Ultrazvučni detektor prepreka mobilnog robota

Smolec, Dominik

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Dominik Smolec

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

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

Student:

Dominik Smolec

Zagreb, 2022.

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

Zahvaljujem se profesoru Mladenu Crnekoviću na pomoći, suradnji i podršci prilikom izrade ovog završnog rada.

Dominik Smolec

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Dominik Smolec** JMBAG: **0035219152**

Naslov rada na hrvatskom jeziku: **Ultrazvučni detektor prepreka mobilnog robota**

Naslov rada na engleskom jeziku: **Ultrasonic obstacle detector for mobile robot**

Opis zadatka:

Gibanje mobilnog robota ovisno je o njegovoj percepciji okoline u kojoj se kreće. Percepcija okoline robota ostvaruje se odgovarajućim senzorima i obradom signala prema očekivanom modelu okoline. Ako je upravljački sustav manje procesorske moći (npr. 8 bitni kontroler) obrada velike količine podataka (npr. signal s kamere) nije moguća. Zato se koriste jednostavniji senzori kao što su laserski, ultrazvučni i infracrveni.

Od ultrazvučnog daljinomjera potrebno je konstruirati samostalni snimač okoline (engl. scaner) koji bi obuhvaćao kut od 180°, a podatke prenosio nadređenom računalu. Konstrukcija snimača treba biti takva da se može montirati na mobilnog robota.

U radu je potrebno:

- projektirati model ultrazvučnog snimača,
- odabrati i obrazložiti komponente snimača,
- napisati program za upravljanje snimača i prikaz rezultata,
- procijeniti vrijednost uređaja.

Potrebno je navesti korištenu literaturu, te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
30. 11. 2021.	**1. rok:** 24. 2. 2022. **2. rok (izvanredni):** 6. 7. 2022. **3. rok:** 22. 9. 2022.	**1. rok:** 28. 2. – 4. 3. 2022. **2. rok (izvanredni):** 8. 7. 2022. **3. rok:** 26. 9. – 30. 9. 2022.

Zadatak zadao:
Prof. dr. sc. Mladen Crneković

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
s	m	Udaljenost detektiranog predmeta
v_{zvuka}	m/s	Brzina zvuka u zraku
t	s	Trajanje putovanja ultrazvučnog signala
V_{ul}	V	Ulazni napon
V_{izl}	V	Izlazni napon
R_i	Ω	Električni otpor
I_{uk}	A	Ukupna struja uređaja
f_{kv}	Hz	Frekvencija kristalnog oscilatora

SAŽETAK

Za snalaženje mobilnih robota u prostoru potrebni su senzori te obrada signala koji dolaze s tih senzora. Senzori s velikom količinom podataka zahtijevaju i puno procesorske moći za obradu podataka. Ako nije potrebna detaljna slika okoliša robota, koriste se jednostavniji senzori poput laserskog, ultrazvučnog i infracrvenog.

U ovom radu projektiran je ultrazvučni detektor prepreka za mobilnog robota. Dva ultrazvučna senzora daljine su pričvršćeni na koračni motor. Podatke o udaljenosti koje šalju senzori obradi mikroupravljač, te ih šalje preko bluetooth-a na računalo gdje se ti podatci prikazuju. Odabrane komponente će biti detaljno opisane. U radu će biti prikazan i objašnjen kod na mikroupravljaču i na računalu.

Ključne riječi: mobilni robot, ultrazvučni senzor, mikroupravljač

SUMMARY

For mobile robots to be able to navigate through environment, they need sensors and data processing. Sensors which deliver large amounts of data require a lot of processing power. If the detailed picture of environment is not required, simpler sensors are used such as laser, ultrasonic and infrared sensors.

Ultrasonic obstacle detector for mobile robots is designed in this thesis. Two ultrasonic sensors are mounted on a stepper motor. Obstacle distance data is being processed by microcontroller, which sends the data over Bluetooth to a computer. Selected components will be described in detail. Microcontroller and computer code will be shown and explained in this thesis.

Key words: mobile robot, ultrasonic sensor, microcontroller

1. UVOD

U zadnjih nekoliko godina možemo vidjeti rast broja mobilnih robota na komercijalnom tržištu. Sve više i više domova ima svoje robote usisavače koji spadaju u kategoriju mobilnih robota. Mobilne robote možemo prepoznati po tome što:

- Može se kretati - kretanje može biti izvršeno na više načina. Može imati noge, kotače, gusjenice, …
- Senzori - svakom mobilnom robotu su potrebni senzori kako bi se snalazio u prostoru.
- Snalaženje - ukoliko senzorima vidi prepreku, može ju i zaobići
- Samostalnost - ne trebamo nadgledati robota dok radi svoj posao.
- Programiranje orijentirano zadatku - robotu možemo zadati zadatak i on će ga izvršiti, a ne da mu moramo objasniti kako napraviti zadatak.

Slika 1. Mobilni robot

Kako bi robot mogao izvršavati svoj zadatak treba imati sliku o okolini. U ovom radu, baviti ćemo se problemom orijentacije mobilnih robota u svojoj okolini. Kao medij kojim dobiva informacije o okruženju robot koristi senzore.

Neki od često korištenih senzora na mobilnim robotima su:

- Taktilni senzor
- Infracrveni senzor
- LiDAR
- Ultrazvučni senzor
- Kamera

Taktilni senzor od ovih je najjeftiniji, ali isto tako je i najneprecizniji. Taktilni senzor može samo detektirati prisutnost predmeta, znači predmeta ili ima ili nema. Također domet senzora je fizički ograničen ticalom koje senzor ima. Recimo popularni kućni roboti koji usisavaju i/ili peru pod imaju taktilni senzor sa prednje strane koji detektira zid.

Nešto napredniji senzori su LiDAR i kamera. Ovi senzori su skupi u odnosu na druge senzore, ali zato daju jako puno informacija o okolini robota. Puno informacija znači da je potrebno i puno procesorske moći. Često su korišteni kada su potrebna precizna mjerenja.

Na kraju nam ostaju infracrveni i ultrazvučni senzori. Pošto ćemo koristiti mikroupravljač relativno male procesorske moći, koristiti ćemo jedan od ova dva senzoraZa ovaj rad sam se odlučio na ultrazvučni senzor.

Zašto ultrazvučni senzor? Ultrazvučni senzori, za razliku od infracrvenih senzora, nemaju problema s bojama ili osvjetljenjem objekta koji se mjeri. Također nude manje minimalne blizine detekcije, te veći konus detekcije.

Pošto je u radu traženo da ultrazvučni snimač pokriva kut od 180 °, korištena su dva ultrazvučna senzora, zakrenuta za 90 °, tako da koračni motor koji pokreče konstrukciju za koju su oni vezani ne radi put od 180 °, nego samo od 90 °.

Također će u ovom radu biti dan prijedlog izgleda tiskane pločice.

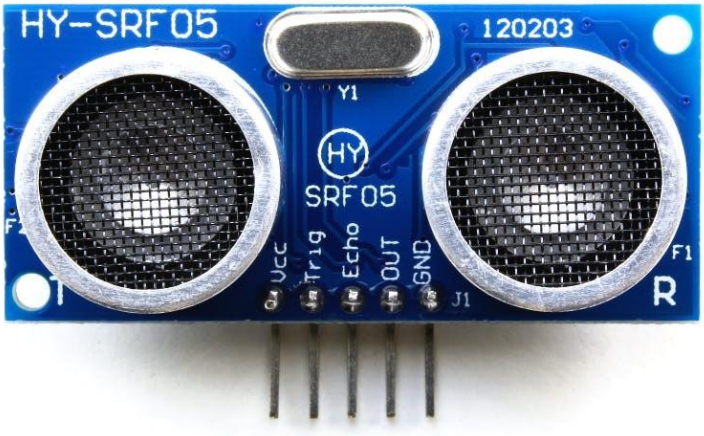
2. ODABRANE KOMPONENTE

2.1. Ultrazvučni senzor

Ultrazvučni senzor koji je odabran za ovu svrhu je HY-SRF05. Ovaj ultrazvučni senzor ima odvojeni prijemnik i odašiljač što mu daje mogućnost detekcije jako bliskih predmeta pošto nema potrebu mijenjanja između načina rada odašiljača i prijemnika. Ovaj model je unaprijeđena i jeftinija verzija starijeg modela HY-SRF04. Još neke prednost s obzirom na stariji model su bolja razlučivost, veći domet, te mogućnost slanja i primanja signala preko istog pina.

Osnovne značajke:

- Domet mjerenja: 2 – 400 cm
- Preciznost mjerenja: 3 mm
- Mjerni kut: 15°
- Maksimalna učestalost mjerenja: 40 Hz
- Napon napajanja: 4,5 – 5,5 V
- Nazivna struja: 4 mA



Slika 2. Ultrazvučni senzor HY-SRF05

Senzor ima 5 pin-ova::

- VCC - Napajanje senzora, spajamo na 5V
- Trig - Na ovaj pin šaljemo signal za početak mjerenja udaljenosti predmeta

- Echo - Na ovom pinu mjerimo vrijeme putovanja ultrazvučnog signala
- OUT - Ako je ovaj pin uzemljen omogućen je rad senzora preko jednog pina
- GND - Uzemljenje senzora

Senzor radi tako da se pošalje pozitivni signal na pin „Trig“ od 10 µs, senzor šalje u okoliš 8 pulsova ultrazvučnog signala na frekvenciji od 40 kHz. Nakon toga senzor čeka jedno kratko određeno vrijeme kako bi i sporiji mikro kontroleri mogli uloviti pozitivan brid signala na pinu „Echo“. Vrijeme trajanja pozitivnog signala na pinu „Echo“ je jednako vremenu putovanja signala kroz prostor. Brzina zvuka u zraku ovisi o:

1. Temperaturi zraka
2. Atmosferskom tlaku
3. Gustoći zraka
4. Vlažnosti zraka

Iako ima puno faktora brzine zvuka u zraku za potrebe ovog rada uzeti ću prosječnu brzinu zvuka koja će iznositi oko 343 m/s. Znači možemo zapisati jednostavnu jednadžbu:

$$udaljenost = brzina * vrijeme \tag{1}$$

Ali, pošto ultrazvučni signal putuje od senzora do predmeta, te nazad do senzora on prođe dvostruko veću udaljenost nego što je stvarna udaljenost između senzora i predmeta, iz tog razloga moramo jednadžbu podijeliti s dva.

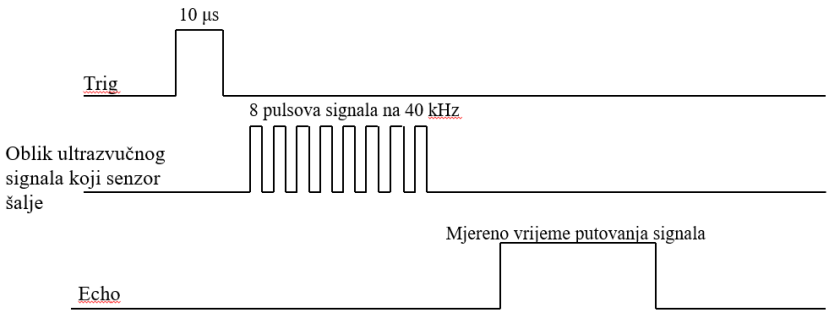
$$s = (v_{\text{zvuka}} * t)/2 \tag{2}$$

gdje je:

-s udaljenost senzora od predmeta

-v_{zvuka} prosječna brzina zvuka u zraku

-t izmjereno vrijeme putovanja signala



Slika 3. Skica dijagrama rada ultrazvučnog senzora

2.2. Mikroupravljač

Mikroupravljač je „mozak“ ovog rada. Služiti će nam za pokretanje senzora i motora, te obradu i slanje podataka. Postoji puno faktora na koje moramo obratiti pozornost oko izabira mikroupravljača koji će zadovoljiti sve naše potrebe. Neki faktori koji su meni bili važni pri odabiru mikroupravljača su:

- Radni napon - tražio sam mikroupravljač koji može raditi na napajanju od 5V, pošto i drugi uređaji koriste već taj napon.
- Brzina rada čipa - tražio sam mikroupravljač koji može raditi minimalno na 8 MHz, a sve više je još bolje
- Mogućnosti čipa - ovdje sam obavezno tražio da čip podržava vanjske prekide programa(Interrupt), da ima minimalno 2 Timera, te da ima mogućnost serijske komunikacije
- Memorija - tražio sam mikroupravljač sa minimalno 4 kB programibilnog prostora i minimalno 2 kB RAM-a.

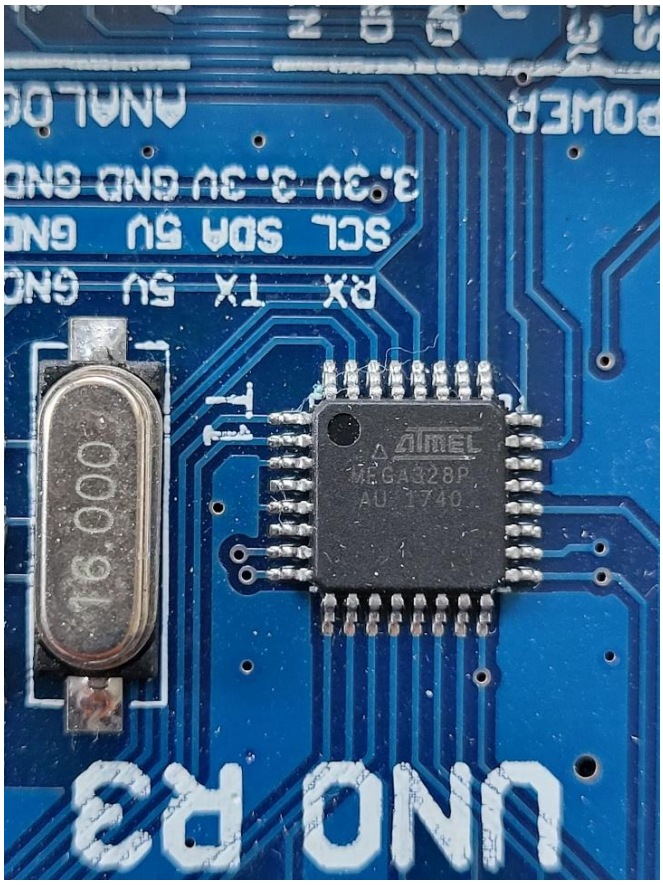
Zaključio sam, pošto će mikroupravljač imati samo jedan zadatak, 8- bitni kontroler bi trebao zadovoljiti sve ove uvjete. Čak i sa ovim ograničenjima koja sam postavio, svejedno postoji puno različitih modela mikroupravljača. Tako da sam se odlučio da ću za rad uzeti neki od mikroupravljača dostupnih na platformi arduino.

Za ovaj završni rad sam odabrao mikroupravljač Atmega328P na platformi „Arduino“. Arduino je poznat po svojoj jednostavnosti i intuitivnosti programiranja. Nedostatak kod Arduino platforme je taj što sastavljeni kod (compile) zauzima puno veći dio memorije nego što bi to zauzeo kod iz nekog drugog compiler -a. Mikroupravljač Atmega328P, koji se koristi u ovom radu, je jedan od najrasprostranjenijih čipova na Arduino platformi, i to s dobrim razlogom.

