



UNIVERZITET U BIHAĆU  
TEHNIČKI FAKULTET BIHAĆ  
ELEKTROTEHNIČKI ODSJEK  
INFORMATIKA

---

## Projektovanje lunaparka

---

SEMINARSKI RAD IZ PREDMETA SISTEMI U REALNOM  
VREMENU

**Student:**  
**Amira Midžić, 985**

**Profesor: Van. prof. dr. Edin Mujčić**  
**Asistent: Mr. Una Drakulić**

Bihać,  
akademska godina 2019/20.

# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>1</b>
<b>2</b>	<b>Historija lunaparka</b>	<b>2</b>
<b>3</b>	<b>Dizajn i implementacija</b>	<b>5</b>
3.1	Maketa sistema . . . . .	5
<b>4</b>	<b>Programiranje sistema</b>	<b>10</b>
4.1	Arduino kôd . . . . .	10
4.2	Android Studio kôd . . . . .	14
4.3	Komunikacija mikrokontrolera i vanjskih uređaja . . . . .	17
<b>5</b>	<b>Eksperimentalna analiza rada sistema</b>	<b>18</b>
<b>6</b>	<b>Zaključak</b>	<b>21</b>
	<b>Literatura</b>	<b>22</b>
	<b>Prilozi</b>	<b>23</b>
<b>A</b>	<b>Programski kôd</b>	<b>24</b>

# 1. Uvod

U ovom radu prikazan je postupak pri izradi jednog lunaparka kao mikroprocesorskog sistema. Lunapark ili zabavni park je naziv za skup zabavnih sadržaja, vožnji i sličnih zbivanja koji se trajno odvijaju na jednoj lokaciji, namijenjenih za zabavu većeg broja ljudi. Složeniji je od igrališta ili gradskog parka, pošto ima sadržaje za sve starosne skupine.

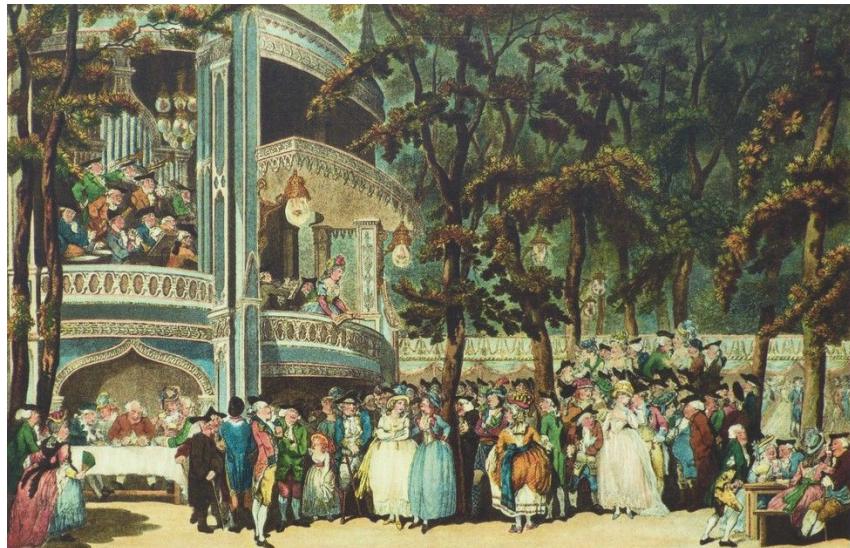
Za projektovanje lunaparka korištena je pločica Arduino Mega, te veliki broj senzora i motora. Pokretanje sistema realizirano je kao sistem ulaznica sa Android aplikacijom, razvijenom pomoću softvera Android Studio, te Tensorflow Lite, framework otvorenog kôda za duboko učenje koji je namijenjen mobilnim uređajima. Korisnik pokrene aplikaciju, odmah se otvorí kamera njegovog mobitela ispred koje treba staviti ulaznicu za sistem. Ulaznice su napravljene posebno za ovaj sistem korištenjem softvera Adobe Photoshop, a prepoznavanje istih urađeno je treniranjem neuronske mreže. Kad aplikacija prepozna ulaznicu, korisniku se automatski pokrene izbornik za povezivanje preko bluetootha. Nakon što se korisnik poveže, otvorí mu se ekran s izbornikom za pokretanje i zaustavljanje određenih dijelova sistema, uključivanje i isključivanje svjetala, pokretanje i zaustavljanje vrteški, pokretanje igre, te pokretanje i zaustavljanje muzike.

Svi dijelovi lunaparka pravljeni su od kartona u koji su smještene elektronske komponente, te povezane s ostatkom sistema žicama. Glavni dio sistema, veliki šator sa Arduino Mega pločicom, driverima za motore, te bluetooth modulom i modulom za SD karticu, smješten je u sredinu makete.

