Model auta na daljinsko upravljanje baziran na Arduino platformi

Zidar, Petar

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

PETAR ZIDAR

MODEL AUTA NA DALJINSKO UPRAVLJANJE BAZIRAN NA ARDUINO PLATFORMI

ZAVRŠNI RAD

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

PETAR ZIDAR

MODEL AUTA NA DALJINSKO UPRAVLJANJE BAZIRAN NA ARDUINO PLATFORMI

ARDUINO BASED REMOTE CONTROL CAR

ZAVRŠNI RAD

Mentor:
Jurica Trstenjak, v. pred.

ČAKOVEC, 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
ODBOR ZA ZAVRŠNI RAD

Čakovec, 12. siječnja 2021.

država:	**Republika Hrvatska**
Predmet:	**Osnove elektrotehnike i elektronike**
Polje:	**2.09 Računarstvo**

ZAVRŠNI ZADATAK br. 2020-RAČ-I-55

Pristupnik:	**Petar Zidar (0313017831)**
Studij:	izvanredni preddiplomski stručni studij Računarstvo
Smjer:	Inženjerstvo računalnih sustava i mreža

Zadatak: **Model auta na daljinsko upravljanje baziran na Arduino platformi**

Opis zadatka:

Potrebno je izraditi model auta (ili koristiti gotovo rješenje kao bazu za izradu završnog rada) te ga upravljati bežičnom Bluetooth komunikacijom zasnovanoj na Arduino platformi. Platformu odabrati prema složenosti zadatka. Platforma koristi joystick kao ulaznu jedinicu za upravljanje, dok će izlazni dio (izvršni članovi) biti motori. Model će imati mogućnost kretanja naprijed-natrag, te opciju skretanja. Potrebno je odabrati odgovarajuće ulazno-izlazne jedinice. Provesti testiranje rada pojedinih dijelova modela kao i cijelog sustava.

Rok za predaju rada: 20. rujna 2021.

Mentor:

Jurica Trstenjak, v. pred.

Predsjednik povjerenstva za završni ispit:

ZAHVALA

Najprije bih se želio zahvaliti svome mentoru Jurici Trstenjaku, v. pred., koji mi je pružio veliku čast da mi bude mentor prilikom izrade ovog rada. Hvala Vam na posvećenom vremenu i znanju koje ste mi prenijeli kroz godine studiranja.

Od srca se zahvaljujem svojim roditeljima, zaručnici, prijateljima i radnim kolegama koji su mi pružali podršku, bili uz mene i vjerovali u mene tijekom mog obrazovanja.

Petar Zidar

SAŽETAK

U radu je realiziran model automobila na daljinsko upravljanje baziran na Arduino platformi. U ovom sustavu koristi se Arduino UNO s ATmega328P mikrokontrolerom za upravljanje senzorima i ostalim komponentama od kojih se sastoji automobil. Pogon automobila bit će baziran na upravljačkoj ploči motora TB6612FNG. Na motor će se spojiti različiti senzori koji će omogućiti mnoštvo funkcija, kao što su praćenje linije koja je nacrtana na podu, izbjegavanje prepreka pomoću RGB senzora i senzora za izbjegavanje prepreka, praćenje svjetlosti te praćenje pokreta rukom. Podnica automobila bit će izrađena od šperploče debljine 5 mm na koju će biti pričvršćene sve komponente. Poklopac automobila, odnosno karoserija bit će modelirana na računalu te nakon toga isprintana pomoću 3D printera u dva dijela koji će se naknadno spojiti s čvrstim ljepilom. Materijal korišten za 3D printanje bit će plastika. Automobil za kretanje koristi 4 kotača od plastike i gume te se svaki kotač spaja na svoj zaseban motor. Automobilu Bluetooth senzor, koji je spojen na upravljačku ploču motora, omogućuje povezivanje i komunikaciju Arduina s pametnim telefonom. Pritiskom na dugme za paljenje pokrenut će se Arduino UNO i time cijeli sklop automobila. Nakon paljenja sklopa potrebno je upaliti aplikaciju na pametnom uređaju i povezati se pomoću Bluetootha. U sučelju aplikacije možemo odabrati hoćemo li samo upravljati automobilom ili možemo odabrati neku od funkcija kao što su praćenje linija, praćenje svjetlosti, praćenje pokreta ili izbjegavanje prepreka. U aplikaciji također možemo odabrati funkciju koja nam omogućuje upravljanje putem daljinskog upravljača. Ako odaberemo manualni način upravljanja, pomoću tipki u aplikaciji upravljamo automobilom te također možemo mijenjati brzinu kojom će se automobil kretati. U slučaju odabira automatskog upravljanja, odnosno opcije da se automobil kreće sam, automobil počinje koristiti ultrazvučni senzor za detekciju prepreka. Ultrazvučni senzor spojen je na servo motor koji mu omogućuje okretanje za 180 stupnjeva te odabir najbolje putanje kojom će izbjeći eventualnu prepreku.

***Ključne riječi**: Arduino, Arduino UNO, upravljačka ploča TB6612, Bluetooth, ultrazvučni senzor, senzor za praćenje linija, 3D printer*

SADRŽAJ

1. UVOD

Tema završnog rada je izrada automobila na daljinsko upravljanje baziranog na Arduino platformi. Navedeno će se postići korištenjem Arduino UNO ploče i različitih senzora koji će biti pričvršćeni na drvenu podnicu te će se na to zalijepiti plastična konstrukcija u obliku automobila. Komponente su kupljene putem internet trgovine, dok je podnica napravljena od šperploče korištenjem ubodne pile. Konstrukcija automobila modelirana je i isprintana pomoću 3D printera. Nakon nabavke dijelova, komponente se pričvršćuju na drvenu podnicu te spajaju s Arduino UNO pločom. Zatim slijedi programiranje te testiranje funkcija. Na samom kraju plastična konstrukcija se postavlja na drvenu podnicu kako bi projekt imao dizajn automobila.

2. CILJ RADA

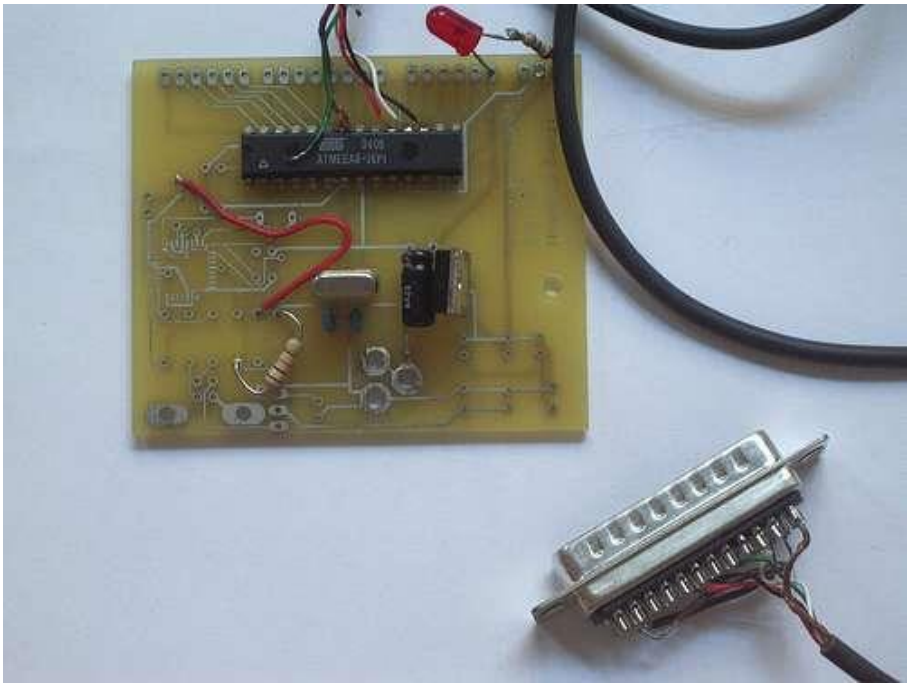
Cilj završnog rada je zainteresirati i upoznati čitatelje s mogućnostima koje nam nudi Arduino platforma kroz projekt izrade automobila na daljinsko upravljanje. Arduino je vrlo zanimljiva platforma zbog toga što je jednostavna za korištenje i pristupačne je cijene. Kroz izradu ovog završnog rada primijenit će se stečeno znanje iz područja elektrotehnike, programiranja te digitalnih i elektroničkih sklopova.

3. ARDUINO

Povijest Arduino uređaja započela je 2003. godine u gradu Ivrea u Italiji gdje se nalazi Institut dizajna interakcije (IDII). Student Hernardo Barragan kreirao je platformu *Wiring* kao projekt za završni rad pod vodstvom Massima Banzia i Caseya Reasa. Cilj projekta bio je izraditi alat za kreiranje digitalnih projekata koji će ne samo biti jeftini, nego i laki za korištenje i ljudima koji nisu inženjeri. *Wiring* platforma sastojala se od nekoliko komponenata i funkcija:

1. PCB – isprintana matična ploča (engl. *Printed Circuit Board*)
2. ATmega168 mikrokontroler
3. Integrirano razvojno okruženje – IDE
4. Funkcija biblioteke za jednostavno programiranje.

2005. godine Massimo Banzi je, uz pomoć studenata IDII instituta Davida Cuartiellesa, Davida Mellisa, Toma Igoea i Gianluce Martina, proširio platformu *Wiring* dodavši jeftiniji mikrokontroler ATmega8 i nazvao ga Arduino.



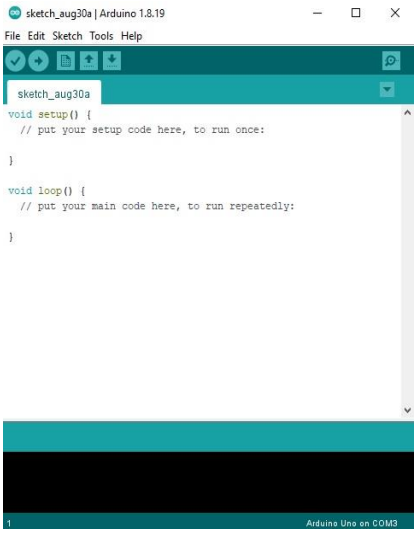
Slika 1. Prvi Arduino uređaj.
Izvor: https://www.ahirlabs.com/wp-content/uploads/2017/10/arduino_first.jpg (17.08.2021.)

Arduino platforma je platforma otvorenog koda (engl. *open source*) koja služi za kreiranje elektroničkih uređaja poput automobila na daljinsko upravljanje koji će biti opisan u ovom završnom radu. Arduino je vrlo popularan kod korisnika zahvaljujući jednostavnosti korištenja, ali i cijeni koja je, u usporedbi s drugim uređajima, vrlo jeftina. Dodatna pogodnost je to što se studentima i svima zainteresiranima može prikazati kako izraditi električne uređaje u vrlo kratkom roku. Arduino se sastoji od:

1. sklopovlja (engl. *hardware*), odnosno mikrokontrolera
2. modularne pločice (engl. *shields*)
3. programskog dijela (engl. *software*), odnosno integriranog razvojnog okruženja (IDE).

3.1. Programski dio Arduino platforme

Programski dio Arduino platforme, točnije integrirano razvojno okruženje (engl. *Integrated Development Environment*), sastoji se od uređivača teksta u kojeg korisnik upisuje programski kod, od tekstualne konzole, od programske trake na kojoj se nalaze gumbi s funkcijama za korištenje i nekoliko izbornika koji olakšavaju korištenje samog programa. On se spaja i komunicira sa sklopovljem putem USB-a (engl. *Universal Serial Bus*) kako bi se mogli prenijeti programi.



Slika 2. Arduino IDE.
Izvor: Autor.