Osnovne značajke:

- 8-bitni mikroupravljač visokih performansi
- 23 programibilna ulazno/izlazna pina
- Dva 8-bitna i jedan 16-bitni Timer
- Osam 10-bitnih analogni digitalnih konvertera
- Šest PWM kanala

- Razne mogućnosti serijskih komunikacija
- Frekvencije rada do 16 MHz
- Napajanje: 2,7 - 5,5 V
- Nazivna struja pri 5V, 16 MHz i 25°C: 9mA



Slika 4. Atmega328P

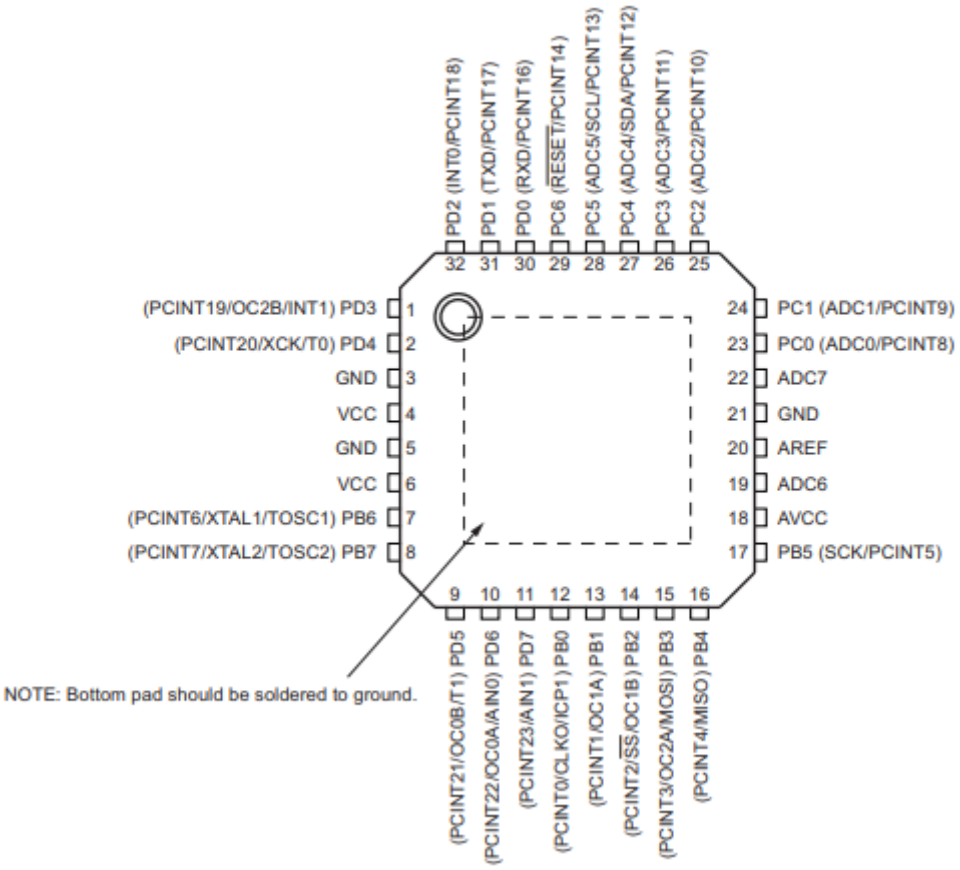
U ovom radu na mikroupravljaču je korišteno:

1. Digitalni ulazni/izlazi
2. Vanjski prekidi
3. Timer 1 i 2
4. UART serijska komunikacija

Na mikroupravljaču postoji nekoliko pinova koji su potrebni za normalan rad mikroupravljača.

1. Pin 4 i 6 – VCC - Napajanje mikroupravljača
2. Pin 3, 5 i 21 – GND - Uzemljenje mikroupravljača

3. Pin 7 i 8 – XTAL 1 i XTAL 2 - Pinovi za ulaz vanjskog davača takta
4. Pin 18 – AVCC - Napajanje za analogno digitalne konvertere. Ovaj pin se preporuča spojiti čak i ako se ADC ne koristi.
5. Pin 20 – AREF - Referentni napon ADC-a
6. Pin 29 – RESET - Kada je ovaj pin na niskom potencijalu (uzemljen) dulje od minimalnog dozvoljenog, mikroupravljač će se resetirati.

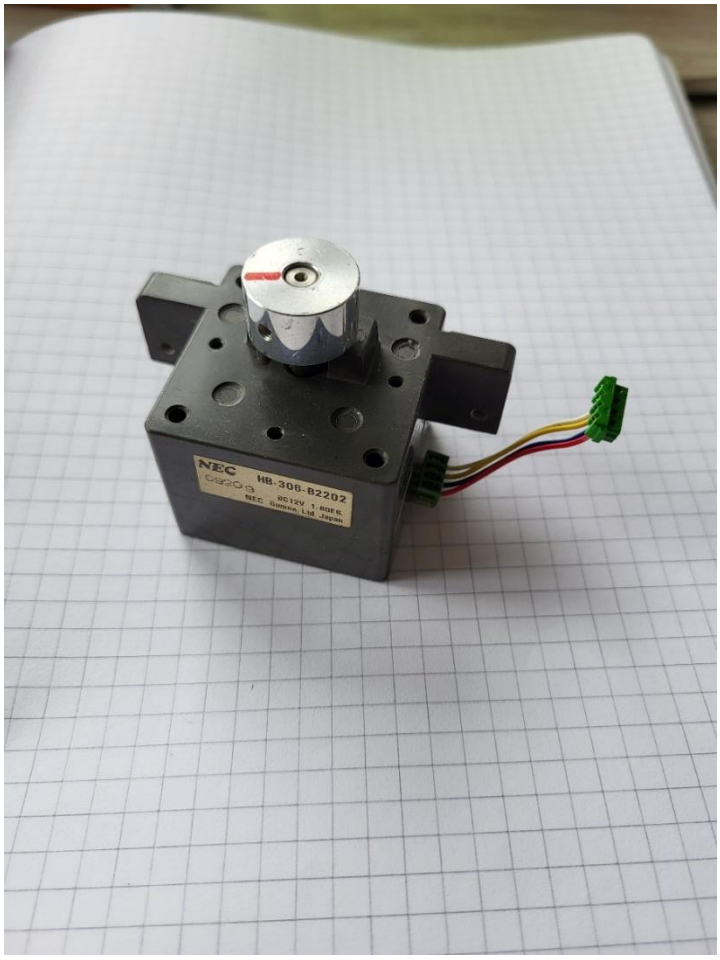


Slika 5. Raspored pinova mikroupravljača

2.3. Koračni motor i driver

2.3.1. Koračni motor

U radu je korišten koračni motor proizvođača NEC. Ovo je relativno star model motora, te nisam mogao puno informacija o motoru. Srećom, najvažnije informacije su otisnute na motoru, a to je da se jednim korakom zaokrene za 1,8°, te da mu je napajanje na 12 V. Odabrao sam koračni motor iz razloga što se relativno lagano upravljaju i imaju veliku preciznost pozicioniranja. Mane ovakvih motora su da nema povratnu informaciju o poziciji, te da ukoliko ne koristimo neki sustav povratne informacije o položaju motora, kao što je apsolutni enkoder, prilikom svakog pokretanja motor mora proći kroz proces inicijalizacije položaja.

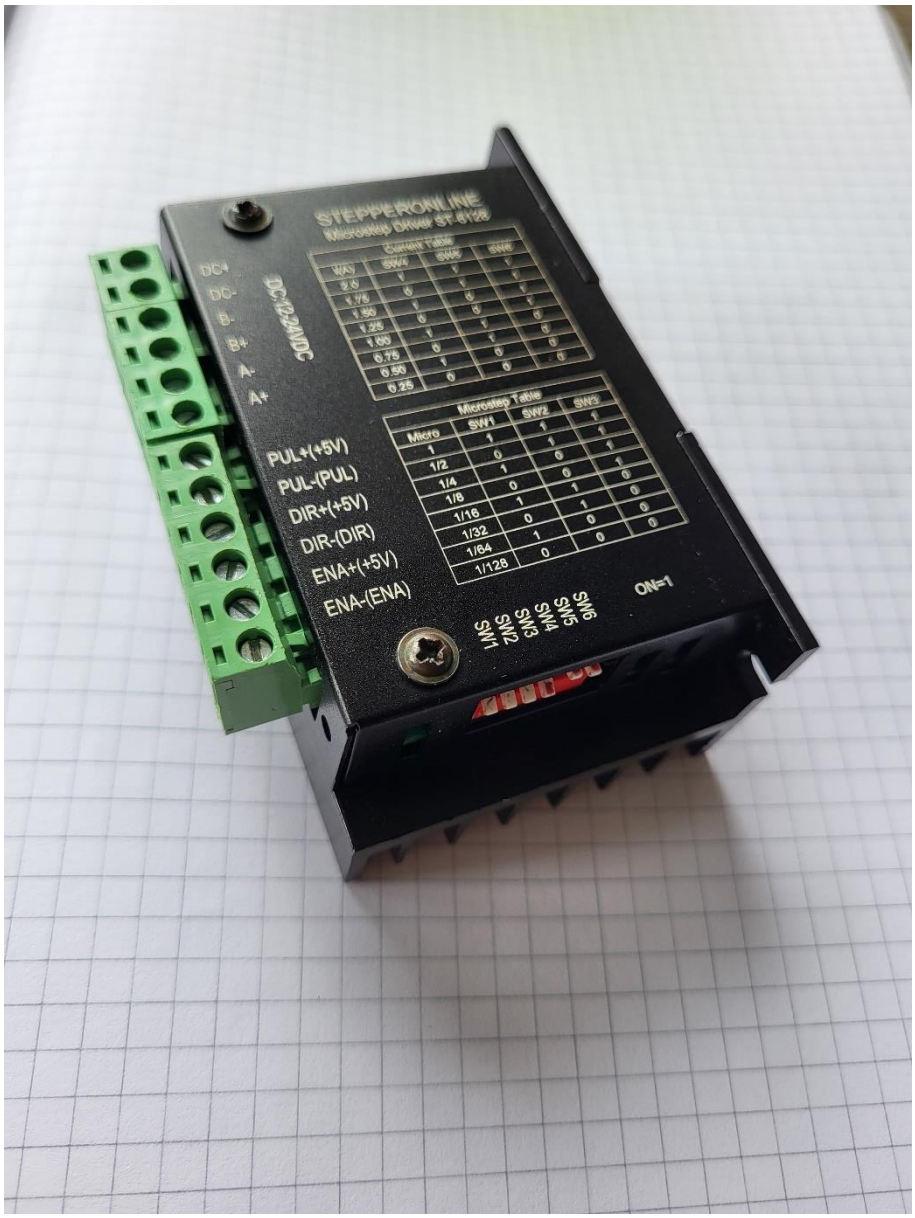


Slika 6. Koračni motor

Pošto motor vuče oko 100 mA, što je previše za jedan običan izlaz sa mikroupravljača, potrebno je koristiti driver koračnog motora.

2.3.2. Driver koračnog motora

Na slici [Slika 7] je driver koji sam koristio u radu. Relativno jednostavan, s puno mogućnosti što se tiče pokretanja koračnog motora. Driver može regulirati izlaznu struju u rasponu od 0,25 do 2 A. Također ima mogućnost mikrokoraka. Spajanje ovog drivera je relativno jednostavno, te ćemo to vidjeti u nekom od kasnijih poglavlja.



Slika 7. Driver koračnog motora

Pinovi na driveru su:

- DC+ - Pozitivnih 12-24V za napajanje drivera
- DC- - Uzemljenje
- A+, A-, B+, B- - Pinovi na koje se spaja motor kako bi ga upravljali
- PUL+ i PUL- - Kada se stvori razlika potencijala između ova dva pina motor se zakrene za jedan korak.
- DIR+ i DIR- - Kada su ovi pinovi na istom potencijalu, motor se vrti u jednu stranu, a kada su na različitom potencijalu, tj. između pinova je napon od 5V, motor se vrti u drugu stranu
- ENA+ i ENA- - Kada su ovi pinovi na istom potencijalu, motor ne radi i može se ručno okretati, a kada su na različitom potencijalu, motor radi i ukoliko ne dajemo pulseve na PUL pin motor čvrsto drži svoju poziciju

Također imamo mogućnosti pomoću malih prekidača na dnu drivera ograničiti struju, te mogućnost mikrokoraka. Struju sam ograničio na 0,25 A, pošto nam nije potrebna velika snaga, te ćemo koristiti motor u režimu rada punog koraka.

U praktičnoj izvedbi ovoga rada, nije nam potreban ovako veliki hladnjak drivera, pošto koračni motor savladava samo inercije malih predmeta, što i nije neko opterećenje za koje bi nam trebao veliki hladnjak. Tako da, ću ovdje dati prijedlog čipa za kontrolu koračnog motora

Odabrani čip je TMC2208-LA. Ovaj čip je jedan od popularnijih u svijetu 3d printanja. Ima opcije mikrokoraka, koje mi nećemo koristiti. Podržava struje i do 2 A, ali mi nećemo prelaziti preko pola ampera. Još jedno bitno svojstvo je da mu je napajanje od 5 do 12 V, što znači da ga možemo napajati isto kao i mikroupravljač. Podržava i upravljanje mikroupravljača putem serijske komunikacije, ali mi ćemo koristiti jednostavno davanje pulsa i smjera okretanja pomoću dvije žice.

2.4. Bluetooth modul

Za prijenos podataka o udaljenosti i poziciji detektiranog predmeta obrađenih na mikroupravljaču na nadređeno računalo koristio sam Bluetooth modul koji se zasniva na čipu

HC-05. Ovaj modul ima vrlo jednostavnu serijsku komunikaciju s mikroupravljačem putem TXD i RXD pinova.

