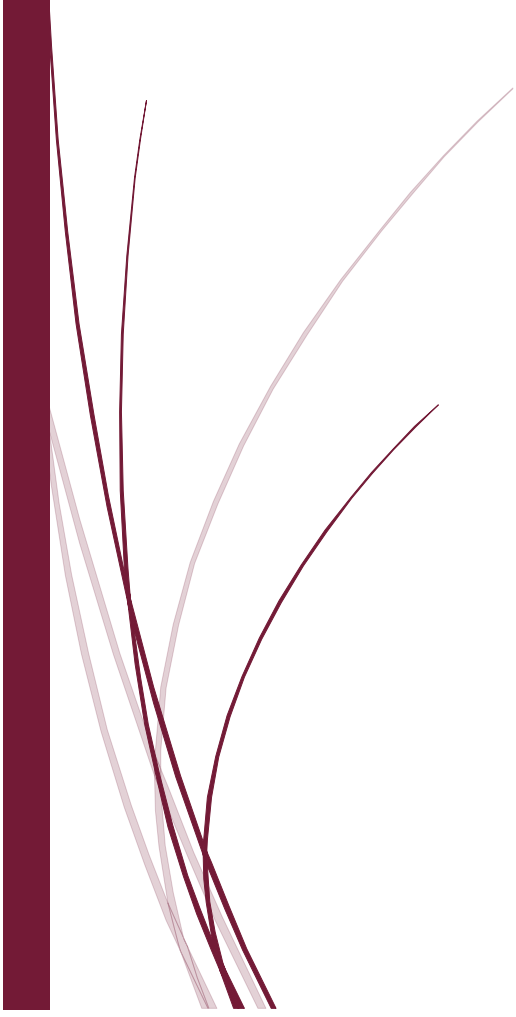
**Elektrotehnički fakultet**

**Univerziteta u Beogradu**

Avgust, 2020.

# Brodić

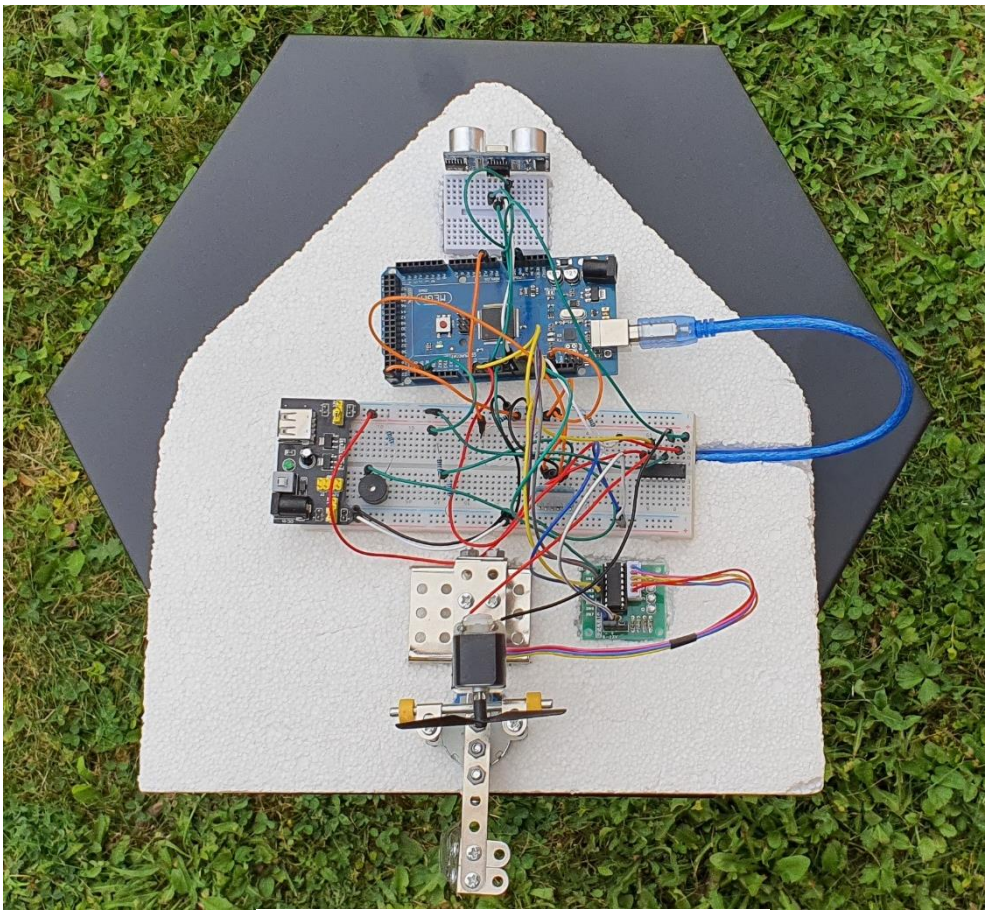


**Student:**
Nikola Zarić 2017/0494

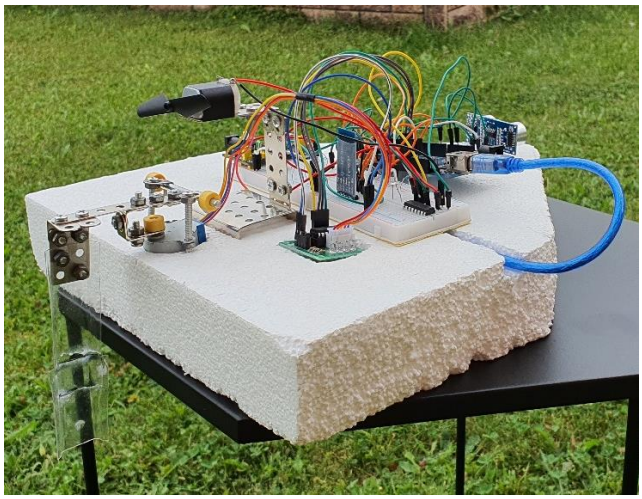
# Sadržaj

# 1. Kratak opis projekta

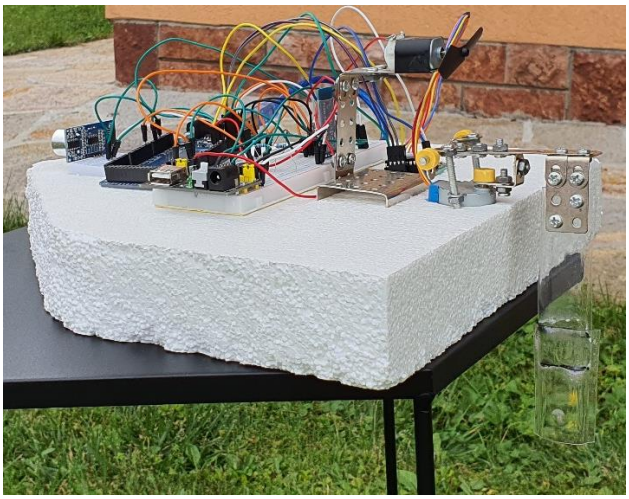
Glavni deo projekta je brodić, čije se upravljanje izvršava pomoću mobilne aplikacije. Komunikacija između brodića i aplikacije se uspostavlja bežično, pomoću Bluetooth-a. Preko mobilne aplikacije može se kontrolisati kretanje broda, paliti/gasiti motor, aktivirati sirena i paliti/gasiti svetlo. Osim ovih ručnih komandi, tu se takođe nalazi automatsko kočenje, odnosno isključenje motora ukoliko brodić naiđe na neku prepreku. Takođe, na brodiću se nalazi senzor koji omogućava automatsko paljenje i gašenje svetla u zavisnosti da li je napolju noć ili dan. Radi što većeg smanjenja težine i boljeg plutanja brodić je napravljen od stiropora. Na njemu su, zatim, udubljivanjem, ugrađene sve ostale komponente. Veća eksterna baterija koja se koristi za napajanje arduina, kao najteža komponenta, postavljena je u dubini stiropora, ispod protoborda, kako bi se očuvala ravnoteža broda. Kormilo je napravljeno od plastike da bi predstavljalo što manji teret step motoru i takođe je, pomoću metalne šipke, omogućen protiv teg, koji značajno smanjuje vibracije step motora prilikom njegovog okretanja. Na slikama 1.1 – 1.9 prikazan je izgled realizovanog brodića.