4. MATERIJALI POTREBNI ZA IZRADU

U ovom poglavlju završnog rada opisat ćemo koje su sve komponente i materijali korišteni za izradu automobila na daljinsko upravljanje. U niže navedenoj tablici nalazi se popis svih komponenata za izradu ovog projekta.

Tablica 1. Materijali potrebni za izradu projekta.

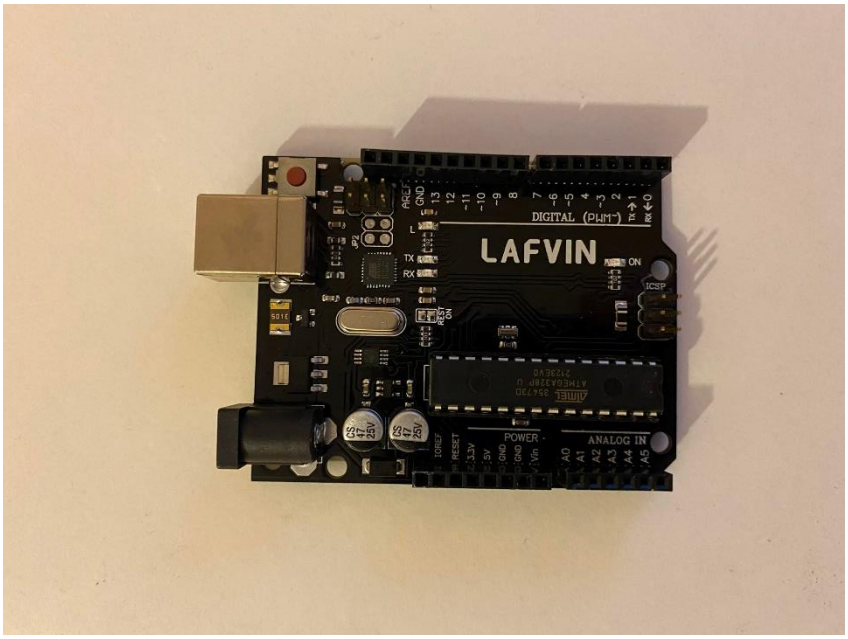
Naziv na hrvatskom jeziku	Naziv na engleskom jeziku	Broj komada
Arduino UNO	Arduino UNO	1
Upravljačka ploča TB6612FNG	Expanding board TB6612FNG	1
RGB ultrazvučni senzor	RGB ultrasound module	1
SG90 servo motor	SG90 servo motor	1
Bluetooth	Bluetooth module	1
Infracrveni senzor za izbjegavanje prepreka	IR avoid module	2
Senzor za praćenje linija	Line tracking module	3
Fotoelektrični senzor	Photoelectric module	2
TT 130 motor	TT 130 motor	4
Kotači	Wheels	4
Držač baterije	Battery holder	1
Žice	Wires	14

Izvor: Autor.

Osim u tablici navedenih komponenta, za izradu ovog projekta koristila se šperploča kao podnica automobila te se koristila konstrukcija koja je modelirana i isprintana pomoću 3D printera, a za pričvršćivanje komponenata na podnicu koristili su se vijci M3x12 mm s pripadajućim maticama.

4.1. Arduino Uno

Arduino UNO je jedna od najboljih i najpoznatijih ploča za početak rada s Arduino platformom i za izradu prvog projekta radi jednostavnosti, ali i mnoštva mogućnosti koje pruža. Arduino UNO je ploča mikrokontrolera baziranog na ATmega328P. Na ploči se nalazi ukupno 14 pinova. Od tih 14 pinova, njih 6 se može koristiti kao izlaz, a drugih 6 se može koristiti kao analogni ulaz. Ploča ima procesor frekvencije 16 MHz. Za povezivanje s računalom Arduino UNO ploča ima USB[1] konekciju i ona je jedna od prvih ploča koja nam nudi tu mogućnost. Za napajanje se nude mogućnosti napajanja pomoću baterija i napajanja pomoću AC/DC adaptera. Na Arduino UNO ploči također se nalazi i gumb za resetiranje s kojim možemo resetirati cijelu ploču ukoliko je to potrebno.



Slika 3. Arduino UNO.
Izvor: Autor.

[1] USB – služi za komunikaciju računala s vanjskim uređajem.

U tablici ispod mogu se vidjeti neke od specifikacija Arduino UNO ploče.

Tablica 2. Specifikacije Arduino UNO ploče.

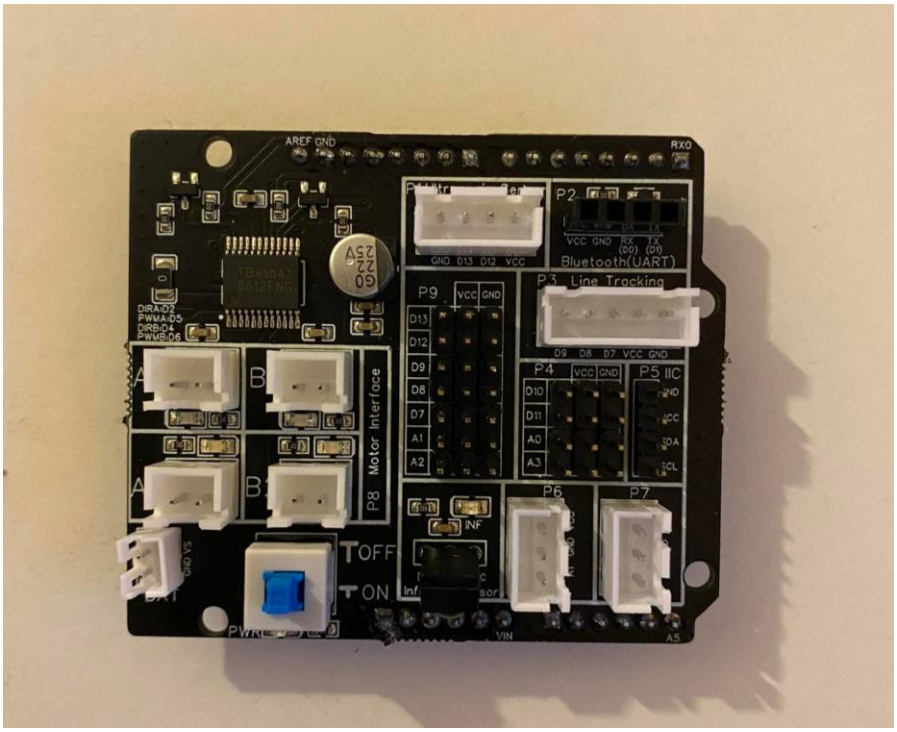
Mikrokontroler	ATmega328P
Operacijska voltaža	5 V
Ulazna voltaža (preporučena)	7-12 V
Ulazna voltaža (min/max)	6-20 V
Digitalni ulazni i izlazni pinovi	14 kom
PWN digitalni ulazni i izlazni pinovi	6 kom
Analogni ulazni pinovi	6 kom
DC struja po ulazno/izlaznom pinu	20 mA
DC struja za 3.3V pin	50 mA
Memorija	32 KB
SRAM	2 KB
EEPROM	1 KB
Brzina	16 MHz
Dužina/Širina	68.6*53.4 mm
Težina	25 g

Izvor: Autor.

4.2. Upravljačka ploča TB6612FNG

Na ovom projektu kao pogonski čip motora koristi se TB6612FNG koji se spaja na Arduino UNO ploču. TB6612FNG je proizvod koji proizvodi *Toshiba Semiconductor Corporation*. Upravljačka ploča TB6612FNG je *H-Bridge* motorni kontroler koji omogućuje upravljanje smjerom, dvokanalni izlaz strujnog kruga te u isto vrijeme može pokretati 2 motora. Ploča nudi 4 načina upravljanja motorom – naprijed, nazad, kočenje i zaustavljanje. Ploča u sebi ima ugrađeni

toplinski krug za isključivanje, kondenzatore za filtriranje na oba dovodna voda i zaštitu od povratnog napajanja na napajanju motora.



Slika 4. Upravljačka ploča TB6612FNG.
Izvor: Autor.

Tablica ispod prikazuje specifikacija upravljačke ploče TB6612FNG.

Tablica 3. Specifikacije upravljačke ploče TB6612FNG.

Maksimalan napon	15 V
Preporučen napon	4,5 – 13,5 V
Maksimalna izlazna struja	3 A po kanalu
Kontinuirana izlazna struja	1 A po kanalu
Maksimalna PWM frekvencija	100 kHz

Izvor: Autor.

Modulacija širine impulsa, odnosno PWM (engl. *Pulse Width Modulation*), ima vrijednost u dometu od 0 do 255. Što je veća vrijednost, motor se brže vrti.

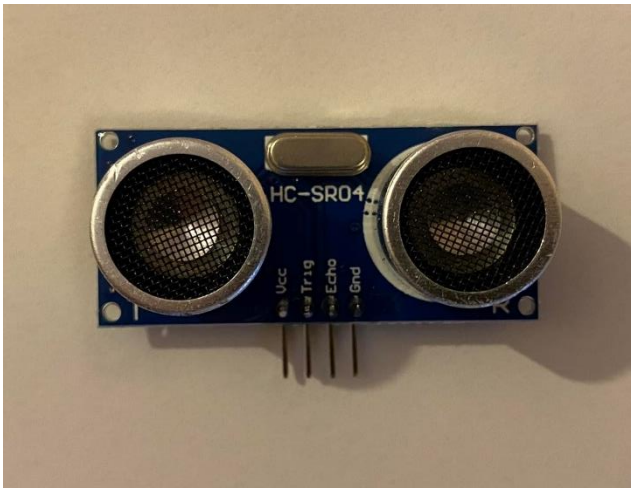
Tablica 4. Vrijednosti motora.

	D2	D5 (PWM)	D4	D6 (PWM)
Naprijed	HIGH	0 – 255	LOW	0 – 255
Nazad	LOW	0 – 255	HIGH	0 – 255
Lijevo	LOW	0 – 255	LOW	0 – 255
Desno	HIGH	0 – 255	HIGH	0 – 255
Stop	/	0	/	0

Izvor: Autor.

4.3. HC – SR04 Ultrazvučni senzor

Za utvrđivanje udaljenosti u ovom projektu se koristio HC-SR04 ultrazvučni senzor. On određuje udaljenost od objekta ispred sebe primanjem visokofrekventnih zvučnih valova, kao što to rade i šišmiši. Kao što možemo vidjeti na slici ispod, na njemu se nalaze dva objekta koja nalikuju očima – jedan objekt je odašiljatelj, a drugi objekt je primatelj. Ovaj senzor se koristi u puno projekata za pomoć pri utvrđivanju udaljenosti i otkrivanju prepreka.



Slika 5. HC-SR04 senzor.
Izvor: Autor.

U tablici ispod možemo vidjeti specifikacije HC-SR04 senzora.

Tablica 5. Specifikacije HC-SR04 senzora.

Napajanje	+ 5 V DC
Struja mirovanja	< 2 mA
Radna struja	15 mA
Kut učinkovitosti	< 15º
Domet	2 cm – 400 cm
Rezolucija	0,3 cm
Kut mjerenja	30º
Širina ulaznog impulsa okidača	10 uS

Izvor: Autor.