## 2. Historija lunaparka

Zabavni parkovi ili lunaparkovi su prostori ispunjeni različitim sadržajima namijenjenim zabavi. Podgrupa zabavnih parkova su tematski parkovi, kod kojih su atrakcije i struktura bazirani na jednoj temi. Prvi tematski parkovi su nastali sredinom dvadesetog stoljeća, među kojima se ističe najpopularniji, Disneyland, otvoren 1955. godine [1]. Najposjećeniji tematski park na svijetu je Magic Kingdom u Floridi, dio Walt Disney Worlda, kojeg je samo 2019. godine posjetilo preko 20 miliona ljudi [2]. Procjenjuje se da na svijetu ima nekoliko hiljada zabavnih parkova, koji ukupno godišnje zarade preko 220 milijardi dolara [3].

Zabavni parkovi su se razvili iz nekoliko starih tradicija, a to su: putujući sajmovi, vrtovi zadovoljstva, te međunarodne izložbe, odnosno svjetski sajmovi. Za razliku od navedenih, zabavni parkovi su trajnog karaktera. Jedan od najranijih poznatih sajmova bio je Bartholomew Fair u Engleskoj, koji je počeo sa radom 1133. godine. Prvi vrtovi zadovoljstva ili zabave nastali su u Europi u 16. stoljeću. Tako su se zvali jer su imali puno atrakcija koje su zabavljale ljude, poput muzike, igara i vožnji. Jedan od njih je i najstariji zabavni park koji je još u funkciji, Bakken u Klampenborgu u Danskoj. Postoji od 1583. godine, a smatra se da su njegovi pojedini dijelovi još stariji [3]. Još jedan popularan vrt zadovoljstva je Vauxhall Gardens, nastao 1661. godine u Londonu. Često je okupljaо ogromni broj ljudi svojim brojnim atrakcijama, kao što su akrobati, baloni vrućeg zraka, koncerti, te vatrometi. Imao je sistem plaćanja ulaza. Iako je isprva bio namijenjen eliti, uskoro je postao mjesto za sve slojeve društva [4]. Prater u Beču u Austriji započeo je sa radom kao kraljevsko mjesto za lov 1766. godine. Uskoro su se oko tog mjesta otvorili kafići, te razne atrakcije, što je dovelo do stvaranja zabavnog parka. Danas se u njemu nalazi najduži vrtuljak na svijetu, te je jedan od najbolje ocijenjenih zabavnih parkova svijeta [3].



Slika 2.1: Vauxhall Gardens [3]

Svjetski sajmovi utjecali su na razvijanje koncepta trajnog parka namijenjenog zabavi. Prvi svjetski sajam počeo je 1851. godine konstrukcijom Kristalne palače u Londonu u Engleskoj. Njegova svrha je bila proslava industrijskog napretka, te je dizajniran za edukaciju i zabavu posjetitelja. Posjetilo ga je 6 miliona ljudi, te je po ugledu na njega organiziran niz svjetskih sajmova širom Europe, a kasnije i Amerike. Svjetska kolumbijska izložba, organizirana 1893. godine u Chicagu u Sjedinjenim Američkim Državama se smatra pretečom modernih zabavnih parkova, s obzirom na brojne sadržaje koje su kasnije svi zabavni parkovi imali, poput vožnji, igara, svjetala, iluzija, te prvog željeznog vrtuljka[5]. Uskoro su postala popularna različita odmarališta uz rijeke, jezera ili mora, koja su nudila različite zabavne sadržaje za svoje posjetitelje. U njima je izgrađen i prvi voz smrti (eng.*rollercoaster*), izgrađen u Pennsylvaniji u Sjedinjenim Američkim Državama, te najstariji vrtuljak (karusel), izgrađen 1780. godine za princa Wilhelma IX od Hessen-Kassela, te, iako nije funkcionalan, može se vidjeti u Wilhelmsbad Parku u Hanauu u Njemačkoj, rodom gradu braće Grimm [3].



**Slika 2.2:** Prvi svjetski sajam, ulje na platnu od Davida Robertsa [6]

Početkom 20. stoljeća, željezničke kompanije su, kako bi ostvarivale profit i vikendom, kad se ljudi manje kreću pošto ne putuju na posao i natrag, počeli otvarati takozvane *trolley* parkove, koji su u početku imali dešavanja poput plesova, koncerata, vatrometa, a kasnije dobili gotovo sve sadržaje koje imaju današnji zabavni parkovi, poput vrtuljaka, vrteški, vozova smrti, i slično, te raznih restorana, vožnji brodovima, bazena, te brojnih drugih zabavnih sadržaja. Smatralju se pretečama današnjih zabavnih parkova. Jedan od najpopularnijih i najuspješnijih bio je Coney Island, pokraj autoputa u Coney Islandu u Brooklynu. Procjenjuje se da je u ovom periodu u Sjedinjenim Američkim Državama bilo između 1500 i 2000 *trolley* parkova. Njihova popularnost se smanjila kad se povećala popularnost automobila [7].

