SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

BEŽIČNO UPRAVLJANI ROBOT TEMELJEN NA ARDUINO PLATFORMI

IVAN SIČAJA

Split, rujan 2019.

SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Preddiplomski sveučilišni studij: **Elektrotehnika i informacijska tehnologija**

Smjer/Usmjerenje: **Automatika i sustavi**

Oznaka programa: 111

Akademska godina: 2018./2019.

Ime i prezime: **IVAN SIČAJA**

Broj indeksa: 602-2016

ZADATAK ZAVRŠNOG RADA

Naslov: **BEŽIČNO UPRAVLJANI ROBOT TEMELJEN NA ARDUINO PLATFORMI**

Zadatak: U završnom radu potrebno je opisati mogućnosti Arduino prototipne platforme u primjenama upravljanja mobilnih robota. U praktičnom dijelu rada izraditi vozilo i mobilnu aplikaciju za Arduino kojom će se upravljati vozilom. Omogućiti upravljanje preko bežične mreže. Opisati korištene senzore, komunikaciju te korištene algoritme. Prikazati i komentirati dobivene rezultate.

Datum obrane:

25.09.2019.

Mentor:

Prof. dr. sc. Vladan Papić

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom BEŽIČNO UPRAVLJANI ROBOT TEMELJEN NA ARDUINO PLATFORMI, pod mentorstvom prof. dr. sc. VLADANA PAPIĆA pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo/la u završnom radu citirao/la sam i povezao/la s korištenim bibliografskim jedinicama.

Student:

Ivan Sičaja

SADRŽAJ

1. UVOD 1
2. GRAĐA MIKROKONTROLERA ARDINO UNO 3
   2.1. Podjela i funkcije pinova mikrokontrolera Arduino Uno 3
      2.1.1. Analogni pinovi 3
      2.1.2. Digitalni pinovi 4
      2.1.3. Pinovi za napajanje 4
      2.1.4. Pinovi za ICSP komunikaciju 5
   2.2. Ostale komponente mikrokontrolera Arduino Uno 6
      2.2.1.1. Mikrokontroler ATmega128 7
      2.2.2. Komponente za vanjsko napajanje 8
      2.2.3. Restart tipkalo 9
      2.2.4. Naponski regulator 9
      2.2.5. Kristalni oscilator 9
      2.2.6. Čip za serijsku komunikaciju (UART most) 9
      2.2.7. USB tipa B 10
      2.2.8. Ledice 10
      2.2.9. „Pull-up“ i „pull-down“ otpornici 11
3. SIGNALI I PROTOKOLI 12
   3.1. PWM signal 12
   3.2. UART komunikacijski protokol 13
   3.3. ICSP komunikacijski protokol 14
   3.4. I2C komunikacijski protokol 16
   3.5. Ukratko o ADC i DAC pretvaraču 18
4. ARDUINO WI-FI ROBOT S PROTOKOLOM ZA ZAOBILAŽENJE PREPREKA 21
   4.1. Popis i objašnjenja komponenti 21
      4.1.1. Servo motor SG90 22
      4.1.2. Ultrazvučni senzor HC-SR04 24
      4.1.3. Modul ESP-32 26
      4.1.4. USB (UART) most 31
      4.1.5. LED indikatori 31
      4.1.6. UBS priključak tipa mikro A (eng. Universal Serial Bus) 31

4.1.7. „EN“ i „Boot“ tipkalo 32
4.1.8. Mikrokontroler L298N 32
4.1.9. Troznamenkasti 7-segmentni LED ekran 36
4.1.10. Baterije 38
4.2. Spajanje komponenti 42
4.3. Programsko okruženje 42
4.4. Osnovne naredbe u Arduino programskom jeziku 43
4.5. Optimizacija programskog koda 44
4.5.1. Uklanjanje mrtvih dijelova koda 44
4.5.2. Odabiranje adekvatnog tipa varijable 45
4.5.3. Upotreba lokalnih varijabli 46
4.5.4. Upotreba programske memorije za pohranu podataka 46
4.5.5. Izbjegavanje korištenja rekurzivnih funkcija 47
4.6. Izrada Android aplikacije za upravljanje robotom 47
4.6.1. Izrada programskog sučelja 47
4.6.2. Blokovsko programiranje korisničkog sučelja 48
4.7. Test 49
5. ZAKLJUČAK 52
SAŽETAK 54
KLJUČNE RIJEČI 54
TITLE 55
SUMMARY 55
KEY WORDS 55
POPIS LITERATURE 56
DODATAK A 58

1. UVOD

Na samom početku trebamo razjasniti što je to Arduino. Arduino je „open-source“ elektronička platforma bazirana na softveru i hardveru koji je vrlo jednostavan za korištenje, a uz to nudi mogućnosti realizacije gotovo svakog projekta kojeg možete zamisliti po vrlo pristupačnim cijenama, što ga razlikuje od većine ostalih elektroničkih platformi pomoću kojih vam za realizaciju najobičnijeg projekta treba cijelo bogatstvo (npr. PLC platforma). Ideja o realizaciji ovakve elektroničke platforme rođena je na „Interaction Design Institute Ivrea“ u gradu Ivera u sjevernoj Italiji, a vrlo brzo je pronašla svoju primjenu u raznim rekreativnim industrijskim i znanstvenih područjima, a tomu svjedoče brojni primjeri projekta kojih su izrađeni od strane pojedinaca, preko studenata koji za svoje završne radove odabiru konkretne projekte iz područja mehatronike koji se pritom pomoću Arduino elektroničke platforme mogu realizirati na vrlo jednostavan i učinkovit način te do profesora i znanstvenika koji u prezentiranju svojih predmeta ili istraživanja koriste upravo ovu doista moćnu platformu. Jedna od važnijih karakteristika je ta da je Arduino elektronička platforma podržana od strane najpopularnijih operativnih sustava kao što su MAC, Windows i Linux. Također većina programa za izrađivanje elektrotehničkih shema za spajanje Arduino elektroničke pločice s određenim komponentama nudi gotove grafičke interpretacije Arduino komponenti tako da ako želite napraviti određenu elektroničku shemu spajanja pinova određenih komponenti ne morate gubiti vrijeme na dizajniranje komponenti jer one kao takve već postoje i samo ih treba ukomponirati u vašu shemu. Za odgovor na pitanje da li ste sve spojili na adekvatan način tu su Arduino simulatori koji su najčešće ukomponirani u programe za izradu električnih shema te je potrebno samo ubaciti vaš Arduino programski kod te pokrenuti simulaciju, ukoliko vam simulator ne javlja grešku, a uz to se simulacija projekta odvija na upravo onaj način na koji ste vi priželjkivali vaše komponente su povezane na pravilan način te možete početi s fizičkom realizacijom vašeg projekta bez straha da će te neku komponentu strujno uništiti. Uz različite tipove Arduino mikrokontrolera kao što su Arduino Uno, Arduino Mega, Arduino Nano i ESP-32 i brojne druge mikrokontrolere, tu se nalazi i ogroman broj popratnih senzora i uređaja izrađenih i kompatibilnih upravo za Arduino elektroničku platformu pomoću kojih možete realizirati gotovo svaki projekt koji zamislite. Još jedna u nizu prednosti Arduino platforme je ta da Arduino programski jezik predstavlja blagu modifikaciju jednog od najpopularnijih programskih jezika današnjice, a to je programski jezik C++, koji je uz C programski jezik predmet izučavanja velikog broja srednjih škola i fakulteta diljem svijeta koji prema svojoj strukturi imaju određenih dodirnih točaka s programiranjem kao takvim, što dodatno olakšava već vrlo olakšano korištenje Arduino elektroničke platforme. Arduino programsko okruženje moguće je preuzeti sa službene Arduino stranice, a uz samo programsko okruženje dolazi i paket s primjerima kodova i biblioteka koji vam vrlo detaljno pružaju uvid u samo funkcioniranje programskog koda i što je najvažnije cijelo programsko okruženje je potpuno besplatno te na svim pretraživačima i internet servisima postoji ogroman broj realiziranih detaljno objašnjenih projekta s popratnim kodovima i shemama spajanja što ovu platformu čini vrlo logičnim prvim odabirom za jako veliki broj rekreativnih znanstvenih i industrijskih projekta.

Poglavlja kao i kratka objašnjenja tema koje se obrađuju u datom poglavlju dana su u nastavku;

Građa mikrokontrolera Arduino Uno-detaljan opis građe i funkcije svake prisutne komponente na ovom mikrokontroleru.

Signali i protokoli- detaljan opis signala i protokola koje ćemo koristiti u ovom projektu

Arduino Wi-Fi robot s protokolom za autonomno zaobilaženje prepreka- uz opis svih komponenti koje ćemo koristiti, u ovom poglavlju su objašnjeni procesi električnog spajanja svih komponenti, odabira programskog okruženja, izrade mobilne aplikacije za upravljanje robotom, test te i sam programski kod koji smo primijenili na robotu.

Zaključak-najbitnija opažanja koja smo primijetili tokom izrade ovog robota

Popis literature-navest ćemo izvore koje smo koristili pri izradi ovog projekta

2. GRAĐA MIKROKONTROLERA ARDINO UNO

2.1. Podjela i funkcije pinova mikrokontrolera Arduino Uno

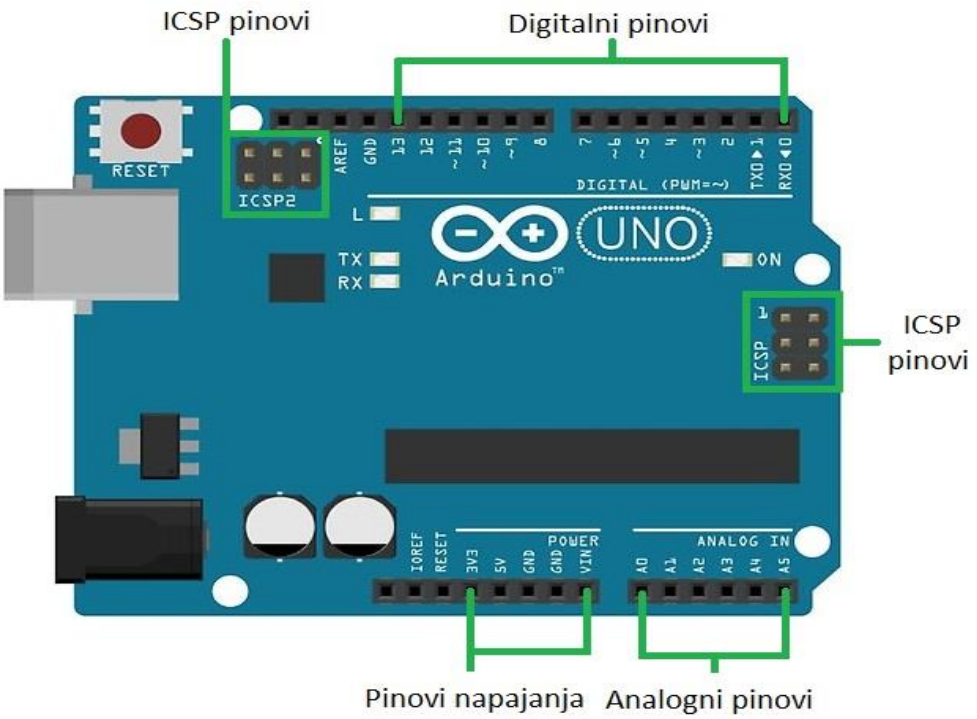
Budući da je Arduino Uno jedan od najčešće korištenih Arduino mikrokontrolera u nastavku ćemo detaljno objasniti funkciju pinova i komponenti koji su integrirani u ovaj mikrokontroler. Pinove na mikrokontroleru Arduino Uno dijelimo u četiri skupine kao što je vidljivo i sa slike 2.1, a to su;

-analogni pinovi

-digitalni pinovi

-pinovi napajanja

-ICSP pinovi



Slika 2.1. Arduino Uno podjela pinova

2.1.1. Analogni pinovi

Analogni pinovi mikrokontrolera Arduino Uno su pinovi koji služe za pisanje i očitavanje analognih vrijednosti određenih komponenti kao što su potenciometri odnosno promjenjivi otpornici kojima najčešće zakretanjem određene osovine mijenjamo vrijednost otpora uz činjenicu da svaki analogni pin daje konstantnu vrijednost struje prema Ohmovom zakonu, gdje automatski mijenjamo iznos izlaznog napona potenciometra koji kao takav bez pretvorbe

u digitalni oblik, odnosno analogan ulazi u analogne pinove te nam pomoću programskog koda omogućuje mjerenje, očitavanje i upisivanje određenih vrijednosti na same pinove. Jednostavnije rečeno zakretanjem osovine otpornika dolazi do postupnog mijenjanja vrijednosti izlaznog napona potenciometra koju možemo pratiti u našem Arduino programskom okruženju. Ostali analogni uređaji koji koriste analogne vrijednosti su foto-otpornici, senzori provjere kvalitete zraka i sl. Analogni pinovi se kao što je prikazano na slici se nalaze u donjem desnom kutu slike 2.1, označavaju se oznakama A0 A1 A2 A3 A4 A5. Rezolucija analognih pinova je 10-bitna što znači da imaju sposobnost razlikovanja 2^{10} različitih analognih vrijednosti signala. Pinovi A4 i A5 posjeduju u mogućnosti za korištenje I2C komunikacijskog protokola koji je objašnjen u poglavlju; „I2C komunikacijski protokol“.

2.1.2. Digitalni pinovi

Za razliku od analognih pinova koji primaju analogni signal, digitalni pinovi primaju digitalni signal. Razlika između analognog i digitalnog signala je ta da je analogni signal, signal čija vrijednost predstavlja stvarnu vrijednost dok je vrijednost digitalnog signala aproksimativna vrijednost analognog signala te će uvijek postojati određena odstupanja u digitalnom i analognu zapisu vrijednosti, iako se radi zapravo o istom signalu. Data odstupanja su prouzročena zbog konačnosti sklopova za analogno-digitalnu pretvorbu. Digitalni signal je uveden zbog određenih prednosti koje posjeduje u odnosu na analogni signal, a to su prije svega brzina, lakoća i preciznost pri očitavanju i proračunu potrebnih vrijednosti. U odnosu na pet analognih pinova mikrokontrolera Arduino Uno, postoji 14 digitalnih pinova koji se označavaju brojevima od 0 do 13. Funkcija je vrlo slična kao i funkcija analognih pinova, a to je upisivanje i očitavanje vrijednosti pinova pomoću Arduino programskog okruženja što se fizički manifestira na samim pinovima u obliku promjene napona što uzrokuje daljnje promjene na komponentama koje ovise o načinu izvedbe samih komponenti, npr. ako se radi o servo motoru doći će do njegovog zakretanja za određeni broj stupnjeva, koji naravno mi definiramo, ako se radi o zvučniku nastupit će promjena frekvencije zvuka koja je u srži uzrokovana promjenom napona određenog pina. Komponente koje se obično povezuju na digitalne pinove su servo motori, zvučnici, različiti senzori (npr. ultrazvučni senzor) i sl. Rezolucija digitalnih pinova na ovom mikrokontroleru je 8-bita što znači da imaju sposobnost razlikovanja 2^8 različitih vrijednosti signala. Kao i analogni pinovi određeni digitalni pinovi posjeduju mogućnost korištenja SPI komunikacijskog protokola te korištenje PWM signala što je posebnost u odnosu na analogne pinove Arduino Uno mikrokontrolera koji nemaju ove mogućnosti. Spomenuti protokol je detaljno objašnjen u poglavlju; „ICSP komunikacijski protokol“, dok je objašnjenje PWM signala dano u poglavlju; „PWM signal“.

2.1.3. Pinovi za napajanje

Pinovi napajanja se nalaze u srednjem donjem dijelu Arduino Uno mikrokontrolera prikazanog na slici 2.1, označeni su oznakama redom idući s lijeva na desno; 3,3 V, 5 V, GND, GND te Vin. U nastavku ćemo objasniti funkciju svakog pojedinog pina.

Pin 3,3 V omogućuje napajanje određenog vanjskog elementa konstantnim istosmjernim naponom iznosa 3,3 V te maksimalnom strujom od 40 mA.

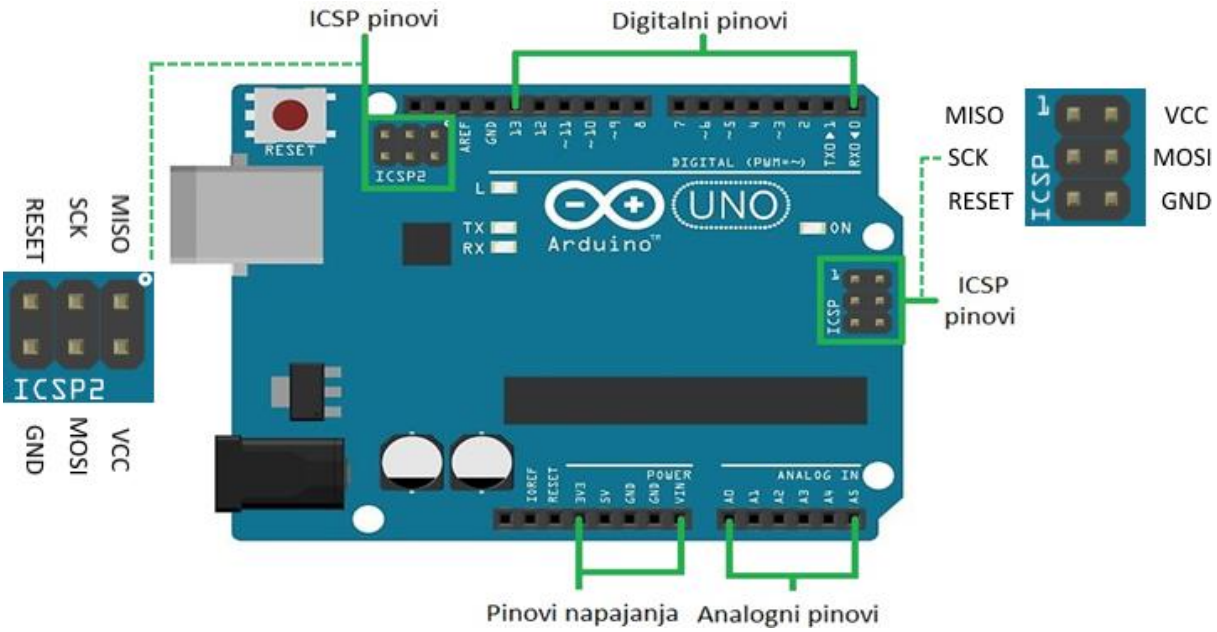
Pin 5 V omogućuje napajanje određenog vanjskog elementa konstantnim istosmjernim naponom iznosa 5 V te maksimalnom strujom od 40 mA.

GND pinovi su pinovi koji predstavljaju neutralni vodič, funkcija im je da prime određenu struju te na taj način zatvore električni strujni krug. Bez ovih pinova ne bi bilo moguće napajati vanjski element s Arduino mikrokontrolera jer strujni krug ne bi bio zatvoren.

Vin pin je pin koji ima dvije funkcije. Prva je ta da daje napon od 5 V koji možemo iskoristiti za napajanje određene vanjske komponente, dok je druga funkcija napajanje Arduino mikrokontrolera naponima u rasponu od 7 V do 12 V pri tome je maksimalna struja koju ovaj pin prima iznosi 0,5 A.

2.1.4. Pinovi za ICSP komunikaciju

Osim prethodne tri skupine pinova koje smo nabrojali postoje i dvije skupine od po 6 pinova koje se nalaze izdvojeno od ostalih na Arduino Uno mikrokontroleru, a to su pinovi za ICSP serijski komunikacijski protokol. Svaka od dvije skupine pinova sastoji se od pinova označenih istim oznakama, a to su oznake; MISO, MOSI, SCK, Vcc, GND te RESET, raspoređeni su na način kako je prikazano na slici 2.2.



Slika 2.2. Oznake ICSP pinova

Značenja ICSP pinova su sljedeća;

MISO-(eng. Master In Slave Out)

MOSI-(eng. Master Out Slave In)

SCK-(eng. Serial Clock)

RESET-(eng. Reset)

Vcc-napajanje

GND-neutralni vodič

Funkcije pinova objašnjene su u potpoglavlju; „ICSP komunikacijski protokol“.

2.2. Ostale komponente mikrokontrolera Arduino Uno

Iz slike 2.3 vidljivo je da su osim pinova koje smo prethodno nabrojali na mikrokontroleru prisutne i još neke komponente koje ćemo pobrojati u nastavku. To su redom;

- mikrokontroler ATmega328

-komponente za vanjsko napajanje

- restart tipkalo

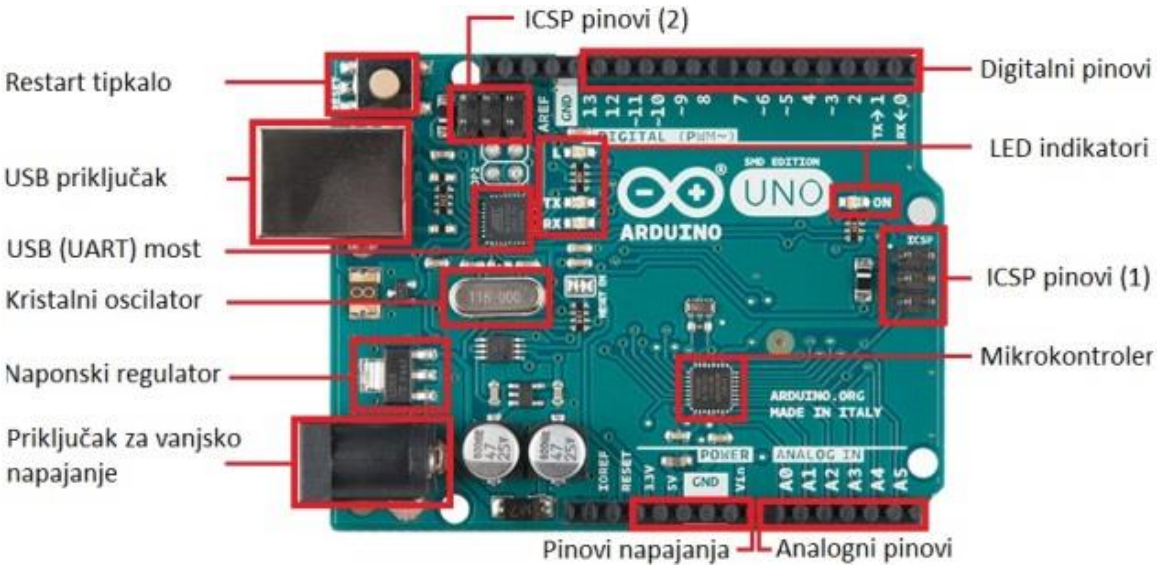
- naponski regulator

-kristalni oscilator

- čip za serijsku komunikaciju (UART most)

- USB tipa B (eng. Universal Serial Bus)

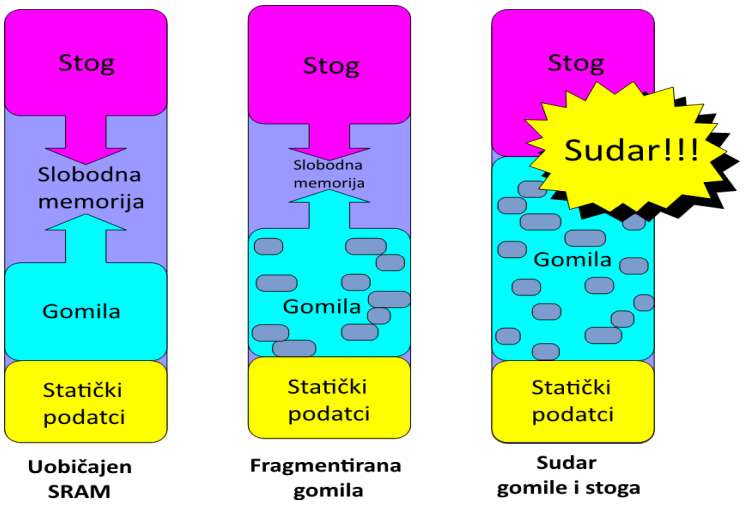
- LED (eng. Light Emitting Diode)



Slika 2.3. Ostale komponente Arduino Uno mikrokontrolera

2.2.1.1. Mikrokontroler ATmega128

Najvažniji dio Arduino pločice je svakako mikrokontroler ATmega328 koji se osim na ovoj ploči koristi i na Arduino Uno, Arduino Internet, Arduino Menta i mnogim drugim pločicama , a raspolaže s 32 KB programske 2 KB SRAM i 1 KB EEPROM memorije što je oko 100.000 puta manje od memorije prosječnog računala. Struktura mikrokontrolera je 8-bitna što znači da memorijske adrese mogu poprimiti 2^8 različitih vrijednosti. Programska memorija predstavlja količinu memorije koju posjeduje određeni mikrokontroler čija je svrha primanje određene količine programskog koda. Npr. ako kod koji želimo primijeniti na mikrokontroleru zauzima više od 32 KB programske memorije, prijenos koda na mikrokontroler neće biti moguć bez adekvatne redukcije koda. Druga i ne manje važna memorija mikrokontrolera ATmega328 je SRAM memorija (eng. Static Random Access Memory). Uloga ovog dijela memorije je ta da se u njega mogu čitati pisati određeni podaci prilikom izvršavanja programa. SRAM memorija se sastoji od četiri dijela na način koji je prikazan na slici 2.4.



