SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Vinko Bohnec, Antonio Bernobić,

Mateo Zović, Jakov Begović

Studij: Informacijski i poslovni sustavi

ROBOT UPRAVLJAN KONTROLEROM

PROJEKT

Mentor/Mentorica:

mag. inf. Lovro Posarić

Varaždin, siječanj 2023.

Vinko Bohnec, Antonio Bernobić,

Mateo Zović, Jakov Begović

Izjava o izvornosti

Izjavljujemo da je naš projekt izvorni rezultat našeg rada te da se u izradi istoga nismo koristili drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Sažetak

Projekt se bavi izradom robota kojim je moguće upravljati preko vlastito izrađenog kontrolera (daljinskog upravljača). Robot omogućuje razne mogućnosti i funkcionalnosti, od kojih su neke, upravljanje kretanjem robota, paljenje i gašenje led svjetla u definiranim intervalima, puštanje raznih zvučnih signala i melodija putem zujalice te automatsko kočenje. Temelj projekta jesu dvije Dasduino pločice koje međusobno komuniciraju i razmjenjuju podatke. Jedna od njih služi kao kontroler za slanje naredbi, dok druga Dasduino pločica obavlja definirane radnje. Projekt ne iziskuje samo trud oko slaganja robota i spajanja dijelova, već se bazira i na kvalitetnom programskom rješenju. Svaka Dasduino pločica ima zaseban program koji izvršava svake sekunde. Ovaj projekt predstavlja savršen spoj mehanike (dijelovi robota), elektronike (elektroničke komponente) i informatike (programiranja), koji u konačnici tvore robota.

Ključne riječi: robot, daljinsko upravljanje, kretanje, Dasduino pločice, komunikacija programsko rješenje, mehanika, elektronika, informatika

Sadržaj

1. Uvod 1
2. Modeli 2
2.1. Općenita arhitektura sustava 2
2.1.1. Dasduino CONNECTPLUS (ESP32) 3
2.1.2. Modul za upravljanje motorima - L298N DC motor driver 4
2.1.3. Senzor udaljenosti – HC-SR04 Ultrasonic Sensor 6
2.1.4. Zujalica 8
2.1.5. Svjetleće diode 8
2.1.6. Maska odašiljača 9
2.2. Složena arhitektura sustava 11
2.3. Komunikacijski dijagrami 12
2.3.1. Korisnik ne pritišće niti jedan gumb 12
2.3.2. Korisnik pritišće gumb gore 13
2.3.3. Korisnik pritišće gumb dolje 14
2.3.4. Korisnik pritišće gumb lijevo 15
2.3.5. Korisnik pritišće gumb desno 15
2.3.6. Korisnik pritisne gumb muzika 16
2.4. Dijagram aktivnosti 17
2.4.1. Staza korisnika 17
2.4.2. Staza upravljača 18
2.4.3. Staza robota 20
2.5. Shema ili dijagram spajanja 22
3. Izvršni program robota 24
3.1. Klase 24
3.2. Upravljač-Odašiljač 30
3.3. Robot-Primatelj 32
4. Kratki troškovnik i financijska analiza 34
5. Korisnička dokumentacija 36
5.1. Kretanje robota 36
5.2. Ostale funkcionalnosti 38
6. Zaključak 41
Popis literature 42
Popis slika 43

1. Uvod

U današnjem svijetu sve više i više je prisutno raznih robota. Svi smo se, već sigurno bar jednom, susreli s pojmom robota, no što je zapravo robot i kako on djeluje? „Robot je automatizirani stroj višestruke namjene, koji može obavljati neke zadaće slične ljudskom djelovanju. S obzirom na stupanj autonomnosti, mogućnosti interakcije s okolinom i inteligencije, razlikuje se nekoliko skupina (generacija) robota.“ [1] Prvoj skupini pripadaju programirani roboti, u drugu skupinu spadaju roboti opremljeni nizom senzora, a trećoj skupini robota pripadaju inteligentni roboti. „Programirani roboti ne koriste povratnu informaciju o svojem stvarnom stanju i ne mogu korigirati pogrješke vođenja. Roboti druge generacije opremljeni su nizom senzora, koje koriste za dobivanje povratnih informacija o svojem stvarnom stanju i stanju okoline. Inteligentni roboti imaju sposobnost učenja, rezoniranja i donošenja zaključaka.“ [1] S obzirom na brzi razvoj robota, kojem trenutno najviše pridonosi ubrzani razvoj umjetne inteligencije, nije ni čudo da su danas roboti prisutni u mnogim djelatnostima. Shodno tome, roboti imaju različite primjene, od industrijskih i kućanskih robota, pa sve do svemirskih i humanoidnih robota. Iz svega navedenog je na posljetku, i tema našeg projekta proizašla iz ideje i želje za izradom vlastitog robota. Izrada robota nije jednostavna jer ona uključuje složeni sustav koji objedinjuje računalnu opremu i komponente te programsku podršku to jest izvršni program. Upravo robot i njegove komponente predstavljaju pojam ugrađenog sustava, a njegovim izvršnim programom se ostvaruje ideja programiranja za ugrađene sustave. Iz tog razloga, robot upravljan vlastoručno izrađenim kontrolerom predstavlja savršenu temu za stjecanje novih znanja i vještina iz područja ne samo robotike, već i iz čitavog područja ugrađenih sustava.

2. Modeli

Model u tehničkom svijetu predstavlja pojednostavljenu stvarnost. Dakle, prema nekim predmetima iz stvarnosti moguće je izraditi (najčešće) manju, pojednostavljenu repliku tog predmeta. Izradom modela se nastoji omogućiti promatranje kompleksnog sustava i njegovih ključnih dijelova. U ovome slučaju, model robota je prikazan na način da se dijelovi robota prvo promatraju kao samostalna cjelina, a tek kasnije kao složena cjelina, to jest složeni sustav.

2.1. Općenita arhitektura sustava

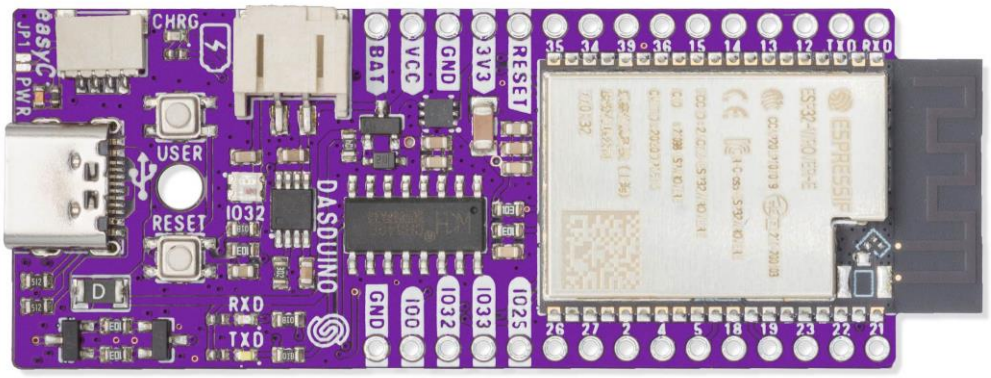
Osnovna konstrukcija robota sastoji se od postolja robota, četiri kotača, četiri istosmjerna (DC) motora, te vijaka i matica. Takva konstrukcija tvori temelj za daljnji razvoj robota. Na osnovnu konstrukciju su nadalje dodani razni elektronički dijelovi potrebni za ispravno funkcioniranje robota. Ti dijelovi će biti predstavljeni pojedinačno u sljedećim poglavljima.

Slika 1: osnovna konstrukcija robota - Dijelovi

Slika 2: osnovna konstrukcija robota - Sastavljeno

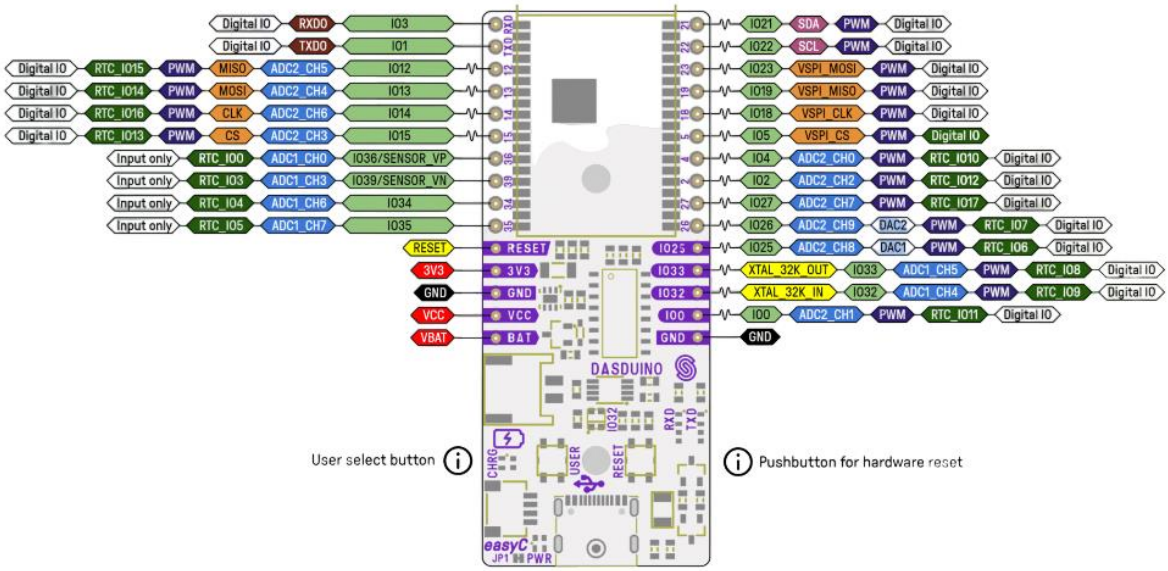
2.1.1. Dasduino CONNECTPLUS (ESP32)

Dasduino CONNECTPLUS pločica je elektronička pločica razvijena od hrvatske kompanije Soldered Electronics. Soldered Electronics nalazi se u Osijeku, a bavi se razvojem, dizajniranjem te proizvodnjom vlastitih elektroničkih proizvoda. Jedan od njihovih proizvoda jest Dasduino pločica. Postoje različite vrste pločica Dasduina, koje se razlikuju po čipu koji koriste i elektroničkoj shemi. U sljedećem dijelu će biti opisana pločica korištena u ovome radu - Dasduino CONNECTPLUS (ESP32).



Slika 3: Dasduino CONNECT PLUS

Dasuino CONNECTPLUS pločica koristi mikročip ESP32-WROVER-E, te koristi napon rada 3.3V (integrirani 5V regulator). Pločica se sastoji od 30 pinova, od kojih neki rade s digitalnim signalima, analognim te onih pinova koji imaju mogućnost upravljanja širinom (duljinom) impulsa (PWM). Na sljedećoj slici [Slika 4] je prikazana Dasduino CONNECTPLUS pločica skupa sa svojima konekcijama.

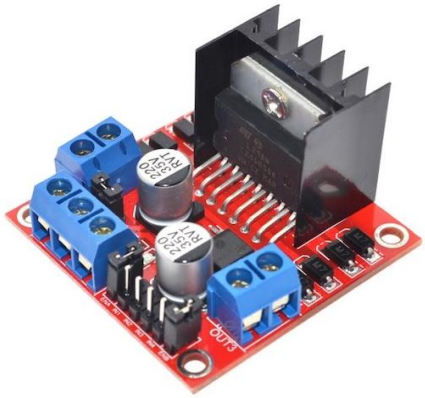


Slika 4: Dasduino CONNECTPLUS (ESP32) - Prikaz konekcija

Na prikazu konekcija se može uočiti da većina pinova sadrži opciju upravljanja širinom impulsa (PWM – Pulse Width Modulation), a samim time, ti pinovi se mogu koristiti za rad s analognim signalima. Skoro svi pinovi omogućavaju rad s ulaznim i izlaznim signalima (IO – Input Output).

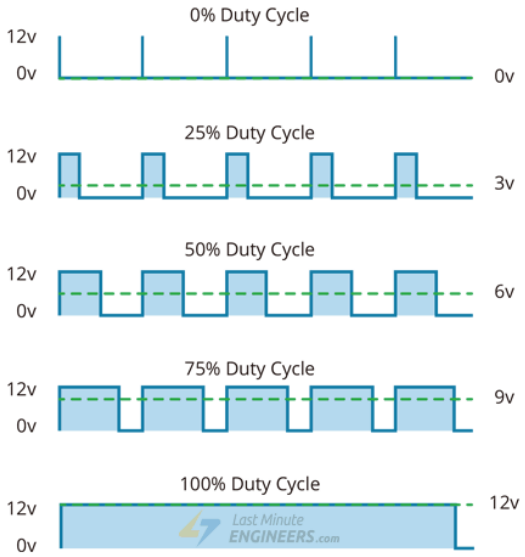
2.1.2. Modul za upravljanje motorima - L298N DC motor driver

Za upravljanje motorima robota nije dovoljna samo Dasduino pločica, već je potreban i upravljač motora, koji je u ovom slučaju to L298N DC motor driver. Na taj način se preko Dasduino pločice šalju naredbe upravljaču motora o promjenama brzine i smjeru vrtnje motora.



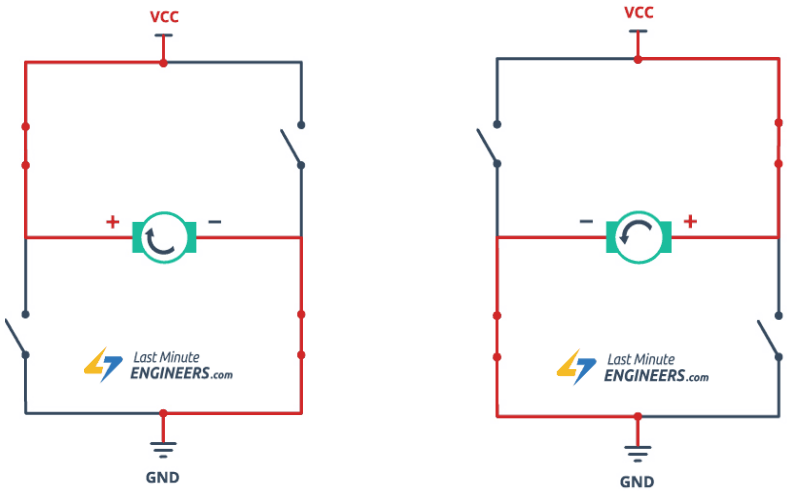
Slika 5: L298N DC motor driver

Promjena brzine vrtnje motora se ostvaruje promjenom ulaznog napona na motorima. Taj princip rada se zasniva na promjeni širine impulsa (PWM – Pulse Width Modulation) tako što slanjem duljih impulsa dajemo veću brzinu vrtnje motora, a slanjem manjih odnosno kraćih impulsa dajemo manju brzinu vrtnje motora, što je vidljivo na sljedećoj slici [Slika 6].