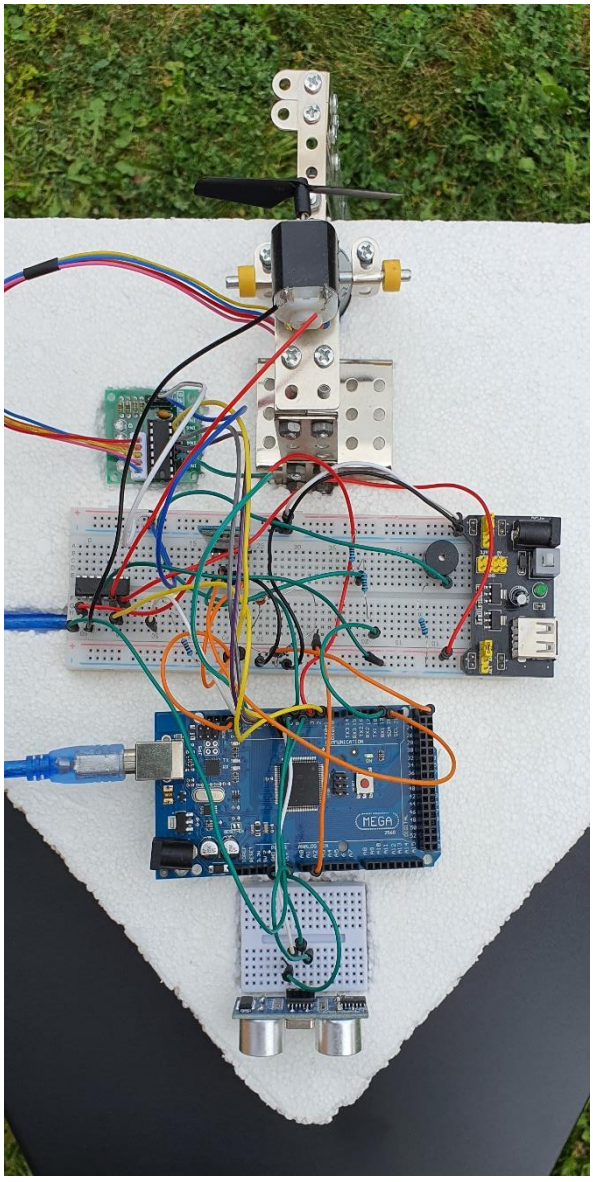
*Slika 1.1 Izgled brodića*



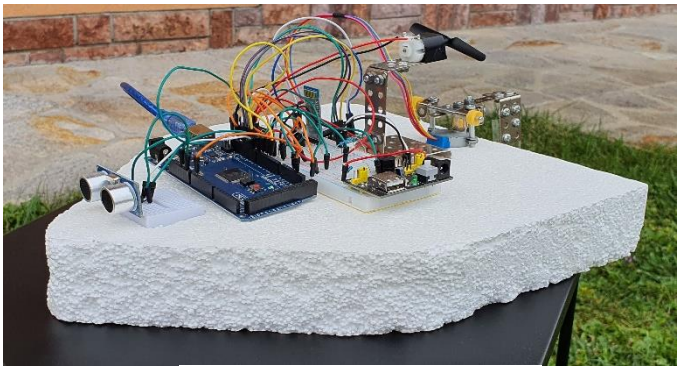
*Slika 1.2 Izgled brodića*



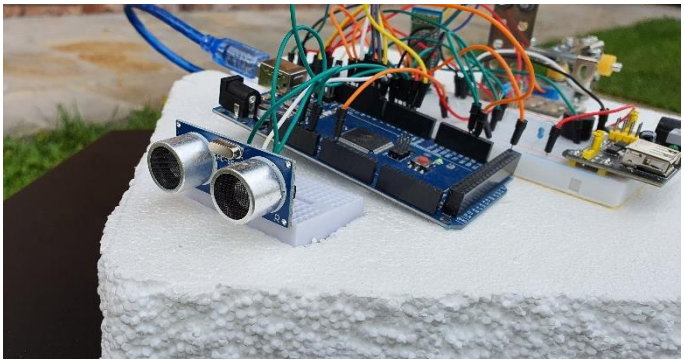
*Slika 1.3 Izgled brodića*



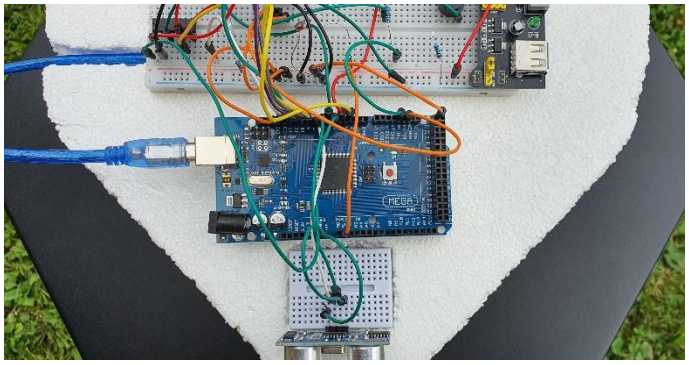
*Slika 1.4 Izgled brodića*



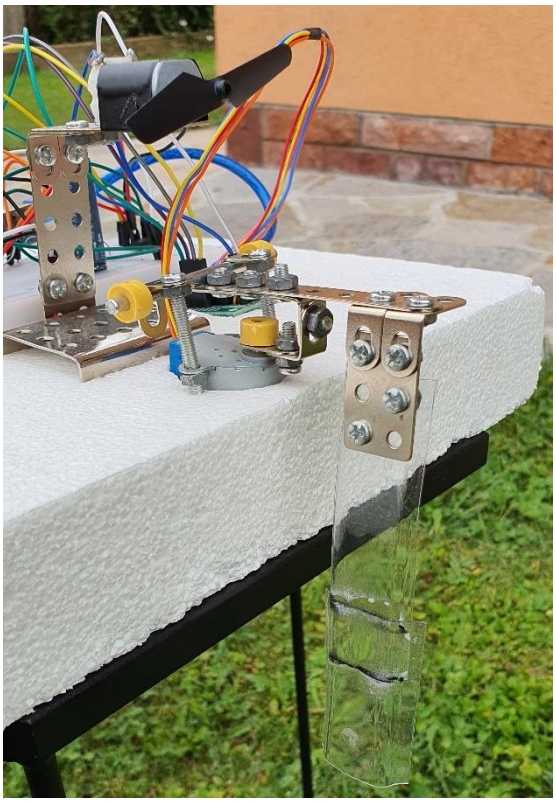
*Slika 1.5 Izgled brodića*



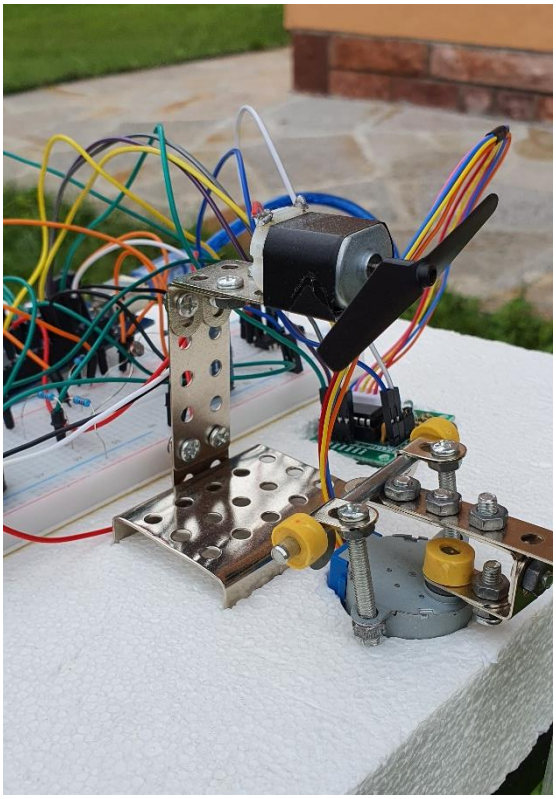
*Slika 1.6 Izgled brodića*



*Slika 1.7 Izgled brodića*



*Slika 1.8 Izgled brodića*



*Slika 1.9 Izgled brodića*

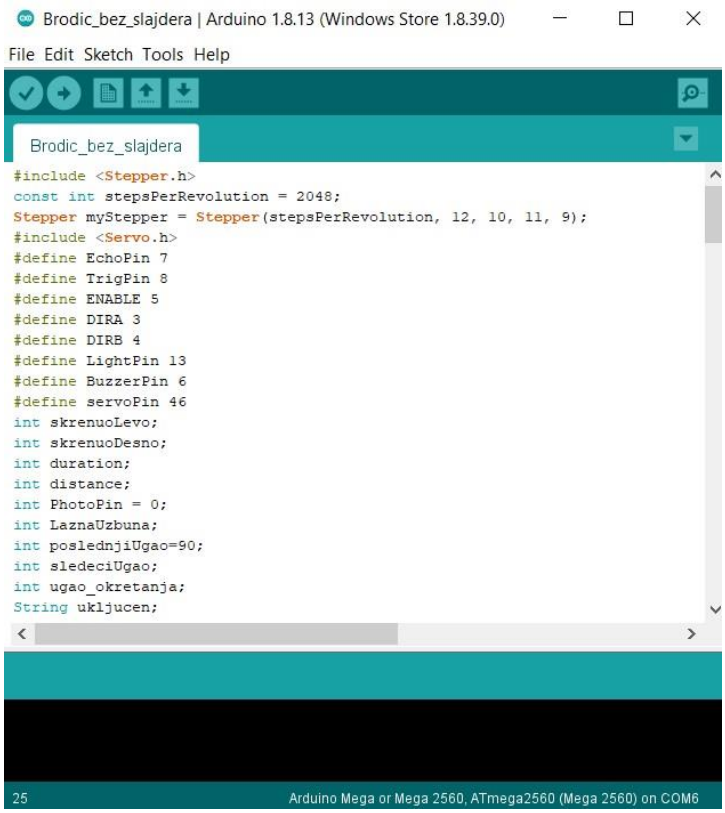
Hardver korišćen u izradi projekta:

- Arduino Mega 2560
- Baterija (9V)
- Eksterna baterija
- Bluetooth modul HC-05
- Stepper motor i drajver
- DC motor (3V)
- Zujalica (Buzzer)
- Led dioda (bela)
- Modul za napajanje (Power supply module)
- Ultrazvučni senzor
- Fotootpornik
- Otpornici (1 KOhm, 2Kohm)
- Žice
- Protobord

# 2. Radno okruženje ARDUINO IDE

Arduino IDE (*Integrated Development Environment*) je softver koji se koristi za pisanje kodova za kontrolu Arduino mikrokontrolera. Ovo okruženje se zasniva na platformi *Processing* koja omogućuje pisanje aplikacija na pojednostavljenoj varijanti jezika *Java*. Iz razloga što se pisanje kodova zasniva na jeziku *Java* postoje određene prednosti i mane. Prednost je što je IDE izuzetno jednostavan za korišćenje i u njemu će se bez problema snaći svaki početnik, a mana je da će iskusniji korisnici pronaći mnoštvo ograničenja za rad sa složenijim projektima. Na Slici 2.1 prikazan je izgled Arduino IDE okruženja.