Slika 2.4. Prikaz SRAM memorije [1]

Stog (eng. Stack) je dio SRAM memorije koji se puni od vrha prema dolje prema principu LIFO (eng. Last In First Out) što se upotrebljava za pohranu lokalnih varijabli, održavanje evidencije prekida te za pozivanje funkcija. Svaki prekid (eng. Interrupt) lokalna varijabla ili poziv funkcije uzrokuje porast stoga. Kada se funkcija izvrši dođe do povratka iz prekida, memorija koju su zauzimali se oslobodi.

Slobodna memorija (eng. Free Memory) je memorija koja se nalazi između stoga i gomile (eng. Heap) na čiji se račun omogućuje povećanje memorije stoga i gomile. Kada se stog i gomila prošire u tolikoj mjeri da premaše vrijednost slobodne memorije, to jest slobodna memorija više ne postoji, te kao takva ne predstavlja granicu između stoga i gomile, dolazi do preklapanja vrijednosti memorija gomile i stoga odnosno dolazi do miješanja vrijednosti adresnih lokacija stoga odnosno gomile te program više nije funkcionalan i automatski dolazi do restarta Arduino ATmega328 mikrokontrolera, programski kod se počinje izvršavati iz početka. Različiti mikrokontroleri se ponašaju različito u situacijama kada dođe do preklapanja pojedinih dijelova memorije. Sam način ponašanja definira sam proizvođač.

Gomila (eng. Hip) je dio SRAM memorije koji se puni od dna prema vrhu. Funkcija mu je realiziranje naredbi za dinamičko alociranje odnosno dealociranje memorije.

Statički podaci su dio SRAM memorije koji služi za pohranu globalnih varijabli, konstanti iz programa koji je pohranjen u programskoj (eng. flash) memoriji. Postoje posebne makro naredbe kojima se omogućuje prebacivanje memorije statičkih podataka u programsku memoriju te na taj način nam je omogućena ušteda SRAM memorije. O samom postupku bit će više govora u poglavlju; „Optimizacija programskog koda“.

2.2.2. Komponente za vanjsko napajanje

Komponente za vanjsko napajanje omogućuju napajanje Arduino Uno mikrokontrolera istosmjernim naponima u rasponu od 7 V do 12 V pomoću priključka standardnog promjera 2,1 mm. Ulazna struja napajanja pomoću ovog priključka mora biti minimalno 250 mA, a za energetski zahtjevnije potrebe ulazna struja može ići do iznosa 1 A.

2.2.3. Restart tipkalo

Restart tipkalo je komponenta koja se nalazi u gornjem lijevom dijelu slike 2.3, a funkcija mu je da se pritiskom na tipkalo, zadaća koja je definirana na mikrokontroleru nevažno u kojem se trenutku nalazi ,počne izvršavati iz početka.

2.2.4. Naponski regulator

Funkcija naponskog regulatora na pločici Arduino Uno je ta da regulira odnosno stabilizira napon na točno određenu vrijednost istosmjernog napona koja je potrebna za pravilno funkcioniranje mikroprocesora ATmega328 i svih ostalih komponenti iz razloga što mikrokontroler za svoj rad zahtjeva vrlo stabilno napajanje što mu omogućava naponski regulator na način da smanjuje naponski oscilacije te samim tim indirektno smanjuje samu potrošnju električne energije.

2.2.5. Kristalni oscilator

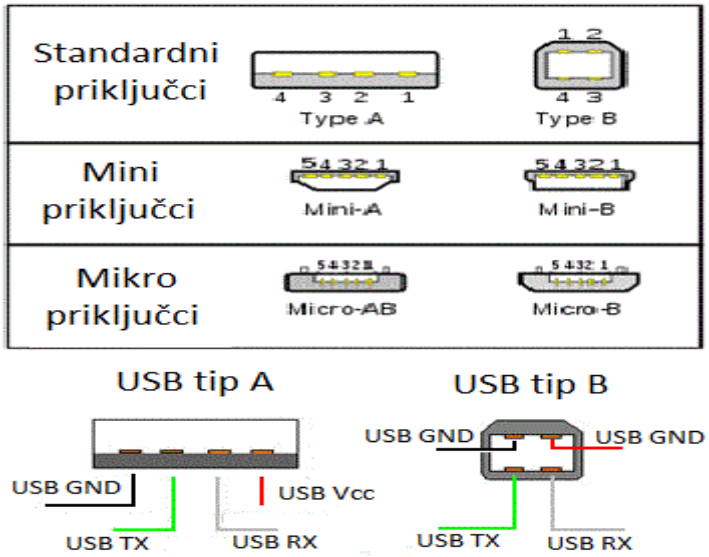
Kristalni oscilator predstavlja titrajni izvor frekvencije 16 MHz koji služi kao izvor taktnog signala čija se frekvencija može prilagođavati ovisno o našim potrebama. Prilagođavanje frekvencije koja nam je potrebna vrši se u određenim registrima, shodno tomu dobivene frekvencije mogu biti samo frekvencije dobivene dijeljenjem osnovne frekvencije djeliteljem većim od 1. Također ovo je komponenta bez koje bi rad mikrokontrolera bio nemoguć jer sam mikrokontroler ne bih imao nikakvu predodžbu o vremenu, što bi za sobom povuklo vrlo ozbiljne posljedice, to jest ne bi znali da li neki proces kasni i koliko kasni, da li je to kašnjenje beznačajno ili ne, što bi izazvalo ogromnu pomutnju u radu mikroprocesora koji sam po sebi zahtjeva vrlo stroge i precizne naredbe kako bi obradio jako veliki broj informacija u što kraćem vremenskom periodu što je odlika suvremenih mikroprocesora.

2.2.6. Čip za serijsku komunikaciju (UART most)

Kao što i sam naziv ove komponente kaže, čip za serijsku komunikaciju omogućava komuniciranje između Arduino mikrokontrolera i uređaja na kojem se nalazi Arduino programsko okruženje, najčešće je to laptop, PC ili pametni mobitel. Serijska komunikacija predstavlja razmjenu informacija putem USB priključka tipa B koje se u realiziranom projektu pojavljuju na mikrokontroleru ATmega328, a služi kao provjera stanja procesa uređaja s kojim smo serijski povezani (USB kabelom tipa B) ne određenom ekranu, prije priključivanja samog uređaja ili projekta na eksterni izvori energije.

2.2.7. USB tipa B

USB tipa B (eng. Universal Serial Bus) koji se nalazi na Arduino Uno pločici služi za fizičko ostvarivanje serijske komunikacije između mikroprocesora i računala ili drugog pametnog uređaja. Ako uđemo malo dublje u samo strukturu građe USB priključka tipa B vidjet ćemo da među žicama (kanalima) koje se granaju iz glavnog kabela, osim žica odnosno kanala za napajanje koji su označeni oznakama 5 V odnosno crvenom bojom i neutralnog vodiča označenog oznakom GND odnosno crnom bojom postoje i dvije žice koje služe za podatkovni prijenos, a označene su oznakama $DATA^+$ odnosno $DATA^-$ pri čemu je žica $DATA^+$ najčešće označena zelenom bojom dok je žica oznake $DATA^-$ najčešće označena bijelom bojom. Ove dvije podatkovne žice zapravo predstavljaju dva komunikacijska kanala gdje se jednim kanalom šalju podaci s mikrokontrolera, dosta često označeno oznakom TX skraćeno od (eng. Transmitter), prema računalu koje taj kanal tretira kao kanal kojim prima određene informacije, dosta česta oznaka je oznaka kanala kojim primamo informacije je RX koja potječe od skraćenog oblika (eng. Receiver). Nakon što CPU (eng. Central Procesor Unit) računala obradi datu informaciju drugim komunikacijskim kanalom se odvija gotovo identičan proces kao u prethodno opisanom procesu s tim da CPU računala i mikroprocesor ATmega328 zamjene uloge odnosno računalo šalje obrađenu informaciju Arduino Uno pločici te se ponaša kao uređaj odašiljač informacije TX, dok se Arduino Uno pločica ponaša kao uređaj prijemnik informacije RX. Unutarnja građa standardnih priključaka tipa A i tipa B te primjeri nekih od standardnih priključaka prikazani su na slici 2.5.



Slika 2.5. Unutarnja građa standardnih priključaka tipa A i tipa B te primjeri nekih od standardnih USB priključaka. [2]

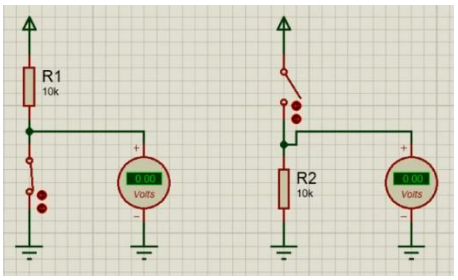
2.2.8. Ledice

Na Arduino Uno pločici postoje četiri ledice odnosno LED (eng. Light Emitting Diode) s oznakama; ON, TX, RX i L. Ledica označena oznakom „ON“ signalizira da je na ulaz

Arduino pločice doveden zadovoljavajući iznos napona pomoću kojeg se mogu izvršavati različite operacije unutar samog mikrokontrolera. Ledice označene oznakama TX i RX se aktiviraju kada imamo proces primanja odnosno slanja određenih informacija između Arduino pločice i uređaja odašiljača odnosno prijemnika signala. Najjednostavniji primjer aktiviranja ovih dioda je aktiviranje prilikom unošenja (eng. Upload) programskog koda s računala na Arduino Uno mikrokontroler. Kada je prisutno obostrano slanje i primanje informacija između računala i Arduino Uno pločice. Ledica označena oznakom „L“ predstavlja diodu koja je direktno spojena s digitalnim pinom označenim brojem 13, te preko tog pina omogućava kontroliranje odnosno njeno uključivanje i isključivanje u određenim vremenskim intervalima koje definiramo u programskom kodu kojeg upisujemo u sam mikrokontroler ATmega328. Također važno je napomenuti da su neki od prvih primjera programiranja Arduino pločica upravo primjeri kontroliranja ledica što ovoj diodi daje dodatan značaj.

2.2.9. „Pull-up“ i „pull-down“ otpornici

„Pull-up“ i „pull-down“ otpornici su otpornici koji se ugrađuju u strujne krugove kako bi se povećala pouzdanost i očitanje željene veličine zbog smanjivanja utjecaja šuma. Ako se ovaj otpornik nalazi između referentne točke na kojoj vršimo očitanje željene veličine i napajanja tada se ovaj otpornik naziva „pull-up“ otpornik, a ako se otpornik nalazi između referentne točke i neutralnog vodiča tada se on naziva „pull-down“ otpornik. „Pull-up“ i „pull-down“ otpornici su prikazani na slici 2.6.

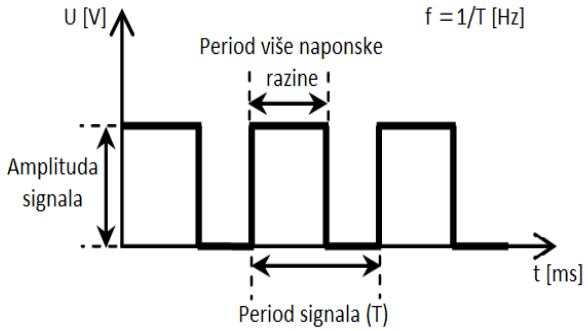


Slika 2.6. Prikaz „pull-up“ (lijevo) i „pull-down“ (desno) otpornika [3]

3. SIGNALI I PROTOKOLI

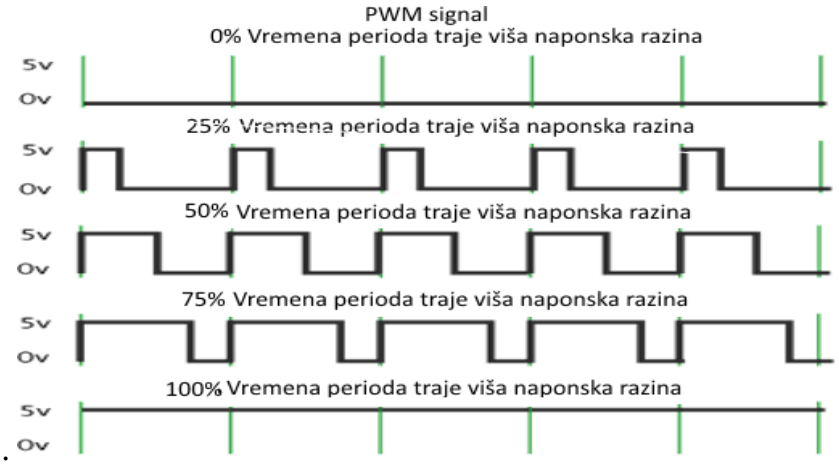
3.1. PWM signal

PWM (eng. Pulse Width Modulation) signal je digitalni signal čija izlazna vrijednost može poprimiti 256 različitih razina. Primjer PWM signala dat je na slici 3.1. Svaka od ovih 256 razina vrijednosti dobivena je s različitim odnosima vrijednosti aktivnog i pasivnog dijela pravokutnog signala. Naime PWM signal posjeduje samo dvije amplitudne vrijednosti koje može poprimiti, jedna vrijednost je napon Vcc koji je uglavnom ustaljen na vrijednosti od 5 V dok je druga razina amplitude napon od 0 V.



Slika 3.1. Karakteristike PWM signala [4]

Unutar jednog perioda signala ove dvije vrijednosti mogu imati različito trajanje te na osnovu srednjih vrijednosti njihovih odnosa unutar jednog perioda pravokutnog signala dobijemo vrijednost jedne od 256 mogućih razina. Kao primjer uzmimo slučaj da vrijednost signala za prvu polovinu perioda pravokutnog signala iznosi 5 V dok vrijednost signala tijekom druge polovice perioda pravokutnog signala iznosi 0 V. Iz ovog primjera vrlo jednostavnom primjenom formule za proračunavanje srednje vrijednosti pravokutnog signala dobijemo vrijednost od 2,5 V što možemo kodirati kao jednu naponsku razinu PWM signala.



Slika 3.2. Primjeri različitih vrijednosti PWM signala [5]

Također ako je signal cijelo vrijeme iznosa 5 V odnosno 0 V tada je srednja vrijednost signala 5 V odnosno 0 V što možemo kodirati kao drugu odnosno treću razinu od mogućih 256 naponskih razina PWM signala što znači da je ostalo slobodnih još 253 različitih odnosa više i niže razine pravokutnog signala u koju možemo pohraniti korisnu informaciju.

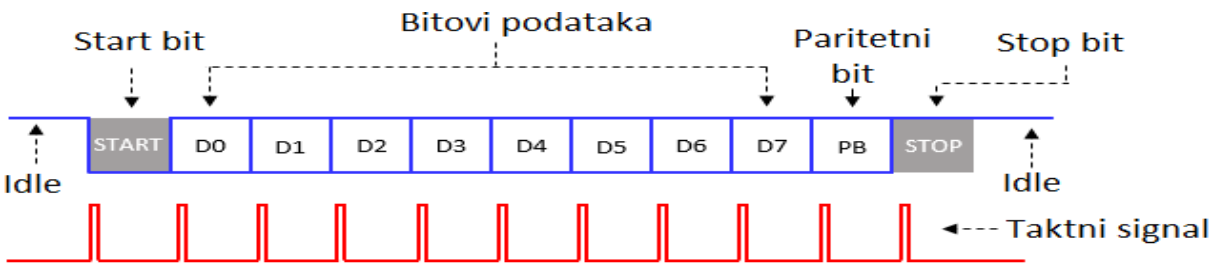
3.2. UART komunikacijski protokol

UART (eng. Universal Asynchronous Receiver and Transmitter) je serijski asinkroni komunikacijski protokol, stoga u svojoj strukturi ne sadrži takni signal. Zbog nedostatka taktnog signala potrebno je postaviti brzinu slanja podataka odašiljača na točno određenu brzinu očitavanje podataka prijemnika (eng. Bit Rates) . Ukoliko je brzina slanja podataka veća od brzine kojom prijemnik interpretira poslane podatke doći će do buke u komunikacijskom kanalu, informacija neće biti interpretirana u željenu obliku. Ista stvar vrijedi i za obratno situaciju, situaciju u kojoj je brzina interpretiranja poslanih informacija veća u odnosu na brzinu slanja informacija od strane odašiljača iz čega zaključujemo da će za slučaj kada se brzine slanja i interpretiranja podataka razlikuju jedna od druge, komunikacija između odašiljača i prijemnika bit će nepotpuna dok za slučaj kada su brzina slanja odnosno primanja podataka između odašiljača odnosno prijemnika usklađene, odnosno istog su iznosa, komunikacija će biti potpuna. Brzine slanja određene količine podataka kroz određeni kanal mjere se u bitovima po sekundi (eng. Bits Per Second), a neke od standardiziranih vrijednosti su prikazane u tablici 3.1.

Tablica 3.1 Standardne vrijednosti brzine serijske komunikacije izražene u bitovima po sekundi [bps]

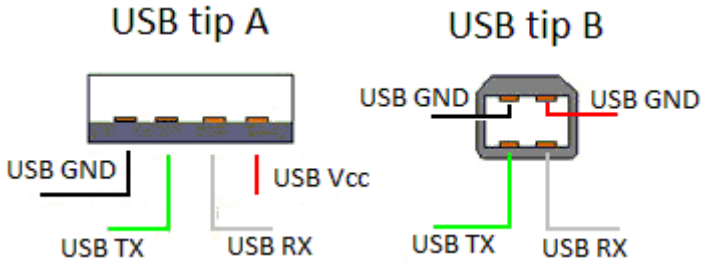
1 200	19 200
4 800	38 400
9 600	115 200

1.200 [bps] odnosno 1 200 bita po sekundi znači da će se određenim kanalom moći slati podatak maksimalne veličine 1 200 bitova svaku sekundu, analogno tomu princip je isti i za sve ostale vrijednosti brzine komuniciranja. UART protokol slanja podataka definiran je strukturom prikazanoj na slici.



Slika 3.3. Struktura UART protokola [6]

Stanje označeno oznakom „Idle“ (eng. Idle) koje je na našem primjeru stanje više razine prisutno prije početka i poslije završetka slanja podataka serijskim putem. Kada se kanal nalazi u stanju „Idle“ kanal je onemogućen za primanje i slanje podataka. Prelaskom iz višeg energetskog u niže energetsko stanje dogovoreno se šalje signal za omogućavanje početka serijske komunikacije označen „START“ bitom (slika 3.3) nakon kojeg slijede podatkovni bitovi čiji broj ovisi o dužini podatka kojeg želimo poslati. Za naš slučaj koristit ćemo 8-bitne podatke. Ako koristimo paritetni bit na slici označen oznakom „PB“ podatak će biti kodiran pomoću 7-bitne kodne riječi dok će paritetni bit služiti za provjeru utjecaja šuma na poslani podatak. Podatak je kodiran pomoću 7-bitne kombinacije logičkih nula i jedinica koje u ASCII tablici (eng. American Standard Code for Information Interchange) predstavljaju određene znakove. Recimo da je podatak kodiran sljedećim binarnim izrazom; „1110000“, odnosno pomoću tri jedinice i četiri nule, ako smo uzeli da paritet podatka bude paran potrebno je paritetni bit postaviti u vrijednost jedan kako bi poslani podatak sadržavao paran broj jedinica, odnosno četiri jedinice. Ako želimo neparni paritet potrebno je postaviti paritetni bit u nulu te će poslani podatak ostati kodiran s neparnim brojem jedinica, odnosno s tri jedinice. Paritet omogućava lakše detektiranje postojanja šuma koji svojim prisustvom uzrokuje promjenu bitova podataka iz 0 u 1 ili iz 1 u 0 što automatski mijenja paritet podatka što se detektira kao utjecaj šuma. Nakon paritetnog ili posljednjeg bita podataka dolazi jedan ili dva stop bita koje prelaskom i zadržavanjem u višem energetskom stanju označavaju kraj podatka nakon kojih ponovno dolazi u stanje „Idle“ u kojem je onemogućeno slanje i primanje podataka. Ovakav način serijske komunikacije fizički se odvija preko pinova (koji se u podatkovnom prijenosu nazivaju kanali) označenih oznakama „TX“ (eng. Transmitter) i „RX“ (eng. Receiver) koji osim integriranosti na sam mikrokontroler dolaze dodatno integrirani i u USB priključak odnosno kabel kao što je vidljivo sa slike.



Slika 3.4. Prikaz unutarnje građe USB-a tipa A i tipa B. [2]

UART protokoli se fizički odvijaju i koriste u komunikacijama putem USB kabla.

3.3. ICSP komunikacijski protokol

ICSP (eng. In Circuit Serial Programming) protokol je protokol koji spada u SPI protokole (eng. Serial Periferal Interface) to jest protokole komunikacije između dvaju ili više uređaja. Uređaj koji šalje informaciju naziva se odašiljač (eng. Master), uređaj koji prima informaciju naziva se prijemnik (eng. Slave). SPI protokole karakteriziraju sljedeća svojstva a to su;

-Moguća komunikacija u smjeru odašiljač i prijemnik te u smjeru prijemnik odašiljač (eng. Full duplex serial communication protocol).

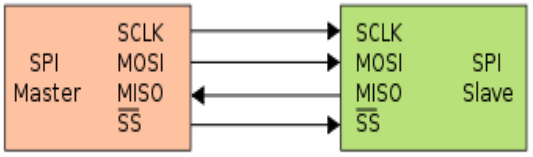
-Odvija se uz prisutnost taktnog signala produciranog od strane odašiljača koji služi za sinkroniziranje podataka.

-Potrebne minimalno 4 kanala odnosno fizička kontakta za komunikaciju.

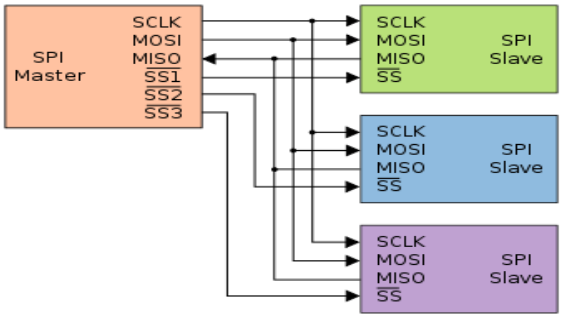
-Moguće komunicirati uz postojanje samo jednog odašiljača te jednog ili više prijemnika.