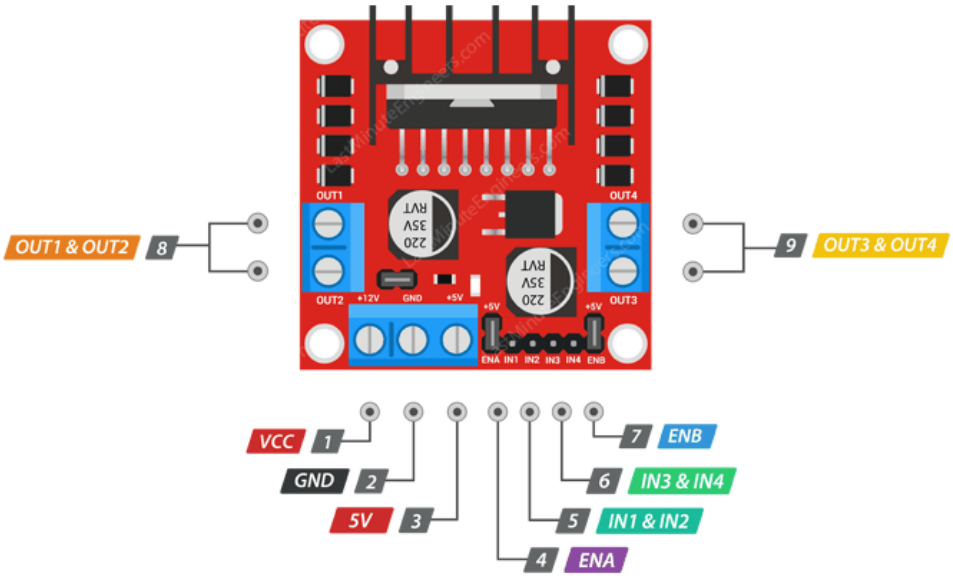
Slika 6: promjenom širine impulsa dobiva se različiti napon na izlazu

Promjena smjera vrtnje motora se ostvaruje promjenom polariteta ulaznog napona, a takav način rada zasnovan je na principu H-Bridge. H označava izgled strujnog kruga u kojem imamo 4 sklopke od koje su dvije sklopke zatvorene, dok su druge dvije otvorene. Time mijenjamo polaritet napona na motorima. Bridge, što u prijevodu označava most, isto predstavlja izgled i način rada strujnog kruga, čiji je princip rada prikazan na sljedećoj slici [Slika 7].



Slika 7: mijenjanje polariteta napona na motorima H-Bridge principom

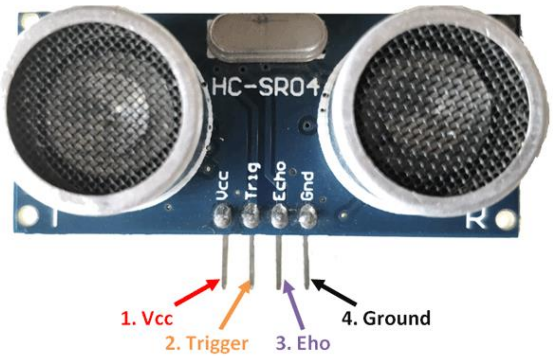
Kod L298N DC motor drivera L298N predstavlja naziv čipa koji koristi, koji je srce toga modula. DC (Direct Current) označava da taj modul koristi istosmjernu struju, a motor driver opisuje funkciju modula – upravljanje motorima. Pomoću L298N DC motor drivera možemo upravljati dvama istosmjernim motorima istovremeno, koji se napajaju od 5 do 35 V. Njih spajamo na izlaze 1,2,3 i 4 (OUT1, 2 ,3, i 4). Pomoću ENA i ENB upravljamo brzinama vrtnje motora, a ulazima 1,2,3 i 4 (IN1, 2, 3, i 4) upravljamo smjerom vrtnje motora. VCC je napajanje, koje služi za pogon motora, a može biti između 5 i 35 V (preporuča se između 7 i 12 V). GND označava minus gdje se strujni krug zatvara. Za ispravan rad modula, potrebno je dovesti 5 V na pin 5V kako bi modul radio.



Slika 8: L298N DC motor driver – Prikaz pinova

2.1.3. Senzor udaljenosti – HC-SR04 Ultrasonic Sensor

Senzor udaljenosti ili ultrazvučni senzor je senzor za mjerenje udaljenosti pomoću ultrazvučnih valova. Glava senzora emitira ultrazvučni val koji se odbija od nekog predmeta ili objekta te se nazad vraća do glave senzora. Ultrazvučni senzor mjeri udaljenost objekta mjerenjem vremena između emisije i prijema, odnosno između slanja i primanja signala. Za ispravan rad senzora udaljenosti potrebno je pravilno spojiti četiri pina samog senzora. Vcc pin služi za napajanje senzora, a najčešće je to 5 V. Trigger pin se koristi za slanje signala, dok se Echo pin koristi za primanje signala. Ground pin se, kao i obično, spaja na minus to jest uzemljenje, a služi za zatvaranje strujnog kruga. Izgled senzora udaljenosti i njegovih pinova prikazan je na sljedećoj slici [Slika 9].

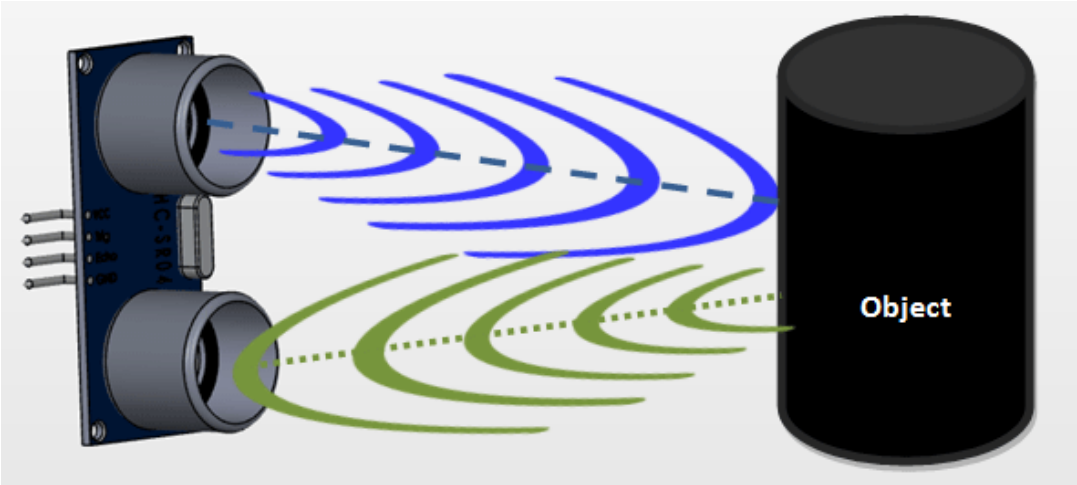


Slika 9: HC-SR04 Ultrasonic Sensor - Prikaz pinova

HC-SR04 ultrazvučni senzor udaljenosti, u teoriji, može izmjeriti udaljenost između 2 cm i 450 cm, dok je praksi ta udaljenost puno manja – od 2 cm do 80 cm. Senzor računa udaljenost s preciznošću od 3 mm. Za izračun udaljenosti između objekta i senzora koristi se formula:

$$L = \frac{1}{2} * T * C$$

gdje L predstavlja udaljenost, T vrijeme proteklo između slanja i primanja signala, te C koji predstavlja brzinu zvuka. Umnožak se množi s $\frac{1}{2}$ kako bi se dobila vrijednost udaljenosti za jedan smjer (a ne za oba). Vrijednost koja se uzima za brzinu zvuka je 330m/s.



Slika 10: princip rada ultrazvučnog senzora

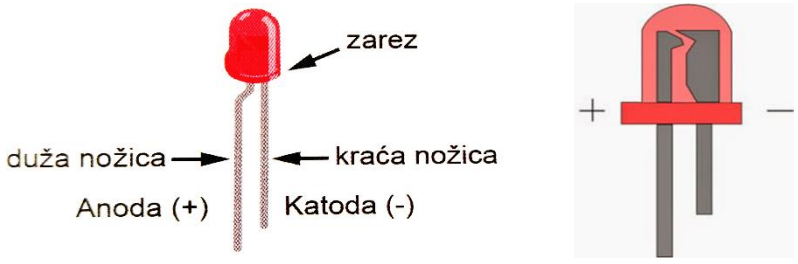
2.1.4. Zujalica

Zujalice su elektronički elementi predviđeni za generiranje zvuka. Postoje dvije vrste zujalica – aktivna i pasivna zujalica. Aktivna zujalica ima u sebi ugrađen oscilatorski krug, te ju je dovoljno spojiti na izvor napajanja i ona će proizvoditi zvuk određene frekvencije. Kod zujalice bez oscilatorskog kruga, to jest kod pasivne zujalice, potrebno je dovesti oscilirajući signal kako bi proizvela zvuk. Oscilirajući signal jednostavno je postići jer se programiranjem može odrediti početak i kraj generiranja zvuka, odnosno tona. Aktivna i pasivna zujalica se mogu razlikovati tako što aktivna zujalica ima crno dno, dok pasivna ima dno zelene boje. Njihovo spajanje se izvodi na način da se + (plus) zujalice spaja na neki od digitalnih pinova na Dasduino pločici, dok se – (minus) zujalice spaja na GND.

Slika 11: zujalice - Aktivna zujalica (lijevo) i Pasivna zujalica (desno)

2.1.5. Svjetleće diode

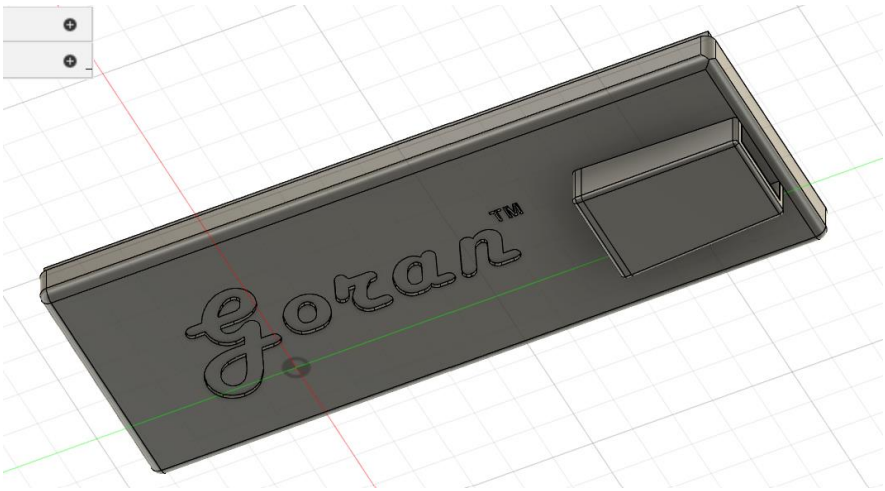
Naziv LED dolazi iz engleskog jezika i kratica je za Light Emitting Diode, što bi u prijevodu značilo dioda koja emitira svjetlost. Hrvatski naziv je svjetleća dioda, međutim u praksi se najčešće koristi izraz LE-dioda. LE-dioda ima dvije nožice od kojih je jedna duža, dok je druga kraća. Duža nožica se zove anoda i spaja se na + (plus), a kraća nožica, to jest katoda, spaja se na – (minus). Osim po duljini nožice, anodu je moguće prepoznati jer je u kućištu presjek (zarez) dviju nožica bliži anodi, odnosno površina katode u kućištu je veća od površine anode. Kako bi LE-dioda zasvijetlila potrebno ju je priključiti na napon od oko 2 V. Dakle, anoda se spaja na digitalni pin, dok se katoda spaja na GND. Međutim, svaki digitalni pin daje 5 V, zbog čega je potrebno dodati otpornik od 330 Ω u strujni krug kako LE-dioda ne bi pregorjela.



Slika 12: svjetleća dioda - Prepoznavanje anode i katode

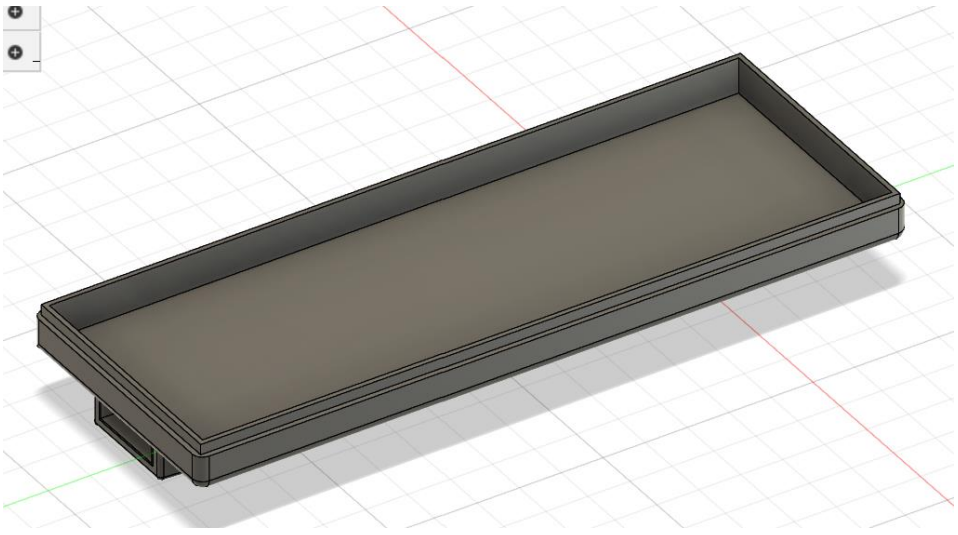
2.1.6. Maska odašiljača

U svrhu uljepšavanja i unaprjeđenja našeg proizvoda, koristili smo 3D printer kako bi isprintali masku za odašiljač radio signala. U svrhu toga koristili smo alat Fusion 360.



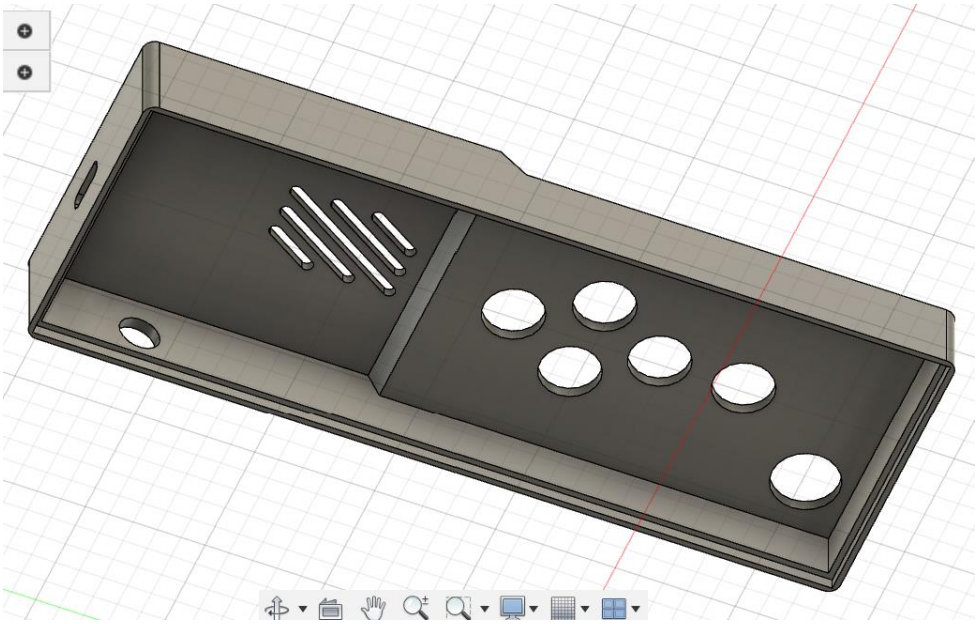
Slika 13: donji dio maske

Na slici gore moguće je vidjeti donji dio maske. On sadrži držač za bateriju i logo „Goran™“. Držač za bateriju napravljen je za bateriju koja će napajati odašiljač. Nakon printanja otvor je bio pre uzak te smo stoga trebali koristiti brusilicu kako bi ga proširili za oko pola milimetra. Nepreciznost 3D printera dovela je do užeg otvora nego očekivano.