Prvo trajno zatvoreno područje namijenjeno zabavi kojim upravlja jedna kompanija osnovano je 1895. godine u Coney Islandu u Brooklynu, a zvalo se Sea Lion Park. To je bio prvi zabavni park u kojem se, osim ulaza, plaćao i pristup pojedinim atrakcijama. Kad mu se 1897. godine pridružio Steeplechase Park, zbog blizine centra sa jako velikim brojem stanovništva, te lakoće pristupa, Coney Island je postao utjelovljenje američkog zabavnog parka. Njegovi dijelovi su bili i Luna Park, te Dreamland [9]. Zbog velike popularnosti, uskoro su diljem svijeta otvoreni brojni zabavni parkovi pod imenom lunapark, a samo ime je ušlo u bosanski jezik kao sinonim za zabavni park.

Period od 1920. godine do Velike depresije 1930. godine, te Drugog svjetskog rata se smatra zlatnim dobom zabavnih parkova. Radno vrijeme u Sjedinjenim Američkim Državama se skratilo, a plate povećale, pa su ljudi imali više slobodnog vremena i novca. U ovom periodu



Slika 2.3: Coney Island u New Yorku [8]

je najviše zabavnih parkova otvoreno, te je izmišljeno najviše atrakcija za njih [9]. U pedesetim godinama se promijenio način kako ljudi provode slobodno vrijeme, populacija se zbog rata preselila izvan gradova, te je porasla popularnost televizije kao izvora zabave, a popularnost zabavnih parkova se jako smanjila. Tad su brojni zabavni parkovi zatvoreni ili zapaljeni. Walt Disney je napravio veliki rizik otvorivši veliki zabavni park pod svojim imenom, odnosno jedan od prvih tematskih parkova na svijetu, te mu se isti i isplatio, pošto je postao uzor za sve kasnije otvorene tematske parkove, te je svaki otvoreni Disneyev tematski park i danas na listi od 20 najposjećenijih tematskih parkova na svijetu [3][2].

Velike međunarodne kompanije, kao što su već spomenuta Walt Disney Company, te Universal Studios, Six Flags Entertainment Corporation, te Merlin Entertainments, vlasnik Legoland-a, do 2020. godine su samo u Aziji uložile 24 milijarde dolara za izgradnju 60 tematskih parkova. Planira se gradnja ruskog odgovora Disneylandu, Magical World of Rusia, koji će vrijediti 4 milijarde dolara. Prvotno planiran za 2020. godinu, a zbog pandemije Covid-19 odgođen na oktobar 2021. godine, 2020 World Expo (Svjetski sajam 2020. godine) bit će organiziran u Dubaiju, te će sadržavati brojne zabavne i tematske parkove [3].

Zabavni parkovi, osim zabave, ostvaruju veliku ekonomsku dobit, dovode ogroman broj turista, te zapošljavaju veliki broj ljudi, bilo na projektovanju atrakcija ili na njihovom održavanju.

# 3. Dizajn i implementacija

Proces dizajniranja i implementacije, odnosno, projektovanja sistema, može se posmatrati kao proces rješavanja problema. Sam proces zahtijeva da se prvo pronađe niz mogućih rješenja, a zatim od pronađenih izabere najbolje rješenje. Svaki sistem mora biti projektovan na vrijeme, u okviru sredstava, imati prihvatljive performanse, te biti ispravan i pouzdan. Dobro projektovan sistem ne treba samo zadovoljiti trenutne zahtjeve, već i imati mogućnost da se prilagodi novim zahtjevima.

Neki veliki lunapark obično projektuje tim ljudi. Određen broj ljudi zadužen je za planiranje arhitekture lunaparka, drugi tim ljudi zadužen je za planiranje tehničkog dijela, a posebni timovi zaduženi su za sam proces gradnje, a često sistem ima i tim zadužen za marketing. S obzirom da je ovo studentski projekt, simulacija lunaparka, sav posao, umjesto timova, radi jedan student. Sistem je sastavljen od podsistema, koje čine određene komponente uklopljene u objekte, najčešće napravljene od kartona. Sve to povezuje Arduino Mega mikrokontroler, programiran kôdom predstavljenim u nastavku seminarског rada.

## 3.1 Maketa sistema

Sistem se sastoji od jako detaljne makete, te elektronskih komponenti koje mu daju funkcionalnost. Dijelovi sistema su karusel, vrtuljak, šoljice, vračara, te igra koja uključuje sortiranje boja. Svi dijelovi su raspoređeni u ručno napravljene šatore ili na postolja.