-Široka primjena u komuniciranju na kratke udaljenosti uglavnom su to ugradbeni sustavi alarmi i različita senzorika i slično.

Kao što je objašnjeno u poglavlju 2.1.4 na Arduino Uno pločici se nalaze dvije posebne skupine od po šest pinova s oznakama ; MISO, MOSI, SCK, Vcc, GND te RESET. Na slikama 3.5 i 3.6 prikazan je jednostavan primjer SPI komunikacije koja koristi ICSP pinove Arduino Uno pločice.



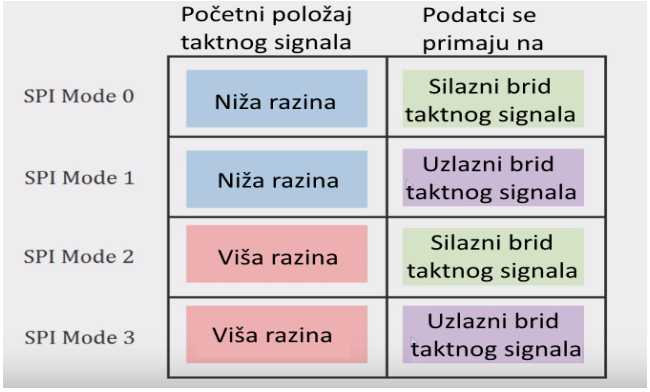
Slika 3.5. Fizičko ostvarivanje ICSP komunikacije [7]



Slika 3.6. Fizičko ostvarivanje ICSP komunikacije s više „Slave“ uređaja [7]

Pinovi V_{CC} i GND se koriste kao napajanje i nisu prikazani na prethodnoj slici zbog jednostavnijeg grafičkog prikaza i objašnjenja samog principa ICSP komunikacijskog protokola. Pomoću pina SCK (SCLK) (eng. Serial Clock) odašiljač prijemniku odašilje takni signal određene frekvencije kako bi se podaci mogli međusobno sinkronizirati. Takni signal je uglavnom pravokutni signal koji može biti invertiran ili neinvertiran. Parametar pomoću kojeg saznajemo da li se radi o invertiranom ili o neinvertiranom obliku taktnog signala je polaritet koji se dogovorno označava oznakom; „CPOL“ (eng. Clock Polarity). Ako je vrijednost ovog parametra jednaka 0, radi se o ne invertiranom taktnom signalu, a ako vrijednost parametra iznosi 1, radi se o invertiranom taktnom signalu. Ako je taktni signal neinvertiran neaktivni dio signala predstavljen je logičkom nulom dok je aktivni dio signala predstavljen logičkom jedinicom. Za invertiranje signala imamo obratnu situaciju čiji je neaktivni dio taktnog signala predstavljen je logičkom jedinicom dok je aktivni dio taktnog signala predstavljen logičkom nulom. Budući da se radi o pravokutnom periodičnom taktnom signalu koji se osim aktivnog i pasivnog dijela sastoji od uzlaznog i silaznog brida pomoću kojih se može odrediti vrsta sinkronizacije da li se podaci uzrokuju za vrijeme uzlaznog ili za vrijeme silaznog brida. Vrstu sinkronizacije opisuje parametar faze taktnog signala koji se

dogovorno označava oznakom; „CPHS“ (eng. Clock Phase). Ako je iznos parametra faze taktnog signala 0, uzorkovanje signala vrši se za vrijeme uzlaznog brida, a ako je iznos faze taktnog signala jednak 1, uzorkovanje signala vrši se za vremena silaznog brida taktnog signala. Budući da parametri polarnosti i faze taktnog signala mogu imati samo dvije vrijednosti odnosno 0 i 1, između njih postoje četiri SPI moda rada ovisno o međusobnim kombinacijama vrijednosti faze i polariteta taktnog signala, kako je prikazano na slici 3.7, a označavaju se brojevima od 0 do 3.



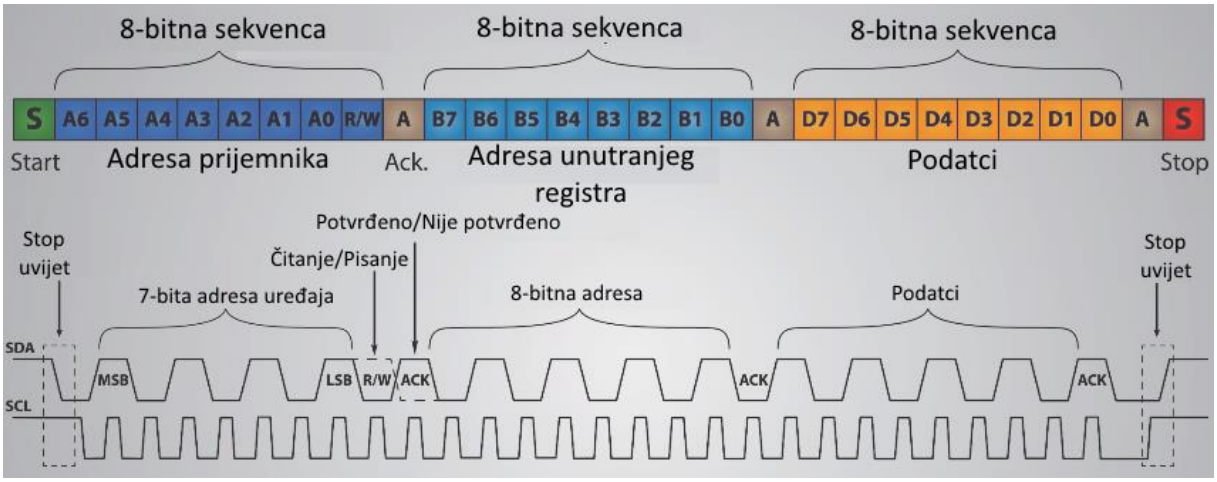
Slika 3.7. Načini rada ICSP protokola [8]

Pin označen oznakom MOSI (eng. Master Out Slave In) šalje određenu informaciju određenog uređaja odašiljača prema istoimenom pinu na određenom uređaju prijemniku. Za odabir s kojim će se uređajem prijemnikom komunicirati služi pin označen oznakom; „SS“, (eng. Slave Select) na uređaju prijemniku, dok za istu svrhu na uređaju odašiljaču uz pin SS mogu služiti bilo koji drugi pinovi opće namjene tj. GPIO pinovi (eng. General Purpose Input or Output). Uređaj prijemnik s kojim će se komunicirati odabere se na način da se na SS pin uređaja prijemnika s kojim želimo komunicirati dovede vrijednost logičke nule, dok su na ostalim uređajima prijemnicima koji su spojeni na uređaj odašiljač vrijednosti SS pina predstavljene logičkim jedinicama. Dok se pomoću pinova „MOSI“ vrši serijska komunikacija na relaciji uređaj odašiljač uređaj prijemnik, pomoću pinova označenim oznakom; „MISO“ vrši se komunikacija u suprotnom smjeru, odnosno uređaj odašiljač postaje uređaj prijemnik dok uređaj prijemnik postaje uređaj odašiljač te je time, uz pinove napajanja, zatvoren krug SPI komunikacije.

3.4. I2C komunikacijski protokol

Za razliku od SPI serijskog komunikacijskog protokola koji omogućuje istovremeno dvosmjernu komunikaciju u smjeru uređaj odašiljač uređaj prijemnik, i u suprotnom smjeru odnosno uređaj prijemnik uređaj odašiljač, I2C (eng. Inter Integrated Circuits) protokol nema tu mogućnost ali je i dalje moguća dvosmjerna komunikacija u smjeru prijemnik (eng. Slave) odašiljač (eng. Master), ali tek nakon nekog vremena odnosno nakon što se obavi komunikacija u smjeru odašiljač-prijemnik, što je poznato kao „half duplex“ serijski komunikacijski protokol, za razliku od ICSP komunikacijskog protokola koji je „full duplex“

serijski komunikacijski protokol. Također razlika I2C protokola u odnosu na ICSP protokol je ta da su za komunikaciju između odašiljača i prijemnika potrebne samo dvije žice odnosno dva kanala u odnosu na 4 potrebna kanala odnosno žice kod ICSP protokola. Dok ICSP serijski komunikacijski protokol omogućava komunikaciju između samo jednog uređaja odašiljača i više uređaja prijemnika, I2C serijski komunikacijskih protokol omogućava komunikaciju između više odašiljača i više prijemnika što nailazi na dosta široku primjenu. Zajedničke karakteristike ova dva protokola su te da oba protokola spadaju u sinkrone komunikacijske protokole te da imaju veliku primjenu u serijskom komuniciranju na kratke udaljenosti. I2C protokol se koristi i za unutarnju komunikaciju između određenih komponenti određenog uređaja odnosno uređaja koji na sebi imaju integriran veći broj senzora (npr. ADXL345 akcelerometar) te omogućava komunikaciju između 128 uređaja kada se koristi 7-bitno adresiranje, odnosno 1024 uređaja kada se koristi 10-bitno adresiranje. Protokol se odvija na način objašnjen nastavku prikazan na slici 3.8.



Slika 3.8. Princip rada I2C protokola [9]

Već smo naučili da je I2C protokol, protokol za čiju su komunikaciju potrebna dva kanala (odnosno dva fizička kontakta, žice), a to su kanali označeni oznakama SDA (eng. Serial Data) i kanal označen oznakom SCL (eng. Serial Clock). Signali koji se šire duž SDA i SCL kanala su pravokutni signali. Cijeli proces komunikacije počinje kada pravokutni signal SDA kanala iz definirano dugog stanja logičke više razine prijeđe u logičku nižu razinu što se interpretira kao Start-bit serijske komunikacije kao što je prikazano na slici 3.8, te pokreće sinkronizaciju podataka s taktnim signalom u SCL kanalu. Nakon što je početak sinkronizacije uspostavljen potrebno je pronaći uređaj s kojim će se komunicirati putem I2C serijskog komunikacijskog protokola, tj. uređaj u kojeg ćemo upisati odnosno uređaj s kojeg želimo čitati određenu informaciju. Za objašnjavanje ćemo koristiti slučaj sedam-bitnog adresiranja zbog jednostavnosti prikaza. Potrebno pratiti sliku 3.8 u nastavku. U obliku kodne riječi dužine sedam bitova se zapisuje binarna vrijednost adrese uređaja s kojim želimo uspostaviti komunikaciju. Unutar programskog okruženja vrijednosti adrese najčešće zapisujemo u heksadecimalnom obliku pri čemu je jedna znamenka heksadecimalnog oblika predstavljena s četiri znamenke odnosno bita binarnog brojevnog sustava. Npr. ako imamo binarnu vrijednost 01010011 koja je zapisana u obliku binarnog brojevnog decimalnog sustava u heksadecimalnom zapisu ova vrijednost će iznositi 0x53 pri čemu prefiks 0x ne

predstavlja nikakvu vrijednost , zapravo označava da se radi o zapisu u heksadecimalnom brojevnom obliku. Fizikalna interpretacija binarnom zapisu broja 01010011 je ta da se pravokutni signal SDA kanala ako želimo zapisati vrijednost nula, nalazi u nižoj i odnosno višoj logičkoj razini ako želimo zapisati vrijednost binarne jedinice. Kada zapišemo sedam-bitnu vrijednost adrese uređaja s kojim želimo komunicirati sljedeći bit definira da li želimo upisivati informaciju u odabrani uređaj ili pak želimo pročitati informaciju s odabranog uređaja. Osmi bit adrese uređaja kao što je vidljivo sa slike najčešće se označava oznakom R/W (eng. Read or Write). Ako želimo informaciju upisati u određeni uređaj vrijednost posljednjeg bita 8-bitne sekvence adrese uređaja bit će 0 to jest pravokutni signal SDA kanala bit će u nižoj logičkoj razini odnosno ako informaciju želimo pročitati iz određenog uređaja vrijednost posljednjih bita 8-bitni sekvence adrese uređaja bit će jedan, odnosno pravokutni signal SDA kanala bit će u vašoj razini. Nakon 8-bitne sekvence za prepoznavanje uređaja s kojim želimo komunicirati dolazi bit pomoću kojeg razaznajemo dali je uređaj uspješno prepoznat ili ne. Ako uređaj nije uspješno prepoznati vrijednost pravokutnog signala SDA kanala bit će u logičkoj nuli, a ako je uređaj uspješno prepoznati vrijednost pravokutnog signala SDA kanala bit će u logičkoj jedinici. Ovaj bit za prepoznavanje na slici najčešće se označava oznakom A (eng. Acknowledgement). Ako se kojim slučajem dogodi da traženi uređaj nije uspješno pronađen, a razlozi mogu biti vrlo različiti od raznih smetnji, šumova do toga da je uređaj s kojim želimo komunicirati trenutno izvršavao neku drugu zadaću te nije stigao odgovoriti na naš zahtjev, postupak se ponavlja ispočetka. Nakon što je uređaj uspješno prepoznan slijedi pronalaženje potrebnog registra unutar pronađenog uređaja pomoću nove 8-bitne sekvence. Prepoznavanje se vrši na način gotovo identičan načinu prepoznavanja adrese uređaja koji smo prethodno opisali uz naknadnu provjeru da li je registar uspješno prepoznat pomoću bita kojeg označavamo sa slovom „A“ vidljivo iz slike 3.8, a jedina razlika u tome, u odnosu na prvi korak je ta da se cijela 8-bitna sekvenca koristi za unošenje adrese registra s kojim želimo stupiti u I2C serijsku komunikaciju, dok je se za prethodni postupak 7 od 8 bitova koristilo za prepoznavanje uređaja s kojim želimo komunicirati dok se posljednji osmi bit koristio za definiranje funkcije da li želimo čitati odnosno da li želimo upisivati određenu vrijednost u dati registar, što u ovom koraku nije slučaj. Nakon što je prepoznat i registar uređaja s kojim želimo komunicirati dolazi do slanja ili primanja 8-bitne informacije čija će se uspješnost transakcije ponovno provjeriti pomoću bita označenog po treći put sa slovom A. Posljednji korak unutar komunikacije pomoću ovog protokola je označavanje završetka komunikacije koji se vrši pomoću bita označenog slovom S (eng. Stop) na crvenoj pozadini kao što je prikazano na slici 3.8. Fizikalno stop bit se definira pomoću prelaska pravokutnog signala iz stanja niže u stanje više razine nakon određenog vremenskog perioda unutar kojeg je vrijednost pravokutnog signala predstavljena nižom razinom napona. Ovim korakom zatvoren je krug I2C serijske komunikacije te se cijeli postupak po potrebi može ponovno vrtjeti u krug s istim ili različitim vrijednostima podataka koje želimo poslati odnosno primiti.

3.5. Ukratko o ADC i DAC pretvaraču

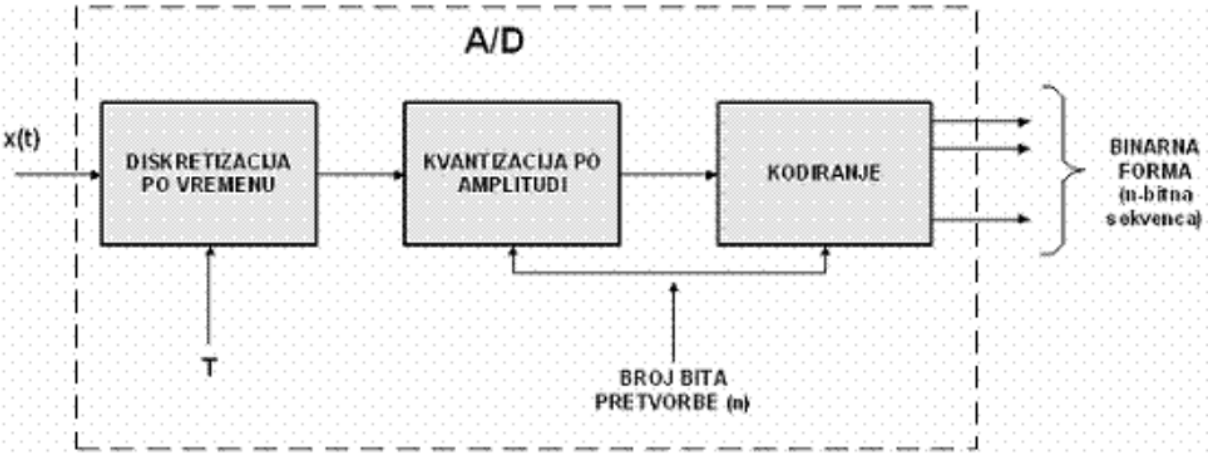
Analogno-digitalni pretvarač ADC (eng. Analog to Digital Converter) je realni uređaj pomoću kojeg vršimo postupak uzorkovanja odnosno pretvaranja analognog signala u diskretni, a potom kodiranja u digitalni signal. Digitalizacija je postupak koji se odvija u sljedeća tri koraka;

-kvantizacija po vremenu

-kvantizacija po amplitudi

-kodiranje u n-bitnu formu

Shematski prikaz A/D pretvarača nalazi se na slici 3.9.



Slika 3.9. Shematski prikaz A/D pretvarača [10]

Pri čemu je x(t) analogni signal definiran u vremenu, a T period uzorkovanja A/D pretvarača. Kvantizacija po vremenu je postupak u kojem se definira vrijednost signala u točno određenim vremenskim trenucima, a usko je povezana s frekvencijom uzorkovanja koja mora biti minimalno dvostruko veća od najveće frekvencije signala kako bi se digitalni signal mogao jednoznačno rekonstruirati. Frekvencija uzorkovanja usko je povezana s periodom uzorkovanja koji je naznačen na slici 3.9 s oznakom „T“, a vidljivo je iz formule 3.1.

$$f_s = \frac{1}{T} \qquad (3.1)$$

T-period uzorkovanja f_s-frekvencija uzorkovanja

Rekonstrukcija je postupak pretvorbe iz digitalnog oblika signala u analogni oblik signala u kojem će biti govora u nastavku pri objašnjavanju D/A pretvorbe.

1928. Harry Nyquist je postavio teoriju koja glasi;“ Sinusni signal se može u potpunosti rekonstruirati iz uzorkovanog signala u koliko je frekvencija uzorkovanja f_s barem dvostruko veća od frekvencije sinusnog signala f_0 .“

$$f_s > 2f_0 \qquad (3.2.)$$

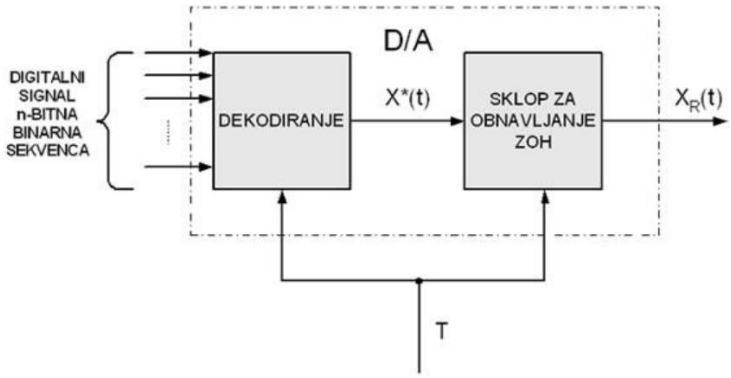
f_0-frekvencija sinusnog signala

Claude E. Shannon je 21 godinu kasnije proširio Nyquistov teorem na bilo kakve signale, a ne samo sinusne kako je bilo definirano od strane Nyquista, te i matematički dokazao cjeloviti teorem uzimanja uzoraka u koji nećemo detaljno ulaziti i prihvatit ćemo ga kao takvog u daljnjim razmatranjima. Kvantizacija po amplitudi provodi se na način da amplitudna vrijednost zabilježena u trenutku uzorkovanja ostaje nepromijenjena sve do sljedećeg trenutka uzorkovanja. Kada su izvršeni postupci kvantizacije po vremenu i amplitudi slijedi pridavanje vrijednosti dobivenom signalu pomoću postupka kodiranja određenom n-bitnom rezolucijom. A/D pretvarač karakteriziraju dvije temeljne značajke, a to su;

-vertikalna rezolucija

-frekvencija uzorkovanja

Digitalno analogni pretvarač DAC (eng. Digital to Analog Converter je realni uređaj pomoću kojeg se vrši rekonstrukcija, odnosno postupak pretvorbe iz digitalnog u analogni oblik signala. D/A pretvarač se sastoji od uređaja za dekodiranje te sklopa za obnavljanje kako je prikazano na slici 3.10.



Slika 3.10. Shematski prikaz D/A pretvarača [11]

Dekoder, kao što i sam naziv kaže vrši postupak dekodiranja, odnosno očitavanja informacija pohranjenih u digitalnom obliku signala te digitalni signal pretvara u diskretni signal koji potom odlazi do sklopa za obnavljanje. U sklopu za obnavljanje dolazi do pretvorbe iz diskretnog u analogni oblik signala na način da se vrijednost u trenutku rekonstruiranja signala zadrži nepromijenjena do sljedećeg trenutka uzorkovanja. Dvije temeljne karakteristike D/A pretvarača su;

-broj bita kojom se iskazuje amplitudna vrijednost digitalnog signala

-frekvencija obnavljanja

Frekvencija obnavljanja uglavnom je usklađena s frekvencijom uzorkovanja.

A/D i D/C pretvarači dolaze integrirani na većini mikrokontrolera.

4. ARDUINO WI-FI ROBOT S PROTOKOLOM ZA ZAOBILAŽENJE PREPREKA

U ovom poglavlju ćemo detaljno objasniti;

-način izrade samog robota to jest hardversko spajanje komponenti

-građu pojedinih komponenti od kojih je robot građen

-programsko okruženje pomoću kojeg ćemo programirati robot

-arhitekturu komunikacije mobilnog uređaja i robota kojim želimo upravljati

-načine optimizacije programskog koda

-cjelokupni postupak izrade mobilni Wi-Fi android aplikacije pomoću koje upravljamo robotom

-testa

-programski kod

Ukratko rečeno proizvesti ćemo robot kojeg ćemo moći upravljati pomoću aplikacije koju u potpunosti mi izrađujemo uz sposobnost aktiviranja posebnog načina rada u kojem će robot potpuno samostalno zaobilaziti prepreke koje mu se nalaze u okolini.

4.1. Popis i objašnjenja komponenti

U nastavku ćemo napisati popis komponenti koje će biti korištene u izradi ovog projekta. Ovaj projekt moguće je realizirati i korištenjem različitih modifikacija komponenti koje će biti pobrojane u nastavku s tim da će biti potrebne određene preinake u spajanju komponenti što će automatski povući za sobom potrebne preinake u programskom kodu. Potrebno je razumjeti princip izrade iz razloga što unatoč svim promjenama koje nastaju korištenjem različitih uređaja na realizaciji ovog projekta, princip ostaje isti.