Programi koji se pišu nazivaju se *Sketch* i samo okruženje sadrži veliki broj različitih primera programa. Svaki kod se sastoji od dve osnovne funkcije: *setup()* i *loop()*. U funkciji *setup*() se vrši definisanje svih pinova koji će biti korišćeni kao ulazi i izlazi, a u *loop*() funkciji se nalazi glavni deo programa koji se neprestano izvršava. Nakon što završimo pisanje samog programa, možemo izvršiti kompajliranje koda klikom na dugme *Verify* i ako je sve tačno, možemo izvršiti učitavanje koda na ploču klikom na dugme *Upload*.



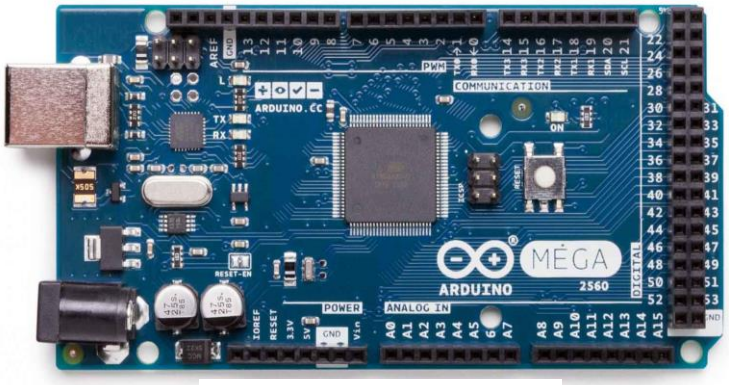

*Slika 2.1 Izgled Arduino IDE okruženja*

# 3. Hardver korišćen u izradi projekta

## 3.1. Arduino Mega

Arduino MEGA je mikrokontrolerska ploča sa ATMEGA2560 mikrokontrolerom. Ona ima 54 digitalna ulazno/izlazna pina (od kojih 14 mogu da se koriste kao PWM izlazi), 16 analognih ulaza, 4 UARTa (hardverski serijski portovi), 16 MHz kristalni oscilator, USB konekciju, priključak za napajanje, ICSP konektor i taster za reset. Može se napajati preko AC-DC adaptera, preko kompjutera ili direktno preko baterije. Ploča takođe sadrži i tri LED diode. L dioda svetli kada se neki kod spušta na ploču , dok Rx i Tx diode svetle kad mikrokontroler prima (*receive*) ili šalje (*transmit*) neke podatke.

U odnosu na neki standardni mikrokontroler (npr. Arduino Uno) ovaj se odlikuje većom memorijom (256 KB), što omogućava pisanje većih i kompleksnijih kodova, kao i mnogo veći broj pinova. Još jedna značajna prednost je što poseduje čak 4 para RX/TX ulaza , koji omogućavaju serijsku komunikaciju sa više uređaja odjednom, u našem slučaju RX0/TX0 je bilo rezervisano za serijsku komunikaciju sa kompjuterom, pa smo neki od preostala tri koristili da bi ostvarili komunikaciju preko Bluetooth-a. Na Slici 3.1 prikazana je mikrokontrolerska ploča Arguino MEGA.




*Slika 3.1 Arduino MEGA*

Tehnički podaci su dati u tabeli 3.1.

| | |
|---|---|
| **Mikrokontroler** | ATmega2560 |
| **Radni napon** | 5 V |
| **Napajanje** | 7-12 V |
| **Ulazni napon (limiti)** | 6-20 V |
| **Digitalni I/O pinovi** | 54 ( 14 sa PWM izlazom) |
| **Analogni ulazi** | 16 |
| **Flash Memorija** | 256 KB / 8 KB za bootloader |
| **SRAM** | 8 KB |
| **EEPROM** | 4 KB |
| **Takt** | 16 MHz |

*Tabela 3.1 Tehniči podaci mikrokontrolera Arduino Mega*

## 3.2. Napajanje

Za napajanje celog brodića korišćene su dve baterije, iz razloga što su zajedno sve komponente crpele preveliku struju iz samog kontrolera, što je samim tim uticalo i na regularnost njegovog rada.

Deo komponenti napajao se preko samog kontrolera, koji je bio povezan na DC bateriju od 9V, koja je prikazana na Slici 3.2.1, tj. preko pina 5V, dok se drugi deo napajao preko eksterne baterije koja je bila priključena za sam protobord preko modula za eksterno napajanje (Slika 3.2.2).


*Slika 3.2.1 DC baterija od 9V*

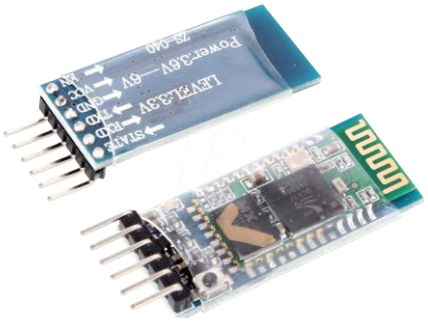

*Slika 3.2.2 Modul za eksterno napajanje*

## 3.3. Bluetooth modul HC-05

Ovaj Bluetooth modul ima opciju da radi i u *master* i u *slave* konfiguraciji. Sastoji se od 6 pinova:

1. Key/EN – koristi se za prebacivanje modula iz *Command mode* u *Data mode* i obrnuto; u zavisnosti od moda koji se odabere menja se i učestanost (38 400 bps *u Command mode* ili 9600 bps u *Data mode*)
   *Data mode* se koristi za razmenu podataka među uređajima, a *Command mode* se koristi za promene podešavanja blutut modula.
   U našem slučaju se ostvaruje komunikacija sa telefonom pa je potrebno da se modul nalazi u *Data mode,* pa postavljamo *baud rate* da je 9600.
2. VCC – za ovaj pin se povezuje napajanje ili od 5V ili od 3,3V.
3. GND – za ovaj pin se povezuje uzemljenje.
4. TXD – preko ovog pina se šalju podaci.
5. RXD – preko ovog pina se primaju podaci.
6. State – govori da li je modul konektovan ili ne.

Na samom Bluetooth modulu se nalazi i crvena lampica na osnovu koje utvrđujemo status povezanosti, tj. da li je povezan ili ne (pre uparivanja lampica treperi ubrzano a nakon povezivanja se treperenje usporava). Bluetooth modul HC-05 prikazan je na Slici 3.3.1.



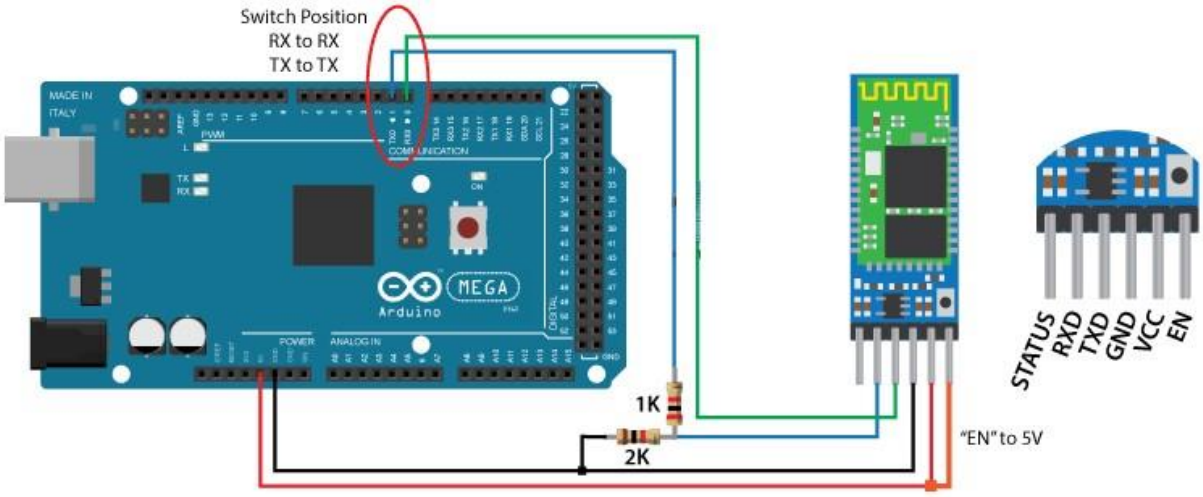
*Slika 3.3.1 Bluetooth modul HC-05*

Konekcija između samog Blutut modula i Arduina se ostvaruje serijski, pa se zato koriste Rx i Tx pinovi.

Prilikom povezivanja Bluetooth modula na arduino, treba voditi računa o više stvari:

- Ulaze Rx i Tx blutut modula spajamo sa ulazima Tx i Rx arduina, respektivno
- Serijska komunikacija arduina i računara se vrši preko Rx0 i Tx0, iz tog razloga prilikom spuštanja koda sa računara na ploču potrebni su nam ti ulazi da budu slobodni, zato biramo neke od preostalih ulaza da ostvarimo serijsku komunikaciju arduina i blututa
- Rx i Tx pinovi blutut modula rade na nivou od 3.3V, a pošto je i sam mikrokontroler u stanju da prepozna taj nivo onda je moguće Tx pin blututa direktno povezati na Rx pin mikrokontrolera. Međutim pošto se Tx pin mikrokontrolera nalazi na većem naponskom nivou u odnosu na Rx pin blututa, potrebno je izvršiti smanjenje napona uvođenjem naponskog razdelnika, koji koristi otpornike otpornosti 1Kohm i 2Kohm.