Osnovne značajke:

- Napajanje: 3,3 – 6 V
- Napon logike čipa: 3,3 V
- Brzina komunikacije: 1 Mb/s
- Nazivna struja: 30 mA

Slika 8. Bluetooth modul

Pinovi na bluetooth modulu su:

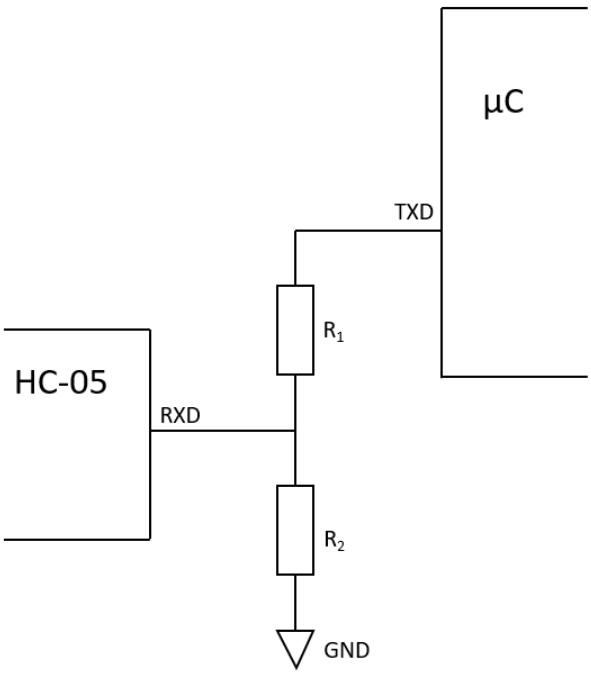
- Enable/Key - Ako je ovaj pin na niskom potencijalu onda je u modu prijenosa podataka, koji ćemo i mi koristiti, a ako je na visokom potencijalu onda modul ulazi u mod promjene postavki.
- VCC - Pozitivno napajanje(3,3-6V)
- GND - Uzemljenje
- TX - Pin na koji bluetooth šalje primljene podatke

- RX - Pin na koji bluetooth dobiva podatke koje mora poslati
- STATE - Pin na koji možemo spojiti LED da vidimo radi li modul ispravno

Kao što je i navedeno pod osnovnim značajkama, ovaj modul šalje i prima signale od 3,3 V, dok naš mikroupravljač šalje i prima signale od 5 V. Mikroupravljač nema problema s primanjem signala od 3,3 V, pošto i tu razinu napona prihvaća kao pozitivni signal. Ali s druge strane, da probamo Bluetooth modulu poslati signal od 5 V, vrlo vjerojatno bi se elektronika unutar njega spržila. Zato kada mikroupravljač šalje signal, taj signal prvo mora proći kroz neki strujni krug kako bi mu se snizio napon. Postoji puno boljih i efektivnijih rješenja, ali pošto su signali malog napona i struje, koristit ću naponsko dijelilo. Naponsko dijelilo je takav spoj otpornika gdje se napon raspodijeli na određene dijelove. Formula za izlazni napon je:

$$V_{izl} = \frac{V_{ul} * R_2}{(R_1 + R_2)} \tag{3}$$

Za nas slučaj vrijedi da je ulazni napon 5 V, a izlazni napon 3,3 V. Pošto nam nakon toga ostaju dvije nepoznanice, tj. dva otpora, a samo jedna jednadžba, jedan otpor se proizvoljno zada. Za otpor R_1 sam odabrao 1 kΩ. Tako da prema formuli otpor R_2 je jednak 2 kΩ. Shema spajanja izgleda ovako:[Slika 9]



Slika 9. Shema djelitelja napona

Ovaj modul ima mogućnosti komuniciranja različitim brzinama, tzv. „Baud rate“. Najčešća brzina u serijskoj komunikaciji u elektronici je 9600 bitova po sekundi, pa tako i ovaj Bluetooth modul po tvorničkim postavkama komunicira brzinom od 9600 bitova po sekundi. Ali, ovaj modul je također sposoban komunicirati i brzinama kao što su 19200 bps, 38400 bps, čak i do 460800 bitova po sekundi! Ali s većom brzinom prijenosa podataka dolazi i do mogućnosti pogreške u slanju, zato ćemo mi koristiti 9600 bps pošto je standard za siguran prijenos podataka.

2.5. Ostale komponente

2.5.1. Granični mikroprekidač

U svrhu inicijalizacije položaja kod svakog pokretanja uređaja korišteni su mikroprekidači na krajnjim pozicijama. Mikoprekidač koji sam ja koristio u ovom radu ima 3 pina: „Common“, „Normally Open“ i „Normally Closed“. Za potrebe ovog rada koristio sam samo „Common“ i „Normally Closed“.

Slika 10. Granični mikroprekidač

2.5.2. Baterije

Pošto će se uređaj nalaziti na mobilnom robotu, također mu treba i mobilni način napajanja. Za ovaj rad odlučio sam se na jednostavne alkalne baterije, kakve se mogu pronaći u raznim kućnim uređajima. Za prvu iteraciju uređaja sam koristio 9 V 6F22 bateriju. Ali kada sam spojio sve komponente, napon na bateriji bi drastično pao. Zato sam se odlučio na paket od 4 1,5 V LR6 baterija.

Slika 11. Baterija

Potrošnja sustava zavisi od svake pojedine komponente uređaja. Proizvođač komponente najčešće da okvirnu potrošnju te komponente, tako da jednostavno možemo procijeniti ukupnu potrošnju sustava:

- Mikroupravljač – mikroupravljač na 25 °C, pri radu od 8 MHz, prema dijagramima danim od proizvođača možemo zaključiti da mikroupravljač približno vuče **9** mA. Iako ovaj broj jako zavisi od toga što zapravo mikroupravljač radi, vidjeti ćemo da je on samo jedan mali potrošač u usporedbi sa drugim komponentama.
- Ultrazvučni senzor – pri brzini uzorkovanja od 40 Hz jedan ultrazvučni senzor provjereno vuče oko **4** mA. Pošto su dva senzora to ispada ukupno **8** mA
- Koračni motor – motori su najčešće prepoznati kao veliki trošioci u bilo kojem sustavu. Driverom koračnog motora ograničiti ćemo struju koju može povući motor na **250** mA.
- Bluetooth modul – iako pri maksimalnom iskorištenju Bluetooth modul može vuči i do 40 mA, mjerenjem struje tokom normalnog rada uređaja pokazalo se da HC-05 vuče samo oko **20** mA. Jedino obrazloženje za duplo manju struju može biti ne potpuno iskorištenje količine prijenosa podataka. Moj program šalje samo oko 700-800 bitova po sekundi.

Kada zbrojimo sve potrošače dobivena je procijenjena struja potrebna za normalan rad uređaja koja iznosi oko:

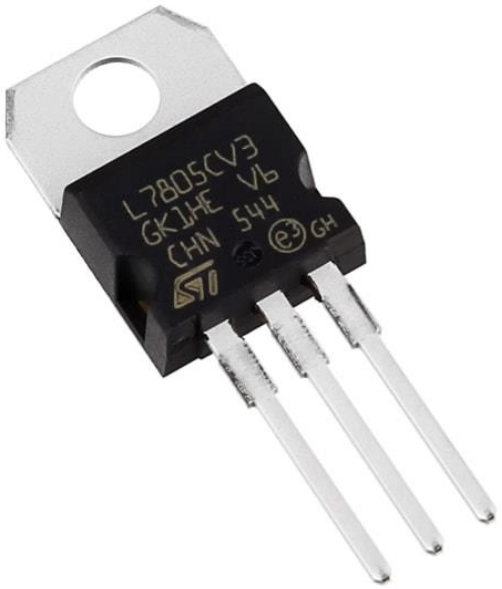
$$I_{uk} = 287 \text{ mA} \quad \textbf{(4)}$$

Alkalna 1,5 V baterija u prosječnim uvjetima rada ima kapacitet od otprilike 2500 mAh. Pošto su baterije spojene u seriju, to znači da se njihov napon zbraja dok im kapacitet ostaje jednak. U ovakvom su spoju iz razloga što je na ulazu stabilizatora potrebno barem 6 V za normalan rad. Procijenjeno trajanje paketa baterija je:

$$t_{bat} = \frac{2500\ mAh}{287\ mA} = 8{,}7 \text{ sati} \quad \textbf{(5)}$$

2.5.3. Stabilizator napona

U ovom radu za stabilizator napona sam koristio L7805CV. Ovaj stabilizator uzima bilo koji napon veći od 6 V na ulazu i pretvara ga u konstantnih 5 V.



Slika 12. Stabilizator napona L7805CV

Pinovi na ovom stabilizatoru su:

- Pin 1 - Ulazni napon
- Pin 2 - Uzemljenje
- Pin 3 - Izlazni napon

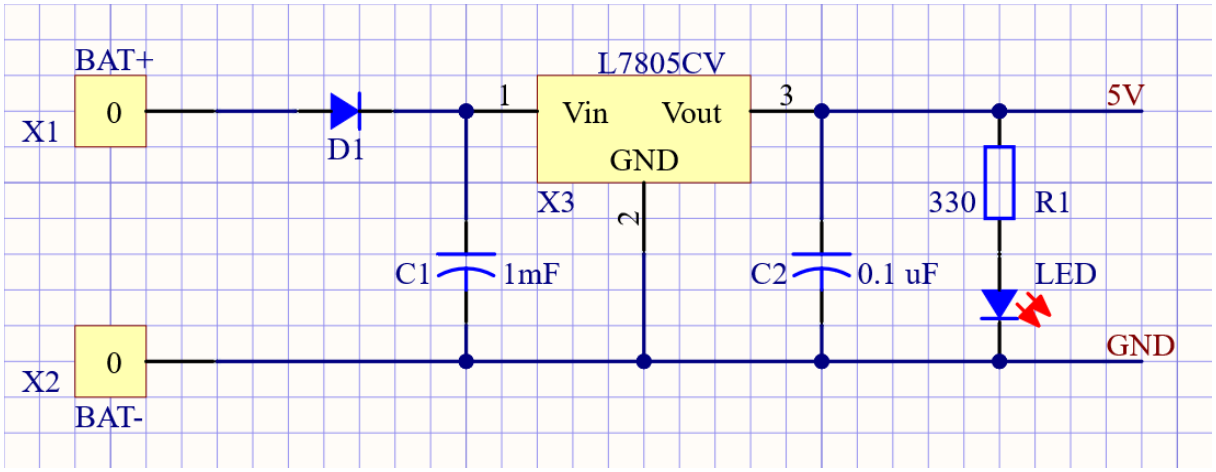
Postoji puno različitih stabilizatora napona. L7805CV je jedan od jeftinijih, a radi svoju zadaću efikasno. Kako je iz prošlog podglavlja vidljivo procijenjena struja koju će ovaj stabilizator trebati puštati je oko 280 mA. Prema karakteristikama od proizvođača ovaj stabilizator je sposoban puštati i do 1,5 A, ali uz jako dobro hlađenje. Za struju od 280 mA dovoljno je pasivno hlađenje na zraku. Električna shema koja prikazuje spoj stabilizatora napona će biti prikazana u idućem poglavlju.

3. ELEKTRIČNA SHEMA UREĐAJA

Pošto je električna shema cijelog uređaja dosta velika, shema će biti rastavljena na određene funkcionalne dijelove. Cijela električna shema je izrađena u programu Altium. Potpuna električna shema će biti priložena uz ovaj rad.

3.1. Shema spajanja stabilizatora napona i paketa baterija

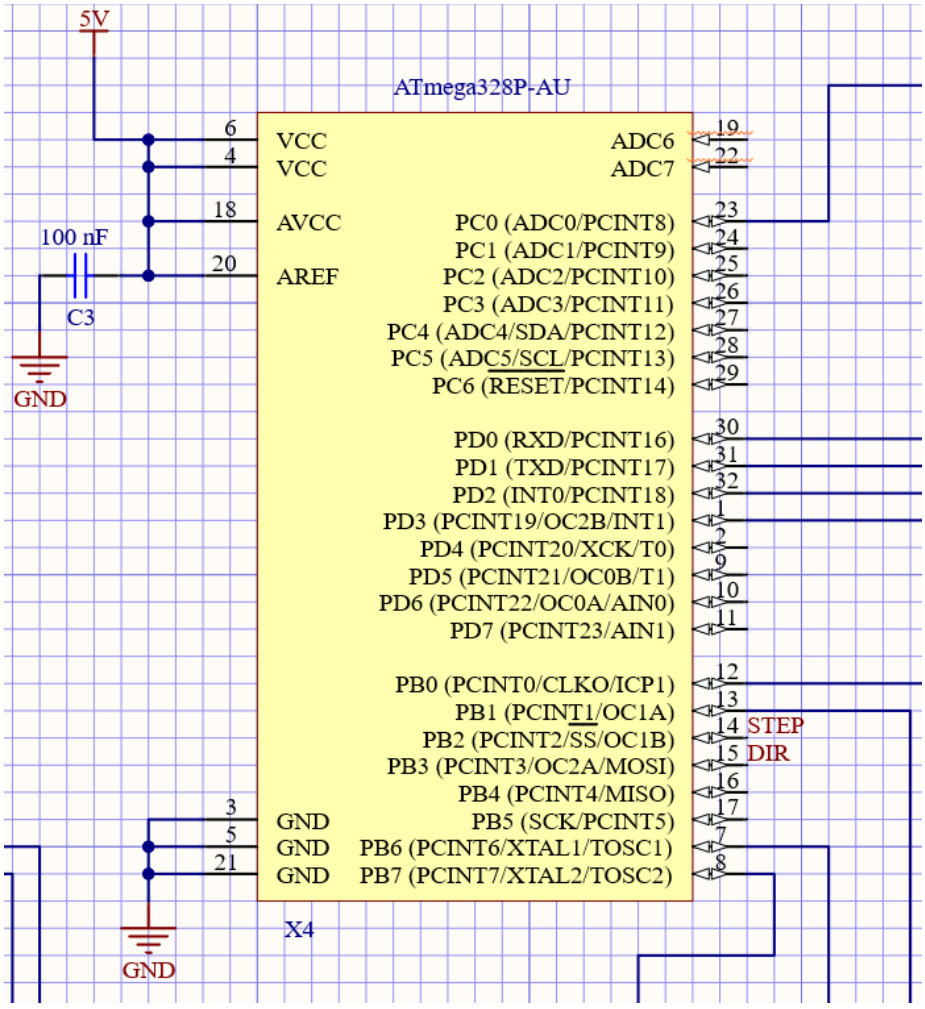
Na [Slika 13] je prikazana shema spoja stabilizatora, te spojna mjesta paketa baterija. Žica višeg potencijala na paketu baterija se spaja na mjesto BAT+. Dioda D1 služi kao zaštita baterija. Za kvalitetan rad stabilizator napona L7805CV zahtijeva spoj dva kondenzatora kako je prikazano na shemi. Napon između pina V_{out} i GND je 5 V ± 7%. Te na desnom kraju sheme je vidljiv spoj otpornika i LED diode koja daje vizualni signal o tome radi li uređaj ili ne. Dalje se svi uređaji kojima trebaju spajaju na 5V i na GND.