Komponente koje ćemo koristiti u projektu su sljedeće;

-Servo motor SG90

- Ultrazvučni senzor HC-SR04

- Modul ESP-32

-Istosmjerni motori

- Mikrokontroler L298N model X

- Troznamenkasti 7-segmentni LED ekran

- Baterije

4.1.1. Servo motor SG90

Servo motor je sustav koji se sastoji od;

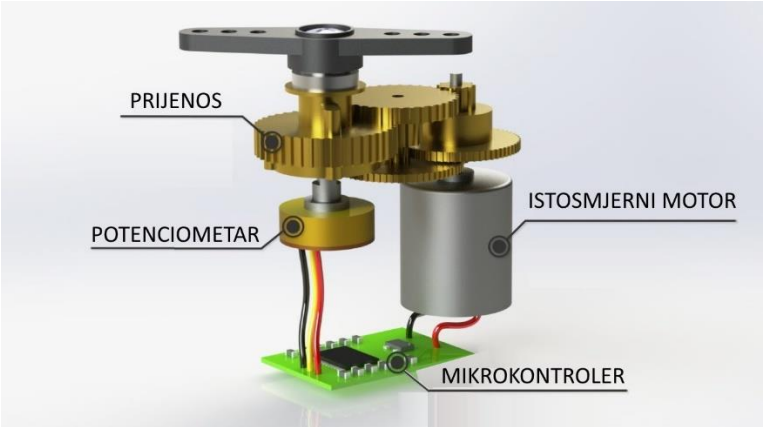
-istosmjernog motora

-detektora položaja

-mikrokontrolera

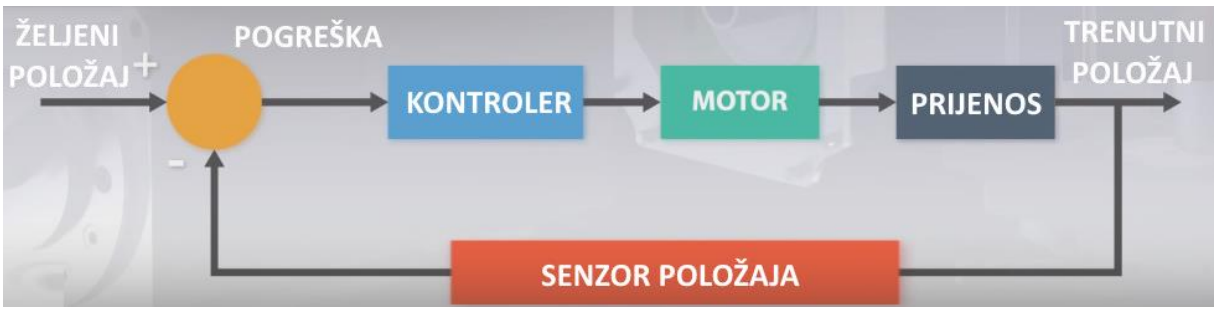
-prijenosa

Raspored pobrojanih komponenti prikazan je na slici 4.1.



Slika 4.1. Unutarnja građa servo motora SG90 [12]

Servo motor je zatvoreni sustav s negativnom povratnom vezom kojeg blokovski možemo prikazati na sljedeći način.



Slika 4.2. Blokovski prikaz principa rada servo motora SG90 [12]

Objašnjenje prethodnog blokovskog dijagrama je sljedeći. Negativna povratna veza je mehanizam koji stanje s izlaza s negativnim predznakom dovodi na komparator pomoću kojeg dobivamo informaciju koju veličinu treba proslijediti izlazu. U slučaju negativne povratne veze ako za primjer uzmemo kutnu brzinu vrtnje izlazne osovine istosmjernog motora koja je veća odnosno manja od brzine vrtnje koja je nama potrebna, pomoću negativne

povratne veze brzina vrtnje će biti smanjena odnosno povećana na određenu brzinu vrhnje koja je nama potrebna. Da na prethodni primjer djelujemo pozitivnom povratnom vezom, ako se izlazna osovina vrti većom brzinom od brzine koja je nama potrebna sustav će izlazu poslati naredbu da se brzina vrtni dodatno poveća što neće dati željeni učinak, a motor će se početi kontinuirano ubrzavati samim tim vući će sve veću struju dok ne dođe do pregaranja vodiča namota motora, osim ako u sam motor nije ugrađen mehanizam koji ga štiti od samouništenja, te će motor biti uništen. Također isto vrijedi i za slučaj ako se izlazna osovina motora vrti brzinom manjom od željene brzine vrtnje osovine motora, pomoću pozitivne povratne veze doći će do smanjenja brzine vrtnje motora dok se osovina motora u potpunosti ne prestane vrtjeti što nije željeni ishod. Na ulazu servo motora nalaze se tri pina označena oznakama; Power, GND i Signal.

Slika 4.3. Oznake pinova servo motora SG90 [13]

Pomoću pina označenog oznakom „Power“, motor dobiva potrebnu električnu energiju odnosno struju i napon za ispunjavanje svoje funkcije, dok pin „Ground“ služi za zatvaranje električnog strujnog kruga. Kada je na servo motor doveden zadovoljavajući izvor električne energije mikrokontroler može početi interpretirati PWM signal koji je poslan od nekog drugog mikrokontrolera putem pina označenog oznakom „Signal“. Detaljno objašnjenje PWM signala je dano u poglavlju 3.1. Kada mikrokontroler koji se nalazi unutar servo motora interpretira primljeni PWM signal tada on šalje struju u istosmjerni motor koji se također nalazi unutar kućišta servo motora, te se osovina istosmjernog motora počinje vrtjeti određenom kutnom brzinom u određenom smjeru. Smjer okretanja osovine istosmjernog motora mijenja se promjenom smjera struje koja teče kroz motor što se u ovom sustavu ostvaruje korištenjem H-mosta integriranog na sam mikrokontroler. Način rada H-mosta detaljno je objašnjen u poglavlju 4.1.8. Osovina istosmjernog motora pokreće zupčanik koji pokreće izlaznu osovinu servo motora. Zupčani prijenos može biti realizirana na različite načine ovisno o potrebi može biti plastični, metalni i sl, a temeljna uloga mu je da s osovine istosmjernog motora na određeni način prenese potreban zakretni moment sile na izlaznu osovinu tako da je jedan od načina podjele servo motora upravo podjela prema iznosu zakretni momenta sile po jedinici dužine udaljenosti od centra vrtnje izlazne osovine koji se najčešće označava iznosom kilograma po centimetru. Npr. ako imamo iznos zakretnog momenta sile definiranog od strane proizvođača servo motora od 50 N/cm (5 kilograma po centimetru) to znači da je iznos momenta sile na udaljenosti 1cm od centra vrtnje izlazne osovine servo motora 50 N (Njutn) dok bi isti taj iznos na udaljenosti od 2 cm od centra vrtnje izlazne osovine servo motora iznosio 25 N. Na izlaznu osovinu koji pokreću zupčanici s

unutarnje strane je smješten promjenljivi otpornik odnosno potenciometar čiji se klizač rotira istovremeno s izlaznom osovinom servo motora te na taj način mijenja iznos svog otpora koji ako uzmemo za primjer da je struja koja protječe kroz potenciometar konstantna, direktno mijenja napon na izlazu potenciometra. Napon na izlazu potenciometra se kodira na određeni način da za različite vrijednosti napona na izlazu potenciometra kodiramo određeni stupanj zakreta izlazne osovine te na taj način imamo uvid u položaj izlazne osovine servo motora. Izlazni napon potenciometra se dovede na komparator koji uspoređuje trenutni položaj izlazne osovine servo motora i položaj osovine koji želimo postići te ukoliko ta dva položaja nisu identična nastupit će za kretanje izlazne osovine na prethodno opisan način čime smo zatvorili cjelokupni način funkcioniranja procesa koji se ovija unutar servo motora.

Servo motor SG90 karakterizira;

-okretni moment od 13 N/cm odnosno 15 N/cm za ulazni napon od 4,8 V odnosno 6 V

-struja od 100 miliampera odnosno 120 mA

-širina perioda pripadnog PWM signala od 900 do 2.100 mikrosekundi [μs].

Ove i sve ostale informacije vezane za tip servo motora SG90 možete pronaći u podacima proizvođača za dati proizvod (eng. Data sheet). Rotiranje izlazne osovine je moguće unutar intervala od 180 stupnjeva što je ujedno i maksimalni kut zakreta izlazne osovine servo motora tipa SG90.

Servo motor će imati ulogu rotiranja ultrazvučnog senzora koji mjeri udaljenost robota do najbližih prepreka u okolini ako takve postoje.

4.1.2. Ultrazvučni senzor HC-SR04

Ultrazvučni senzor HC-SR04 je senzor koji služi za mjerenje udaljenosti robota od prepreka koje ga okružuju u okolini, ako takve postoje. Montiran je na izlaznu osovinu servo motora s kojom se zajedno okreće, pokrivajući kut od 180° kako bi robot uz prepreke koje se nalaze na pravcu njegovog kretanja uspješno zaobišao i bočne prepreke, to jest prepreke koje mu se nalaze s bočnih strana, a koje sa zamišljenim pravcem kretanja zatvaraju kut od $\pm$ 90°.

Ultrazvučni senzor HC-SR04 kao što je prikazano na slici 4.4 sastoji se od;

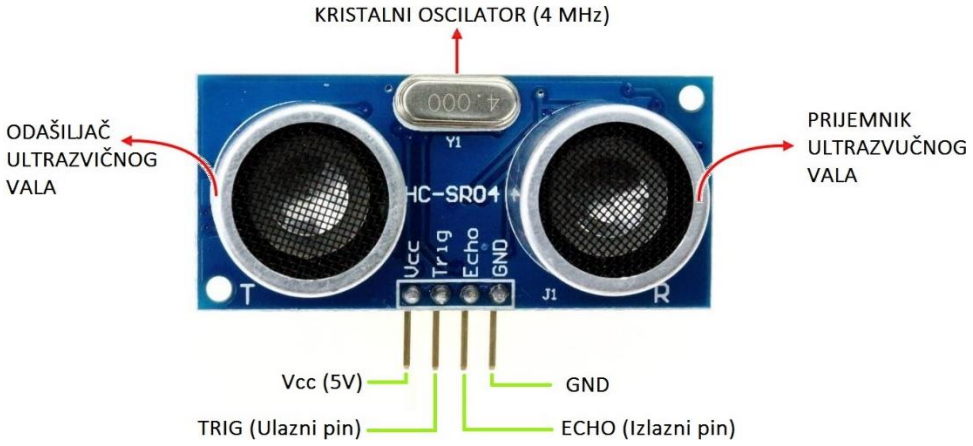
-četiri ulazna pina označena oznakama redom s lijeva na desno; V_{cc} , Trig, Echo i Ground

-odašiljača ultrazvučnog signala

-prijemnika ultrazvučnog signala

- frekvencijskog oscilatora

-mikrokontrolera



Slika 4.4. Ultrazvučni senzor HC-SR04

Uloge pinova su sljedeće.

Pin označen oznakom V_{cc} služi za dovođenje određene količine električne energije potrebne za pravilno funkcioniranje ovog senzora.

Ground-služi za zatvaranje električnog strujnog kruga.

Trig-šalje mikrokontroleru senzora HC-SR04 pravokutni impuls amplitude 5 V perioda 10 mikro-sekundi [μs].

Echo-daje iznos vremena za koje se poslani ultrazvučni signal reflektira od prepreke.

U nastavku ćemo objasniti način rada ultrazvučnog senzora HC-SR04 prikazanog na slici 4.4. Pomoću programskog koda imamo pristup dvama pinovima, to su pinovi „Trig“ i „Echo“. Za primjer uzmimo da preko pina označenog oznakom „Trig“ prema procesoru ultrazvučnog senzora pošaljemo pravokutni impuls amplitude 5 V i vremenskog perioda 10 mikro-sekundi [μs]. Taj se signal poslan od strane mikrokontrolera vanjskog uređaja preko ulaznog Trig pina mikrokontroleru ultrazvučnog senzora interpretira na način da pomoću odašiljača u određenom smjeru pošalje 8 pravokutnih impulsa frekvencije 40 KHz koji putuju kroz zrak približno brzinom 343 m/s. Brzina ultrazvučnih valova na temperaturi od 20℃ iznosi 343 m/s što varira ovisno o temperaturi, tlaku, kemijskom sastavu zraka i sl. Ako ovu brzinu širenja ultrazvučnog vala u zraku pretvorimo u nama pogodniji oblik za interpretiranje ona iznosi 0.0343 cm/μs. Nakon što je poslan i zadnji u našem slučaju 8 impuls preko odašiljača, prijemnik počinje slušati s namjerom da osluhne poslani slijed impulsa od strane uređaja odašiljača koje će se ukoliko prepreka postoji reflektirati u suprotnom smjeru od smjera u kojem su poslani. Uspješnost reflektiranja poslanih signala uvelike će ovisiti o površini prepreke koju želimo interpretirati kao i o udaljenosti same prepravke od prijemnika signala. Ako prijemnik ne primi reflektirani signal unutar 38 ms prijemnik prestaje s procesom slušanja radi uštede električne energije, jer iako ne interpretira reflektirane signale, uređaj prijemnik i dalje troši najveću količinu električne energije na ultrazvučnom senzoru čak više i od uređaja odašiljača u aktivnom radu. Kada prijemnik interpretira i posljednji reflektirani impuls od ukupno osam po slanih ultrazvučnih impulsa od strane uređaja odašiljača, prelazi u stanje mirovanja. Iznosu vremena od početka do prestanka slušanja prijemnika moguće je pristupiti pomoću „Echo“ pina, a označit ćemo ga oznakom „t“. Promotrimo sljedeću jednadžbu pomoću koje ćemo izračunati udaljenost određene prepreke od samog senzora.

$$v = \frac{s}{t} \quad (4.1.)$$

$$s = vt \quad (4.2.)$$

v-brzina širenja ultrazvučnog vala t-vrijeme putovanja ultrazvučnog vala

s-put koji prijeđe ultrazvučni val

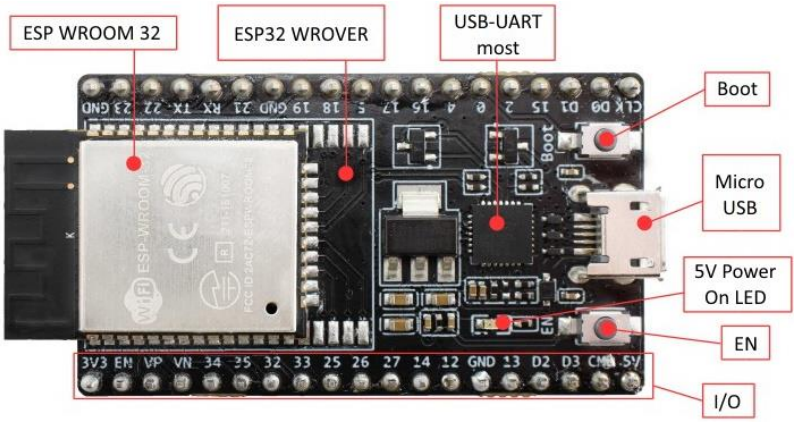
Budući da računamo udaljenost koju signal pređe brzinom v za vrijeme do određene prepreke potrebna nam je samo polovica iznosa vremena kojem je moguće pristupiti pomoću Echo pina iz razloga što je iznos vremena kojem je moguće pristupiti pomoću Echo pina vrijeme izmjereno od uređaja odašiljača do prepreke i od prepreke do uređaja prijemnika to jest mjereno je vrijeme za dvostruko veću udaljenost od udaljenosti koja nam je potrebna. Zanima nas izmjereno vrijeme putovanja signala od uređaja odašiljača do prepreke što predstavlja upravo polovici izmjerenog vremena kojem možemo pristupiti pomoću Echo pina, pomoću kojeg računamo polovici puta koju emitirani ultrazvučni signal pređe, što nam upravo odgovara udaljenosti od senzora do prepreke. Stoga konačni izraz za proračun udaljenosti prepreke od senzora glasi;

$$s = v\frac{t}{2} \quad (4.3.)$$

Maksimalna udaljenost na kojoj ultrazvučni senzor može percipirati adekvatnu prepreku je 4 m. Istosmjerni napon koji je potreban za aktivan rad na frekvenciji od 40 KHz je 5 V dok je struja koju vuče ovaj senzor iznosa 15 mA. Također važna karakteristika ovog senzora je kut mjerenja $\pm$ 15° od željenog pravca kretanja.

4.1.3. Modul ESP-32

ESP-32 je mikrokontroler koji uz funkcije koje posjeduje pločica Arduino Uno koje su detaljno objašnjene u poglavlju 2.1, posjeduje mogućnost spajanja na globalnom internet mrežu, korištenje „Bluetooth“ protokola, te posjeduje i određene pinove koji mogu detektirati dodir i još mnoštvo zanimljivih i korisnih mogućnosti koje ćemo objasniti u nastavku. Građa mikrokontrolera prikazana je na slici 4.5.



Slika 4.5. Komponente ESP-32 modula [15]

Iz slike je vidljivo da se mikrokontroler ESP-32 sastoji od;

-Mikroprocesora ESP-WROOM-32

-USB-UART mosta

-Ulazno-izlaznih pinova (jedna skupina na slici označena s I/O, gotovo svi pinovi na ovom mikrokontroleru su ulazno-izlazni pinovi)

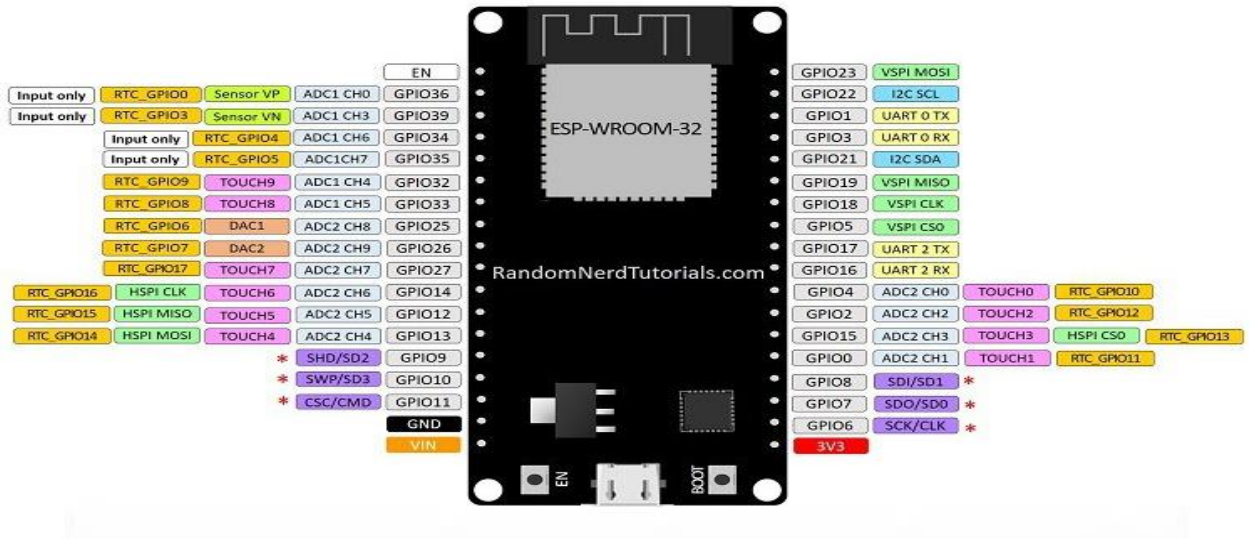
-Ledica

-„EN“ i „Boot“ tipkala

-Priključka micro-USB tipa A

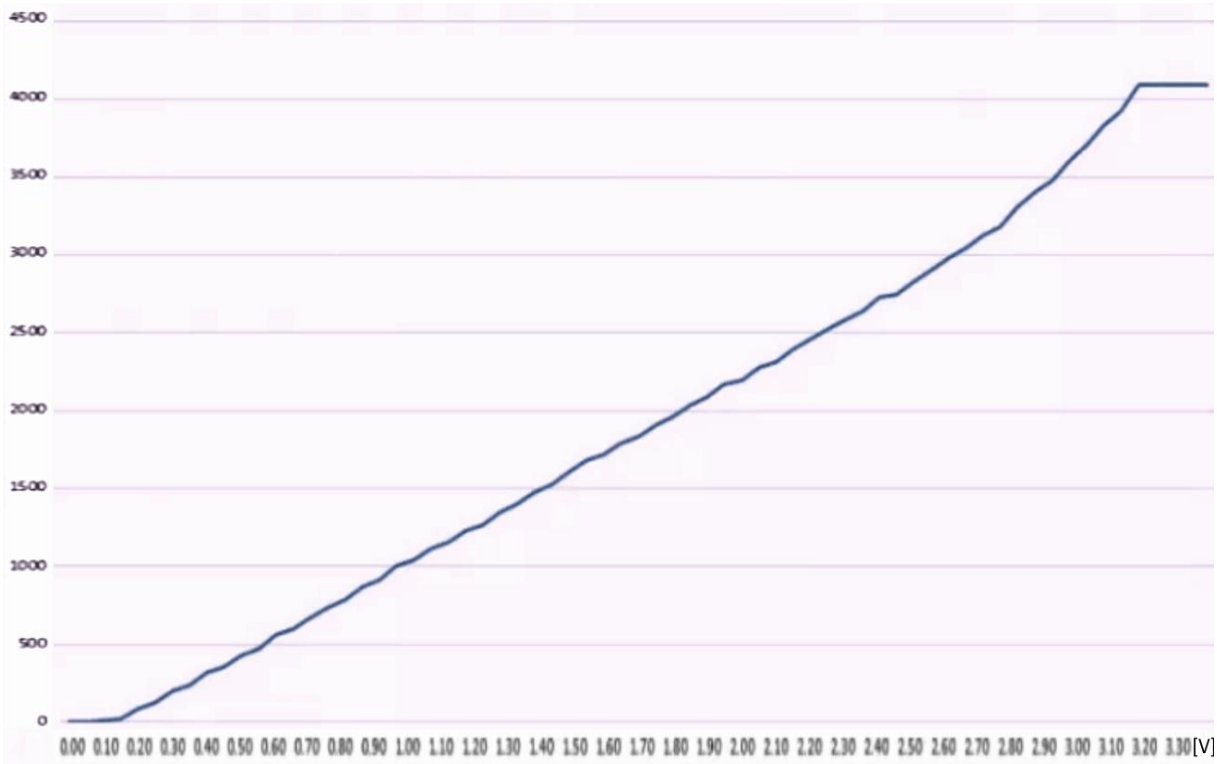
Pinovi i funkcije pinova mikrokontrolera ESP-32.

Kao što je vidljivo sa slike 4.6 velika većina pinova mikrokontrolera ESP-32 je multifunkcionalna.



Slika 4.6. Prikaz funkcija pinova ESP-32 modula [16]

Također velika većina pinova označena je oznakom „GPIO“ (eng. General Purpose Input or Output pin) čija je funkcija slanje ili primanje određene informacije koja je u osnovi na određeni način kodirana pomoću različitih vrijednosti struje, odnosno napona ovih pinova što nailazi na ogromnu primjenu u realizaciji različitih projekta. Iznimke su pinovi označeni oznakama GPIO34, GPIO35, GPIO36 i GPIO39 koji u svojoj konstrukciji nemaju ugrađene „pull-up“ ili „pull-down“ otpornike te služe samo kao ulazi u mikrokontroler. „Pull-up“ i „pull-down“ otpornici detaljno su objašnjeni u poglavlju 2.2.9. Uz GPIO funkciju tu su i mnoge dodatne funkcije koje ćemo objasniti u nastavku. Mikrokontroler ESP-32 posjeduje 16 kanala koji omogućuju interpretiranje PWM signala čiji je princip funkcioniranja detaljno objašnjen u poglavlju 3.1. Za korištenje ovog tipa signala unutar programskog okruženja potrebno je definirati frekvencije signala, vremenski odnos između više i niže naponski razine unutar jednog perioda signala te odabrati kanal, to jest pin, na koji ćemo dovesti generirani signal. Također uz prethodno nabrojane vrste pinova mikrokontroler ESP-32 posjeduje i 18 ADC pinova (eng. Analog to Digital Converter) za analogno-digitalnu pretvorbu signala koja je detaljno objašnjena u poglavlju 3.5. Rezolucija ADC pinova od 12 bitova omogućuje razlikovanje 4.096 različitih naponskih razina u rasponu od 0 do 3,3 V. Iznimka su ADC2 pinovi koji ne mogu biti u uporabi kada je ovaj mikrokontroler koristi Wi-Fi protokol te je za svaku analogno-digitalnu pretvorbu za slučaj kada koristimo Wi-Fi protokol potrebno koristiti analogno-digitalni pinove označene oznakom ADC1. Budući da podjela ovih 4.096 naponski razina nije u potpunosti linearna odnosno prisutna su određena odstupanja na početku i na kraju karakteristike odnosa mogućih naponskih razina i napona kao što je vidljivo sa slike 4.7, nećemo moći razlikovati informaciju za vrijednosti napona između 0 V i 0,15 V odnosno između 3,1 V i 3,3 V zbog već spomenutih odstupanja na početku i na kraju prijenosne karakteristike dok je karakteristika svih ostalih naponskih intervala pretežno linearna te je moguće razlikovati gotovo svaku od preostalih 4.096 naponskih razina što nam omogućava pohranu isto tolikog broja različitih informacija.