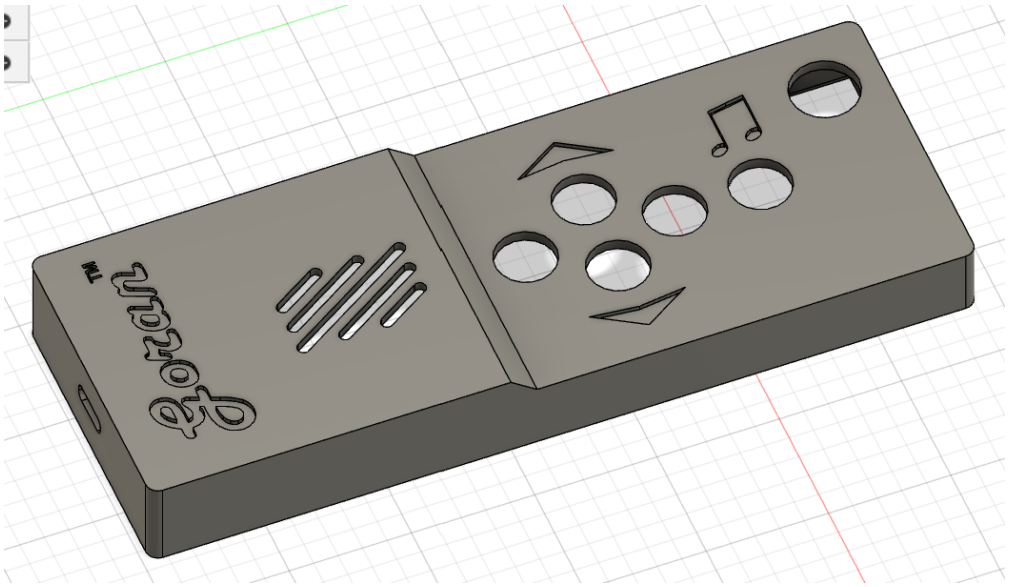


Slika 14: ptičja perspektiva na donji dio maske

Uz to, ovaj donji dio poklopca ima uvlaku na samom rubu, kao što je prikazano na slici gore. Svrha tome je da se spoji sa gornjim dijelom maske bez korištenja ljepila. Na taj način se baterija može izmijeniti nakon što se potroši. Tolerancija između uvlake gornjeg i donjeg dijela maske je ukupno 0,1 mm.



Slika 15: mravlja perspektiva na gornji dio maske

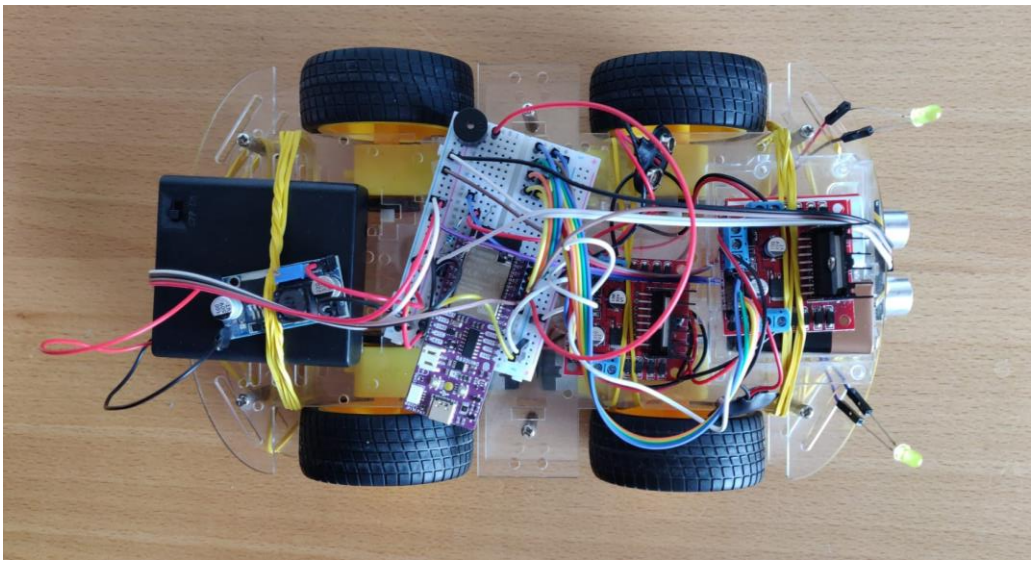


Slika 16: ptičja perspektiva na gornji dio maske

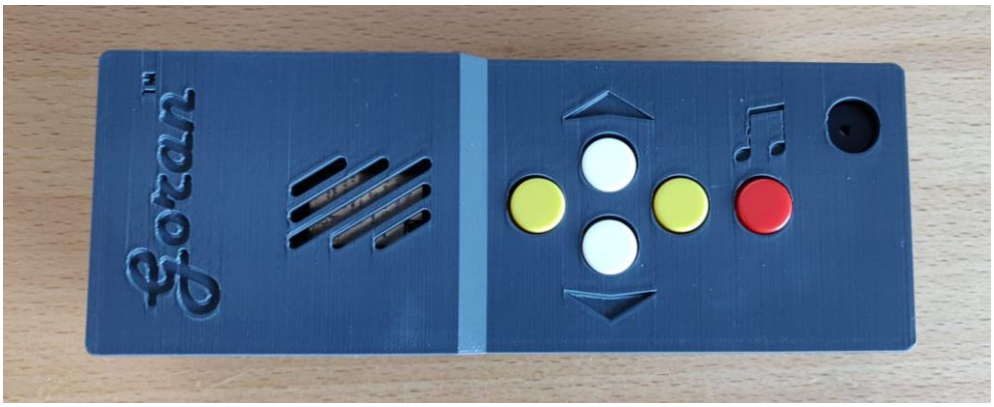
Gornji dio maske sadrži prostor za 5 gumba, rupu za zujalicu, uzdužene rupe za hlađenje mikroupravljača, rupu za USB-C konektor, rupu za spajanje baterije na mikroupravljač, sugestivne reljefe vezane uz funkcionalnosti gumbova i logo projekta. Gumbovi na odašiljaču imaju toleranciju od oko 2 mm od rupa koje su za njih izrađene.

2.2. Složena arhitektura sustava

Elektroničke komponente navedene u prošlim poglavljima, ovdje čine jednu zajedničku cjelinu, odnosno predstavljaju složeni sustav robota. Kako točno robot funkcionira, razmjenjuje podatke i prima naredbe biti će razrađeno u ostalim poglavljima, dok je cilj ovog poglavlja vizualno prikazati izgled robota. Shema spajanja će također biti predstavljena kasnije. Slijedeće slike [Slika 17], [Slika 18] služe za lakše razumijevanje poglavlja koja slijede, pritom pružajući čitatelju jasnu sliku o predmetu promatranja.



Slika 17: Prikaz složenog robota



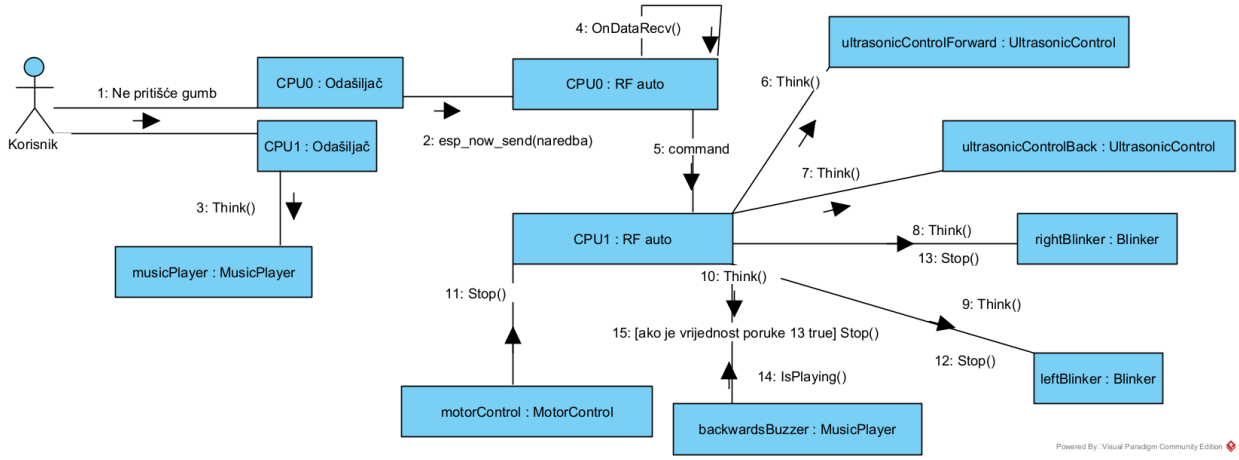
Slika 18: Prikaz daljinskog kontrolera (odašiljača)

2.3. Komunikacijski dijagrami

Prema [5] komunikacijski dijagram je tip UML dijagrama koji prikazuje interakciju između objekata, dijelova ili podsistema. Isključujući temporalnu komponentu, on daje iste informacije kao dijagram slijeda.

U sljedećih par dijagrama odnosit ću ne na gumbove „gore“, „dolje“, „lijevo“, „desno“ i „muzika“. To su 4 gumba nađena na odašiljaču. Kada se on usmjeri tako da su gumbovi bliži desnoj ruci, tada je vidljiv njihov točan poredak. Gumb „muzika“ je prepoznatljiv po silueti dvaju povezanih nota na kućištu odašiljača.

2.3.1. Korisnik ne pritišće niti jedan gumb



Slika 19: dijagram komunikacije kada korisnik ne pritišće niti jedan gumb

Na slici gore možete vidjeti kako komponente ovog robota međusobno komuniciraju u slučaju da korisnik ne pritišće niti jedan gumb. Mikrokontroler RF auta, kao i mikrokontroler odašiljača su tipa ESP32. To znači da sadrže dvije procesorske jedinice – CPU0 i CPU1. [6] CPU0 je zaslužan za procesiranje mrežnih zadataka, kao što je komuniciranje putem WiFi ili Bluetooth tehnologije. CPU1 je zaslužan za izvršavanje ostalog koda, kao što je upravljanje motorima i paljenje i gašenje LED-ica.

Poruka 2 predstavlja slanje paketa koji sadrži naredbu sa odašiljača na robot auto. U slučaju kada korisnik ne pritišće niti jedan gumb naredba koja se šalje je cijeli broj 0. Taj se

paket šalje sa CPU0 na odašiljaču na CPU0 na primatelju. Poruka 3. predstavlja dodjeljivanje procesorskog vremena objektu tipa *MusicPlayer*.

Poruka 4 i 5 predstavlja prijepis vrijednosti primljene naredbe u varijablu dijeljenu između CPU0 i CPU1 kod robot auta. *OnDataRecv()* je zapravo funkcija pozvana nakon primanja paketa.

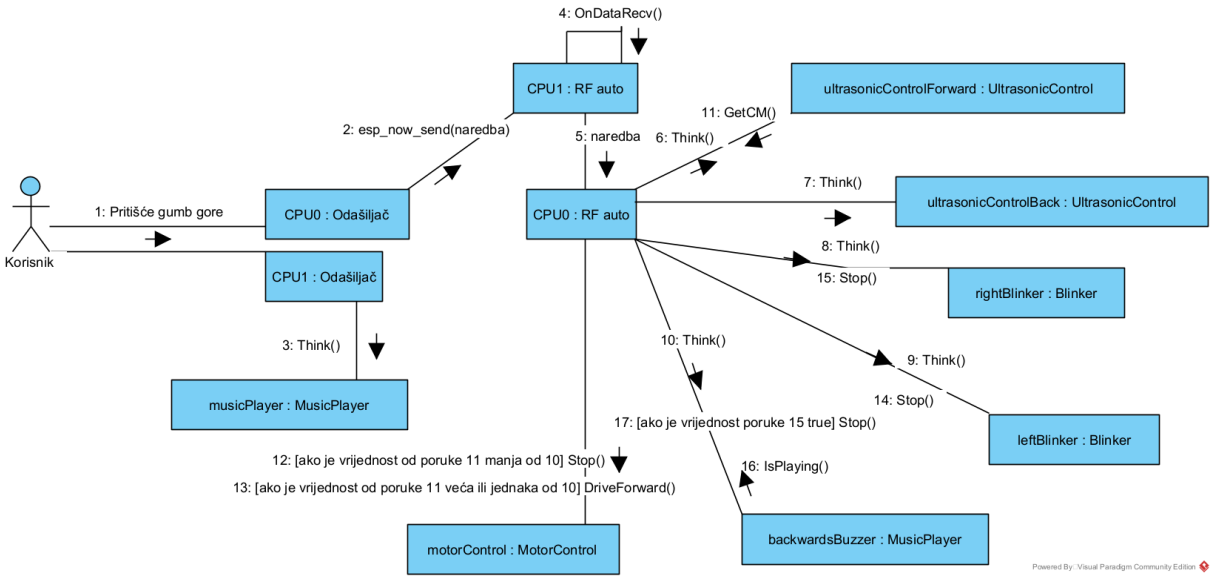
Poruke 6, 7, 8, 9 i 10 se odnose na dodjeljivanje objektima *ultrasonicControlForward*, *ultrasonicControlForward*, *rightBlinker*, *leftBlinker* i *backwardsBuzzer* procesorsko vrijeme. Svrha toga je održavanje pravilnog vremena treptanja LED-ica, pravog vremena i duljine sviranja nota na zujalici, provjeravanje udaljenosti prednjeg i stražnjeg branika robot auta od drugih objekata.

U slučaju kada se niti jedan gumb ne pritišće naredba kojom se upravljaju motori je *Stop()*. Ona se šalje motorima koji upravljaju kotačima robota. To je vidljivo putem poruke 11.

Poruke 12 i 13 se odnose na gašenje LED-ica. Naime, one bi trebale biti upaljene samo kada robot auto skreće lijevo ili desno.

Poruka 14 daje informaciju je li upaljena zujalica koja signalizira vožnju u natrag. Ako je, treba ju ugasiti. To se radi pomoću naredbe 15.