Slika 13. Električna shema spajanja stabilizatora napona

3.2. Shema spajanja mikroupravljača

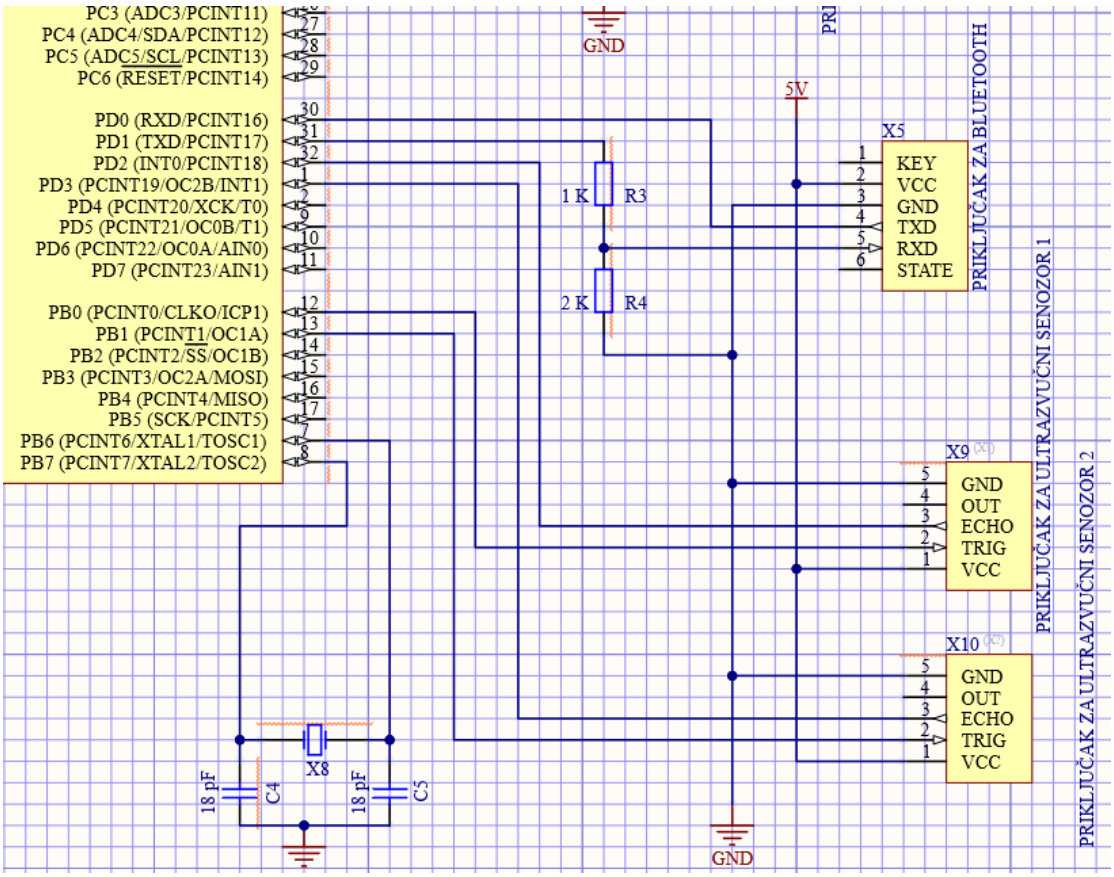
Na [Slika 14] je prikazan spoj mikroupravljača, te sve što je nužno potrebno da mikroupravljač radi. Za pravilan rad mikroupravljača nužni su spojevi na pinovima 3, 4, 5, 6, 7, 8, 18, 20, 21. Pinovi 4 i 6 se spajaju na pozitivnih 5 V, dok se pinovi 3 i 5 spajaju na uzemljenje. U ovoj shemi komponenta X8 je kvarcni kristal koji oscilira na 8 MHz te daje takt mikroupravljaču. Mikroupravljač bi mogao i bez njega raditi sa svojim unutarnjim oscilatorom ali je praksa ukoliko se ide na više frekvencije (8…16 MHz) koristi se vanjski kristal. Kristal također za pravilan i pouzdan rad zahtijeva 2 kondenzatora od 18 pF. Analogno digitalni konverter ne koristimo u ovom radu, ali preporučeno je da se pin 18 i 20 svejedno spoje na pozitivni potencijal.



Slika 14. Električna shema spajanja mikroupravljača

3.3. Shema spajanja ultrazvučnih senzora i bluetooth modula

Na [Slika 15] je prikazan spoj ultrazvučnih senzora i bluetooth modula. X9 i X10 su fizički samo priključci za ultrazvučne senzore koji će se nalaziti na rotirajućoj konstrukciji koja će biti kasnije prikazana. Isto tako je i X5 fizički priključak, tj. pinovi na pločici na koje će se priključiti bluetooth modul. Priključci će biti prikazani na 3D modelu tiskane pločice. I senzori i bluetooth modul zahtijevaju napajanje na pinovima VCC i na uzemljenje na pinovima GND. Kako je i opisano prilikom objašnjavanja komponente ultrazvučnog senzora, senzor radi tako da mu se na pin *TRIG* pošalje signal od 10 µs. Što znači da pinove *TRIG* oba senozra moramo spojiti na pinove na mikroupravljaču koje ćemo programirati da odašiljau takve signale. Za ovaj rad sam proizvoljno izabrao da ti pinovi na mikroupravljaču budu pin 12 i 13, tj. PB0 i PB1. Nakon što je signal bio poslan, senzor na pin *ECHO* šalje signal u trajanju vremena putovanja ultrazvučnog signala do predmeta i nazad Pinovi *ECHO* su spojeni na mikroupravljač na pinove 32 i 1 tj. PD2 i PD3. Ovi pinovi imaju i sekundaru svrhu, a ta je da služe kao vanjski interrupt-i mikroupravljača. Pinovi *OUT* ostaju ne spojeni.

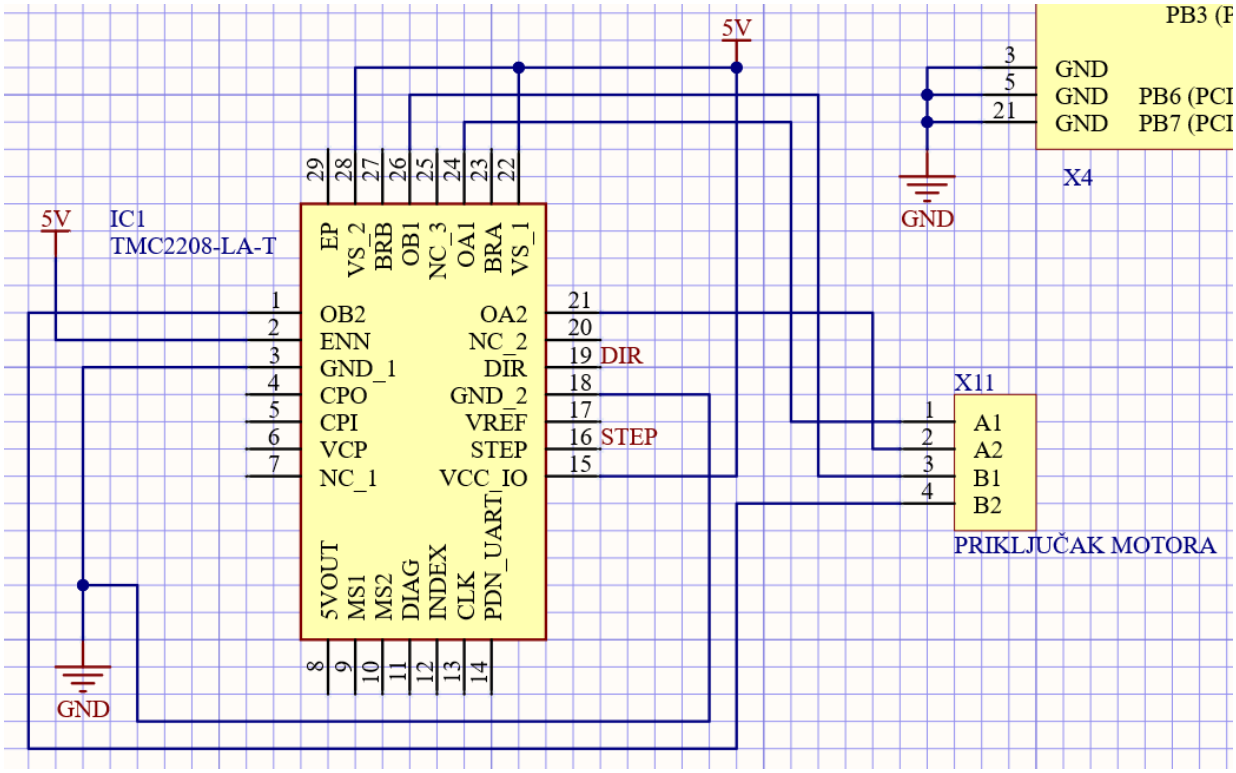


Slika 15. Električna shema spajanja ultrazvučnih senzora i bluetooth modula

X5 je samo mjesto gdje će se Bluetooth modul priključiti u pločicu putem 6 pinskog sučelja. Kao i svaki drugi uređaj Bluetooth modul zahtijeva napajanje preko pinova VCC i GND koji se spajaju na 5V i GND na pločici. Na mikroupravljaču pinovi 30 i 31 osim što imaju funkciju kao i svi drugi ulazno izlazni pinovi imaju i sekundarnu funkciju RXD i TXD u serijskoj komunikaciji. Kao što je bilo napisano RXD znači „Receive Data" tj. na taj pin mikroupravljač prima podatke. TXD znači „Transmit Dana", tj. mikroupravljač šalje podatke preko ovog pina. Bluetooth modul šalje podatke mikroupravljaču preko svojeg pina TXD koji se onda mora spojiti na pin RXD na mikroupravljaču. Kako je i prije napisano Bluetooth modul šalje signale napona 3,3 V, što je mikroupravljaču dovoljno da prepozna to kao pozitivni signal, tako da je spoj između TXD na Bluetooth modulu i RXD na mikroupravljaču direktan. S druge strane mikroupravljač svoje podatke na pinu TXD šalje pri naponu od 5 V što bi bio previsok napon za Bluetooth modul, zato koristimo djelitelj napona kako je prikazano na [Slika 9] i izračunato pomoću jednadžbe (3). Pinove „State" i „Key" sa Bluetooth modula ne spajamo nigdje.

3.4. Shema spajanja drivera motora

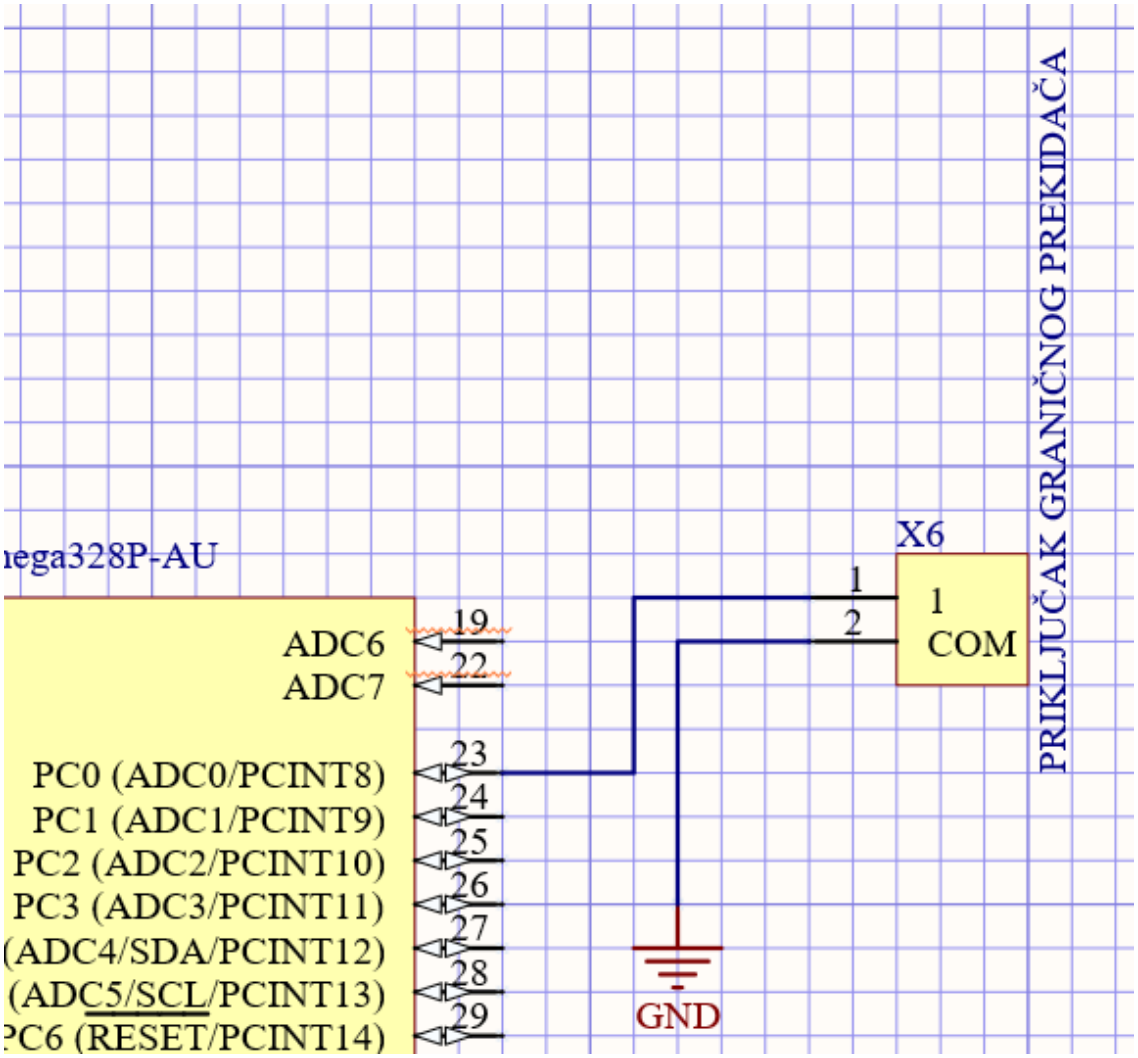
Na [Slika 17] je prikazan je spoj čipa TMC2208. Spojio sam samo one pinove koji su nama bitni. Pinove STEP i DIR sam spojio na pinove 14 i 15 na mikroupravljaču. Pinove 15, 22 i 28 čipa sam spojio na napajanje od 5V, dok sam pinove 3 i 18 spojio na uzemljenje. Pin 2, tj „Enable“ sam spojio na pozitivnih 5V kako bi motor bio uključen cijelo vrijeme. I posljednje sam pinove nazivom A1, A2, B1 i B2 spojio na priključak X11 koji će se dalje spojiti na koračni motor.



Slika 16. Električna shema spajanja drivera motora

3.5. Shema spajanja graničnih mikroprekidača

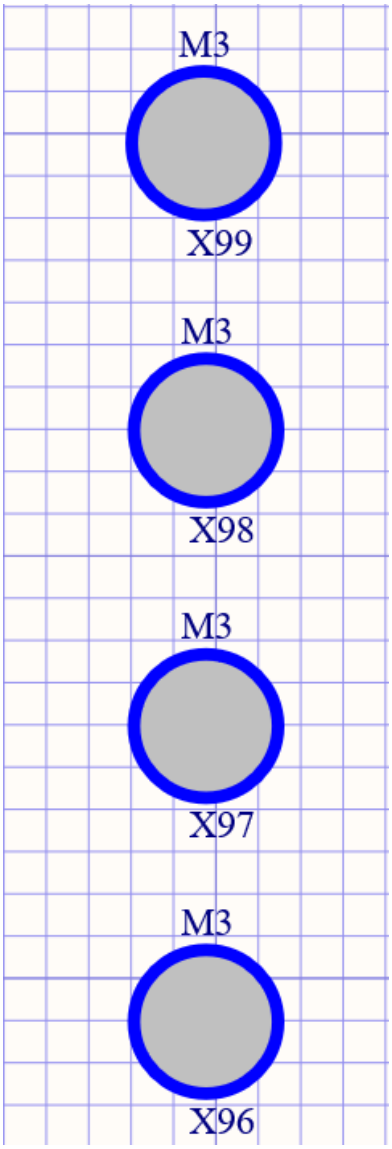
Na [Slika 17] je prikazana shema spajanja priključaka graničnog mikroprekidača. Kao i kod ultrazvučnih senzora, mikroprekidač se ne nalazi na tiskanoj pločici već je pričvršćen na konstrukciji na mjestu koje će signalizirati krajnji položaj. Pin COM na priključcima je zapravo pin „Normally Open“ na mikroprekidaču, što znači kada mikroprekidač postane stisnut onda poveže pin 1 i COM. Mikroprekidač je spojen na pin PC0 na mikroupravljaču.