4.3.1 Izvorni kôd za HC-SR04 senzor

```
void Ultrasonic_Avoidance() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    int Front_Distance = 0;
    int Left_Distance = 0;
    int Right_Distance = 0;
    int Right_IR_Value = 1;
    int Left_IR_Value = 1;
    Left_IR_Value = digitalRead(A1);
    Right_IR_Value = digitalRead(A2);
    Front_Distance = checkdistance();
    Serial.println(Front_Distance);
    if (Left_IR_Value == 0 && Right_IR_Value == 1) {
      digitalWrite(2,HIGH);
      analogWrite(5,255);
      digitalWrite(4,LOW);
      analogWrite(6,12);

    } else if (Left_IR_Value == 1 && Right_IR_Value == 0) {
      digitalWrite(2,HIGH);
      analogWrite(5,12);
```

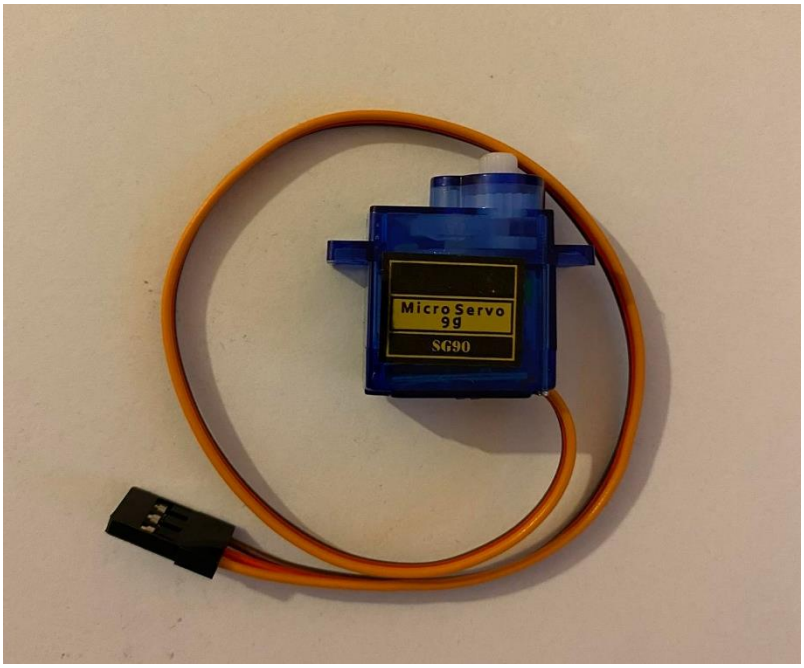
```
      digitalWrite(4,LOW);
      analogWrite(6,255);
    } else {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 22.5));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 22.5));

    }
    if (Front_Distance <= D_mid) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
      if  (Front_Distance  <=  D_mix  ||  Left_IR_Value  ==  0  &&
Right_IR_Value == 0) {
        digitalWrite(2,LOW);
        analogWrite(5,(4.5 * 22.5));
        digitalWrite(4,HIGH);
        analogWrite(6,(4.5 * 22.5));
        delay(300);
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);

      }
      myservo.write(165);
      delay(500);
      Serial.println(Left_Distance);
      delay(100);
      Left_Distance = checkdistance();
      myservo.write(15);
      delay(500);
      Serial.println(Right_Distance);
      delay(100);
      Right_Distance = checkdistance();
      myservo.write(90);
      if ((D_mix < Left_Distance && Left_Distance < D_max) && (D_mix
< Right_Distance && Right_Distance < D_max)) {
        if (Right_Distance > Left_Distance) {
          digitalWrite(2,HIGH);
          analogWrite(5,(9 * 22.5));
          digitalWrite(4,HIGH);
          analogWrite(6,(9 * 22.5));
          delay(250);

        } else {
          digitalWrite(2,LOW);
          analogWrite(5,(9 * 22.5));
          digitalWrite(4,LOW);
```

```
          analogWrite(6,(9 * 22.5));
          delay(250);

        }

      } else if (D_mix < Left_Distance && Left_Distance < D_max ||
D_mix < Right_Distance && Right_Distance < D_max) {
        if (D_mix < Left_Distance && Left_Distance < D_max) {
          digitalWrite(2,LOW);
          analogWrite(5,(7 * 22.5));
          digitalWrite(4,LOW);
          analogWrite(6,(7 * 22.5));
          delay(250);

        } else if (D_mix < Right_Distance && Right_Distance < D_max)
{
          digitalWrite(2,HIGH);
          analogWrite(5,(7 * 22.5));
          digitalWrite(4,HIGH);
          analogWrite(6,(7 * 22.5));
          delay(250);
        }
      } else if (Right_Distance < D_mix && Left_Distance < D_mix) {
        digitalWrite(2,HIGH);
        analogWrite(5,0);
        digitalWrite(4,LOW);
        analogWrite(6,(9 * 22.5));
        delay(510);
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);
      }
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if    ('%'    ==    String(BLE_value).charAt(0)    &&    'Q'    ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
```

```
        }
    }
}
```

4.4. SG90 servo motor

Servo motor se koristi kako bi se ultrazvučni senzor iz prethodnog poglavlja mogao okretati da bi bolje odredio gdje se nalaze prepreke. Njegov raspon kuta rotacije je od 0 stupnjeva do 180 stupnjeva. U ovom radu se za to koristio SG90 servo motor koji se sastoji od kućišta, ploče za strujni krug, motora bez jezgre, senzora položaja i zupčanika. On radi tako da prima signal od prijamnika te nakon toga proizvodi signal s periodom od 20 ms i širine 1,5 ms. Nakon toga uspoređuje istosmjerni pred napon s naponom potenciometra i dobiva izlaznu razliku napona.



Slika 6. SG90 servo motor.
Izvor: Autor.

U tablici ispod možemo vidjeti specifikacije SG90 servo motora.

Tablica 6. Specifikacije SG90 servo motora.

Kontrola	Analogna
Napon	5 V

Brzina	60º za 0,1 s
Težina	9 g
Dimenzije	23*12*29 mm

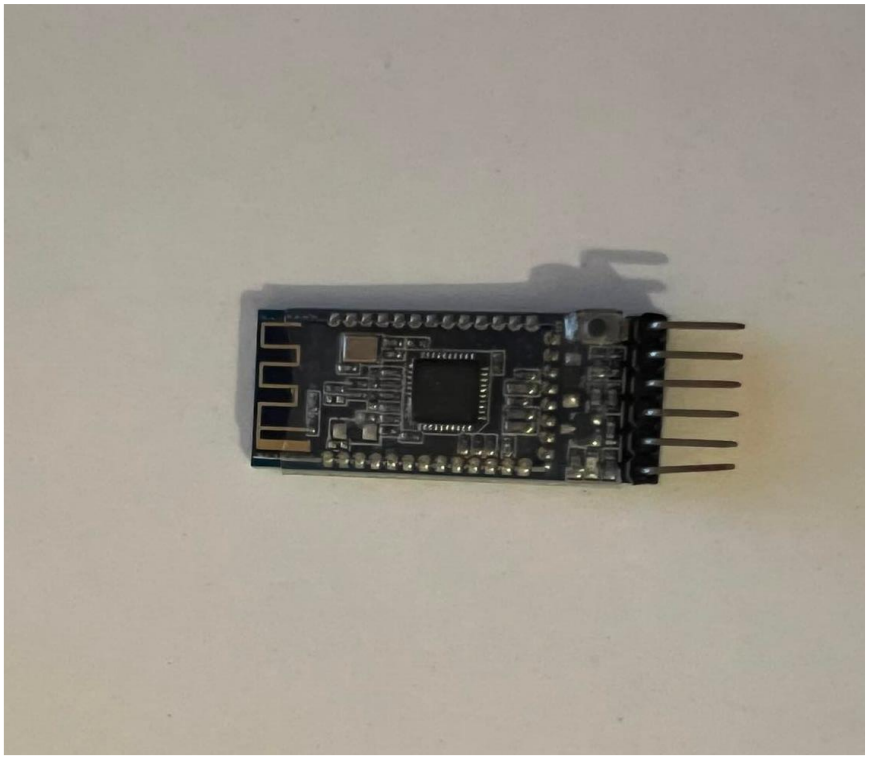
Izvor: Autor.

4.4.1 Izvorni kôd za servo motor

```
Servo myservo;
float checkdistance() {
  digitalWrite(12, LOW);
  delayMicroseconds(2);
  digitalWrite(12, HIGH);
  delayMicroseconds(10);
  digitalWrite(12, LOW);
  float distance = pulseIn(13, HIGH) / 58.00;
  delay(10);
  return distance;
}
```

4.5. Bluetooth JDY-16

U izradi ovog projekta za komunikaciju između Arduina i drugog pametnog uređaja, u ovom slučaju pametnog telefona, koristio se Bluetooth dodatak (engl. *shield*) modela JDY-16. Prijenosni modul JDY-16, temelji se na protokolu standarda 4.2 koji je kompatibilan s Bluetoothom. U sebi ima ugrađenu PCB (engl. *Printed Circuit Board*) antenu. JDY-16 Bluetooth može ostvariti prijenos podataka ne samo između senzora i pametnog uređaja, nego i između senzora i drugog senzora što znači da može ostvariti komunikaciju dva Arduino uređaja. Spajanje na Bluetooth uređaj vrlo je jednostavno – potrebno je spojiti uređaj na upravljačku ploču, na pametnom uređaju odabrati Bluetooth JDY-16 te ih povezati.



Slika 7. JDY-16 Bluetooth senzor.
Izvor: Autor.

U tablici ispod se nalaze karakteristike JDY-16 Bluetootha.

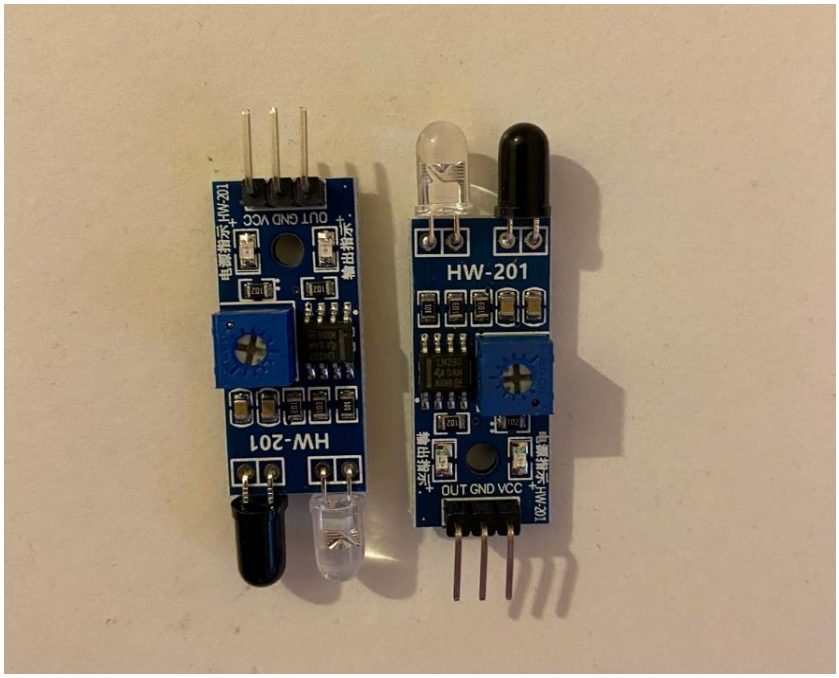
Tablica 7. Specifikacije JDY-16 Bluetooth senzora.

Radna frekvencija	2,4 G
Sučelje za komunikaciju	UART ili IIC
Radni napon	1,8 V – 3,6 V
Radna temperatura	-40º C - 80º C
Udaljenost prijenosa	60 metara
Osjetljivost primanja	-97 dbm
Transparentna brzina prijenosa	115200 bps/s

Izvor: Autor.

4.6. Infracrveni senzor za izbjegavanje prepreka

Infracrveni senzor za detekciju prepreka na sebi ima par infracrvenih dioda za odašiljanje i primanje kako bi obavio svoj posao. Kad senzor naiđe na neki objekt ili na prepreku odašiljač emitira infracrvene zrake određene frekvencije. Infracrvene zrake se zatim reflektiraju natrag gdje ih prima dioda za primanje. Kad se to dogodi na senzoru će početi svijetliti indikator, odnosno zeleno LED[2] (engl. *Light Emitting Diode*) svijetlo. Nakon što strujni krug obradi tu informaciju, terminal signala će početi emitirati digitalni signal. Na uređaju postoji potenciometar gdje možemo podešavati udaljenost detekcije. Za izradu ovog projekta koristila su se dva takva senzora, jedan na lijevoj i jedan na desnoj strani za bolju i točniju detekciju prepreke.