Na Slici 3.3.2 data je šema povezivanja blutut modula i mikrokontrolera.




*Slika 3.3.2 Šema povezivanja Bluetooth modula sa Arduinom*

## 3.4. DC motor 3V

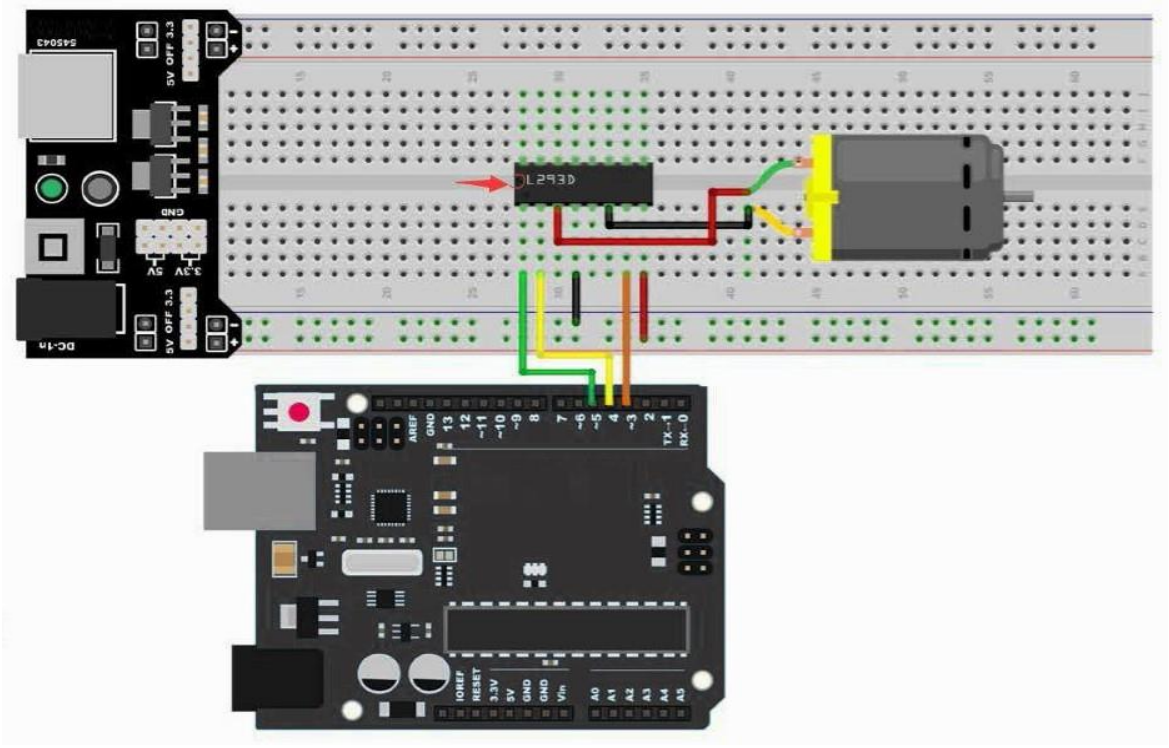
Ovaj motor se koristi kao pogon brodića. Za pokretanje ovog motora potrebna je velika struja , čak i veća nego što Arduino može da isporuči, iz tog razloga uvodimo, ranije pomenuto, eksterno napajanje koje priključujemo za protobord. Regulacija brzine motora vrši se promenom ulaznog napona , to se najčešće radi impulsno širinskom modulacijom (PWM), dok se odabir smera okretanja motora kontroliše menjanjem smera struje motora ,a to se najčešće radi H-mostom. Da bi mogli da vršimo i regulaciju brzine i odabir smera motora koristimo drajver L293D (Slika 3.4.1).

Prilikom startovanja motora, struja prolazi kroz namotaje u motoru. Kasnije, kada se motor isključi, deo struje ostaje u tim namotajima, pa u slučaju da je motor povezan direktno na Arduino, ta struja bi se vratila na ulaze Arduina, čime bi ga spržila. Iz tog razloga se motor mora povezati preko ovog drajvera.

Pomoću jednog L293D drajvera mogu se kontrolisati čak dva DC motora. Mi prilikom povezivanja koristimo samo jednu stranu drajvera, jer želimo da kontrolišemo samo jedan motor.

*Slika 3.4.1 Driver L293D*

Na Slici 3.4.2 data je šema povezivanja DC motora i mikrokontrolera.



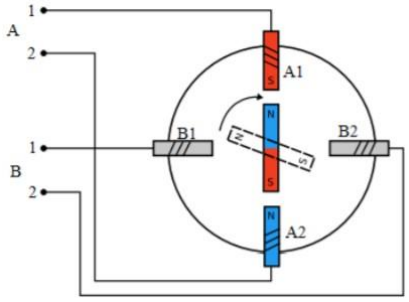

*Slika 3.4.2 DC motor povezan preko L293D za Arduino*

## 3.5. Stepper motor

Ovaj motor koristimo za upravljanje brodića, tj. za okretanje kormila. Motor se sastoji iz dva osnovna dela, statora i rotora. Rotor je onaj deo koji se okreće i za njega je fiksirano vratilo motora, dok je stator deo motora koji je stacionaran. Stator se sastoji od više namotaja žice, dok rotor može imati više različitih konstrukcija, pa u zavisnosti od toga postoje:

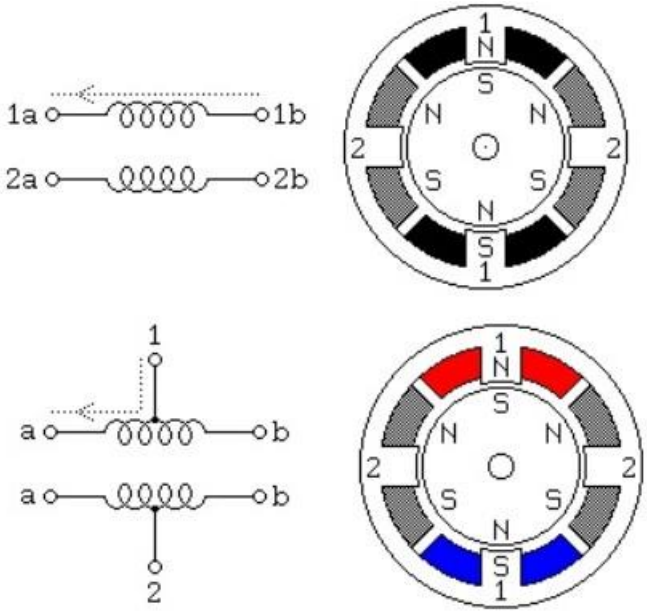
- Step motori sa stalnim magnetom,
- Step motori sa promenljivom reluktansom,
- Hibridni step motori.

Kroz namotaje u statoru prolazi struja koja uzrokuje nastajanje elektromagnetnog polja. To polje utiče da rotor počne da se okreće tako da teži da im se poklope suprotni polovi čime bi se poništilo delovanje te sile. Iz tog razloga, ukoliko želimo da nam se rotor i dalje okreće, potrebno je da pustimo struju kroz neki drugi namotaj, čije će elektromagnetno polje nastaviti da utiče i da time okreće rotor. Princip rada step motora prikazan je na Slici 3.5.1.




*Slika 3.5.1 Princip rada step motora*

Postoje dva osnovna tipa step motora, unipolarni i bipolarni, koji su prikazani na Slici 3.5.2. Unipolarni se sastoje od dva namotaja koji na sredini imaju izvedenu žicu, pa takav motor ima ukupno 5 ili 6 žica. Bipolarni motor se sastoji takođe od dva namotaja ali on uvek ima 4 žice.



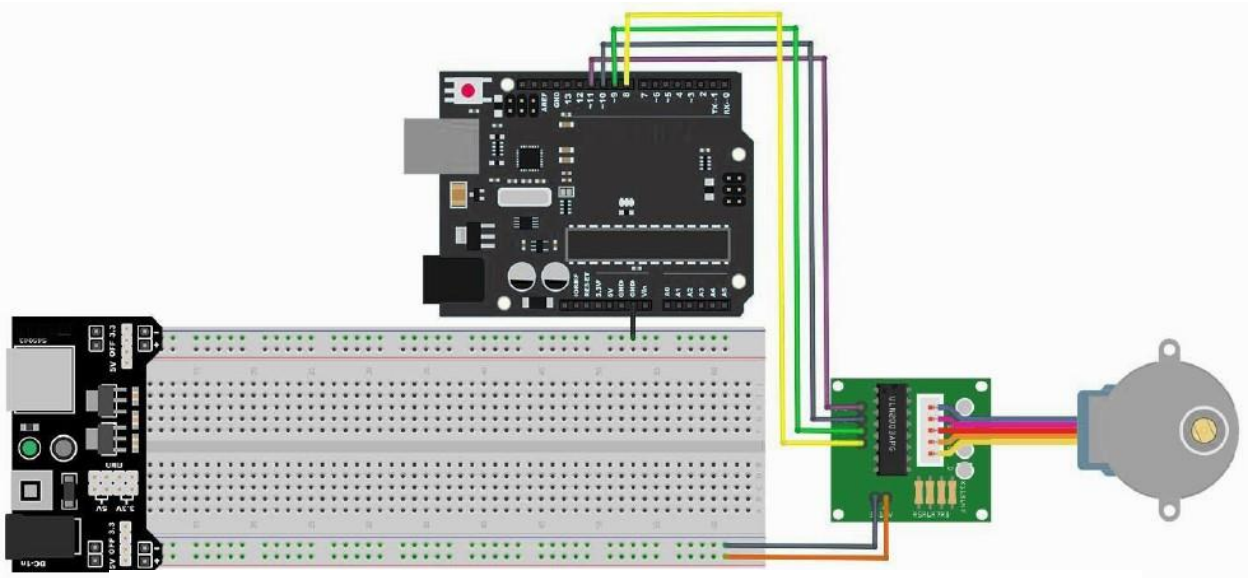

*Slika 3.5.2 Unipolarni i bipolarni motor*

Za ovaj tip motora je karakteristično da može da kontroliše ugaonu poziciju bez upotrebe povratne sprege, što znači da se ne koriste senzori za očitavanje pozicije (npr. enkoderi, pa je time i cena ovih motora manja).

