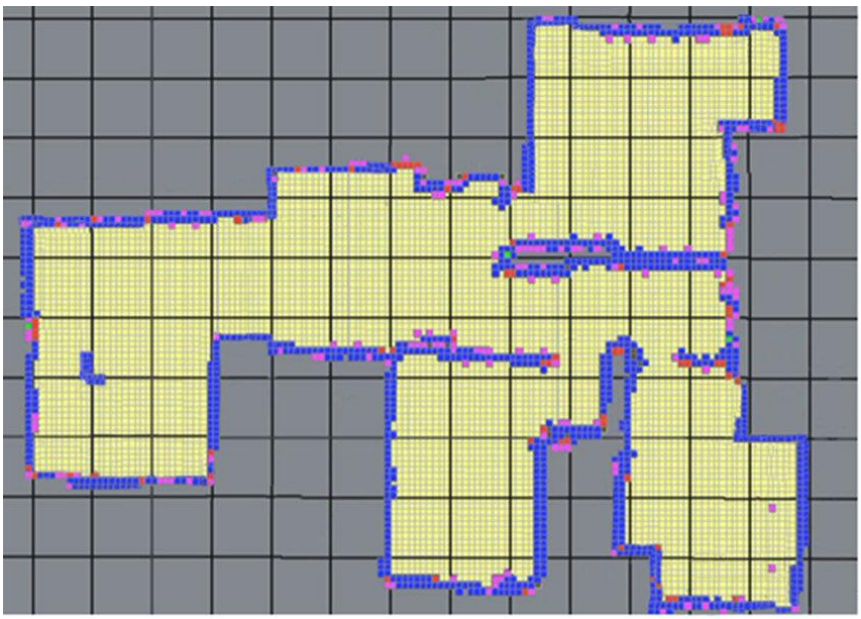
3.4. SLAM

Jedan softverski dio mobilne platforme prikazan je u ovom potpoglavlju. Pojam istovremene lokalizacije i mapiranja, poznatiji pod nazivom SLAM, predstavlja temeljni način upravljanja autonomnim vozilima i robotima u unutarnjim prostorima. Kao što mu i sam naziv govori, SLAM (Simultaneous Localization And Mapping), omogućava rješavanje lokaliziranja i mapiranja. Početna istraživanja vezana uz ovu tematiku sežu u osamdesete godine 20. stoljeća. Predlagala su se razna rješenja najčešće korištenjem kamera s različitim karakteristikama, no sva ta rješenja su pala u vodu nakon upotrebe LIDAR senzora. U usporedbi s kamerama LIDAR je davao mnogo preciznija mjerenja, te nije ovisio o rasvjeti, a isto tako je davao i mnogo više korisnih podataka.

3.4.1. Mapiranje

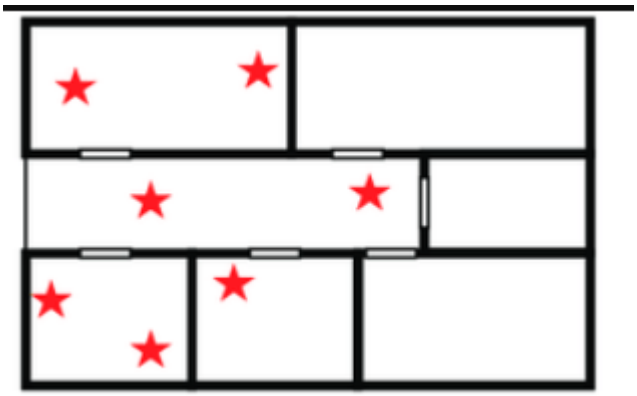
Iako se mapiranje u praksi najčešće izvodi korištenjem ROS-a (Robot Operating System), u ovom završnom radu ROS nije korišten, već je mapiranje izvedeno korištenjem programskog koda koji je opisan u praktičnom dijelu u poglavlju 4. Postoje tri osnovne vrste mapa koje se koriste kod mapiranja okoline, a navedene su u nastavku ovog potpoglavlja.

Prvi tip mape koji se koristi je mapa temeljena na mreži, odnosno mapa koja okolinu prikazuje pomoću mnogo sitnih kvadratića, a vrijednost svakog kvadratića odgovara dijelu prostora u realnom svijetu. Ovisno o tome kakva rezolucija se definira, takva će biti i veličina kvadratića. Takav tip mape prikazan je na slici 3.10 i korišten je u ovom završnom radu zbog toga što pruža najviše detalja, a rezolucija se može lako promijeniti, uz dodatnu prednost da popunjenost mape ne zahtijeva nikakve predefinirane simbole. Nedostatak ovog tipa mape jest veoma složen izračun kod velike rezolucije i velikog broja podataka.



Slika 3.10 Mapa temeljena na mreži [12]

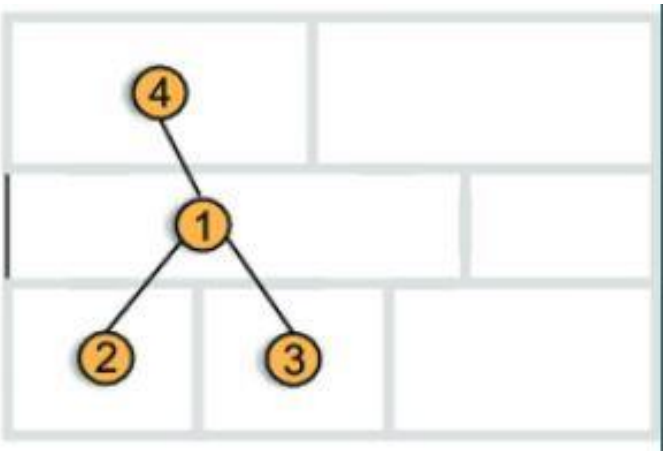
Drugi tip mape jest mapa koja prepreke u okolini prikazuje pomoću prije definiranih simbola. Simboli mogu biti linije, točke ili bilo kakvi drugi oblici koji će robotu dati do znanja da se radi o prepreci i da će svaki doticaj s tim simbolom izazvati koliziju. Primjer takve mape prikazan je na slici 3.11. Prednosti vezane uz mapiranje pomoću simbola su da je mapa fleksibilna i svaki simbol je neovisan o bilo kom drugom simbolu. Također se može vršiti mapiranje korištenjem različitih senzora u isto vrijeme. Nedostaci su da podaci koje senzor daje moraju biti spremljeni kao klasa za koju su definirani simboli, ukoliko to nije slučaj, tada će svi podaci, koji nisu spremljeni na adekvatan način, biti odbačeni.



Slika 3.11 Mapa temeljena na simbolima [12]

Posljednji tip mape prostora je topološka mapa koja je prikazana na slici 3.12. Ovo je zapravo najjednostavniji tip mape koji prikazuje prepreke pomoću čvorova i povezuje te čvorove te tako prikazuje jesu li te prepreke povezane ili nisu, odnosno postoji li prolaz do

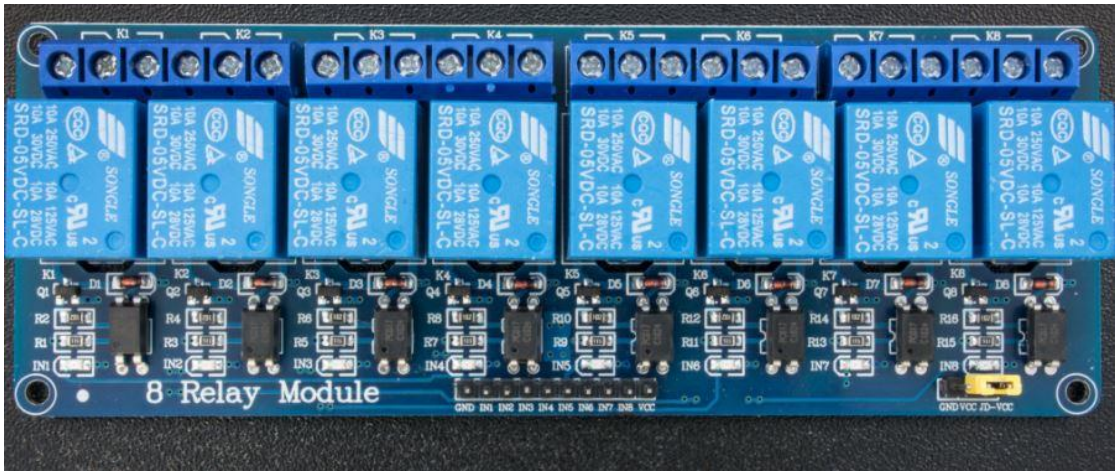
tih prepreka, poput vrata, stepenica ili slično. Zadržavaju se samo najbitnije informacije pa ne postoje podaci o udaljenosti ili smjeru. Prednost ovog tipa mape jest jednostavno korištenje, posebno za planiranje puta, no gubi se dosta informacija o prepreci, a isto tako je teško prepoznati sam čvor pa to treba dodatno opisati u programu.



Slika 3.12 Topološki tip mape [12]

3.5. Relejni modul

Relejni modul, korišten u završnom radu, sadržavao je 8 releja i prikazan je na slici 3.13. Svaki relej sadržavao je zavojnicu, željeznu kotvu i kontakte. Kada struja prolazi kroz zavojnicu, počne se stvarati magnetsko polje koje onda privuče željeznu kotvu i samim time zatvori kontakte koji se nalaze na kotvi. Napon koji je potrebno dovesti zavojnici kako bi se relej aktivirao iznosi 5 V, a struja 20 mA. Maksimalni napon trošila koji se može priključiti iznosi 30 V (DC) i 250 V (AC), a struja 10 A (DC) i 10 A (AC).



Slika 3.13 Relejni modul 8-kanalni

3.6. Raspberry Pi 3B+

Raspberry Pi model 3B+ glavno je računalo mobilne platforme, a služi za procesiranje informacija. Ovaj model Raspberry računala, prikazan na slici 3.14, odabran je zato što ima ugrađen modul za WiFi komunikaciju i zato što nam nije bilo dostupno novije i bolje, Raspberry Pi 4 računalo. Glavne značajke Raspberry Pi modela 3B+ [30] su 64-bitni procesor frekvencije 1.4 GHz, 1GB RAM memorije, Gigabit Ethernet, WiFi na frekventnom području 2.4 GHz i 5 GHz, a može pružiti brzine do nekoliko gigabita po sekundi, u teoriji. Postoji i mogućnost napajanja Raspberry Pi-a korištenjem UTP kabla poznatije pod nazivom Power over Ethernet. Postoji mogućnost povezivanja korištenjem Bluetootha, mikro SD kartica predstavlja trajnu memoriju, a Raspberry Pi sadrži 40 GPIO (General Purpose Input Output) pinova.