Slika 17. Električna shema spajanja priključaka graničnih prekidača

3.6. Dodaci

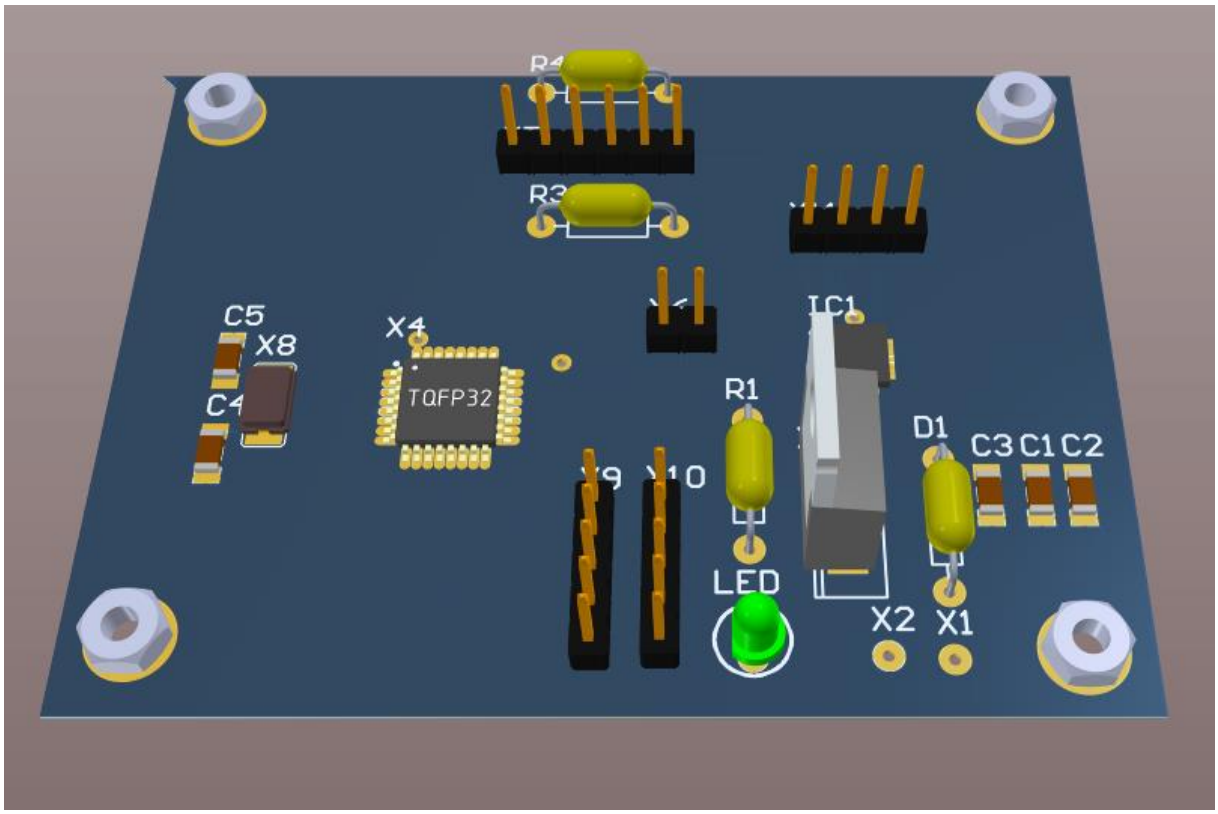
Na [Slika 18] su prikazani dodatni elementi koji će biti potrebni kod izrađene tiskane pločice. X99…96 su M3 vijci kojima će se tiskana pločica biti pričvršćena za konstrukciju. Izgled tiskane pločice biti će prikazan u idućem poglavlju.



Slika 18. Vijci

3.7. Fizički izgled tiskane pločice

U programu u kojem sam i napravio električnu shemu ovoga rada(Altium), sam i napravio 3D model tiskane pločice koji je vidljiv na slici [Slika 19]. Na internetu sam našao informaciju da ove vrste pinova podržavaju i do 400 mA. Ukoliko dođe potreba za jačim pinovima na priključku motora, to se lagano zamjeni.



Slika 19. 3D model tiskane pločice

4. KONCEPT RADA

4.1. Interrupt-ovi

Interrupt u mikroupravljaču znači da kada se dogodi određeni događaj, mikroupravljač zaustavlja trenutnu radnju, te ide izvršiti interrupt rutinu. Mi ćemo u ovom radu koristiti vanjski interrupt i Overflow interrupt. Vanjski interrupt je okinut kada na određeni pin mikroupravljača dođe određeni signal ili određena promjena signala. Interrupt rutine su zapravo funkcije koje odredimo da se izvrše kada se okine interrupt.

Naše obje interrupt rutine će biti okidane na svaku promjenu signala pošto, kako je vidljivo na slici [Slika 3], senzor daje pozitivan brid za početak mjerenja, koji dolazi u neko ne određeno vrijeme nakon slanja signala na pin *TRIG*, te je zato potrebno kada je okinut interrupt a pin *ECHO* pozitivan, pokrenuti Timer, a kada je okinut interrupt, a pin *ECHO* negativan, onda zaustavljamo Timer i određujemo udaljenost detektiranog predmeta od senzora. U programu ćemo koristiti Interrupt 0 i Interrupt 1.

4.2. Timer-i

Za precizno mjerenje vremena trajanja signala koristimo Timer-e. Timer-i su zapravo brojači pulsova, a pulsovi koje on mjeri dolaze od vanjskog kvarcnog kristala, čija je frekvencija poznata, te tako možemo jednom pulsu dodijeliti određeno vremensko razdoblje. Pošto naš kvarcni kristal ima frekvenciju od f_{kv} = 16 MHz, što znači da vremensko razdoblje jednog pulsa ispada t = 62,5 ns. Ako recimo detektiramo predmet na udaljenost jednog metra, prema formuli

$$2 * udaljenost\ predmeta = brzina\ zvuka * pulsova * 62{,}5\ ns$$

Te ako za brzinu zvuka u zraku uzmemo broj od 343 m/s, ukupni broj pulsova potrebnih da detektiramo ovaj predmet je skoro 100 000, a jedan puls je proporcionalan udaljenosti od 1 μm. Ovakva preciznost nam nije potrebna jer u specifikaciji senzora piše da ima rezoluciju od oko 3 milimetra, a i mogli bi imati problema ako bi trebali brojati tako velike brojeve.

Kako bi smanjili te brojeve koristimo prescaler-e koji dijele frekvenciju ulaznog pulsa. Time automatski i smanjujemo preciznost mjerenja, ali zato ćemo uzeti prescaler koji neće narušavati preciznost. Tako sam se odlučio na prescaler 64, pošto će s tim prescalerom jedan

puls biti proporcionalan udaljenost od 0.6 mm, što je još uvijek dosta pre precizno, ali idući veći prescaler je 256, što već postaje pregrubo.

Pošto ću u radu koristiti funkcije poput *delay* i *delayMicrosecond* mikroupravljač će uzeti Timer 0 za sebe kako bi te funkcije mogle dobro raditi, te onda nama ostaju Timer 1 i Timer 2. Timer 1 je 16-bitni dok je Timer 2 8-bitni. To znači da Timer 1 može brojati do broja $2^{16} = 65536$, a Timer 2 može brojati do $2^8 = 256$. Kada timeri dođu do tih brojeva resetiraju se i počinju od početka Za predmet ja udaljenost od oko 2 metra će nam sada trebati oko

$$puls = \frac{2\ \mathrm{m}}{0{,}6\ \mathrm{mm}} = 3333$$

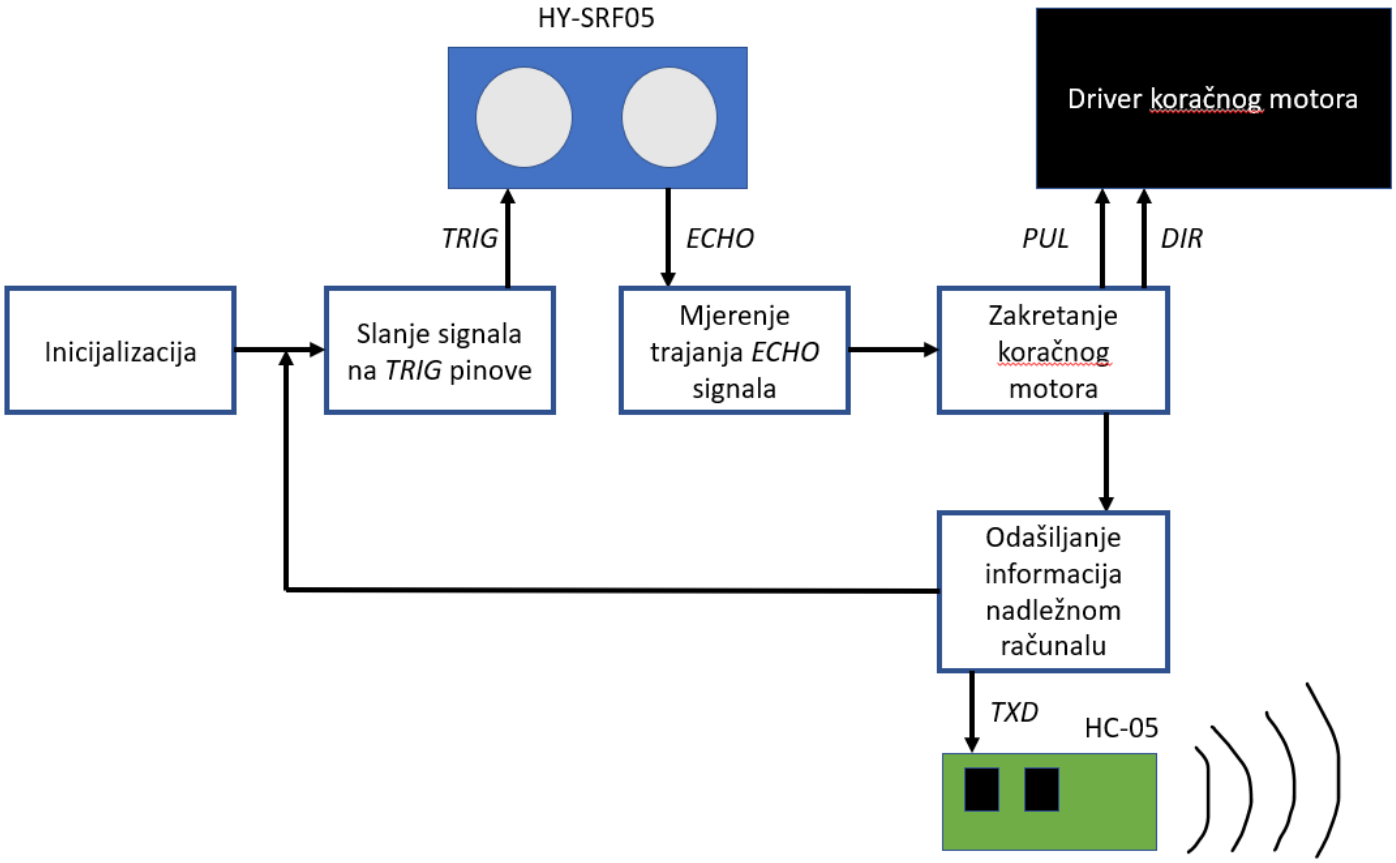
Što je puno više od 256 koliko Timer 2 može nabrojati. Što znači da ćemo nekako morati brojati koliko puta je Timer 2 došao do 256. Tu dolazi Overflow interrupt. Overflow interrupt radi slično kao i interrupt-ovi koje smo u prošlom poglavlju objašnjavali, znači da program zaustavlja sav rad i ide izvršiti određenu rutinu. Jedina razlika je ta što se obični interrupt-ovi okidaju vanjskim signalom dok se Overflow interrupt-ovi okidaju „Prelijevanjem“ Timer-a, tj. okidaju se kada je Timer dostigao svoju maksimalnu vrijednost, što znači, u slučaju Timer-a 2, kada dostigne 256, prelije se i resetira, što okida Overflow interrupt.

U Overflow rutini Timer-a 2 ćemo brojati koliko puta se je Timer 2 prelio.

4.3. UART komunikacija

UART komunikacija je jedna od mnogo vrsta serijskih komunikacija. Zove se serijska komunikacija pošto se informacije šalju serijski, tj. bitovi „dolaze“ jedan za drugim, dok bi recimo paralelna komunikacija značila da skup bitova „putuje“ paralelno te cijela informacija dođe od jednom. Prednost serijske komunikacije je njezina jednostavnost. Potrebne su nam samo dvije žice, jednom šaljemo informacije, a drugom primamo.

4.4. Tijek programa

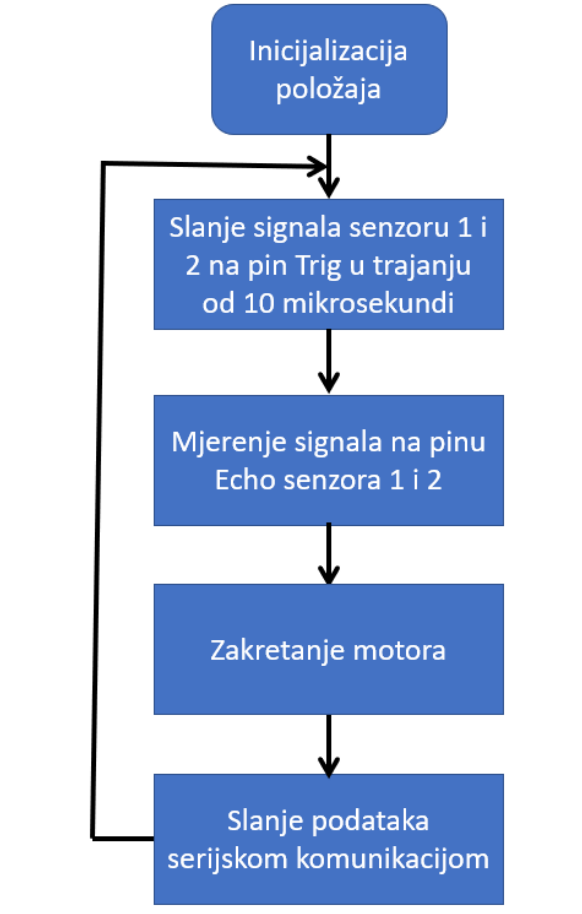


Slika 20. Koncept rada programa mikroupravljača

Na slici [Slika 20] je ilustriran koncept rada programa koji će se nalaziti na mikroupravljaču. Sve počinje inicijalizacijom globalnih varijabli te raznih funkcija koje ćemo koristiti u programu. Nakon toga mikroupravljač šalje signal na oba *TRIG* pina. Ultrazvučni senzori počinju mjerenje te šalju signal na pin *ECHO* u trajanju mjerenja. Nakon primitka signala, možemo odrediti koliko je udaljen detektirani predmet od pojedinog senzora. Dalje, driveru koračnog motora šaljemo informaciju o smjeru vrtnje te signal da može zakrenuti motor. Dok je motor u pokretu, kod ide dalje te bluetooth modulu serijskom komunikacijom šalje sve informacije potrebne. Bluetooth modul prosljeđuje te informacije nadležnom računalu. Kada je krug završen mikroupravljač čeka potvrdu od računala da može nastaviti, te se opet vraćamo na slanje signala *TRIG* pinovima.