Motor koji mi koristimo u našem projektu ima oznaku *28BYJ-48*. On pretvara električne impulse u diskretnu mehaničku rotaciju.

Treba napomenuti da nam je prvi izbor za okretanje kormila bio servo motor, ali smo promenili izbor zato što nam je odgovaralo to što step motor ima veći obrtni momenat pri malim brzinama, pa nam je time i samo okretanje kormila bilo glađe, tj. nije bilo praćeno velikim vibracijama koje su postojale kada smo to isto kretanje obavljali preko servo motora.

Na Slici 3.5.3 data je šema povezivanja Step motora i njegovog drajvera sa mikrokontrolerom.



*Slika 3.5.3 Šema povezivanja Step motora i njegovog drajvera sa mikrokontrolerom*

## 3.6. Zujalica (Buzzer)

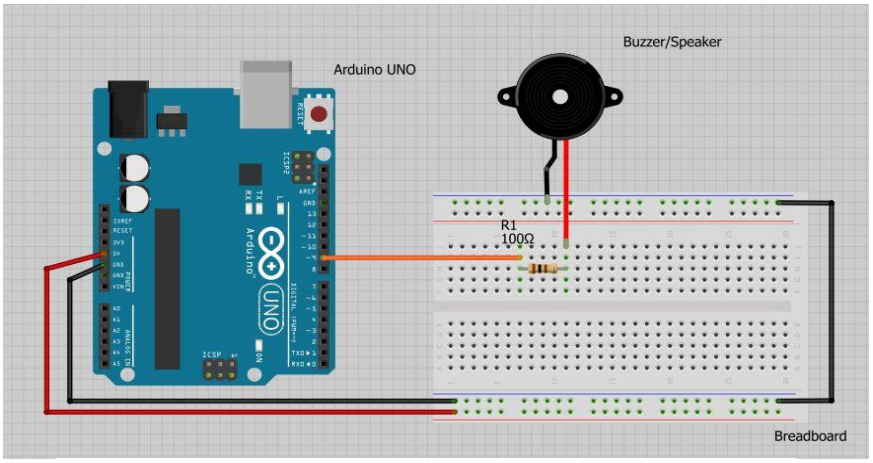
Zujalica se drugačija naziva *Piezo* zvučnik. Deli se u dve osnovne grupe, aktivni i pasivni. Spolja se može uočiti razlika između njih u tome što se kod pasivnog (*Slika 3.6.1*) na donjoj strani može videti zelena ploča od integrisanog kola, dok se kod aktivnog (*Slika 3.6.2*) ta ploča ne može videti jer je pokrivena crnim poklopcem.

Glavna razlika između ova dva tipa je što aktivna zujalica ima u sebi ugrađen oscilatorni izvor, pa stvara zvuk kada mu se dovede električni signal. Da bi se proizveo zvuk na pasivnoj zujalici potrebno je na ulaz dovesti povorku pravougaonih četvrtki. Mi koristimo u našem projektu koristimo aktivnu zujalicu.

*Slika 3.6.1 Pasivni zvučnik*

*Slika 3.6.2 Aktivni zvučnik*

Šema povezivanja zujalice i Arduina data je na Slici 3.6.3.



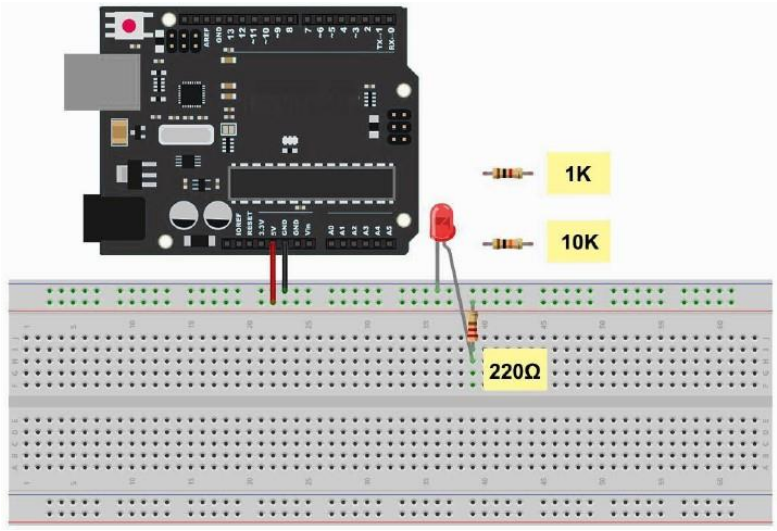

*Slika 3.6.3 Šema povezivanja zujalice sa mikrokontrolerom*

## 3.7. LED diode

LED (*Light Emitting Diode*) je izvor svetlosti koji koristi malu struju. Iz tog razloga diodu moramo povezati za napajanje preko otpornika, jer će u suprotnom struja koja prolazi kroz nju biti prevelika i doći će do probijanja diode. Najčešće korišćene veličine diode su 3mm, 5mm i 7mm. Mi koristimo belu diodu veličine 5mm. Na Slici 3.7.1 prikazan je izgled jedne LED diode.

*Slika 3.7.1 LED dioda*

Šema povezivanja LED diode i mikrokontrolera je data na Slici 3.7.2.




*Slika 3.7.2 Šema povezivanja LED diode sa miktrokontrolerom*

## 3.8. Ultrazvučni senzor

Koristi se za merenje rastojanja od nekog predmeta i često se koristi za izbegavanje prepreka. Senzor se sastoji iz dva osnovna dela *Trigger*-a i *Echo*-a. Preko *Trigger*-a se emituju periodični kratkotrajni visokofrekventni (iznad 20 kHz) zvučni impulsi. Ovi impulsi se prostiru vazduhom brzinom zvuka (oko 340,26 m/s). Ukoliko se ispred senzora nalazi prepreka, impulsi se onda odbijaju i vraćaju se ka senzoru gde ih registruje *Echo* deo. Bitna karakteristika ovih talasa je što prilikom kretanja kroz sredinu i eventualnog odbijanja o neki predmet, talasi gube veoma malu količinu energije. Drugim rečima ukoliko se talasi i odbiju od nekog objekta i dalje se kreću istom brzinom. Zahvaljujući ovoj karakteristici, lako se izračunava rastojanje. Na osnovu vremena kada je senzor emitovao signal, do trenutka prijema reflektovanog signala vrši se izračunavanje rastojanja od objekta.


*Slika 3.8.1 Ultrazvučni senzor*

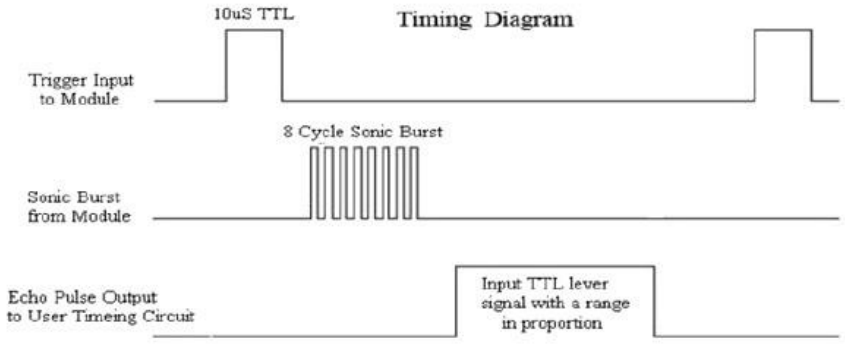
Mi koristimo model *HC-SR04*, koji je prikazan na Slici 3.8.1. Njegov domet merenja rastojanja je od 2cm do 400cm, sa preciznošću od 3mm. Sastoji se od 4 pina (*Vcc, Trig*, *Echo*, *Gnd*). Pinovi *Vcc* i *Gnd* se koriste za napajanje senzora.