Slika 3.14 Raspberry Pi 3B+

3.7. Arduino Due

Arduino Due [31] je prva Arduino pločica bazirana na 32-bitnom mikroupravljaču. Sadrži 54 digitalnih ulazno-izlaznih pinova pri čemu se 12 od tih 54 mogu koristiti kao PWM izlazi, ima 12 analognih ulaza, 2 digitalno-analogna pretvarača, 4 UART porta, tipku za reset, tipku za brisanje programa, SDA i SCL pinove koji služe za ostvarenje serijske komunikacije. Arduino Due prikazan je na slici 3.15.

Slika 3.15 Arduino Due

4. SPAJANJE I KONFIGURIRANJE DIJELOVA

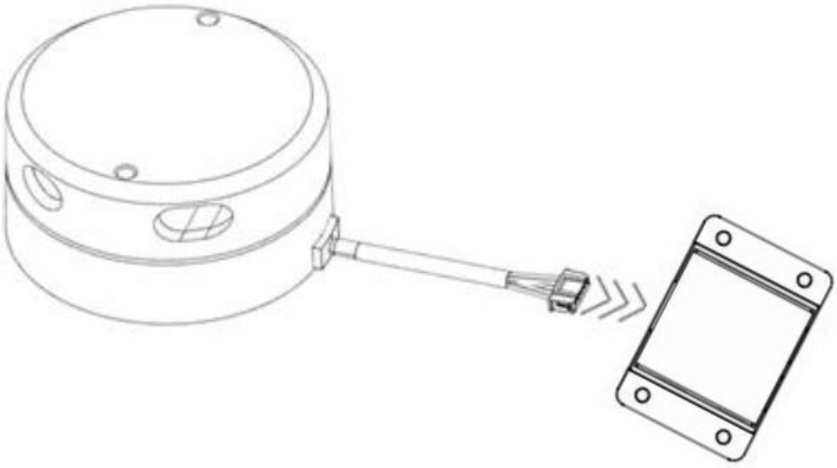
4.1. Dobivanje podataka iz LIDAR-a

Za izradu završnog rada korišten je razvojni komplet YDLIDAR serije TG koji je namijenjen grafičkom prikazivanju prepreka pomoću mnogo sitnih točaka. Razvojni komplet sadrži:

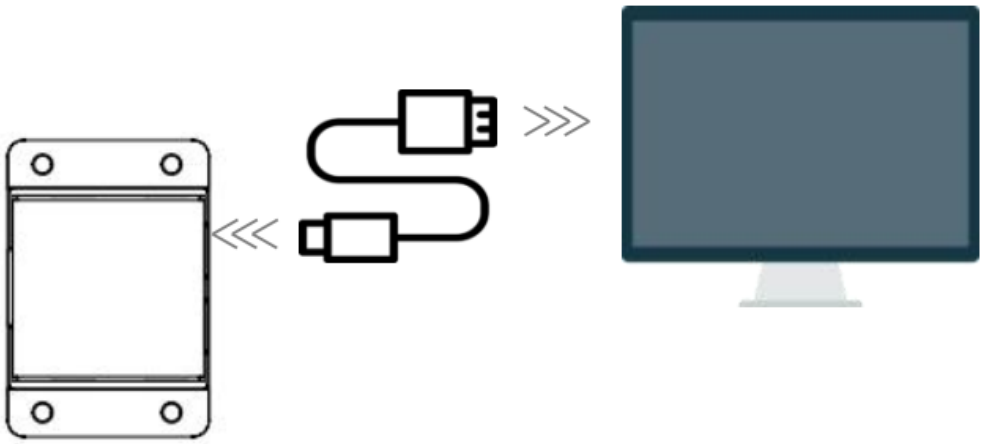
- LIDAR serije TG koji sadrži integrirani motorni pogon
- USB kabel tip C
- USB adapter pločica koja služi za povezivanje LIDARA i računala preko USB kabla; sadrži dva USB priključka, jedan priključak je za podatke, a drugi za pomoćno napajanje koje se koristi ukoliko je glavni izvor napajanja nedovoljne snage

Slika 4.1 LIDAR, USB C kabel, USB adapter [6]

Kako bi LIDAR uopće mogao raditi, potrebno je povezati kabel sa 5 pinova, koji izlaze iz LIDAR senzora, s USB adapterom (slika 4.2) koji omogućava da se senzor poveže s računalom preko USB kabela (slika 4.3).



Slika 4.2 Povezivanje LIDAR senzora s USB adapterom [6]



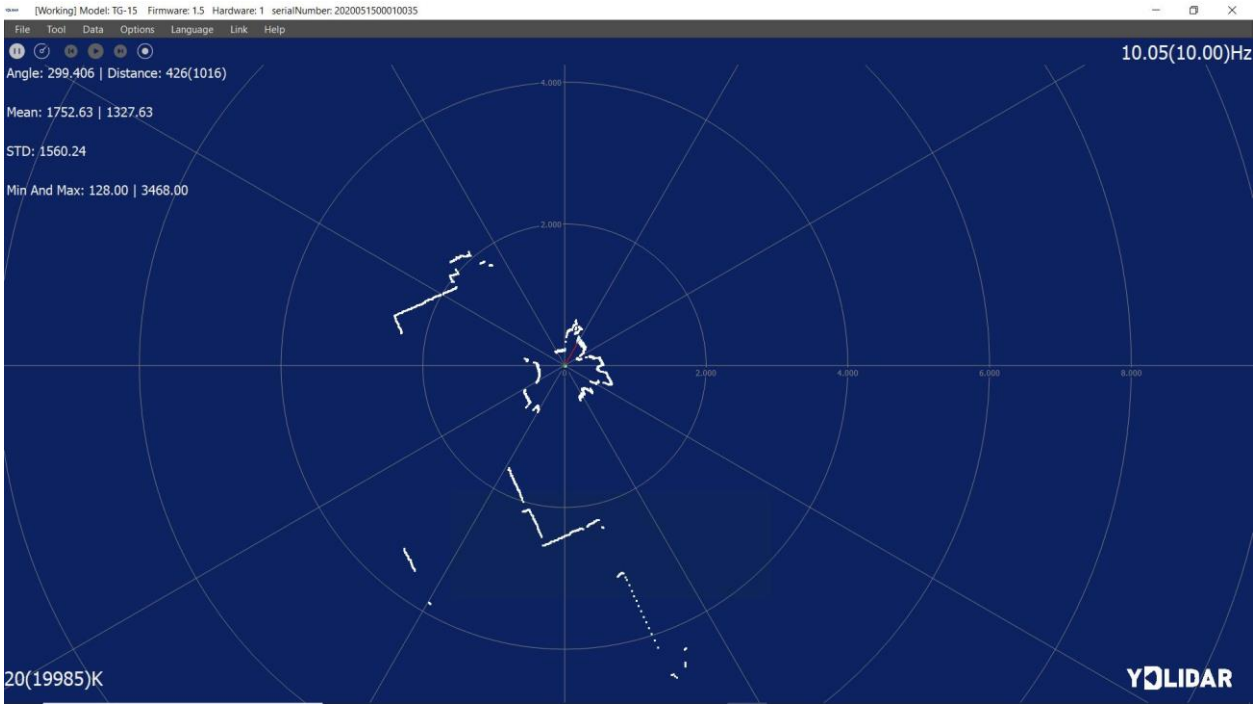
Slika 4.3 Povezivanje USB adaptera s računalom [6]

Nakon što se USB adapter poveže s računalom, LIDAR se nalazi u načinu mirovanja te s obzirom da ima ugrađen integrirani driver neće pokrenuti skeniranje tako dugo dok mu se ne da znak. U ovom slučaju nije se koristio dodatni izvor napajanja. Kako bi se osposobila komunikacija između PC-a i USB adaptera, bilo je potrebno instalirati driver za USB adapter port UART.

YDLIDAR omogućava *Point Cloud View* i skeniranje u realnom vremenu te isto tako spremanje skeniranih podataka u obliku vanjske datoteke kako bi se kasnije mogli koristiti za daljnju obradu.

Nakon pokretanja softvera *PointCloudViewer.exe* otvara se prozor na kojem se može pokrenuti LIDAR i podesiti potrebna frekvencija vrtnje motora koja može biti u rasponu od 5 Hz do 15 Hz. Ukoliko se poveća frekvencija, povećat će se i brzina motora, a ukoliko se

unese nova frekvencija potrebno je ponovno pokrenuti LIDAR da bi promjene bile snimljene. Nakon skeniranja bilježe se podaci koji izgledaju kao na slici 4.4, gdje je skeniranje izvedeno pri frekvenciji 10 Hz. Udaljenost od prepreka prikazana je pomoću koncentričnih kružnica koje prikazuju udaljenosti u metrima.



Slika 4.4 LIDAR-primjer skeniranja

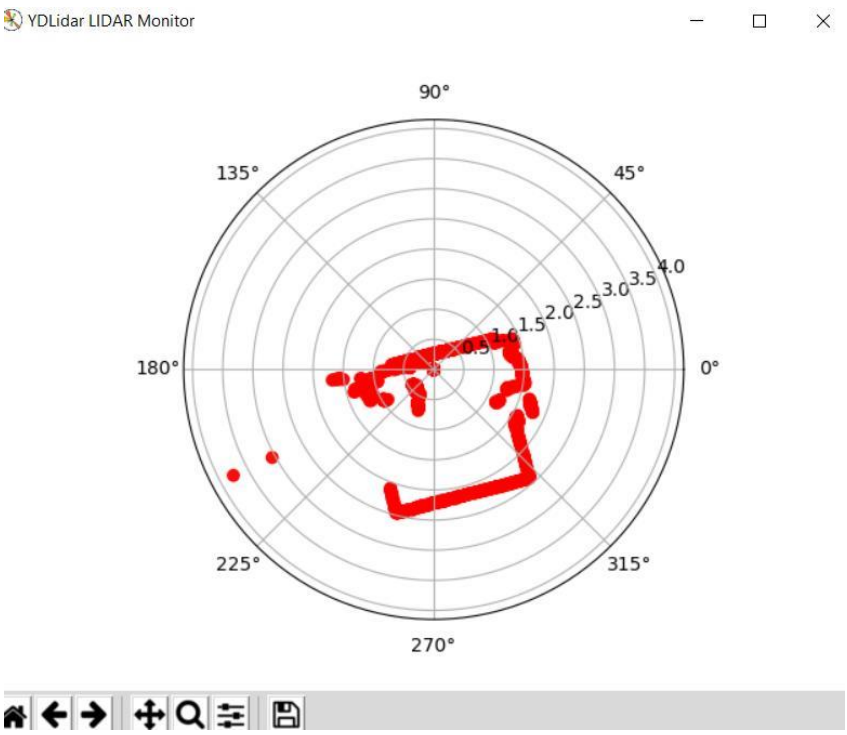
Prilikom rada s LIDAR senzorom ne postoje velika ograničenja, no treba obratiti pažnju na činjenicu da je predviđena temperatura za rad LIDAR-a u rasponu od 0 do 50 °C. Ukoliko radi izvan tog raspona temperatura, imat će smanjenu preciznost.