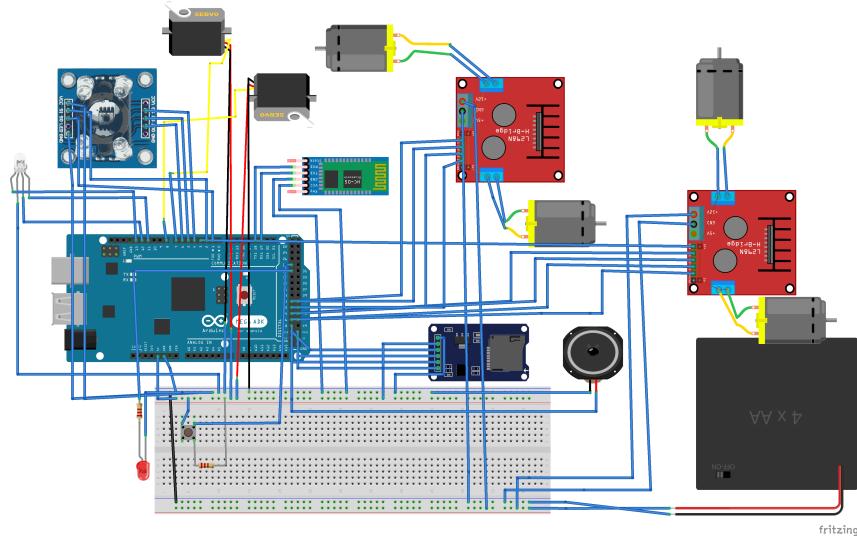


**Slika 3.1:** Maketa sistema

Maketa sistema prikazana je na slici 5.4. Podloga je napravljena od šperploče, s tim da su zidovi, radi sigurnosti i stabilnosti, učvršćeni letvicama. Za nju nije korišteno ljepilo, već

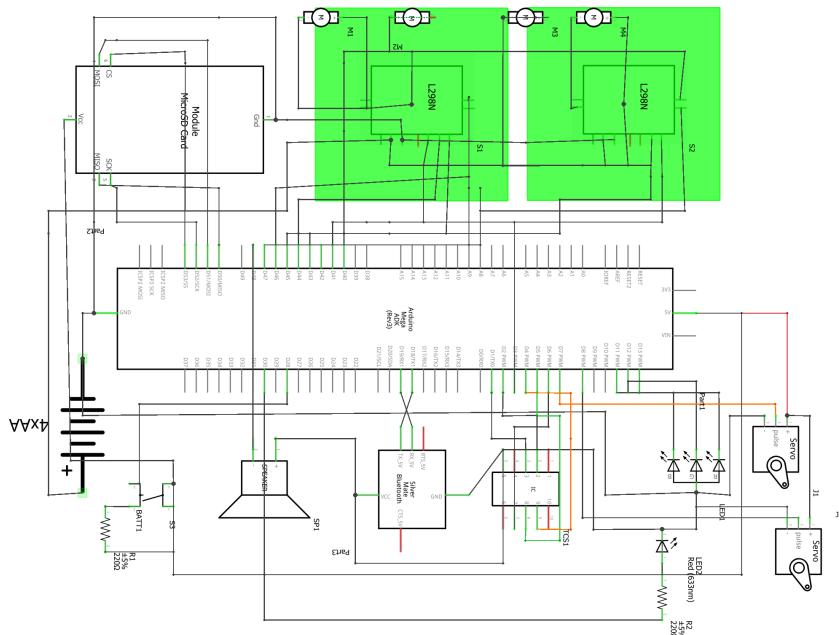
je pričvršćena ekserima. Posteri na zidovima su isprintani. Većina dijelova je napravljena od kartona, na koji su zalijepljeni šabloni od papira, koji su ili napravljeni u Photoshopu, ili preuzeti s Interneta, ali prilagođeni projektu.

Shema sistema prikazana je na slici 3.2.