2.3.2. Korisnik pritišće gumb gore



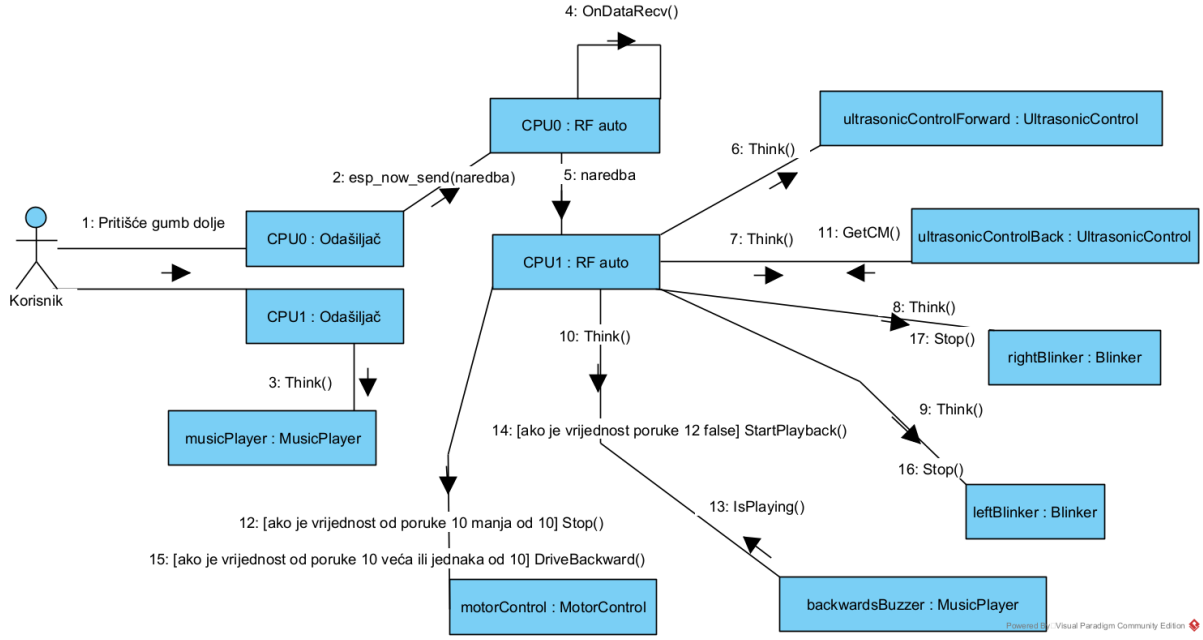
Slika 20: dijagram komunikacije kada korisnik pritišće gumb gore

Na slici gore možete vidjeti kako komponente ovog robota međusobno komuniciraju u slučaju da korisnik pritišće gumb gore. Poruke 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 su iste kao i na prošlom dijagramu.

Svrha poruke 11 je prikupljanje informacije koliko je prednji branik robot auta udaljen od sljedećeg objekta. Ta vrijednost je zatim korištena u porukama 12 i 13 gdje se njihova pojava bazira na toj vrijednosti. Vrijednost 10 predstavlja 1 centimetara od sljedećeg objekta. To je udaljenost na kojoj bi robot auto trebao blokirati vožnju dalje.

Poruke 14 i 15 identične su porukama 12 i 13 na prošlom dijagramu. Poruke 16 i 17 identične su porukama 14 i 15 na prošlom dijagramu.

2.3.3. Korisnik pritišće gumb dolje



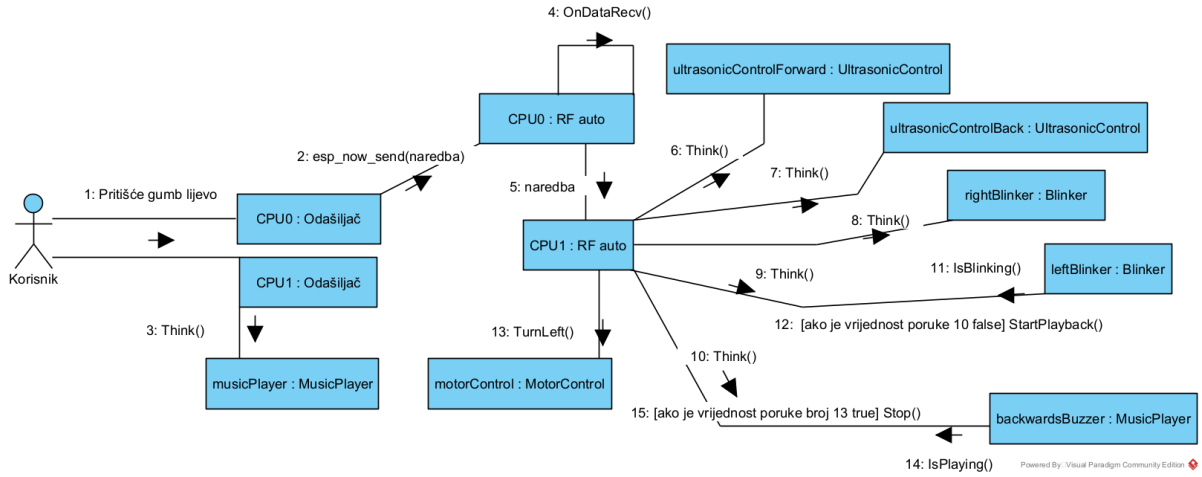
Slika 21: dijagram komunikacije kada korisnik pritišće gumb dolje

Na slici gore možete vidjeti kako komponente ovog robota međusobno komuniciraju u slučaju da korisnik pritišće gumb dolje. Poruke 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 su iste kao i na prošlom dijagramu.

Svrha poruke 11 je prikupljanje informacije koliko je prednji branik robot auta udaljen od sljedećeg objekta. Ta vrijednost je zatim korištena u porukama 12 i 13 gdje se njihova pojava bazira na toj vrijednosti. Vrijednost 10 predstavlja 1 centimetara od sljedećeg objekta. To je udaljenost na kojoj bi robot auto trebao blokirati vožnju dalje.

Poruke 14 i 15 identične su porukama 12 i 13 na prošlom dijagramu. Poruke 16 i 17 identične su porukama 14 i 15 na prošlom dijagramu.

2.3.4. Korisnik pritišće gumb lijevo



Slika 22: dijagram komunikacije kada korisnik pritišće gumb lijevo

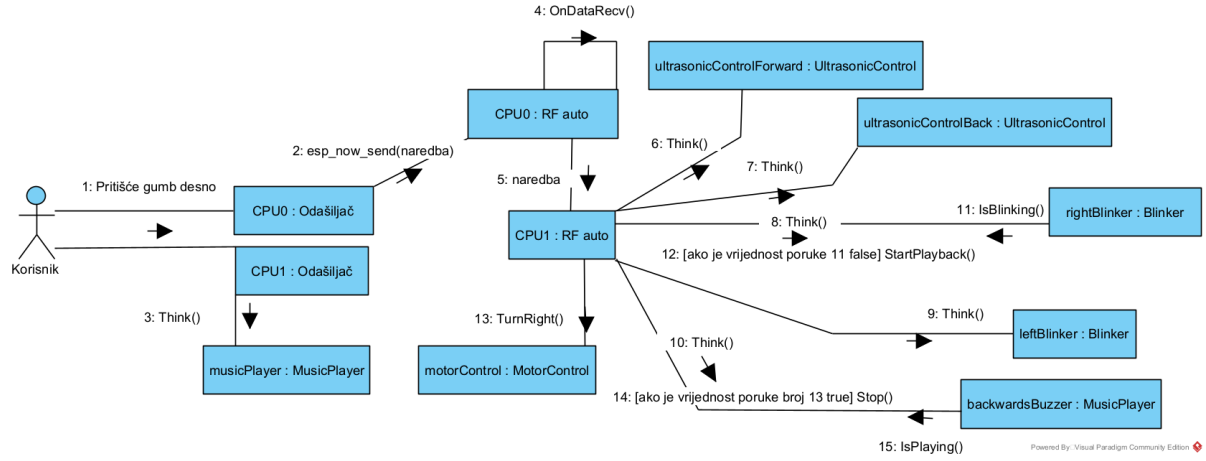
Na slici gore možete vidjeti kako komponente ovog robota međusobno komuniciraju u slučaju da korisnik pritišće gumb dolje. Poruke 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 su iste kao i na prošlom dijagramu.

Poruka 11 provjerava svijetli li lijevi žmigavac. Ako ne svijetli, treba ga uključiti, što i prikazuje poruka broj 12.

Poruka 13 naređuje motorima da skreću lijevo.

Poruke 14 i 15 isključuju zujalicu ako je uključena.

2.3.5. Korisnik pritišće gumb desno



Slika 23: dijagram komunikacije kada korisnik pritišće gumb desno

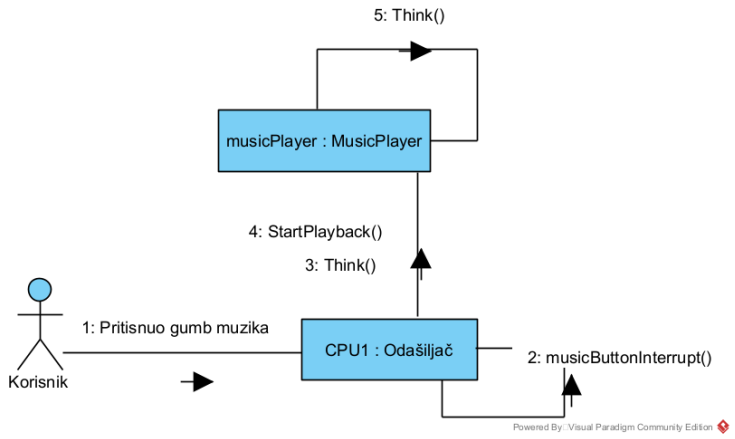
Na slici gore možete vidjeti kako komponente ovog robota međusobno komuniciraju u slučaju da korisnik pritišće gumb desno. Poruke 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 su iste kao i na prošlom dijagramu.

Poruka 11 provjerava svijetli li desni žmigavac. Ako ne svijetli, treba ga uključiti, što i prikazuje poruka broj 12.

Poruka 13 naređuje motorima da skreću desno.

Poruke 14 i 15 isključuju zujalicu ako je uključena.

2.3.6. Korisnik pritisne gumb muzika



Slika 24: dijagram kada korisnik pritišće gumb muzika

Na slici gore možete vidjeti kako komponente ovog robota međusobno komuniciraju u slučaju da korisnik pritisne gumb muzika. Nakon pritiska gumba se pozove prekid u obliku funkcije *musicButtonInterrupt()*. Svrha tog prekida je postavljanje zastavice za prijelaz na sljedeću pjesmu. U sebi ima sadržan tzv. *switch debouncing* sa odgodom detekcije od 200 milisekundi. Korak 3 izvršava se neovisno o tom pritisku – u svakom slučaju. Zbog pritiska gumba muzika aktivira se korak 4. Tada se objektu *musicPlayer* pošalje naredba StartPlayback() koja završi sa izvađanjem trenutne i krene sa izvađanjem sljedeće pjesme.

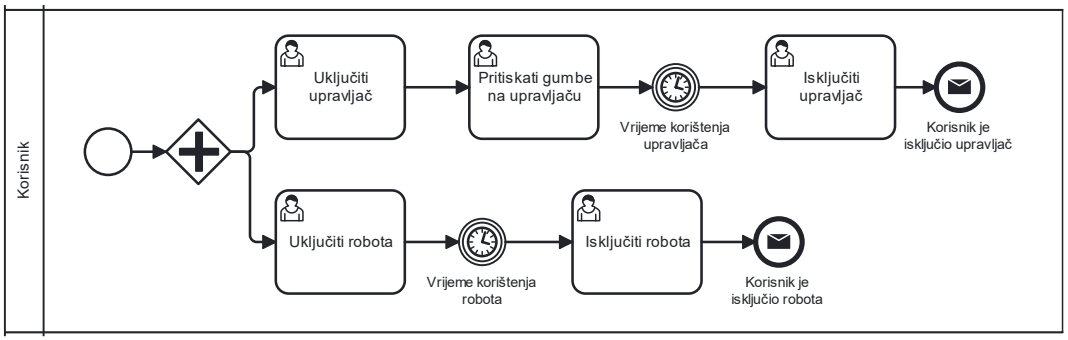
Poruka broj 5 je pozivanje metode *Think()* unutar same klase. Svrha toga je dodjeljivanje procesorskog vremena toj klasi.

2.4. Dijagram aktivnosti

Prema [7] dijagrami aktivnosti opisuju kako su aktivnosti koordinirane. To prikazuje na različitim razinama apstrakcije. Kako bi izgradili model aktivnosti koristili smo notaciju BPMN. Ona pruža puno više mogućnosti u smislu definiranja vrste prekida, vrste događaja, vrste grananja i slično.

Cijeli dijagram je podijeljen na 3 staze: stazu korisnik, stazu upravljača-odašiljača i stazu prijemnika-robota.