Slika 8. Infracrveni senzor za izbjegavanje prepreka.
Izvor: Autor.

U tablici koja slijedi možemo vidjeti specifikacije infracrvenog senzora za izbjegavanje prepreka.

[2] LED – poluvodički elektronički element koji pretvara električni u optički signal.

Tablica 8. Specifikacije infracrvenog senzora za izbjegavanje prepreka.

Napon	3 V – 5 V DC
Udaljenost detekcije	2 – 30 cm
Udaljenost kuta	35°

Izvor: Autor.

4.6.1. Izvorni kôd za senzor za izbjegavanje prepreka

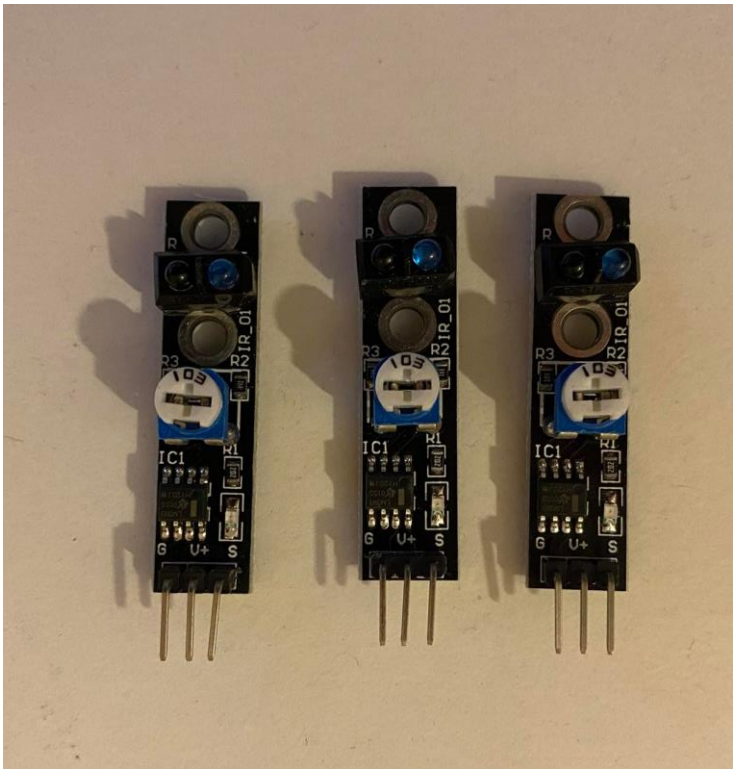
```
void Ultrasonic_Follow() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    int Infrared_Trigger_Flag = 0;
    int Front_Distance = 0;
    int Left_Distance = 0;
    int Right_Distance = 0;
    int Right_IR_Value = 1;
    int Left_IR_Value = 1;
    Left_IR_Value = digitalRead(A1);
    Right_IR_Value = digitalRead(A2);
    Front_Distance = checkdistance();
    if (Front_Distance < 5 && (Left_IR_Value != Infrared_Trigger_Flag
&& Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(3 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(3 * 25.5));

    } else if (Front_Distance < 5 && (Left_IR_Value ==
Infrared_Trigger_Flag && Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(0.056 * (4 * 255)));
    } else if (Front_Distance < 5 && (Left_IR_Value !=
Infrared_Trigger_Flag && Right_IR_Value == Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(0.056 * (4 * 255)));
      digitalWrite(4,HIGH);
      analogWrite(6,(4 * 25.5));
    } else if (Front_Distance < 5 && (Left_IR_Value ==
Infrared_Trigger_Flag && Right_IR_Value == Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(3 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(3 * 25.5));
```

```
    } else if (Front_Distance > 10 && (Left_IR_Value !=
Infrared_Trigger_Flag && Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 25.5));
    } else if (Front_Distance > 10 && (Left_IR_Value ==
Infrared_Trigger_Flag && Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 25.5));
    } else if (Front_Distance > 10 && (Left_IR_Value !=
Infrared_Trigger_Flag && Right_IR_Value == Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(4 * 25.5));
    } else if ((5 <= Front_Distance && Front_Distance <= 10) &&
(Left_IR_Value != Infrared_Trigger_Flag && Right_IR_Value ==
Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,LOW);
      analogWrite(6,(0.056 * (4 * 25.5)));
    } else if ((5 <= Front_Distance && Front_Distance <= 10) &&
(Left_IR_Value == Infrared_Trigger_Flag && Right_IR_Value !=
Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(0.056 * (4 * 25.5)));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 25.5));
    } else if ((5 <= Front_Distance && Front_Distance <= 10) &&
(Left_IR_Value != Infrared_Trigger_Flag && Right_IR_Value !=
Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if ('%' == String(BLE_value).charAt(0) && 'Q' ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
```

```
        analogWrite(6,0);

      }
    }
}
```

4.7. Senzor za praćenje linija

Jedna od mogućnosti koju će imati automobil iz ovog projekta je praćenje nacrtane linije. Za to ćemo koristiti 3 senzora za praćenje linija (lijevi, srednji i desni). Senzor za praćenje linija je zapravo infracrveni senzor. Komponenta koja je korištena u izradi ovog rada je infracrvena dioda TCRT5000. Princip rada senzora čini korištenje različite refleksije infracrvenog svijetla u odnosu na boju, zatim pretvaranje snage reflektiranog signala u trenutni signal. U ovom slučaju riječ je o crnoj boji. Kad senzor primijeti crnu boju na signalni pin senzora izlazi HIGH (1), a status na LED svijetlu je isključen. Kad senzor primijeti bijelu boju događa se suprotno, na signalni pin senzora izlazi LOW (0), a status na LED svijetlu je uključen.



Slika 9. Senzori za praćenje linija.
Izvor: Autor.

U tablici ispod možemo vidjeti specifikacije za senzor za praćenje linija.

Tablica 9. Specifikacije senzora za praćenje linija

Radni napon	3,3 V – 5 V
Izlazni signal	Digitalan
Visina detekcije	0 – 3 cm

Izvor: Autor.

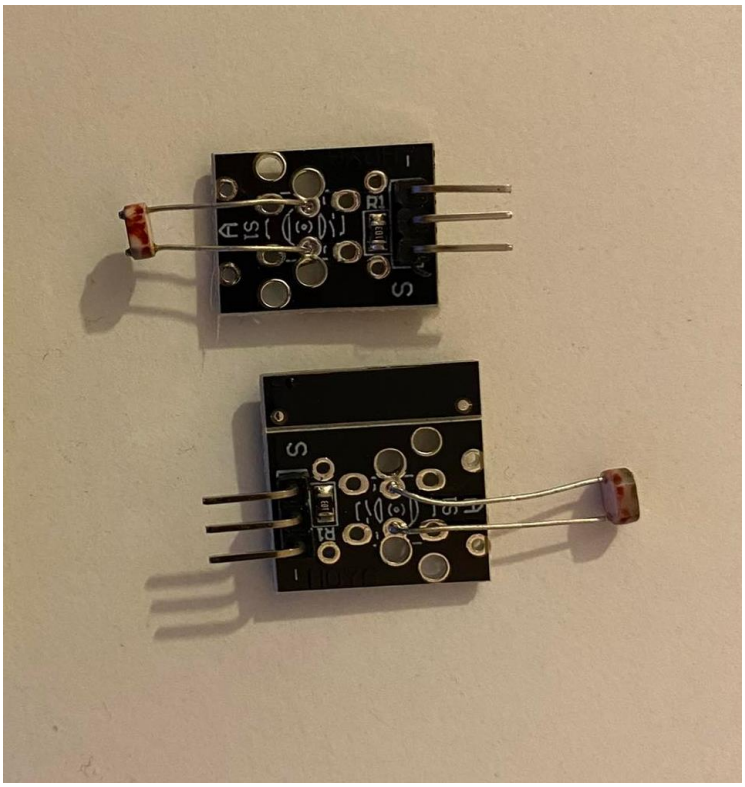
4.7.1. Izvorni kôd za senzor za praćenje linija

```
void Infrared_Tracing() {
  Funtion_FLag = true;
  int Left_Tra_Value = 1;
  int Center_Tra_Value = 1;
  int Right_Tra_Value = 1;
  int Black = 1;
  int white = 0;
  while (Funtion_FLag) {
    Left_Tra_Value = digitalRead(7);
    Center_Tra_Value = digitalRead(8);
    Right_Tra_Value = digitalRead(9);
    if (Left_Tra_Value != Black && (Center_Tra_Value == Black &&
Right_Tra_Value != Black)) {
      digitalWrite(2,HIGH);
      analogWrite(5,120);
      digitalWrite(4,LOW);
      analogWrite(6,120);

    } else if (Left_Tra_Value == Black && (Center_Tra_Value == Black
&& Right_Tra_Value != Black)) {
      digitalWrite(2,LOW);
      analogWrite(5,120);
      digitalWrite(4,LOW);
      analogWrite(6,120);
    } else if (Left_Tra_Value == Black && (Center_Tra_Value != Black
&& Right_Tra_Value != Black)) {
      digitalWrite(2,LOW);
      analogWrite(5,80);
      digitalWrite(4,LOW);
      analogWrite(6,80);
    } else if (Left_Tra_Value != Black && (Center_Tra_Value != Black
&& Right_Tra_Value == Black)) {
      digitalWrite(2,HIGH);
```

```
      analogWrite(5,80);
      digitalWrite(4,HIGH);
      analogWrite(6,80);
    } else if (Left_Tra_Value != Black && (Center_Tra_Value == Black
&& Right_Tra_Value == Black)) {
      digitalWrite(2,HIGH);
      analogWrite(5,120);
      digitalWrite(4,HIGH);
      analogWrite(6,120);
    } else if (Left_Tra_Value == Black && (Center_Tra_Value == Black
&& Right_Tra_Value == Black)) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    } else if (false) {
    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if    ('%'    ==    String(BLE_value).charAt(0)    &&    'Q'    ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}
```

4.8. Fotoelektrični senzor

Fotoelektrični senzor u ovom projektu radi tako da kad senzor prepozna svijetlo automobil ga automatski počne pratiti. Zbog toga su u ovom radu korištena dva senzora, jedan s lijeve i jedan s desne strane. Kad lijevi senzor prepozna da je jače svijetlo s njegove strane počet će se kretati u lijevo. Isto tako, kad desni senzor prepozna svijetlo počet će se kretati u desno. Fotoelektrični senzor je poluvodički rezistor izrađen fotoelektričnim efektom. Vrlo je osjetljiv na ambijentalno svjetlo, pa mu se vrijednost otpora mijenja s obzirom na intenzitet svjetla. Kad se intenzitet svjetla poveća, otpor se smanjuje i izlazni napon se smanjuje. U slučaju u kojem se intenzitet svjetla smanji, otpor se povećava, a izlazni napon raste.



Slika 10. Fotoelektrični senzor.
Izvor: Autor.

U tablici ispod možemo vidjeti specifikacije za fotoelektrični senzor.

Tablica 10. Specifikacije fotoelektričnog senzora.

Kompilator	LM393
Radni napon	5V DC
Napajanje	20 mA

Izvor: Autor.