Slika 4.7. Ovisnost ADC očitanja o naponu na pinu [16]

Ovaj mikrokontroler osim analogno-digitalne pretvorbe omogućava i obratan proces, to jest digitalno-analognu pretvorbu pomoću 2 DAC pina (eng. Digital to Analog Converter) čiji je postupak objašnjen u poglavlju 3.5. Rezolucije DAC pinova su 8-bitne što omogućava kodiranje 2^8 odnosno približno 256 različitih informacija.

Tu su i pinovi označeni oznakom; „Touch“, (eng. Touch) koji imaju sposobnost detektiranja dodira, a funkcioniraju na sljedeći princip. Unutar programskog koda definiramo određeni pin označen oznakom „Touch“ kojeg želimo postaviti u način rada da bude osjetljiv na dodir te na taj način odabranom pinu postavimo određenu frekvenciju titranja koja se mijenja za određeni iznos ako taj isti pin dodirnemo nekim predmetom. Budući da je došlo do promjene iznosa frekvencije signala zbog vanjskog utjecaja, tu promjenu možemo iskoristiti za kodiranje informacije;“ Pin je dodirnuti nekim tijelom iz okoline“, što na vrlo efikasan način možemo primijeniti u realizaciji različitih projekta. Ovaj princip ima dosta široku primjenu u projektima realizacije buđenja mikrokontrolera iz različitih režima rada.

Posebna skupina pinova su pinovi označeni oznakama „RTC“ (eng. Real-Time Clock) koji ovom mikrokontroleru pružaju informacije o realnom vremenu što nailazi na jako veliku primjenu u projektima u kojim želimo aktivirati ili deaktivirati određeni sustav u točno određeno vrijeme pripadne vremenske zone ili aktivirati određeni sustav za točno određeni vremenski period. Sposobnost prepoznavanja realnog vremena ovim pinovima omogućuje ULP procesor (eng. Ultra Low Power) koji zbog svoje iznimno male potrošnje ostaje aktivan čak i u najneaktivnijem načinu rada mikrokontrolera poznat pod nazivom duboko spavanje (eng. Deep Sleep). Zahvaljujući ovoj karakteristici unutar ULP procesora moguće je definirati vrijeme u kojem će mikrokontroler prijeći iz jednog u drugi način rada, npr. buđenje iz dubokog spavanja i prelazak u energetski zahtjevniji način rada. Osim nabrojanih vrsta pinova

tu su i pinovi označeni oznakama; GROUND, 3,3 V, Vin te pinovi koji služe za ostvarivanje komunikacije putem SPI, UART, I2C i I2S protokola koji su objašnjeni u poglavlju 3. Važno je napomenuti da su SPI pinovi označeni oznakama od GPIO 6 do GPIO 11 spojeni kao indikatori na ESP-WROOM-32 modul i nisu preporučeni za korištenje u druge svrhe.

Mikroprocesor ESP-WROOM-32

Mikroprocesor ESP-WROOM-32, mikrokontrolera ESP-32 u interakciji s ostalim komponentama mikrokontrolera omogućuje spajanje i generiranje vlastitog signala prema Wi-Fi i Bluetooth protokolu na frekvenciji od 2.4 GHz. Rad ovog mikrokontrolera moguć je u dva režima to su;

-režim rada kao pristupna točka

-režim rada kao stanica

U režimu rada kao pristupna točka, ESP-32 mikrokontroler omogućuje kreiranje vlastitog Wi-Fi signala na lokalnoj razini na kojeg se možemo spojiti pomoću bilo kojeg pametnog uređaja. Za kreiranje vlastitog Wi-Fi signala ESP-32 mikrokontroler koristi bilo koju dostupnu Internet mrežu koja nema zabranu korištenja određenih Wi-Fi mogućnosti. Unutar programskog koda uz naredbe za kreiranje vlastitog servera sa željenim nazivom, lozinkom određivanje maksimalnog broja uređaja koji istovremeno mogu biti spojeni na kreirani server potrebno je unijeti naziv i kod pristupne internetske mreže. Programirani ESP-32 mikrokontroler omogućuje ispisivanje IP adrese (eng. Internet Protokol) u serijskom monitoru programskog okruženja. Nakon povezivanja pametnog uređaja na novokreiranu mrežu unošenjem IP adrese ispisane u serijskom monitoru te njenim unošenjem u preglednik uređaja kojim želimo upravljati robotom, omogućeno je upravljanje projekta u našem slučaju robota putem Wi-Fi protokola.

Režim rada kao stanica se od režima rada kao pristupna točka razlikuje u tome što mikrokontroler ne stvara svoju vlastitu mrežu nego se spoji na dostupnu internetu mrežu na istu IP adresu na koju se spoji i pametni uređaj kojim želimo upravljati robota. Budući da ESP-32 podržava XML te JSON programski jezik na njemu je moguće kreirati internet aplikaciju, po našim željama i potrebama , pomoću koje možemo upravljati neki proces bilo gdje u svijetu. Ovaj mikrokontroler posjeduje ugrađene antene, releje, pojačala snage, pojačala slabih signala, razne filtere, elektromagnetni modul i slično. Razlika u performansama u odnosu na prethodno objašnjen Arduino Uno mikrokontroler je ogromna. Počnimo od većeg broja jezgri procesora, arhitekture dužine memorijskih lokacija podataka koji se obrađuju, frekvencije procesora CPU (koja je veća čak 10 puta) preko integriranih Wi-Fi i Bluetooth modula, količine memorije koje su praktički neusporedive, do mogućnosti korištenja više tipova protokola, većeg broja ADC i DAC pinova.

SPECIFIKACIJE/MIKROKONTROLER	ESP32	ESP8266	ARDUINO UNO
Broj jezgri	2	1	1
Arhitektura	32 Bit	32 Bit	8 Bit
CPU Frekvencija	160 MHz	80 MHz	16 MHz
WiFi	DA	DA	NE
BLUETOOTH	DA	NE	NE
RAM	512 KB	160 KB	2 KB
FLASH Memorija	16 MB	16 MB	32 KB
GPIO Pinovi	36	17	14
Protokoli	SPI, I2C, UART, I2S, CAN	SPI, I2C, UART, I2S	SPI, I2C, UART
ADC Pinovi	18	1	6
DAC Pinovi	2	0	0

Slika 4.8. Usporedba performansi mikrokontrolera ESP-32, ESP-8266 i Arduino Uno [17]

Uz sve navedeno ovaj mikrokontroler spada u skupinu mikrokontrolera niske cijene i potrošnje što ga svrstava među najkompletnije i najčešće korištene mikrokontrolere na globalnom razini.

4.1.4. USB (UART) most

USB (UART) most služi za uspostavljanje asinkrone serijske komunikacije između odašiljača i prijemnika na način koji je opisano u poglavlju 2.2.6.

4.1.5. LED indikatori

LED indikatori (eng. Light-Emitting Diode) su svjetleće diode koje daju povratne informacije o radu određenih dijelova integriranih na ovaj mikrokontroler. Također ove diode imaju mogućnost programiranja to jest testiranja programskog koda za određenu ledicu, bez potrebe za posjedovanjem eksternog fizičkog hardverskog dijela odnosno vanjske ledice tj. ovi LED indikatori u potpunosti mogu zamijeniti vanjsku diodu koju bi priključili na određeni pin radi testiranja programskog koda.

4.1.6. UBS priključak tipa mikro A (eng. Universal Serial Bus)

Ovaj tip priključka omogućuje fizičku realizaciju serijske komunikacije odnosno odvijanje UART protokola između odašiljača i prijemnika signala, u našem slučaju to su računalo i mikrokontroler ESP-32. Slanje i primanje podataka vrši se prema UART protokolu preko pinova označenih oznakama TX i RX koji su prisutni unutar svakog USB kabla.

4.1.7. „EN“ i „Boot“ tipkalo

Tipkala na mikrokontroleru ESP-32 su tipkala označena oznakama „EN“ (eng. Enable) i „Boot“ (eng. Bootloader), a imaju sljedeće funkcije;

EN- restartira mikrokontroler

Boot- omogućuje zapisivanje programskog koda unutar memorije mikrokontrolera

Tipkalo označeno oznakom „EN“ ima pristup regulatoru napona od 3,3 V kojem zbog svoje građe to jest ugrađenog „pull-up“ otpornika može omogućiti odnosno onemogućiti rad te na taj način restartira cijeli ESP-32 mikrokontroler. Tipkalo označeno oznakom „Boot“ omogućuje zapisivanje programskog koda unutar memorije mikrokontrolera ESP-32. Da bi programski kod bio primijenjen oba tipkala moraju surađivati na način da kao prvi korak pritisnemo tipkalo označeno oznakom „Boot“ kako bi omogućili zapisivanje (eng. Upload) programskog koda te u vrlo kratkom vremenskom intervalu dok još držimo tipkalo „Boot“ pritisnuto, pritisnemo i otpustimo tipkalo označeno oznakom „EN“ kako bi resetirati cijeli mikrokontroler te na taj način omogućili primjenu posljednjeg učitanog programskog koda, potom otpustimo tipkalo „Boot“ koje smo držali pritisnutim cijelim vremenom učitavanja tako da je na taj način završen proces upisivanja programskog koda na mikrokontroler ESP-32. USB (UART) adapter ima pristup kontroli tipkalima „EN“ i „Boot“ te nije potrebno ručno raditi prethodno opisani postupak za upisivanje programskog koda u memoriju mikrokontrolera.

4.1.8. Mikrokontroler L298N

Temeljna funkcija mikrokontrolera L298N je napajanje motora električnom energijom te određivanje smjerova i brzina vrtnje motora. Iz slike 4.9 je vidljivo da se mikrokontroler L298N sastoji od;

-H-mosta

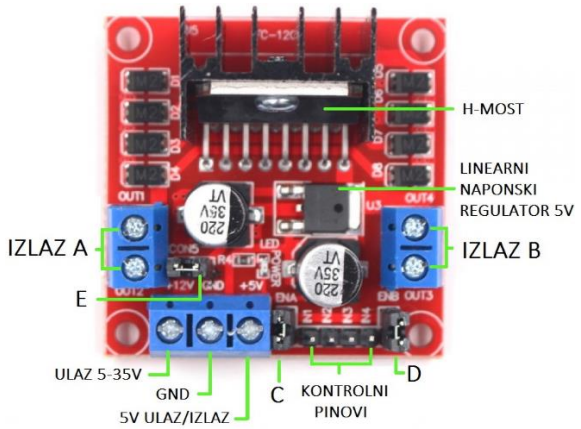
-pinova napajanja (5 V-36 V, GND, 5 V ULAZ/IZLAZ)

-osigurača (C , D, E)

-kontrolnih pinova (N1, N2, N3, N4)

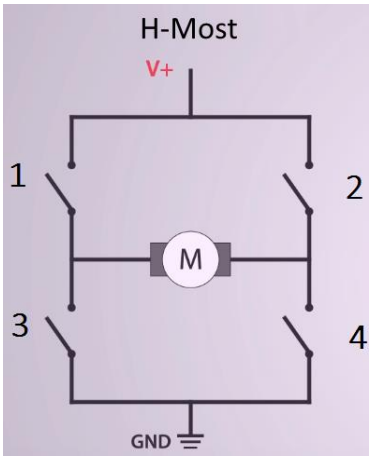
-naponskog regulatora

-izlaza (A i B)



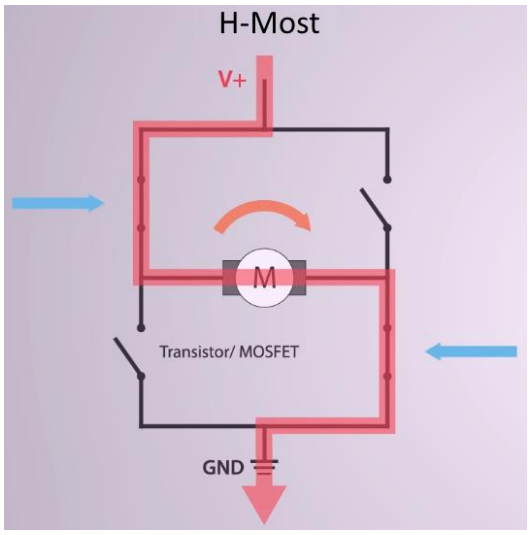
Slika 4.9. Komponente mikroprocesora L298N [18]

Temeljna funkcija H-mosta je promjena smjera toka električne struje što automatski za sobom vuče promjenu smjera vrtnje električnih motora. Građen od dviju stezaljki Vcc i Ground koji omogućuju napajanje, odnosno tok električne struje koji želimo mijenjati pomoću četiri sklopke koje ovisno o željenom smjeru električne struje mogu biti otvorene ili zatvorene različitim kombinacijama. Kada je sklopka otvorena iznos električnog otpora teži u beskonačnost te struja kroz tu granu neće teći, dok u slučaju kada je sklopka zatvorena električni otpor teži ka nuli te će struja teći ovo granom. Za primjer uzmemo da su sve sklopke otvorene.



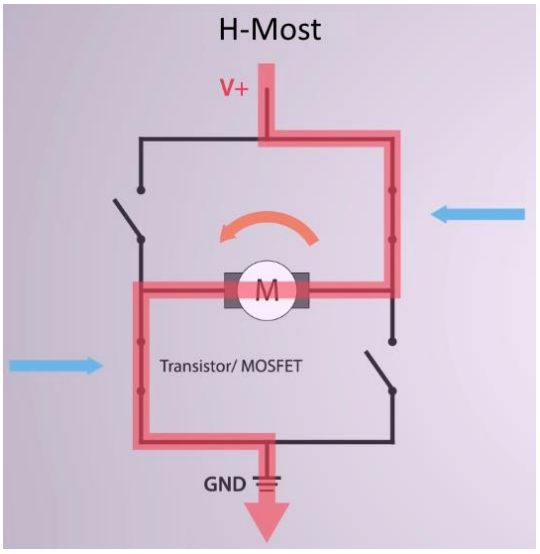
Slika 4.10. Prikaz H-mosta, sve stezaljke otvorene [18]

Budući da otpor svake pojedine sklopke teži u beskonačnost strujni krug nije zatvoren te električna energija uopće neće doći do stezaljki motora stoga će motor biti u stanju mirovanja. Ako zatvorimo sklopke označene brojevima 1 i 4, a preostale sklopke ostavimo otvorene sklopkama jedan i četiri će se zatvoriti strujni krug u smjeru prikazanom na slici 4.11.



Slika 4.11 H-most, stezaljke 1 i 4 zatvorene [18]

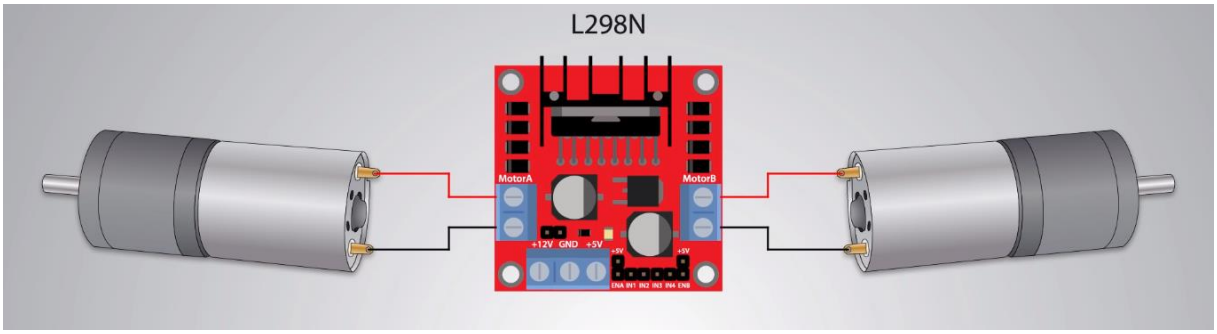
Međutim ako zatvorimo sklopke dva i tri, a preostale sklopke ostavimo otvorene, to jest sklopke jedan i četiri, kroz sklopke dva i tri će se zatvoriti strujni krug sa smjerom toka električne struje prikazanim na slici 4.12.



Slika 4.12 H-most, stezaljke 2 i 3 zatvorene [18]

Ako usporedimo smjer električne struje kroz stezaljke motora iz slika 4.11 i 4.12 vidljivo je da će se za prethodno opisane dvije kombinacije otvaranja odnosno zatvaranja sklopki smjer električne struje kroz stezaljke motora odnosno kroz sam motor promijeniti (motor je na slici označen oznakom „M“), što će se manifestirati u promjeni smjera vrtnje izlazne osovine istosmjernog motora. Električnu energiju L298N mikrokontrolera potrebnu za normalno funkcioniranje i napajanje ostalih komponenti dovodimo preko ulaza na slici 4.9 označenog s 5-35V što znači da su vrijednosti napona koje su predefinirane za ovaj ulaz vrijednosti u intervalu od 5-35 V. Ako će se na ovaj ulaz dovoditi naponi veći od 12 V potrebno je

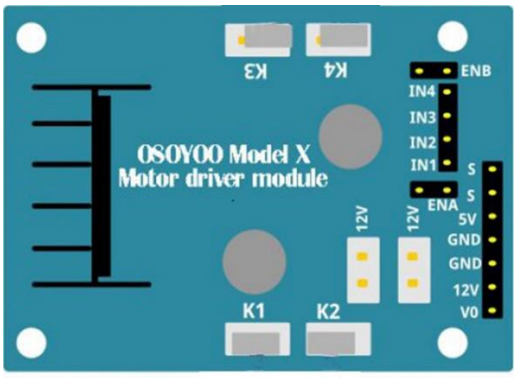
iskopčati osigurač na slici 4.9 označen s oznakom E. Desno od ulazne stezaljke nalazi se stezaljka „GND“ koja služi za zatvaranje ulaznog kruga s ulaznom stezaljkom ali i s ulazno-izlaznom stezaljkom koja se prema slici 4.9 nalazi lijevo od GND stezaljke, a služi za davanje napona od 5 V u određeni strujni krug. Izlazne stezaljke na koje u našem slučaju spajamo motore su stezaljke na slici označene oznakama A i B. Jedna od dvije izlazne stezaljke koje zajedno čine par je stezaljka višeg potencijala dok druga stezaljka služi za zatvaranje strujnog kruga odnosno stezaljka je nižeg potencijala, za obije skupine izlaznih stezaljki. Ovisno o smjeru u kojem se vrti izlazna osovina istosmjernog motora i ove stezaljke mogu biti izlazne i ulazne, zbog jednostavnijeg objašnjavanja nazivat ćemo iz izlaznim stezaljkama jer daju električnu energiju motorima.



Slika 4.11. Spajanje motora na mikroprocesor L298N [18]

Temeljna funkcija kontrolnih pinova je kontroliranje brzine i smjera vrtnje motora A i B pomoću PWM signala kojeg generiramo pomoću mikrokontrolera ESP-32. Detaljno objašnjenje PWM signala dano je u poglavlju 3.1. Osigurači označeni oznakama C i D na slici 4.9 automatski postavljaju brzinu vrtnje motora na maksimalnu moguću brzinu motora. Ako želimo postaviti brzinu vrtnje na određenu vrijednost koja nije maksimalna vrijednost potrebno je otkopčati ove osigurače te željeni iznos brzine vrtnje motora programirati u programskom kodu, što se u konačnici manifestira pomoću promjene trajanja perioda više odnosno niže energetske razine PWM signala.

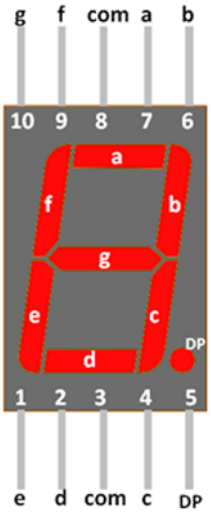
Mikrokontroler koji ćemo koristit u našem projektu je blaga modifikacija mikrokontrolera L298N, a on je L298N model X. Princip funkcioniranja ovog mikrokontrolera gotovo je identičan mikrokontroleru L298N, razlikuje se uglavnom u rasporedu pinova, te posjeduje dodatne pinove napajanja koji su nam dobrodošli, ali nisu i nužni. Mikrokontroler L298N je mnogo popularniji, odnosno višestruko češće korišteniji mikrokontroler u sličnim primjenama od mikrokontrolera L298N model X te je stoga upravo ovaj mikrokontroler detaljno objašnjen, a ne mikrokontroler L298N model X.



Slika 4.12 Mikrokontroler L298N model X. [26]

4.1.9. Troznamenkasti 7-segmentni LED ekran

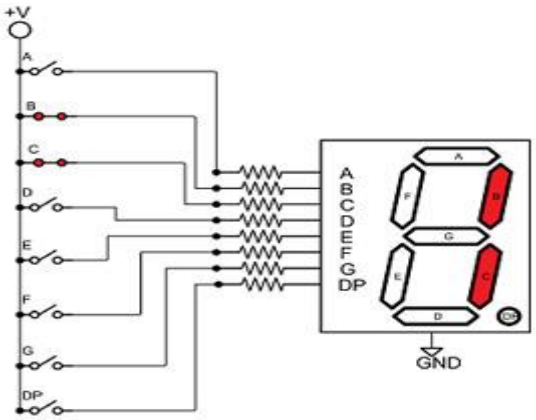
Temeljna funkcija ovog ekrana je prikazivanje iznosa napona izvora električne energije kojim napajamo robota, odnosno u našem slučaju ovaj ekran služi kao voltmetar. Ovisno o potrebi ovaj tip ekrana može prikazivati različite simbole koji se mogu jednoznačno prikazati pomoću 7 segmenta, u prvom planu predviđen je za prikaz arapskih brojeva u rasponu od 0 do 999 te shodno tomu osim funkcija voltmetra, ampermetra i vatmetra može služiti i kao brojač, indikator i slično. Segmenti jednog decimalnog mjesta ovog LED ekrana prikazani su na slici 4.13. i označeni su slovima; a, b, c, d, e, f i g.