Nakon što se izvršilo testiranje mogućnosti koje LIDAR nudi i naučio princip rada, bilo je potrebno povezati LIDAR s Raspberry Pi uređajem kako bi se skenirani podaci spremali direktno u Raspbian gdje su se kasnije obrađivali korištenjem programa napisanih u Python programskom jeziku. Kako bi se LIDAR povezao s Raspberry Pi uređajem bilo je potrebno, nakon prespajanja USB kabela s računala na Raspberry Pi, korištenjem terminala instalirati nekoliko programskih knjižnica, knjižnica koje omogućavaju komunikaciju LIDAR senzora i Raspbiana. Nakon toga, mogao se opet pokrenuti senzor na dva različita načina: jedan način bio je korištenjem gotovog programa dobivenih od samog proizvođača unutar naprednog uređivača programskog koda pod nazivom *Thonny*, dok je drugi način bio korištenje terminala i opet pokretanje tog istog programa. Rezultat je u konačnici bio isti te su se mogli dobiti rezultati u dva različita načina zapisa. Jedan način zapisa rezultata skeniranja LIDAR-a je čisti tekstualni prikaz (slika 4.5) i prikazuje identifikacijski broj skena,

domet te frekvenciju skeniranja. Drugi način zapisa jest grafički (slika 4.6) i prikazuje praktički identični zapis kao što izgleda i zapis u samom *Windows* operacijskom sustavu, no ovoga puta je zapis nešto kompaktniji i zornije prikazan s obzirom da su rezultati prikazani u polarnom koordinatnom sustavu tako da se lakše može odrediti pozicija prepreke.

```
Scan received[1616014236240895000]: 2857 ranges is [4.201697]Hz
Scan received[1616014236383565000]: 2855 ranges is [4.204642]Hz
Scan received[1616014236525899000]: 2867 ranges is [4.187037]Hz
Scan received[1616014236672096000]: 2867 ranges is [4.187037]Hz
Scan received[1616014236812670000]: 2867 ranges is [4.187037]Hz
Scan received[1616014236959298000]: 2871 ranges is [4.181201]Hz
Scan received[1616014237100589000]: 2869 ranges is [4.184117]Hz
Scan received[1616014237245143000]: 2869 ranges is [4.184117]Hz
Scan received[1616014237387706000]: 2865 ranges is [4.189961]Hz
Scan received[1616014237531494000]: 2861 ranges is [4.195821]Hz
Scan received[1616014237673319000]: 2861 ranges is [4.195821]Hz
Scan received[1616014237817160000]: 2859 ranges is [4.198757]Hz
Scan received[1616014237959641000]: 2861 ranges is [4.195821]Hz
Scan received[1616014238103250000]: 2857 ranges is [4.201697]Hz
Scan received[1616014238246688000]: 2863 ranges is [4.192889]Hz
Scan received[1616014238393173000]: 2863 ranges is [4.192889]Hz
Scan received[1616014238532928000]: 2869 ranges is [4.184117]Hz
Scan received[1616014238676793000]: 2869 ranges is [4.184117]Hz
Scan received[1616014238822338000]: 2869 ranges is [4.184117]Hz
Scan received[1616014238962672000]: 2865 ranges is [4.189961]Hz
Scan received[1616014239107397000]: 2861 ranges is [4.195821]Hz
Scan received[1616014239249980000]: 2861 ranges is [4.195821]Hz
Scan received[1616014239393121000]: 2853 ranges is [4.207591]Hz
Scan received[1616014239535865000]: 2853 ranges is [4.207591]Hz
Scan received[1616014239678468000]: 2855 ranges is [4.204642]Hz
Scan received[1616014239821591000]: 2855 ranges is [4.204642]Hz
Scan received[1616014239963772000]: 2855 ranges is [4.204642]Hz
Scan received[1616014240111263000]: 2851 ranges is [4.210543]Hz
Scan received[1616014240249730000]: 2849 ranges is [4.213500]Hz
Scan received[1616014240391941000]: 2849 ranges is [4.213500]Hz
Scan received[1616014240533240000]: 2847 ranges is [4.216461]Hz
Scan received[1616014240676850000]: 2845 ranges is [4.219426]Hz
Scan received[1616014240819782000]: 2849 ranges is [4.213500]Hz
Scan received[1616014240967710000]: 2851 ranges is [4.210543]Hz
Scan received[1616014241111361000]: 2849 ranges is [4.213500]Hz
Scan received[1616014241247392000]: 2845 ranges is [4.219426]Hz
```

Slika 4.5 Tekstualni prikaz skeniranja LIDAR-a



Slika 4.6 Grafički prikaz skeniranja LIDAR-a

4.2. Obrađivanje podataka korištenjem Raspberry Pi računala

Raspberry Pi pločica je ključan dio ovog završnog rada s obzirom da se pomoću nje provodi programiranje koje bi mobilnu platformu trebalo učiniti inteligentnom, u smislu da zna kako postupiti kada naiđe na prepreku. Prije početka bilo kakvog programiranja trebalo je povezati Raspberry Pi pločicu s laptopom obzirom da se programiranje vršilo korištenjem prijenosnog računala. To je izvedeno na način da se prije prvog povezivanja koristio SSH protokol unutar samog Raspbiana, a nakon što se pristupilo grafičkom sučelju mogla se očitati i IP adresa Raspberry Pi priključka, koja je kasnije poslužila za bežično povezivanje korištenjem *Remote Desktop Connection* aplikacije. Za prvo povezivanje koristio se Ethernet kabel s obzirom da bežično povezivanje nije bilo omogućeno, no nakon toga se komunikacija konstantno odvijala korištenjem WiFi veze i to preko pristupne točke.

Programiranje se odvijalo u razvojnom okruženju za Python programski jezik, razvojnom okruženju pod nazivom *Thonny*. Upravljanje mobilnom platformom izvelo se korištenjem kursorskih tipki na tipkovnici računala, dok je isto tako realizirano i mapiranje korištenjem SLAM metode. SLAM je metoda razvijena za autonomna vozila koja omogućava da se istovremeno vrši mapiranje prostora, kako bi vozilo spoznalo prepreke koje se nalaze oko njega, te lokaliziranje kako bi vozilo saznalo gdje se nalazi u tom prostoru.

4.2.1. Upravljanje korištenjem kursorskih tipki

U ovom načinu upravljanja koristile su se kursorske tipke na tipkovnici prijenosnog računala kako bi se upravljalo vozilom, što znači da se podaci sa senzora nisu koristili. Ovakav način upravljanja izveden je iz razloga da se postepeno ulazi u ideju i shvaćanje upravljanja vozilom. Za korištenje kursorskih tipki iskorištena je Python programska knjižnica pod nazivom *Pygame* koja je dizajnirana za pisanje videoigara te zbog toga omogućava pristup hardverskim komponentama poput tipkovnice, koja se u ovom slučaju i koristila.

Dio programskog koda koji je ključan za izvedbu ovog načina upravljanja prikazan je u sekvenci Program 1.

```
def getKey(KeyValue):
    answer = False
    for eve in pygame.event.get():
        pass
    keyInput = pygame.key.get_pressed()
    myKey = getattr(pygame, 'K_{}'.format(KeyValue))
    if keyInput [myKey]:
        answer = True
    pygame.display.update()

    return answer
```

Program 1 Funkcija koja vraća informaciju koja tipka je pritisnuta

Ovdje je prikazana upotreba funkcije ***getKey()*** koja preko *Pygame* platforme pokušava otkriti koja tipka je pritisnuta i ukoliko to uspije otkriti sprema vrijednost u *myKey* varijablu, te funkcija vraća vrijednost, odnosno postavlja varijablu *answer* u *True*. U protivnom, ukoliko nije pritisnuta tipka na tipkovnici, funkcija ne vraća nikakvu vrijednost te varijablu *answer* postavlja u *False*.

```
def main():
    if getKey('UP'):
        bus.write_byte(addr, 56)

    elif getKey('DOWN'):
        bus.write_byte(addr, 57)

    elif getKey('LEFT'):
        bus.write_byte(addr, 58)

    elif getKey('RIGHT'):
        bus.write_byte(addr, 59)

    else:
        bus.write_byte(addr, 65)
```

Program 4 Funkcija koja šalje podatke u Arduino preko sabirnice

U dijelu programskog koda Program 2 prikazana je glavna funkcija koja koristi vrijednost prethodno definirane funkcije ***getKey()*** kako bi tu vrijednost prenijela Arduinu. Dakle, pomoću funkcije ***getKey()*** dobiva se informacija koja tipka na tipkovnici je pritisnuta. Zatim se upotrebljava i funkcija ***bus.write_byte()*** koja, kao parametre, prima fiksnu adresu preko koje se odvija komunikacija s Arduinom, te parametar *byte* u koji se sprema informacija o pritisnutoj tipki. Ta dva parametra se preko sabirnice šalju na Arduino. Konkretno, ukoliko je pritisnuta tipka *UP* tada se preko adrese 0x08 uspostavlja

komunikacija i šalje se bajt iznosa 56 koji predstavlja informaciju da je pritisnuta upravo ta tipka i nijedna druga.

Kako bi se mogli prenijeti podaci na Arduino potrebno je definirati sabirnicu preko koje će se vršiti komunikacija te samu adresu koja će biti rezervirana isključivo za tu komunikaciju. U ovom slučaju definirana je sabirnica 1 te adresa 0x08 na toj sabirnici što se može vidjeti na sekvenci Program 3.

```
bus = smbus.SMBus(1)
addr = 0x08
```

Program 6 Definiranje sabirnice za komunikaciju

4.2.2. Mapiranje korištenjem SLAM metode

SLAM metoda, koja sadržava niz algoritama, omogućava da vozilo postane autonomno, odnosno da može samostalno proći određeni put od jedne do druge točke, a da se pritom ne sudari s preprekama koje se nalaze u okolini. U ovom radu prikazan je jedan dio SLAM metode. Radilo se o mapiranju koje je vozilu omogućavalo spoznaju okoline, odnosno da vozilo snimi kako prostor oko njega izgleda i gdje bi se mogle nalaziti pojedine prepreke.

4.2.2.1. Mapiranje

Zadatak mapiranja bio je skenirati okolinu korištenjem LIDAR senzora te na temelju toga dobiti prikaz te okoline u obliku mnogo sitnih točkica. Kada se završilo skeniranje okoline, zapravo se dobila mapa prostora koja je određivala prostor unutar kojeg se vozilo smije kretati. Ukoliko na nekim mjestima nije bilo istočkane površine, vozilu je to govorilo da na tom mjestu nema prepreka te da se tu slobodno smije kretati pa je trebalo obratiti pažnju da se kod mapiranja dobije zatvoreni prostor, kako vozilo ne bi moglo napustiti mapu.