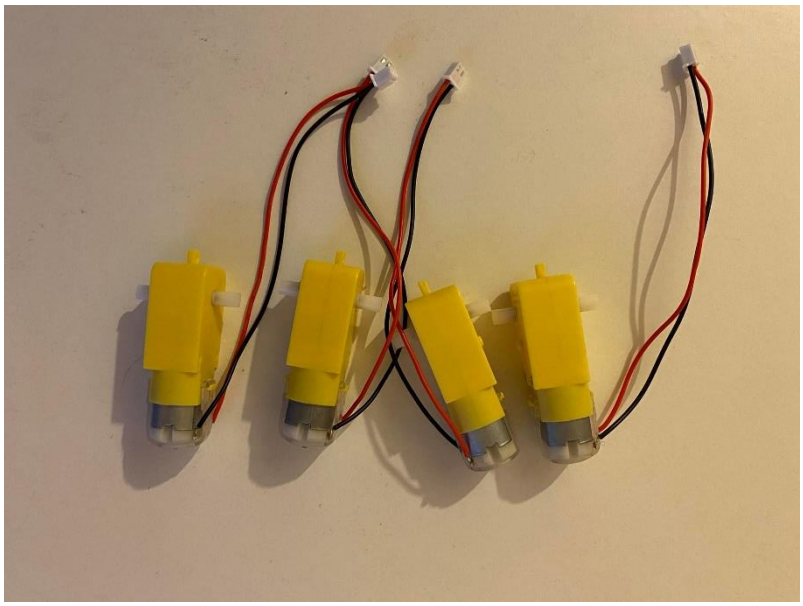
4.8.1. Izvorni kôd za fotoelektrični senzor

```
void Light_Seeking() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    Left_photosensitive = analogRead(A0) / 10;
    Right_photosensitive = analogRead(A3) / 10;
    if (Left_photosensitive > 55 && Right_photosensitive > 55) {
```

```
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    } else {
      if (Left_photosensitive > Right_photosensitive) {
        Lightseeking_Degree    =    ((float)(Right_photosensitive    /
Left_photosensitive)) * 90;
        Serial.println("float:");
        Serial.println(((float)(Right_photosensitive                /
Left_photosensitive)));

      } else if (Left_photosensitive <= Right_photosensitive) {
        Lightseeking_Degree  =  180  -  ((float)(Left_photosensitive  /
Right_photosensitive)) * 90;
      }
      if (Lightseeking_Degree <= 90) {
        f = ((float)(Lightseeking_Degree)) / 90;
        digitalWrite(2,HIGH);
        analogWrite(5,speed_value);
        digitalWrite(4,LOW);
        analogWrite(6,(speed_value * f));

      }
      if (Lightseeking_Degree > 90) {
        f = ((float)(180 - Lightseeking_Degree)) / 90;
        digitalWrite(2,HIGH);
        analogWrite(5,(speed_value * f));
        digitalWrite(4,LOW);
        analogWrite(6,speed_value);

      }

    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if    ('%'    ==    String(BLE_value).charAt(0)    &&    'Q'    ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}
```

4.9. TT 130 Motor

TT 130 Motor je istosmjerni motor koji se koristi za pokretanje kotača na automobilu na daljinsko upravljanje. U ovom projektu ćemo koristiti 4 takva motora, što znači da svaki kotač ima svoj zasebni motor. Na upravljačkoj ploči sučelje se sastoji od ulaza A1, A2, B1 i B2. Motori spojeni na sučelje A1 i A2 imaju istu brzinu i isti smjer. Isto tako, motori spojeni na B1 i B2 imaju istu brzinu i isti smjer. D2 ulazno-izlazni port kontrolira smjer portova A, dok D4 ulazno-izlazni port kontrolira brzinu portova A. D4 ulazno-izlazni port kontrolira smjer portova B, a D6 ulazno-izlazni port kontrolira brzinu portova B.



Slika 11. TT 130 Motori.
Izvor: Autor.

U tablici ispod možemo vidjeti specifikacije za TT 130 Motor.

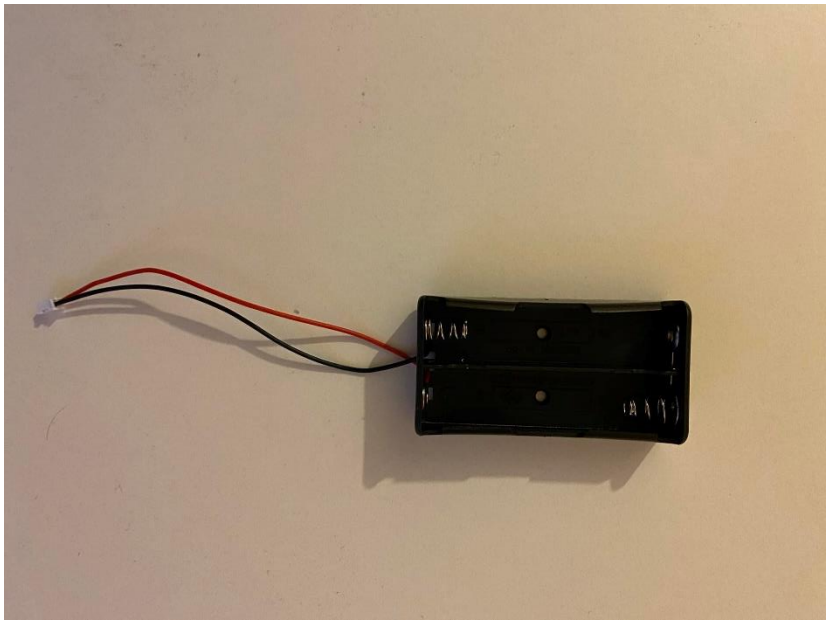
Tablica 11. Specifikacije TT 130 motora.

Operativna voltaža	3 – 12 V
Okretni moment	800 gf
Brzina praznog hoda	1:48 s
Struja opterećenja	70 – 250 mA

Izvor: Autor.

4.10. Napajanje

Napajanje motora vrši se pomoću dvije litijeve baterije (18650) koje su stavljene u držač baterija koji je spojen na upravljačku ploču motora. Osim baterija, Arduino UNO na sebi ima priključak za AC/DC adapter te je stoga napajanje moguće i pomoću adaptera.



Slika 12. Držač 18650 baterija.
Izvor: Autor.

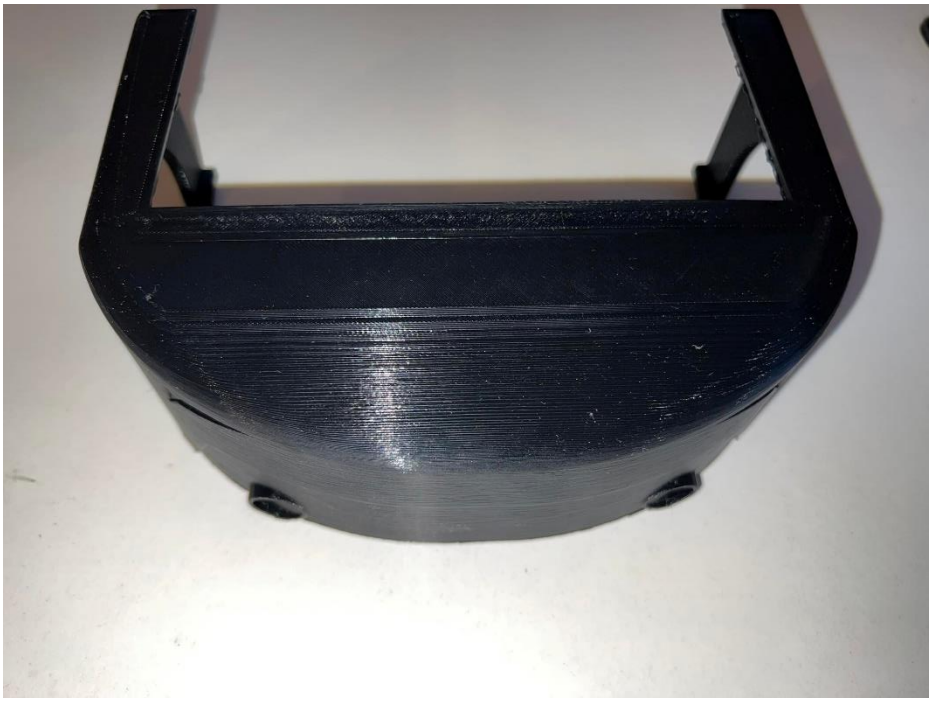
5. KONSTRUKCIJA

Za konstrukciju automobila koristilo se drvo i plastika. Podnica automobila napravljena od šperploče dimenzija 500x500x5 mm. Podnica je naknadno izrezana po mjeri u potreban oblik automobila pomoću ubodne pile. Nakon što je ploča izrezana na potreban oblik, na njoj su se prvo izbušile potrebne rupe za pričvršćivanje komponenata te je obojana u crnu boju.

Slika 13. Podnica auta.
Izvor: Autor.

Drugi dio konstrukcije je modeliran u program *Prusa Slicer*. Nakon toga je isprintan pomoću 3D printera modela Prusa i3 MK3S+ u dva dijela. Isprintan je u dva dijela jer printer nije bio dovoljno velik da bi isprintao sve u cjelini. Printanje komponenti trajalo je 22 sata po komponenti.

Slika 14. Prednji dio 3D modela.
Izvor: Autor.



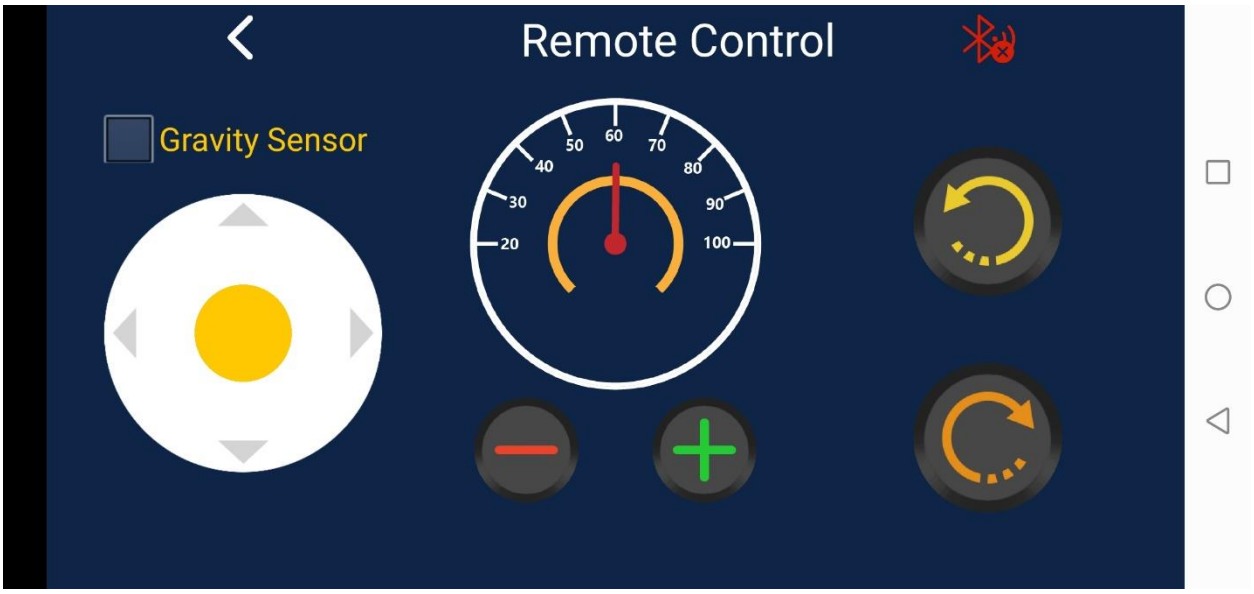
Slika 15. Stražnji dio 3D modela.
Izvor: Autor.

Nakon što je bilo gotovo i podnožje i konstrukcija automobila, obje komponente su spojene s čvrstim ljepilom koje se koristi u industrijskoj proizvodnji za lijepljenje plastike.