Princip rada:

Na *Trig* pin se dovodi visok naponski nivo u trajanju od 10us. Senzor automatski šalje osam zvučnih impulsa frekvencije 40 kHz i detektuje da li je primljen reflektovan signal. Ukoliko je signal reflektovan, na *Echo* pinu dužina *HIGH* stanja odgovara vremenu od emitovanja do prijema reflektovanog signala. Rastojanje do prepreke se računa kao:

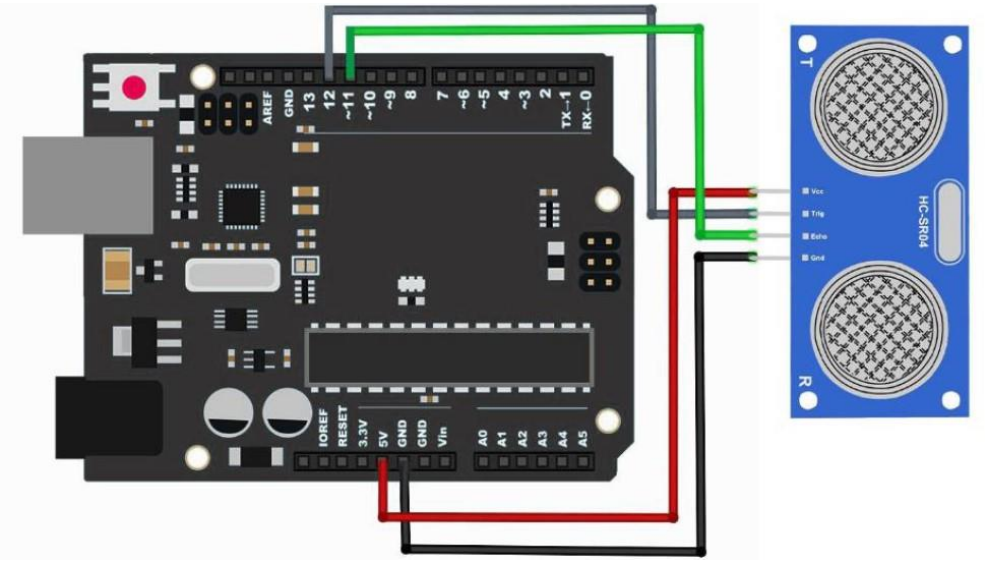
*Rastojanje = Trajanje_impulsa_na_echo_pinu * Brzina_zvuka(oko 340m/s) / 2*

Na Slici 3.8.2 je dat vremenski dijagram ultrazvučnog senzora.




*Slika 3.8.2 Vremenski dijagram ultrazvučnog senzora*

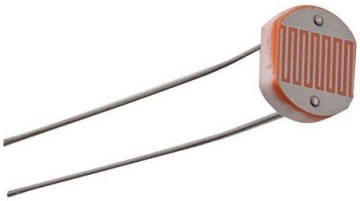
Šema povezivanja ultrazvučnog senzora i mikrokontrolera je data na Slici 3.8.3.




*Slika 3.8.3 Šema povezivanja ultrazvučnog senzora sa mikrokontrolerom*

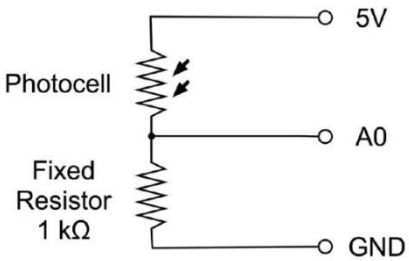
## 3.9. Fotootpornik

Ova komponenta se ponaša isto kao i običan otpornik, sa razlikom da mu se otpornost menja sa promenom intenziteta svetlosti kojoj je izložen. Na Slici 3.9.1 prikazan je izgled jednog fotootpornika.



*Slika 3.9.1 Fotootpornik*

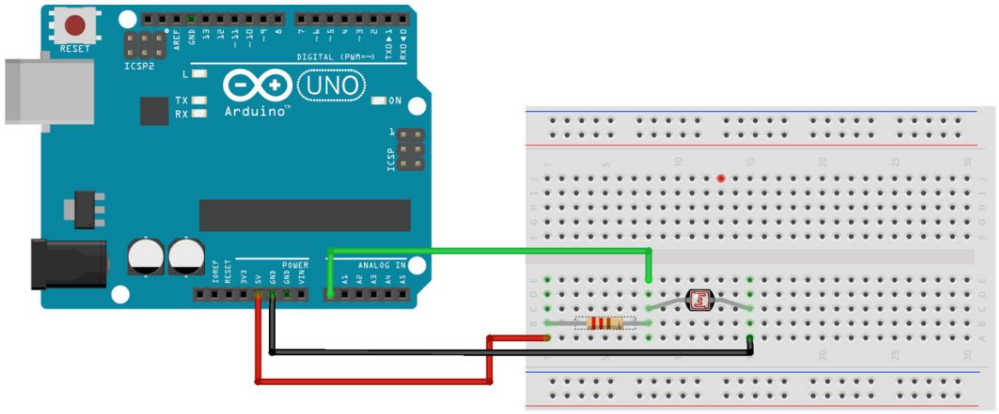
Fotootpornik koji mi koristimo ima otpornost od 50 kOhm kada se nalazi u potpunom mraku, a 500 Ohm kada je izložen jakom svetlu. Posto ne mozemo da merimo promenu otpornosti na analognom pinu Arduina, onda merimo promenu napona na otporniku koji ima konstantnu otpornost od 1 kOhm.




Kada se nalazi u mračnijoj sredini otpornost fotootpornika je velika pa očekujemo da će napon na fiksnom otporniku biti mali. Ukoliko se nalazi u svetlijoj sredini otpornost fotootpornika je mala, pa će napon na fiksnom otporniku biti veliki. Princip merenja napona otpornika je dat na Slici 3.9.2.

*Slika 3.9.2 Princip merenja napona otpornika*

Šema povezivanja fotootpornika i mikrokontrolera data je na Slici 3.9.3.




*Slika 3.9.3 Šema povezivanja fotootpornika i Arduina*

# 4. Kreiranje mobilne aplikacije

Za kreiranje mobilne aplikacije kojom će se upravljati brodić korišćena je aplikacija *MIT App Inventor 2.* Aplikacija sadrži dva ekrana. Prvi je uvodni, a drugi je onaj na kome se nalaze sve komande.


*Slika 4.1 Početni ekran aplikacije*

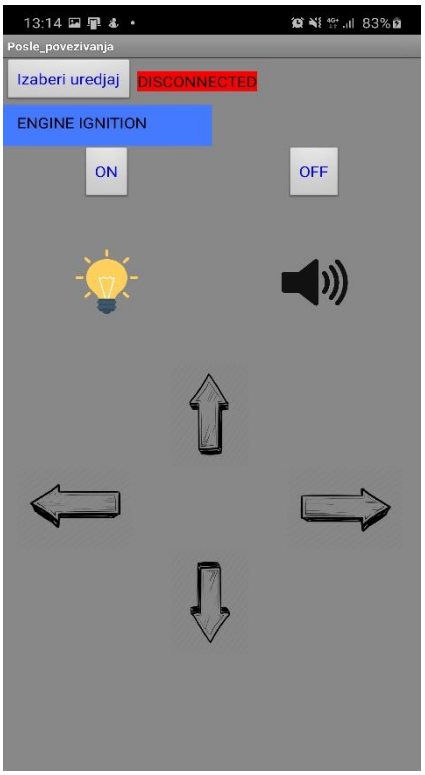
Vizuelni izgled ekrana tj. aplikacije se radi u odeljku *Designer* . Kod koji omogućava funkcionisanje aplikacije radi se u odeljku Blocks. Za MIT aplikaciju je karakteristično da se kod "piše" grafički, tj. sklapaju se blokovi različitih uloga (logička, matematička…) u smisaone celine.

Prilikom startovanja aplikacije *Brodic_bez_slajdera* izlazi uvodni ekran. Kod za ovaj ekran je prilično jednostavan, sastoji se samo od jednog bloka koji kada se pritisne dugme *Nastavi ka programu* prelazi se na ekran pod nazivom *Posle_povezivanja*. Na Slici 4.1 prikazan je početni ekran aplikacije, dok je na Slici 4.2 prikazan odgovarajući program za dati ekran aplikacije.

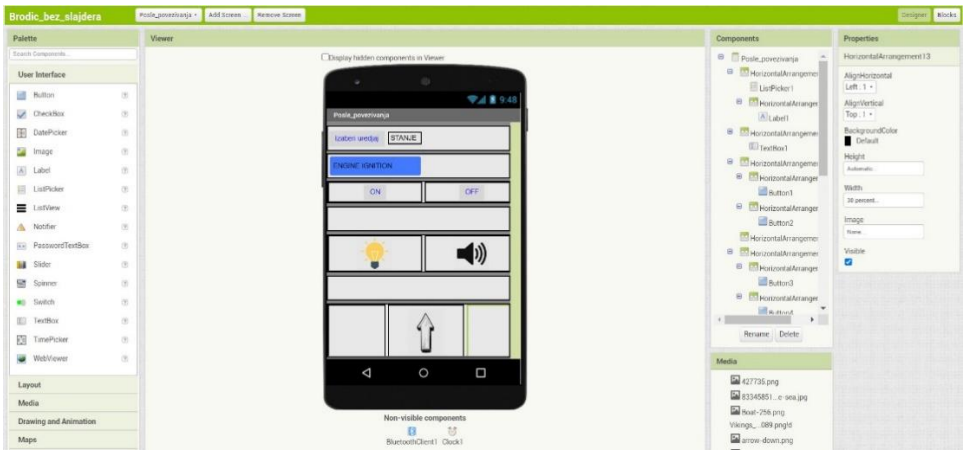

*Slika 4.2 Kod za početni ekran*

Izgled glavnog ekrana je prikazan na *Slici 4.3.* Sastoji se od:

- Dugme za odabir uređaja sa kojim želimo da se povežemo, pored njega se nalazi labela koja u zavisnosti da li smo povezani ili ne, sija crveno/zeleno i piše DISCONNECTED/CONNECTED.
- Dugmići ON i OFF koji služa za paljenje i gašenje motora brodića.
- Sijalica čijim pritiskom se uključuje i isključuje led dioda.
- Zvučnik čijim se pritiskom oglašava sirena.Strelice (napred, nazad, levo, desno) kojima se upravlja brodom.
- Strelicama napred/nazad se motoru menja smer okretanja, a strelicama levo/desno se pomera kormilo.




