UNIVERZITET U TUZLI

Fakultet elektrotehnike
Odsjek: Automatika i robotika (AR)
Godina studija: IV
Predmet: **Projektovanje sistema na čipu**

Primjena mobilnog robota za automatsko zalijevanje bašte (Smart garden)

Projektni zadatak

Profesor:
Dr. sc. Lejla Banjanović – Mehmedović, vanr. prof.
Asistent:
Mr. Sc. Ivan Bosankić, viši asistent

Studenti:
Amila Jahić
Asja Hadžić
Damir Mijatović

Tuzla, april 2023.

Sadržaj

Popis slika

1. Uvod

Različiti tipovi biljaka i zelenilo su neizbježan dio naše svakodnevnice. Bilo da se radi čisto o ukrasnim biljkama ili agrikulturi biljaka, s porastom otpuštanja stakleničkih plinova (ugljikov dioksid, vodena para, metan, ozon itd.) značajno je porastao i utjecaj istih na globalno zatopljenje i klimatske promjene.

Nažalost, jedan od glavnih razloga izumiranja biljaka, pored sječe šuma, jeste upravo nedostatak vode i adekvatnog zalijevanja. U svrhu olakšanja problema zalijevanja razvijeni su mnogi sistemi za automatsko navodnjavanje bašte i zemljišta, međutim mnogi od njih nisu portabilni i često su izuzetno skupi.

S tim u vezi, ideja ovog projekta jeste upravo da omogući ekonomski prihvatljiv i veoma agilan sistem: Mobilni robot za automatsko zalijevanje bašte. Ovaj sistem će služiti kao lični vrtlar i biće blagodat za ljude koji vole da uzgajaju biljke, ali ne mogu se redovno brinuti o njima radi obaveza. Sistem je u potpunosti prilagodljiv svakom okruženju i uzima u obzir potrebe za zalijevanjem biljaka pomoću senzora vlažnosti. U radu je opisana hardverska arhitektura potpuno automatizovanog sistem za navodnjavanje koji koristi bežičnu komunikaciju za interakciju mobilnog robota i senzorskog modula. Ovaj vrtlarski robot je potpuno prenosiv i opremljen je modulom za radiofrekventnu identifikaciju (RFID), mikrokontrolerom, ugrađenim rezervoarom za vodu i priključenom pumpom za vodu. Sposoban je osjetiti potrebe biljaka za navodnjavanjem, locirati ih i konačno ih zalijevati samostalno bez ikakve ljudske intervencije.

1.1. Ideja projekta

Kao što je već navedeno, ideja ovog projekta jeste upotreba mobilnog robota za automatsko vodosnabdijevanje bašte. Ovakav sistem se sastoji od dva dijela, gdje je prvi dio dio upravo mobilni robot, dok je drugi dio bašta, odnosno zemlja sa odgovarajućim senzorima i komponentama koje signaliziraju pokretanje datog robota. Okvirna funkcionalnost sistema je sljedeća:

1. Za realizaciju projekta korištene su FPGA i Raspberry Pi pločica

2. Područje koje će zalijevati autonomni mobilni robot može biti bilo koje polje sa biljkama postavljenim u liniji duž unaprijed definirane putanje koju će robot slijediti. Staza koju robot treba da prati mora biti crna na bijeloj pozadini zbog IR senzora koji apsorbuju svjetlost i razlikuju boje na osnovu količine svjetlosti koju apsorbuju.

3. U polju sa biljkama se nalaze senzori vlažnosti tla (YL69), koji signaliziraju nedostatak vode u zemlji. U zavisnosti o kojoj vrsti biljke se radi, na senzoru je postavljena granica potrebne vlažnosti tla. Izlaz senzora je analogni signal, te svaka vrsta zahtijeva senzor sa unaprijed podešenom granicom vlažnosti (npr. u slučaju da vlažnost zemlje padne ispod 40% potrebno je zalijevanje tla). Vrijednost praga za analogni signal može se podesiti pomoću potenciometra, tj. može se podesiti na kojem levelu će se očitati da je zemlja dovoljno vlažna (u zavisnosti od potrebe biljke).

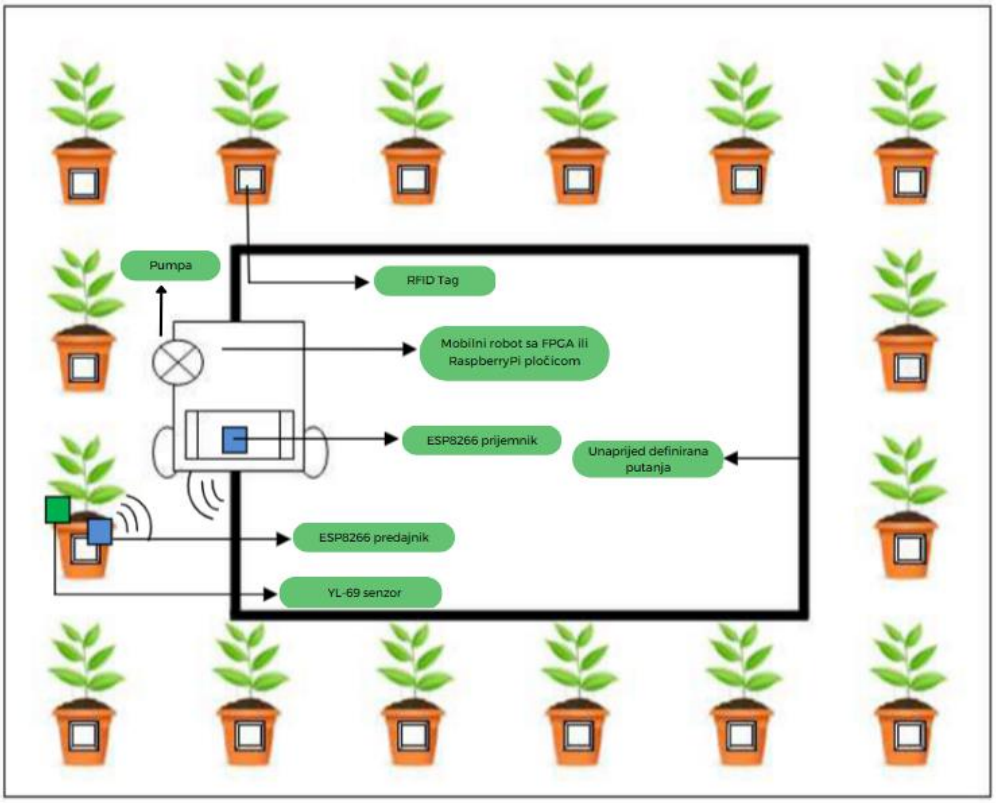
4. Ukoliko je potrebno zaliti tlo, iz bašte se, pomoću Wi-Fi mehanizma šalje signal mobilnom robotu koji će značiti da je vrijeme za zalijevanje. Koristit će se ESP2866 nodeMCU mikrokontroler za slanje takvog signala robotu i robot započinje kretanje.

5. Ispred svake biljke se nalazi RFID oznaka. Domet RFID EM-18 čitača modula je oko 4 inča što znači da postrojenje mora biti unutar 4 inča od autonomnog robota i RFID oznaka mora biti okrenuta prema modulu RFID čitača. Na robotu se nalazi RFID čitač i on očitavajući RFID oznake omogućava skretanje robota i finalnu lokaciju. U našem slučaju bi to bilo usmjeravanje robota na odgovarajuću lokacju u bašti koju treba zaliti.

6. Kada robot dobije informaciju da je potrebno zaliti tlo, on kreće iz svoje pozicije prema mjestu gdje je neophodno zalijevanje. Robot slijedi unaprijed određenu putanju koja će ga dovesti u blizinu biljke koju treba zaliti. Za pokretanje robota korišteni su DC motori. Robot osjeća vlažnost biljke, locira je, zalijeva i nastavlja do sljedeće biljke. Biljke su postavljene duž putanje po kojoj će ići autonomni robot.

7. Za navodnjavanje se koristi kontejner napunjen vodom u kojem se nalazi pumpa. Sistem može nositi samo određenu količinu vode u jednom potezu jer je kapacitet nošenja vode ograničen zbog težine koju DC motori mogu pokretati. Ovo također ograničava broj saksija i veličinu saksija koje treba zalijevati. Budući da se pumpa za vodu napaja jednosmjernom strujom, konfiguracija vodene pumpe se uzima na 12V ili manje. Pumpa za vodu se pokreće pomoću 6V releja koji djeluje kao prekidač, ali umjesto da ga fizički dodirnemo kako bismo ga uključili/isključili, mi dovodimo napon kako bismo ga uključili. Visina saksija se uzima prema visini mobilnog robota tako da voda može lako doprijeti u zemlju pomoću vodovodne cijevi. U slučaju da se voda koju nosi robot iscrpi, onda je potrebno uključivanje ljudske snage u ovaj proces.

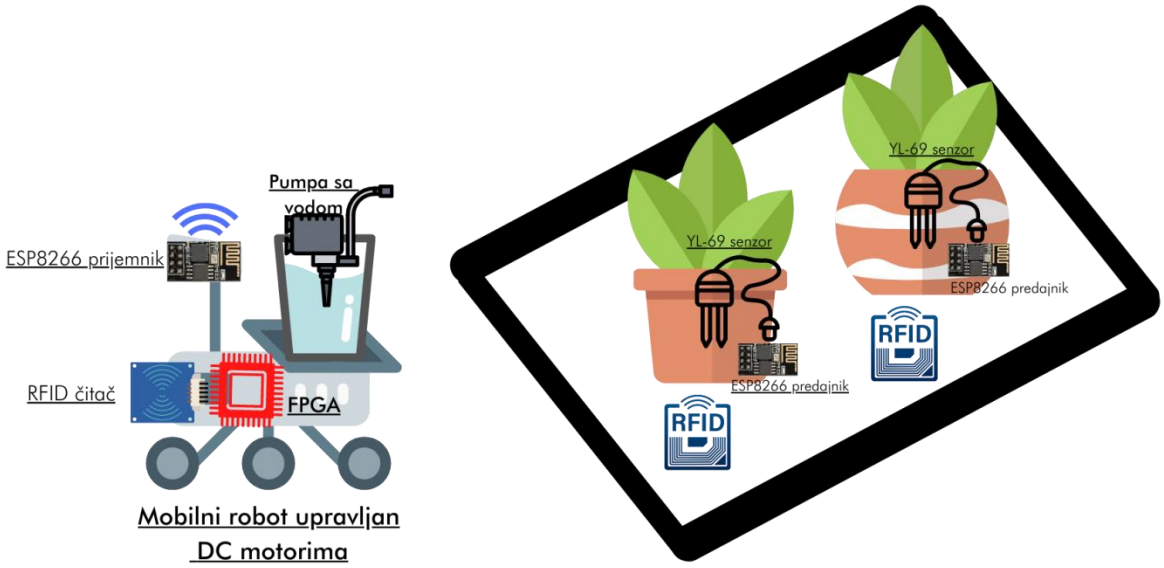
Skica prikaza funkcionalnosti sistema je prikazana na slici ispod:



Slika 1: Ideja projekta

1.2. Cilj projekta

Cilj nam je predstaviti mogućnosti upotrebe FPGA i RaspberryPi pločice realizacijom potpuno autonomnog sistema koji pomaže u zalijevanju saksijskih biljaka koje su raspoređene duž unaprijed definisane putanje. Mobilni robot treba biti sposoban da obavlja tri glavne funkcije: otkrivanje potrebe za zalijevanjem biljaka kojima je potrebna voda, lociranje mjesta koje treba zaliti i konačno zalijevanje koje je autonomno bez ikakve ljudske intervencije. Sistem se sastoji od autonomnog vozila, FPGA ili RapspberryPi pločice, modula RFID čitača, pumpe za vodu, releja, senzora vlažnosti (YL-69) i uređaja za bežičnu komunikaciju. Performanse sistema su procijenjene između broja biljaka koje se zalijevaju i vremena potrebnog mobilnom robotu da izvrši operaciju navodnjavanja, što pokazuje da sistem nije samo isplativ, već i efikasan u smislu vremena.



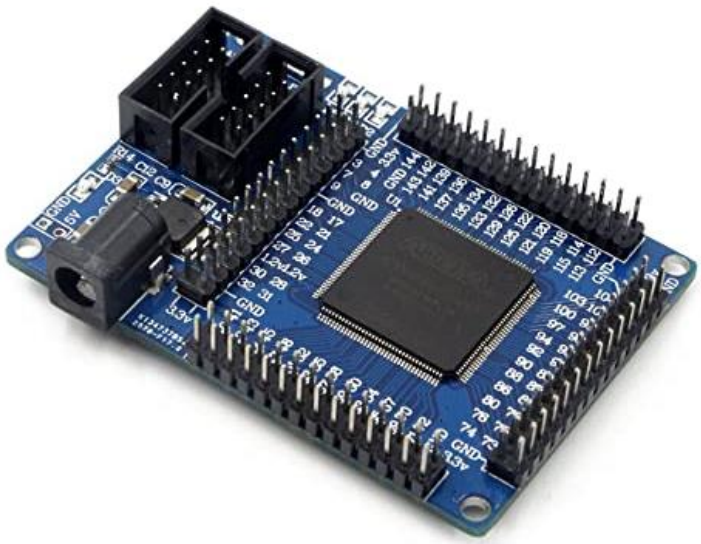
Slika 2: Skica projekta

2. Popis korištene opreme i njihove specifikacije

2.1. Cyclone II FPGA

Alterina porodica Cyclone II FPGA temelji se na 1,2V, 90-nm SRAM procesu s gustoćom preko 68K logičkih elemenata (LE) i do 1,1 Mbita ugrađenog RAM-a. Sa karakteristikama poput ugrađenog 18 × 18 množitelja za podršku DSP aplikacija visokih performansi, fazno zaključane petlje (PLL) za upravljanje taktom sistema i podrška za interfejs eksterne memorije velike brzine za SRAM i DRAM uređaje. Cyclone II uređaji su isplativo rješenje za aplikacije velikog volumena. Cyclone II uređaji podržavaju diferencijalne i single-ended I/O standarde, uključujući LVDS pri brzinama podataka do 805 megabita u sekundi (Mbps) za ulaze i 622 Mbps za izlaze, te 64-bitni, 66-MHz PCI i PCI- X za povezivanje s procesorima i ASSP i ASIC uređajima.