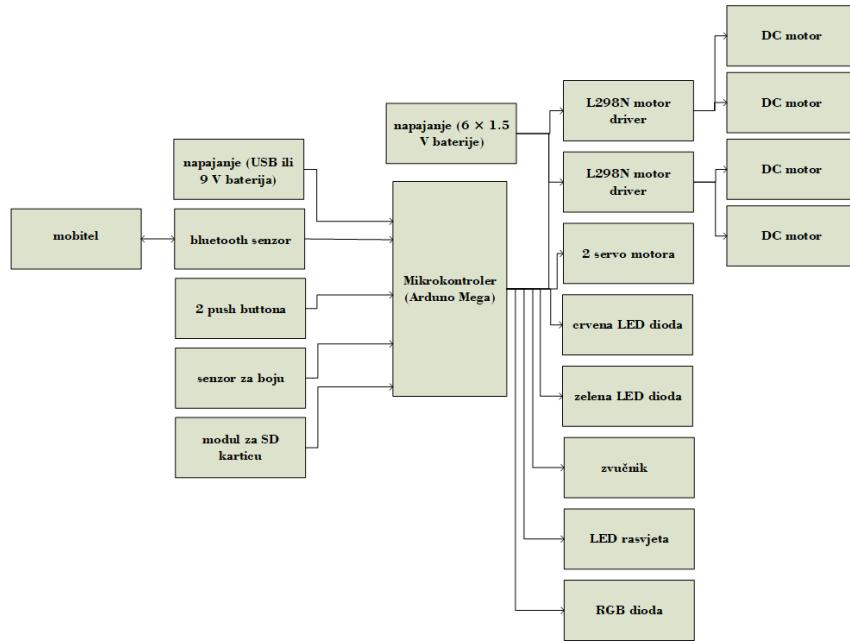
**Slika 3.2:** Shema sistema

Shema je malo pojednostavljena, duple diode i push buttoni nisu stavljeni. Električna shema prikazana je na slici 3.3.



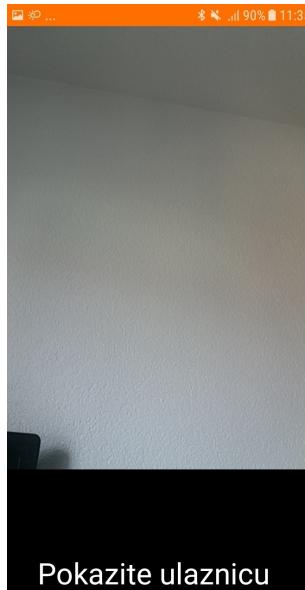
**Slika 3.3:** Električna shema sistema

Na slici 3.4 prikazan je blokovski dijagram sistema. Na dijagramu su označeni ulazi i izlazi iz sistema. Ulazi u sistem su napajanje, poruke primljene putem bluetootha, te informacije od senzora, modula i push buttona. Izlazi su signali koji se daju LED diodama, servo motorima, zvučniku, te motor driverima, koji ih prosljeđuju DC motorima. Motor driveri, osim signala od mikrokontrolera, primaju i vlastito napajanje, pošto nije dobro DC motore napajati zajedno s mikrokontrolerom.



**Slika 3.4:** Blokovski dijagram sistema

Osim komponenti, sistem se sastoji od Android aplikacije. Aplikacijom se simulira sistem za ulaznice u sistem. Korisnik pokaže ulaznicu radniku koji ima aplikaciju na mobitelu, te ako je ona odgovarajuća, korisnik može ući u sistem. Kad se pokrene Android aplikacija, otvorit će se prozor za prikazivanje ulaznice, kao na slici 3.5.



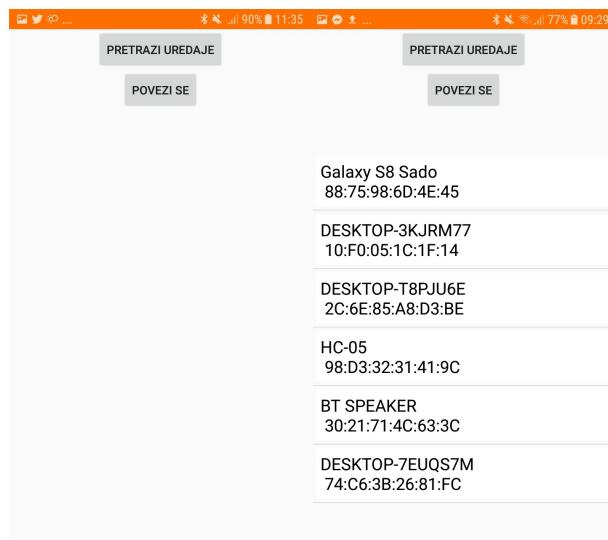
**Slika 3.5:** Izbornik za prikazivanje ulaznice

Izrađene ulaznice prikazane su na slici 3.6.



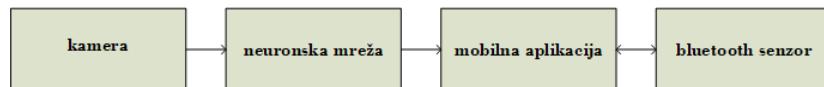
**Slika 3.6:** Ulažnice za sistem

Kad aplikacija prepozna ulaznicu, pokaže se izbornik za pretragu i povezivanje, kao na slici 3.7. Na istoj slici je prikazana i lista bluetooth uređaja koja se prikazuje kad se pritisne izbornik za pretragu. Nakon povezivanja, otvoriti se izbornik kao na slici 3.9. Uz pomoć ovog izbornika, korisnik upravlja sistemom.