Prije mapiranja trebalo je u programskom kodu inicijalizirati LIDAR senzor prateći postavke proizvođača. Trebalo je podesiti parametre poput priključka, na koji je senzor povezan, brzine prijenosa, koja je bila postavljena na 512 000, zatim tip LIDAR senzora, koji je bio ToF, frekvencija skeniranja bila je postavljena na 10.0 Hz, brzina uzorkovanja bila je 9 s, maksimalni kut bio je 180.0 °, a minimalni -180.0°. Maksimalna udaljenost koju je senzor mogao mjeriti bila je postavljena na 32.0 m, a minimalna na 0.01 m. Podešenje prethodno objašnjenih postavki prikazano je u sljedećem dijelu programskog koda Program 4.

```
laser = ydlidar.CYdLidar();

    laser.setlidaropt(ydlidar.LidarPropSerialPort, port);
    laser.setlidaropt(ydlidar.LidarPropSerialBaudrate, 512000);
    laser.setlidaropt(ydlidar.LidarPropLidarType, ydlidar.TYPE_TOF);
     laser.setlidaropt(ydlidar.LidarPropDeviceType,
     ydlidar.YDLIDAR_TYPE_SERIAL);
    laser.setlidaropt(ydlidar.LidarPropScanFrequency, 10.0);
    laser.setlidaropt(ydlidar.LidarPropSampleRate, 9);
    laser.setlidaropt(ydlidar.LidarPropSingleChannel, False);
    laser.setlidaropt(ydlidar.LidarPropMaxAngle, 180.0);
    laser.setlidaropt(ydlidar.LidarPropMinAngle, -180.0);
    laser.setlidaropt(ydlidar.LidarPropMaxRange, 32.0);
    laser.setlidaropt(ydlidar.LidarPropMinRange, 0.01);
```

Program 7 Inicijalizacija LIDAR senzora

Mapiranje se izvodilo na način da je na početku trebalo prilagoditi podatke koje senzor prikazuje. Drugim riječima, trebalo je od svih podataka, prikazivati samo kut i udaljenost od prepreke. Ta dva podatka su bila ključna kako bi se mogla nacrtati mapa prostora korištenjem Bresenhamove metode [26]. Filtriranje podataka izvršeno je tako da se unutar programskog koda, kojim se upravlja radom LIDAR-a, izbrišu postojeći podaci za prikaz i korištenjem naredbe ***scan.points[i].angle*** prikaže trenutni kut, a korištenjem naredbe ***scan.points[i].range*** trenutna udaljenost. Podaci se prikazuju tako dugo dok postoje skenirane točke, odnosno dok je uključeno skeniranje.

Sljedeći korak bio je spremiti podatke koje senzor prikazuje, u obliku .csv (comma-separated values) datoteke, kako bi se kasnije mogli koristiti za crtanje mape. Kako bi se podaci mogli spremati u .csv datoteku trebalo ih je pretvoriti u tekst korištenjem funkcije *str()*. Prilikom svakog prolaska kroz petlju zapisao se jedan redak u .csv datoteku i to tako dugo dok je senzor vraćao kut i udaljenost, što se može vidjeti na dijelu programskog koda Program 5 koji je dan u nastavku.

```
ret = laser.turnOn();
        scan = ydlidar.LaserScan();

        while ret and ydlidar.os_isOk() :
            r = laser.doProcessSimple(scan);
            if r:
                for i in range(scan.points.size()):
                    print(scan.points[i].angle, scan.points[i].range);
                     csvresult.write(str(scan.points[i].angle) + ", "
                     + str(scan.points[i].range) + "\n");
            else :
                print("Failed to get Lidar Data")
```

Program 10 Uzimanje kuta i udaljenosti sa senozra te pretvorba u .csv datoteku

Nakon što su podaci bili spremljeni, moglo se krenuti na crtanje i to na način da su se prethodno spremljeni podaci učitali u programski kod korištenjem funkcije ***file_read()***. Ta funkcija je zapravo otvorila .csv datoteku i čitala redak po redak podatke koji su bili zapisani, te ih spremala kao niz u obliku decimalnih brojeva što je prikazano u dijelu programskog koda Program 6.

```
def file_read(f):

    with open(f) as data:
        measures = [line.split(",") for line in data]
    angles = []
    distances = []
    for measure in measures:
        angles.append(float(measure[0]))
        distances.append(float(measure[1]))
    angles = np.array(angles)
    distances = np.array(distances)
    return angles, distances
```

Program 12 Funkcija koja čita podatke u .csv datoteci

Za obradu tih podataka služio je Bresenhamov algoritam koji je omogućavao odabir točaka i spajanje tih točaka kako bi se dobila aproksimacija ravne linije između dviju točaka. Za korištenje Bresenhamovog algoritma trebalo je zadati početne uvjete, odnosno koordinate početnih i konačnih točki. Nakon toga se određivalo koliko je linija strma te ukoliko je strma, trebalo je zarotirati liniju, odnosno zamijeniti početne i konačne točke. Zatim je trebalo stvarati točke između definiranih krajnjih i početnih točaka na način da se ponavljalo generiranje točaka tako dugo dok se nije izlazilo iz granica početne i krajnje točke. Nakon što se dobila lista točaka, trebalo ju je okrenuti ukoliko su prethodno definirane

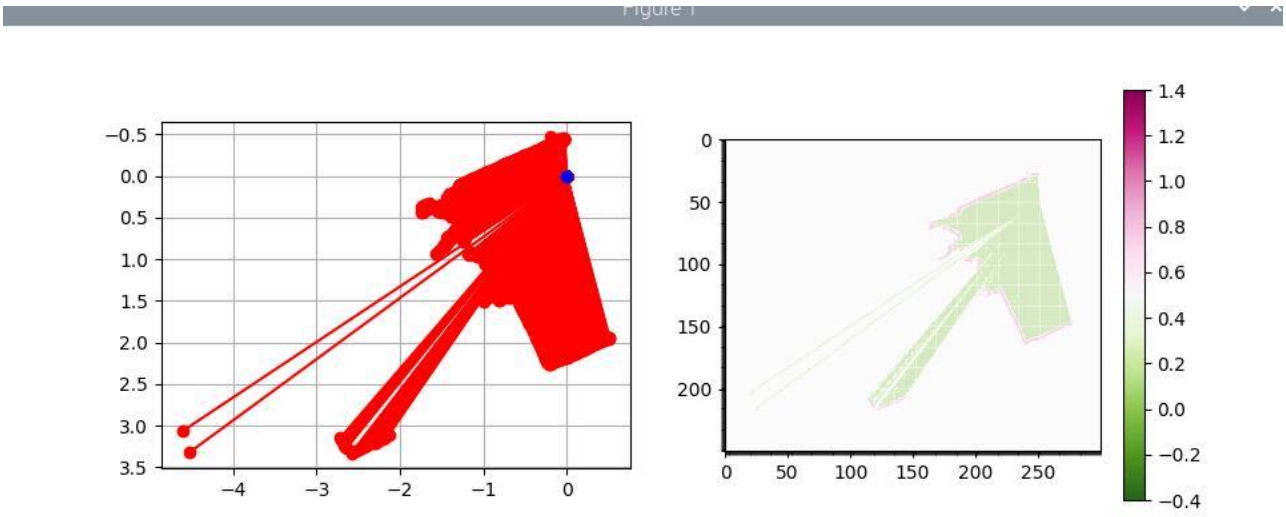
početne i krajnje točke također bile zamijenjene. Na taj način su dobivene točke koje je trebalo jednostavno povezati. U sekvenci Program 7 je prikazan dio programskog koda koji omogućuje stvaranje točaka na temelju Bresenhamovog algoritma:

```
def bresenham(start, end):

    # postavljanje početnih uvjeta
    x1, y1 = start
    x2, y2 = end
    dx = x2 - x1
    dy = y2 - y1
    is_steep = abs(dy) > abs(dx)  # određivanje strmine linije
    if is_steep:  # rotiranje linije ako je strma
        x1, y1 = y1, x1
        x2, y2 = y2, x2

    # zamjena početnih i konačnih točaka
    swapped = False
    if x1 > x2:
        x1, x2 = x2, x1
        y1, y2 = y2, y1
        swapped = True
    dx = x2 - x1
    dy = y2 - y1
    error = int(dx / 2.0)  # izračun greške
    y_step = 1 if y1 < y2 else -1

    # generiranje točaka između početnih i konačnih točaka
    y = y1
    points = []
    for x in range(x1, x2 + 1):
        coord = [y, x] if is_steep else (x, y)
        points.append(coord)
        error -= abs(dy)
        if error < 0:
            y += y_step
            error += dx
    if swapped:  # zamjena generiranih točaka
        points.reverse()
    points = np.array(points)
    return points
```

Program 14 Bresenhamov algoritam

Osim što je nacrtana mapa koja prikazuje laserske zrake LIDAR senzora, nacrtana je i mapa koja mnogo jasnije prikazuje prepreke. Površina koja je slobodna za kretanje, na kojoj nema prepreka na putu, obojana je zelenom bojom. Zidovi i ostale prepreke koje postoje na putu, prikazane su ljubičastom bojom kao što se može vidjeti na desnoj dijelu slike 4.7. Također, kako bi crtanje tih mapa bilo omogućeno, trebalo je postaviti odgovarajuću rezoluciju. Ukoliko je postavljena premala rezolucija, crtanje nije bilo moguće jer je bilo

zadano previše točaka. Ako je s druge strane bila zadana velika rezolucija, tada je za crtanje bilo potrebno mnogo vremena.

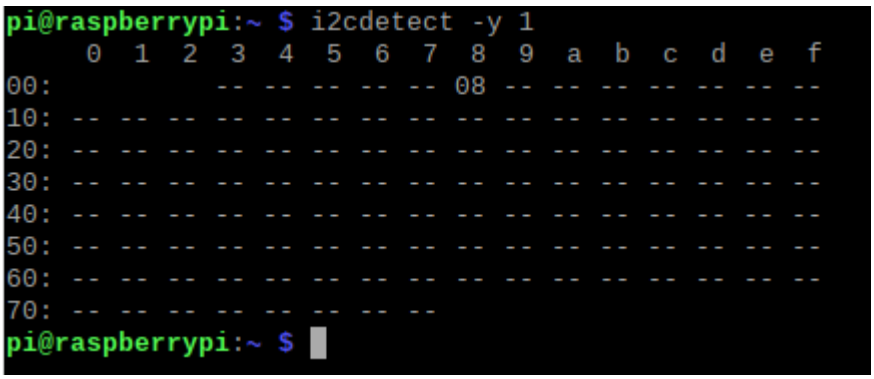


Slika 4.7 Mapiranje prostora

4.3. Upravljanje pogonskim elektromotorima mobilne platforme

Kao što je već prethodno navedeno, kako bi se podaci mogli prenositi s Raspberry Pi pločice na Arduino, potrebno je omogućiti komunikaciju između ta dva digitalna sustava. Komunikacija je omogućena korištenjem I2C protokola. Za uspostavljanje I2C komunikacije bilo je potrebno povezati tri fiksna pina na obje pločice. Prvi pin bio je SDA (Serial Data Line) te je bio vezan uz same podatke koji se prenose. Drugi pin bio je SCL (Serial Clock Line) te je služio sinkronizaciji vremena odnosno da se podaci prenose sa što manjim vremenskim kašnjenjem. Posljednji pin koji je trebalo povezati bio je GND (Ground), odnosno uzemljenje. Sva tri pina bila su definirana na obje pločice te je za njihovo povezivanje bio iskorišten Raspberry Pi GPIO adapter koji je omogućio jednostavnije spajanje i korištenje s eksperimentalnom pločicom.