2.4.1. Staza korisnika

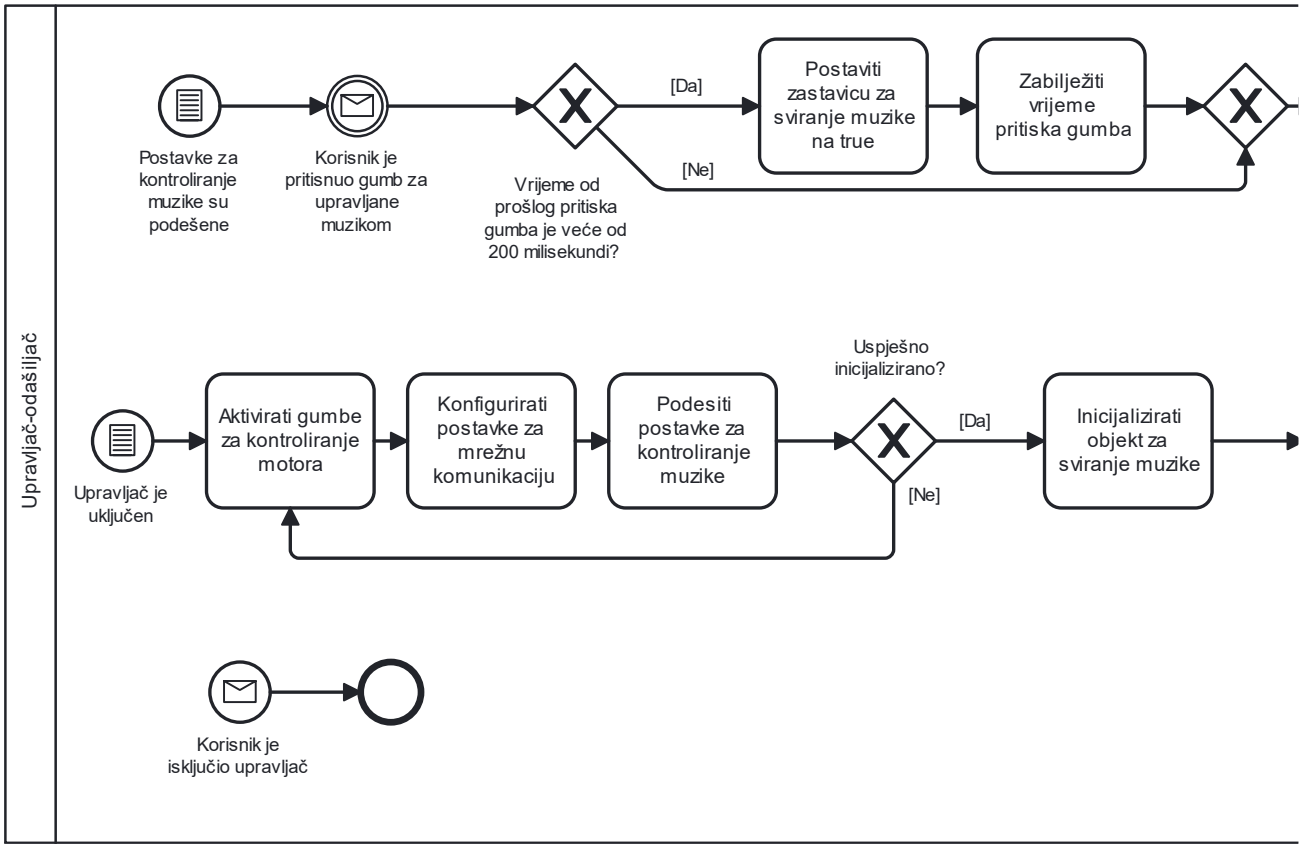


Slika 25: staza korisnika dijagrama aktivnosti

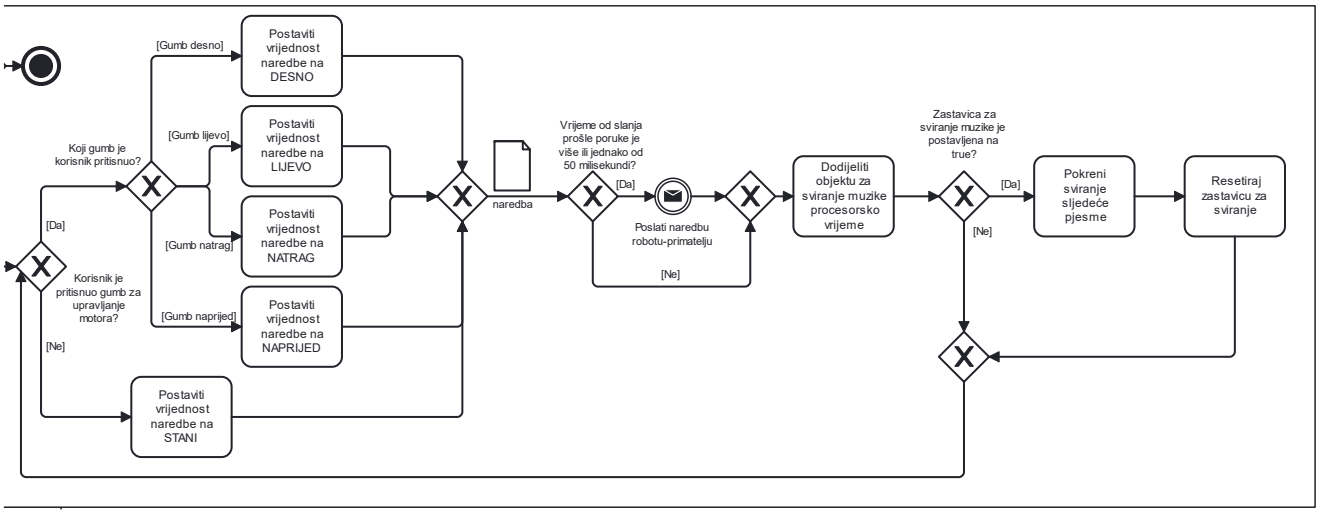
Aktivnosti korisnika se dijele na dva slijeda – slijed uključivanja upravljača i slijed uključivanja robota. Nakon što korisnik izvede aktivnost uključivanja robota on njime upravlja. Nakon što je gotov isključuje upravljač. To šalje poruku u stazu upravljača.

Nakon što prođe neko vrijeme kroz koje korisnik koristi robota on ga isključuje i šalje se poruka stazi robota.

2.4.2. Staza upravljača



Slika 26: staza upravljača dijagrama aktivnost – 1 od 2



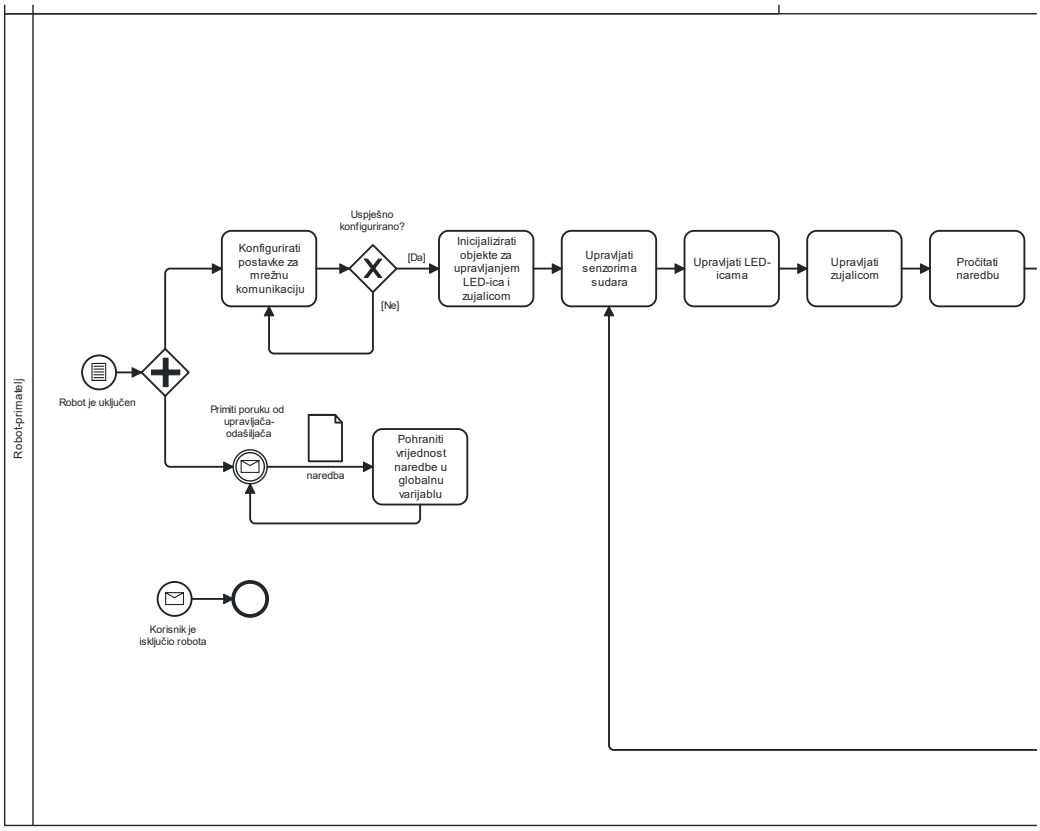
Slika 27: staza upravljača dijagrama aktivnosti – 2 od 2

Slike 26 i 27 prikazuju stazu upravljača. Početak izvođenja slijeda su 3 uvjeta: uvjet za uključivanjem upravljača, uvjet za podešavanjem postavkama muzike i uvjet za isključenjem upravljača. Ovaj dijagram slijeda prati izvorni kod koji izgrađuje funkcionalnosti upravljača. Zbog toga postoji duga povratna veza u srednjem slijedu upravljača – aktivnosti unutar te povratne veze se nalaze u *loop()* funkciji. Svako ekskluzivno grananje sa *da/ne* rezultatima predstavlja *if* grananje. Grananje provjere koji je gumb pritisnut predstavlja *if else* slijed grananja.

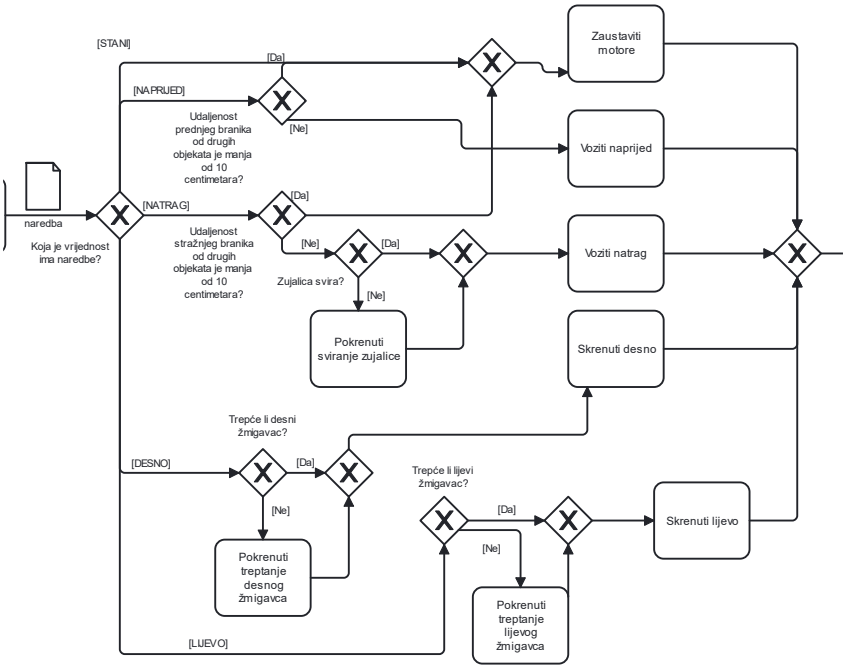
Gornji slijed se izvršava svaki put kada korisnik pritisne gumb za upravljanje muzikom ako su postavke za kontroliranje podešene. Terminirajući događaj terminira samo taj slijed, a ne i izvođenje ostalih sljedova. S druge strane, kada se isključi upravljač, svi sljedovi se isključuju.

Događaj slanja naredbe na slici 27 ilustrira mrežnu komunikaciju između upravljača-odašiljača i robota-primatelja poruka.

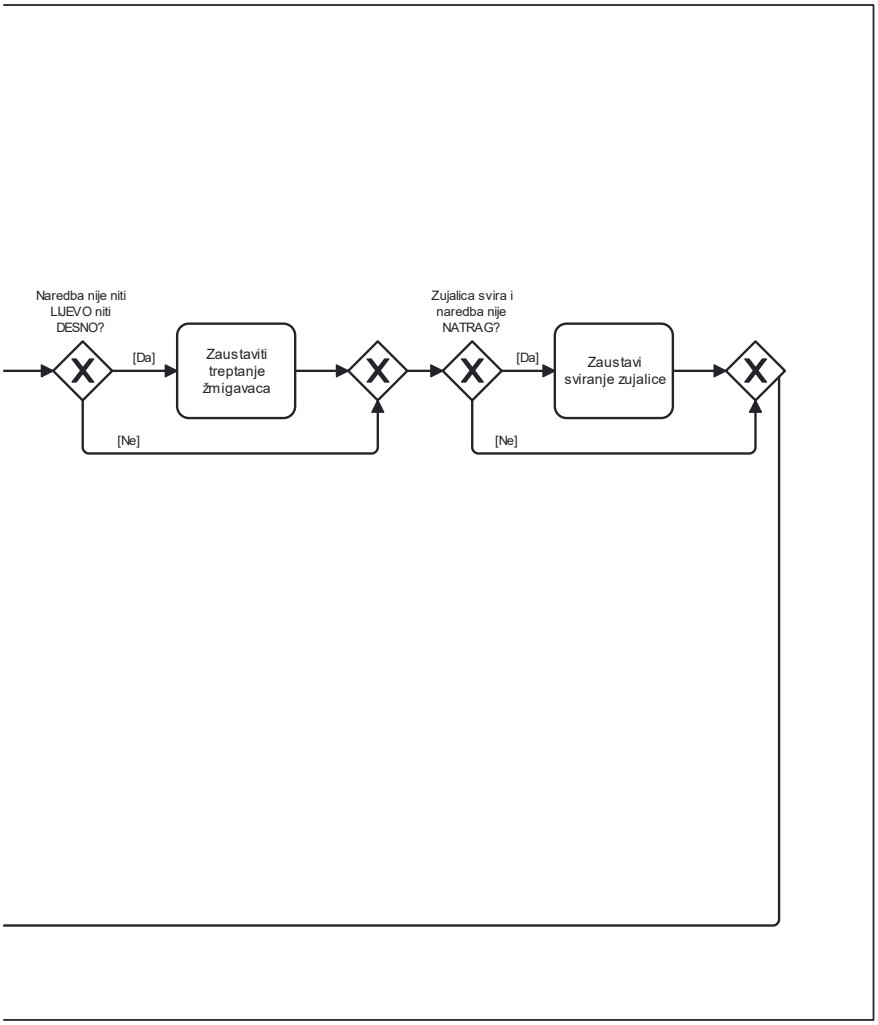
2.4.3.Staza robota



Slika 28: staza robota dijagrama aktivnosti - 1 od 3



Slika 29: staza robota dijagrama aktivnosti - 2 od 3



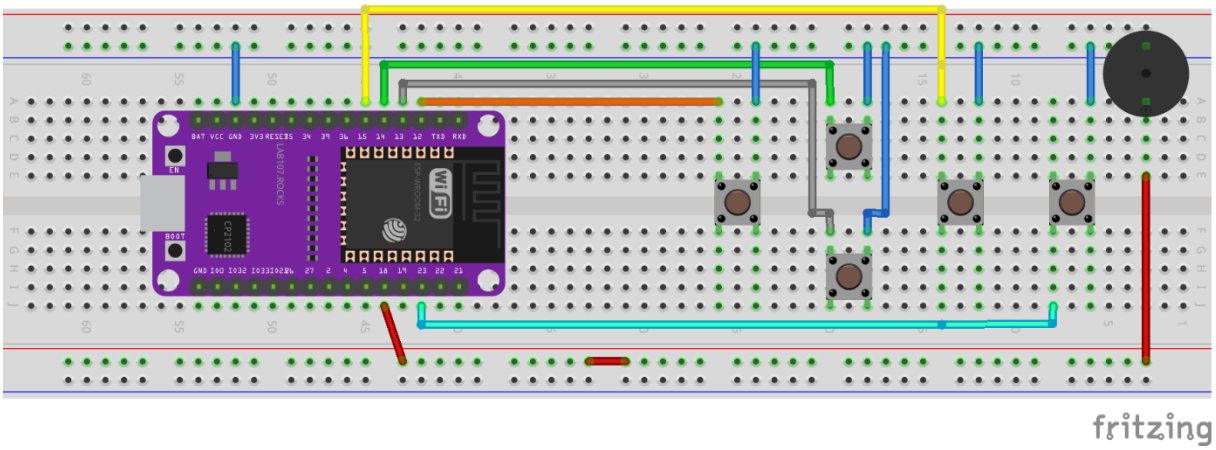
Slika 30: staza robota dijagrama aktivnosti – 3 od 3

Kada se robot uključi krene se izvršavati jedan slijed aktivnosti. Drugi slijed se krene izvršavati kada korisnik isključi robota. Slijed nakon uključivanja robota podrazumijeva izvršavanje 2 paralelna slijeda: 1 koji predstavlja izvođenje setup() i loop() funkcija na CPU1 jezgri ESP32 mikrokontrolera i 1 koji predstavlja izvršavanje ESP NOW mrežnog protokola na CPU0 jezgri. Događaj primanja poruke od upravljača sadrži poruku naredbe koja se šalje. U slučaju gašenja robota terminiraju se svi sljedovi na stazi.

Ponovno, postoji dugačka povratna veza jer aktivnosti u njoj predstavljaju izvršavanje *loop()* funkcije. Ekskluzivno grananje koje provjerava vrijednost naredbe predstavlja *switch* ključnu riječ u kodu. Ponovno, svako ekskluzivno grananje sa *da/ne* rezultatima predstavlja *if* grananje.

2.5. Shema ili dijagram spajanja

U nastavku su prikazani dijagrami spajanja odašiljača i robota. Iz prikazanog je vidljivo da spajanje odašiljača nije kompleksno. Na pinovima 12-15 su spojeni gumbovi koji određuju kratanje robota, odnosno signal koji odašiljač šalje. Na pin 23 je spojen gumb koji kontrolira sviranje, dok je zujalica spojena na pin 16.