*Slika 4.3 Izgled glavnog ekrana na telefonu*

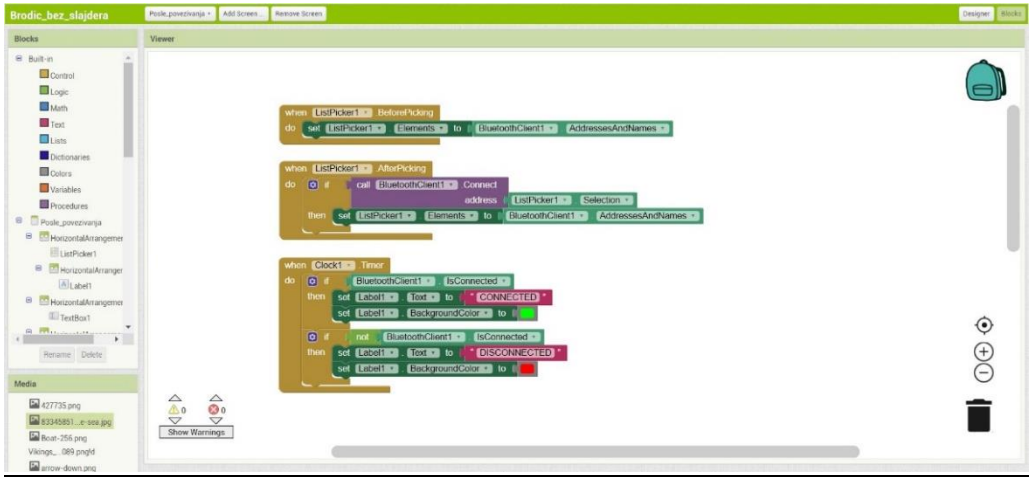



*Slika 4.4 Izrada aplikacije u MIT App Inventoru.*

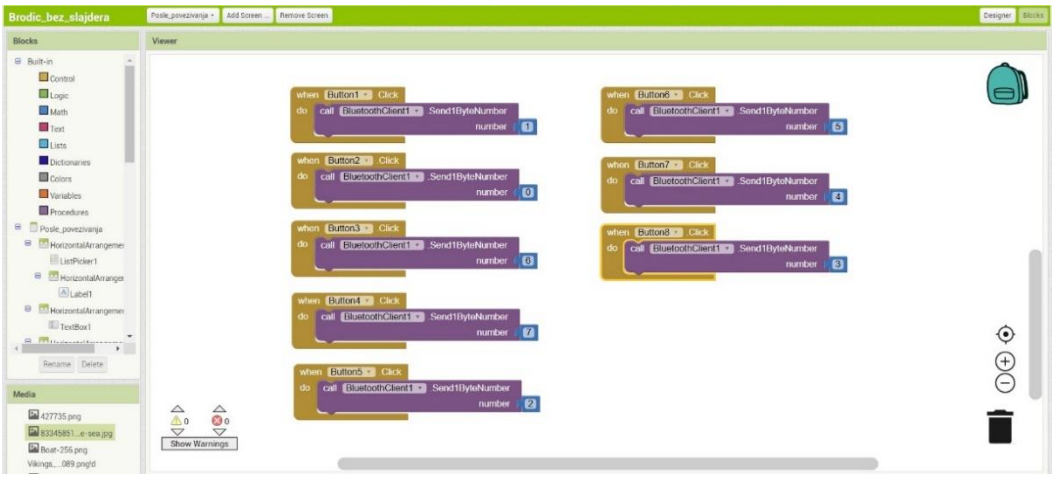
Izgled aplikacije u MIT App Inventoru dat je na Slici 4.4, dok je izgled koda za glavni ekran dat je na Slikama *4.5* i *4.6.*

Prva dva segmenta koda odnose se na uspostavljanje Bluetooth veze sa drugim uređajem. Odabir uređaja sa kojim želimo da se povežemo vrši se preko formirane liste dostupnih uređaja (*ListPicker*). U trećem segmentu se, na osnovu toga da li je ostvarena konekcija ili ne, menja boja i reč labele.

Ostatak koda definiše samo koji se broj šalje kada se neko dugme pritisne. Na primer, kada se pritisne dugme ON, poslaće se broj 1.




*Slika 4.5 Kod za glavni ekran*




*Slika 4.6 Kod za glavni ekran*

# 5. Pisanje koda za brodić

### OBJAŠNJENJE KODA

Na početku koda uključujemo sve biblioteke potrebne za rad nekih komponenti, u našem slučaju potrebno je samo uključiti biblioteku za step motor jer pomoću nje kontrolišemo i njegov drajver.

```
#include <Stepper.h>                                    //Biblioteka za stepper motor
```

Takođe, potrebno je definisati i ukupan broj koraka koji ima model step motora koji mi koristimo. Jedan pun krug je podeljen na 2048 delova tj. koraka.

```
const int stepsPerRevolution = 2048;                    //Ukupan broj koraka stepper motora
```

Na osnovu sledeće komande formira se objekat *myStepper*, on se incijalizuje brojem koraka u jednoj revoluciji, kao i brojevima pinova koji su povezani za sam drajver step motora.

```
Stepper myStepper = Stepper(stepsPerRevolution, 12, 10, 11, 9);
```

Definisanje pinova korišćenih za ultrazvučni senzor:

```
#define EchoPin 7
#define TrigPin 8
```

Definisanje pinova koji su povezani na drajver DC motora:

```
#define ENABLE 5
#define DIRA 3
#define DIRB 4
```

Definisanje pina kojim se kontroliše paljenje i gašenje LED diode:

```
#define LightPin 13
```

Definisanje pina kojim se kontroliše paljenje i gašenje zvučnika:

```
#define BuzzerPin 6
```

U nastavku se definišu razne promenljive koje se koriste u ostatku programa:

```
int skrenuoLevo;
int skrenuoDesno;
int duration;
int distance;
int PhotoPin = 0;
int LaznaUzbuna;
int poslednjiUgao=90;
int sledeciUgao;
int ugao_okretanja;
String ukljucen;
int pozicija_kormila=0;
int  ukljuciti=0;
```

Radi bolje iskorišćenosti prostora i praktičnosti, napravljena je funkcija *NapraviPuls()* koja se poziva više puta u ostatku koda. Ova funkcija formira impulse ultrazvučnog senzora, tako što ručno postavlja ulaz pina *Trig* na logičku nulu, pa nakon 2 mikrosekunde na logičku jedinicu i na kraju , nakon 10 mikrosekundi opet na logičku nulu.

```
void NapraviPuls(){                                   // Fja za pravljenje impulsa za ultrasonic senzor
digitalWrite(TrigPin, LOW);
delayMicroseconds(2);
digitalWrite(TrigPin, HIGH);
delayMicroseconds(10);
digitalWrite(TrigPin, LOW);

 }
```

U funkciji *void setup()* definišemo da li su korišćeni digitalni pinovi ulazni ili izlazni, postavljamo *baud rate* serijskih komunikacija između arduina i računara i između arduina i telefona, na 9600. I na kraju podešavamo i brzinu step motora pomoću funkcije *setSpeed().*

```
void setup() {

   pinMode(EchoPin, INPUT);
   pinMode(TrigPin, OUTPUT);
   pinMode(ENABLE,OUTPUT);
   pinMode(DIRA,OUTPUT);
   pinMode(DIRB,OUTPUT);
   Serial.begin(9600);      // serijska komunikacija arduina i računara
   Serial1.begin(9600);     // serijska komunikacija arduina i telefona
   myStepper.setSpeed(10);    // podešavanje brzine step motora

}
```

Ostatak koda nalazi se glavnoj petlji *void loop()* koja se stalno ponavlja.
U samoj inicijalizaciji postavili smo promenljivu ***ukljuciti*** na 0, to znači da je brodić ugašen i želimo da onemogućimo rad svih ostalih komandi pre njegovog paljenja. Dok brodić nije uključen krećemo se samo kroz prvi if blok. Naredbom *if (Serial1.available()>0)* proveravamo da li je neka komanda stigla preko blutut modula, ukoliko jeste upisujemo je u promenljivu ***ukljucen.***

```
void loop() {

if(ukljuciti==0){                                     //Ako jos uvek nije ukljucen krecemo se samo kroz ovaj blok
  if (Serial1.available()>0){
  ukljucen=Serial1.read();
  Serial.println("Ukljucen je");
  Serial.println(ukljucen);}
```

Ispitujemo da li je ova promenljiva jednaka ‘1’, jer ako jeste to znači da je na telefonu bilo pritisnuto dugme ON, a ako je pritisnuto bilo koje drugo dugme onda to zanemarujemo. Ukoliko je ispunjen uslov promenljivoj ***ukljuciti*** se dodeljuje vrednost 1.