Cyclone™ II uređaji sadrže dvodimenzionalnu arhitekturu zasnovanu na redovima i kolonama za implementaciju prilagođene logike. Interkonekcije kolona i redova različitih brzina pružaju signalne interkonekcije između blokova logičkog niza (LAB), ugrađenih memorijskih blokova i ugrađenih množača. Logički niz se sastoji od LAB-ova, sa 16 logičkih elemenata (LE) u svakom LAB-u. LE je mala logička jedinica koja pruža efikasnu implementaciju korisničkih logičkih funkcija. [1]



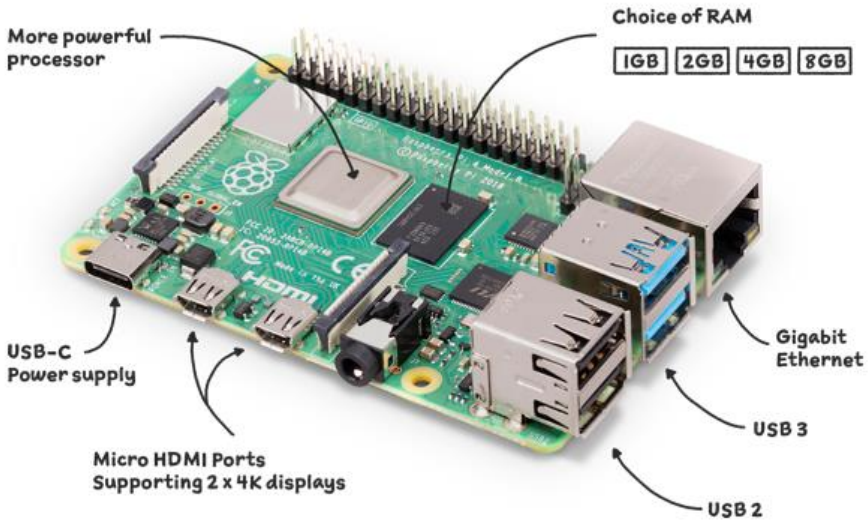
Slika 3: EP2C5T144 Cyclone II FPGA

Najmanja logička jedinica u arhitekturi Cyclone II, LE je kompaktna i obezbjeđuje napredne karakteristike. Svaki LE se sastoji od:

- 4 – ulazne look-up tabele (LUT), koja predstavlja generator funkcija koji implementira bilo koju funkciju sa 4 varijable
- Programabilni registar
- Carry chain connection
- Register chain connection
- Mogućnost pokretanja svih interkonekcija
- Registarski niz podataka, direktne link interkonekcije
- Podrška za registerpacking i registerfeedback

[1] pdf_CycloneII_datasheet

2.2. Raspberry Pi



Slika 4: Raspberry Pi

Raspberry Pi je serija malih single-board računara (SBC) koju je razvila u Ujedinjenom Kraljevstvu Raspberry Pi Fondacija u saradnji sa Broadcomom. Raspberry Pi SBC imaju Broadcom sistem na čipu (SoC) sa integrisanom ARM kompatibilnom centralnom procesorskom jedinicom (CPU) i grafičkom procesorskom jedinicom na čipu (GPU). CPU dostiže brzinu od 700 MHz do 1.2 GHz za sa memorijom ploče u opsegu od 256 MB do 1 GB RAM. SD kartice se koriste za čuvanje operativnog sistema i programske memorije u ili SDHC ili MicroSDHC veličinama. Low-level izlaz snabdjeven je brojem GPIO pinova koji podržavaju zajedničke protokole kao sto je I2C.

Broadcom BCM2835 SoC koji se koristi u prvoj generaciji Raspberry Pi uključuje 700 MHz 32-bitni ARM1176JZF-S procesor, VideoCore IV grafičku procesorsku jedinicu (GPU) i RAM. Ima nivo 1 (L1) keš memorije od 16 KB i nivo 2 (L2) keš memorije od 128 KB. SoC je naslagan ispod RAM čipa, tako da je vidljiva samo njegova ivica. GPU obezbeđuje 1 Gpiksela/s ili 1,5 Gteksel/s grafičke obrade ili 24 GFLOPS računarskih performansi opšte namjene.[2]

[2] https://en.wikipedia.org/wiki/Raspberry_Pi

2.3. ESP8266 NodeMCU

Slika 5: ESP8266 NodeMCU

ESP8266 je jeftin Wi-Fi mikročip, sa ugrađenim TCP/IP mrežnim softverom i mogućnošću mikrokontrolera, koji proizvodi Espressif Systems u Šangaju, Kina. Čip je populariziran u zajednici proizvođača engleskog govornog područja u avgustu 2014. putem modula ESP-01, koji je napravio treći proizvođač Ai-Thinker. Ovaj mali modul omogućava mikrokontrolerima da se povežu na Wi-Fi mrežu i naprave jednostavne TCP/IP veze koristeći komande u Hayes stilu.

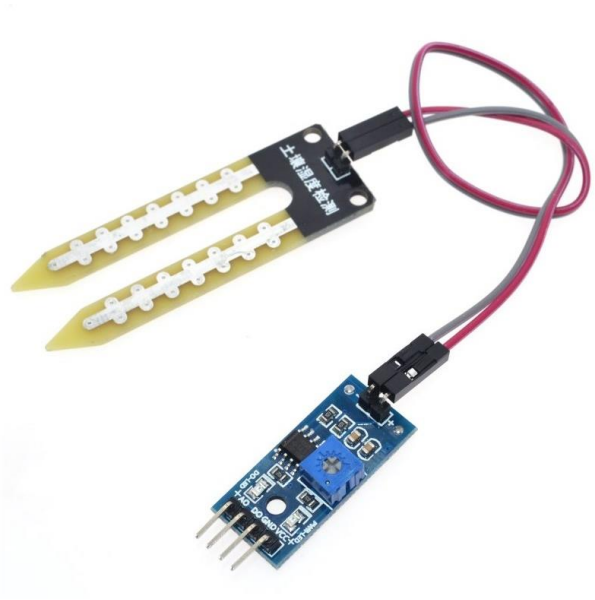
Karakteristike:

- Procesor: L106 32-bitno RISC jezgro mikroprocesora zasnovano na Tensilica Diamond Standard 106Micro radi na 80 ili 160 MHz
- Memorija:
  - 32 KiB instrukcija RAM
  - 32 KiB instrukcija cache RAM
  - 80 KiB user-data RAM
  - 16 KiB ETS sistemski podaci RAM
- Eksterni QSPI flash: podržano je do 16 MiB (obično uključeno 512 KiB do 4 MiB)
- IEEE 802.11 b/g/n Wi-Fi
  - Integrisani TR prekidač, balun, LNA, pojačalo snage i odgovarajuća mreža
  - WEP ili WPA/WPA2 autentikacija ili otvorene mreže
- 17 GPIO pinova
- Serial Peripheral Interface Bus (SPI)
- I²C (implementacija softvera)
- I²S interfejs sa DMA (dijeljenje pinova sa GPIO)
- UART na namjenskim pinovima, plus transmit-only UART može se omogućiti na GPIO2
- 10-bitni ADC (successive approximation ADC) [3]

[3] https://en.wikipedia.org/wiki/ESP8266

2.4. Ulazi u sistem

2.4.1. YL-69 senzor vlage tla



Slika 6: YL-69 Soil moisture senzor

Senzor vlažnosti tla ili higrometar se obično koristi za detekciju vlažnosti tla. Koristan je ukoliko se želi izgraditi automatski sistem za navodnjavanje ili jednostavno samo pratiti vlažnost tla biljaka.

Senzor se sastoji od dva dijela: elektronske ploče i sonde sa dva jastučića, koja detektuje sadržaj vode. Ima ugrađen potenciometar za podešavanje osjetljivosti digitalnog izlaza, LED za napajanje i LED za digitalni izlaz. YL69 senzor vlažnosti tla daje nam dvije vrste izlaza: digitalni i analogni.

Izlaz može biti digitalni signal 1 ili 0, ovisno o sadržaju vode. Ako je vlažnost tla prisutna izlaz će biti 0, u suprotnom izlaze je 1. Vrijednost praga za digitalni signal može se podesiti pomoću potenciometra, tj. možemo podesiti na kojem levelu će se očitati da je zemlja dovoljno vlažna.

Izlaz može biti i analogni ali u našem slučaju nismo koristili zbog kompleksnosti, a i jednostavno nije nam bilo potrebno, dovoljno je da znamo da li je zemlja zalivena ili je suha i potrebno je zaliti.

Izrađen je od materijala otpornog na koroziju što mu daje izvrstan vijek trajanja.

Pinovi na ovom senzoru:

- VCC: 3.3V, 5V,
- GND,
- DO: digitalio izlaz (0 i 1)
- AO: analogni izlaz[4]

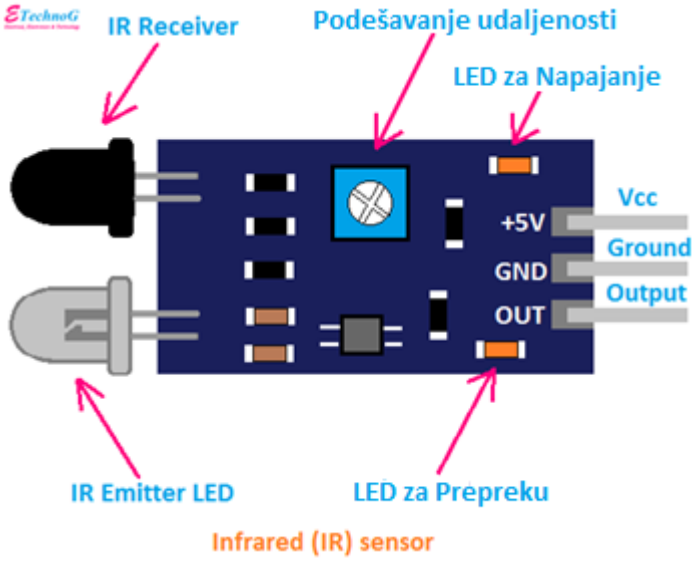
[4] https://randomnerdtutorials.com/guide-for-soil-moisture-sensor-yl-69-or-hl-69-with-the-arduino/

2.4.2. IR senzor

Slika 7: IR senzor

Infracrveni senzor je elektronski uređaj, koji emituje svjetlost da bi osjetio neke aspekte okoline. IR senzor može mjeriti toplinu objekta kao i detektirati kretanje. Ovi tipovi senzora mjere samo infracrveno zračenje, umjesto da ga emituju, to čine senzori koji se nazivaju pasivni IR senzor (PIR).

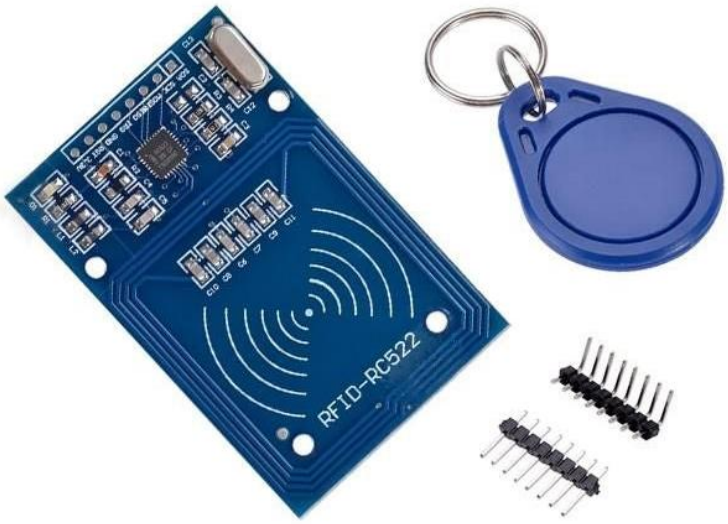
Performanse IR senzora ograničene su njihovom slabom tolerancijom na refleksije svjetlosti kao što su ambijentalno svjetlo ili svijetle boje objekata. IR senzori također daju netačne rezultate detekcije sa transparentnim ili svijetlim materijalima u boji. Rezultati detekcije također zavise od vremenskih uslova, a pouzdanost opada sa vlagom i vlažnošću. [5]



Slika 8: Komponente IR senzora

[5] https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/

2.4.3. RFID



Slika 9: RFID čitač i tag

RFID (Radio-frequency identification) je bežična i beskontaktna tehnologija koja koristi radio frekvenciju kako bi se razmjenjivale informacije između prijenosnih uređaja/memorija i host računara. RFID sistem je sastavljen iz RFID čitača i RFID medija koji izmjenjuju signale putem radio valova, i tako izmjenjuju informacije. Mediji za identifikaciju koji su dostupni mogu biti kartice, privjesci, narukvice, NFC (Near Field Communication) mobilni uređaj itd., pa čak i ključevi automobila.

Prednosti RFID tehnologije:

- zaštita podataka (RFID karticu je gotovo nemoguće duplicirati)
- nema utjecaja vanjskog elektromagnetskog djelovanja na samu karticu ili neki drugi medij
- brzina čitanja traje u milisekundama
- velika otpornost na ostale vanjske utjecaje (vlaga, nečistoća, visoke temperature, mehanička otpornost)
- dug vijek trajanja
- mali troškovi za energiju i održavanje
- mogućnost zapisa dodatnih podataka na karticu povezivanje i korištenje iste kartice s ostalim sistemima (plaćanje sadržaja, otvaranje sefova, kontrola pristupa, evidencija i planiranje radnog vremena…) [6]

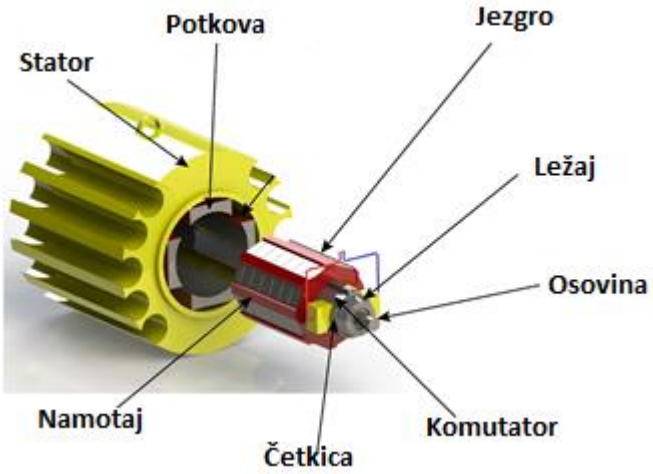
U našem projektu smo RFID čitač spojili sa ESP8266 Node MCU, prema sljedećem korespondiranju pinova:

RFID pinovi	**FPGA pinovi**
RST	D1 (GPIO5)
SPI SDA	D2 (GPIO4)
SPI MOSI	D7 (GPIO13)
SPI MISO	D6 (GPIO12)
SPI SCK	D5 (GPIO14)
3.3V	3.3V
GND	GND

[6] https://hr.wikipedia.org/wiki/RFID

2.5. Izlazi sistema

2.5.1. DC motori



Slika 10: Komponente DC motora

Električni DC motor je kontinualni aktuator koji pretvara električnu energiju u mehaničku. Ovo se postiže proizvodnjom kontinualne ugaone rotacije koja se može koristiti za pokretanje pumpi, ventilatora, kompresora, točkova, itd.