Slika 31: shema spajanja odašiljača

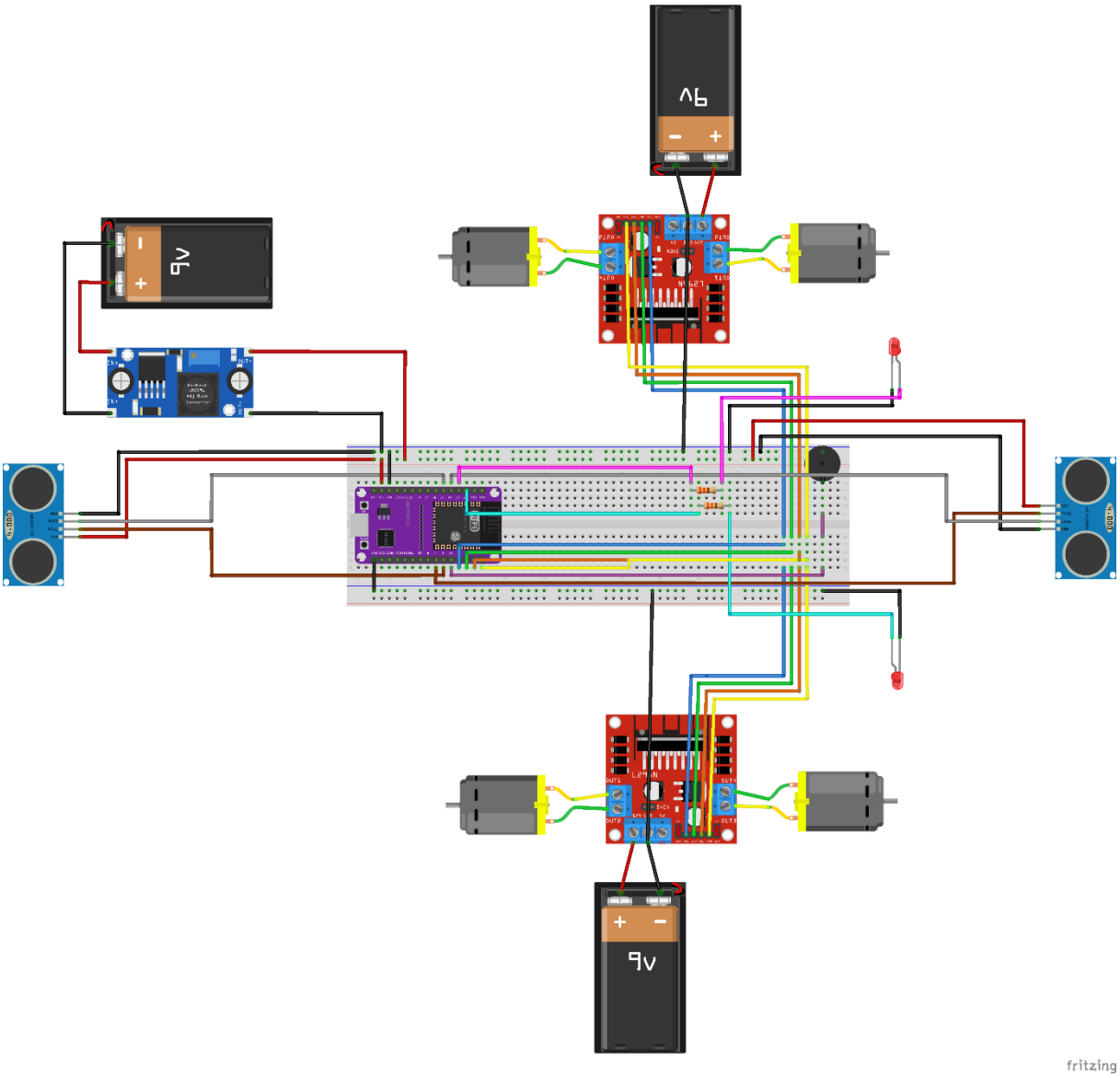
S druge strane shema robota je nešto kompleksnija pa će se opisati u dijelovima. Kao prvi dio treba navesti napajanje same pločice koje se vrši na način da se bateriju od 9V spoji na DC adapter koji napon spušta na 5V s kojima pločica i ostali senzori mogu sigurno raditi. Izlaz adaptera je spojen na breadboard na dijelu za napajanje koji se proteže duž cijelog breadborda. Pločica se spaja na spomenuto napajanje putem Vcc pina.

Drugi dio robota su senzori udaljenosti koji su spojeni na napajanje, Echo pinovi su spojeni na pinove 14 i 15, dok su Trig pinovi spojeni na pinove 4 i 5 koji pružaju opciju analognog inputa koja je potrebna.

Žmigavci robota, odnosno LED diode su spojene na uzemljenje dok su anode spojene preko otpornika na pinove 12 i 13. A zujalica je spojena na uzemljenje i na pin 18.

Na kraju su ostali motor driveri koji su spojeni na način da se kontroliraju istim pinovima. Svaki motor driver je postavljen na način da upravlja jednim lijevim i jednim desnim motorom, tako da se motor driveri cijelo vrijeme ponašaju jednako. Oni rade na način da imaju vlastito napajanje spojeno na bateriju od 9V, ulazi za upravljanje su spojeni na pinove 19, 21, 22 i 23 pločice te paralelno na ulaze za upravljanje drugog motor drivera, a izlazi kojih ima 4 spojeni su u parovima za svaki od 2 motora. ENA i ENB pinovi koji služe za određivanje brzine vrtnje

motora su kratko spojeni tako da se motori cijel vrijeme vrte maksimalnom brzinom. Kako bi pločica mogla upravljati motor driverima potrebno im je spojiti uzemljenje. Na taj način pločica samo upravlja motor driverima koji koriste vlastito napajanje za pokretanje motora.



Slika 32: shema spajanja robota

3. Izvršni program robota

Nakon spajanja robota, potrebno je napisati program po kojemu će robot raditi. Program je pisan pomoću alata Visual Studio Code, u programskom okruženju PlatformIO IDE i potom je učitan u Dasduino CONNECTPLUS pločicu preko USB kabla. Na sljedećim stranicama prikazani su dijelovi programa skupa s njihovim objašnjenjem.

Izvršni kod je podijeljen na dva programa: “Robot-Primatelj” i “Upravljač-Odašiljač”. Robot-Primatelj upravlja robotom i njihovim komponentama a Upravljač-Odašiljač a drugi program obrađuje slanje ulaza robotu i sviranje muzike.

3.1. Klase

Naše rješenje mora pratiti i upravljati više uređajima i funkcijama istovremeno, kako bi nam taj posao bio lakše pratiti, izradili smo par klasa koje upravljaju pojedinoj vrsti uređaja.

CBlinker i **CMusicPlayer** rade na sličnom principu, samo da jedan upravlja LED-icama a drugi Buzzer-om, tako da ćemo objasniti samo jedan pa onda istaknuti razlike kada potrebno.

Klase se sastoje od 5 metoda:

- Init – Poziva se na početku programa i inicijalizira klasu, postavlja koji PIN se koristi za funkcije klase te istog postavlja na ispravni način rada

```
void CBlinker::Init( int iLEDPin ) {
    m_pBlinkPattern = NULL;
    m_iLEDPin = iLEDPin;
    pinMode(m_iLEDPin, OUTPUT);
}
```

- StartPlayback – Glavna metoda korisničke strane, kao ulaz uzima polje koje predstavlja uzorak koji će se reproducirati, duljina polja, brzinu sviranja i zastavicu da li će se uzorak ponovno pokrenuti ili ne Metoda sprema postavke i započinje reprodukciju

```
void CBlinker::StartPlayback( const int *pattern, int length, float playback_speed, bool loop = false ) {
    length /= sizeof(int);
    // No pattern, dont blink
    if( !pattern )
        return;

    // Lenght invalid, don't blink
    if( length % 2 != 0 )
        return;

    if( m_pBlinkPattern )
        delete[] m_pBlinkPattern;

    m_pBlinkPattern = new int[length];
    memcpy( m_pBlinkPattern, pattern, length * sizeof(int) );

    m_iCurrentPos = 0;

    m_iPatternLength = length;

    m_flPlaybackSpeed = playback_speed;

    m_iNextPatternAt = millis();

    m_bLoop = loop;

    // Do first think immediately when we start the blinker!
    Think();
}
```

- Think – Ova metoda se mora pozvati svaki ciklus u programu, ne radi ništa kada se muzika ne mora reproducirati, no kada smo pokrenuli reprodukciju pomoću StartPlayback metode, ova metoda pomiče uzorak na sljedeći korak kada vrijeme trenutne akcije završi. Kod Blinkera, akcija je uključivanje/isključivanje LED-ice, a kod Buzzera, to je sviranje određene note

```
void CBlinker::Think() {
    if( !m_pBlinkPattern )
        return;

    if( m_iNextPatternAt > millis() )
        return;

    int pattern_time = m_pBlinkPattern[m_iCurrentPos + 1];
    int blink_time = 0;
    if( m_flPlaybackSpeed > 0 ) {
        blink_time = pattern_time * m_flPlaybackSpeed;
    }

    digitalWrite(m_iLEDPin, m_pBlinkPattern[m_iCurrentPos]);

    m_iNextPatternAt = millis() + blink_time;

    m_iCurrentPos += 2;

    if( m_iCurrentPos >= m_iPatternLength ) {
        if( m_bLoop )
            m_iCurrentPos = 0;
        else {
            Stop();
        }
    }
}
```

- Stop – Ova metoda trajno zaustavlja reprodukciju, automatski se poziva ako je petljanje isključeno, u slučaju da nije, ova metoda se treba ručno pozvati kada želimo da se reprodukcija zaustavi

```
void CBlinker::Stop() {
    digitalWrite(m_iLEDPin, LOW);

    if( m_pBlinkPattern ) {
        delete[] m_pBlinkPattern;
        m_pBlinkPattern = NULL;
    }
    m_iCurrentPos = 0;
}
```

- IsBlinking/IsPlaying – Ova metoda vrača true ako je trenutna reprodukcija aktivna, a false ako nije, što znamo ako je postavljen uzorak

```
bool CBlinker::IsBlinking() {
    return m_pBlinkPattern != NULL;
}
```

Glavne razlike između CBlinker i CMusicPlayer klasa je način izračuna vremena između koraka i signal koji se šalje na pin. CBlinker koristi milisekunde i HIGH/LOW signal, a CMusicPlayer koristi bpm i muzičku notaciju za određivanje vremena svakog koraka, i signal koji šalje je frekvencija Buzzer-a, koja predstavlja notu.

Izračun vremena za muziku, frekvencija nota i pjesme su preuzete s github projekta otvorenog koda [10].

Ultrazvučni senzor treba obavezno barem 10 milisekundi skenirati prije svakog učitavanja, a pošto mi trebamo jako često ažurirati udaljenost radi kretanja robota, moramo imati način da to činimo konstantno i konzistentno, što radimo pomoću **CUltrasonicControl** klase.

Klasa radi na sličnom principu kao CBlinker i CMusicPlayer gdje imamo Think metodu koja se odvija svaki ciklus.

- Konstruktor – Postavlja pinove ultrazvučnog senzora i početne postavke kao što su trenutačna duljina i trenutačna akcija

```
CUltrasonicControl::CUltrasonicControl( int trigPIN, int echoPIN ) {
    m_iTrigPIN = trigPIN;
    m_iEchoPIN = echoPIN;

    pinMode(trigPIN, OUTPUT);
    pinMode(echoPIN, INPUT);

    m_bTask = SENDING;
    m_dLenght = 0.0f;
    // Change our task in 10 milis, since that's how long we have to send for
    m_dChangeTaskAtTime = millis() + 10;
}
```

- Think – Glavna metoda, odvija se svaki ciklus
  Kada je akcija postavljena na “SENDING” tj. slanje, pišemo na ultrazvučni senzor kako bi slao zvučne valove i učitavao udaljenost, ova akcija traje 10 milisekundi, nakon kojih se prebacimo na akciju “MEASURING”
  Kada je akcija postavljena na “MEASURING”, postavimo Ultrazvučni senzor na “LOW”, te nakon toga učitamo udaljenost koju je izmjerio i spremimo ga u varijablu m_dLenght, ova akcija traje samo jedan ciklus, te se odmah prebaci nazad na “SENDING” u sljedećem ciklusu

```
void CUltrasonicControl::Think() {
    if( m_dChangeTaskAtTime <= millis() ) {
        // Swap the current task to the next one
        if( m_bTask == SENDING )
            m_bTask = MEASURING;
        else if( m_bTask == MEASURING )
            m_bTask = SENDING;

        // If we are sending, change the task in 10 miliseconds, since we need to measure for that long
        if( m_bTask == SENDING )
            m_dChangeTaskAtTime = millis() + 10;
        // If we are measuring, change the task on the next loop immediately
        else if( m_bTask == MEASURING )
            m_dChangeTaskAtTime = millis();
    }

    // To send with the ultrasonic sensor, we have to set it to high
    if( m_bTask == SENDING ) {
        digitalWrite(m_iTrigPIN, HIGH);
    }
    else if( m_bTask == MEASURING ) {
        // When we want to measure the lenght, we first have to set our sensor to LOW
        digitalWrite(m_iTrigPIN, LOW);

        // Read the signal from the sensor: a HIGH pulse whose
        // duration is the time (in microseconds) from the sending
        // of the ping to the reception of its echo off of an object.
        m_dLenght = pulseIn(m_iEchoPIN, HIGH);
    }
}
```

- GetCM – Ova metoda nam vraća zadnju izmjerenu udaljenost pretvorenu u centimetre

```
double CUltrasonicControl::GetCM() {
    // duration is the time it takes from sending the signal to recieving it
    // Divide by 29.1 or multiply by 0.0343 to convert from ms to cm
    return (m_dLenght/2) / 29.1;
}
```

Zadnja klasa koju koristimo je **CMotorControl** koja služi za kretanje kotača.