Slika 4.13 Prikaz segmenata 7-segmentnog LED ekrana [19]

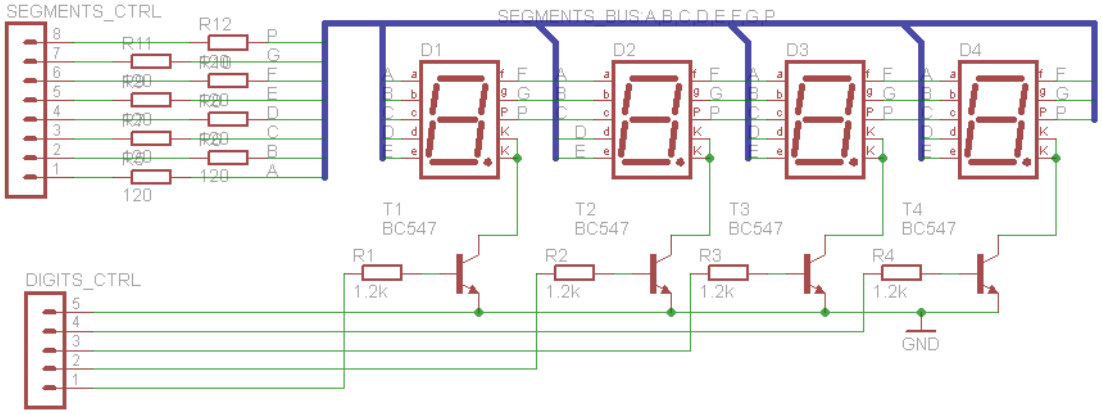
Svaki segment 7-segmentnog ekrana fizički je realiziran LED diodom što je karakteristika LED ekrana. Ako želimo aktivirati određeni segment, za primjer uzmimo da u jedno decimalno mjesto želimo upisati broj jedan u ovom slučaju to su segmenti „b“ i „c“ potrebno je aktivirati LED diode segmenata „b“ i „c“ na način da im osiguramo iznos napona jednak ili veći od napona koljena diode koji se ovisno o tipu diode kreće u intervalu od 0,2-0,7 V što

definira sam proizvođač. Napon koljena je napon koji nastaje zbog same konstrukcijske realizacije diode i uvijek je prisutan na diodi. Ako želimo zatvoriti strujni krug u kojem imamo samo istosmjerni idealni naponski izvor i diodu, potrebno je da naponski izvor ima veći iznos električnog napona propusne polarizacije od napona koljena date diode jer u protivnom za iznose napona propusne polarizacije koji su manji od iznosa napona koljena diode izvor će otpor diode vidjeti kao beskonačan te nećemo imati zatvoren električni strujni krug. Različitim kombinacijama aktivnosti odnosno neaktivnosti određenih LED dioda odnosno segmenta moguće je generirati 10 različitih znamenki u našem slučaju to su brojevi od 0 do 9 za svaki pojedini 7-segmentni ekran. Detaljna shema spajanja prikazana je na slici 4.14.



Slika 4.14. Električna shema realizacija broja 1 u 7-segmentnom LED ekranu [20]

Iz sheme je vidljivo da je za svaku znamenka uz signale za sedam segmenata potrebno dovesti napajanje i signal za prikaz decimalni točke. Svaka prikazana linija se programira u odabranom programskom okruženju, a budući da zbog podudaranja istih vrijednosti pinova na svakom od 3 LED ekrana koristi se programiranje u matičnom obliku.



Slika 4.15. Električna shema četiri 7-segmentna LED ekrana [21]

Budući da definiranje svakog segmenta pomoću matrica odnosno petlja donosi sa sobom određenu dozu kompliciranja razumljivosti programskog koda, ovakvi tipovi jednoznamenkastih i višeznamenkastih LED ekrana dolaze integrirani s mikroprocesorom koji

uvelike pojednostavljenje količinu i složenost programskog koda potrebnog za programiranje ovakvih i sličnih tipova ekrana.

4.1.10. Baterije

Prilikom izbora tipa bakterije važno je obratiti pažnju na sljedeći parametre;

-kapacitet

-voltaža

-maksimalna struja pražnjenja

-mogućnost punjenja

Slika 4.16. 18650 Li-ion baterija [22]

Kapacitet električne energije baterije je pokazatelj koliko je električne energije pohranjeno unutar same baterije. Uobičajeno se označava jedinicom mAh (mili-Amper-sati) što znači ako je na samoj bateriji naznačen kapacitet električne energije od 2.600 mAh to znači da baterija može dati konstantnu struju jakosti 2.600 mA (mili-Ampera) kroz sat vremena. Veći kapacitet električne energije baterije omogućava istom sustavu da funkcionira duži vremenski period, tako da ovisno o primjeni potrebno je odabrati optimalnu vrijednost uskladištene električne energije u samoj bateriji odnosno kapacitet električne energije baterije. Svaki uređaj (npr. električni motor, prijemnik, odašiljač i sl.) pri početku svog rada, odnosno dok ne postigne stabilne uvjete za aktivan rad, vuče struju višestruko veću od one koja je od strane proizvođača deklarirana za aktivan rad pojedinog uređaja ili pak neki uređaji, najčešće su to motori, zahtijevaju višestruko veću jakost električne struje od konstantne struje pražnjenja baterije kroz jedan sat, stoga je važno poznavati C parametar pražnjenja baterije (eng. C-rating). Ako je iznos parametra C za određenu bateriju npr. C=2, to znači da baterija posjeduje mogućnost davanja dvostruko veće jakosti električne struje od struje kojom punimo bateriju. Za naš slučaj za bateriju od 2.600 mAh uobičajeno punimo strujom od 2,6 A jedan sat vremena, zbog vrijednosti parametra C koji iznosi 2 moguće je pražnjenje baterije strujom jakosti 5.200 mA odnosno 5,2 A. Ako je vrijednost parametra C=3, bateriju ćemo moći prazniti jakošću struje od 7.800 mA odnosno 7,8 A odnosno strujom trostruko većom do struje kojom punimo bateriju. Ovisno o zahtjevima motora koje navodi proizvođač odabirom bateriji s adekvatnim iznosom parametra C moći ćemo koristiti određeni motor što ne bi bio

slučaj za baterije s nedostatnim iznosom C parametra. Formule za proračun električne energije, snage istosmjerne struje te ostale nama bitne formule su prikazane u nastavku.

$$W = UIT \tag{4.4}$$

$$P = \frac{UI}{T} \tag{4.5}$$

$$C_1 = C_2 \tag{4.6}$$

$$C_1 = I_1\ T_1 \tag{4.7}$$

$$C_2 = I_2\ T_2 \tag{4.8}$$

$$I_1\ T_1 = I_2\ T_2 \tag{4.9}$$

C je iznos električnog kapaciteta pohranjenog u bateriji izraženog u mili-Amper-satima za naš slučaj *C* je konstantan, *I* je jakost struje pražnjenja koja nije stalna veličina i ovisi o tome što se nalazi u strujnom krugu, *U* je napon koji je također konstantan, *T* je vrijeme koje ovisno o struji pražnjenja baterije konstantnog iznosa uskladištene energije može biti duže odnosno kraće. Prema relaciji 4.9, ako pretpostavimo da je iznos električnog kapaciteta pohranjenog u bateriji 2 600 mAh, a istu bateriju praznimo jakošću struje od 5.200 mA, a budući da su *C* i *U* konstantne vrijednosti da bi se održala jednakost lijeve i desne strane relacije 4.9 za slučaj kada se jakost struje dvostruko poveća vrijeme pražnjenja baterije se dvostruko smanji, te će od deklariranih jedan sat za jakost struje pražnjenja 2,6 A za iznos pražnjenja jakosti struje od 5,2 A vrijeme pražnjenja bit će dvostruko manje to jest 30 minuta. Ista stvar vrijedi i ako se baterija prazni dvostruko manjom jakosti električne struje pražnjenja od predviđenog, baterija će moći podnijeti ovakvu struju pražnjenja dvostruko duži vremenski period od predviđenog. Prethodno objašnjeno direktno je izvučeno iz zakona očuvanja energije koji kaže da se energija ne može uništiti niti ni iz čega stvoriti samo može prijeći iz jednog oblika u drugi što se u našem slučaju manifestira kao prijelaz veće količine električne energije u koristan rad kraći vremenski period ili pretvorba manje količine električne energije u koristan rad duži vremenski period.

Ovisno o specifikacijama određenog uređaja kojeg želimo pokretati, a koje definira sam proizvođač važno je izabrati adekvatnu voltažu električnog izvora to jeste baterija koje želimo koristiti. Standard jedne ćelije za napon predstavljen je intervalom od 3,7-4,2 V. Za minimalnu vrijednost napona jedne ćelije dogovorno je uzet iznos od 3,7 V dok je za maksimalnu vrijednost napona jedne ćelije dogovorno uzet iznos napona od 4,2 V. Standard jedne ćelije napona od 3,7 volta do 4,2 volta označava se oznakom „S“. Ako proizvođač kao napon uređaja navodi vrijednost 3S to znači da je potreban napon u intervalu od minimalno tri puta veći od 3,7 V odnosno do maksimalnog tri puta većeg 4,2 V što je interval od 11,1-12,6 V. Ukoliko se radi o napajanjima koji nisu višekratnici ovih standardnih napona , vrijednost napona moguće je prilagoditi transformatorima napona.

Ako sam proces zahtjeva višestruko i relativno učestalo pražnjenje baterija poželjno je koristiti punjive baterije, to jest baterije koje se nakon pražnjenja pomoću adekvatnog punjača

mogu puniti što je puno isplativije u financijskom i ekološkom smislu od kupnje jednokratnih baterija koje je prilikom ispražnjenja potrebno adekvatno reciklirati te kupiti nove. Postoje razni tipovi punjivih baterija koji se osim po svom kemijskom sastavu razlikuju po voltaži, kapacitetu, amperaži, mogućem broju punjenja i pražnjenja. Najčešći tipovi punjivih baterija su Litij-Ionske koje imaju nekoliko različitih izvedbi, a najčešće upotrebljavani tipovi su ICR, IMR i IFR Litij-Ionske baterije čije su specifikacije prikazane u tablici 4.1.

Tablica 4.1 Usporedba najpopularnijih tipova punjivih baterija [23]

Karakteristike	Olovna kiselina	NiCd	NiMH	Li-ion[1] Cobalt	Li-ion[1] Mangan	Li-ion[1] Fosfat
Specifična energija (Wh/kg)	30–50	45–80	60–120	150–250	100–150	90–120
Unutarnji otpor	Vrlo nizak	Vrlo nizak	Nizak	Umjeren	Nizak	Vrlo nizak
Broj punjenja	200–300	1,000[3]	300–500[3]	500–1,000	500–1,000	1,000–2,000
Vrijeme punjenja	8–16h	1–2h	2–4h	2–4h	1–2h	1–2h
Tolerancija na prenapon	Visoka	Umjerena	Niska	Niska		
Samopražnjenje/ mijesec	5%	20%[5]	30%[5]	<5%		
Napon ćelije	2V	1.2V[6]	1.2V[6]	3.6V[7]	3.7V[7]	3.2–3.3V
Granični napon punjenja (V/ćelija)	2.40	Samoograničen		4.20		3.60
Granični napon pražnjenja (V/ćelija)	1.75V	1.00V		2.50–3.00V		2.50V
Maksimalna struja pražnjena	5C[8] 0.2C	20C 1C	5C 0.5C	2C <1C	>30C <10C	>30C <10C
Temperatura punjenja	–20 do 50°C (–4 do 122°F)	0 do 45°C (32 do 113°F)		0 do 45°C[9] (32 do 113°F)		
Temperatura pražnjenja	–20 do 50°C (–4 do °F)	–20 do 65°C (–4 do 49°F)		–20 do 60°C (–4 do 140°F)		
Zahtjevi održavanja	3–6 mjeseci max. napuniti	Ako su u stalnoj uporabi svaki 90 dana ispazniti		Slobodno		
Sigurnosne značajke	Termo-stabilne	Termo-stabilne, zaštićene osiguračem		Zaštita kruga obavezna		
U upotrebi od	1800-te	1950	1990	1991	1996	1999
Toksičnost	Vrlo visoka	Vrlo visoka	Niska	Niska		
Cijena	Niska	Umjerena		Visoka		

Za ovaj projekt ćemo koristiti Litij-Ionske 18650 ICR baterije. Oznaka 18650 ima sljedeće značenje;

-širina 18 mm

-dužina 65 mm

-0 označava da se radi o cilindričnom obliku baterije

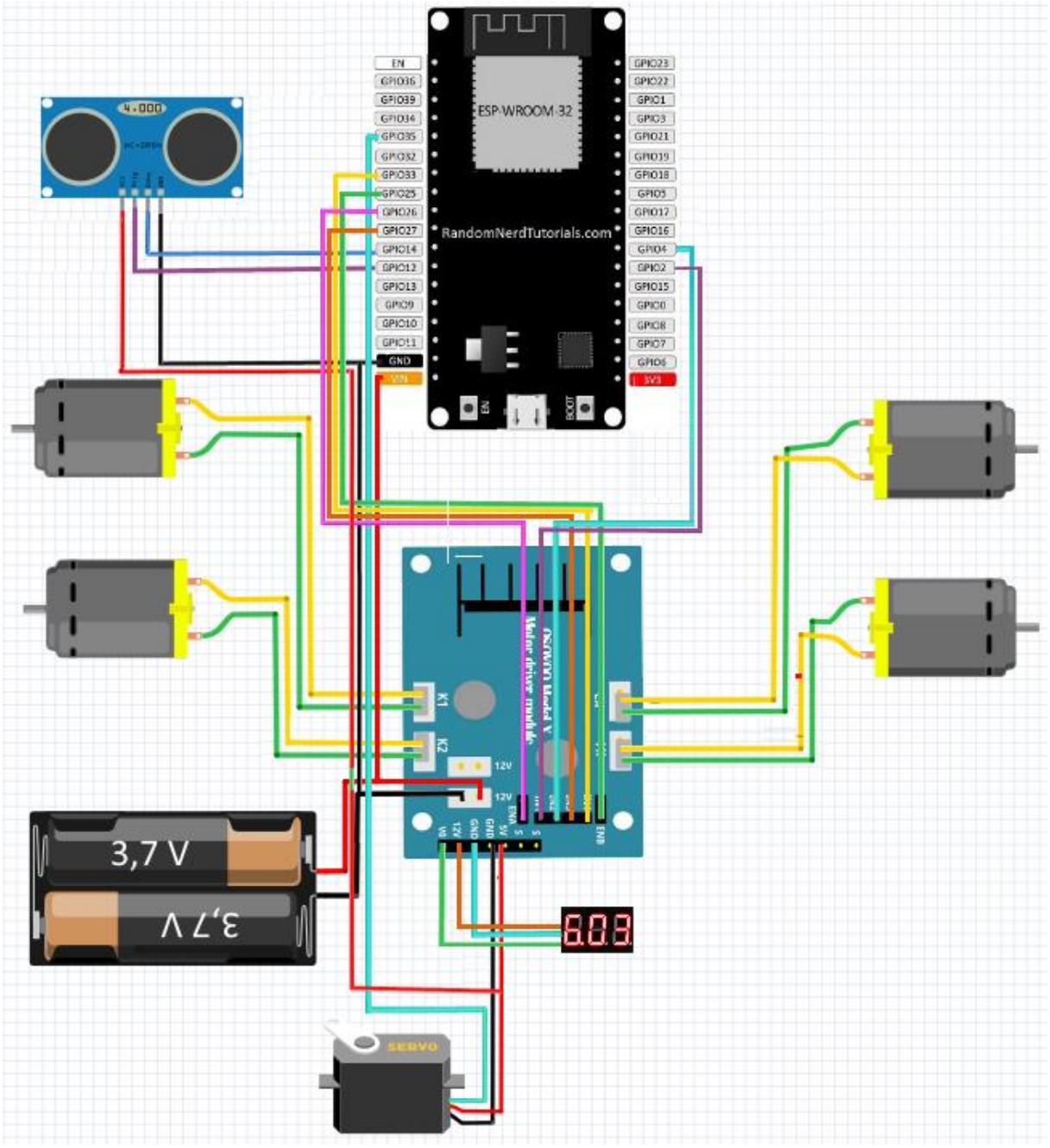
Oznaka ICR označava kemijski sastav Litij-Ionske baterije, primjeri su dani u tablici 4.2.

Tablica 4.2. Objašnjenja oznaka kemijskih sastava 18650 Litij-Ionskih baterija [24]

Tip	Kemijski sastav
ICR	LiCoO2
IMR	LiMn
IFR	LiFePO4

4.2. Spajanje komponenti

Prikaz fizičkog spajanja komponenti prikazan je na slici 4.17.



Slika 4.17 Prikaz fizičkog spajanja komponenti. [16][26]

4.3. Programsko okruženje

Kao programsko okruženje za programiranje ESP-32 mikrokontrolera koristit ćemo softver Atom i programski paket Platformio IDE s kojim dobivamo mogućnost programiranja u

Arduino programskom jeziku. Kao programsko okruženje moguće je koristiti i službeni softver koji se besplatno može preuzeti na službenoj stranici Arduino-a. Kao i Arduino, Atom programsko okruženje je potpuno besplatno za preuzimanje, u odnosu na Arduino programsko okruženje omogućuje brži pristup određenim direktorijima, poboljšani način instaliranja datoteka, konfiguriranja pločica, učinkovitiji način dijeljenja programskog koda, bolju preglednost programskog koda i slično.

4.4. Osnovne naredbe u Arduino programskom jeziku

U nastavku ćemo objasniti osnovne principe i naredbe u Arduino programskom jeziku, stoga počnimo.

Svaki Arduino programski kod mora sadržavati dvije petlje, a to su;

- `void setup(){}`
- `void loop(){}`

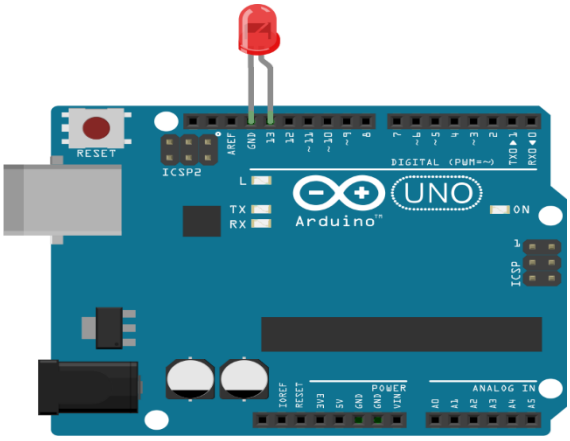
`void` - predstavlja tip petlje.

`void setup()` - je petlja koja se izvršava samo jednom i to na početku izvođenja programa te kao takva služi za postavljanje početnih uvjeta sustava.

`void loop()` - je petlja koja se izvršava beskonačno mnogo puta te predstavlja petlju u koju se upisuje glavni programski kod kojim ćemo moći upravljati sustavom u realnom vremenu.

Programski Arduino kod mora sadržavati ove dvije petlje uprotvinom kompajler će javiti grešku i implementacija Vašeg programskog koda na mikrokontroler će biti nemoguća.

Primjer jednostavnog programskog koda koji pali i gasi LED kao i njegova fizička realizacija prikazana je u nastavku.



Slika 4.18. Fizička realizacija programskog koda [25]

```
void setup() {
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

-pinMode(pin, MODE)- je naredba koje prima dva argumenta. Prvi argument je broj pina na koji želimo djelovati dok je drugi argument način rada ovog pina odnosno da li će slati struju u strujni krug odnosno OUTPUT ili će primati struju iz strujnog kruga odnosno INPUT.

-digitalWrite(pin, value) - je naredba koje prima dva argumenta. Prvi argument je broj pina na koji želimo djelovati dok je drugi argument način rada ovog pina odnosno da li će odabrani pin biti u višoj HIGH ili nižoj LOW naponskoj razini.

-delay(ms) - je naredba koja prima jedan argument, odnosno brojčanu vrijednost u milisekundama (ms) za koju će izvođenje programa biti pauzirano.

4.5. Optimizacija programskog koda

Prilikom kompajliranja programskog koda Platformio IDE nas obavještava o veličini kompajliranoga koda. Budući da je veličina memorije kojom raspolažu mikrokontroleri nekoliko tisuća puta manja od količine memorije kojom raspolažu suvremena računala, vrlo je moguće da programski kod prekorači određena memorijska ograničenja koja su pred njega postavljena. Kako bi uspjeli programski kod „ugurati“ u određena memorijska ograničenja potrebno je uz način rada pojedinog dijela memorije (objašnjeno u poglavlju 2.2.1) poznavati i načine optimizacije programskog koda. U nastavku ćemo nabrojati i objasniti neke od metoda optimizacije programskog koda.

4.5.1. Uklanjanje mrtvih dijelova koda

Ukoliko se programski kod sastoji od biblioteka koji su uključene u programski kod, a koje u programskom kodu nemaju korisnu funkciju potrebno ih je ukloniti iz programskog koda te na taj način osloboditi određenu količinu memorije u koju po potrebi možemo upisati korisne naredbe. Isto vrijedi i za nedostupne dijelove koda.

Pri izvođenju željenog procesa važnu ulogu ima količina slobodne memorije pomoću koje se vrši alociranje i dealociranje memorije koju zauzimaju funkcije pomoću kojih želimo izvršiti

određenu zadaću stoga ćemo navesti i objasniti nekoliko metoda optimizacije SRAM memorije.

4.5.2. Odabiranje adekvatnog tipa varijable

Potrebno je koristiti tip varijable koji najbliže odgovara našim zahtjevima. Za primjer uzmimo senzor temperature koji kroz određeno vrijeme treba poslati iznos izmjerene temperature. Prilikom definiranja tipa podatka u kojem će biti pohranjena vrijednost koju šalje senzor potrebno je odrediti primjenu podatka poslanog od strane senzora, odnosno da li je dovoljna cjelobrojna vrijednost ili primjena našeg sustava zahtjeva višestruko preciznije podatke odnosno podatke s decimalnim vrijednostima. Iz tablice 4.3. je vidljivo da tipovi podataka koji imaju veću preciznost zauzimaju višestruko veću količinu memorije odnosno, količina memorije koju zauzima cjelobrojni tip podataka bez predznaka „uint8_t“ (eng. Unsigned Integer) iznosi 1 By (eng. Byte) odnosno 8 B (eng. Bit) odnosno omogućava pohranu 256 različitih vrijednosti, u odnosu na tip podatka koji omogućuje zapisivanje decimalnih vrijednosti (eng. Float) koji zauzima 4 By odnosno omogućuje pohranu 4 294 967 296 različitih vrijednosti, što je ogromna razlika u zauzetoj količini memorije za najobičniji unos podataka s decimalnom vrijednošću. Prikaz dužine memorijskih adresa prikazan je u tablici 4.3.

Tablica 4.3. Količina memorije koju zauzimaju određeni tipovi podataka [1]

Tip podatka	Veličina u Bajtovima	Može sadržavati:
boolean	1	Istina (1) ili laž (0)
char	1	ASCII simbol ili vrijednsti od -128 do 127
unsigned char, byte, uint8_t	1	ASCII simbol ili vrijednosti od 0 do 255
int, short	2	Vrijednosti od -32 768 do 32 767
unsigned int, word, uint16_t	2	Vrijednosti od 0 do 65 535
long	4	Vrijednoti između -2,147,483,648 i 2,147,483,647
unsigned long, uint32_t	4	Vrijednosti između 0 i 4,294,967,295
float, double	4	Decimalne vrijdnosti između 3.4028235E+38 i 3.4028235E+38 (NAPOMENA) float i double tipovi podataka na ovoj platformi su istoznačni

Upravo iz ovog razloga uštede memorije, Arduino programski jezik posjeduje posebno prilagođene tipove podataka koji omogućuju pohranu samo pozitivnih vrijednosti što na primjenu nailazi pri mjerenju određenih veličina koje ne mogu biti negativne kao što su vrijeme, vlažnost i slične veličine ili pak određenih veličina koje mogu ali neće biti negativne na primjer mjerenje temperature u pustinji tijekom dana što daje mogućnost pohrane dvostruko većeg broja vrijednosti varijabli. Za primjer uzmimo „int“ tip podatka (eng. Integer) koji omogućava pohranu vrijednosti u intervalu od -32 768 do +32 767 što predstavlja neefikasno korištenje memorije u slučaju kada nam trebaju samo pozitivne vrijednosti iz razloga što 32 768 mjesta ne možemo koristiti jer omogućuju samo pohranu negativnih vrijednosti te je u svrhu optimizacije potrošnje memorije uveden je tip podatka uint16_t (eng. Unsigned Integer) koji zauzima također dva bajta memorije ali omogućuje zapisivanje 65 536 različitih pozitivnih vrijednosti što je ogroman napredak u pohranjivanju podataka čija vrijednost neće mijenjati predznak u negativan.

4.5.3. Upotreba lokalnih varijabli

Iz same konstrukcije memorija mikrokontrolera objašnjenih u poglavlju 2.2.1 saznali smo da su različite globalne i statičke varijable one koje se prvo učitavaju u SRAM memoriju te zauzmu količinu memorije koja ostaje nepromjenljiva u vremenu do sljedećeg programiranja mikrokontrolera, stoga je važno pohraniti što je manji mogući broj globalnih i statičkih varijabli unutar memorije statičkih podataka SRAM memorije, jer se memorija statičkih podataka za svaku dodatnu globalnu ili statičku varijablu povećava, te svojim povećavanjem automatski smanjuje iznos radne memorije dostupne gomili i stogu koji imaju mogućnost promjene iznosa zauzete memorije u vremenu, to jest omogućuju manevriranje u memoriji tijekom vremena što u današnjoj tehnologiji predstavlja ogroman napredak u načinu korištenja memorije što rezultira brzim izvođenjem zadaće mikrokontrolera.

4.5.4. Upotreba programske memorije za pohranu podataka

Budući da je SRAM odnosno radna memorija vrlo dragocjena memorija, uvijek ju nastojimo što je više moguće učinkovitije koristiti, stoga je potrebno premjestiti podatke koji se ponavljaju u vremenu s nepromijenjenom vrijednošću iz SRAM memorije u dio memorije predviđen za konstantne podatke, a to je programska memorija. Za primjer uzmimo tekst; „Temperatura“, koji se nalazi unutar određene funkcije te je samim tim automatski pohranjen u stog memoriju te se pojavljuje nakon određenog vremenskog perioda potpuno identičan, i možemo ga smatrati konstantnom varijablom te premjestiti ga u dio memorije pogodniji za ovakve vrijednosti, a to je programska memorija, te na taj način osloboditi određeni udio SRAM memorije. Oslobađanje SRAM memorije odvija se na način da se oni podaci koji će se ponavljati umjesto u SRAM memorije pohrane u programsku memoriju, a ovaj postupak omogućuje makro naredba F( ) koja se koristi na način da unutar zagrada F-makroa postavimo podatak koji želimo premjestiti iz SRAM memorije u programsku memoriju na način prikazan u nastavku.