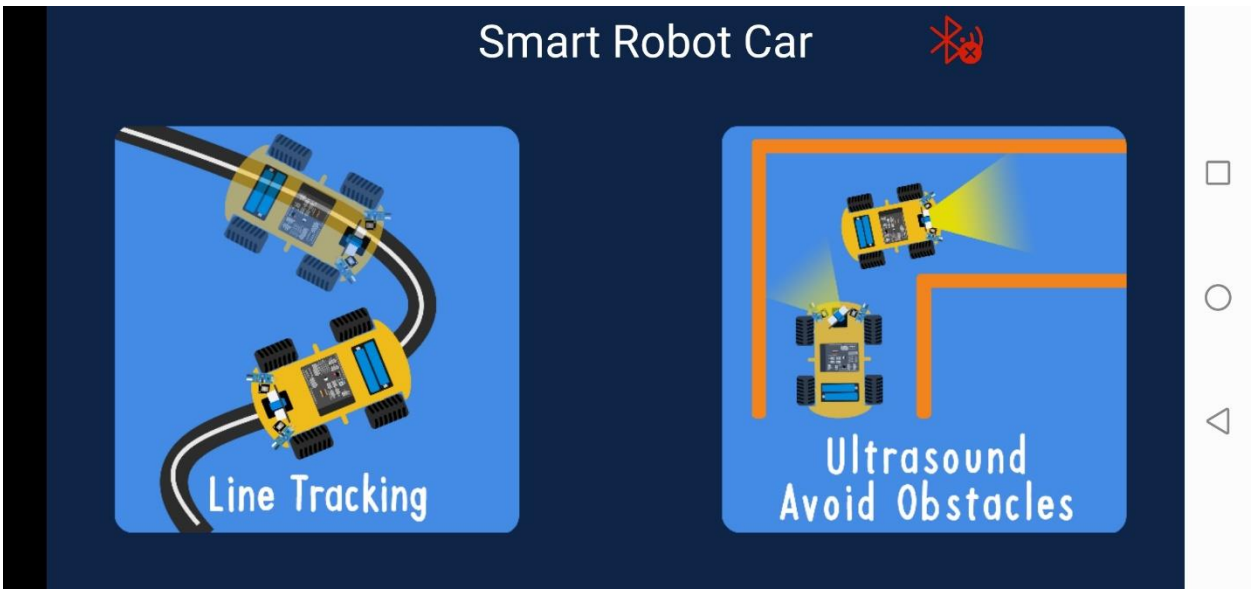
Slika 16. Auto u cjelini.
Izvor: Autor.

6. APLIKACIJA ZA UPRAVLJANJE AUTOMOBILOM

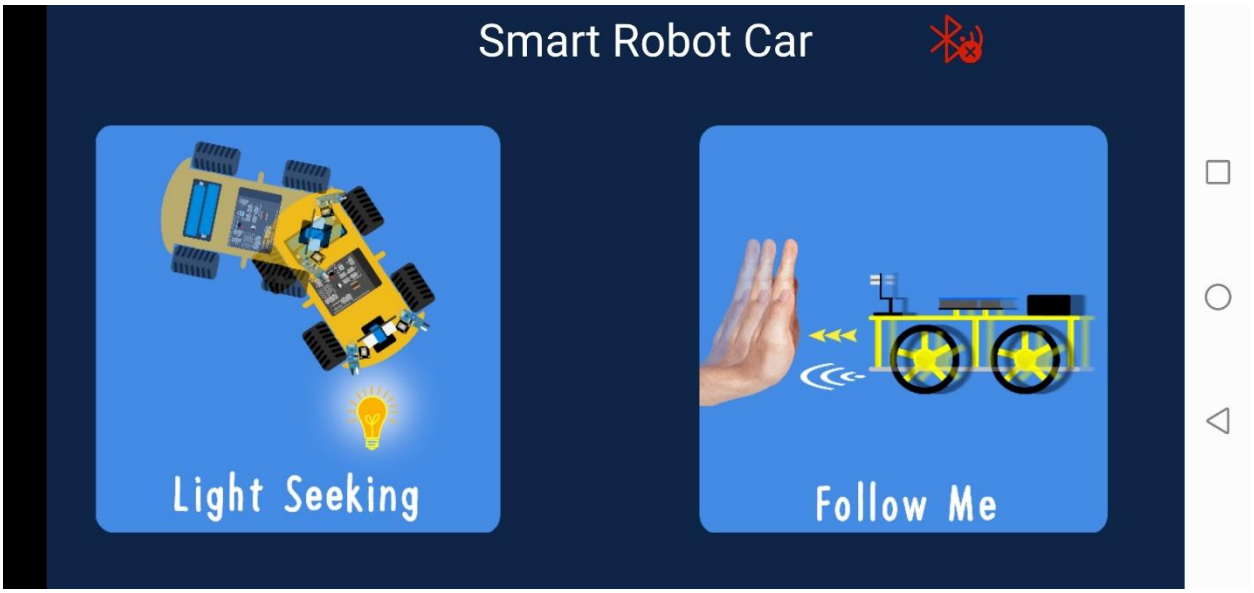
Za upravljanje automobilom koristi se mobilna aplikacija *Smart Robot Car*. Aplikacija pruža mogućnost manualne vožnje automobila te možemo, osim upravljanja, mijenjati brzinu automobila. Postoji i mogućnost uključivanja upravljanja automobilom fizičkim pokretom mobilnog uređaja. Aplikacija nam također nudi automatsko pokretanje automobila u funkcijama praćenja linija, izbjegavanja prepreka, praćenja svjetlosti i praćenja pokreta.



Slika 17. Sučelje aplikacije za manualno upravljanje.
Izvor: Autor.

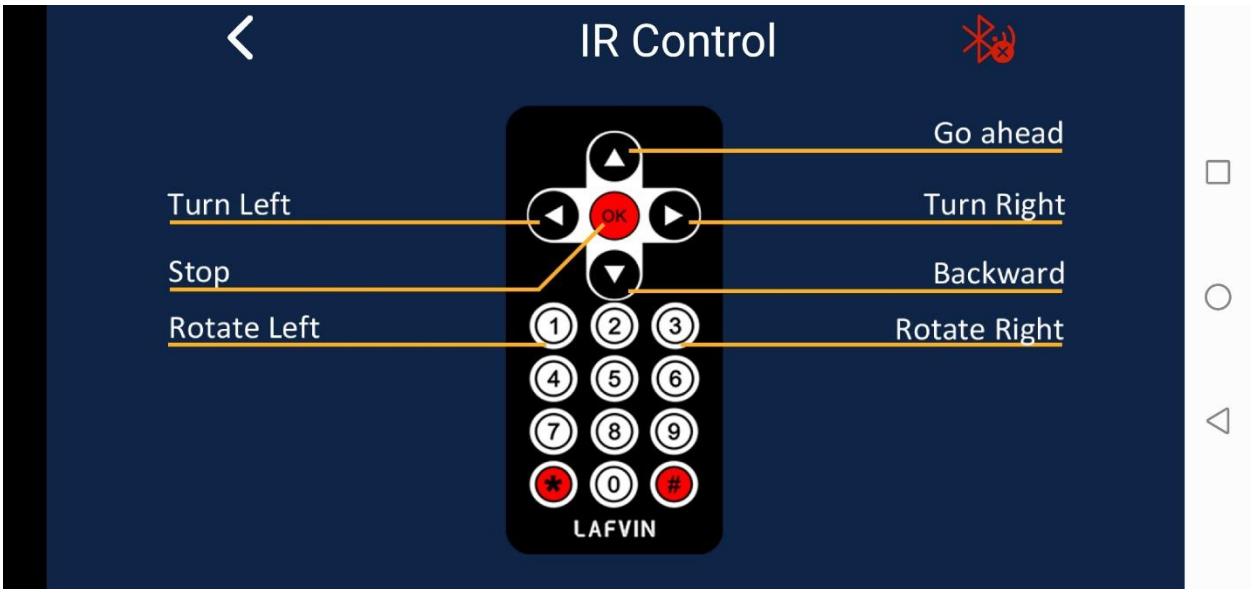


Slika 18. Sučelje aplikacije za odabir funkcija.
Izvor: Autor.



Slika 19. Sučelje aplikacije za odabir funkcija.
Izvor: Autor.

Za upravljanje se također može koristiti i daljinski upravljač pomoću infracrvenog senzora koji je ugrađen na upravljačku ploču TB6612FNG.



Slika 20. Kontrole za daljinski upravljač.
Izvor: Autor.

7. ZAKLJUČAK

Tijekom izrade ovog rada prikazano je nekoliko od mnoštva mogućnosti koje nam nudi Arduino platforma. Zbog svoje jednostavnosti platforma je idealna za korištenje prilikom izrade projekata kako početnicima tako i iskusnijima. Osim svoje jednostavnosti, kod Arduina se ističe njegova pristupačna cijena. Arduino platforma odlična je za početak učenja ili nadogradnju znanja iz područja elektrotehnike, digitalnih i elektroničkih sklopova, programiranja te robotike. Navedeno je u ovom radu prikazano kroz izradu automobila na daljinsko upravljanje. Za izradu ovog projekta koristili su se senzori za praćenje linija, senzori za izbjegavanje prepreka, fotoelektrični senzor i mnoštvo drugih. Upravljanje automobilom moguće je pomoću pametnog uređaja nakon spajanja s Bluetooth senzorom ili daljinskim uređajem koji se spaja s infracrvenim senzorom na upravljačkoj ploči. Za ovaj automobil postoji mogućnost nadogradnje, poput dodavanja web kamere koja bi nam omogućila da na zaslonu pametnog uređaja gledamo kuda se automobil kreće kad ga vozač ne može vidjeti. Također, jedna od nadogradnji bila bi stavljanje dvije LED žarulje s prednje strane što bi nam omogućilo lakšu vožnju po mraku. Arduino platforma je idealna zbog toga što korisnicima uvijek nudi mogućnost nadogradnje projekata.

8. POPIS LITERATURE

[1] Circuitstoday.com. *Story and History of Development of Arduino*, https://www.circuitstoday.com/story-and-history-of-development-of-arduino (07.08.2021.)

[2] Sgbotic.com. *Infrared Obstacle Avoidance Sensor*, https://www.sgbotic.com/index.php?dispatch=products.view&product_id=2527 (17.06.2022.)

[3] Cytron.io. *IR Line Trancking Sensor (Single Bit)*, https://www.cytron.io/p-ir-line-tracking-sensor-single-bit (17.06.2022.)

[4] Octopart.com. *Arduino UNO*, https://datasheet.octopart.com/A000066-Arduino-datasheet-38879526.pdf (19.06.2022.)

[5] Brighton12.com. *Program 4: Ultrasonic Sensor*, https://www.brightonk12.com/cms/lib/MI02209968/Centricity/Domain/517/Arduino%20Program%204%20Ultrasonica.pdf (23.07.2022.)

[6] E-radionica.com. *TB6612FNG H-BRIDGE DRIVER*, https://e-radionica.com/hr/tb6612fng-hbridge-driver.html (10.07.2022.)

8.1. Popis slika

8.2. Popis tablica

8.3. Popis kratica

RGB (engl. *Red Green Blue*) – aditivni model boja.

MOSFET (engl. *Metal Oxide Semiconductor Field Effect Transistor*) – metalni oksidni poluvodički tranzistor s efektom polja.

PCB (engl. *Printed Circuit Board*) – tiskana ploča.

IDE (engl. *Integrated Development Environment*) – integrirano razvojno okruženje.

USB (engl. *Universal Serial Bus*) – univerzalna serijska sabirnica.

IR (engl. *InfraRed*) – infracrveno.

PWN (engl. *Pulse Width Modulation*) – modulacija širine impulse.

LED (engl. *Light Emitting Diode*) – poluvodički elektronički element.