Nakon fizičkog spajanja trebalo je konfigurirati sučelje Raspberry Pi-a tako da podrži I2C komunikaciju, a nakon toga se korištenjem naredbe *i2cdetect -y 1*, u terminalu Raspberry Pi-a, mogla uočiti adresa koja je korištena za komunikaciju i s jedne i s druge strane. Slika 4.8 prikazuje da se u ovom završnom radu koristila adresa 0x08, no mogla se definirati i neka druga adresa u rasponu od 0x03 do 0x77.



Slika 4.8 Adresa sabirnice za komunikaciju između Raspberry Pi i Arduina

Nakon što je uspostavljena komunikacija na Arduinu, trebalo je prihvatiti podatke koje Raspberry Pi šalje, a to je izvedeno korištenjem funkcije ***Wire.onReceive()*** koja je omogućila da se unutar nje aktivira dodatna funkcija kada preko sabirnice dolaze podaci. Upravo u toj dodatnoj funkciji je bio isprogramiran način upravljanja motorima u ovisnosti o podacima iz Raspberry Pi-a što je prikazano dijelom programskog koda Program 8.

```
void receiveEvent(int howMany)
{
	for (int i = 0; i < howMany; i++)
	{
	Tipka = Wire.read();
	if (Tipka == 56)
	{
		// Motor lijevo
		digitalWrite(in1, HIGH);
		digitalWrite(in2, LOW);

		// Motor desno
		digitalWrite(in3, HIGH);
		digitalWrite(in4, LOW);

		brzina_motor_1 = 250;    // Postavljanje brzine
		brzina_motor_2 = 250;
	}
```

Program 17 Funkcija u Arduinu koja čita podatke pristigle sa sabirnice

Funkcija ***receiveEvent()*** zaprima podatke sa sabirnice korištenjem predefinirane funkcije ***Wire.read()*** i sprema ih u varijablu *Tipka*. Zatim se provjerava koji podatak je u pitanju i ukoliko se radi o bajtu 56, koji označava da je pritisnuta tipka za smjer naprijed, tada se omogućava vožnja prema naprijed, na način da se pinovi za naprijed postave na visoku naponsku razinu, te se zadaje brzina. Unutar ovog dijela programskog koda prikazano je upravljanje samo za smjer naprijed, no identična situacija je i sa ostalim smjerovima, uz korigiranje brzine te naponskih razina na pinovima.

Za upravljanje su definirani digitalni pinovi kako je prikazano u dijelu programskog koda Program 9.

```
// Motor lijevo

  int en_lijevo = 10;      //omogućavanje lijevog motora
  int in1 = 9;             //vrtnja naprijed
  int in2 = 8;             //vrtnja natrag

// Motor desno

  int en_desno = 6;        //omogućavanje desnog motora
  int in3 = 5;             //vrtnja naprijed
  int in4 = 4;             //vrtnja natrag
```

Program 19 Inicijalizacija digitalnih pinova na Arduinu

Kod vrlo niskih brzina vrtnje motora, javljao se šum motora pa je to riješeno na način da su se vrlo male brzine izjednačile s nulom kako bi se eliminirao taj problem što prikazuje i dio programskog koda Program 10.

```
if (brzina_motor_1 < 8)brzina_motor_1 = 0;
if (brzina_motor_2 < 8)brzina_motor_2 = 0;
```

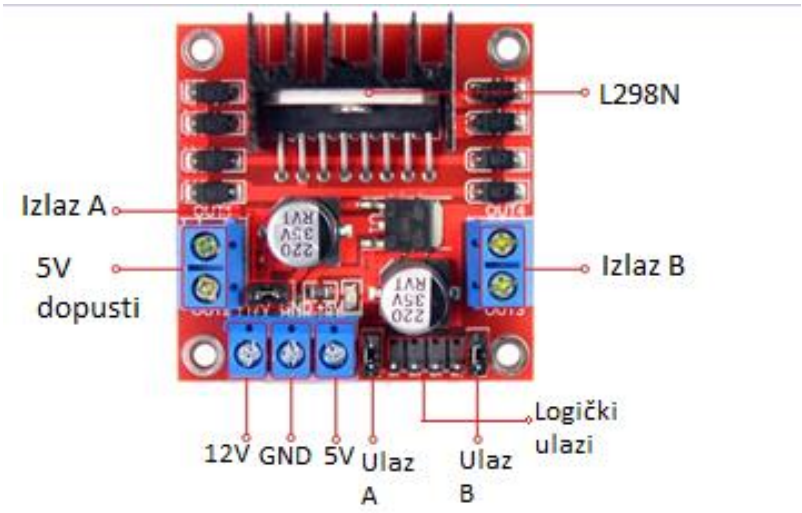
Program 20 Sprječavanje šuma motora

Kako bi se brzina zaista mogla prenijeti na motore, iskorištena je funkcija ***analogWrite()*** koja zapisuje analognu vrijednost na pinove Arduina (pin 10, pin 6), pri čemu ti pinovi obavezno moraju podržavati PWM jer se korištenjem PWM-a upravlja izlaznim naponom tranzistorskog mosta L298N i brzinom vrtnje motora. Dio programskog koda koji to omogućuje prikazan je u sekvenci Program 11.

```
analogWrite(en_lijevo, brzina_motor_1);
analogWrite(en_desno, brzina_motor_2);
```

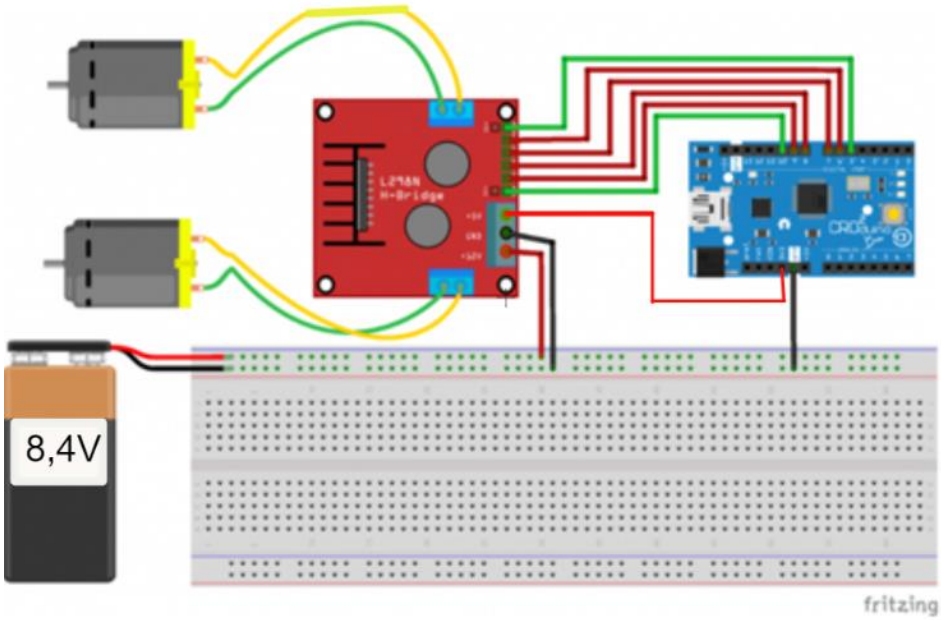
Program 22 Funkcija koja daje analogni podatak na pinove Arduina

Nakon što su podaci pripremljeni za slanje na motor, potrebno je omogućiti da zaista stignu do motora pa je za tu svrhu iskorišten tranzistorski most L298N, često zvani i H-most. Razlog, zbog kojeg se koristi most, leži u tome da Arduino ne može osigurati dovoljno izlazne snage za pokretanje motora.



Slika 4.9 H-most L298N [14]

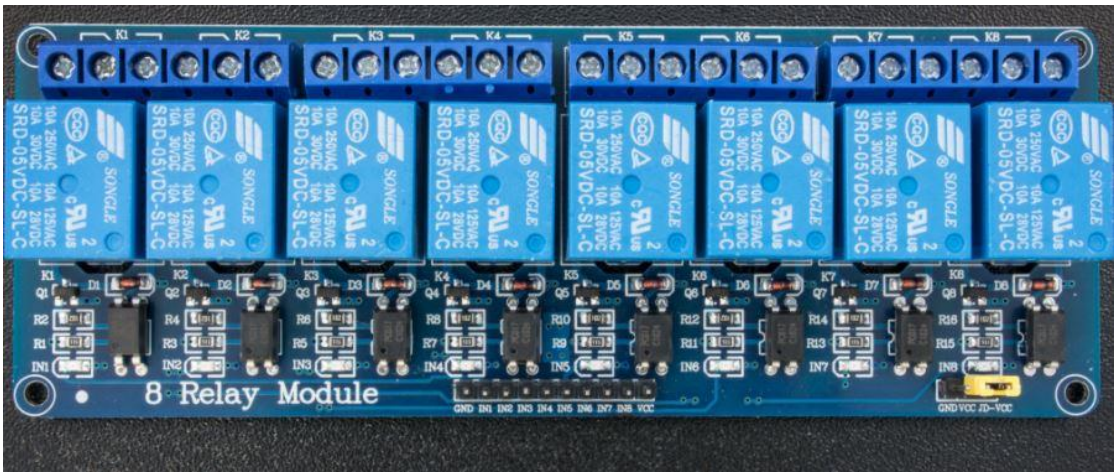
Za povezivanje motora bili su iskorišteni priključci označeni sa Izlaz A i Izlaz B (slika 4.9) dok se za napajanje motora koristio priključak 12 V koji je predstavljao ulaz vanjskog izvora napajanja. Napajanje, označeno s 5 V, predstavljalo je ulaz u koji ide vanjski izvor napajanja od 5 V, u ovom slučaju napajanje s Arduina. Pinovi označeni s Ulaz A i Ulaz B služili su za kontrolu brzine vrtnje motora A, odnosno motora B, korištenjem PWM. Logički ulazi koristili su se za prenošenje binarne logike (0 ili 1) u ovisnosti o tome u koju stranu je trebalo okretati motor. Također, obavezno je trebalo ostaviti *jumper* koji je označen s *5 V dopusti* jer se koristi napajanje motora koje je veće od 7 V. Princip povezivanja Arduina s L298N mostom i s motorima, prikazan je na slici 4.10.



Slika 4.10 Primjer povezivanja mosta L298N [15]

4.4. Postavljanje relejnog modula

Nakon što je bilo izvedeno upravljanje te nakon što su motori radili sukladno programskim naredbama u Arduinu, moglo se prijeći na posljednji korak. Ideja u ovom koraku bila je postaviti relejni modul, koji je u ovom slučaju sadržavao 8 releja što je značilo da može kontrolirati 8 trošila odjednom (slika 4.11). Naravno, u radu je trebalo upravljati sa samo dva trošila tako da su bila iskorištena dva releja.