5. LISTING KODA

5.1. Algoritam



Slika 21. Algoritam programa

Na slici [Slika 21] je prikazan osnovni izgled algoritma koji ćemo sada prikazati u obliku programa. Pošto programiram mikroupravljač na platformi Ardino, program ću napraviti u Arduino programu. Programski jezik je u suštini C jezik.

Kod će, radi lakšeg razumijevanja, biti podijeljen na funkcionalne cjeline. Prvo će biti prikazan kod koji se nalazi na mikroupravljaču. Tako će po redu biti prikazano: setup, interrupt i Overflow rutine, inicijalizacija položaja, određivanje udaljenosti detektiranog predmeta od senzora, slanje informacija računalu, te posljednje, zakretanje motora. Nakon koda na mikroupravljaču biti će prikazan kod na računalu.

5.2. Setup, interrupt i overflow rutine

5.2.1. Deklaracija pinova i varijabli

```
const unsigned int TRIG_PIN1=12;
const unsigned int ECHO_PIN1=2;
const unsigned int TRIG_PIN2=13;
const unsigned int ECHO_PIN2=3;
const unsigned int BAUD_RATE=9600;
const unsigned int PUL=8;
const unsigned int DIR=9;
const unsigned int ENA=10;
const unsigned int prekidac = 4;

volatile int vr1;
volatile int vr2;

volatile float dur1;
volatile float dur2;

volatile int ovf1 = 0;
volatile int ovf2 = 0;

volatile bool mj1;    //ove varijable ce biti true dokle god je mjerenje u tij
volatile bool mj2;

bool dir = HIGH;

volatile int i = 1;
```

Slika 22. Deklaracija pinova i globalnih varijabli

Odmah na samom početku programa ćemo odrediti pinove koji će nam biti potrebni, te neke globalne varijable koje ćemo koristiti unutar funkcija.

TRIG_PIN1 i *ECHO_PIN1* su pinovi prvog senzora te smo odredili da će oni nama biti spojeni na fizičke pinove 12 i 2. Tako su i *TRIG_PIN2* i *ECHO_PIN2* pinovi drugog senzora te smo odredili da su oni spojeni na pinove 13 i 3. Još nam trebaju upravljački pinovi drivera koračnog motora, *PUL, DIR* i *ENA*. Oni će biti spojeni na pinove 8, 9 i 10.

Dalje ćemo deklarirati globalne varijable koje će se koristiti u interrupt i Overflow funkcijama, tj. rutinama. Prvo deklariramo *volatile int vr1* i *vr2*. Ove varijable ćemo koristiti za izračun udaljenosti detektiranog predmeta od senzora 1 ili 2. Kasnije će biti prikazana funkcija kojom ćemo to računati. Pošto će rezultat udaljenosti biti u milimetrima, zaključio sam da nam

neće biti potrebna preciznost od desetinke milimetra(a ni nisam siguran koliko senzor može točno odrediti udaljenost), tako da koristimo tip varijable *int*. Dodatak *volatile* je tu samo da compiler tu varijablu ne provuče kroz svoje optimizacijske funkcije zato što će se ova varijabla mijenjati na više mjesta unutar programa, što bi moglo predstavljati problem ukoliko ju compiler proba optimizirati. Dalje nam su deklarirane varijable *volatile int ovf1* i *ovf2*. Ove varijable će pokazivati koliko puta se je okinuo interrupt Overflow, jer je to važno prilikom izračuna udaljenosti, što ćemo vidjeti i kasnije u radu. Iduće su deklarirani *volatile bool mj1* i *mj2,* Ove varijable su tipa *bool* što znači da mogu imati samo dva logička stanja, tj. *true* ili *false*. Kada je ova varijabla *true*, znači da taj senzor odrađuje mjerenje. I posljednje su deklarirani *bool dir* koji će u početnom stanju imati vrijednost *HIGH* a ta varijabla će određivati smjer kretanja koračnog motora i varijabla *volatile int i* koja će označavati u kojoj poziciji je koračni motor.

5.2.2. Interrupt rutine

```
void mjerenje1()
{
  if (digitalRead(ECHO_PIN1) == HIGH)
  {
    TCNT1 = 0;
    ovf1 = 0;
    mj1 = true;
  }
  else
  {
    if (mj1 == true)
    {
     vr1 = 343000*(ovf1*65536+TCNT1)/500000; //34300 je brzina zvuka
     mj1 = false;
    }
  }
}

void mjerenje2()
{
  if (digitalRead(ECHO_PIN2) == HIGH)
  {
    TCNT2 = 0;
    ovf2 = 0;
    mj2 = true;
  }
  else
  {
    if (mj2 == true)
    {
     vr2 = 343000*(ovf2*256+TCNT2)/500000;
     mj2 = false;
    }
  }
}
```

Slika 23. Interrupt rutine

Funkcije *void mjerenje1()* i *void mjerenje2()* su interrupt rutine, koje ćemo vidjeti kako sam odredio kasnije u radu, te ćemo vidjeti da sam ih postavio da se okidaju prilikom svake promjene na interrupt pinu. Odmah na početku funkcije provjeravamo dali je pin u pozitivnom

stanju. Ukoliko je znači da je senzor dao signal za početak mjerenja, te registar Timer-a 1 postavljamo u nulu i varijablu *mj1* stavljamo u stanje *true* kako bi označili da je senzor 1 u procesu mjerenja. Ukoliko ustvrdimo da pin nije u pozitivnom stanju, onda to znači da je senzor dao signal prestanka mjerenja, te možemo izračunati udaljenost detektiranog predmeta. Udaljenost nam predstavlja varijabla *vr1* koja je računata formulom

$$2 * vr_1 = v_{zvuka} \frac{ovf_1 * 65536 + TCNT1}{f_{prescaler}}$$

Gdje je:

- v_{zvuka} = 343 000 mm/s
- ovf_1 - koliko puta se je Timer 1 prelio
- *TCNT1* - broj do kojeg je Timer 1 nabrojao
- $f_{prescaler}$ - frekvencija nakon prescalera = 250 000 Hz

Također varijablu *mj1* stavljamo u stanje *false* kako bi označili da je senzor gotov sa mjerenjem. Ovo se praktički potpuno ponavlja za funkciju *mjerenje2()*, jedina veća razlika je kod formule za računanje udaljenosti gdje sad dolazi broj 256 umjesto broja 65536, pošto drugi senzor mjeri pomoću Timer-a 2 koji je 8 bitni za razliku od Timer-a 1 koji je 16- bitni.

```
ISR(TIMER1_OVF_vect)
{
  ovf1++;
}

ISR(TIMER2_OVF_vect)
{
   ovf2++;
}
```

Slika 24. Overflow rutine

Još imamo i Overflow interrupt rutine koje su jako jednostavne pošto samo povećavamo broj koliko se puta pojedini Timer prelio.

5.3. Inicijalizacija položaja

Inicijalizacija položaja koračnog motora je prilično jednostavna. Pokrećemo motor u jednu stranu sve dok ne dotakne granični prekidač, te onda stanemo i možemo započeti dalje s normalnim radom.

```
void initStep()
{
 while(1){
   if(digitalRead(prekidac == HIGH))
   {
     digitalWrite(PUL, LOW);
     delay(20);
     digitalWrite(PUL, HIGH);
     delay(20);
   }

   else
   {
     dir = !dir;
     digitalWrite(DIR, dir);
     i = 0;
     break;
   }
 }

}
```

Slika 25. Inicijalizacija položaja

5.4. Setup

Dalje ulazimo u *void setup()* koji će se izvršiti samo prilikom prvog pokretanja mikroupravljača, i u njemu određujemo neke početne vrijednost i uloge pojedinih pinova. Prvo određujemo svim pinovima dali su ulazni ili izlazni naredbom *pinMode()* koja prima dva argumenta. Prvi argument je pin kojem određujemo funkciju, a drugi argument je ili *input* ili *output*. Iduće koristimo funkciju *digitalWrite()* kako bi postavili početne vrijednosti pinova.

Dalje, imamo funkciju *attachPinToInterrupt* koju koristimo kako bi spojili interrupt rutinu sa određenim pinom. Prvi argument je pin na koji će doći interrupt signal, drugi argument je naziv funkcije koja će se izvršiti kada se okine interrupt, te treći argument može biti *LOW, CHANGE, RISING, FALLING*. Ja sam izabrao *CHANGE*.

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Slika 27. TCCRxB registar

Iduće moramo inicijalizirati Timer 1 i Timer 2. To radimo tako da upisujemo određene vrijednosti u određene registre prema dokumentaciji proizvođača. Jedan od važnijih registara je TCCRxB (x je broj Timera, 1 ili 2). U kojem su nama najvažniji bitovi 0, 1 i 2, koji određuju prescaler.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Slika 26. Kombinacije CSxx bitova

Pošto nama treba prescaler 64, bitove CS11 i CS10 postavljamo u jedinicu. Napomena: kod Timer-a 2 kombinacije CS-ova su malo drugačije tako da za Timer 2 stavljamo jedinicu u CS22, što nam onda daje prescaler od 64.

Dalje, još jedan bitan registar je TIMSK registar, jer njegov nulti bit pali Overflow interrupt, tako da ćemo i taj bit prebacit u jedinicu.

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Slika 28. TIMSKx registar

Te, na kraju setup-a započinjemo serijsku komunikaciju sa našim bluetooth modulom. Funkcija *Serial.begin()* ima argument BAUD_RATE što označuje brzinu prijenosa informacija, tj. broj prenesenih bitova u sekundi. Ključno je da oba uređaja komuniciraju istom brzinom. Default baud rate bluetooth modula je 9600, tako je i tu ta brzina.

5.5. Loop program

```
void loop() {
 while(Serial.available() == 0)
 {
 }
 int a = Serial.read();
 //initStep();
 digitalWrite(TRIG_PIN1, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIG_PIN1, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN1, LOW);
 mj1 = true;

 digitalWrite(TRIG_PIN2, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIG_PIN2, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN2, LOW);
 mj2 = true;
 while ( mj1 || mj2) //while ce se izvrsavati dokle god j
 {
  if (ovf1 == 1 || ovf2 == 12)  //ukoliko ovf ijednog tim
  {
    if (mj1 == true)    //ako je mj1 true znaci da senzor
    {
      mj1 = false;
      vr1 = 0;
    }

    if (mj2 == true)
    {
      mj2 = false;
      vr2 = 0;
    }

    break;
  }
```

Slika 29. Loop program - 1

```
if(i == 85 || i == 0)
{
  dir = !dir;
  digitalWrite(DIR, dir);
}
digitalWrite(PUL, LOW);
delay(20);
digitalWrite(PUL, HIGH);
delay(20);
if(dir == HIGH)
{
 i++;
}
else
{
 i--;
}
Serial.print("x");
Serial.print(vr1+1000);
Serial.print("y");
Serial.print(vr2+1000);
Serial.print("z");
Serial.print(i+10);
delay(2);
}
```

Slika 30. Loop program - 2

Loop program je program na mikroupravljaču konstantno ponavlja. Program počinje tako da čekamo bilo koju poruku od nadležnog računala serijskom vezom kako bi započeli detekciju. Kada poruka dođe program izlazi iz while petlje, te čistimo dolazni registar tako da očitamo dolaznu poruku u neku varijablu. Nakon toga ćemo pomoću funkcija *digitalWrite* i *delayMicroseconds* postlati na pin *TRIG_PIN1* i *TRIG_PIN2* poslati pozitivan signal u trajanju od 10 mikrosekundi. Te nakon slanja signala varijable *mj1* i *mj2* prebacujemo u stanje *true* kako bi označili da su senzori u procesu mjerenja.

Dalje program ulazi u while petlju u kojoj ostajemo sve dok se i *mj1* i *mj2* ne vrate u stanje *false*. Ovdje čekamo rezultate mjerenja, te ukoliko se dogodi da se recimo Timer 2 prelije 12 puta automatski zaustavljamo mjerenje, pošto nismo ništa detektirali unutar određenog dometa, te kako bi se ubrzao program, prestajemo sa mjerenjem i nastavljamo dalje.

Nakon toga imamo zakretanje koračnog motora. U prvoj if naredbi, ukoliko je koračni motor došao do krajnjeg položaja, mijenjamo smjer vrtnje. Dalje je vidljivo da šaljemo jedan impuls na pin *PUL,* dok pritom koristimo *delay* funkciju. Kada bi smanjivali delay koračni motor bi se brže okretao, ali ovo mi se činila idealna brzina.

Te na kraju loop funkcije šaljemo informacije serijskom komunikacijom. Kako bi pojednostavio programiranje na strani računala, odlučio sam nekako normirati dolaznu informaciju. Tako sam se odlučio da prvo šaljem slovo x, koje će označavati da iza njega dolazi četveroznamenkasti broj koji označuje udaljenost koju je očitao prvi senzor u mm. Kako bih osigurao da broj koji šaljem bude četveroznamenkast, prije slanja sam rezultantnoj udaljenosti pridodao 1000, koji sam onda kasnije na računalu jednostavno oduzeo. Nakon toga dolazi slovo y, te nakon njega dolazi drugi četveroznamenkasti broj koji označuje udaljenost koju je očitao drugi senzor. Tu je također trik sa pridodavanjem broja 1000. Te posljednje dolazi slovo z, iza kojeg se nalazi broj u kojoj poziciji je koračni motor. Nakon slanja informacije loop kreče ispočetka.

5.6. Program na nadležnom računalu

Program na računalu sam programirao u programskom jeziku python. U pythonu koristim library Tkinter koji olakšava izradu grafičkih prikaza. Također koristimo library serial, koji olakšava serijsku komunikaciju, i library math kojeg ćemo koristiti prilikom prikaza rezultata, pošto će nam trebati trigonometrijske funkcije.

U 10. liniji započinjemo serijsku komunikaciju brziom 9600 bitova po sekundi, na COM portu 8 na računalu, te sam stavio timeout, tj. vrijeme koje treba proći da se veza prekine ukoliko ne dođe nikakva informacija putem serijske komunikacije. Dalje, imamo definiranu funkciju *radar()* koja će napraviti prozor prikaza radara. Nakon toga dolazimo do funkcije *radi()* koja će uzimati informacije i iscrtavati ih na radaru. Funkcijom *ser.write(b'1')* mikroupravljaču šaljemo informaciju da može početi sa radom. Nakon toga čekamo informaciju od mikroupravljača u veličini od 13 bitova. Od tih 13 bitova, kako smo normirali u prošlom poglavlju, prvo ide slovo x, a nakon njega 4 brojke, pa y, pa 4 brojke, te slovo z i 2 brojke, što rezultira 13 bitova. Idući problem je što informacije stignu u obliku varijable *char,* a ne *int* koji će nam biti potreban kod računanja pozicije.