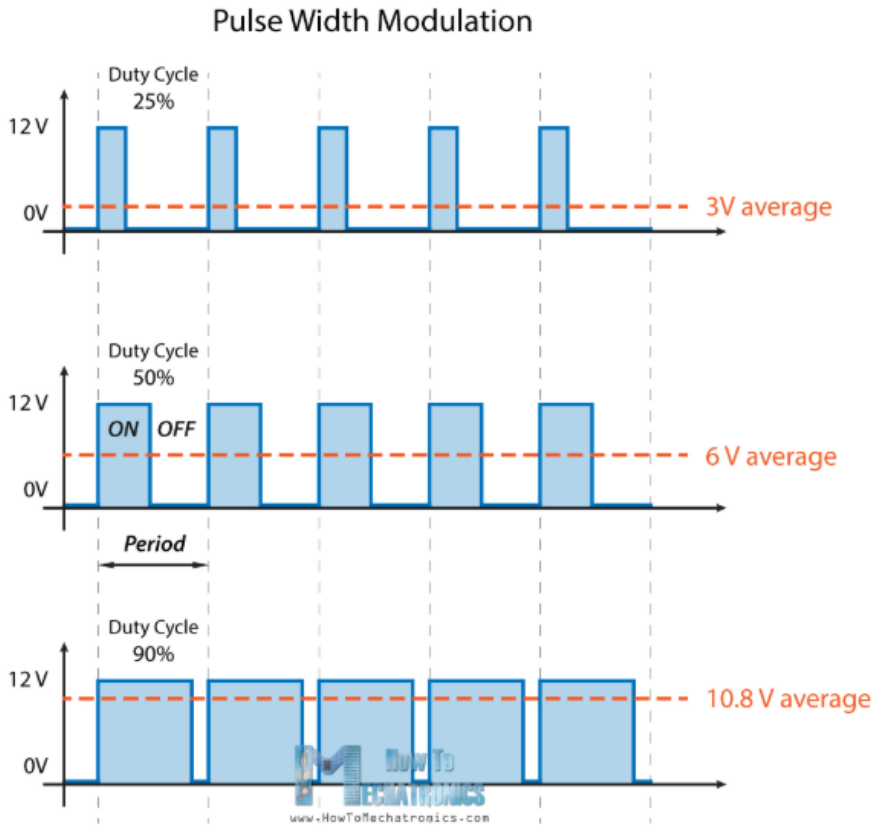
Razlikujemo :

- DC motore sa četkicama
- DC motore bez četkica[7]

U projektu koristimo četiri DC motora za pokretanje četiri točka mobilnog robota.

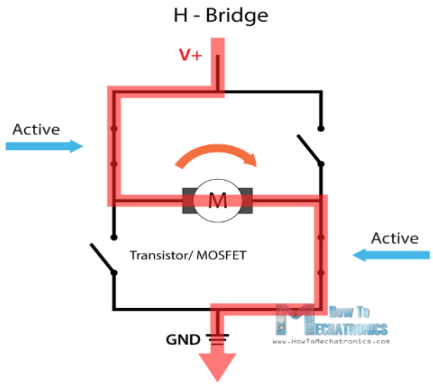
Brzinom vrtnje DC motora upravljamo pomoću PWM signala. PWM je tehnika koja nam dozvoljava da podesimo prosječnu vrijednost napona napajanja tako što uključujemo i isključujemo napajanje pri velikim brzinama. Prosječni napon zavisi od faktora ispunjenosti, to jeste odnosa vremena tokom kojeg je napajanje bilo uključeno (ON) i isključeno (OFF) tokom jednog vremenskog perioda signala napajanja. [8]

[7] https://hr.wikipedia.org/wiki/Istosmjerni_motor
[8] https://en.wikipedia.org/wiki/Pulse-width_modulation



Slika 11: PWM

Upravljanje smjerom vrtnje DC motora se vrši pomoću H-mosta. H- most je kolo koje sadrži četiri prekidača, tranzistora ili MOSFET-a, sa motorom u centru pri čemu na taj način formira konfiguraciju u obliku H slova. Aktiviranjem određena dva prekidača istovremeno možemo mijenjati smjer proticanja struje, a samim tim i smjer vrtnje motora. [9]

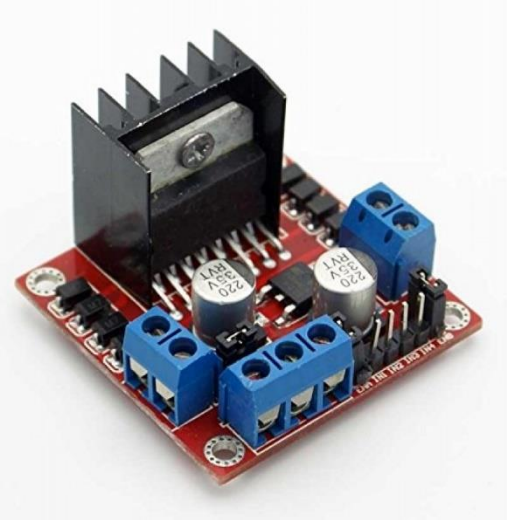


Slika 12: H – most

Kombinacijom ove dvije metode, PWM-a i H-mosta možemo potpuno upravljati DC motorom. Postoji mnogo drivera za DC motore koji imaju ove karakteristike i jedan od njih je upravo sljedeća komponenta koju koristimo, koja ne predstavlja izlaz ali služi za upravljanje jednim od izlaza.

[9] https://soldered.com/hr/learn/kkm-jednostavni-h-bridge/

2.5.2. L298N Motor driver



Slika 13: L298N Motor driver

L298N je driver motora sa dvostrukim H-mostom, koji dozvoljava kontrolu brzine i smjera vrtnje DC motora istovremeno. Napon napajanja ovog drivera je između 5 i 35V. L298 H Bridge je baziran na l298 čipu. To je visokonaponski i visokostrujni full dual bridge drajver dizajniran da prihvati standardni TTL logički nivo i pokreće induktivna opterećenja kao što su releji, solenoidi i DC koračni motor. Ovaj modul ima lakoću za povezivanje i pogon jednosmjernog motora ili koračnog motora omogućava vam da lako i pažljivo kontrolirate motorima.

Specifikacija

- Napajanje motora: 7 do 24 VDC
- Kontrolna logika: Standardni TTL logički nivo
- Izlazna snaga: do 2 A svaki
- Pinovi za omogućavanje i kontrolu smjera
- Heat-Sink
- LED indikator uključenja
- 4 LED indikatora smjera[10]

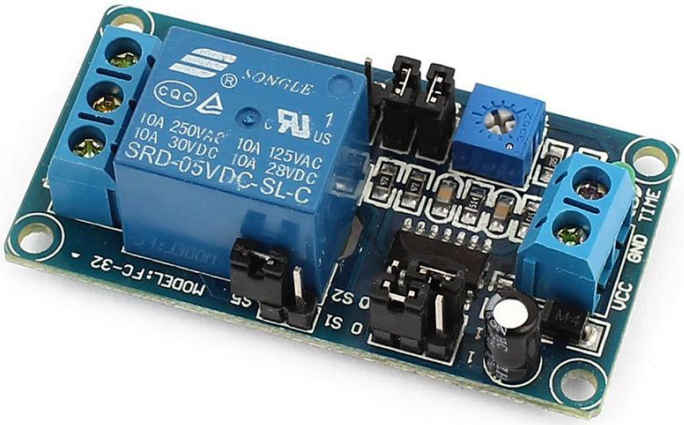
2.5.3. Vodena pumpa

Slika 14: Vodena pumpa

[10] https://www.14core.com/wiring-driving-the-l298n-h-bridge-on-2-to-4-dc-motors/

Budući da se pumpa za vodu napaja jednosmjernom strujom, konfiguracija vodene pumpe se uzima na 12V ili manje. Pumpa za vodu se pokreće pomoću 6V releja koji djeluje kao prekidač, ali umjesto da ga fizički dodirnemo kako bismo ga uključili/isključili, mi dovodimo napon kako bismo ga uključili. Voda se dozira u biljku pomoću cijevčice.

- Napon: CC 3-5 V
- Radna struja: 100-200mA
- Snaga opterećenja: 0,4-1,5 W
- Maksimalna visina: 40-110 cm
- Protok vode: 80-120L/H
- Kontinuirani radni vijek: 500 sati



Slika 15: Jednokanalni relej

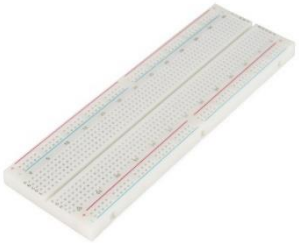
2.6. Dodatna oprema

2.6.1. Baterije

Slika 16: Baterije

Baterije koristimo za napajanje L298N drivera i FPGA. Koriste se dvije baterije od po 3.7V. Također za napajanje mikrokontrolera je korišten Powerbank.

2.6.2. Breadboard



Slika 17: Breadboard

2.6.3. Jumper žice

Slika 18: Jumper žice

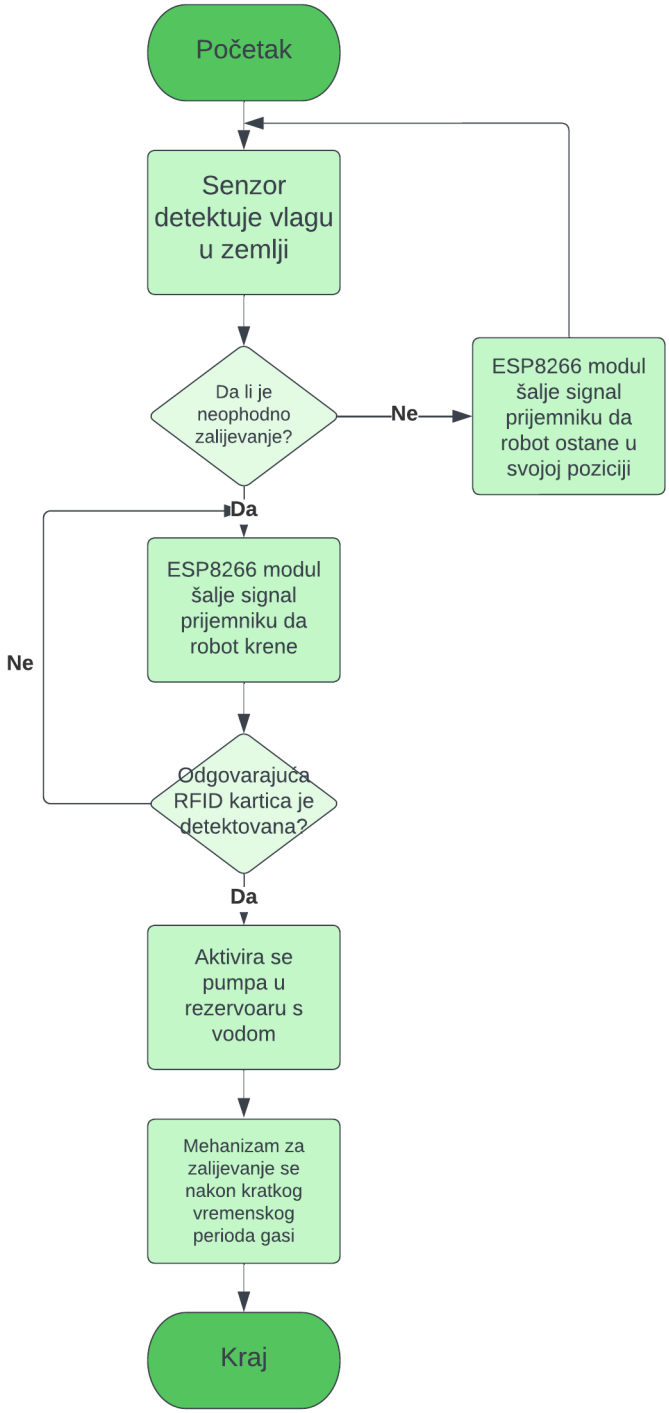
2.6.4. Arudino Smart Robot Car Kit

3. Algoritam rada

U nastavku je detaljno opisan algoritam rada našeg projektnog zadatka:

1.) Bašta, odnosno YL-69 senzori vlage određuju procenat vlage u zemlji.
2.) U ovisnosti od procenta vlage, mikrokontroler (predajnik) na kojeg je povezan senzor vlage određuje da li je vrijeme za zaljevanje, te u ovisnosti od toga šalje određeni podatak mobilnom robotu (ista logika važi za sve saksije, odnosno mikrokontrolere i senzore u bašti)
3.) Podatak koji se šalje je decimalni broj kojeg prima mikrokontroler na robotu (prijemnik) koji potom serijski šalje taj podatak FPGA ili Raspberry Pi pločici. Taj decimalni broj također određuje i adresu saksije koju treba ili ne treba zaliti.
4.) Podatci se šalju neovisno o tome je li potrebno pokrenuti zaljevanje. Razlika je u broju koji se šalje, gdje određeni brojevi signaliziraju zaljevanje, dok drugi signaliziraju mirovanje. Prema tome, ukoliko robot primi podatak koji signalizira zaljevanje, aktiviraju se DC motori.
5.) Robot počinje kretanje po unaprijed definisanoj putanji. IR senzori očitavaju podlogu po kojoj se robot kreće, te u ovisnosti tih ulaza, robot vrši pravolinijsko ili kretanje ulijevo/udesno.
6.) Robot će se kretati po definisanoj putanji sve dok ne dođe do određene RFID kartice. Kada RFID čitač na robotu očita karticu, provjerava se da li je data kartica zapravo saksija koju je neophodno zaliti ili ne (provjeravanje se također vrši serijskim prenosom bita). U slučaju da nije, robot nastavlja kretanje sve dok ne očita karticu sa pravom adresom. U slučaju da jest, robot se zaustavlja.
7.) Nakon zaustavljanja, pali se pumpa koja se nalazi u rezervoaru s vodom, te počinje proces zaljevanja.
8.) Zaljevanje traje određeno vrijeme, dok se pumpa ne ugasi. Nakon toga robot će stojati kratko vrijeme te zatim nastaviti kretanje.
9.) Ukoliko podatci koji pristižu odrede da više nije potrebno zaliti niti jednu saksiju, robot se zaustavlja.