Slika 4.11 Relejni modul s 8 releja

Iako se relejni modul koristi za upravljanje trošilima koja za svoj rad trebaju veliku struju ili veliki napon, ovdje je svakako iskorišten s obzirom da se radi o prototipu. Naime, u radu se koriste motori malih snaga koji za svoj rad trebaju napon od 3 V do 6 V, no kod realnog vozila koristili bi se motori puno većih snaga tako da bi u tim situacijama releji svakako bili korišteni.

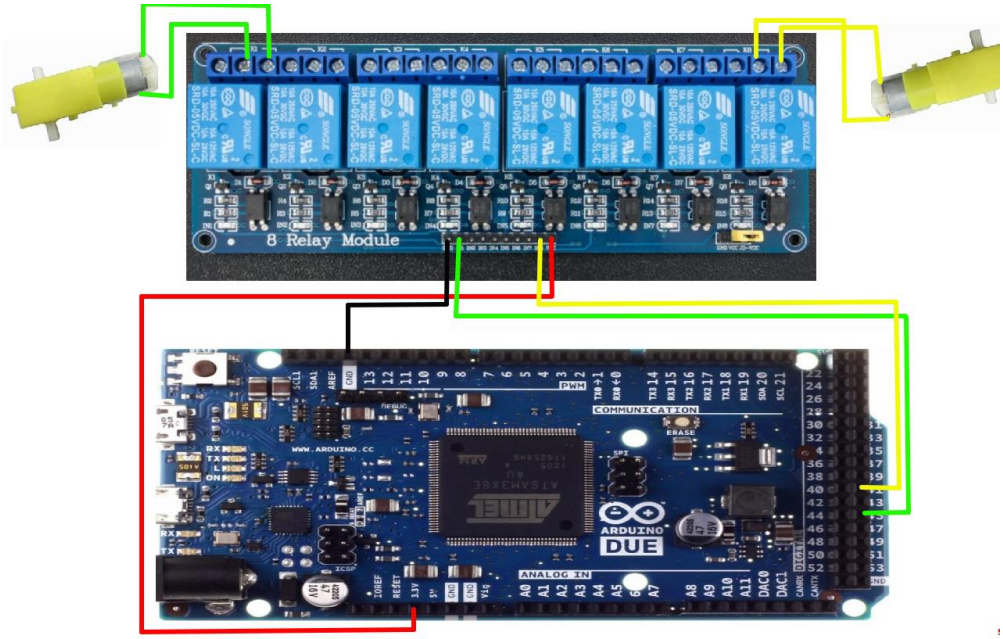
Kako bi mogli aktivirati relejni modul, potrebno mu je dati napajanje koje je bilo omogućeno korištenjem Arduina i njegovog pina koji daje napon od 3.3 V. Nakon toga je trebalo omogućiti releje koji se koriste, a to su bili relej 1 (krajnji lijevi) i relej 8 (krajnji desni), zbog lakšeg povezivanja na platformi. Relej 1 omogućavao je upravljanje lijevim motorima, a relej 2 upravljanje desnim motorima. Također je za upravljanje motorima trebalo povezati i izlazne kontakte releja koji su mogli biti NO (Normal open) ili NC (Normal closed). Korišteni su NO kontakti koji su bez prisustva napona bili u otvorenom stanju pa su i sami motori bili bez energije. Kada se doveo napon na relejni modul, NO kontakti su se zatvorili i struja je potekla do motora te su se mogli zavrtjeti u ovisnosti o tome kako je definirano programom. Dio programskog koda koji je omogućavao rad releja prikazan je u sekvenci Program 12.

```
digitalWrite(RELEJ_1,HIGH);        // Uključi relej lijevog motora
digitalWrite(RELEJ_2,HIGH);        // Uključi relej desnog motora
```

Program 23 Uključivanje releja

Dakle, kada se dovela visoka naponska razina na relej, zatvorio se strujni krug te se moglo upravljati motorima, dok kada je bila dovedena niska naponska razina, tada je strujni krug bio otvoren.

Način povezivanja relejnog modula koji je prije detaljno objašnjen, prikazan je na slici 4.12.

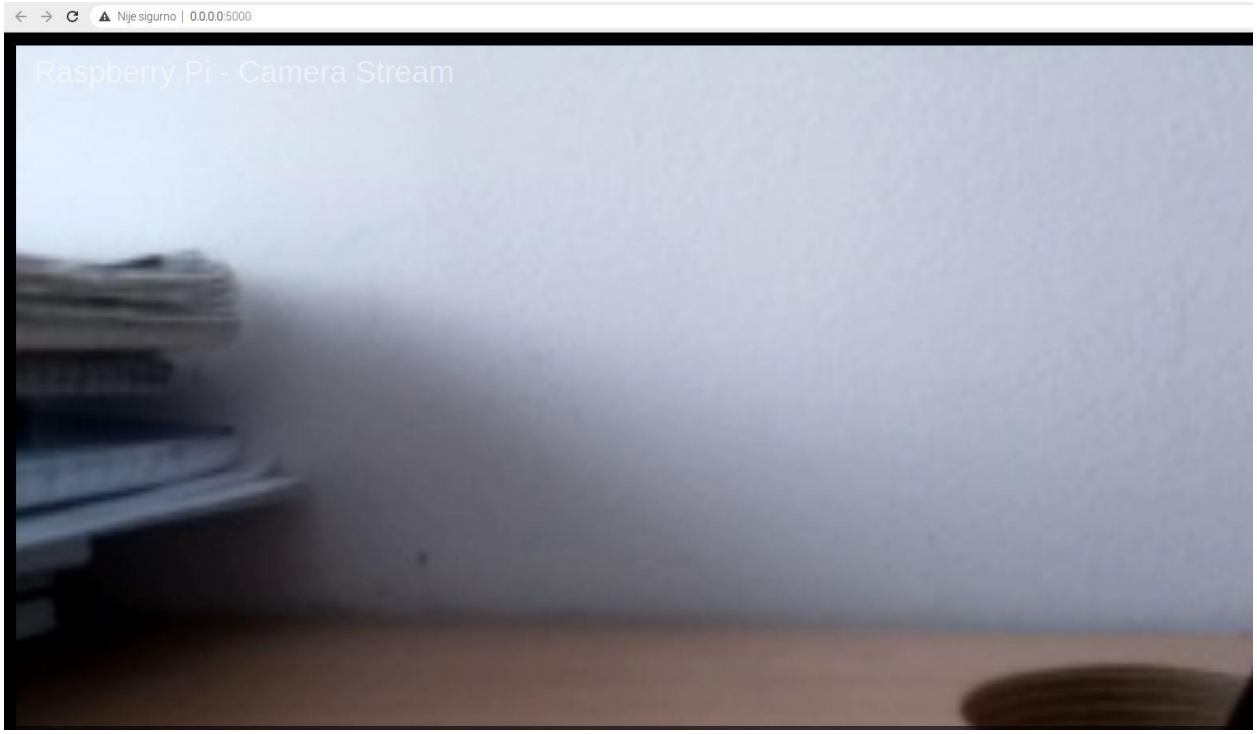


Slika 4.12 Povezivanje relejnog modula s Arduinom i motorima

4.5. Postavljanje Raspberry Pi kamere

Raspberry Pi kamera nema utjecaj na izvedbu upravljanja mobilnom platformom pa se ovaj dio smatra dodatkom, ali svejedno pruža korisnu funkcionalnost. Upotrebom kamere, koja je direktno priključena na Raspberry Pi pločicu, omogućeno je, na zaslonu bilo kojeg uređaja koji ima pristup WiFi mreži, iz prvog lica vidjeti što se nalazi ispred vozila. Iako Raspberry Pi podržava kameru trebalo ju je svakako uključiti u podešavanjima Raspbian operacijskog sustava. Obzirom da se Raspberry Pi pločici pristupa korištenjem WiFi veze, nakon učitavanja programskog koda, dobiva se IP adresa te port preko kojeg se može pristupiti video prijenosu te pratiti kretanje vozila, što može biti korisno ukoliko vozilo skrene u neku drugu prostoriju. Naravno, na video prijenos se istovremeno može spojiti i

preko računala i preko mobitela, ili nekog drugog uređaja koji ima pristup mreži. Na slici 4.13 prikazan je primjer kako izgleda video prijenos s Raspberry Pi kamere montirane na mobilnu platformu.