**Slika 3.7:** Izbornik za pretragu i povezivanje

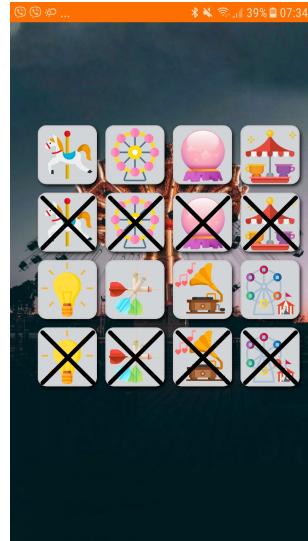
Blokovski dijagram podistema za ulaznice prikazan je na slici 3.8. Akvizicija ulaznice vrši se uz pomoć kamere mobitela. Prepoznavanje ulaznica vrši se uz pomoć neuronske mreže, koja šalje informaciju o prepoznavanju mobilnoj aplikaciji. Aplikacija potom komunicira sa sistemom uz pomoć bluetooth modula.



**Slika 3.8:** Blokovski dijagram sistema za ulaznice

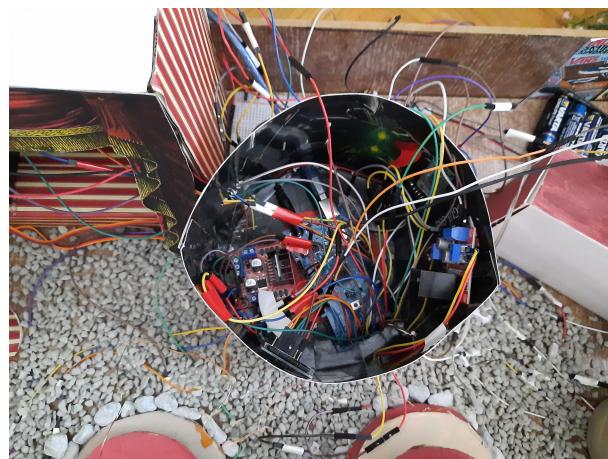
Komponente koje su iskorištene za kreiranje lunaparka su: Arduino Mega 2560 pločica, 4 različita DC motora (2 obična DC motora od 3 V, jedan od 12 V, a jedan sa zupčanicima), L298N driver za motore, modul za SD karticu, zvučnik, HC-05 Bluetooth modul, RGB dioda, senzor za boju, 2 servo motora, 3 push buttona, eksperimentalna pločica, te diode i otpornici.

Većina komponenti smještena je u veliki šator na sredini makete, kako ne bi kvarile sam dizajn projekta. Unutrašnjost šatora prikazana je na slici 3.10. Posebna pažnja morala je da se obrati na to da komponente budu povezane s odgovarajućim pinovima. Bluetooth modul morao je biti povezan sa serijskim portom, kojih Arduino Mega ima čak 4, modul za SD karticu morao je biti povezan sa posljednja 4 pina, pošto su to MISO (*Master In Slave Out*), MOSI (*Master*



**Slika 3.9:** Finalni izbornik aplikacije

*Out Slave In), SCK (Serial Clock) i SS (Slave Select) pinovi, pinovi namijenjeni za asinhronu komunikaciju u kojoj je Arduino pločica master element, dok su RGB dioda i kontrola brzine motora morali biti povezani sa PWM pinovima.*



**Slika 3.10:** Unutrašnjost šatora u kojem je skrivena većina komponenti

Motorima u sistemu obezbijeđeno je vlastito napajanje, pomoću paketa od 6 AA baterija od 1,5 V, dok se Arduino pločica i ostale komponente mogu napajati uz pomoć USB-a, odnosno, električne energije, ili baterije od 9 V. Na ovaj način sistem je osiguran od pregaranja, a istovremeno je omogućeno dovoljno energije da se veliki broj motora kreće kako je planirano.

# 4. Programiranje sistema

## 4.1 Arduino kôd

Arduino programski jezik temelji se na Wiring programskom okviru za upravljanje mikrokontrolerima, a Arduino softver temelji se na Processing softveru. Wiring je sličan C++ programskom jeziku, ali je djelomično pojednostavljen i izmijenjen. Wiring i Processing su, kao i Arduino, *open source*, odnosno otvorenog kôda, dostupni besplatno, zajedno sa svim projektima i doprinosima korisnika.

Cijeli programski kôd ovog sistema dan je prilogu. On se sastoji od uvođenja svih biblioteka potrebnih za rad, deklarisanja varijabli, funkcija za rad pojedinih dijelova, te *setup()* i *loop()* funkcija. U *setup()* funkciji su sve radnje koje se pokreću čim se mikrokontroler pokrene, a u *loop()* funkciji radnje koje se ponavljaju beskonačno mnogo puta dok god mikrokontroler ima ulazni napon.