Ispod je prikazan i dijagram toka našeg algoritma. Korisna stvar kod ovakve implementacije projekta je ta što broj saksija nije fiksan, tj. ovakav algoritam je prilagodljiv promjenama u smislu povećanja/smanjenja broja saksija ovisno o potrebama korisnika.



Slika 19: Dijagram toka

4. Rješenje zadatka

Nakon detaljnije analize alogritma i logike rada našeg projekta, lako je primijetiti da se isti sastoji od dva dijela:

1. Bašta (podrazumijeva senzor(e) vlage, WiFi predajnik(e), RFID karticu/e)
2. Mobilni robot (podrazumijeva mobilnog robota sa RFID čitačem i WiFi prijemnikom)

4.1. Bašta

Kako je već ranije navedeno, bašta (odnosno saksija) u ovom projektu je prilagodljiva promjenama, odnosno moguće je realizovati ovakav projekat sa N saksija, sve dok svaka od saksija ima svoj senzor vlage i WiFi predajnik. Naš projekat je realizovan sa dvije saksije, te ćemo takvo rješenje detaljno i objasniti.

Za početak, u našem slučaju razlikujemo četiri moguća slučaja:

- 00 – Nije potrebno zaljevanje niti jedne saksije
- 01 – Potrebno je zaliti prvu saksiju
- 10 – Potrebno je zaliti drugu saksiju
- 11 – Potrebno je zaliti obje saksije

Senzor vlage smo kalibrirali tako da se signalizira zaljevanje kada je procenat vlage ispod 30%. Ovisno o senzoru i tipu zemlje, ova vrijednost je podložna promjenama. Podatci koje predajnik šalje su decimalni brojevi: 13 i 14 za prvu saksiju, te 15 i 16 za drugu saksiju. Podatke smo namjestili ovako jer će doći do preklapanja, tj. svi podatci koji se serijski šalju moraju biti jedinstveni.

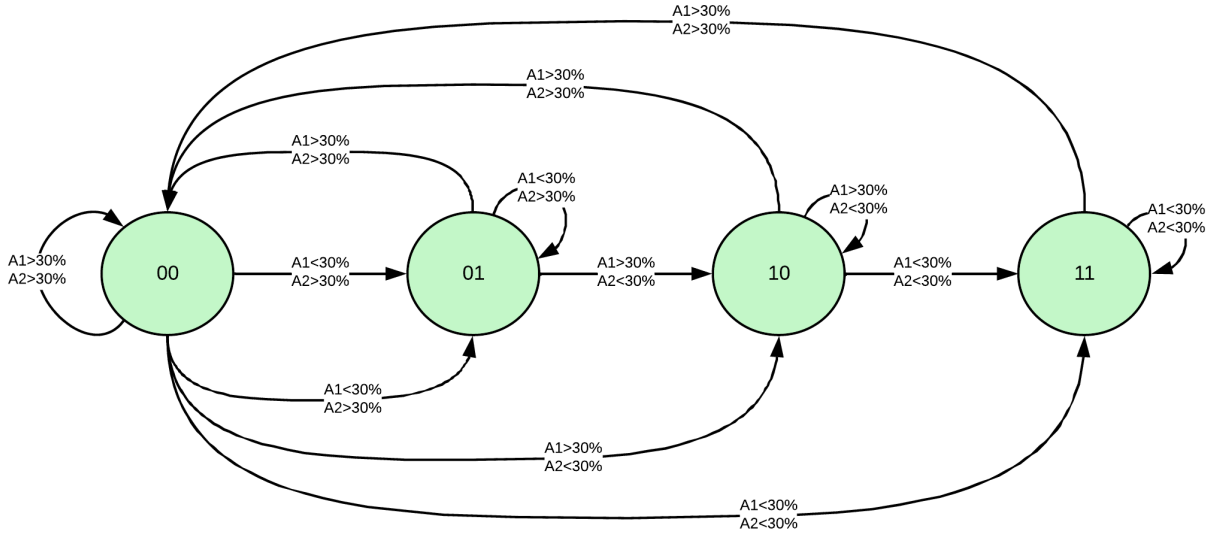
Kada je bašta u stanju 00, WiFi predajnici su očitali vrijednost sa senzora vlage i odlučili da zaljevanje nije potrebno. Predajnici potom šalju podatak prijemniku koji se nalazi na mobilnom robotu, i to decimalne brojeve 13 i 15 respektivno.

Kada je bašta u stanju 01, prvi predajnik je očitao da je potrebno zaliti prvu saksiju, dok kod drugog predajnika to nije slučaj. Samim tim, prvi predajnik će signalizirati zaljevanje brojem 14, dok će drugi ponovno slati 15.

Kada je bašta u stanju 10, drugi predajnik je očitao da je potrebno zaliti drugu saksiju, dok kod prvog to nije slučaj. Ovoga puta, prvi predajnik će poslati broj 13 što označava da nema potrebe za zaljevanjem, a drugi predajnik šalje 16, što označava potrebu za zaljevanjem druge saksije.

Kada je bašta u stanju 11, oba predajnika šalju signale za zaljevanje, odnosno brojeve 14 i 16 respektivno.

Mehanizam bašte je najprikladije opisan mašinom konačnog stanja (FSM) prikazanom ispod (senzori vlage su na dijagramu označeni sa A1 i A2 respektivno)



Slika 20: FSM Bašte

Kod za prikupljanje podataka sa senzora vlage i slanje odgovarajuće poruke prijemniku je prikazan ispod:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const int sensor_pin = A0;
int sensorValue = 10;

const char* ssid ="NAZIV WIFI MREZE";
const char* password ="PASSWORD";
const char *mqtt_broker="IP ADRESA";//ovi podatci moraju biti isti kod
prijemnika i predajnika, tako da se uspostavi sigurna komunikacija

char* message=0;
WiFiClient wifiClient;
PubSubClient client(wifiClient);

int status = WL_IDLE_STATUS;

void setup()
{
      Serial.begin(9600);
      WiFi.begin(ssid, password);

      while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
      }
      Serial.println("Connected to WiFi");
      Serial.println("ESP8266 AS PUBLISHER");
      client.setServer(mqtt_broker, 1883
```

```
void loop()
{
if ( !client.connected() )
{
reconnect();
}
readAndPublishData();
delay(60000); //podatak se ažurira svaki minut
}


void readAndPublishData()
{
  if(client.connected())
  {
    delay(1000);
    float moisture_percentage = ( 100.00 -
( (analogRead(sensor_pin)/1024.00) * 100.00 ) );//očitavanje
vrijednosti analognog pina senzora vlage
    if(moisture_percentage <= 30) sensorValue = 14;//na osnovu
procenta vlage, dodjeljuje se odgovarajuća decimalna vrijednost
    else sensorValue = 13;
    char msg[5];
    sprintf(msg,"%d",sensorValue);
    Serial.print(msg);
    client.publish("PSCgr5/garden/2023",msg);
    Serial.println(" - sent");
  }
}
void reconnect() {
  while (!client.connected()) {
    status = WiFi.status();
    if (status != WL_CONNECTED) {
      WiFi.begin(ssid, password);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      }
      Serial.println("Connected to AP");
    }
    Serial.print("Connecting to Broker …");
    Serial.print("192.168.1.11");