9. PRILOZI

9.1. Izvorni kod

```
#include "IR_remote.h"
#include "keymap.h"

IRremote ir(3);

#include <Servo.h>

volatile char BLE_bit_temp;
String BLE_value;
String G_Bluetooth_value;
volatile int G_degree;
volatile int re_string_len;
volatile float Left_photosensitive;
volatile float Right_photosensitive;
volatile int Lightseeking_Degree;
volatile float f;
volatile int speed_value;
volatile int D_mix;
volatile int D_mid;
volatile int D_max;
volatile boolean Funtion_FLag;
volatile int G_string_len;
volatile int BLE_Change_SPEED;

Servo myservo;
float checkdistance() {
  digitalWrite(12, LOW);
  delayMicroseconds(2);
  digitalWrite(12, HIGH);
  delayMicroseconds(10);
  digitalWrite(12, LOW);
  float distance = pulseIn(13, HIGH) / 58.00;
  delay(10);
  return distance;
}

void Ultrasonic_Avoidance() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    int Front_Distance = 0;
    int Left_Distance = 0;
    int Right_Distance = 0;
    int Right_IR_Value = 1;
    int Left_IR_Value = 1;
    Left_IR_Value = digitalRead(A1);
    Right_IR_Value = digitalRead(A2);
```

```
    Front_Distance = checkdistance();
    Serial.println(Front_Distance);
    if (Left_IR_Value == 0 && Right_IR_Value == 1) {
      digitalWrite(2,HIGH);
      analogWrite(5,255);
      digitalWrite(4,LOW);
      analogWrite(6,12);

    } else if (Left_IR_Value == 1 && Right_IR_Value == 0) {
      digitalWrite(2,HIGH);
      analogWrite(5,12);
      digitalWrite(4,LOW);
      analogWrite(6,255);
    } else {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 22.5));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 22.5));

    }
    if (Front_Distance <= D_mid) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
      if (Front_Distance <= D_mix || Left_IR_Value == 0 &&
Right_IR_Value == 0) {
        digitalWrite(2,LOW);
        analogWrite(5,(4.5 * 22.5));
        digitalWrite(4,HIGH);
        analogWrite(6,(4.5 * 22.5));
        delay(300);
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);

      }
      myservo.write(165);
      delay(500);
      Serial.println(Left_Distance);
      delay(100);
      Left_Distance = checkdistance();
      myservo.write(15);
      delay(500);
      Serial.println(Right_Distance);
      delay(100);
      Right_Distance = checkdistance();
      myservo.write(90);
      if ((D_mix < Left_Distance && Left_Distance < D_max) && (D_mix
< Right_Distance && Right_Distance < D_max)) {
```

```
        if (Right_Distance > Left_Distance) {
          digitalWrite(2,HIGH);
          analogWrite(5,(9 * 22.5));
          digitalWrite(4,HIGH);
          analogWrite(6,(9 * 22.5));
          delay(250);

        } else {
          digitalWrite(2,LOW);
          analogWrite(5,(9 * 22.5));
          digitalWrite(4,LOW);
          analogWrite(6,(9 * 22.5));
          delay(250);

        }

      } else if (D_mix < Left_Distance && Left_Distance < D_max ||
D_mix < Right_Distance && Right_Distance < D_max) {
        if (D_mix < Left_Distance && Left_Distance < D_max) {
          digitalWrite(2,LOW);
          analogWrite(5,(7 * 22.5));
          digitalWrite(4,LOW);
          analogWrite(6,(7 * 22.5));
          delay(250);

        } else if (D_mix < Right_Distance && Right_Distance < D_max)
{
          digitalWrite(2,HIGH);
          analogWrite(5,(7 * 22.5));
          digitalWrite(4,HIGH);
          analogWrite(6,(7 * 22.5));
          delay(250);
        }
      } else if (Right_Distance < D_mix && Left_Distance < D_mix) {
        digitalWrite(2,HIGH);
        analogWrite(5,0);
        digitalWrite(4,LOW);
        analogWrite(6,(9 * 22.5));
        delay(510);
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);
      }
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
    BLE_value = "";
```

```
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if ('%' == String(BLE_value).charAt(0) && 'Q' ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}

void Ultrasonic_Follow() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    int Infrared_Trigger_Flag = 0;
    int Front_Distance = 0;
    int Left_Distance = 0;
    int Right_Distance = 0;
    int Right_IR_Value = 1;
    int Left_IR_Value = 1;
    Left_IR_Value = digitalRead(A1);
    Right_IR_Value = digitalRead(A2);
    Front_Distance = checkdistance();
    if (Front_Distance < 5 && (Left_IR_Value != Infrared_Trigger_Flag
&& Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(3 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(3 * 25.5));

    } else if (Front_Distance < 5 && (Left_IR_Value ==
Infrared_Trigger_Flag && Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(0.056 * (4 * 255)));
    } else if (Front_Distance < 5 && (Left_IR_Value !=
Infrared_Trigger_Flag && Right_IR_Value == Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(0.056 * (4 * 255)));
      digitalWrite(4,HIGH);
      analogWrite(6,(4 * 25.5));
    } else if (Front_Distance < 5 && (Left_IR_Value ==
Infrared_Trigger_Flag && Right_IR_Value == Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(3 * 25.5));
```

```
      digitalWrite(4,HIGH);
      analogWrite(6,(3 * 25.5));
    } else if (Front_Distance > 10 && (Left_IR_Value !=
Infrared_Trigger_Flag && Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 25.5));
    } else if (Front_Distance > 10 && (Left_IR_Value ==
Infrared_Trigger_Flag && Right_IR_Value != Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 25.5));
    } else if (Front_Distance > 10 && (Left_IR_Value !=
Infrared_Trigger_Flag && Right_IR_Value == Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,HIGH);
      analogWrite(6,(4 * 25.5));
    } else if ((5 <= Front_Distance && Front_Distance <= 10) &&
(Left_IR_Value != Infrared_Trigger_Flag && Right_IR_Value ==
Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(4 * 25.5));
      digitalWrite(4,LOW);
      analogWrite(6,(0.056 * (4 * 25.5)));
    } else if ((5 <= Front_Distance && Front_Distance <= 10) &&
(Left_IR_Value == Infrared_Trigger_Flag && Right_IR_Value !=
Infrared_Trigger_Flag)) {
      digitalWrite(2,HIGH);
      analogWrite(5,(0.056 * (4 * 25.5)));
      digitalWrite(4,LOW);
      analogWrite(6,(4 * 25.5));
    } else if ((5 <= Front_Distance && Front_Distance <= 10) &&
(Left_IR_Value != Infrared_Trigger_Flag && Right_IR_Value !=
Infrared_Trigger_Flag)) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if ('%' == String(BLE_value).charAt(0) && 'Q' ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
```

```
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}

void Infrared_Tracing() {
  Funtion_FLag = true;
  int Left_Tra_Value = 1;
  int Center_Tra_Value = 1;
  int Right_Tra_Value = 1;
  int Black = 1;
  int white = 0;
  while (Funtion_FLag) {
    Left_Tra_Value = digitalRead(7);
    Center_Tra_Value = digitalRead(8);
    Right_Tra_Value = digitalRead(9);
    if (Left_Tra_Value != Black && (Center_Tra_Value == Black &&
Right_Tra_Value != Black)) {
      digitalWrite(2,HIGH);
      analogWrite(5,120);
      digitalWrite(4,LOW);
      analogWrite(6,120);

    } else if (Left_Tra_Value == Black && (Center_Tra_Value == Black
&& Right_Tra_Value != Black)) {
      digitalWrite(2,LOW);
      analogWrite(5,120);
      digitalWrite(4,LOW);
      analogWrite(6,120);
    } else if (Left_Tra_Value == Black && (Center_Tra_Value != Black
&& Right_Tra_Value != Black)) {
      digitalWrite(2,LOW);
      analogWrite(5,80);
      digitalWrite(4,LOW);
      analogWrite(6,80);
    } else if (Left_Tra_Value != Black && (Center_Tra_Value != Black
&& Right_Tra_Value == Black)) {
      digitalWrite(2,HIGH);
      analogWrite(5,80);
      digitalWrite(4,HIGH);
      analogWrite(6,80);
    } else if (Left_Tra_Value != Black && (Center_Tra_Value == Black
&& Right_Tra_Value == Black)) {
      digitalWrite(2,HIGH);
      analogWrite(5,120);
      digitalWrite(4,HIGH);
      analogWrite(6,120);
```

```
    } else if (Left_Tra_Value == Black && (Center_Tra_Value == Black
&& Right_Tra_Value == Black)) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    } else if (false) {
    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if ('%' == String(BLE_value).charAt(0) && 'Q' ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}

float mapfloat(float x, float in_min, float in_max, float out_min,
float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
out_min;
}
void G_Drive() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    while (Serial.available() > 0) {
      G_Bluetooth_value = G_Bluetooth_value +
((char)(Serial.read()));
      delay(2);
    }
    if (0 < String(G_Bluetooth_value).length()) {
      Serial.println(G_Bluetooth_value);
      G_string_len = String(G_Bluetooth_value).length();
      if (G_string_len <= 7 && ('@' ==
String(G_Bluetooth_value).charAt(0) && '#' ==
String(G_Bluetooth_value).charAt((G_string_len - 2)))) {
        G_degree =
String(String(G_Bluetooth_value).substring(1,(G_string_len -
2))).toInt();
        float value = (BLE_Change_SPEED / 10) * 16;
        double d_speed = 0.00;
        if (0 <= G_degree && G_degree <= 90) {
```

```
          d_speed = (mapfloat(G_degree, 0, 90, 0, 1));
          digitalWrite(2,HIGH);
          analogWrite(5,value);
          digitalWrite(4,LOW);
          analogWrite(6,(d_speed * value));

        } else if (90 < G_degree && G_degree <= 180) {
          d_speed = (mapfloat(G_degree, 180, 90, 0, 1));
          digitalWrite(2,HIGH);
          analogWrite(5,(d_speed * value));
          digitalWrite(4,LOW);
          analogWrite(6,value);
        } else if (-180 < G_degree && G_degree <= -90) {
          d_speed = (mapfloat(G_degree, (-180), (-90), 0, 1));
          digitalWrite(2,LOW);
          analogWrite(5,(d_speed * value));
          digitalWrite(4,HIGH);
          analogWrite(6,value);
        } else if (-90 < G_degree && G_degree < 0) {
          d_speed = (mapfloat(G_degree, (-90), 0, 1, 0));
          digitalWrite(2,LOW);
          analogWrite(5,value);
          digitalWrite(4,HIGH);
          analogWrite(6,(d_speed * value));
        }

      } else if ('%' == String(G_Bluetooth_value).charAt(0) && 'Q' ==
String(G_Bluetooth_value).charAt(1)) {
        Funtion_FLag = false;
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);
      } else if ('%' == String(G_Bluetooth_value).charAt(0) && '+' ==
String(G_Bluetooth_value).charAt((G_string_len - 2))) {
        BLE_Change_SPEED =
String(String(G_Bluetooth_value).substring(1,(G_string_len -
2))).toInt();
        Serial.println(BLE_Change_SPEED);
      } else if ('%' == String(G_Bluetooth_value).charAt(0) && '-' ==
String(G_Bluetooth_value).charAt((G_string_len - 2))) {
        BLE_Change_SPEED =
String(String(G_Bluetooth_value).substring(1,(G_string_len -
2))).toInt();
        Serial.println(BLE_Change_SPEED);
      } else if ('%' == String(G_Bluetooth_value).charAt(0) && 'L' ==
String(G_Bluetooth_value).charAt(1)) {
        digitalWrite(2,LOW);
        analogWrite(5,((BLE_Change_SPEED / 10) * 22.5));
        digitalWrite(4,LOW);
        analogWrite(6,((BLE_Change_SPEED / 10) * 22.5));
```

```
      } else if ('%' == String(G_Bluetooth_value).charAt(0) && 'R' ==
String(G_Bluetooth_value).charAt(1)) {
        digitalWrite(2,HIGH);
        analogWrite(5,((BLE_Change_SPEED / 10) * 22.5));
        digitalWrite(4,HIGH);
        analogWrite(6,((BLE_Change_SPEED / 10) * 22.5));
      } else if ('%' == String(G_Bluetooth_value).charAt(0) && 'S' ==
String(G_Bluetooth_value).charAt(1)) {
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);
      } else {
        digitalWrite(2,LOW);
        analogWrite(5,0);
        digitalWrite(4,HIGH);
        analogWrite(6,0);

      }
      G_Bluetooth_value = "";

    }
    G_Bluetooth_value = "";
  }
}

void Light_Seeking() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    Left_photosensitive = analogRead(A0) / 10;
    Right_photosensitive = analogRead(A3) / 10;
    if (Left_photosensitive > 55 && Right_photosensitive > 55) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    } else {
      if (Left_photosensitive > Right_photosensitive) {
        Lightseeking_Degree = ((float)(Right_photosensitive /
Left_photosensitive)) * 90;
        Serial.println("float:");
        Serial.println(((float)(Right_photosensitive /
Left_photosensitive)));

      } else if (Left_photosensitive <= Right_photosensitive) {
        Lightseeking_Degree = 180 - ((float)(Left_photosensitive /
Right_photosensitive)) * 90;
      }
      if (Lightseeking_Degree <= 90) {
        f = ((float)(Lightseeking_Degree)) / 90;
```

```
        digitalWrite(2,HIGH);
        analogWrite(5,speed_value);
        digitalWrite(4,LOW);
        analogWrite(6,(speed_value * f));

      }
      if (Lightseeking_Degree > 90) {
        f = ((float)(180 - Lightseeking_Degree)) / 90;
        digitalWrite(2,HIGH);
        analogWrite(5,(speed_value * f));
        digitalWrite(4,LOW);
        analogWrite(6,speed_value);

      }

    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if ('%' == String(BLE_value).charAt(0) && 'Q' ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}

void Infraed_Remote() {
  Funtion_FLag = true;
  while (Funtion_FLag) {
    if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_UP) {
      digitalWrite(2,HIGH);
      analogWrite(5,150);
      digitalWrite(4,LOW);
      analogWrite(6,150);

    } else if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_DOWN) {
      digitalWrite(2,LOW);
      analogWrite(5,150);
      digitalWrite(4,HIGH);
      analogWrite(6,150);
    } else if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_LEFT) {
      digitalWrite(2,LOW);
      analogWrite(5,80);
      digitalWrite(4,LOW);
```

```
      analogWrite(6,80);
      delay(200);
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    } else if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_RIGHT) {
      digitalWrite(2,HIGH);
      analogWrite(5,80);
      digitalWrite(4,HIGH);
      analogWrite(6,80);
      delay(200);
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    } else if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_OK) {
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);
    } else if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_1) {
      digitalWrite(2,LOW);
      analogWrite(5,100);
      digitalWrite(4,LOW);
      analogWrite(6,100);
    } else if (ir.getIrKey(ir.getCode(),1) == IR_KEYCODE_3) {
      digitalWrite(2,HIGH);
      analogWrite(5,100);
      digitalWrite(4,HIGH);
      analogWrite(6,100);
    }
    BLE_value = "";
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if ('%' == String(BLE_value).charAt(0) && 'Q' ==
String(BLE_value).charAt(1)) {
      Funtion_FLag = false;
      digitalWrite(2,LOW);
      analogWrite(5,0);
      digitalWrite(4,HIGH);
      analogWrite(6,0);

    }
  }
}

void setup(){
  Serial.begin(9600);
```

```
  BLE_bit_temp = 'a';
  BLE_value = "";
  G_Bluetooth_value = "";
  G_degree = 0;
  re_string_len = 0;
  IRremote ir(3);

  Left_photosensitive = 0;
  Right_photosensitive = 0;
  Lightseeking_Degree = 0;
  f = 0;
  speed_value = 120;
  D_mix = 5;
  D_mid = 10;
  D_max = 400;
  myservo.attach(10);
  Funtion_FLag = false;
  G_string_len = 0;
  BLE_Change_SPEED = 60;
  myservo.write(90);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(12, OUTPUT);
  pinMode(13, INPUT);
  pinMode(2, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(A0, INPUT);
  pinMode(A3, INPUT);
}

void loop(){
  while (true) {
    while (Serial.available() > 0) {
      BLE_value = BLE_value + ((char)(Serial.read()));
      delay(2);
    }
    if (0 < String(BLE_value).length()) {
      Serial.println(BLE_value);
      if (128 > String(BLE_value).length()) {
        if ('%' == String(BLE_value).charAt(0) && '#' ==
String(BLE_value).charAt(2)) {
          switch (String(BLE_value).charAt(1)) {
           case 'G':
            G_Drive();
            break;
           case 'T':
```

```
             Infrared_Tracing();
             break;
            case 'P':
             Light_Seeking();
             break;
            case 'I':
             Infraed_Remote();
             break;
            case 'F':
             Ultrasonic_Follow();
             break;
            case 'A':
             Ultrasonic_Avoidance();
             break;
           }
           BLE_value = "";

         } else {
           BLE_value = "";

         }

       }
       BLE_value = "";

     }
   }

}
```