Ono što se u ovom sistemu beskonačno mnogo puta ponavlja je čitanje podataka koji se primaju preko Bluetootha. Loop se neprestano vrti dok god je pločica priključena na napon. Tu se čitaju primljeni podaci s bluetooth senzora, i prema njima pokreću funkcije. Ako je korisnik pritisnuo button za pokretanje vrteške, ona se počne vrtiti, ako je pritisnuo za zaustavljanje, ona stane, i tako dalje.

```
1 void loop() {
2
3   while(Serial1.available()==0) ;
4
5   if(Serial1.available()>0) {
6     data = Serial1.parseInt();
7
8   }
9
10  delay(400);
```

U varijablu *data* spremaju se brojevi primljeni preko bluetooth senzora. Njihova vrijednost je definisana u aplikaciji napravljenoj pomoću Android studija. U toj aplikaciji, na primjer, ako korisnik pritisne button za uključenje dioda, preko Bluetootha se šalje broj 67, a ako pritisne button za gašenje dioda, preko bluetootha se šalje broj 76. Shodno tome, aplikacija mikrokontrolera, ako primi broj 67, treba uključiti diode, a ako primi broj 76, treba ih isključiti.

```
1 if (data == 67) {
2   diodeUkljuci();
3 }
4 if (data == 76) {
5   diodeIskljuci();
6 }
```

Na analogan način je napravljena veza između aplikacije i svih drugih dijelova sistema, samo, naravno, sa različitim vrijednostima koje se šalju preko bluetootha.

```
1 if (data == 23) {  
2     vracara();  
3 }  
4 if (data == 32) {  
5     vracaraUgasi();  
6 }
```

Vračara ima svoju kristalnu kuglu, čije se boje mijenjaju promjenom boja RGB diode.

```
1 void vracara() {  
2     RGB_color(255, 0, 0); // crvena  
3     delay(1000);  
4     RGB_color(0, 255, 0); // zelena  
5     delay(1000);  
6     RGB_color(0, 0, 255); // plava  
7     delay(1000);  
8     RGB_color(255, 255, 125); // roza  
9     delay(1000);  
10    RGB_color(0, 255, 255); // cijan  
11    delay(1000);  
12    RGB_color(255, 0, 255); // magenta  
13    delay(1000);  
14    RGB_color(255, 255, 0); // zuta  
15    delay(1000);  
16    RGB_color(255, 255, 255); // bijela  
17    delay(1000);  
18 }  
19  
20  
21 void RGB_color(int crvena, int zelena, int plava) {  
22     analogWrite(crveni_pin, crvena);  
23     analogWrite(zeleni_pin, zelena);  
24     analogWrite(plavi_pin, plava);  
25 }
```

Senzor za boje osvijetli lopticu uz pomoć jedne od tri diode, prvo crvene (R), pa zelene (G), pa plave (B), a onda čita vrijednost dobijene frekvencije da dobije zastupljenost boje koju ta dioda emitira na loptici.

```
1 int readColor() {  
2     // Citanje crvenih fotodioda  
3     digitalWrite(S2, LOW);  
4     digitalWrite(S3, LOW);  
5     frekvencija = pulseIn(sensorOut, LOW);  
6     int R = frekvencija;  
7     delay(50);  
8  
9     // Citanje zelenih fotodioda  
10    digitalWrite(S2, HIGH);  
11    digitalWrite(S3, HIGH);  
12    frekvencija = pulseIn(sensorOut, LOW);  
13    int G = frekvencija;  
14    delay(50);  
15  
16    // Citanje plavih fotodioda  
17    digitalWrite(S2, LOW);
```

```

18   digitalWrite(S3, HIGH);
19   frekvencija = pulseIn(sensorOut, LOW);
20   int B = frekvencija;
21   delay(50);
22
23   if((R<45 & R>32 & G<70 & G>50) || (R>70 & G<10 & B <50 & B>30) ) {
24     color = 1; // Crvena
25   }
26   if(R<45 & R>35 & B<40 & B>30) {
27     color = 2; // Roza
28   }
29   if(R<63 & R>50 & G<55 & G>45 & B<40 & B>29) {
30     color = 3; // Plava
31   }
32   if(R<40 & R>30 & G<50 & G>40 & B<46 & B>38) {
33     color = 4; // Zuta
34   }
35   return color;
36 }
```