To rješavamo tako da, prvo dobivenu informaciju stavimo u listu kako bise njom lake koristili. Kada smo dobivenu informaciju prebacili u listu slova i brojke su se prebacile u

njihove UTF-8 kodove, tj. broj 0 je jednak 48 i tako dalje.. a slovo x, y i z su brojevi 120, 121 i 122. Nakon toga u listi tražimo pozicije slova x, y i z, tj. tražimo indexe brojeva 120, 121 i 122. Kada imamo pozicije tih slova onda listu dijelimo na liste koje nama odgovaraju, tj. dijelimo ju na informaciju senzora 1, senzora 2 i informaciju o zakretu motora. Nakon toga te informacije prebacujemo u *int* varijablu.

Sada imamo sve potrebno za prikaz informacije. U liniji 97 je provjeravamo ako je udaljenost 0, onda preskačemo crtanje točke, ako je bilo što drugo, onda crtamo točku veličine 2x2 pixela. Te u 102. liniji vidimo da ukoliko smo dosegli maksimalni broj točaka, izbrišemo najstariju točku. I posljednje u 120. liniji funkcijom *win.after(40, radi)* ponovo pokrečemo funkciju radi nakon 40 ms. Izabrao sam ovo vremensko razdoblje pošto sve manje od toga rezultira lošim izvođenjem programa.

```
1   from turtle import delay
2   import serial
3   import time
4   from tkinter import *
5   from tkinter import font
6   import math
7   import tkinter
8
9
10  ser = serial.Serial()
11  ser.baudrate = 9600
12  ser.port = 'COM8'
13  ser.timeout = 10
14
15  ser.close()
16
17  ser.open()
18
19
20  global i
21  i = 0
22
23  win = Tk()
24
25  win.title("Ultrazvučni radar")
26
27  def radar():
28      for widget in win.winfo_children():
29          widget.destroy()
30      win.title("Prikaz radara")
31      win.geometry("1400x900")
32      ekran = Canvas(win, bg = "black", height = 700, width=1300)
33
34      stupnjevi = [0, math.pi/6, math.pi/3, math.pi/2, math.pi*2/3, math.pi*5/6, math.pi]
35      for stup in stupnjevi:
36          linija = ekran.create_line((650, 650, (650+math.cos(stup)*600), (650-math.sin(stup)*600)), fill = "green", width = "3")
37
38      for aud in range(100, 700, 100):
39          ark = ekran.create_arc((650-aud, 650-aud, 650+aud, 650+aud), outline = "green", width = "3", style=tkinter.ARC, extent=180)
40
41      tekst1 = ekran.create_text(550,670, fill = 'red', font=('Times', 20, 'bold'), text='30 cm')
42      tekst2 = ekran.create_text(450,670, fill = 'red', font=('Times', 20, 'bold'), text='60 cm')
43      tekst3 = ekran.create_text(350,670, fill = 'red', font=('Times', 20, 'bold'), text='90 cm')
44      tekst4 = ekran.create_text(250,670, fill = 'red', font=('Times', 20, 'bold'), text='120 cm')
45      tekst5 = ekran.create_text(150,670, fill = 'red', font=('Times', 20, 'bold'), text='150 cm')
46
47      tekststup1 = ekran.create_text(25,650, fill = 'red', font=('Times', 20, 'bold'), text='90°')
48      tekststup2 = ekran.create_text(1275,650, fill = 'red', font=('Times', 20, 'bold'), text='90°')
49      tekststup3 = ekran.create_text(110,340, fill = 'red', font=('Times', 20, 'bold'), text='60°')
```

Slika 31. Program 1

```
    tekststup4 = ekran.create_text(1190,340, fill = 'red', font=('Times', 20, 'bold'), text='60°')
    tekststup5 = ekran.create_text(340,110, fill = 'red', font=('Times', 20, 'bold'), text='30°')
    tekststup6 = ekran.create_text(960,110, fill = 'red', font=('Times', 20, 'bold'), text='30°')
    tekststup7 = ekran.create_text(655,30, fill = 'red', font=('Times', 20, 'bold'), text='0°')
    ekran.pack()

    time.sleep(1)

    def radi():

        max_tocaka = 100        #maksimalni broj tocaka na ekranu
        global i

        print(ser)
        ser.write(b'1')
        a = ser.read(13)
        print(a)
        print(type(a))
        lista = list(a)
        print(lista)
        pozx = lista.index(120)
        pozy = lista.index(121)
        pozz = lista.index(122)

        listadaljina1 = lista[pozx+1:pozy]

        print(listadaljina1)

        listadaljina2 = lista[pozy+1:pozz]

        print(listadaljina2)

        listapozicije = lista[pozz+1:]
        print(listapozicije)

        daljina1 = (listadaljina1[0]-48)*1000+(listadaljina1[1]-48)*100+(listadaljina1[2]-48)*10+(listadaljina1[3]-48)-1000
        print(daljina1)

        daljina2 = (listadaljina2[0]-48)*1000+(listadaljina2[1]-48)*100+(listadaljina2[2]-48)*10+(listadaljina2[3]-48)-1000
        print(daljina2)

        pozicija = (listapozicije[0]-48)*10+listapozicije[1]-48-10      #ovo je u stupnjevima i treba pretvoriti u radijane
        print(pozicija)
        pozicijarad = pozicija*math.pi/180

        kut = 90*math.pi/180    #kut izmedu senzora
```

Slika 32. Program 2

```
        if (daljina1 == 0):
            pass
        else:
            pred = ekran.create_oval(649-(daljina1/3)*math.cos(pozicijarad), 649-(daljina1/3)*math.sin(pozicijarad), 651-(daljina1/3)*math.cos(pozicijarad), 651-(daljina1/3)*math

            if i == max_tocaka:
                zadnji = ekran.find_above(25)
                ekran.delete(zadnji)
                i = max_tocaka-1
            ekran.pack()
            i+=1

        if (daljina2 == 0):
            pass
        else:
            pred = ekran.create_oval(649-(daljina2/3)*math.cos(pozicijarad+kut), 649-(daljina2/3)*math.sin(pozicijarad+kut), 651-(daljina2/3)*math.cos(pozicijarad+kut), 651-(dalj

            if i == max_tocaka:
                zadnji = ekran.find_above(25)
                ekran.delete(zadnji)
                i = max_tocaka-1
            ekran.pack()
            i+=1
        win.after(40, radi)        #ovo moze i krace, ali posto je 40 ms jedan puls za pokretanje motora, ovo je taman vrijeme


    radi()


pokr = Button(win, text = "Pokreni radar", font = ("Helvetica", 30), command = radar)
pokr.grid(column=0, row=0)

win.geometry("500x200")

win.mainloop()
```

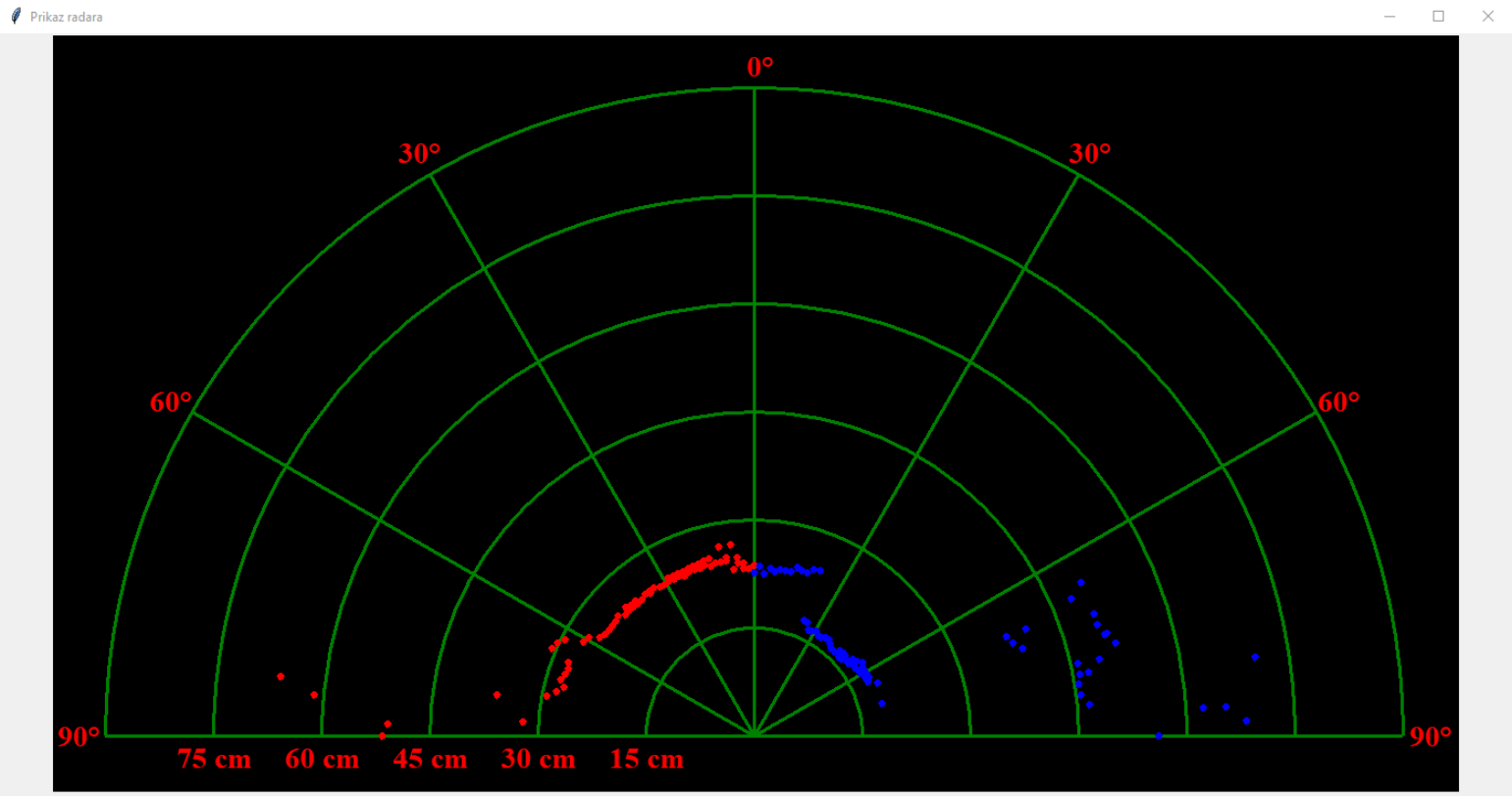
Slika 33. Program 3

6. PRIKAZ REZULTATA

Za simulaciju rada senzora napravio sam jednostavnu okolinu od nekoliko ravnih površina kao što je vidljivo na slici [Slika 34].

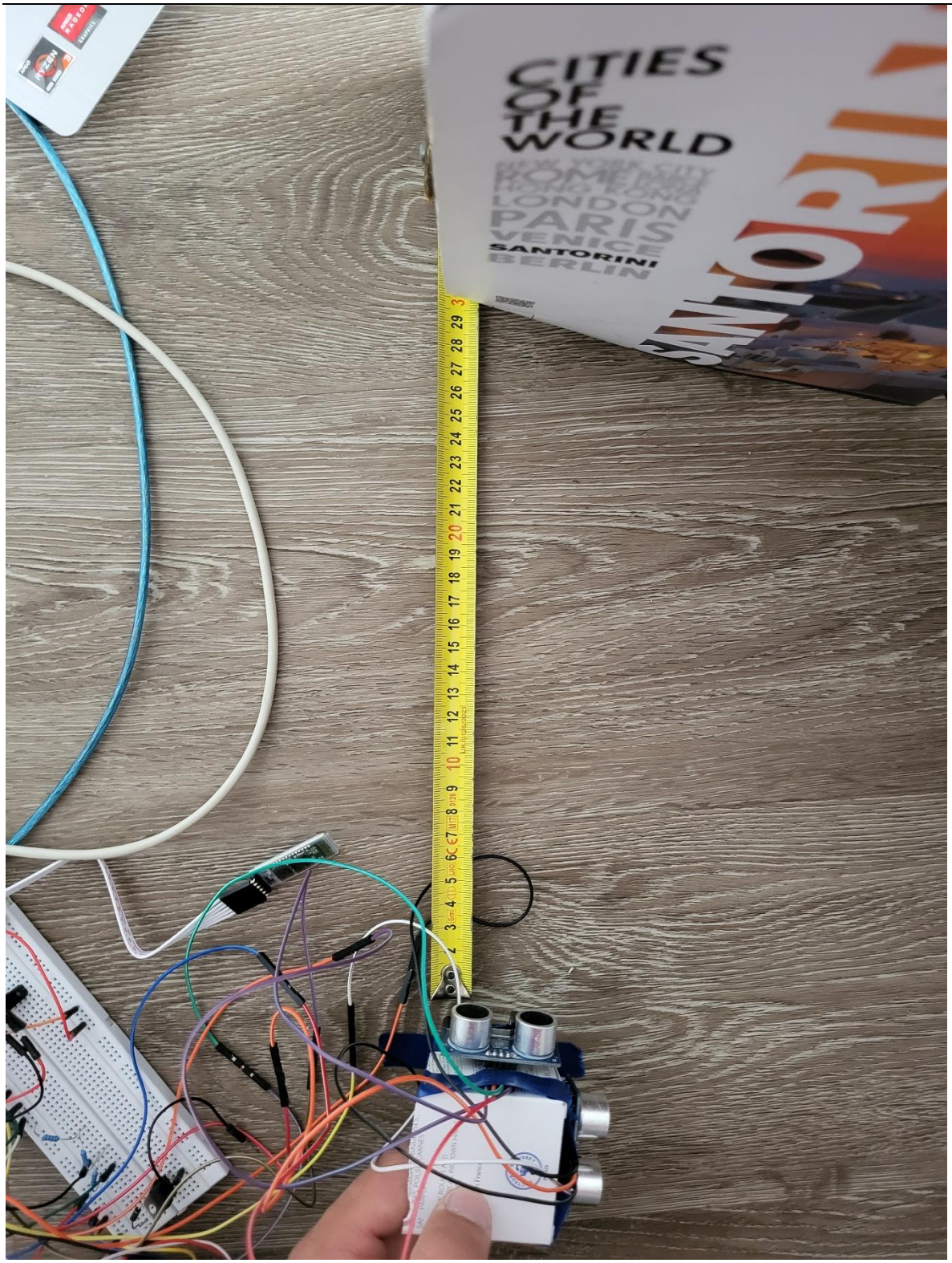
Slika 34. Okolina senzora – pokus 1

Pustio sam senzor da napravi sken prostora nekoliko puta, te sam zabilježio rezultat, što je i vidljivo na slici [Slika 35].