```
Serial.println(F("Temperatura"));
```

U ovom primjeru tekst; „Temperatura“, prebačen je iz SRAM memorije u programsku memoriju.

4.5.5. Izbjegavanje korištenja rekurzivnih funkcija

Rekurzivna funkcija je ona funkcija koja poziva samu sebe. Budući da je funkcija rekurzivna jer poziva samu sebe prilikom pozivanja same sebe ponovno poziva funkciju koja poziva samu sebe i tako u nedogled, te na taj način vrši zauzimanje memorijskih lokacija ponovno i ponovno te bi se memorija zauzimala u nedogled da ne postoje fizička ograničenja, odnosno konačni iznosi memorije kojom raspolažu mikrokontroleri. Ovisno o načinu na koji se mikrokontroler ponaša kada se prekorače određena dinamička memorijska ograničenja doći će do odstupanja od uobičajenog načina rada mikrokontrolera što će vrlo vjerojatno imati nepoželjan učinak na željeni proces. Arduino programski jezik pomoću algoritma mikroprocesorskih naredbi omogućuje očitavanje dijela zauzetosti programske memorije, SRAM memorije te EEPROM memorije kako bi u svakom trenutku imali predodžba o preostaloj slobodnoj količini memorije te eventualno poznavali uzrok neobičnog ponašanja elektroničkog sustava.

4.6. Izrada Android aplikacije za upravljanje robotom

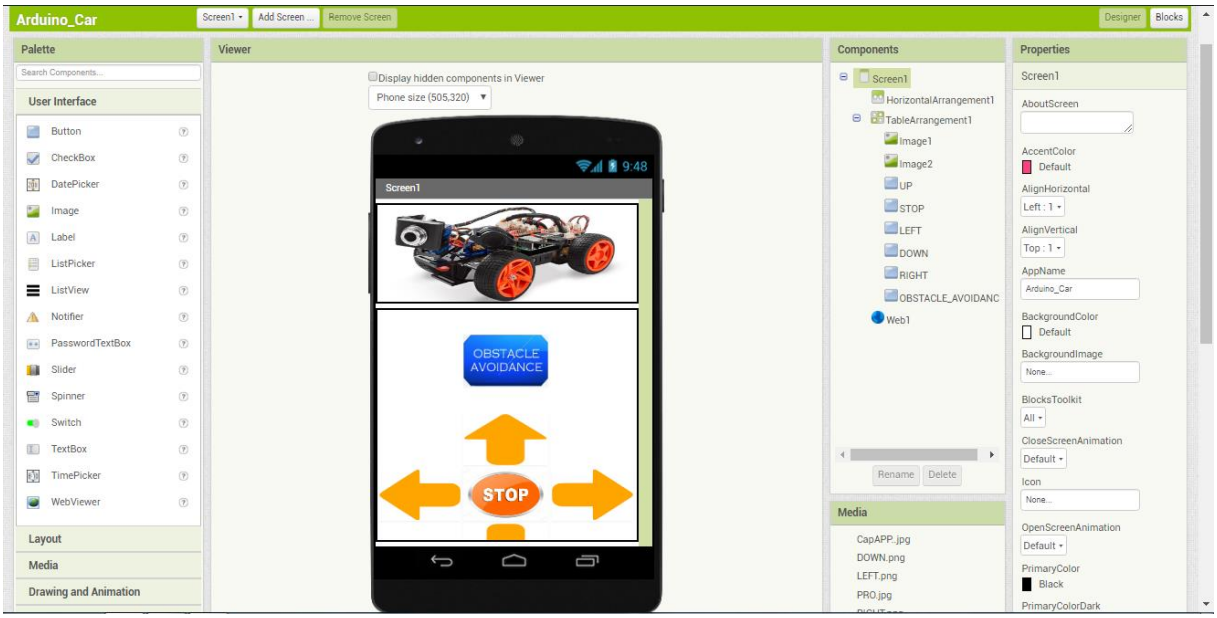
Aplikaciju u potpunosti izrađujemo pomoću programskog okruženja „MIT App Invertor 2“, koji je u potpunosti besplatan za preuzimanje, vrlo moćan i jednostavan za korištenje stoga nećemo ići predetaljno u objašnjavanje samog postupka. Izrada aplikacije se provodi u dva koraka, a to su;

-izrada korisničkog sučelja

-blokovsko programiranje korisničkog sučelja

4.6.1. Izrada programskog sučelja

Korisničko sučelje se izrađuje pomoću povlačenja komponenti koje su nam potrebne iz izbornika „User Interface“ koji se nalazi na lijevoj strani slike 4.19, a to su našem slučaju tipkala kojima potom pridajemo različita svojstva u izborniku „Properties“ koji se nalazi na desnoj strani slike 4.19, a to su razna svojstva koja omogućavaju mijenjanje dimenzija, položaja, boje, teksta, oblikovanje tipkala u obliku slike po našoj želji i još dosta vrlo korisnih svojstava.



Slika 4.19. Izrada korisničkog sučelja pomoću aplikacije MIT App Invertor 2

4.6.2. Blokovsko programiranje korisničkog sučelja

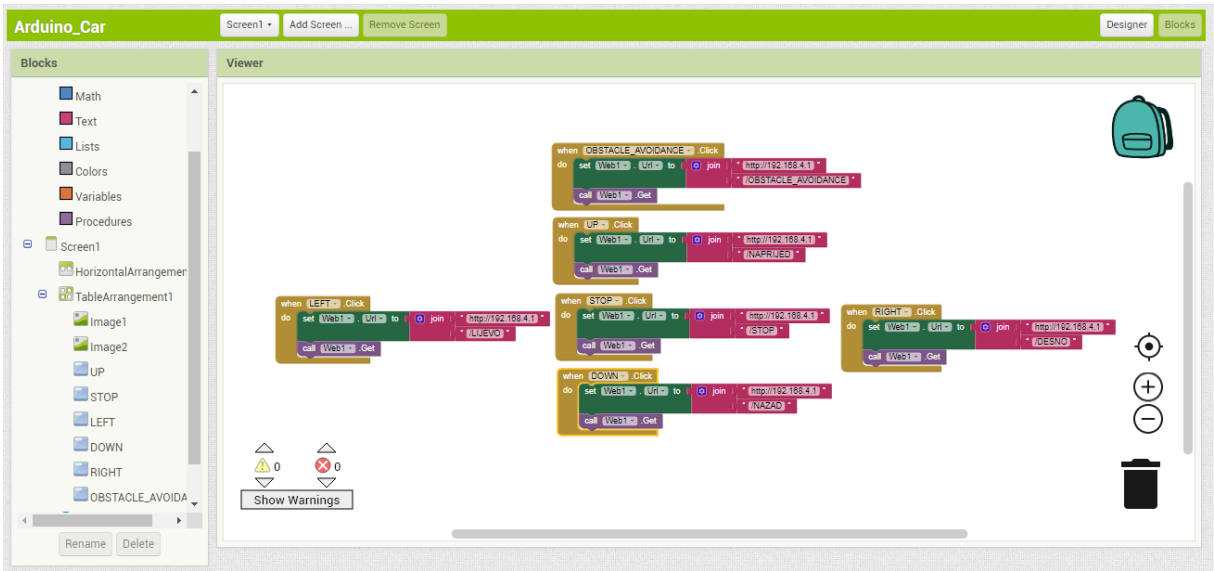
Blokovsko programiranje korisničkog sučelja predstavlja pridavanje funkcija tipkalima koje smo dodali na korisničko sučelje.

Primjer programiranja tipkala „STOP“ prikazan je na slici 4.20, a sva ostala tipkala slijede u potpunosti isti princip.

Slika 4.20. Programiranje tipkala „STOP“

Programski blok ćemo tumačiti odozgo prema dolje te paralelno s lijeva na desno, tumačenje pseudo-jezikom je sljedeće.

Kada pritisnem tipkalo STOP uradi to da postaviš Url Web pretraživača na vrijednost „http://192.168.4.1/STOP“, te da ga dohvatiš. Dati link smo sami generirali prema našoj želji što je vidljivo u programskom kodu.

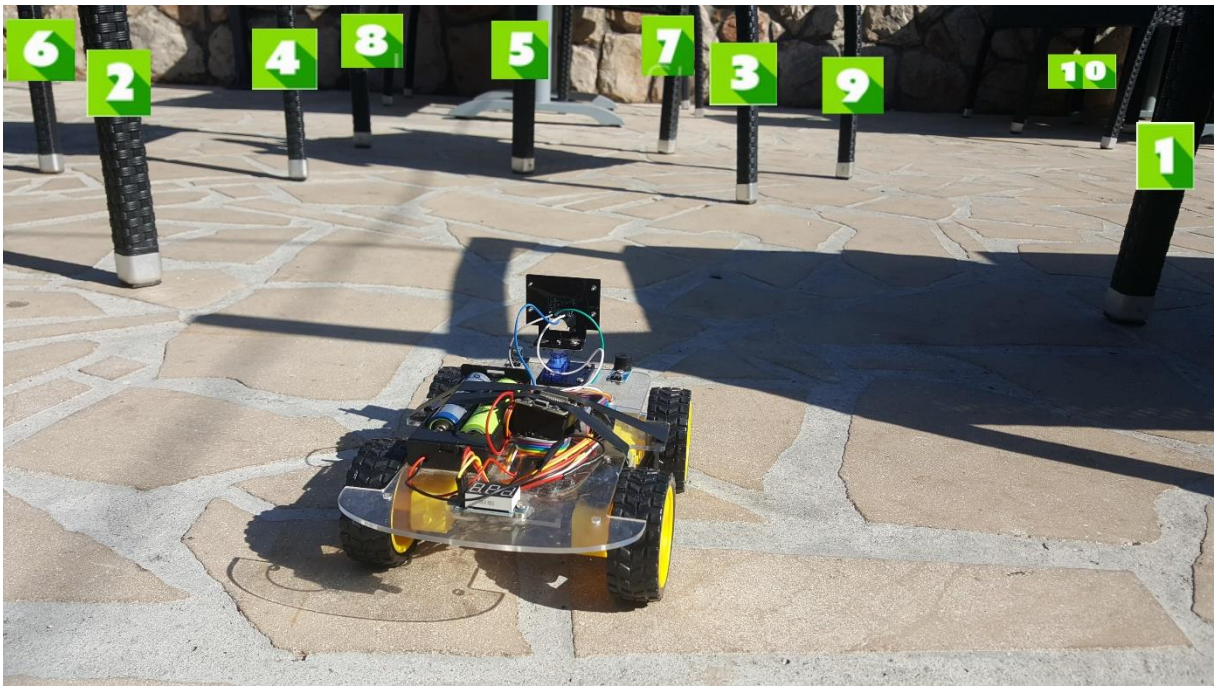


Slika 4.21. Kompletni programski kod android aplikacije.

Kreiranu aplikaciju možete vrlo jednostavno preuzeti skeniranjem generiranog programskog koda na bilo koji pametni uređaj na kojem se može instalirati program „MIT AI2 Companion“. Ova aplikacija je podržana od strane IOS, Windows te Android operativnog sustava.

4.7. Test

Testiranje ćemo provesti na području iz našeg svakodnevnog okruženja, pritom ćemo paziti da dato područje nema naglih padova nadmorske visine (npr. stepenica i sl.) jer ovaj robot sa senzorikom koju trenutno posjeduje nema mogućnosti da prepozna opasnost takve prirode na vrijeme te može doći do uništenja robota. Također trebamo paziti da na površini na kojoj se vrši testiranje ne postoji vodena površina kao i sve ostale specifičnosti koje mogu općenito nauditi nekom elektroničkom sustavu. Testiranje će se provoditi unutar zatvorenih prostorija te na otvorenom prostoru kako bi uvidjeli utjecaj raznovrsnih signala i pojava u našoj okolini koje ne primjećujemo svakodnevno, ali koje ipak postoje.



Slika 4.22. Arduino robot u testnom okruženju [28]

Prilikom testiranje Wi-Fi konekcije zaključeno je za sve ostale Wi-Fi dostupne mreže potrebno isključiti opciju; „Automatski ponovno spoji“, jer u slučaj malo slabijeg Wi-Fi signala robota doći će do automatskog spajanja, na bilo koju drugu nama dostupnu mrežu, te robot ostaje bez mogućnosti kontrole.

Također je izmjereno kašnjenje od 0.7 sekundi na svaku pojedinu naredbu koja dolazi s pametnog uređaja kojim upravljamo robota. Kašnjenje je neophodno iz razloga što radio valovi (valovi općenito) trebaju izvjesno vrijeme da dođu od uređaja odašiljača do uređaja prijemnika, tj. ne dolaze trenutno, a tu je i proces odvijanja protokola, u našem slučaju Wi-Fi protokola, koji se također ne odvija trenutno odnosno za izvršavanje mu je potrebno određeno vrijeme. Pitanje je samo da li će kašnjenje biti veće odnosno manje. S ovim kašnjenjem robot funkcionira vrlo uspješno.

Testiranje načina rada autonomnog zaobilaženja prepreka dalo je sljedeće rezultate.

Tablica 4.4. Izmjereni rezultati testiranja autonomnog zaobilaženja prepreka

Redni broj ciklusa	Broj provjera po ciklusu	Broj uspješnih provjera	Broj neuspješnih provjera	Uspješnost po ciklusu u postotcima	Ukupna uspješnost u postotcima
1.	135	134	1	99 %	78,4 %
2.	135	112	23	83 %	
3.	135	101	34	75 %	
4.	135	95	40	70 %	
5.	135	88	47	65 %	

Dobiveni rezultati nisu posljedice nefunkcioniranja pojedinih komponenti robota, već posljedica tzv. „mrtvog kuta“, koji se više očituje pri zaobilaženju prepreka relativno manjih dimenzija, za naš slučaj to su prepreke čija površina posjeduje horizontalnu širinu 10 cm i manje. Također prepreke koje su smještene na nižem položaju od položaja ultrazvučnog senzora neće biti primijećena. Ovaj problemi se mogu vrlo jednostavno riješiti uvođenjem još jednog senzora koji će biti zadužen za „niske“ prepreke te programiranjem servo motora da pravi dodatna mjerenja na položajima trenutnih mrtvih kutova za koji ponavljam ovise o veličini i obliku prepreke te su prisutni uglavnom prisutni pri mjerenju relativno malih aktivnih površina date prepreke.

5. ZAKLJUČAK

Kroz stadije realizacije samog projekta dolazilo je do promjena percipiranja složenosti projekta. Na samom početku pokušaja realiziranja projekta prevladavalo je mišljenje kako je projekt iznimno složen, a to je prouzročeno dotadašnjim neznanjem o principima funkcioniranja elektroničkih komponenti kao i relativno siromašnom programerskom znanju. Nakon što je velika većina principa shvaćena projekt je se doimao vrlo jednostavnim za realizaciju, dok nije počela faza programiranja, koja je vratila veliku većinu stvari na početak. Prilikom programiranja korištena su tri softvera Arduino, Atom te Visual Studio Code. Svaki softver ima svoje prednosti i nedostatke za čije su shvaćanje trebali mjeseci. Također tu su i greške koju Vam javi kompajler na koju izgubite tri tjedna i shvatite da trebate instalirati drugo programsko okruženje, koje se ponovno zbog nekog banalnog razloga instalira cijela tri dana i ne nazire se kraj instaliranju, a vrijeme ide, a rokovi su tu. Nakon što su tri programska okruženja promijenjena te velika većina programskog koda realizirana, pojavljuju se komponente koje unatoč svim preinakama programskog koda te svim istraživanjima odbijaju da se ponašaju onako kako je definirano u programskom kodu i još jako mnogo „usputnih“ stvari koje Vam uzimaju enormno mnogo vremena. Sve u svemu neprocjenjivo pozitivno iskustvo.

Očekivanja od postupka realizacije projekta bila su;

-ogroman napredak u vještinama programiranja

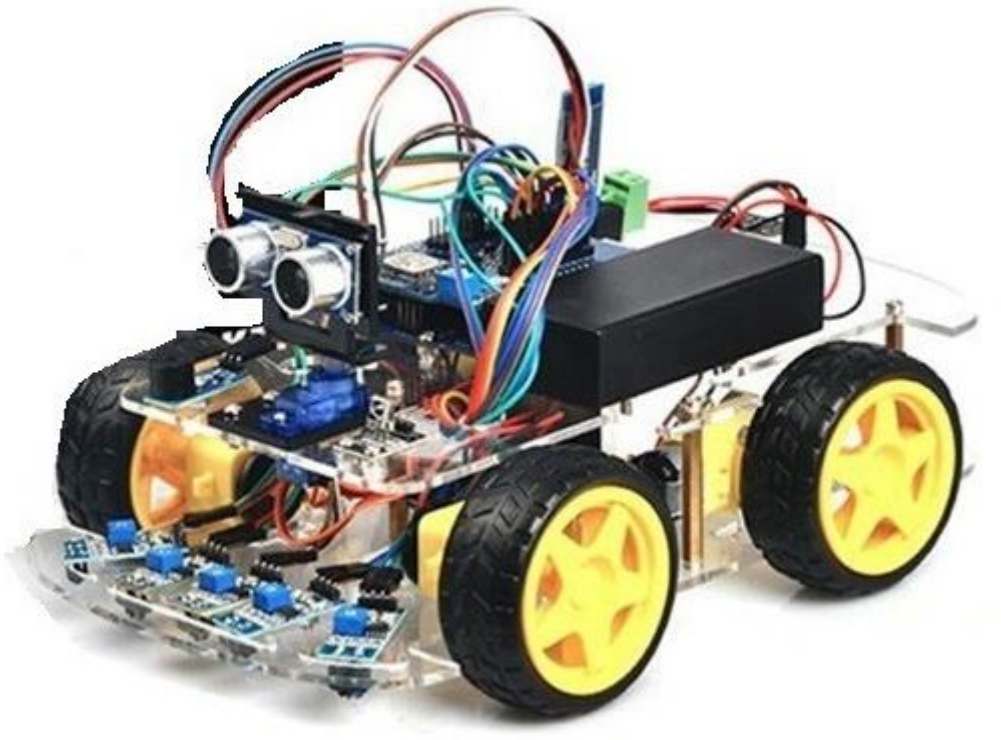
-napredak u shvaćanju principa funkcioniranja suvremenih elektroničkih komponenti

-napredak u istraživačkom pristupu rješavanja problema

Sva očekivanja su ispunjena u potpunosti. Na greškama se uči, stoga ako Vam je projekt uspio iz prvog pokušaja, to za sobom vuče određeni negativan utjecaj, jer ostajete uskraćeni za određeno iskustvo u snalaženju u situacijama u kojima ne ide sve onako kako Ste predvidjeli.

Konkretan projekt je najbolji način učenja onoga što Vas zanima i nadograđivanja željenih vještina, a najjednostavniji način realizacije konkretnih projekta je svakako Arduino elektronička platforma, koja uz vrlo povoljne cijene nudi mnoštvo dostupnih materijala i korisnih principa koji se primjenjuju u elektronici.

Zaključak testiranja je taj da na otvorenom prostoru postoje određena ometanja Wi-Fi signala kojim upravljamo našeg robota koja su djelomično prisutna zbog postojanja drugih signala koji djeluju u relativno uskom dozvoljenom frekvencijskom pojasu, za naš slučaj to je frekvencija 2,4 GHz, na kojoj osim našeg robota djeluju i velika većina svih Wi-Fi i Bluetooth protokola koje koristimo na našim mobilnim uređajima.



Slika 5.1. Arduino pametni robot [27]

SAŽETAK

Kao i što sam naziv kaže u ovom radu je detaljno objašnjen način funkcioniranja robota baziranog na Arduino platformi. Detaljno je objašnjena Arduino elektronička platforma kao i sve komponente koje čine ovaj robot. Također objašnjeni su svi signali i protokoli koji se koriste u realizaciji samog projekta. Također objašnjeno je i električno spajanje svih komponenti i senzora ovog robota kao i sam postupak izrade Android aplikacije pomoću kojeg ćemo upravljati našeg robota. Uz upravljanje robota Wi-Fi aplikacijom ovaj robot ima sposobnost samostalnog zaobilaženje prepreka. Uz sve navedeno unutar ovog rada se nalazi i detaljno objašnjen programski kod te određena opažanja koja smo primijetili prilikom realizacije ovog projekta.

KLJUČNE RIJEČI

-Arduino

-Wi-Fi

-ESP-32

-Vozilo

-Robot

TITLE

WIRELESS CONTROLLED ROBOT BASED ON THE ARDUINO PLATFORM

SUMMARY

Like the title says, the principle of working robot based on the Arduino platform is precisely explained in this document. Arduino electronics platform, all components which make this robot is precisely explained too. You have an explanation of all signals and protocols which we used in this document. Also, you have wiring schematic of all components, and all steps of production mobile Android app, which we will use for robot car Wi-Fi controlling. With Wi-Fi controlling ability this robot has obstacle avoidance ability. Programming code which we used in this project with an explanation and conclusion of project realization also stay in this document.

KEY WORDS

-Arduino

-Wi-Fi

-ESP-32

-Car

-Robot

POPIS LITERATURE

[1] Toni Perković; „Bežične senzorske mreže - Lab 2“, s Interneta, https://github.com/toperkov/WiSe-2018-19/blob/master/instructions/lab-2.md, 1. rujna 2019.

[2] Wikipedia; „USB hardware“, s Interneta, https://en.wikipedia.org/wiki/USB_hardware, 1. rujna 2019.

[3] Circuits DIY; „PULL UP and PULL DOWN resistor | basic electronics | Tutorial“, s Interneta, https://www.youtube.com/watch?v=m5v4nrEqp3s, 1. rujna 2019.

[4] Ahmed Elmahalawy; „PWM signal with its two basic time periods“, s Interneta, https://www.researchgate.net/figure/PWM-signal-with-its-two-basic-time-periods_fig4_271437313, 1. rujna 2019.

[5] ABelectronics; „Generating a PWM Signal“, s Interneta, https://www.abelectronics.co.uk/kb/article/1078/generating-a-pwm-signal, 1. rujna 2019.

[6] Solution ; „UART PROTOCOL VALIDATION SERVICE“, s Interneta, https://www.solitontech.com/uart-protocol-validation-service/, 1. rujna 2019.

[7] Mario Čagalj; „Sensor node architecture“, s Interneta, http://marjan.fesb.hr/~mcagalj/wise/#/predavanja, 1.rujna 2019.

[8] Augmented Startups; „Fun and Easy SPI - How the SPI Protocol Works“, s Interneta, https://www.youtube.com/watch?v=AuhFr88mjt0, 1.rujna 2019.

[9] How To Mechatronics; „How I2C Communication Works and How To Use It with Arduino“, s Interneta, https://www.youtube.com/watch?v=6IAkYpmA1DQ&t=312s, 1. rujna 2019.

[10] Darko Stipaničev i Jadranka Marasović; „ Analogno / digitalni (A/D) pretvarač “, s Interneta, http://laris.fesb.hr/digitalno_vodjenje/text_2-6.htm, 1. rujna 2019.

[11] Darko Stipaničev i Jadranka Marasović; „ Digitalno - analogni (D/A) pretvarač “, s Interneta, http://laris.fesb.hr/digitalno_vodjenje/text_2-11.htm, 1. rujna 2019.

[12] How To Mechatronics; „How Servo Motors Work & How To Control Servos using Arduino“, s Interneta, https://www.youtube.com/watch?v=LXURLvga8bQ, 1.rujna 2019.

[13] DYIelectronics; „TOWERPRO MICRO 9 GRAM HOBBY SERVO - SG90“, s Interneta, https://www.diyelectronics.co.za/store/servos/63-towerpro-micro-9-gram-hobby-servo-sg90.html, 1. rujna 2019.

[14] TheEngineeringProjects; „Introduction to HC-SR04 (Ultrasonic Sensor)“, s Interneta, https://www.theengineeringprojects.com/2018/10/introduction-to-hc-sr04-ultrasonic-sensor.html, 1. rujna 2019.

[15] Espressif; „Espressif Systems ESP32-DevKitC Development Boards“, s Interneta, https://au.mouser.com/new/espressif/espressif-esp32-devkitc-boards/, 1. rujna 2019.

[16] Random Nerds Tutorials; „ESP32 Pinout Reference: Which GPIO pins should you use?“, s Interneta, https://randomnerdtutorials.com/esp32-pinout-reference-gpios/, 1.rujna 2019.

[17] Future Electronics; „ESP32 Development Board (WIFI - Bluetooth)“, s Interneta“, https://store.fut-electronics.com/products/esp-32, 1.rujna 2019.

[18] Electronics Hobbyists; „Controlling DC Motors with Arduino | Arduino L298N Tutorial“, s Interneta, https://electronicshobbyists.com/controlling-dc-motors-arduino-arduino-l298n-tutorial/, 1. rujna 2019.

[19] Circuits 4 You; „4-digits 7-Segment Display Interfacing with AVR microcontroller“, s Interneta, https://blog.circuits4you.com/2015/06/4-digits-7-segment-display-interfacing.html, 1. rujna 2019.

[20] Jameco; „Working with Seven Segment LED Displays“, s Interneta, https://www.jameco.com/Jameco/workshop/TechTip/working-with-seven-segment-displays.html, 1.rujna 2019.

[21] Electronics Stack Exchange; „Multiplexing with or without transistors“, s Interneta, https://electronics.stackexchange.com/questions/69477/multiplexing-with-or-without-transistors, 1. rujna 2019.

[22] Lightorati; „Samsung 18650 2200mAh 3.7v Rechargeable Li-ion Battery (Flat Top, Unprotected)“, s Interneta, http://lightorati.in/samsung-18650-2200mah-flat-top-battery, 1. rujna 2019.

[23] Battery University; „BU-107: Comparison Table of Secondary Batteries“, s Interneta, https://batteryuniversity.com/learn/article/secondary_batteries, 1.rujna 2019.

[24] Candle Power Forums; „Lithium IMR vs ICR vs IFR“, s Interneta, https://www.candlepowerforums.com/vb/showthread.php?317698-Lithium-IMR-vs-ICR-vs-IFR, 1. rujna 2019.

[25] Johnny Five; „Basic LED example“, s Interneta, http://johnny-five.io/examples/led/, 1. rujna 2019.

[26] AliExpress; „OSOYOO Model-X Motor Driver Module Dual H Bridge Stepper Motor Driver Board for Arduino Smart Robot Car 5-35V Drive voltage“, s Interneta, https://www.aliexpress.com/item/32900808914.html, 1. rujna 2019.

[27] Deal Extreme; „Smart Robot Car Kit with Four-Wheel Drives for Arduino UNO R3 Project, Christmas Gifts“, s Interneta, https://www.dx.com/p/smart-robot-car-kit-with-four-wheel-drives-for-arduino-uno-r3-project-christmas-gifts-2061627#.XXjMA9NR200, 11. rujna 2019.

[28] Adobe; „Numbers / Red square Icons“, s Interneta, https://stock.adobe.com/lt/images/numbers-red-square-icons/119112336, 17. rujna 2019.

DODATAK A

Programski kod koji je se koristi u projektu dan je u nastavku.