    if (client.connect("ESP8266 Device", TOKEN, NULL)) {
      Serial.println("[DONE]");
    } else {
      Serial.println(" : retrying in 5 seconds]");
      delay(5000);
    }
  }
}
```

4.2. Mobilni robot

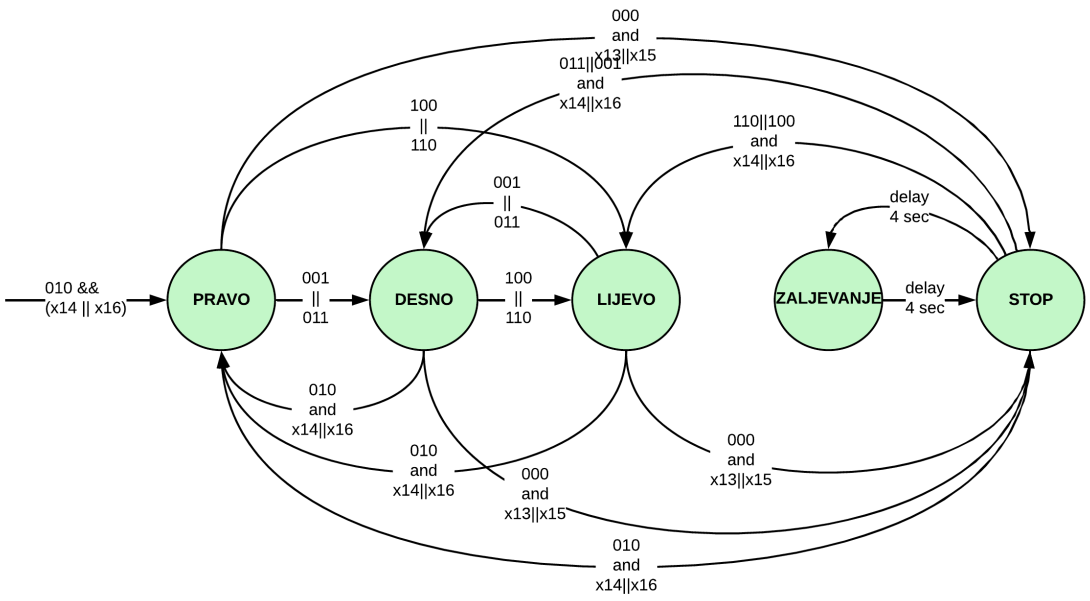
Nakon što bašta pošalje signal, tada se fokus usmjerava na mobilnog robota, odnosno drugi dio našeg projekta. Kako je i ranije naznačeno, svaki podatak koji pristigne od predajnika (bašte) direktno utiče na ponašanje mobilnog robota, i to će biti detaljno objašnjeno u nastavku.

Tx pin prijemnika (NodeMCU) je vezan za Rx pin Cyclone II/Raspberry Pi, putem kojeg podatci serijski pristižu iz bašte. Taj prvi ulaz u sistem određuje prvi korak našeg mobilnog robota, tj. ukoliko pristignu podatci koji signaliziraju da je bašta zalivena i da nije potrebno zaljevanje, robot će ostati u stanju mirovanja. S druge strane, ukoliko barem jedna od saksija signalizira da je neophodno zaljevanje (pristigao je podatak x14 ili x16) robot će početi kretanje po crnoj liniji (unaprijed definisana putanja). Ovo kretanje je kontrolisano IR senzorima koji daju izlaz 0 ili 1, ovisno da li se radi o crnoj ili bijeloj podlozi. Ovi senzori generalno svrhu pronalaze u detektovanju udaljenosti objekata, međutim dobri su i za korištenje pri praćenju linije.

Mobilni robot će pratiti liniju sve dok ne dođe do RFID taga saksije koju treba zaliti. Kada RFID čitač očita odgovarajuću karticu, robot će se zaustaviti. Nakon toga slijedi kraća pauza, i pokreće se pumpa u rezervoaru s vodom. Proces zaljevanja traje cca 4 sekunde i onda se pumpa gasi. Opet ide kraća pauza (da se voda iz cijevi ne bi prelila izvan saksije) i robot nastavlja kretanje dok podatci koji pristižu ne odrede da zaljevanje više nije potrebno. RFID tagovi su također parsirani tako da kada čitač prepozna određeni tag, na prijemnika proslijedi decimalni broj slično kao što to urade i predajnici iz bašte. S ovim mehanizmom je osigurano da će uvijek prvo pristići podatak iz bašte, pa tek onda podatak koji predstavlja adresu očitanog taga. Time možemo prilagoditi naš program tako da tačno određena adresa taga predstavlja zapravo saksiju koju treba zaliti. U našem slučaju imamo dvije saksije, dakle dvije RFID kartice s adresama koje smo parsirali u decimalne vrijednosti 5 i 6, odnosno serijski proslijeđeno kao b'\x05' i b'\x06'.

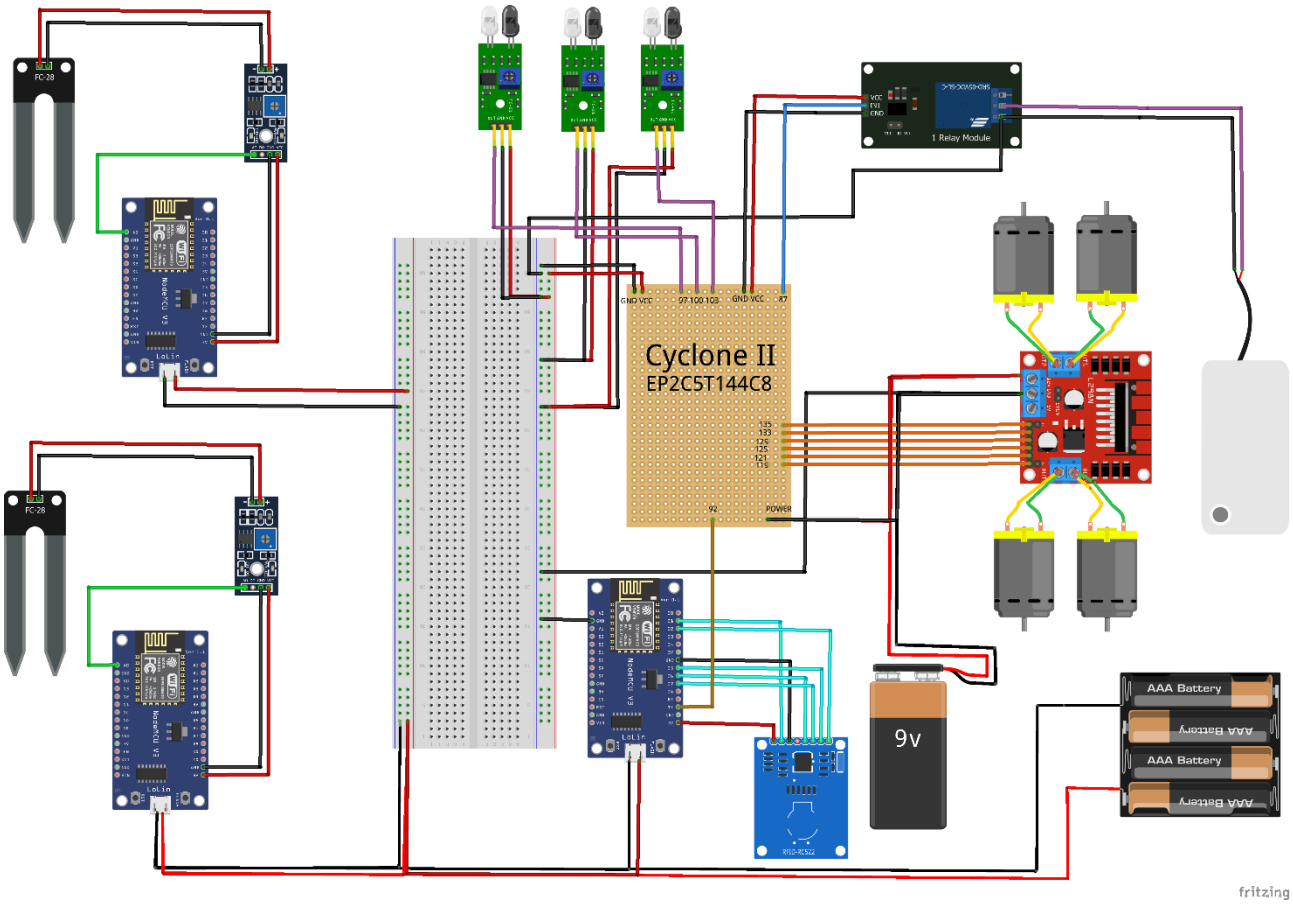
Detaljna specifikacija ulaza/izlaza i FSM su prikazani ispod:

ULAZI	IZLAZI
IR senzori (praćenje crne linije)	DC Motori (točkovi robota)
Rx (podatak koji dolazi sa NodeMCU)	Pumpa (zaljevanje)



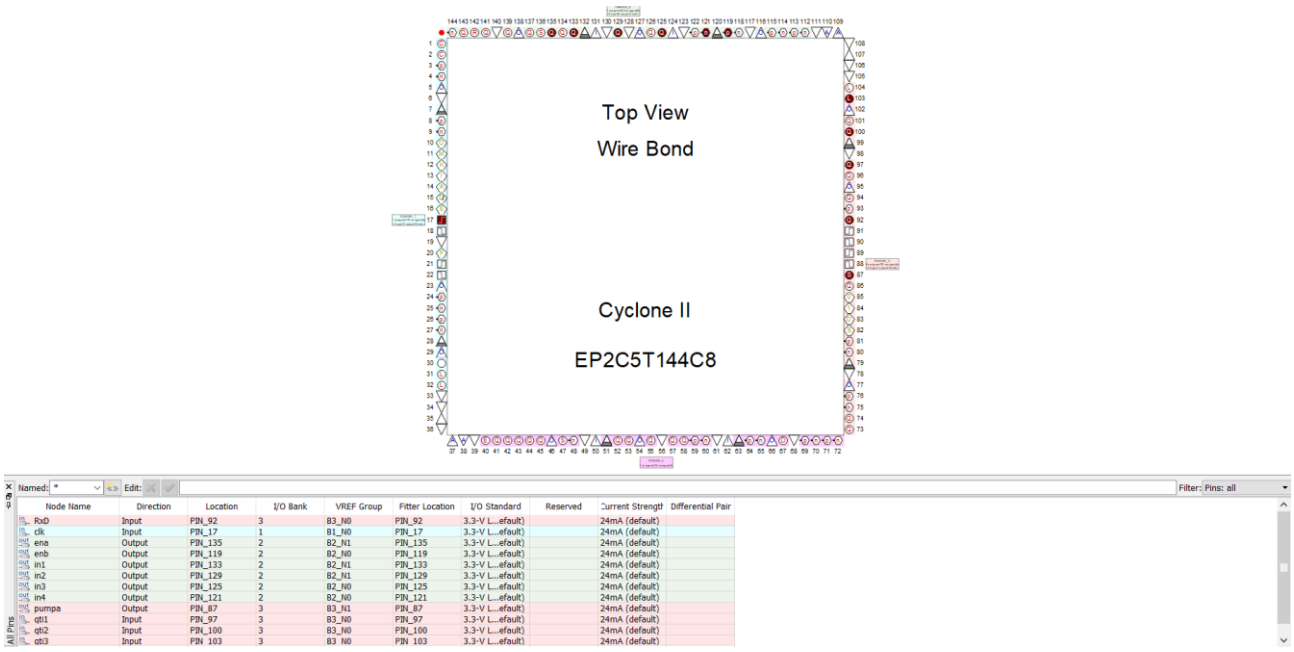
Slika 21: FSM Mobilni robot

4.3. Shema povezivanja



Slika 22: Shema spajanja - FPGA

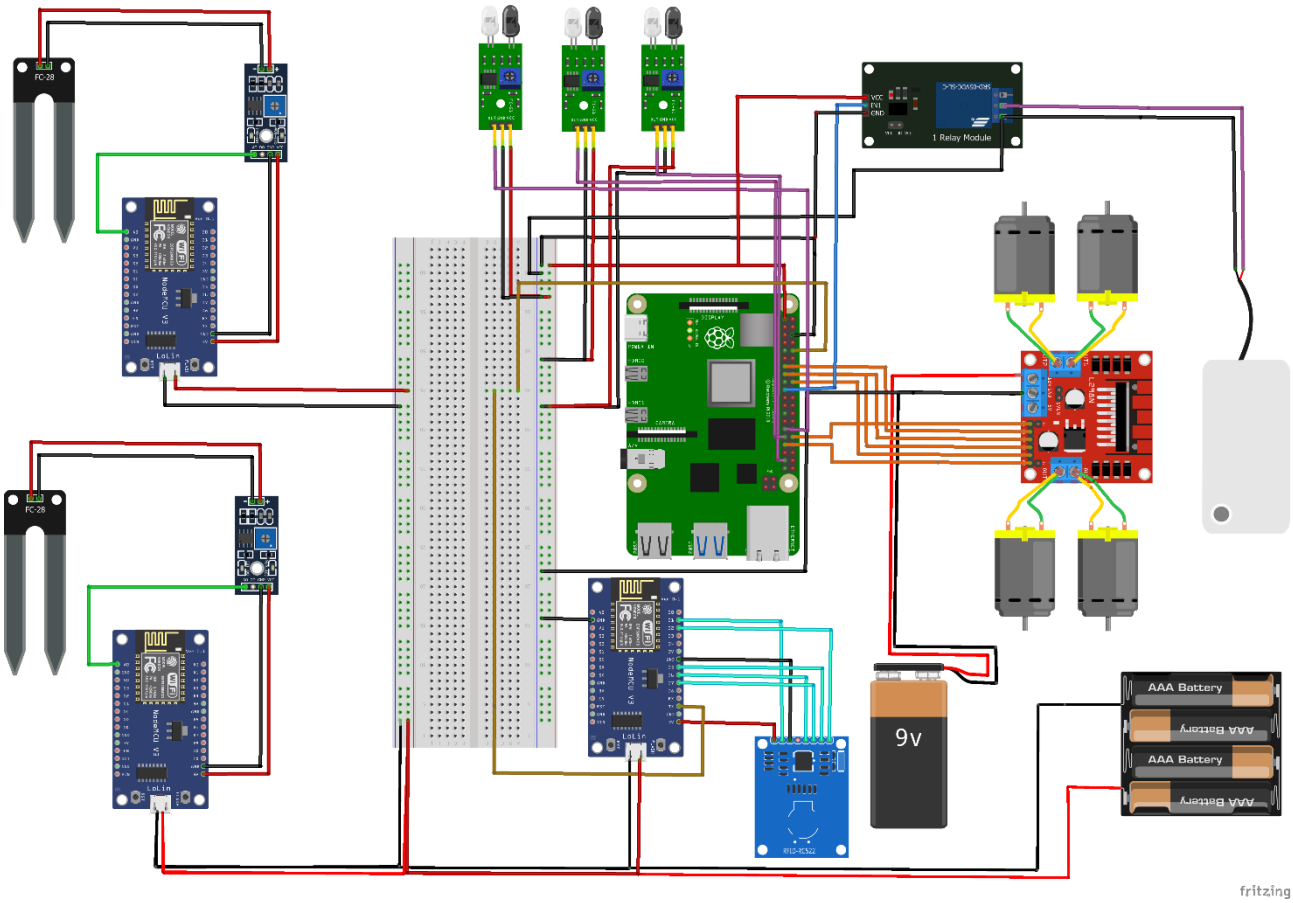
Na shemi iznad je prikazano kompletno uvezivanje našeg sistema sa Cyclone II FPGA pločicom, dok na lijevoj strani vidimo dva mikrokontrolera koja predstavljaju baštu. Iako su na shemi prikazani kao spojeni na isti breadboard i napajanje, uživo to nije slučaj, tj. saksije imaju vlastito napajanje uz pomoć Powerbank-a. Ispod je prikazan i Pin Planner, gdje se vidi tačan raspored svih korištenih pinova na Cyclone II pločici.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strengt	Differential Pair
RxD	Input	PIN_92	3	B3_N0	PIN_92	3.3-V L...efault)		24mA (default)	
clk	Input	PIN_17	1	B1_N0	PIN_17	3.3-V L...efault)		24mA (default)	
ena	Output	PIN_135	2	B2_N1	PIN_135	3.3-V L...efault)		24mA (default)	
enb	Output	PIN_119	2	B2_N0	PIN_119	3.3-V L...efault)		24mA (default)	
in1	Output	PIN_133	2	B2_N1	PIN_133	3.3-V L...efault)		24mA (default)	
in2	Output	PIN_129	2	B2_N1	PIN_129	3.3-V L...efault)		24mA (default)	
in3	Output	PIN_125	2	B2_N0	PIN_125	3.3-V L...efault)		24mA (default)	
in4	Output	PIN_121	2	B2_N0	PIN_121	3.3-V L...efault)		24mA (default)	
pumpa	Output	PIN_87	3	B3_N1	PIN_87	3.3-V L...efault)		24mA (default)	
qti1	Input	PIN_97	3	B3_N0	PIN_97	3.3-V L...efault)		24mA (default)	
qti2	Input	PIN_100	3	B3_N0	PIN_100	3.3-V L...efault)		24mA (default)	
qti3	Input	PIN_103	3	B3_N0	PIN_103	3.3-V L...efault)		24mA (default)	

Slika 23: Pin Planner

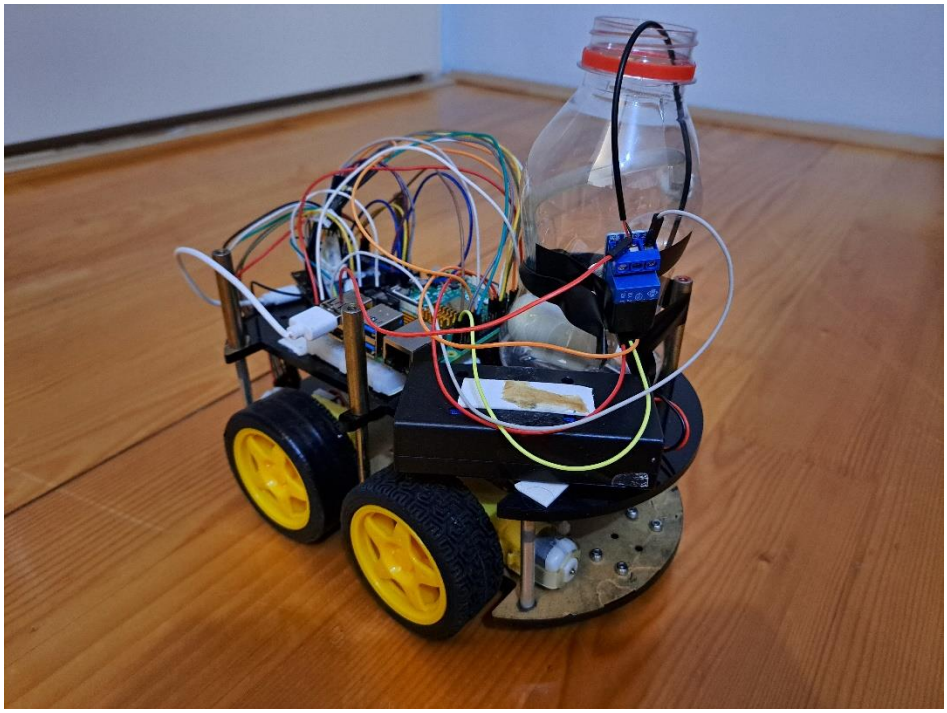
Na sličan način je izvedeno uvezivanje i na Raspberry Pi, prikazano u nastavku.



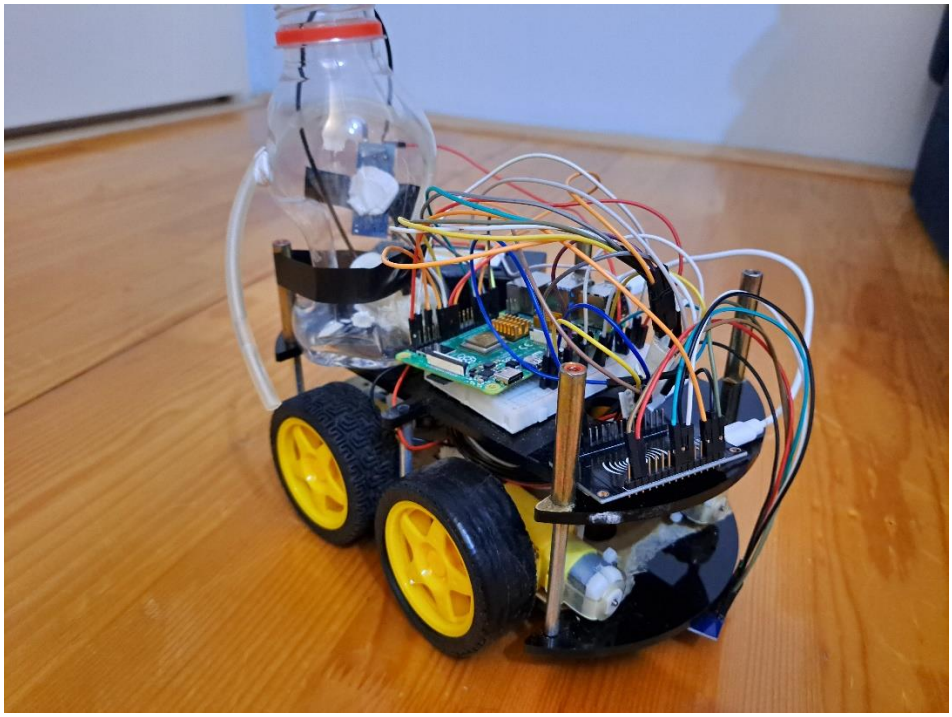
Slika 24: Shema povezivanja - Raspberry Pi

4.4. Fizička izvedba

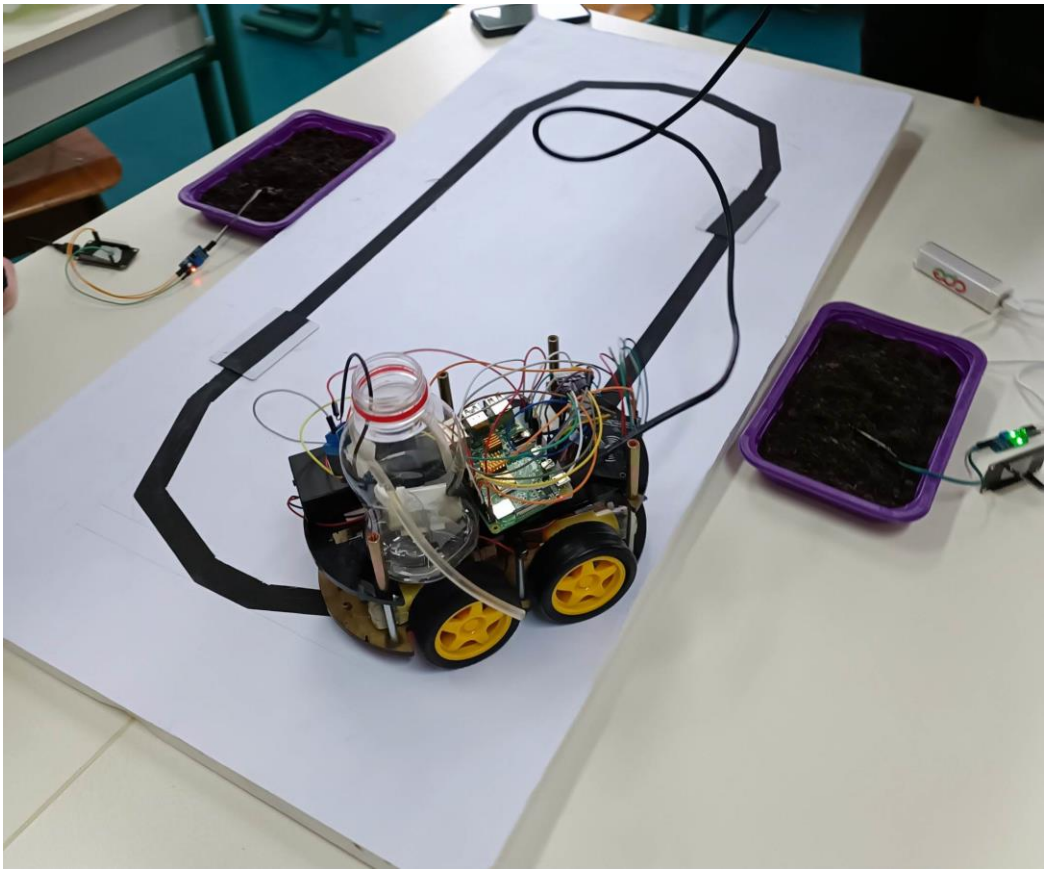
Na fotografijama u nastavku je prikazana fizička izvedba našeg mobilnog robota, kao i kompletne makete tj. modela bašte.



Slika 25: Fizička izvedba robota (prednja strana)



Slika 26: Fizička izvedba robota (zadnja strana)



Slika 27: Smart garden

5. Grafička interpretacija rezultata

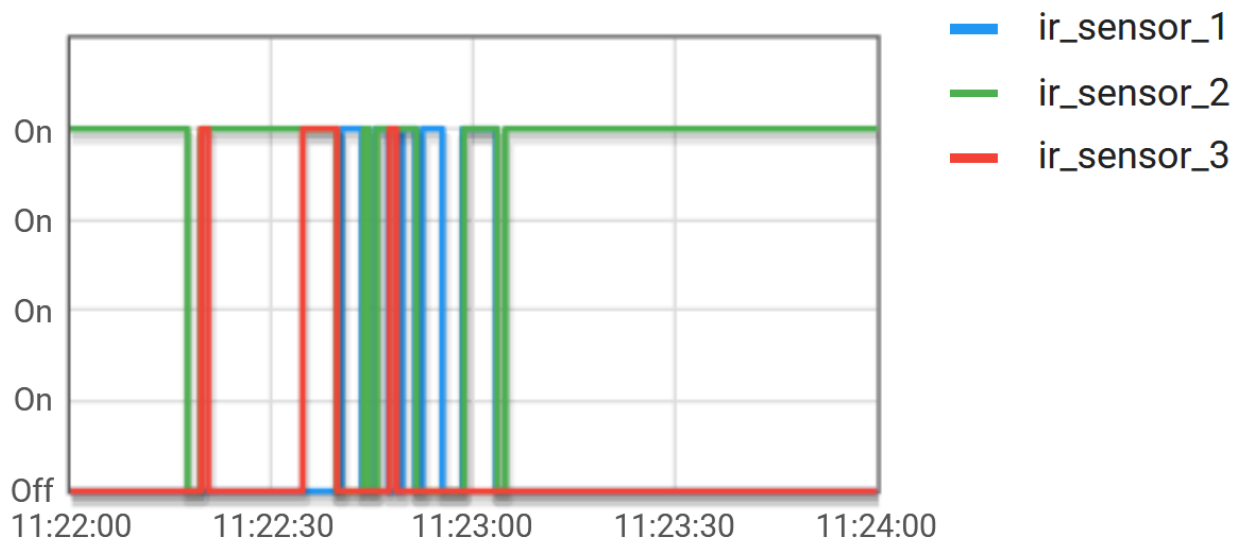
Za grafičku interpretaciju našeg projekta, koristili smo Arduino, Thingsboard i MATLAB. Prvo ćemo prikazati različite slučajeve praćenja robota, a potom i grafikon iz bašte, odnosno sa senzora vlage.

5.1. Grafički prikaz praćenja robota na osnovu IR senzora i RFID čitača

U svrhu prikaza real-time rezultata, koristili smo Arduino i Thingsboard. U arduinu smo očitavali vrijednosti senzora i čitača, dok smo Thingsboard koristili za plotanje rezultata, obzirom da se radi o diskretnim vrijednostima, pogodnije je u odnosu na MATLAB jer direktno prosljeđujemo sve podatke u realnom vremenu. Kod za akviziciju podataka je prikazan ispod:

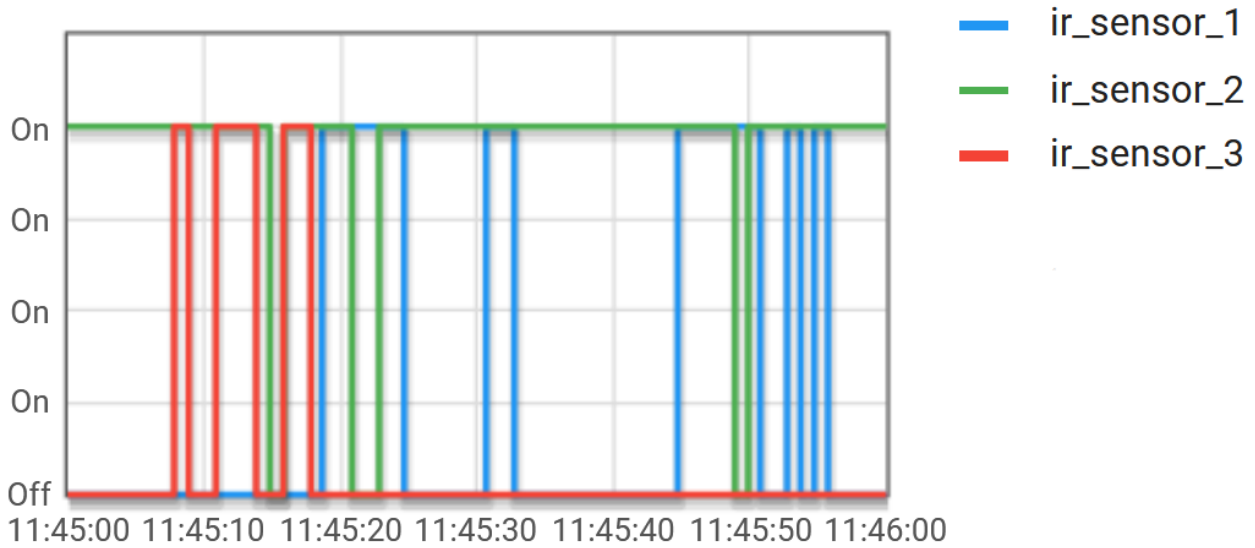
```
#include "ThingsBoard.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <MFRC522.h>