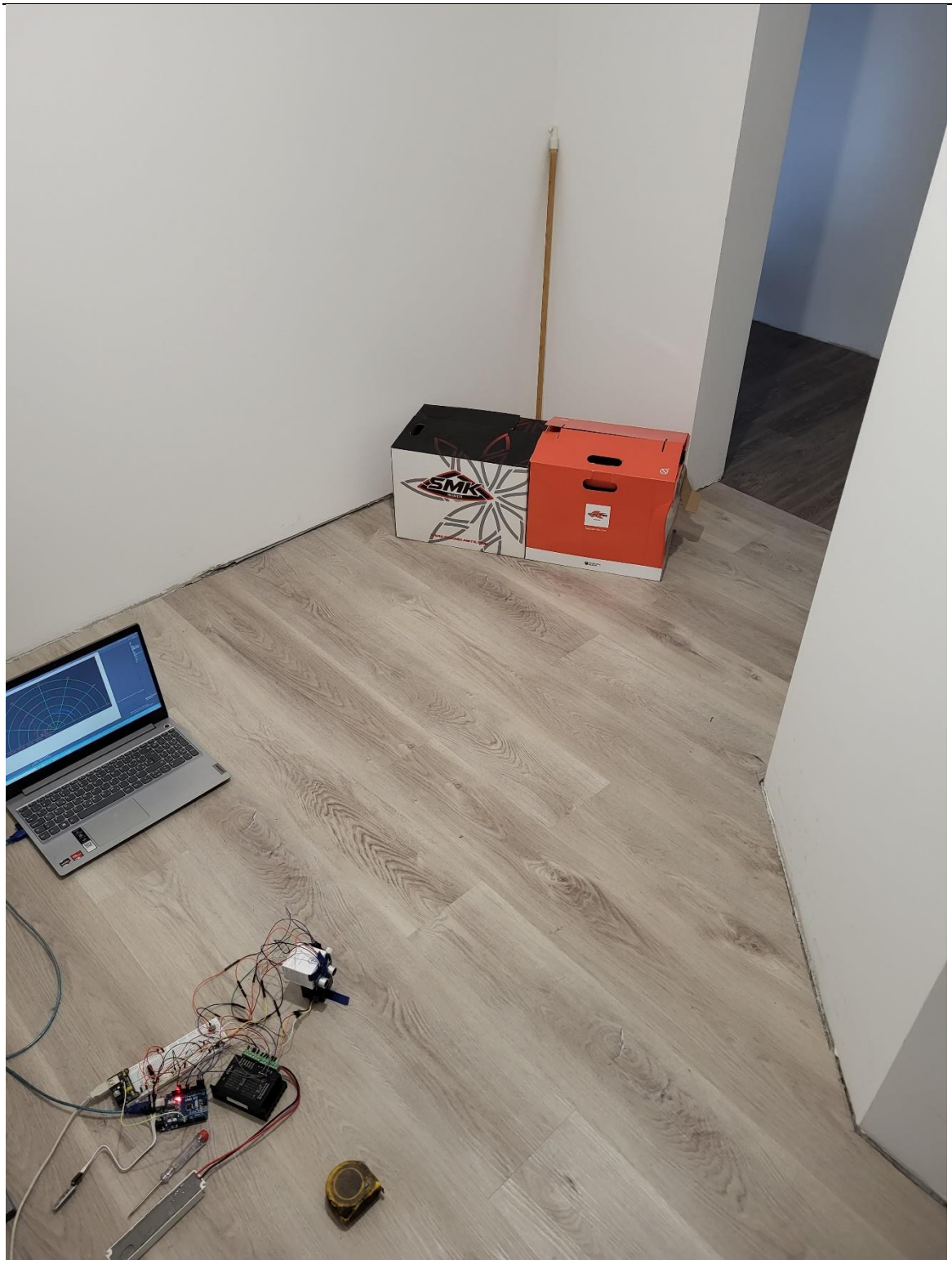


Slika 35. Prikaz rezultata na radaru – pokus 1

Da malo prokomentiram rezultat. Točkice koje je detektirao senzor na desno su plave, a točkice detekcije crvene boje su od senzora na lijevo. Mogu se razaznati 3 ravne površine. Spoj bilježnica je više zakrivljen nego što bi morao biti, ali se svejedno mogu razaznati dvije različite ravne površine. Vidljivo je sa desne strane da je jedna ravna površina bliže nego ostale, što je i vidljivo na slici stvarne okoline. Ravna površina, tj. srednja bilježnica je postavljena u zonu detekcije i jednog i drugog senzora, te se može vidjeti na sredini radara da se linija lijepo nastavlja sa jednog senzora na drugi. Vidimo da je s lijeve strane bilježnica udaljena nešto manje od 30 cm. Mjerenjem udaljenost, što je vidljivo na slici [Slika 36], došli smo do zaključka da je senzor i precizan.

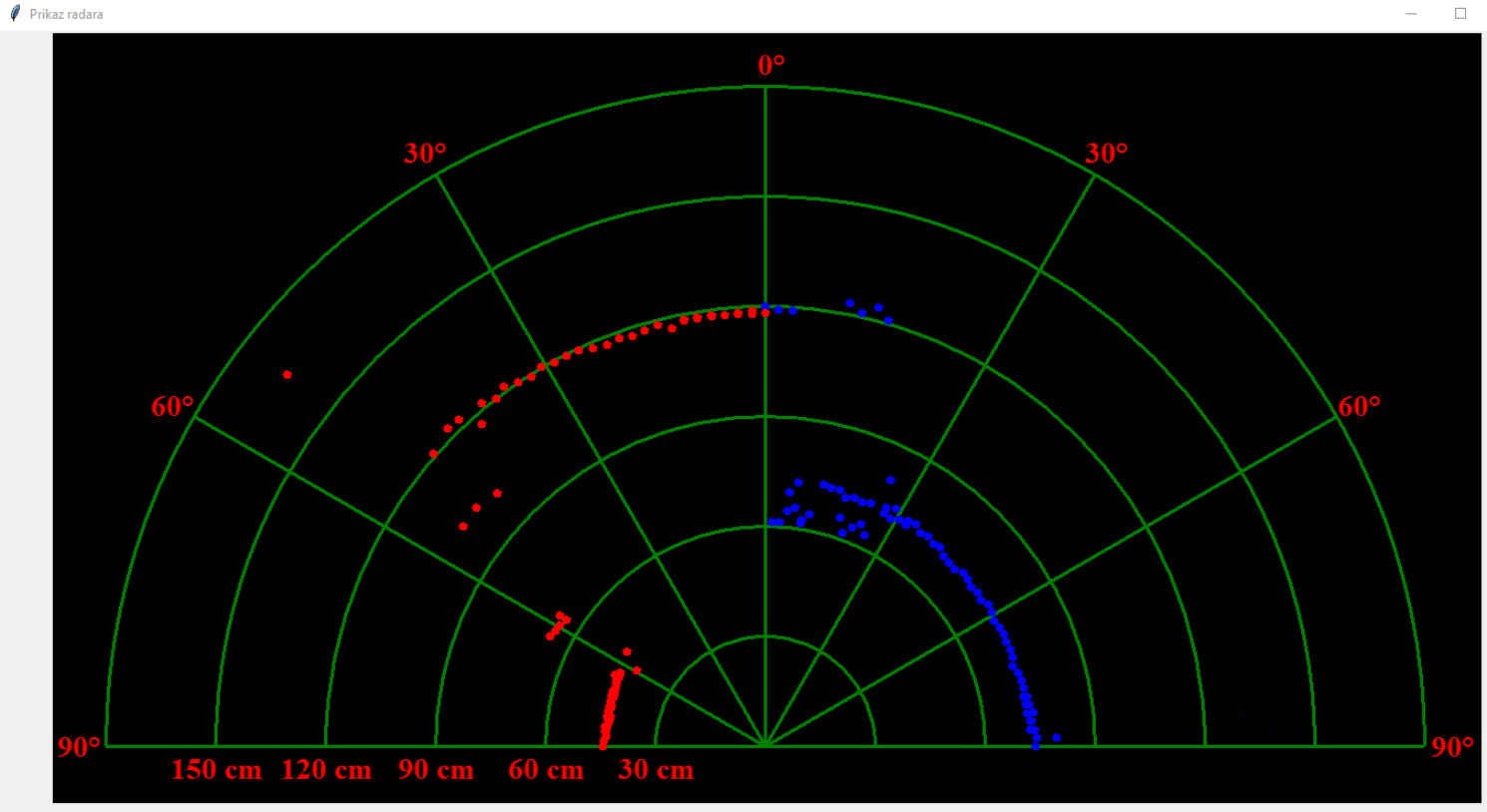


Slika 36. Udaljenost kuta bilježnice – pokus 1



Slika 37. Okolina senzora – pokus 2

U drugom pokusu malo sam povećao razmak između senzora i predmeta koji mora detektirati. Također sam i na prikazu radara na računalu malo povećao domet prikazivanja.



Slika 38. Prikaz rezultata na radaru – pokus 2

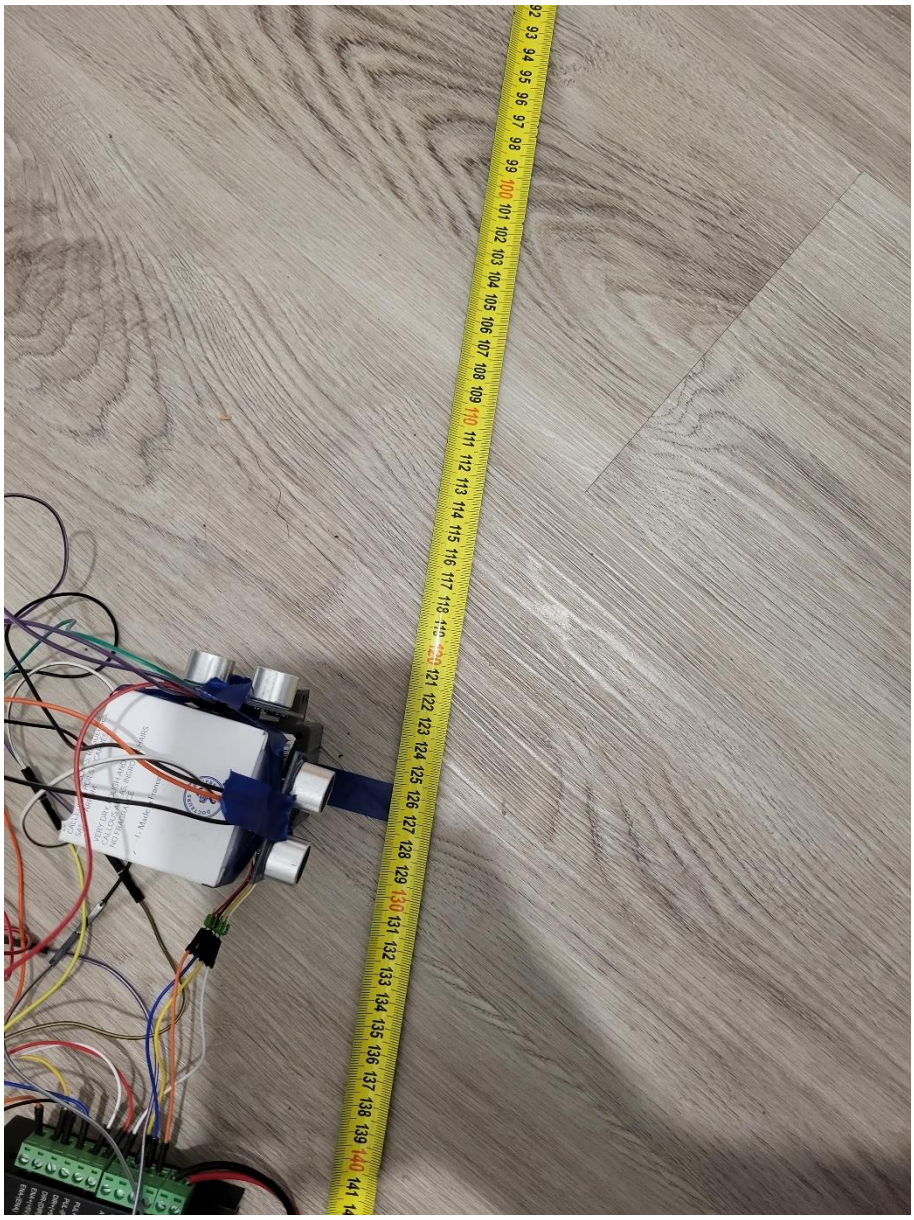
Malo da prokomentiram rezultat. Skroz lijevo u zoni od 90 do 60 stupnjeva se može razaznati laptop. U lijevoj zoni od 45 stupnjeva pa sve do desne zone do 5 stupnjeva se mogu vidjeti dvije postavljene kutije. Sa desne strane u zoni od 90 do 30 stupnjeva se može vidjeti kosi zid.

Vidimo da kada senzor mora mjeriti na veće udaljenosti dobivamo svakakve šumove i neobične rezultate. Recimo zona na desnoj strani između 30 i 5 stupnjeva bi trebala biti potpuno prazna jer taj dio gleda u prolaz u hodniku. Također na lijevoj strani možemo vidjeti u zoni od 60 do 45 stupnjeva neobične stepenice kojih nema u stvarnosti.

Skoro pa u potpunosti sam siguran da ove neobične rezultate možemo pridijeliti nedostatku čvrste konstrukcije na koračnom motoru i mogućem preslikavanju između dva senzora.

Kutija koja drži senzore se stalno naginjala zbog težine senzora i povlačenja žica, senzori sigurno ne gledaju paralelno tlu nego u tlo ili u zrak, te se sami motor radi svojih vibracija malo zakreče u odnosu na pod. Također radar možda radi malo prebrzo te je moguće da se još od prošlog mjerenja odbio signal do senzora te ga je on očitao kao važeći.

Uz pravilnu konstrukciju siguran sam da bi dobili puno bolje rezultate.



Slika 39. Udaljenost kutije – pokus 2

Na slici [Slika 39] vidljivo je da su kutije udaljene oko 120 cm, točno kako je prikazano na radaru.

7. ZAKLJUČAK

Radar radi zadovoljavajuće i relativno brzo. Treba mu oko 3 sekunde za zakret od 90 stupnjeva. Uz malo optimizacije koda, možda bi mogao raditi i još brže. U prikazu rezultata sam pokazao i da je relativno točan, ali bi mogao biti točan i u par milimetara kada bi imao pravo kućište, a ne komad kartona i ljepljive trake.

Prilikom izrade ovog rada shvatio sam da uz malo truda robot može dobiti jako dobru sliku svoje okoline, a da pritom ne trebamo uložiti mnogo novaca. Za potpunu izradu ovog rada trebalo bi izdvojiti oko 40€. Ovim radarom robot bi mogao detektirati predmete do 4 metra u zoni od 180 stupnjeva pred njim, dok bi recimo samo LiDAR senzor sa dometom od 12 metara koštao preko 60€.

Ovaj radar je savršen za male projekte sa niskim budžetom, za recimo samostalna vozila koja se ne kreču velikom brzinom.

LITERATURA

[1] Getting Started with HC-05 Bluetooth Module[Internet], <raspoloživo na: https://create.arduino.cc/projecthub/electropeak/getting-started-with-hc-05-bluetooth-module-arduino-e0ca81 >, [30.1.2022.]

[2] PySerial Documentation[Internet], <raspoloživo na: https://pyserial.readthedocs.io/en/latest/ >, [16.2.2022.]

[3] Tkinter Documentation[Internet], <raspoloživo na: https://docs.python.org/3/library/tk.html >, [18.2.2022.]