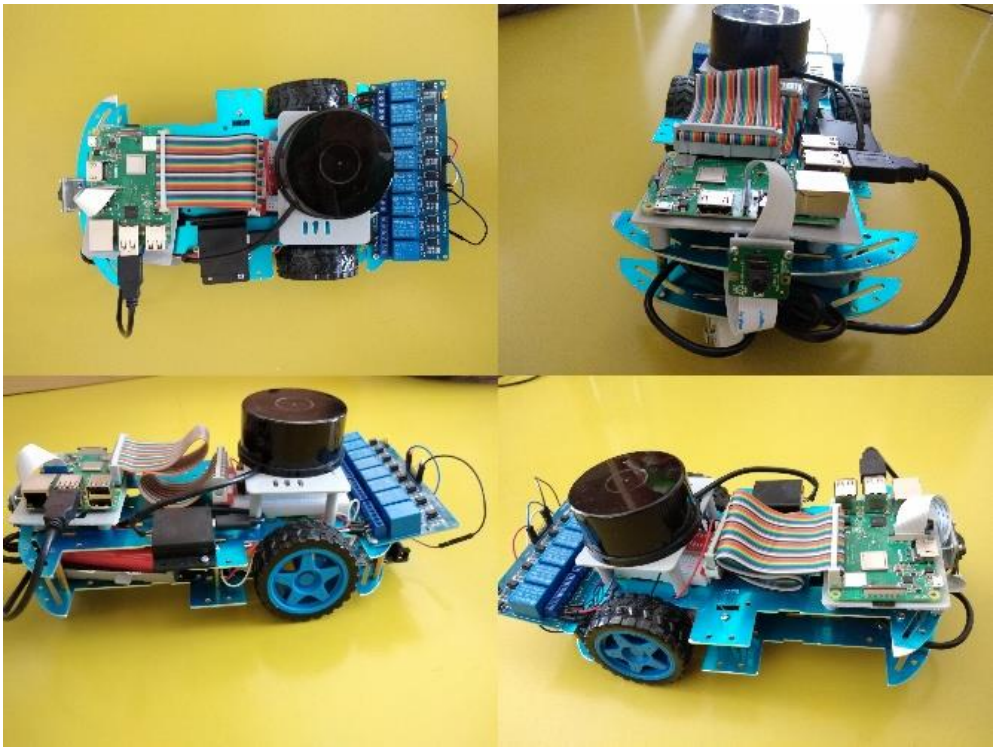


Slika 4.13 Video prijenos s Raspberry Pi kamere

5. ISPITIVANJE RADA MOBILNE PLATFORME

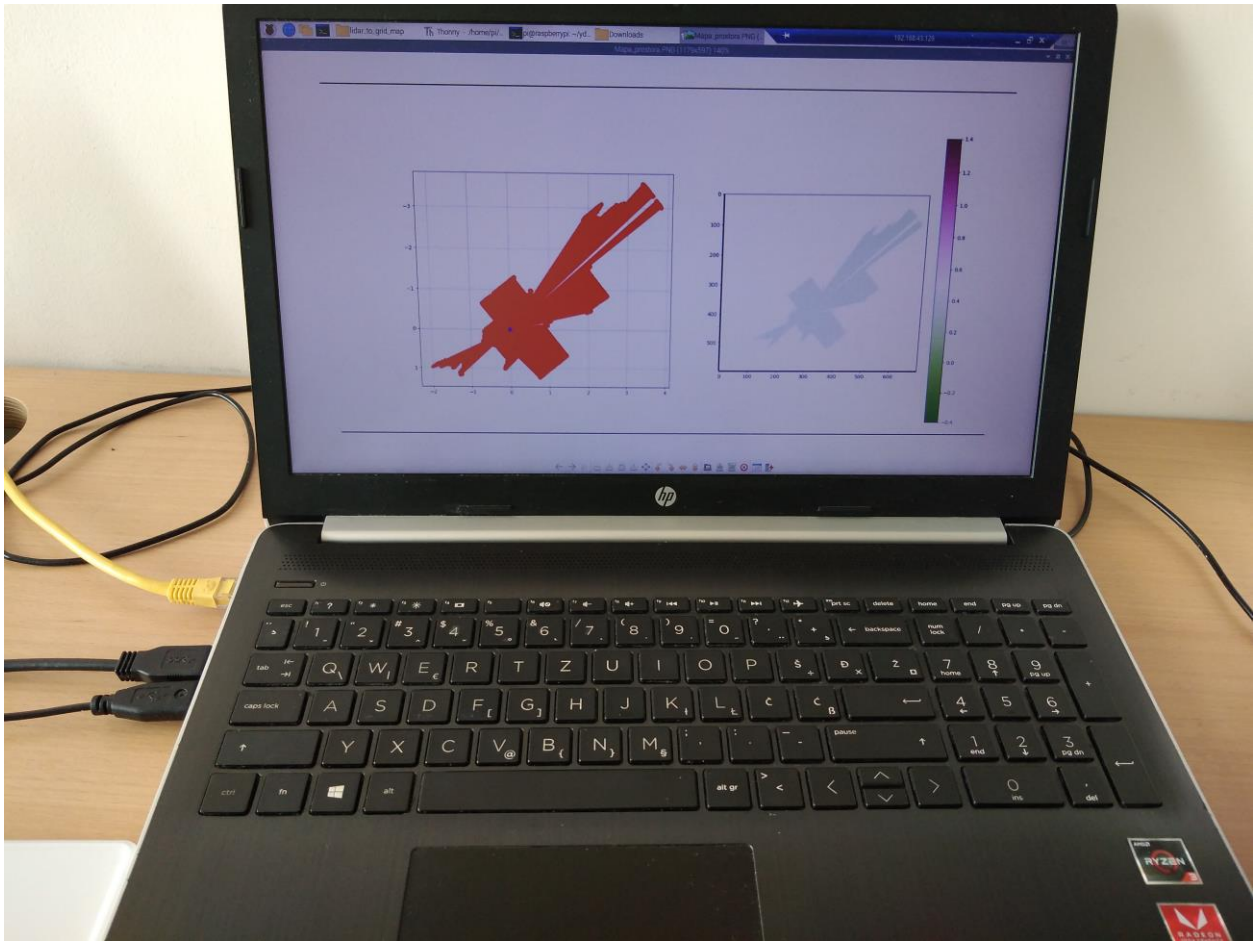
Ispitivanje rada mobilne platforme provodi se korištenjem prijenosnog računala i WiFi veze. Prilikom pritiska tipke na prijenosnom računalu, informacija o pritisnutoj tipki morala se prenijeti u Arduino, preko kojeg se vršilo upravljanje elektromotorima, kao što je opisano u poglavlju 4. Drugim riječima, ako se pritisne kursorska tipka sa smjerom prema gore, nakon što se unutar odgovarajućeg Python programa dozna koja tipka je pritisnuta i nakon što se informacija pošalje u Arduino, te nakon što informacija dođe do mosta L298N, tada se daje napajanje motorima da se počnu okretati u smjeru kazaljke na satu i mobilna platforma kreće prema naprijed. Nakon što se tipka otpusti, mobilna platforma se ne kreće. Ako se pritisne kursorska tipka sa smjerom lijevo, mobilna platforma počne skretati u lijevom smjeru na način da most L298N daje napajanje samo motoru desnog kotača, dok lijevi miruje. Ako se pritisne kursorska tipka za smjer desno tada mobilna platforma skreće u desnom smjeru na način da se motor lijevog kotača okreće prema naprijed, a motor desnog kotača miruje. Ako se pritisne tipka za smjer prema natrag tada se okreću elektromotori oba kotača, ali u suprotnom smjeru od kazaljke na satu, odnosno mobilna platforma se kreće unatrag.

Na slici 5.1 prikazano je kako izgleda mobilna platforma na kojoj se vrši ispitivanje preko udaljenog prijenosnog računala. Platforma omogućava razvoj SLAM algoritama za navigaciju i prikazana je iz različitih kutova.



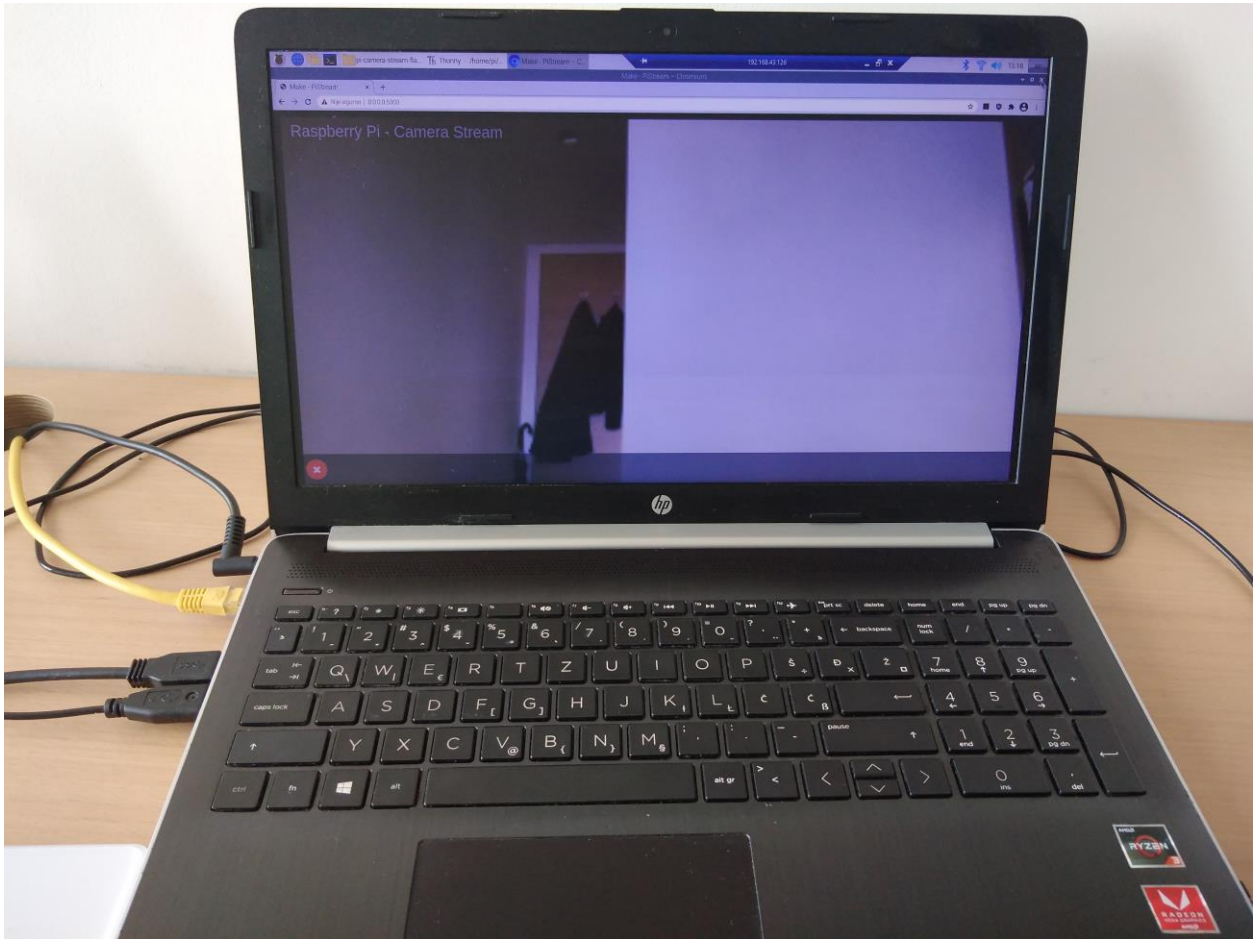
Slika 5.1 Prikaz mobilne platforme

Još jedna funkcionalnost koju je bilo potrebno ispitati bila je crtanje mape prostora, a provedena je na način kao što je opisano u poglavlju 4. Mapa prostora, koja se crta na prijenosnom računalu, nalazi se na slici 5.2. Osim mape prostora, na slici 5.2 vidi se i prijenosno računalo na kojem se sama mapa crta. Nakon što je nacrtana mapa prostora, mobilna platforma je spremna za razvoj navigacijskih algoritama.



Slika 5.2 Prijenosno računalo i mapa prostora

Raspberry Pi kamera prikazivala je okolinu na način kao što je prikazano na slici 5.3. Na toj slici može se vidjeti prijenosno računalo s kojeg se vrši upravljanje i slika s Raspberry Pi kamere.



Slika 5.3 Prijenosno računalo i slika s Raspberry Pi kamere

6. TROŠKOVI IZRADE MOBILNE PLATFORME

Ovo poglavlje ima zadatak prikazati gdje su kupljene određene komponente i koliko iznose cijene pojedinih komponenata koje su prikazane u tablici 1. Sve komponente nabavljene su po relativno jeftinim cijenama, no u tablici 1 se može primijetiti da LIDAR senzor ima uvjerljivo najveću cijenu što i ne čudi, obzirom na mogućnosti koje pruža, a opisane su u poglavlju 3.

Sve komponente korištene u radu nabavljene su online putem. Najviše je bila korištena web trgovina pod nazivom *e-radionica.com* [27] koja je pružala mnogo tutorijala, uputa i detalja kako koristiti njihove proizvode. Dane su slike, primjeri kodova, a za neke složenije elemente čak i videozapisi, dostupni na njihovim web stranicama. Dijelovi kupljeni u navedenoj web trgovini su aluminijska platforma s 4 kotača, 8-kanalni modul s relejom, tranzistorski most L298N, Raspbery Pi GPIO adapter i spojni elementi poput kablića za povezivanje te eksperimentalne pločice.