constexpr uint8_t RST_PIN = 5;
constexpr uint8_t SS_PIN = 4;
MFRC522 rfid(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;
byte nuidPICC[4];
uint8 q = 1;

#define WIFI_AP "WiFi"
#define WIFI_PASSWORD "SIFRA"

#define TOKEN "PrvPRVjkTgXjB6q8z8h1"  //P R O M I J E N I T I
#define THINGSBOARD_SERVER "demo.thingsboard.io"
#define SERIAL_DEBUG_BAUD 115200

WiFiClient espClient;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;

int number;
int ir_1_tmp = 5;   //D1
int ir_2_tmp = 4;   //D2
int ir_3_tmp = 14;  //D5

void setup() {
  Serial.begin(SERIAL_DEBUG_BAUD);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();
SPI.begin();      // Init SPI bus
  rfid.PCD_Init();  // Init MFRC522
  pinMode(ir_1_tmp, INPUT);
```

```
  pinMode(ir_2_tmp, INPUT);
  pinMode(ir_3_tmp, INPUT);
}
void loop() {
  delay(1000);
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
  if (!rfid.PICC_IsNewCardPresent())
    return;
  if (rfid.PICC_ReadCardSerial()) {
    String rfidCard = toString(rfid.uid.uidByte, rfid.uid.size);
    checkCard(rfidCard);
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
  }
  if (!tb.connected()) {
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }
  Serial.println("Sending data...");
  int ir_1 = digitalRead(ir_1_tmp);
  int ir_2 = digitalRead(ir_2_tmp);
  int ir_3 = digitalRead(ir_3_tmp);
  tb.sendTelemetryInt("ir_sensor_1", ir_1);
  tb.sendTelemetryInt("ir_sensor_2", ir_2);
  tb.sendTelemetryInt("ir_sensor_3", ir_3);
  tb.loop();
}