```
#include <Arduino.h>


// Uključivanje potrebnih biblioteka
#include <WiFi.h>
#include <Servo.h>


//Definiranje pinova L298N dvostrukog H-Mosta
#define DIRECTION1_PIN_L    2     // Smjer motora
#define DIRECTION2_PIN_L    4     // Smjer motora
#define SPEED_PIN_L         26    // Pin za kontroliranje brzine motora
#define DIRECTION1_PIN_R    27    // Smjer motora
#define DIRECTION2_PIN_R    33    // Smjer motora
#define SPEED_PIN_R         25    // Pin za kontroliranje brzine motora

// Definiranje pinova HC-SR04 ultrazvučnog senzora
#define ECHO_PIN            13    // Ultrazvučni Echo pin povezan na pin 13
#define TRIG_PIN            12    // Ultrazvučni Trig pin povezan na pin 12


// Definiranje varijabli HC-SR04 ultrazvučnog senzora
long Duration; // Vrijeme potrebno da senzor osluhne odaslani ultrazvučni val
float Distance;// Trenutna udaljenost

//Varijable autonomnog zaobilaženja prepreka
float CenterDistance,LeftDistance, RightDistance, LeftDiagonalDistance, RightD
iagonalDistance;
float DistanceLimit = 30;
float CentrDistanceLimit=45;

// Definiranje brzine koju ćemo koristiti
const uint16_t SPEED = 150;
const uint16_t RIGHT_SPEED =(int)SPEED*1.2 ;
const uint16_t OBSTACLE_AVOIDANCE_SPEED = 75;
const uint16_t OBSTACLE_AVOIDANCE_RIGHT_SPEED=95;
const uint16_t LeftAndRightSPEED = 200; //Brzina rotiranja robota
int RotateTime=2000;// Vrijeme rotiranja


// Definiranje kanala i PWM karaktreristika signala brzine vrtnje motora
const uint16_t FREQUENCY = 5000;
const uint8_t RESOLUTION = 8;
const uint8_t CHANNEL_L =  0;
const uint8_t CHANNEL_R =  1;
```

```
// Definiranje kanala i PWM karaktreristika signala položaja vrtnje servo
motora
int PWM_FREQUENCY = 50;
int PWM_CHANNEL = 0;
int PWM_RESOUTION = 8;
int GPIOPIN =14;


// Postavljanje naziva i lozinke kreirane lokalne Wi-Fi mreže
const char* ssid     = "ESP32-Access-Point";
const char* password = "123456789";


// Postavljnje Web servera na port 80
WiFiServer server(80);
// Varijabla koja pohranjuje HTTP zahtjev
String header;


// Prototipi funkcija
void InitRobotCar();
void SetMotorSpeed(const uint16_t speed_L, const uint16_t speed_R);
void GoForward();
void GoBack();
void GoLeft();
void GoRight();
void StopCar();
int READ_DISTANCE();
void OBSTACLE_AVOIDANCE();
void SERVO_ROTATE();

 // Funkcija za provjeru položaja servo motora
void SERVO_ROTATE( ){

// Kreiranje PWM signala za određeni položaj
int dutyCycle2 = 1;
ledcSetup(PWM_CHANNEL, PWM_FREQUENCY, PWM_RESOUTION);
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(300);

// Kreiranje PWM signala za određeni položaj
dutyCycle2 = 11 ;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(300);
Serial.print("Dijagonala");

// Kreiranje PWM signala za određeni položaj
dutyCycle2 = 20 ;
ledcWrite(PWM_CHANNEL, dutyCycle2);
```

```
delay(300);

// Kreiranje PWM signala za određeni položaj
dutyCycle2 = 24 ;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(300);
Serial.print("Dijagonala");

// Kreiranje PWM signala za određeni položaj
dutyCycle2 = 40 ;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(300);
}


// Postavljanje početnog uvjeta robota
void InitRobotCar(){
         // Ispisivanje podataka u serijski monitor
        Serial.println();
        Serial.println(F("Initializing Robot Car..."));
        // Postavljanje svojstava kanala
        ledcSetup(CHANNEL_L, FREQUENCY, RESOLUTION);
        ledcSetup(CHANNEL_R, FREQUENCY, RESOLUTION);
        // Postavljanje kanala na pin pomoću kojeg će bit kontroliran
        ledcAttachPin(SPEED_PIN_L, CHANNEL_L);
        ledcAttachPin(SPEED_PIN_R, CHANNEL_R);
        // Postavljanje početnih uvjeta HC-SR04 ultrazvučnog senzora
        pinMode(DIRECTION1_PIN_L, OUTPUT);
        pinMode(DIRECTION2_PIN_L, OUTPUT);
        pinMode(SPEED_PIN_L, OUTPUT);
        pinMode(DIRECTION1_PIN_R, OUTPUT);
        pinMode(DIRECTION2_PIN_R, OUTPUT);
        pinMode(SPEED_PIN_R, OUTPUT);
        StopCar();

        // Pauza određeni broj mikro-sekundi
        delay(4000);
        // Ispisivanje podataka u serijski monitor
        Serial.println(F("Robot Car initialized successfully."));
}

// Funkcija za postavljanje brzine vrtnje motora ESP-32 mikrokontrolera
void SetMotorSpeed(const uint16_t speed_L, const uint16_t speed_R){
        ledcWrite(CHANNEL_L, speed_L);
        ledcWrite(CHANNEL_R, speed_R);
        /* analogWrite() nije dostupna na ESP-32
           analogWrite(speedPinL,speed_L);
           analogWrite(speedPinR,speed_R);   */
}
```

```
/*  Kontrola motora */

// Idi Naprijed
void GoForward(){
        SetMotorSpeed(SPEED, RIGHT_SPEED);
        digitalWrite(DIRECTION1_PIN_L, HIGH);
        digitalWrite(DIRECTION2_PIN_L, LOW);
        digitalWrite(DIRECTION1_PIN_R, HIGH);
        digitalWrite(DIRECTION2_PIN_R, LOW);
}

 // Idi Naprijed u načinu rada autonomnog zaobilaženja prepreka
 void ObstacleGoForward(){
        SetMotorSpeed(SPEED,RIGHT_SPEED);
        digitalWrite(DIRECTION1_PIN_L, HIGH);
        digitalWrite(DIRECTION2_PIN_L, LOW);
        digitalWrite(DIRECTION1_PIN_R, HIGH);
        digitalWrite(DIRECTION2_PIN_R, LOW);
}

// Idi Lijevo
void GoLeft(){
        SetMotorSpeed(LeftAndRightSPEED, LeftAndRightSPEED);
        digitalWrite(DIRECTION1_PIN_L, HIGH);
        digitalWrite(DIRECTION2_PIN_L, LOW);
        digitalWrite(DIRECTION1_PIN_R, LOW);
        digitalWrite(DIRECTION2_PIN_R, HIGH);
}

// Idi Desno
void GoRight(){
        SetMotorSpeed(LeftAndRightSPEED, LeftAndRightSPEED);
        digitalWrite(DIRECTION1_PIN_L, LOW);
        digitalWrite(DIRECTION2_PIN_L, HIGH);
        digitalWrite(DIRECTION1_PIN_R, HIGH);
        digitalWrite(DIRECTION2_PIN_R, LOW);
}

// Idi nazad
void GoBack(){
        SetMotorSpeed(SPEED, SPEED);
        digitalWrite(DIRECTION1_PIN_L, LOW);
        digitalWrite(DIRECTION2_PIN_L, HIGH);
        digitalWrite(DIRECTION1_PIN_R, LOW);
        digitalWrite(DIRECTION2_PIN_R, HIGH);
}

// Stop
```

```
void StopCar() {
        digitalWrite(DIRECTION1_PIN_L, LOW);
        digitalWrite(DIRECTION2_PIN_L, LOW);
        digitalWrite(DIRECTION1_PIN_R, LOW);
        digitalWrite(DIRECTION2_PIN_R, LOW);
}


int READ_DISTANCE(){

// Retartiranje Trig pina
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);

// Postavljane Trig pina u aktivan rad 10 mikro-sekundi
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);

// Očitavanje Echo piva koji vraća vrijednost trajanja putovanj ultrzvučnog
vala u mikrosekundama
Duration = pulseIn(ECHO_PIN, HIGH);

// Računanje udaljenosti
Distance= Duration* 0.01657;
return (Distance);

// Ispisivanje udaljenosti u serijski monitor
Serial.println("");
Serial.println("");
Serial.print("Distance: ");
Serial.println(Distance);
Serial.println("");
Serial.println("");
}

// Funkcija autonomnog zaobilaženja prepreka
void OBSTACLE_AVOIDANCE(){

WiFiClient client = server.available();

// Brojač
int i=0;

void SetMotorSpeed();
 // Ispisivanje podataka u serijski monitor
Serial.println("Speed iznosi");
Serial.println(SPEED);
ledcSetup(PWM_CHANNEL, PWM_FREQUENCY, PWM_RESOUTION);
```

```
while( i<15) {

i++;
// Ispisivanje podataka u serijski monitor
Serial.print(" I iznosi ");
Serial.println(i);

ObstacleGoForward();

int ObstacleAvoidanceTime=300;

// Provjera stana udaljenosti za različite položaje servo motora
int dutyCycle2 = 20;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
CenterDistance=Distance;
Serial.print("Centralna udaljenost iznosi ");
Serial.println(CenterDistance);
if(CenterDistance<CentrDistanceLimit){
    StopCar();
    Serial.println("Zaustavljam  auto, centralna kriticna.");
    Serial.println(" ");
}


dutyCycle2 = 33;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
LeftDistance=Distance;
Serial.print("Lijeva udaljenost iznosi ");
Serial.println(LeftDistance);
if(LeftDistance<DistanceLimit){
    GoRight();
    delay(RotateTime/4);
    StopCar();
    Serial.println("Zaustavljam  auto, lijeva kriticna.");
    Serial.println(" ");
}

dutyCycle2 = 24;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
LeftDiagonalDistance=Distance;
Serial.print("Lijeva dijagonalna udaljenost iznosi ");
Serial.println(LeftDiagonalDistance);
```

```
if(LeftDiagonalDistance<DistanceLimit){
    GoRight();
    delay(RotateTime/4);
    StopCar();
    Serial.println("Zaustavljam  auto, lijeva dijagonalna kriticna.");
    Serial.println(" ");
}

dutyCycle2 = 20;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
CenterDistance=Distance;
Serial.print("Centralna udaljenost iznosi ");
Serial.println(CenterDistance);
if(CenterDistance<CentrDistanceLimit){
    StopCar();
    Serial.println("Zaustavljam  auto, centralna kriticna.");
    Serial.println(" ");
}


dutyCycle2 = 11;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
LeftDiagonalDistance=Distance;
Serial.print("Desna dijagonalna udaljenost iznosi ");
Serial.println(LeftDiagonalDistance);
if(LeftDiagonalDistance<DistanceLimit){
    GoLeft();
    delay(RotateTime/4);
    StopCar();
    Serial.println("Zaustavljam  auto, desna dijagonalna kriticna.");
    Serial.println(" ");
}


dutyCycle2 = 1;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
RightDistance=Distance;
Serial.print("Desna udaljenost iznosi ");
Serial.println(RightDistance);
if(RightDistance<DistanceLimit){
    GoLeft();
    delay(RotateTime/4);
    StopCar();
```

```
    Serial.println("Zaustavljam  auto, desna kriticna.");
    Serial.println(" ");
}

 dutyCycle2 = 20;
ledcWrite(PWM_CHANNEL, dutyCycle2);
delay(ObstacleAvoidanceTime);
READ_DISTANCE();
CenterDistance=Distance;
Serial.print("Centralna udaljenost iznosi ");
Serial.println(CenterDistance);
if(CenterDistance<CentrDistanceLimit){
    StopCar();
    Serial.println("Zaustavljam  auto, centralna kriticna.");
    Serial.println(" ");
}


//Dodatni uvjeti kontrole i definiranja ponašanja robota u datim uvjetima
if((CenterDistance<CentrDistanceLimit)&&(LeftDiagonalDistance<DistanceLimit)&&
(RightDiagonalDistance<DistanceLimit)){

    ObstacleGoForward();
    Serial.print("Uvjet iznosi ");
    Serial.println((CenterDistance<CentrDistanceLimit)&&(LeftDiagonalDistance<
DistanceLimit)&&(RightDiagonalDistance<DistanceLimit));
    Serial.println("Idem naprijed");
    Serial.println(" ");
    }

if((CenterDistance<CentrDistanceLimit)&&(LeftDistance<DistanceLimit)&&(RightDi
stance<DistanceLimit)){
    GoRight();
    delay(RotateTime);
    StopCar();
    Serial.print("Uvjet iznosi ");
    Serial.println((CenterDistance<DistanceLimit)&&(LeftDistance<DistanceLimit
)&&(RightDistance<DistanceLimit));
    Serial.println("Idem nazad");
    Serial.println(" ");
    }
  }
 StopCar();
}

// Petlja koja se izvršava samo jednom
void setup() {

// Početak serijske komunikacije
```

```
Serial.begin(9600);

// Definiranje svojstava kanala i PWM signala
ledcSetup(PWM_CHANNEL, PWM_FREQUENCY, PWM_RESOUTION);
ledcAttachPin(GPIOPIN, PWM_CHANNEL);

InitRobotCar();

pinMode(TRIG_PIN, OUTPUT); // Postavnjen Trigpin kao izlaz
pinMode(ECHO_PIN, INPUT); /// Postavnjen Ech pin kao ulaz


// Spajanje na kreiranu Wi-Fi mrežu sa željenim nazivom i lozinkom
Serial.print("Setting AP (Access Point)…");

// Uklonite lozinku ako želite  da vaša Wi-Fi mreža bude slobodna
WiFi.softAP(ssid, password);

IPAddress IP = WiFi.softAPIP();
Serial.print("AP IP address: ");
Serial.println(IP);

server.begin();
}

// Petlja koja se vrti beskonačno mnogo puta
void loop(){
 WiFiClient client = server.available();   // Čekanje na klijente

  if (client) {                      // Ako imamo novog klijenta
    Serial.println("New Client.");  // ispiši poruku u Serijski monitor
    String currentLine = "";   // napravi string koji drži podatke od klijenta
    while (client.connected()) {  // Loop petlja ako je klijet spojen na mrežu
      if (client.available()) {   // Ako se primaju podatci od klijenta
        char c = client.read();      // tad ih ispiši u Serijski monitor
        Serial.write(c);
        header += c;
        if (c == '\n') {        // Ako je podatak za novu praznu liniju
          //  a ako je trenutna lijija prazna napravi dvije prazne linije
          // To je kraj HTTP zahtjeva
          if (currentLine.length() == 0) {
            // HTTP naslov obično počinju s npr. HTTP/1.1 200 OK
            // kreiranje prazne linije na serveru
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

           // Ako IP adresa završava s "NAPRIJED", izvrši if naredbu grananja
```

```
if (header.indexOf("GET /NAPRIJED") >= 0) {

GoForward();
}

// Ako IP adresa završava s "NAZAD", izvrši if naredbu grananja
else if (header.indexOf("GET /NAZAD") >= 0) {

GoBack();
}

// Isti princip kao i za prethodne dvije naredbe grananja
else if (header.indexOf("GET /LIJEVO") >= 0) {

GoLeft();
}

 // Isti princip kao i za prethodne dvije naredbe grananja
else if (header.indexOf("GET /DESNO") >= 0) {

GoRight();
}

 // Isti princip kao i za prethodne dvije naredbe grananja
else if (header.indexOf("GET /STOP") >= 0) {

StopCar();
}

 // Isti princip kao i za prethodne dvije naredbe grananja
else if (header.indexOf("GET /OBSTACLE_AVOIDANCE") >= 0) {

OBSTACLE_AVOIDANCE();
}

// Isti princip kao i za prethodne dvije naredbe grananja
else if (header.indexOf("GET /SERVO_ROTATE") >= 0) {

SERVO_ROTATE();
}

 // Isti princip kao i za prethodne dvije naredbe grananja
else if (header.indexOf("GET /DEGREES90_ROTATE") >= 0) {

GoLeft();
Serial.println("EVO ME U 90 DEGREES");
delay(RotateTime/2);
StopCar();
}
```

```
             // Isti princip kao i za prethodne dvije naredbe grananja
            else if (header.indexOf("GET /DEGREES180_ROTATE") >= 0) {

            GoLeft();
            Serial.println("EVO ME U 180 DEGREES");
            delay(RotateTime);
            StopCar();
            }

            // Definiranje nčina klientskog prikaz HTML Web strance
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=devi
ce-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");


            client.println("<style>html { font-
family: Helvetica; display: inline-block; margin: 0px auto; text-
align: center;}");
            client.println(".button { background-
color: #4CAF50; border: none; color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-
size: 30px; margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-
color: #555555;}</style></head>");

            // Definiranje poveznica na Web stranici
            client.println("<body><h1>ESP32 Web Server</h1>");

            client.println("<p><a href=\"/NAPRIJED\">NAPRIJED</a></p>");

            client.println("<p><a href=\"/NAZAD\">NAZAD</a></p>");

            client.println("<p><a href=\"/LIJEVO\">LIJEVO</a></p>");

            client.println("<p><a href=\"/DESNO\">DESNO</a></p>");

            client.println("<p><a href=\"/STOP\">STOP</a></p>");

            client.println("<p><a href=\"/OBSTACLE_AVOIDANCE\">OBSTACLE_AVOIDA
NCE</a></p>");

            client.println("<p><a href=\"/SERVO_ROTATE\">SERVO_ROTATE</a></p>"
);

            client.println("<p><a href=\"/DEGREES90_ROTATE\">DEGREES90_ROTATE<
/a></p>");
```

```
            client.println("<p><a href=\"/DEGREES180_ROTATE\">DEGREES180_ROTAT
E</a></p>");

            client.println("</body></html>");

            // Kraj HTTP zatjeva percipira se sa sljedećom praznom linijom
            client.println();
            // Izlazak iz "While" petlje
            break;
          }
          else { // Ako imate novu liniju obrišite trenutnu
            currentLine = "";
          }
        }
        else if (c != '\r') {  // ko imate bilo što, vrati karakter
          currentLine += c;      // te ga dodaj na trenutnu liniju
        }
      }
    }
    // Obriši "header" varijablu
    header = "";
    // Prekini vezu
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```