Druga web trgovina koja se koristila za nabavu dijelova je *Chipoteka* [28] koja je pokazala veoma široku ponudu i najveću isplativost kod nabave dijelova kao što su Raspberry Pi model 3B+, zatim Raspberry Pi kamera i Arduino Due.

Treća web trgovina korištena za nabavu je *www.elektor.com* [29] koja potječe iz Nizozemske i služila je za nabavu LIDAR senzora.

Ukupna cijena svih dijelova korištenih za izradu završnog rada iznosila je otprilike 4000 kn što se može vidjeti u zadnjem retku tablice 1.

NAZIV ELEMENTA	**CIJENA (kn)**
YDLIDAR TG15	2800
Aluminijska platforma s 4 kotača za robotiku	159
Modul s relejom 8-kanalni	105
DC motor driver dual H-most L298N	45
Raspberry Pi 3B+	400
Raspberry Pi kamera	100
Arduino Due	360
Raspberry Pi GPIO adapter	20
Spojni elementi	20
UKUPNO	**4000**

Tablica 1 Troškovi pojedinih dijelova mobilne platforme

7. ZAKLJUČAK

Korištenje LIDAR senzora, kao ključnog elementa za skeniranje prepreka, pokazalo se veoma korisnim u ovom završnom radu. Za razliku od ultrazvučnih senzora, koji se dosta često koriste kod autonomnih vozila, LIDAR senzor sasvim sigurno pruža veću preciznost zbog korištenja lasera.

Mapa prostora izgledala je dosta jasno i moglo se jasno primijetiti gdje se u prostoru nalaze prepreke. Također, crtanje same mape prostora potrajalo je prilično dugo zbog velikog broja točaka, no ukoliko se želi dobiti preciznost, bilo je potrebno uzeti što veći broj točaka.

Implementacija upravljanja mobilnom platformom, korištenjem kursorskih tipki i WiFi veze, dokazala je na koji način teoretske stvari spojiti s praksom te je uveliko doprinijela iskustvu pri radu s Python programskim jezikom i samom načinu upravljanja robotskim vozilom. Prilikom prijenosa podataka s Raspberry Pi pločice sabirnicom na Arduino moglo se uočiti da komunikacija neće uvijek biti uspješno provedena ako se istovremeno pritisnu dvije tipke. Na temelju toga moglo se zaključiti da je potrebno dodati malo kašnjenje između slanja naredbi preko sabirnice. Problemi su postojali i kod pogona kotača mobilne platforme jer su elektromotori bili napajani premalim naponom (6 V), obzirom da je dolazilo do pada napona unutar mosta L298N, moglo se zaključiti da takvo napajanje neće biti dovoljno za pokretanje platforme. Nakon toga je uzeto napajanje od 8 V.

Obzirom da su navigacijski algoritmi temelj na kojima se baziraju današnja autonomna vozila, dolazi se do zaključka da je potrebno veliko znanje i upornost prilikom njihove implementacije. Samim time što bi se, između ostalog, trebali koristiti u svakodnevnom prometu dovoljno govori o tome da nakon što se algoritmi počnu koristiti u praksi, više nemaju prava na pogrešku. Korištenjem LIDAR senzora pogreške bi se dodatno trebale smanjiti i omogućiti sigurnije upravljanje autonomnim vozilima, no isto tako treba trenutno obrađivati veoma veliku količinu podataka koji dolaze sa senzora.

Komentar na kraju ovog završnog rada jest da je rad s navigacijom bilo ogromno pozitivno iskustvo koje je isto tako otvorilo mnogo dodatnih područja koja je još potrebno proučiti. Opisana, relativno jeftina mobilna platforma će sasvim sigurno korisno poslužiti za daljnja istraživanja i razvoj navigacijskih algoritama.

Sveučilište
Sjever

SVEUČILIŠTE
SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, LUKA BARKOVIĆ (ime i prezime) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ završnog/ (obrisati nepotrebno) rada pod naslovom IZRADA JEFTINE MOBILNE PLATFORME ZA RAZVOJ NAVIGACIJSKIH ALGORITAMA (upisati naslov) te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)
Luka Barković
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, LUKA BARKOVIĆ (ime i prezime) neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/ (obrisati nepotrebno) rada pod naslovom IZRADA JEFTINE MOBILNE PLATFORME ZA RAZVOJ NAVIGACIJSKIH ALGORITAMA (upisati naslov) čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)
Luka Barković
(vlastoručni potpis)
(vlastoručni potpis)

8. LITERATURA

[1] *An Environmental Potential Field Based RRT Algorithm for UAV Path Planning.* Yang, Hongji, Jia, Qinghzong i Zhang, Weizhong. Beijing : IEEE, 2018. 2018 37th Chinese Control Conference (CCC). str. 6.

[2] *Summary of AGV Path Planning.* Wang, Chen i Mao, Jian. Shanghai : IEEE, 2019. International Conference on Electronic Information Technology and Computer Engineering (EITCE). str. 4.

[3] *Real-time loop closure in 2 d LIDAR SLAM.* Hess, Wolfgang, i dr. Stockholm : IEEE, 2016. Proc. IEEE Int. Conf. Robot. Automat. str. 8.

[4] *LEGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain.* Shan, Tixiao i Englot, Brendan. Madrid : IEEE, 2018. Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. str. 8.

[5] *Multilayer lidar 's environmental sensing in autonomous vehicle.* Duan, Jianmin, Zheng, Kaihua i Zhou, Junjing. Beijing : IEEE, 2014. Journal of Beijing University of Technology.

[6] YDLIDAR. YDLIDAR-Downloads & Support. *YDLIDAR.* [Mrežno] 2020. https://www.ydlidar.com/Public/upload/files/2020-05-27/YDLIDAR%20TG%20Series%20Development%20Manual.pdf.

[7] Banner test lidars robosense. *Blog generation robots.* [Mrežno] 15. Listopad 2019. https://blog.generationrobots.com/en/etude-experimentale-du-lidar-robosense-rs-lidar16/banner-test-lidars-robosense/.

[8] Schweber, Bill. LIDAR and Time of Flight, Part 2: Operation. *MicrocontrollerTips.* [Mrežno] 13. Prosinac 2019. https://www.microcontrollertips.com/lidar-and-time-of-flight-part-2-operation/.

[9] *The difference between multi-line lidar and single-line lidar.* Shanghai : Slamtec, 2019.

[10] *A Comparative Analysis of LiDAR SLAM-Based Indoor Navigation for Autonomous Vehicles.* Zou, Qin, i dr. Shenzhen : IEEE, 2021. 1558-0016.

[11]Puttur, Kiran. EKF-Term2. *Github.* [Mrežno] 21. Kolovoz 2018. https://github.com/kputtur/EKF-Term2.

[12] *A Monocular Vision Sensor-Based Efficient SLAM Method for Indoor Service Robots.* Lee, Taejae, Kim, Chulhong i Cho, Dong-il Dan. s.l. : IEEE, 2018. 0278-0046.

[13] *Intelligent RRT Exploration Mapping Method Based on Evolutionary Cognition in Unknown Environment.* Chen, Xiao, Ruan, Xiaogang i Zhu, Xiaoqing. Chongqing : IEEE, 2020. 978-1-7281-5244-8.

[14] Bjažić, Toni. *Upravljanje istosmjernim motorom.* [Dokument] Zagreb : arm MBED, 2020.

[15] E-radionica. KKM: DC MOTOR DRIVER DUAL H-BRIDGE. *e-radionica.* [Mrežno] 2020. [Citirano: 15.. travanj 2021.] https://e-radionica.com/hr/blog/2017/12/10/kkm-dc-motor-driver-dual-h-bridge/.

[16] sijah_ak. Self Driving Car Using Arduino (autonomous Guided Vechicle). *instructables circuits.* [Mrežno] instructables, 2016. https://www.instructables.com/Self-Driving-Car-Using-Arduinoautonomous-Guided-Ve/.

[17] What is LiDAR and how does it work? *GeoSLAM.* [Mrežno] 15. ožujak 2021. https://geoslam.com/what-is-lidar/.

[18] waveform80, i dr. *Importing GPIO Zero.* [Dokument] Cambridge : gpiozero, 2021.

[19] Kangutkar , Rasika M. Obstacle Avoidance and Path Planning for Smart Indoor Agents . *Scholarworks.rit.edu.* [Mrežno] 23. srpanj 2017. [Citirano: 18. 5 2021.] https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=10672&context=theses.

[20] *2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots.* Zhang, Xuexi, i dr. London : Hindawi, 2020.

[21] *SLAM in Dynamic Environments: A Deep Learning Approach for Moving Object Tracking Using ML-RANSAC Algorithm.* Bahraini, Masoud S., Rad, Ahmad B. i Bozorg, Mohammad. Basel : MDPI, 2019. 1424-8220.

[22] Wang, Xuan. 2D Mapping Solutions for Low Cost Mobile Robot. *diva-portal.* [Mrežno] 2013. [Citirano: 23. travanj 2021.] https://www.diva-portal.org/smash/get/diva2:680998/FULLTEXT01.pdf. 1653-5715.

[23] Gopalan, Anand. Leading lidar technology. *Velodyne Lidar.* [Mrežno] Velodyne, 1983. [Citirano: 17. ožujak 2021.] https://velodynelidar.com/.

[24] Kennedy, Kevin J. LiDAR Products. *Quanergy.* [Mrežno] Quanergy. [Citirano: 17. travanj 2021.] https://quanergy.com/.

[25] Chunxin, Qiu. Smart LiDAR Sensor. *Robosense.* [Mrežno] 2014. [Citirano: 17. travanj 2021.] https://www.robosense.ai/en/index/index.

[26] Scan Conversion a line. *Bresenham's Line Algorithm.* s.l. : Javatpoint. https://www.javatpoint.com/computer-graphics-bresenhams-line-algorithm.

[27] e-radionica. *e-radionica.* [Mrežno] 2012. [Citirano: 17. travanj 2021.] https://e-radionica.com/hr/.

[28] O chipoteci. *Chipoteka.* [Mrežno] Z-el d.o.o., 1990. [Citirano: 17. travanj 2021.] https://www.chipoteka.hr/.

[29] About Elektor. *elektor.* [Mrežno] Elektor International Media, 1960. [Citirano: 15.. ožujak 2021.] https://www.elektor.com/. 1757-0875.

[30] Hattersley, Lucy. Raspberry Pi 3B+ Specs and Benchmarks. *TheMagPi.* [Mrežno] 2018. [Citirano: 5. svibanj 2021.] https://magpi.raspberrypi.org/articles/raspberry-pi-3bplus-specs-benchmarks.

[31] ARDUINO DUE. *Store Arduino.* [Mrežno] [Citirano: 5. svibanj 2021.] https://store.arduino.cc/arduino-due.

Popis slika

Popis programskih kodova

Popis tablica