String toString(byte* buffer, byte bufferSize) {
  String rfidCard;
  for (int i = 0; i < bufferSize; i++) {
    rfidCard += String(buffer[i]);
  }
  // Serial.print(rfidCard);
  return rfidCard;
}
void checkCard(String rfidCard) {
  if (rfidCard == "2301257426") {
    number = 1;
    tb.sendTelemetryInt("RFID x05", number);
  }
  if (rfidCard == "2262102827") {
    number = 1;
    tb.sendTelemetryInt("RFID x06", number);
  }}
```



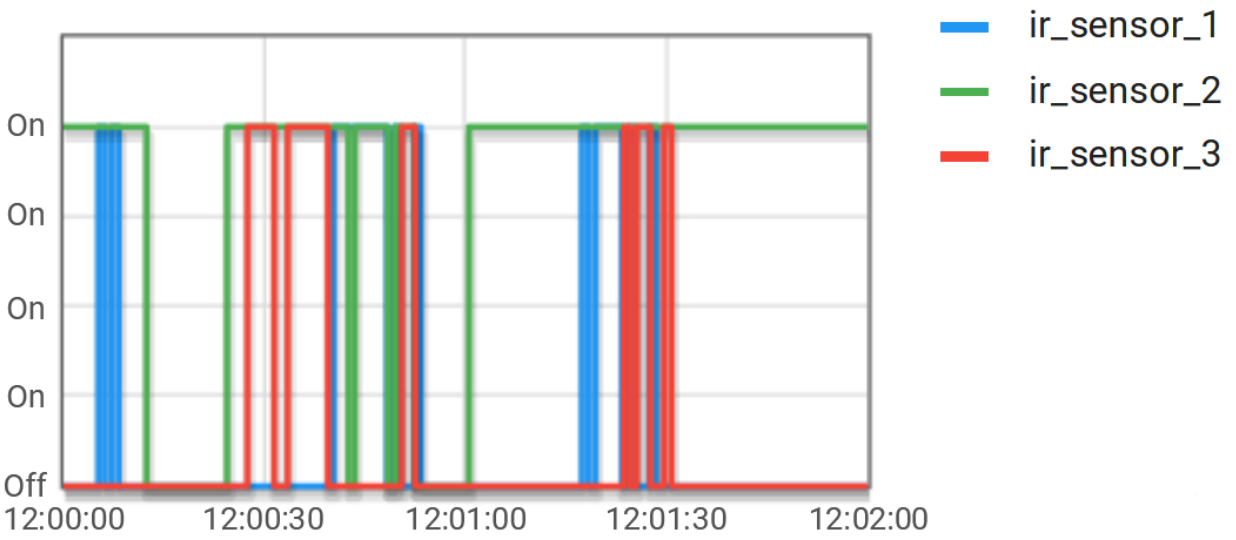
Slika 28: Prvi slučaj: Prikaz zaljevanja prve saksije

Na slici 28 vidimo prvi slučaj odnosno zaljevanje prve saksije. Kao što se da primijetiti, IR senzor 2 jedini očitava logičku jedinicu zato što tada robot stoji u mjestu i nema kalibriranja desno-lijevo. U ostalim vremenskim periodima vidimo kalibraciju odnosno različita očitanja prvog i trećeg IR senzora što označava da se robot uredno kreće po liniji.



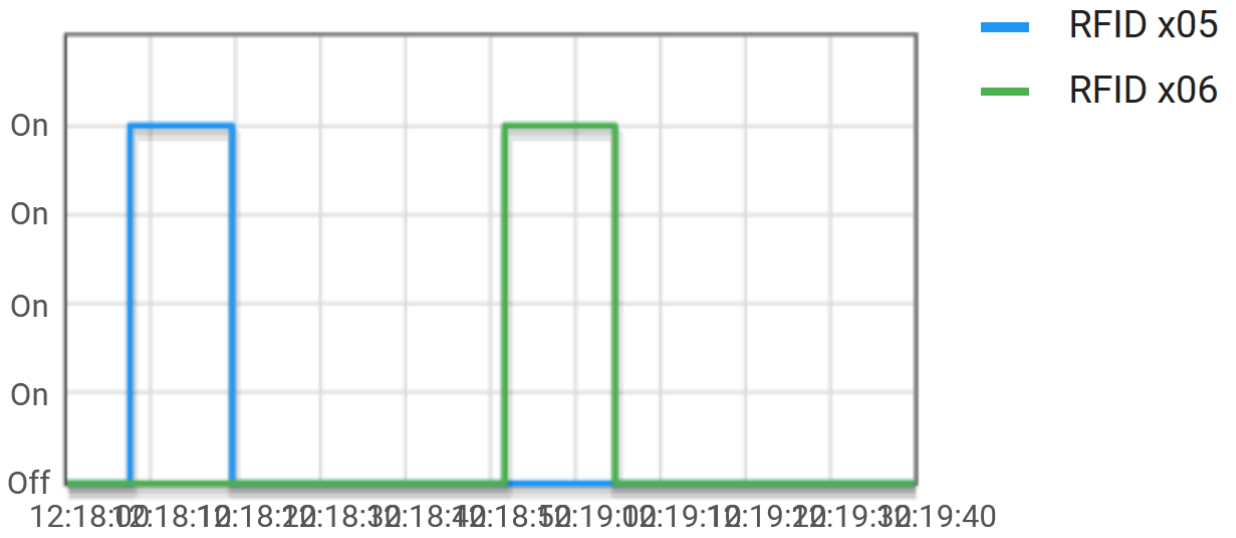
Slika 29: Drugi slučaj: Priakz zaljevanja druge saksije

Na slici 29 je prikazan drugi slučaj, odnosno zaljevanje druge saksije. Slično kao i u prvom sliučaju vidimo da je robot stojao i zaljevao u periodu kada je IR2 bio u stanju logičke jedinice. Obzirom da se druga saksija nalazi na drugom dijelu putanje, tj udaljena je od prve, to se isto može zaključiti i sa slike.



Slika 30: Treći slučaj, zaljevanje obje saksije

Na slici 30, odnosno na trećem slučaju vidimo da se je robot zaustavljao dva puta, tj. kada je zaljevao i prvu i drugu saksiju. U svim ostalim instancama je kalibrirao putanju, tačnije IR senzori 1 i 3 su naizmjenično davali logičke jedinice kada su očitavali crnu putanju.



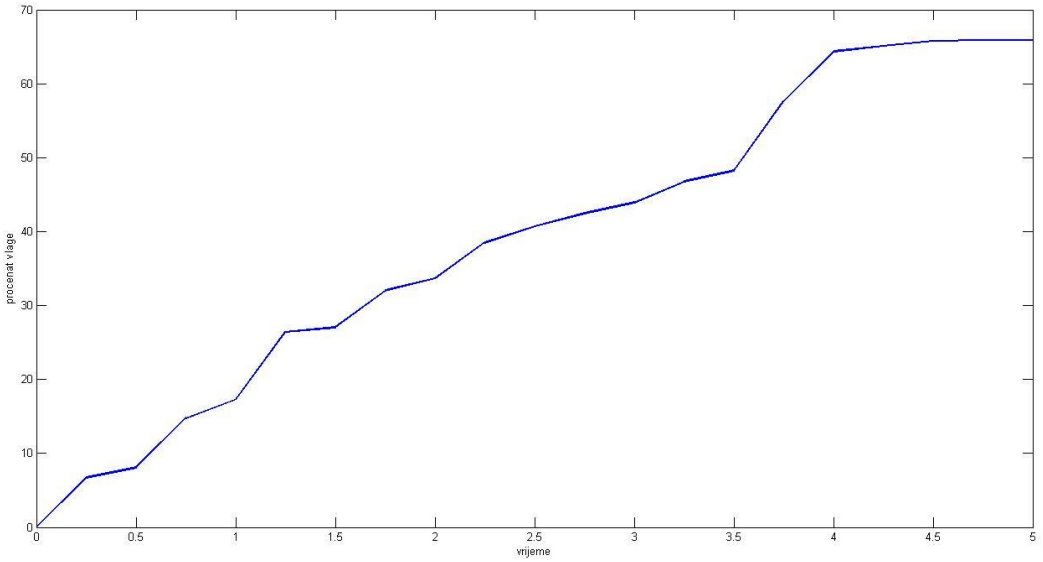
Slika 31: Grafički prikaz očitanja RFID kartica čitačem

Na slici 31 vidimo grafički prikaz očitavanja RFID kartica. Obzirom da čitač uvijek prepozna karticu, ovaj prikaz je jednak za sva tri slučaja. Razlika je u tome što na osnovu koda, u određenim slučajevima će pojedine kartice biti „ignorisane“ dok će se za druge izvršavati zadatak zaustavljanja i zaljevanja.

5.2. Grafički prikaz procesa zaljevanja

Sam proces zaljevanja ovisi u mnogome od senzora vlage koji se koristi. YL-69 senzor vlage je prigodan za demonstraciju ovakvih i sličnih projekata, jer ga je vrlo lako kalibrirati i prilagoditi nivou vlage koji je tražen. U nastavku je prikazan real-time grafikon koji demonstrira proces zaljevanja. Podatke sa senzora u toku procesa zaljevanja smo pokupili u tabelu koju smo potom plotali u MATLABu.

Kao što se da primijetiti, 5 sekundi zaljevanja našim mobilnim robotom je i više nego dovoljno za postizanje željenog nivoa vlage. Ranije smo prikazali da ciljamo na 30% vlage kao graničnu vrijednost za pokretanje zaljevanja u svrhu bolje implementacije, međutim vidimo da zaljevanje koje smo implementirali našim mobilnim robotom postiže dovoljan nivo vlage i kada se kreće od potpuno suhe zemlje.



Slika 32: Real-time prikaz procesa zaljevanja

```
10:44:56.599 -> 0.00
10:44:56.799 -> 6.73
10:44:57.094 -> 8.11
10:44:57.344 -> 14.79
10:44:57.595 -> 17.30
10:44:57.845 -> 26.43
10:44:58.110 -> 27.05
10:44:58.360 -> 32.01
10:44:58.582 -> 33.69
10:44:58.832 -> 38.51
10:44:59.108 -> 40.72
10:44:59.617 -> 42.48
10:45:00.070 -> 43.95
10:45:00.320 -> 46.77
10:45:00.586 -> 48.24
10:45:01.102 -> 64.36
10:45:01.352 -> 65.88
10:45:01.602 -> 65.88
```

Slika 33: Podatci prikupljeni sa senzora vlage u intervalu 5 sekundi

6. Dodatak – kodovi

6.1. Verilog kod – Cyclone II FPGA

```
module smart_garden(
      input clk,
      input qti1,input qti2,input qti3,
      output reg in1,output reg in2,output reg in3,
      output reg in4,output wire ena,output wire enb, input RxD, output
reg pumpa
);

reg[45:0] impuls_dc;
wire RxD_data_ready;
wire [7:0] RxD_data;

reg[29:0] cnt1=30'd0;
reg[29:0] donja=30'd250000000;
reg[29:0] gornja=30'd500000000;

reg[7:0] zaljevanje=0;
reg[7:0] saksija=0;

//PWM za DC motore
initial impuls_dc=45'd350000;
always @(posedge clk)
begin
if( brojac1<impuls_dc ) //Generisanje impulsa
begin
izlaz_dc<=45'b1;
brojac1<=brojac1+45'd1;
end
else if( brojac1 >= impuls_dc && brojac1 < impuls_dc+45'd1000000)
//Pauza 20ms
begin
brojac1<=brojac1+45'd1;
izlaz_dc<=45'b0;
end
else //Resetovanje brojača
begin
brojac1<=45'd0;
end
end

//Dio koda za serijski prenos podataka

reg RxD_endofpacket = 0;
wire RxD_idle;

parameter ClkFrequency = 50000000; // 25MHz
parameter Baud = 115200;

parameter Oversampling = 8;
```

```
generate
if(ClkFrequency<Baud*Oversampling) ASSERTION_ERROR
PARAMETER_OUT_OF_RANGE("Frequency too low for current Baud rate and
oversampling");
if(Oversampling<8 || ((Oversampling & (Oversampling-1))!=0))
ASSERTION_ERROR PARAMETER_OUT_OF_RANGE("Invalid oversampling value");
endgenerate

reg [3:0] RxD_state = 0;

`ifdef SIMULATION
wire RxD_bit = RxD;
wire sampleNow = 1'b1;  // receive one bit per clock cycle

`else
wire OversamplingTick;
BaudTickGen #(ClkFrequency, Baud, Oversampling)
tickgen(.clk(clk), .enable(1'b1), .tick(OversamplingTick));

// synchronize RxD to our clk domain
reg [1:0] RxD_sync = 2'b11;
always @(posedge clk) if(OversamplingTick) RxD_sync <= {RxD_sync[0],
RxD};

// and filter it
reg [1:0] Filter_cnt = 2'b11;
reg RxD_bit = 1'b1;

always @(posedge clk)
if(OversamplingTick)
begin
if(RxD_sync[1]==1'b1 && Filter_cnt!=2'b11) Filter_cnt <= Filter_cnt +
1'd1;
else
if(RxD_sync[1]==1'b0 && Filter_cnt!=2'b00) Filter_cnt <= Filter_cnt -
1'd1;

if(Filter_cnt==2'b11) RxD_bit <= 1'b1;
else
if(Filter_cnt==2'b00) RxD_bit <= 1'b0;
end

// and decide when is the good time to sample the RxD line
function integer log2(input integer v); begin log2=0; while(v>>log2)
log2=log2+1; end endfunction
localparam l2o = log2(Oversampling);
reg [l2o-2:0] OversamplingCnt = 0;
always @(posedge clk) if(OversamplingTick) OversamplingCnt <=
(RxD_state==0) ? 1'd0 : OversamplingCnt + 1'd1;
wire sampleNow = OversamplingTick && (OversamplingCnt==Oversampling/2-
1);
`endif
```

```
always @(posedge clk)
case(RxD_state)
4'b0000: if(~RxD_bit) RxD_state <= `ifdef SIMULATION 4'b1000 `else
4'b0001 `endif;  // start bit found?
4'b0001: if(sampleNow) RxD_state <= 4'b1000;  // sync start bit to
sampleNow
4'b1000: if(sampleNow) RxD_state <= 4'b1001;  // bit 0
4'b1001: if(sampleNow) RxD_state <= 4'b1010;  // bit 1
4'b1010: if(sampleNow) RxD_state <= 4'b1011;  // bit 2
4'b1011: if(sampleNow) RxD_state <= 4'b1100;  // bit 3
4'b1100: if(sampleNow) RxD_state <= 4'b1101;  // bit 4
4'b1101: if(sampleNow) RxD_state <= 4'b1110;  // bit 5
4'b1110: if(sampleNow) RxD_state <= 4'b1111;  // bit 6
4'b1111: if(sampleNow) RxD_state <= 4'b0010;  // bit 7
4'b0010: if(sampleNow) RxD_state <= 4'b0000;  // stop bit
default: RxD_state <= 4'b0000;
endcase

always @(posedge clk)
if(sampleNow && RxD_state[3]) RxD_data <= {RxD_bit, RxD_data[7:1]};

//reg RxD_data_error = 0;
always @(posedge clk)
begin
RxD_data_ready <= (sampleNow && RxD_state==4'b0010 && RxD_bit);  //
make sure a stop bit is received
//RxD_data_error <= (sampleNow && RxD_state==4'b0010 && ~RxD_bit);  //
error if a stop bit is not received
end

`ifdef SIMULATION
assign RxD_idle = 0;
`else
reg [l2o+1:0] GapCnt = 0;
always @(posedge clk) if (RxD_state!=0) GapCnt<=0; else
if(OversamplingTick & ~GapCnt[log2(Oversampling)+1]) GapCnt <= GapCnt
+ 1'h1;
assign RxD_idle = GapCnt[l2o+1];
always @(posedge clk) RxD_endofpacket <= OversamplingTick &
~GapCnt[l2o+1] & &GapCnt[l2o:0];
`endif

endmodule

//////////////////////////////////////////////////////
// dummy module used to be able to raise an assertion in Verilog
module ASSERTION_ERROR();
endmodule

//////////////////////////////////////////////////////
module BaudTickGen(
input clk, enable,
output tick  // generate a tick at the specified baud rate *
oversampling
);
```

```
parameter ClkFrequency = 25000000;
parameter Baud = 115200;
parameter Oversampling = 1;

function integer log2(input integer v); begin log2=0; while(v>>log2)
log2=log2+1; end endfunction
localparam AccWidth = log2(ClkFrequency/Baud)+8;  // +/- 2% max timing
error over a byte
reg [AccWidth:0] Acc = 0;
localparam ShiftLimiter = log2(Baud*Oversampling >> (31-AccWidth));
// this makes sure Inc calculation doesn't overflow
localparam Inc = ((Baud*Oversampling << (AccWidth-
ShiftLimiter))+(ClkFrequency>>(ShiftLimiter+1)))/(ClkFrequency>>ShiftL
imiter);
always @(posedge clk) if(enable) Acc <= Acc[AccWidth-1:0] +
Inc[AccWidth:0]; else Acc <= Inc[AccWidth:0];
assign tick = Acc[AccWidth];

//Glavni dio koda
//Prvo čuvamo podatke koji su pristigli u registre „zaljevanje“ i
„saksija“, tako da lakše možemo implementirati samu funkcionalnost
always @ (posedge clk)
begin
if (RxD_data==8'd13 || RxD_data==8'd14 || RxD_data==8'd15 ||
RxD_data==8'd16)
begin
zaljevanje = RxD_data
end

if (RxD_data==8'd5 || RxD_data==8'd6)
begin
saksija = RxD_data
end

if (zaljevanje == 8'd14)
begin
      if(saksija==8'd5)
      begin
            if(cnt1<donja)
            begin
            pumpa<=1;
            followLine(qti1,qti2,qti3,in1,in2,in3,in4);
            end
            else if(cnt1>=donja && cnt1<gornja)
            begin
            stop(in1,in2,in3,in4);
            pumpa<=0;
            end
            else if(cnt3>=cnttop)
            begin
            cnt1=0;
            end
      cnt1=cnt1+1;
      end
      else if(saksija==8'd6)
```