- Konstruktor – Uzima 4 pinova koja predstavljaju kotače

```
CMotorControl::CMotorControl(int in1, int in2, int in3, int in4){
    m_iRightWheelsForward = in1;
    m_iRightWheelsBack = in2;
    m_iLeftWheelsForward = in3;
    m_iLeftWheelsBack = in4;
}
```

- MotorSetup – Uključuje ispravni pin mode i postavlja pinove na početno stanje

```
void CMotorControl::MotorSetup() {
    // setting up motor control pins
    pinMode(m_iRightWheelsForward, OUTPUT);
    pinMode(m_iRightWheelsBack, OUTPUT);
    pinMode(m_iLeftWheelsForward, OUTPUT);
    pinMode(m_iLeftWheelsBack, OUTPUT);

    // setting motor control pin outputs
    digitalWrite(m_iRightWheelsForward, LOW);
    digitalWrite(m_iRightWheelsBack, LOW);
    digitalWrite(m_iLeftWheelsForward, LOW);
    digitalWrite(m_iLeftWheelsBack, LOW);
}
```

- DriveForward – Postavlja kretanje kotača prema naprijed
- DriveBackward – Postavlja kretanje kotača prema iza
- TurnLeft – Postavlja desne kotače prema naprijed i lijeve prema nazad kako bi se okretali prema lijevo
- TurnRight – Postavlja desne kotače prema nazad i lijeve prema naprijed kako bi se okretali prema desno
- Stop – Zaustavlja sve kotače

```
void CMotorControl::DriveForward(){
    digitalWrite(m_iRightWheelsForward, HIGH);
    digitalWrite(m_iRightWheelsBack, LOW);
    digitalWrite(m_iLeftWheelsForward, HIGH);
    digitalWrite(m_iLeftWheelsBack, LOW);
}
```

```
void CMotorControl::DriveBackward(){
    digitalWrite(m_iRightWheelsForward, LOW);
    digitalWrite(m_iRightWheelsBack, HIGH);
    digitalWrite(m_iLeftWheelsForward, LOW);
    digitalWrite(m_iLeftWheelsBack, HIGH);
}
```

```
void CMotorControl::TurnLeft(){
    digitalWrite(m_iRightWheelsForward, HIGH);
    digitalWrite(m_iRightWheelsBack, LOW);
    digitalWrite(m_iLeftWheelsForward, LOW);
    digitalWrite(m_iLeftWheelsBack, HIGH);
}
```

```
void CMotorControl::TurnRight(){
    digitalWrite(m_iRightWheelsForward, LOW);
    digitalWrite(m_iRightWheelsBack, HIGH);
    digitalWrite(m_iLeftWheelsForward, HIGH);
    digitalWrite(m_iLeftWheelsBack, LOW);
}
```

```
void CMotorControl::Stop(){
    digitalWrite(m_iRightWheelsForward, LOW);
    digitalWrite(m_iRightWheelsBack, LOW);
    digitalWrite(m_iLeftWheelsForward, LOW);
    digitalWrite(m_iLeftWheelsBack, LOW);
}
```

3.2. Upravljač-Odašiljač

Početak programa nam uključuje glavnu funkciju upravljača, što je Wi-Fi interface, te ga postavlja kao stanicu i dodaje adresu našeg robota kao odredište. Nakon što je Wi-Fi postavljen, uključujemo pinove koje koristimo na upravljaču, to su “input” pinovi koje spajaju naše gumbove za kretanje, pin za buzzer i gumb koji uključuje njegovo sviranje te inicijaliziramo klasu koja upravlja sviranjem muzike.

```
void setup() {
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // initialise ESP NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register peer
  memcpy(g_peerInfo.peer_addr, g_iRecieverAdress, 6);
  g_peerInfo.channel = 0;
  g_peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&g_peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  for(int i=0; i<4; i++){
      pinMode(g_iBtnPINs[i], INPUT_PULLUP);
  }

  pinMode( g_iMusicPin, OUTPUT );
  pinMode( g_iMusicButtonPin, INPUT_PULLUP );

  attachInterrupt(g_iMusicButtonPin, MusicButtonInterrupt, RISING);

  g_musicPlayer.Init(g_iMusicPin);
}
```

Loop funkcija programa ciklički učitava vrijednosti na našim gumbima, te ovisno o njihovim vrijednostima, pošalje poruku o kretnji. Gumbi su međusobno isključivi s hijerarhijom, što znači ako je jedan gumb pritisnut, ostali neće biti detektirani, što smanjuje količinu podataka koju moramo poslati. Nakon što je poslana poruka, obavljamo cikličnu funkciju CMusicPlayer-a i detektiramo ako nam je gumb za muziku bio pritisnut, u slučaju da je, počinjemo svirati trenutnu pjesmu i postavimo sljedeću koja će se svirati.

```
void loop() {

  if(!digitalRead(g_iBtnPINs[3])){ // if RIGHT
    g_messageToSend.m_iCommand = 4;
  } else if(!digitalRead(g_iBtnPINs[2])){ // if LEFT
    g_messageToSend.m_iCommand = 3;
  }else if(!digitalRead(g_iBtnPINs[1])){ // if DOWN
    g_messageToSend.m_iCommand = 2;
  }else if(!digitalRead(g_iBtnPINs[0])){ // if UP
    g_messageToSend.m_iCommand = 1;
  }else{
    g_messageToSend.m_iCommand = 0;
  }

  //Serial.println(messageToSend.command);

  //ESP NOW send the message and get the result
  if( millis() - g_flLastSentTime > 50 ){
    //Serial.println("sending " + g_messageToSend.m_iCommand);
    g_flLastSentTime = millis();

    esp_err_t result = esp_now_send(g_iRecieverAdress, (uint8_t *) &g_messageToSend, sizeof(g_messageToSend));
  }


  g_musicPlayer.Think();

  if( g_bMusicButtonPressed ) {
      g_musicPlayer.StartPlayback(g_szSongs[g_iCurrentSong], g_iSongSize[g_iCurrentSong], g_iSongTempos[g_iCurrentSong]);

      g_iCurrentSong++;

      if( g_iCurrentSong >= g_iSongCount )
        g_iCurrentSong = 0;

      g_bMusicButtonPressed = false;
  }
}
```

Prije slanja svakog paketa provjerava se je li od zadnjeg poslanog paketa prošlo barem 50 milisekundi. Kada ne bi to osigurali, procesor odašiljača bi se vrlo brzo pregrijao i prestao pravilno funkcionirati.

3.3. Robot-Primatelj

Početak programa uključuje naš Wi-Fi i postavlja ga kao stanicu, dodaje odašiljač kao dio mreže, te nas postavlja kao slušatelj u mreži. Nakon postavljanja Wi-Fi-ja, uključujemo kontroler kotača, postavljajmo lijevi i desni žmigavac, te postavljamo svirač upozorenja kretanja unatrag.

```
void setup() {
  Serial.begin(115200);
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

   // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register peer
  memcpy(g_peerInfo.peer_addr, g_iSenderAddress, 6);
  g_peerInfo.channel = 0;
  g_peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&g_peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
  // Register for a callback function that will be called when data is received
  esp_now_register_recv_cb(OnDataRecv);

  g_motorControl.MotorSetup();

  g_rightBlinker.Init(blinkerRightPIN);
  g_leftBlinker.Init(blinkerLeftPIN);
  g_backwardsBuzzer.Init(buzzerPIN);
}
```

Na početku loop funkcije obavljamo sve potrebne ciklične funkcije naših aparata, nakon čega učitamo komandu odašiljača i krećemo kotače u skladu s njom.

```
void loop() {
  g_ultrasonicControlForward.Think();
  g_ultrasonicControlBack.Think();
  g_rightBlinker.Think();
  g_leftBlinker.Think();
  g_backwardsBuzzer.Think();
```

U slučaju kretanja naprijed ili nazad, provjerimo udaljenost te u slučaju da je manja od 10 centimetara, ne dozvolimo kretanje.

Ako se okrećemo lijevo ili desno, uključujemo prikladni žmigavac, te ako se krećemo prema iza, upalimo Buzzer za upozorenje.

```
switch( g_iCommand ) {
  case STOP:
    g_motorControl.Stop();
    break;

  case FORWARDS:
    if( g_ultrasonicControlForward.GetCM() < 10 ) {
      g_motorControl.Stop();
      break;
    }
    g_motorControl.DriveForward();
    break;

  case BACKWARDS:
    if( g_ultrasonicControlBack.GetCM() < 10 ) {
      g_motorControl.Stop();
      break;
    }

    if( !g_backwardsBuzzer.IsPlaying() )
      g_backwardsBuzzer.StartPlayback(parking, sizeof(parking), 66);

    g_motorControl.DriveBackward();
    break;

  case RIGHT:
    if( !g_rightBlinker.IsBlinking() )
      g_rightBlinker.StartPlayback(led_blink_back, sizeof(led_blink_back), 1, true);

    g_motorControl.TurnRight();
    break;

  case LEFT:
    if( !g_leftBlinker.IsBlinking() )
      g_leftBlinker.StartPlayback(led_blink_back, sizeof(led_blink_back), 1, true);

    g_motorControl.TurnLeft();
    break;
}
```

Na kraju provjerimo da li se trebaju žmigavci ili Buzzer ugasiti, te ih onda prikladno ugasimo.

```
if( g_iCommand != LEFT && g_iCommand != RIGHT ) {
  g_leftBlinker.Stop();
  g_rightBlinker.Stop();
}

if( g_backwardsBuzzer.IsPlaying() && g_iCommand != BACKWARDS )
  g_backwardsBuzzer.Stop();
```

Primanje signala se odvija na drugoj dretvi, te koristi sljedeći callback:

```
void OnDataRecv(const uint8_t *mac, const uint8_t *incomingData, int len) {
  memcpy(&g_incomingCode, incomingData, sizeof(g_incomingCode)); // incomingData is the data that has been sent from the transmitter
  // incomingCode is the structure that matches the data structure of the transmitted message

  g_iCommand = g_incomingCode.m_iCommand; // alter te value of variables
}
```

4. Kratki troškovnik i financijska analiza

Cijena komponenti za izgradnju uključuje sve dijelove proizvoda, dijelove proizvoda koji su bili korišteni u razvoju no nisu uključeni u finalno rješenje i alate korištene za razvoj.

Izvori cijena: [8], [9]

Dijelovi finalnog proizvoda uključuju:

- 2x Dasduino CONNECTPLUS (2 x 14€)
- 2x eksperimentalna pločica (2 x 5,5€)
- 5x tipkalo 12mm (5 x 0,4€)
- 2x kablići (2x 5€)
- Li-ion baterija 500mAh 3.7V (5,5€)
- 2x DC upravljač motora L298N (2x 6€)
- 4x kotači, 4x DC motori, 2x plastične platforme, žice, šarafi i držači (9€)
- 2x ultrazvučni modul HC-SR04 (4€)
- 3x LED lampica (3x 0,2€)
- 2x zujalica (2x 0,75€)
- 2x industrijska baterija 9V (2x 5€)
- 5x industrijska baterija AA (5x 1,5€)
- Trošak prijevoza 3,5€ po narudžbi

To ukupno daje 102,6€.

Dijelovi koji su bili korišteni u razvoju proizvoda no nisu u finalnoj arhitekturi:

- 433MHz 4-kanalni prijemnik + daljinski upravljač (5,5€)
- 2x industrijska baterija 9V (2x 5€)
- 5x industrijska baterija AA (5x 1,5€)

To ukupno daje 23€.

Alati koji su bili korišteni u razvoju proizvoda:

- Digitalni multimetar (13,5€)
- Iznajmljivanje 3D printera (5€)

To ukupno daje 18,5€.

Ukupna cijena komponenti za izgradnju jedne jedinice je 102,6€ + 23€ + 18,5€ = 142€

Kako bi izgradili više od jedne jedinice trebamo umnažati vrijednost komponenti za sastavljanje finalnog proizvoda. Cijene bi bile:

1 proizvod 142€

2 proizvoda 246,7€

3 proizvoda 349,3€

5 proizvoda 554.5€

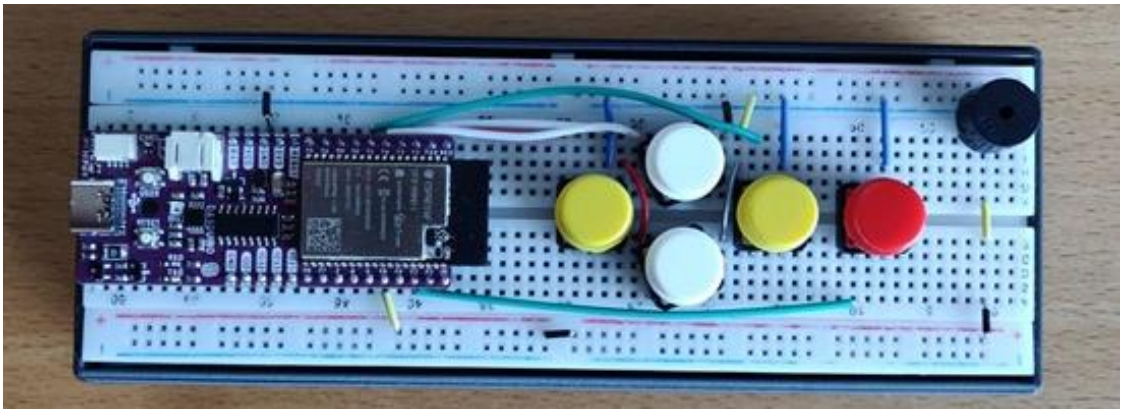
10 proizvoda 1067.5€

5. Korisnička dokumentacija

Cilj korisničke dokumentacije jest na jednostavan i razumljiv način prikazati korisniku načine upravljanja i korištenja robota. Korisnička dokumentacija uključuje uvid u sve funkcionalnosti robota, kao i dodatna pojašnjenja kako robot obavlja određene zadatke. Za jednostavnije i ugodnije iskustvo rukovanja robotom potrebno je imati razumijevanje u osnovne principe rada samog robota. Korisnička dokumentacija, također ima još jednu ulogu, a to je da u slučaju manjeg kvara na robotu, korisnik može sam doći do zaključka koji dio robota ne radi. Korisnik može doći do tog zaključka tako što je u korisničkoj dokumentaciji detaljno opisano kako bi se robot trebao ponašati uslijed izvršavanja određenih naredbi. U slučaju da robot ne izvršava radnje koje su opisane u dokumentaciji, znači da je robot u kvaru i ne radi ispravno. Korisnička dokumentacija je nadalje podijeljena na dva dijela, kretanje robota te ostale funkcionalnosti.

5.1. Kretanje robota

Upravljanje robotom se ostvaruje pomoću odašiljača, odnosno daljinskog kontrolera. Na kontroleru je moguće uočiti Dasduino CONNECTPLUS pločicu, pet tipkala (gumba) u raznim bojama, zujalicu te mnoštvo konekcija (žica). Bitna napomena je da se kontroler uvijek drži u ruci u položaju kako je prikazan na slici [Slika 33], i to da se Dasduino pločica nalazi s lijeve strane kontrolera, a crveno tipkalo (gumb) i zujalica se nalaze na desnoj strani.