Pomoću naredne tri naredbe se uključuje motor i postavlja da se okreće u smeru B, tj. okreće se tako da brodić ide unapred.
Pošto za pomeranje kormila brodića koristimo step motor, a za njega je karakteristično da za razliku od servo motora ne pamti poziciju u kojoj se nalazi, onda to moramo ručno odraditi pa uvodimo promenljivu ***pozicija_kormila*** i postavljamo je na 0. Kada se kormilo bude pomeralo u jednu stranu (desno) ova promenljiva će se inkrementirati, a kad se bude okretalo u drugu (levo) ono će se dekrementirati.

```
  if(ukljucen=='1'){                                  //Ako se pritisne dugme ON onda se salje 1
   ukljuciti=1;
   digitalWrite(ENABLE,HIGH); // enable on
   digitalWrite(DIRA,LOW);                            //Odredjujemo u kom se smeru okrece
   digitalWrite(DIRB,HIGH);
```

```
  pozicija_kormila=0;
  }
```

Ukoliko je brodić već uključen prelazi se u *else* granu. Ovde se prvo meri napon na otporniku i upisuje u promenljivu ***reading*** u zavisnosti od njegove vrednosti i od promenljive ***rucno_ukljucivanje_svetla*** zavisi da li će se dioda upaliti ili neće. Promenljivu ***rucno_ukljucivanje_svetla*** smo uveli, jer smo imali problem kada pošaljemo komandu preko telefona da se upali svetlo, da svetlo bude uključeno vrlo kratak period jer čim ponovo počne da prolazi kroz glavnu petlju ono biva ponovo ugašeno, jer je napon na otporniku veci od 125. Tako da uvevši ovaj uslov dobili smo da dioda sija sve dok ponovo ne pritisnemo dugme za njeno isključenje.

```
else{                                          // Ovaj ostatak koda se izvrsava tek nakon sto se brodic upalio
  int reading= analogRead(PhotoPin);                    //Citanje vrednosti sa fotootpornika
  if(reading<125 || rucno_ukljucivanje_svetla==         //Ako je manje od 125(pao mrak) upali svetlo
   digitalWrite(LightPin,HIGH);
    }
  else{digitalWrite(LightPin,LOW);}
```

Ovde prvi put pozivamo funkciju *NapraviPuls*(). Na osnovu funkcije ***pulseIn()*** određujemo trajanje između primljenog i poslatog talasa, a na osnovu toga i brzine prostiranja zvuka u vazduhu određujemo i rastojanje od prepreke.

```
  NapraviPuls();
  duration = p3ulseIn(EchoPin, HIGH);
  distance= duration*0.034/2;                    //Odredjivanje razdaljine
  //Serial.println(distance);
```

Ukoliko se rastojanje nalazi u opsegu od 0 do 15, onda se nalazimo previše blizu prepreke i potrebno je zaustaviti motor.

```
  if(distance>0 & distance<15)
 {
   LaznaUzbuna=0;
   Serial.println("STANI");
   digitalWrite(ENABLE,LOW);                     // Gasenje motora
   delay(200);
```

Prilikom testiranja brodića primetili smo da ultrazvučni senzor može imati takozvane outliere, tj vrednosti koje uopšte nisu tačne već nastaju usled greške tj. nekog šuma same opreme. Usled ove pojave brodić se često zaustavljao bez pravog razloga. Da bi ovaj problem rešili uveli smo promenljivu ***LaznaUzbuna***, ona se u startu postavlja na 0 kada se registruje neka prepreka. Prvo prolazimo kroz jednu *for* petlju gde ponovo određujemo rastojanje da bi utvrdili da li se ispred nalazi prepreka ili je bila reč o grešci u merenju. Ukoliko sada vrednost rastojanja bude veća od 15, onda je u pitanju bio *outliar*, tako da postavljamo promenljivu ***LaznaUzbuna*** na 1, uključujemo motor i izlazimo iz petlje.

```
for(int i=0;i< 2; i++){
  NapraviPuls();
  duration = pulseIn(EchoPin, HIGH);
  distance= duration*0.034/2;
  Serial.println(distance);
  if(distance>15){
  LaznaUzbuna=1;
  Serial.println("Lazna uzbuna");
  digitalWrite(ENABLE,HIGH);
  delay(500);}
  }
```

Ukoliko je rastojanje ipak manje od 15, reč je o pravoj prepreci, pa ulazimo u novu *if* naredbu. Prvo uključujemo alarm. To radimo pomoću funkcije ***tone()***, ona prima pin na koji je priključena zujalica, frekvenciju (1000), kao i trajanje tona (200 ms). Zatim ide kašnjenje od 400ms, ali ovde treba primetiti da će funcija *tone()* raditi baš tokom prvih 200ms kašnjenja, a zatim će sledećih 200ms zujalica biti isključena tj. neće se čuti ton.

```
if(LaznaUzbuna!=1){
    for(int i=0;i<2;i++){                    // ALARM
    tone(BuzzerPin,1000,200);                //Prilikom koriscenja ove funkcije neki ulazi ne rade jer i oni koriste
fju timer kao i fja tone()
    delay(400);}
```

Zatim uvodimo promenljivu ***iskoci*** koju inicijalizujemo na 0, ona nam služi da kasnije izađemo iz *while* petlje. Sve dok je ona jednaka nuli ne izlazimo iz *while* petlje. Svakim prolaskom kroz petlju merimo rastojanje proveravamo da li je veće od 25, ukoliko jeste izlazimo iz petlje i uključujemo motor.

```
int iskoci=0;
   while(iskoci==0){
     NapraviPuls();
     duration = pulseIn(EchoPin, HIGH);
     distance= duration*0.034/2;

     if(distance>25){                        //Ako je ovaj uslov ispunjen izadji i petlje
     iskoci=1;
     digitalWrite(ENABLE,HIGH);
     delay(200);}
    }}}
```

Nakon svih ovih provera, ponovo proveravamo da li postoji neka nova komanda koja je poslata preko blututa. Ukoliko postoji, upisuje se u promenljivu ***komanda***. Na osnovu *switch-case* strukture utvrđujemo koja komanda je poslata i u skladu sa tim reagujemo.

```
if (Serial1.available()>0){
 int komanda = Serial1.read();

 switch(komanda)
{
```

Ukoliko je komanda=0, to znači da je pritisnuto OFF dugme i da je potrebno ugasiti motor, stavljanjem pina *ENABLE* na logičku nulu. Takođe, tad na osnovu pamćenja pozicije kormila, vraćamo kormilo u centralni položaj. Ranije smo proračunali da je potrebno oko 5,68 koraka step motora da bi se kormilo okrenulo za 1 stepen a pošto želimo da ga okrenemo za 5 stepeni onda 5,68 koraka množimo sa 5. Funkcijom ***mySteppper.step()*** pomeramo step motor, tj. kormilo, a ova funkcija prima broj koraka za koliko motor treba da se pomeri.

```
  case 0:
    Serial.println("Ugasi motor");
    digitalWrite(ENABLE,LOW);
    digitalWrite(DIRA,LOW);                          //Odredjujemo u kom se smeru okrece
    digitalWrite(DIRB,LOW);
    myStepper.step((-pozicija_kormila)*5*5.68);      //Vracanje na pocetnu poziciju (sredina)
    pozicija_kormila=0;
    ukljuciti=0;
    break;
```

Ukoliko je komanda=1, to znači da je pritisnuto dugme ON, pa onda nista ne treba da radimo, jer je motor već uključen. Za kretanje unapred odnosno unazad šaljemo 2 tj. 3 i u zavisnosti od toga menjamo koji od dva pina (*DIRA, DIRB*) stavljamo na logičku nulu, a koji na jedinicu.

```
case 1:
 Serial.println("Motor ostaje ukljucen");
 break;

case 2:
  Serial.println("Napred");
  digitalWrite(DIRA,LOW);                    //Odredjujemo u kom se smeru okrece
  digitalWrite(DIRB,HIGH);
  break;

case 3:
  Serial.println("Nazad");
  digitalWrite(DIRA,HIGH);                   //Odredjujemo u kom se smeru okrece
  digitalWrite(DIRB,LOW);
  break;
```

Ako je komanda jednaka 4 ili 5, tada se kormilo okreće desno ili levo. Okreće se za 5 stepeni odnosno za 5*5,68 koraka i pozicija kormila se inkrementira ako skreće desno. Ako sreće levo, onda se okreće za -5 stepeni i pozicija kormila se dekrementira.

```
case 4:
  Serial.println("Desno");
  myStepper.step(5*5.68);
  pozicija_kormila+=1;
  break;

case 5:
  Serial.println("Levo");
  myStepper.step(-5*5.68);
  pozicija_kormila-=1;
  break;
```

Za komanda=6 ulazimo u *case* 6 u kome se uključuje/isključuje svetlo u zavisnosti od toga da li je prethodno bilo upaljeno ili ugašeno.

```
case 6:
  Serial.println("Ukljuceno/Iskljuceno svetlo");
  if(rucno_ukljucivanje_svetla==0){
  digitalWrite(LightPin,HIGH);
  rucno_ukljucivanje_svetla=1;}
  else {
   digitalWrite(LightPin,LOW);
  rucno_ukljucivanje_svetla=0;}
  break;
```

Ukoliko je komanda=7 tada uključujemo sirenu na sličan način kao i ranije u kodu.

```
case 7:
  Serial.println("Sirena");
  for(int i=0;i<2;i++){                      // Sirena
  tone(BuzzerPin,4000,200);                  //Prilikom koriscenja ove funkcije neki ulazi ne rade jer i oni
koriste fju timer kao i fja tone()
```

```
    delay(400);}
    break;
  }

}}}
```

# 6. Literatura


1. https://www.sk.rs/2016/11/sklp05.html
2. https://docs.rs-online.com/9ab3/0900766b80e8ba22.pdf
3. https://forum.arduino.cc/
4. https://www.electronicwings.com/sensors-modules/bluetooth-module-hc-05-
5. https://www.instructables.com/id/Controlling-a-Stepper-Motor-with-an-Arduino/
6. https://www.youtube.com/watch?v=0qwrnUeSpYQ
7. https://www.seeedstudio.com/blog/2019/03/04/driving-a-28byj-48-stepper-motor-with-a-uln2003-driver-board-and-arduino/
8. https://www.arduino.cc/en/Tutorial-0007/BlinkingLED
9. http://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/Ultrazvucni%20senzor%20HC-SR04.pdf
10. https://www.automatika.rs/projekti/upotreba-ultrazvucnog-senzora-daljine-arduino.html