```
      begin
      followLine(qti1,qti2,qti3,in1,in2,in3,in4);
      end
end

else if (zaljevanje == 8'd16)
begin
      if(saksija==8'd6)
      begin
            if(cnt1<donja)
            begin
            pumpa<=1;
            followLine(qti1,qti2,qti3,in1,in2,in3,in4);
            end
            else if(cnt1>=donja && cnt1<gornja)
            begin
            stop(in1,in2,in3,in4);
            pumpa<=0;
            end
            else if(cnt3>=cnttop)
            begin
            cnt1=0;
            end
      cnt1=cnt1+1;
      end
      else if(saksija==8'd5)
      begin
      followLine(qti1,qti2,qti3,in1,in2,in3,in4);
      end
end

else
begin
stop(in1,in2,in3,in4)
end

//Funkcija za praćenje crne linije
task followLine;
input qti1,qti2,qti3;
output in1,in2,in3,in4;
begin
      if((qti1==1 && qti2==1 && qti3==0) || (qti1==1 && qti2==0 &&
qti3==0) )
      begin
      move_left(in1,in2,in3,in4);
      end
      else if((qti1==0 && qti2==1 && qti3==1)||(qti1==0 && qti2==0 &&
qti3==1))
      begin
      move_right(in1,in2,in3,in4);
      end
      else if(qti1==0 && qti2==0 && qti3==0)
      begin
      move_forward(in1,in2,in3,in4);
      end
```

```
else if(qti1==0 && qti2==1 && qti3==0)
      begin
      stop(in1,in2,in3,in4);
      end
      else
      begin
      stop(in1,in2,in3,in4);
      end
end
endtask

//Funkcije za kontrolisanje pravca robota
task move_left;
output in1,in2,in3,in4;
begin
      impuls_dc<=45'd700000;
      in1<= izlaz_dc;
      in2<=1'b0;
      in3<= izlaz_dc;
      in4<=1'b0;
      end
endtask

task move_right;
output in1,in2,in3,in4;
begin
      impuls_dc<=45'd700000;
      in1<=1'b0;
      in2<= izlaz_dc;
      in3<=1'b0;
      in4<= izlaz_dc;
end
endtask

task move_forward;
output in1,in2,in3,in4;
begin
      impuls_dc<=45'd700000;
      in1<= izlaz_dc;
      in2<=1'b0;
      in3<=1'b0;
      in4<= izlaz_dc;
end
endtask

task stop;
output in1,in2,in3,in4;
begin
      impuls_dc<=45'd700000;
      in1<=1'b1;
      in2<=1'b1;
      in3<=1'b1;
      in4<=1'b1;
end
endtask
```

6.2. Python kod – Raspberry Pi

```
import RPi.GPIO as GPIO
from time import sleep
import serial
import time

ser = serial.Serial("/dev/ttyS0", 9600)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
M1 = 27
M2 = 22
M3 = 23
M4 = 24
pumpa = 10

GPIO.setup(M1, GPIO.OUT, initial=GPIO.HIGH)
GPIO.setup(M2, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(M3, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(M4, GPIO.OUT, initial=GPIO.HIGH)
GPIO.setup(pumpa, GPIO.OUT, initial=GPIO.HIGH)

ENa = 12
ENb = 13
GPIO.setup(ENa, GPIO.OUT)
ENa_pwm = GPIO.PWM(ENa, 50)
ENa_pwm.start(10)
GPIO.setup(ENb, GPIO.OUT)
ENb_pwm = GPIO.PWM(ENb, 50)
ENb_pwm.start(10)

ir1 = 5
ir2 = 6
ir3 = 26
GPIO.setup(ir1, GPIO.IN)
GPIO.setup(ir2, GPIO.IN)
GPIO.setup(ir3, GPIO.IN)

#Funkcije za praćenje linije i zaustavljanje robota
def stopMoving():
    GPIO.output(M1, GPIO.LOW)
    GPIO.output(M2, GPIO.LOW)
    GPIO.output(M3, GPIO.LOW)
    GPIO.output(M4, GPIO.LOW)

def followLine(vr_ir1, vr_ir2, vr_ir3):
    #naprijed
    if vr_ir1 == 0 and vr_ir2 == 0 and vr_ir3 == 0:
        GPIO.output(M1, GPIO.HIGH)
        GPIO.output(M2, GPIO.LOW)
        GPIO.output(M3, GPIO.LOW)
        GPIO.output(M4, GPIO.HIGH)
        ENa_pwm.ChangeDutyCycle(35)
        ENb_pwm.ChangeDutyCycle(35)
```

```
elif vr_ir1 == 0 and vr_ir2 == 1 and vr_ir3 == 1:
        GPIO.output(M1, GPIO.LOW)
        GPIO.output(M2, GPIO.HIGH)
        GPIO.output(M3, GPIO.LOW)
        GPIO.output(M4, GPIO.HIGH)
        ENa_pwm.ChangeDutyCycle(40)
        ENb_pwm.ChangeDutyCycle(35)

    elif vr_ir1 == 0 and vr_ir2 == 0 and vr_ir3 == 1:
        GPIO.output(M1, GPIO.LOW)
        GPIO.output(M2, GPIO.HIGH)
        GPIO.output(M3, GPIO.LOW)
        GPIO.output(M4, GPIO.HIGH)
        ENa_pwm.ChangeDutyCycle(35)
        ENb_pwm.ChangeDutyCycle(35)

    elif vr_ir1 == 1 and vr_ir2 == 1 and vr_ir3 == 0:
        print("ide jako lijevo")
        GPIO.output(M1, GPIO.HIGH)
        GPIO.output(M2, GPIO.LOW)
        GPIO.output(M3, GPIO.HIGH)
        GPIO.output(M4, GPIO.LOW)
        ENa_pwm.ChangeDutyCycle(35)
        ENb_pwm.ChangeDutyCycle(40)

    elif vr_ir1 == 1 and vr_ir2 == 0 and vr_ir3 == 0:
        GPIO.output(M1, GPIO.HIGH)
        GPIO.output(M2, GPIO.LOW)
        GPIO.output(M3, GPIO.HIGH)
        GPIO.output(M4, GPIO.LOW)
        ENa_pwm.ChangeDutyCycle(35)
        ENb_pwm.ChangeDutyCycle(35)

    elif vr_ir1 == 1 and vr_ir2 == 0 and vr_ir3 == 1:
        GPIO.output(M1, GPIO.HIGH)
        GPIO.output(M2, GPIO.LOW)
        GPIO.output(M3, GPIO.LOW)
        GPIO.output(M4, GPIO.HIGH)
        ENa_pwm.ChangeDutyCycle(35)
        ENb_pwm.ChangeDutyCycle(35)

    elif vr_ir1 == 0 and vr_ir2 == 1 and vr_ir3 == 0:
        GPIO.output(M1, GPIO.LOW)
        GPIO.output(M2, GPIO.LOW)
        GPIO.output(M3, GPIO.LOW)
        GPIO.output(M4, GPIO.LOW)

sleep(1)
```

```
while True:
    vr_ir1 = GPIO.input(ir1)
    vr_ir2 = GPIO.input(ir2)
    vr_ir3 = GPIO.input(ir3)
    List = []

    while len(List)<2 :
        List.append(ser.read())

    #Prvi slucaj, ne treba zaljevati nista
    if List[0]==b'\x13' and List[1]==b'\x15':
        stopMoving()
    #Drugi slucaj, treba zaliti prvu saksiju, drugu ne treba
    elif List[0]==b'\x14'and List[1]==b'\x15':
        if ser.read()==b'\x06':
            followLine(vr_ir1,vr_ir2,vr_ir3)
        elif ser.read()==b'\x05':
            #ZALJEVANJE
            stopMoving()
            time.sleep(4)
            GPIO.output(pumpa, GPIO.LOW)
            time.sleep(4)
            GPIO.output(pumpa, GPIO.HIGH)
            time.sleep()
            followLine(vr_ir1,vr_ir2,vr_ir3)
            time.sleep(2)
            stopMoving()
            List.clear()
        else:
            followLine(vr_ir1,vr_ir2,vr_ir3)

    #Treci slucaj, treba zaliti drugu saksiju, prvu ne treba
    elif List[0]==b'\x13'and List[1]==b'\x16':
        if ser.read()==b'\x05':
            followLine(vr_ir1,vr_ir2,vr_ir3)
        elif ser.read()==b'\x06':
            #ZALJEVANJE
            stopMoving()
            time.sleep(4)
            GPIO.output(pumpa, GPIO.LOW)
            time.sleep(4)
            GPIO.output(pumpa, GPIO.HIGH)
            time.sleep()
            followLine(vr_ir1,vr_ir2,vr_ir3)
            time.sleep(2)
            stopMoving()
            List.clear()
        else:
            followLine(vr_ir1,vr_ir2,vr_ir3)

    #Cetvrti slucaj, treba zaliti obje saksije
    elif List[0]==b'\x14'and List[1]==b'\x16':
        followLine(vr_ir1,vr_ir2,vr_ir3)
        if ser.read()==b'\x06' or ser.read()==b'\x05':
```

```
            stopMoving()
             time.sleep(4)
             GPIO.output(pumpa, GPIO.LOW)
             time.sleep(4)
             GPIO.output(pumpa, GPIO.HIGH)
             time.sleep()
             followLine(vr_ir1,vr_ir2,vr_ir3)
             time.sleep(2)
             stopMoving()
             List.clear()
     else:
             followLine(vr_ir1,vr_ir2,vr_ir3)
```

6.3. Arduino kod za ESP8266 prijemnik i RFID čitač

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <MFRC522.h>

//Narednih pet linija koda služi za deklaraciju pinova RFID čitača i
aktiviranje klase tj. funkcija neophodnih za registrovanje
taga/kartice
constexpr uint8_t RST_PIN = 5
constexpr uint8_t SS_PIN = 4;
MFRC522 rfid(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;
byte nuidPICC[4];

const char* ssid ="NAZIV_WIFI";
const char* Password ="SIFRA_WIFI";
const char *mqttServer="IP_ADRESA";

int payl;
int number;
char *topic="PSCGr5/garden/2023";
uint8 q=1;
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);
int mqttPort = 1883;  //Port putem kojeg prolaze podatci
int message = 0;

void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  connectToWiFi();
  setupMQTT();
}
```

```
void loop() {
//Beskonačna petlja u kojoj se konektuje ESP na MQTT broker i
očitava+upisuje vrijednost RFID kartice
  if (!mqttClient.connected())
    reconnect();
  mqttClient.loop();

  if (!rfid.PICC_IsNewCardPresent())
    return;

  if (rfid.PICC_ReadCardSerial()) {

    String rfidCard = toString(rfid.uid.uidByte, rfid.uid.size);
    checkCard(rfidCard);
    Serial.write(number);
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
  }
}
//Funkcija za konverziju adrese kartice u string
String toString(byte* buffer, byte bufferSize) {
  String rfidCard;
  for (int i = 0; i < bufferSize; i++) {
    rfidCard += String(buffer[i]);
  }
  return rfidCard;
}
//Funkcija za dodjeljivanje decimalnog broja RFID karticama
void checkCard(String rfidCard) {
  if (rfidCard == "2301257426") {
    number = 5;
  }
  if (rfidCard == "2262102827") {
    number = 6;
  }
  if (rfidCard == "2261363927") {
    number = 7;
  }
}
//Konektovanje na WiFI
void connectToWiFi() {
  Serial.print("Connecting to ");
  WiFi.begin(ssid, Password);
  Serial.println(ssid);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.print("Connected.");
}

void setupMQTT() {
  mqttClient.setServer(mqttServer, mqttPort);
  // set the callback function
  mqttClient.setCallback(callback);
```

```
void reconnect() {
  Serial.println("Connecting to MQTT Broker...");
  while (!mqttClient.connected()) {
    Serial.println("Reconnecting to MQTT Broker..");
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);

    if (mqttClient.connect(clientId.c_str())) {
      Serial.println("Connected.");
      // subscribe to topic
      mqttClient.subscribe(topic, q);
    }
  }
}
//Konverzija stringa u int
int atoi(const String str) {
  int num = 0;
  int i = 0;
  bool isNegetive = false;
  if (str[i] == '-') {
    isNegetive = true;
    i++;
  }
  while (str[i] && (str[i] >= '0' && str[i] <= '9')) {
    num = num * 10 + (str[i] - '0');
    i++;
  }
  if (isNegetive) num = -1 * num;
  return num;
}
//Callback funkcija koja se kontinualno izvršava i upisuje podatak
koji dobije od udaljenih mikrokontrolera (saksija)
void callback(char* topic, byte* payload, unsigned int length) {
  String pay;
  for (int i = 0; i < length; i++) {
    pay += (char)payload[i];
  }
  int payl = atoi(pay);
  Serial.write(payl);
}
```

7. Reference i izvori

[1] Projektovanje sistema na čipu – Predavanja, ak. 2022/2023, dr.sc. Lejla Banjanović-Mehmedović
[2] Cyclone II datasheet (https://cdrdv2-public.intel.com/654376/cyc2_cii5v1.pdf)
[3] https://en.wikipedia.org/wiki/Raspberry_Pi
[4] https://en.wikipedia.org/wiki/ESP8266
[5] https://randomnerdtutorials.com/guide-for-soil-moisture-sensor-yl-69-or-hl-69-with-the-arduino/
[6] https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/
[7] https://hr.wikipedia.org/wiki/RFID
[8] https://soldered.com/hr/learn/kkm-jednostavni-h-bridge/
[9] https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/
[10] https://www.14core.com/wiring-driving-the-l298n-h-bridge-on-2-to-4-dc-motors/
[11] https://www.electronicwings.com/raspberry-pi/raspberry-pi-uart-communication-using-python-and-c
[12] http://www.steves-internet-guide.com/send-and-receive-integers-and-floats-with-arduino-over-mqtt/
[13] https://tiagohorta1995.medium.com/mosquitto-mqtt-broker-and-esp8266-communcation-in-6-easy-steps-5ea0efae5a10