Slika 33: odašiljač / daljinski kontroler

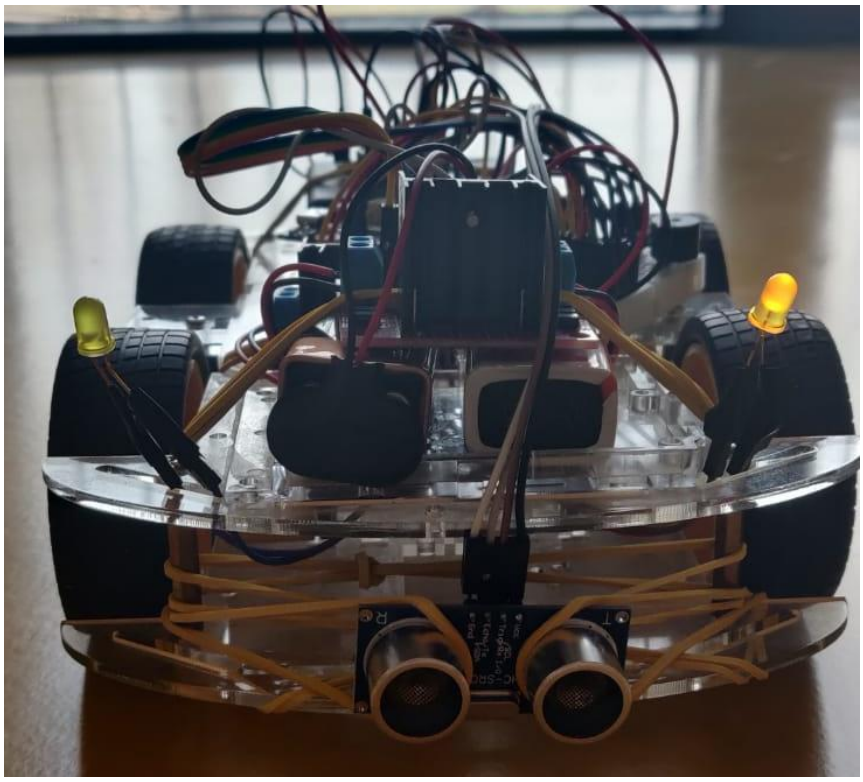
Kretanje robota se postiže pritiskom na bijela ili žuta tipkala. Ovisno o položaju na kojem se tipkala nalaze, u tom smjeru će se i robot kretati. Bijela tipkala služe za pomicanje robota u smjeru naprijed – nazad, dok žuta tipkala služe za okretanje robota u smjeru lijevo – desno. Kretanje je izrađeno vrlo intuitivno pa tako, gornji bijeli gumb predstavlja kretanje unaprijed, a donji bijeli gumb predstavlja kretanje unazad. Lijevo žuto tipkalo služi za skretanje robota u lijevo, a desno žuto tipkalo daje naredbu robotu za skretanje u desnu stranu. Kretanje robota se odvija sve dok korisnik drži pritisnut gumb. Čim korisnik ispusti gumb, robot staje s kretanjem.

Nije moguće držati pritisnuta dva gumba (na primjer davanje naredbi za naprijed i lijevo), već će se u tom slučaju izvoditi naredba samo onog gumba koji ima veći prioritet. Prioriteti izvođenja operacija su definirani unutar programskog rješenja te ih nije moguće mijenjati niti utjecati na njih. Dakle, ukratko rečeno, kretanjem robota se upravlja pritiskom na samo jedno tipkalo odjednom. Za promjenu smjera kretanja, potrebno je ispustiti trenutno tipkalo i stisnuti drugo tipkalo.

5.2. Ostale funkcionalnosti

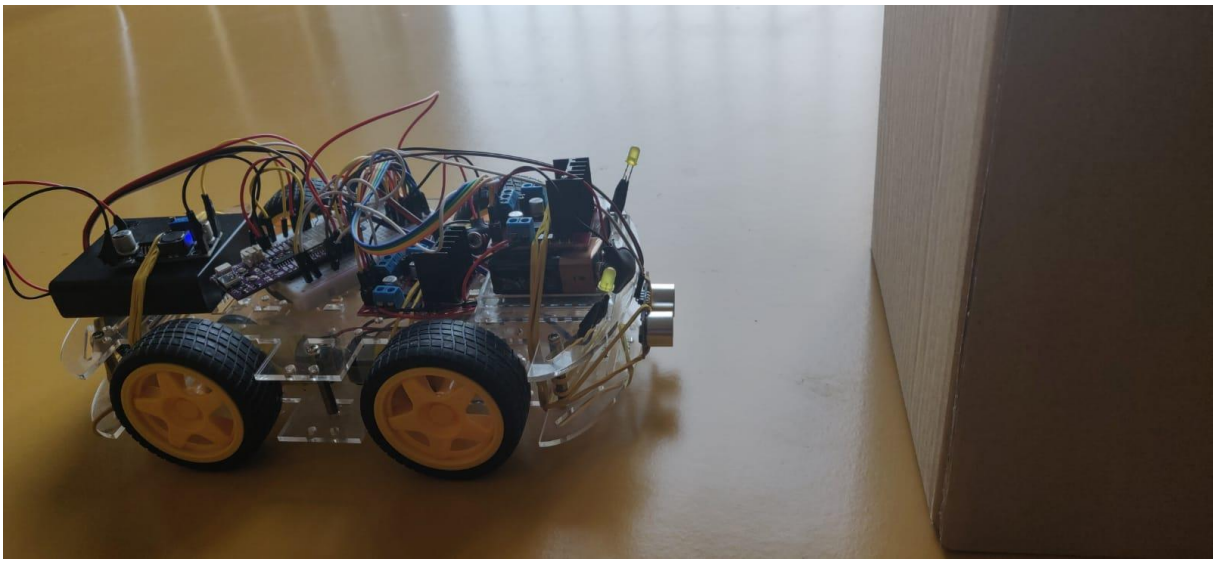
Ostale funkcionalnosti robota, ili bolje rečeno dodatne funkcionalnosti robota, utječu na korisničko iskustvo prilikom upravljanja robotom. U dodatne funkcionalnosti spada sustav žmigavaca, automatskog kočenja, vožnja unazad uz upozorenje, te opcija radija.

Sustav žmigavaca se aktivira prilikom skretanja ulijevo ili udesno. Kada korisnik drži pritisnut gumb za skretanje, robot naizmjenice pali i gasi LED diodu (lijevu ili desnu), ovisno u koju stranu robot skreće.



Slika 34: sustav žmigavaca

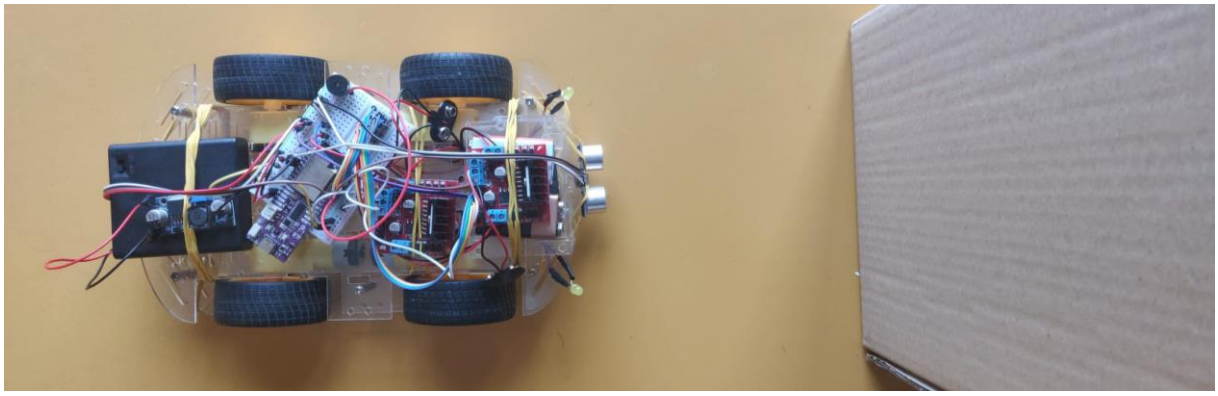
Sustav automatskog kočenja je postignut ugradnjom ultrazvučnih senzora. Ultrazvučni senzori se nalaze na prednjem i zadnjem dijelu robota te neprestano mjere udaljenost. U slučaju ako se robot približi nekoj prepreci na udaljenosti manjoj od 10 cm, robot samostalno koči, kako bi spriječio sudar. Ova funkcionalnost je implementirana kako bi se postigla određena sigurnost robota, ali i sigurnost okoline u kojoj se robot kreće. Na slijedećoj slici se može vidjeti slučaj kočenja robota koji se našao pred preprekom. [Slika 35]



Slika 35: sustav automatskog kočenja

Automatsko kočenje nije moguće aktivirati niti deaktivirati, već je prisutno tijekom cijele vožnje robota. Također, još je potrebno istaknuti što se događa s robotom jednom kada zakoči i stane ispred prepreke. Recimo da je robot vozio naprijed i naišao na prepreku s prednje strane. U tom slučaju korisnik pritiskom na gumb naprijed neće postići ništa. Robot se jednostavno neće htjeti kretati unaprijed. Stoga, potrebno je pritisnuti gumb za unazad, odmaknuti se od prepreke, te kada ultrazvučni senzor više ne očitava prepreku ispred sebe, robot će se ponovno moći kretati unaprijed pritiskom na gumb naprijed. Na isti način automatsko kočenje funkcionira kod vožnje unazad.

Vožnja unazad uz upozorenje uključuje zujalicu. Ideja ovoga je da robot pruži zvučno upozorenje kada se kreće unazad. Prilikom kretanja unazad zujalica se oglašava u intervalima, proizvodeći time zvučno upozorenje.



Slika 36: vožnja unazad uz upozorenje

Opcija radija je jedina funkcionalnost koja se ne izvršava na robotu, već na kontroleru. Naime, ukoliko je atmosfera tijekom vožnje robota dosta monotona i dosadna, korisnik pritiskom na crveno tipkalo na kontroleru, ima mogućnost uključivanja radija. Radio, ili bolje rečeno zujalica, ima spremljene razne prepoznatljive melodije, koje korisnik može mijenjati ponovnim pritiskom na isto crveno tipkalo. Ovakva funkcionalnost omogućuje ne samo pozitivno korisničko iskustvo, već pruža korisniku osjećaj slobode i izbora.

Slika 37: prikaz zujalice / radio na kontroleru

6. Zaključak

U konačnici, izrađeni robot zajedno s daljinskim kontrolerom predstavlja savršeni spoj mehanike, elektronike i informatike. Cjelokupni sustav se temelji na Dasduino CONNECTPLUS pločicama koje omogućuju komunikaciju između kontrolera i robota. Korištene su različite elektroničke komponente poput upravljača motora, ultrazvučnih senzora, zujalica i LED dioda. Ovi dijelovi su integrirani u kompleksnu arhitekturu sustava koja omogućuje upravljanje kretanjem robota, svjetlosnim efektima i reprodukcijom zvuka. Pomoću precizno definiranih komunikacijskih dijagrama, jasno je prikazana interakcija između komponenti sustava u različitim situacijama, od korisničkih komandi do autonomnih aktivnosti robota. Kroz dijagrame aktivnosti, jasno se prikazuje kako su aktivnosti koordinirane između korisnika, upravljača i robota. Shema spajanja pruža detaljan uvid u sve komponente robota spojene u jednu cjelinu i olakšava promatranje i razumijevanje robota kao složenog sustava. Izvršni programi robota i kontrolera sadrže naredbe koje se neprestano odvijaju u Dasduino pločicama, te time pružaju nesmetan rad robota. Prikaz troškovnika i financijske analize daje uvid u cijene komponenata na tržištu, te samim time pruža i uvid u okvirnu vrijednost izrađenog robota. Uloga korisničke dokumentacije je da korisniku olakša i pojednostavi proces upravljanja robotom, te također objašnjava slučajeve korištenja robota i njihove ishode. Može se reći da je korisnička dokumentacija prikaz pravilnog ponašanja robota. Naposljetku, projekt je zahtijevao ne samo tehničke vještine u području elektronike i programiranja već i timsku suradnju i dobru organiziranost. Uzimajući u obzir sve navedeno, možemo zaključiti da je ovaj projekt uspješno ostvaren, pridonoseći stjecanju i razvoju novih znanja i vještina iz područja ugrađenih sustava i programiranja.

Popis literature

[1] robot - Hrvatska enciklopedija. (bez dat.). Pristupano 30. prosinca, 2023, sa https://enciklopedija.hr/natuknica.aspx?ID=53100

[2] Dasduino CONNECTPLUS. (bez dat.). Pristupano 28. prosinca, 2024, sa https://soldered.com/hr/proizvod/dasduino-connectplus/

[3] In-Depth: Interface L298N DC Motor Driver Module with Arduino. (bez dat.). Pristupano 28. prosinca, 2024, sa https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/?utm_content=cmp-true

[4] HC-SR04 Ultrasonic Sensor Working, Pinout, Features & Datasheet. (bez dat.). Pristupano 28. prosinca, 2024, sa https://components101.com/sensors/ultrasonic-sensor-working-pinout-datasheet

[5] „How to Draw UML Communication Diagram?“ Pristupljeno: 13. siječanj 2024. [Na internetu]. Dostupno na: https://www.visual-paradigm.com/tutorials/how-to-draw-communication-diagram.jsp

[6] „ESP32 Dual Core with Arduino IDE | Random Nerd Tutorials“. Pristupljeno: 13. siječanj 2024. [Na internetu]. Dostupno na: https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/

[7] „What is Activity Diagram?“ Pristupljeno: 13. siječanj 2024. [Na internetu]. Dostupno na: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

[8] „Homepage“, Soldered Electronics. Pristupljeno: 13. siječanj 2024. [Na internetu]. Dostupno na: https://soldered.com/

[9] „Amazon.com. Spend less. Smile more.“ Pristupljeno: 13. siječanj 2024. [Na internetu]. Dostupno na: https://www.amazon.com/

[10] GitHub – robsoncouto/arduino-songs Pristupljeno: 14. siječanj 2024. [Na internetu]. Dostupno na: https://github.com/robsoncouto/arduino-songs

Popis slika

Slika 1: osnovna konstrukcija robota - Dijelovi 2
Slika 2: osnovna konstrukcija robota - Sastavljeno 2
Slika 3: Dasduino CONNECT PLUS 3
Slika 4: Dasduino CONNECTPLUS (ESP32) - Prikaz konekcija 4
Slika 5: L298N DC motor driver 4
Slika 6: promjenom širine impulsa dobiva se različiti napon na izlazu 5
Slika 7: mijenjanje polariteta napona na motorima H-Bridge principom 5
Slika 8: L298N DC motor driver – Prikaz pinova 6
Slika 9: HC-SR04 Ultrasonic Sensor - Prikaz pinova 7
Slika 10: princip rada ultrazvučnog senzora 7
Slika 11: zujalice - Aktivna zujalica (lijevo) i Pasivna zujalica (desno) 8
Slika 12: svjetleća dioda - Prepoznavanje anode i katode 8
Slika 13: donji dio maske 9
Slika 14: ptičja perspektiva na donji dio maske 9
Slika 15: mravlja perspektiva na gornji dio maske 10
Slika 16: ptičja perspektiva na gornji dio maske 10
Slika 17: Prikaz složenog robota 11
Slika 18: Prikaz daljinskog kontrolera (odašiljača) 11
Slika 19: dijagram komunikacije kada korisnik ne pritišće niti jedan gumb 12
Slika 20: dijagram komunikacije kada korisnik pritišće gumb gore 13
Slika 21: dijagram komunikacije kada korisnik pritišće gumb dolje 14
Slika 22: dijagram komunikacije kada korisnik pritišće gumb lijevo 15
Slika 23: dijagram komunikacije kada korisnik pritišće gumb desno 15
Slika 24: dijagram kada korisnik pritišće gumb muzika 16
Slika 25: staza korisnika dijagrama aktivnosti 17
Slika 26: staza upravljača dijagrama aktivnost – 1 od 2 18
Slika 27: staza upravljača dijagrama aktivnosti – 2 od 2 18
Slika 28: staza robota dijagrama aktivnosti - 1 od 3 20
Slika 29: staza robota dijagrama aktivnosti - 2 od 3 20
Slika 30: staza robota dijagrama aktivnosti – 3 od 3 21
Slika 31: shema spajanja odašiljača 22
Slika 32: shema spajanja robota 23
Slika 33: odašiljač / daljinski kontroler 36
Slika 34: sustav žmigavaca 38

Slika 35: sustav automatskog kočenja.....39
Slika 36: vožnja unazad uz upozorenje 39
Slika 37: prikaz zujalice / radio na kontroleru.....40