Izvor: Autor.

9.2. Izvorni kod za daljinski upravljač

```
#ifndef IRremote_h
#define IRremote_h

#include <stdint.h>
#include <stdbool.h>
#include <Arduino.h>

#ifdef ME_PORT_DEFINED
#endif // ME_PORT_DEFINED
#ifndef __AVR_ATmega32U4__
#define MARK  0
#define SPACE 1
#define NEC_BITS 32
#define USECPERTICK 50
#define RAWBUF 80
```

```
typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorStatus;

#define NEC_HDR_MARK  9000
#define NEC_HDR_SPACE 4500
#define NEC_BIT_MARK  560
#define NEC_ONE_SPACE 1600
#define NEC_ZERO_SPACE     560
#define NEC_RPT_SPACE 2250
#define NEC_RPT_PERIOD     110000


#define _GAP 5000

#define STATE_IDLE     2
#define STATE_MARK     3
#define STATE_SPACE    4
#define STATE_STOP     5


#define NEC 1
#define SONY 2
#define RC5 3
#define RC6 4
#define DISH 5
#define SHARP 6
#define PANASONIC 7
#define JVC 8
#define SANYO 9
#define MITSUBISHI 10
#define SAMSUNG 11
#define LG 12
#define UNKNOWN -1

#define TOPBIT 0x80000000


#ifdef F_CPU
#define SYSCLOCK F_CPU
#else
#define SYSCLOCK 16000000
#endif


#define _GAP 5000
#define GAP_TICKS (_GAP/USECPERTICK)


#define TIMER_DISABLE_INTR   (TIMSK2 = 0)
#define TIMER_ENABLE_PWM     (TCCR2A |= _BV(COM2B1))
#define TIMER_DISABLE_PWM    (TCCR2A &= ~(_BV(COM2B1)))
```

```
#define TIMER_ENABLE_INTR    (TIMSK2 = _BV(OCIE2A))
#define TIMER_DISABLE_INTR   (TIMSK2 = 0)
#define TIMER_INTR_NAME      TIMER2_COMPA_vect
#define TIMER_CONFIG_KHZ(val) ({ \
  const uint8_t pwmval = F_CPU / 2000 / (val); \
  TCCR2A = _BV(WGM20); \
  TCCR2B = _BV(WGM22) | _BV(CS20); \
  OCR2A = pwmval; \
  OCR2B = pwmval / 3; \
})

#define TIMER_COUNT_TOP      (SYSCLOCK * USECPERTICK / 1000000)
#if (TIMER_COUNT_TOP < 256)
#define TIMER_CONFIG_NORMAL() ({ \
  TCCR2A = _BV(WGM21); \
  TCCR2B = _BV(CS20); \
  OCR2A = TIMER_COUNT_TOP; \
  TCNT2 = 0; \
})
#else
#define TIMER_CONFIG_NORMAL() ({ \
  TCCR2A = _BV(WGM21); \
  TCCR2B = _BV(CS21); \
  OCR2A = TIMER_COUNT_TOP / 8; \
  TCNT2 = 0; \
})
#endif

typedef struct {
  uint8_t recvpin;
  volatile uint8_t rcvstate;
  volatile uint32_t lastTime;
  unsigned int timer;
  volatile uint8_t rawbuf[RAWBUF];
  volatile uint8_t rawlen;
} irparams_t;


class IRremote
{
public:

  IRremote(int pin);
  ErrorStatus decode();
  void begin();
  void end();
  void loop();
  boolean keyPressed(unsigned char r);
  // void resume();

  int8_t decode_type;
```

```
  unsigned long value;
  uint8_t bits;
  volatile uint8_t *rawbuf;
  int rawlen;
  String getString();
  unsigned char getCode();
  String getKeyMap(byte keycode, byte ir_type = 1);
  byte getIrKey(byte keycode,    byte ir_type = 1);
  void sendString(String s);
  void sendString(float v);
  void sendNEC(unsigned long data, int nbits);
  void sendRaw(unsigned int buf[], int len, uint8_t hz);
  void enableIROut(uint8_t khz);
  void enableIRIn();
  void mark(uint16_t us);
  void space(uint16_t us);
private:
  ErrorStatus decodeNEC();
  int16_t irIndex;
  char irRead;
  char floatString[5];
  boolean irReady;
  boolean irPressed;
  String irBuffer;
  String Pre_Str;
  double irDelayTime;
};
#endif // !__AVR_ATmega32U4__
#endif
```

Izvor: Autor.

9.3. Izvorni kod za tipke na daljinskom upravljaču

```
#ifndef _KEYMAY_H_
#define _KEYMAY_H_
#include <Arduino.h>
#define KEY_MAX 21
typedef struct
{
    String keyname;
    byte keycode;
}ST_KEY_MAP;

#define IR_TYPE_NORMAL 1
#define IR_TYPE_EM 2

typedef enum {
    IR_KEYCODE_1 = 0,
    IR_KEYCODE_2,
```

```
    IR_KEYCODE_3,
    IR_KEYCODE_4,
    IR_KEYCODE_5,
    IR_KEYCODE_6,
    IR_KEYCODE_7,
    IR_KEYCODE_8,
    IR_KEYCODE_9,
    IR_KEYCODE_0,
    IR_KEYCODE_STAR,      // *
    IR_KEYCODE_POUND,     // #
    IR_KEYCODE_UP,
    IR_KEYCODE_DOWN,
    IR_KEYCODE_OK,
    IR_KEYCODE_LEFT,
    IR_KEYCODE_RIGHT,
}E_NORMAL_IR_KEYCODE;

typedef enum {
    EM_IR_KEYCODE_A = 0,
    EM_IR_KEYCODE_B,
    EM_IR_KEYCODE_C,
    EM_IR_KEYCODE_D,
    EM_IR_KEYCODE_UP,
    EM_IR_KEYCODE_PLUS,
    EM_IR_KEYCODE_LEFT,
    EM_IR_KEYCODE_OK,
    EM_IR_KEYCODE_RIGHT,
    EM_IR_KEYCODE_0,
    EM_IR_KEYCODE_DOWN,
    EM_IR_KEYCODE_REDUCE,
    EM_IR_KEYCODE_1,
    EM_IR_KEYCODE_2,
    EM_IR_KEYCODE_3,
    EM_IR_KEYCODE_4,
    EM_IR_KEYCODE_5,
    EM_IR_KEYCODE_6,
    EM_IR_KEYCODE_7,
    EM_IR_KEYCODE_8,
     EM_IR_KEYCODE_9,
}E_EM_IR_KEYCODE;

extern ST_KEY_MAP normal_ir_keymap[];
extern ST_KEY_MAP em_ir_keymap[];
#endif  /* _KEYMAY_H_ */
```

Izvor: Autor.