Ako je pritisnut jedan od buttona, pokreće se dio programa. Ako korisnik pritisne odgovarajući button, te pogodi koje je boje loptica, on pobijedi, uključuje se zelena dioda, u suprotnom, uključuje se crvena. Loptica se uz pomoć servo motora i napravljenog mehanizma kreće, prvo do senzora za boju, gdje se očita njena boja, a onda u odgovarajuću posudu.

```

1 void senzorZaBoje() {
2   crveniButtonState = digitalRead(crveniButton);
3   plaviButtonState = digitalRead(plaviButton);
4
5
6   if (crveniButtonState == HIGH || plaviButtonState == HIGH) {
7     gornjiServo.write(120);
8     delay(3000);
9
10    for(int i = 115; i > 55; i--) {
11      gornjiServo.write(i);
12      delay(2);
13    }
14    delay(1000);
15
16    color = readColor();
17    color = readColor();
18    color = readColor();
19    color = readColor();
20
21    delay(1000);
22    switch (color) {
23      case 1: //crvena
24        if (crveniButtonState == HIGH) {
25          digitalWrite(zelenaDioda, HIGH);
26          digitalWrite(crvenaDioda, LOW);
27        }
28        if (plaviButtonState == HIGH ) {
29          digitalWrite(crvenaDioda, HIGH);
30          digitalWrite(zelenaDioda, LOW);
31        }
32        donjiServo.write(50);
33        break;
```

```
34  case 2: //roza
35  digitalWrite(crvenaDioda, HIGH);
36  digitalWrite(zelenaDioda, LOW);
37  donjiServo.write(75);
38  break;
39  case 3: // zuta
40  digitalWrite(crvenaDioda, HIGH);
41  digitalWrite(zelenaDioda, LOW);
42  donjiServo.write(100);
43  break;
44  case 4: // plava
45  if (plaviButtonState == HIGH) {
46  digitalWrite(zelenaDioda, HIGH);
47  digitalWrite(crvenaDioda, LOW);
48  }
49  if (crveniButtonState == HIGH ) {
50  digitalWrite(crvenaDioda, HIGH);
51  digitalWrite(zelenaDioda, LOW);
52  }
53  donjiServo.write(125);
54  break;
55  case 0:
56  break;
57  }
58  delay(300);
59
60  for(int i = 55; i > 20; i--) {
61  gornjiServo.write(i);
62  delay(2);
63  }
64  delay(500);
65
66  for(int i = 29; i < 115; i++) {
67  gornjiServo.write(i);
68  delay(2);
69  }
70  color=0;
71  }
72 }
```

Motori se vrte stavljanjem pozitivnih i negativnih napona na njihove pinove.

```
1 void vrtiMotore(){
2  digitalWrite(motor1pin1, HIGH);
3  digitalWrite(motor1pin2, LOW);
4
5  digitalWrite(motor2pin1, HIGH);
6  digitalWrite(motor2pin2, LOW);
7
8  digitalWrite(motor3pin1, HIGH);
9  digitalWrite(motor3pin2, LOW);
10
11  analogWrite(enA, 160); // upravljanje brzinom
12  digitalWrite(motor4pin1, HIGH);
13  digitalWrite(motor4pin2, LOW);
14 }
```

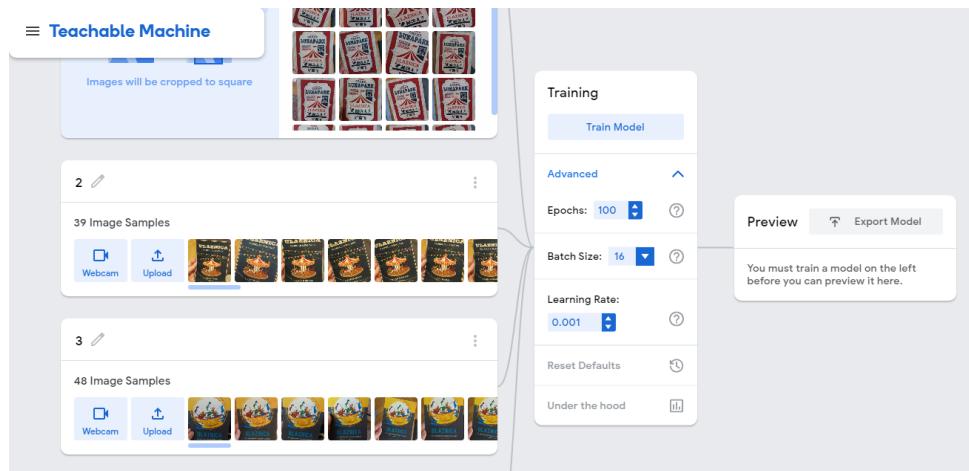
Na ovaj način isprogramiran je svaki dio sistema.

## 4.2 Android Studio kôd

Sve mobilne aplikacije napravljene uz pomoć softvera Android Studio sastoje se od takozvanih aktivnosti (eng. *activities*). Jedna aktivnost predstavlja jednu radnju, koja najčešće zauzima cijeli ekran. Iz jedne aktivnosti u drugu prelazi se automatski, ili aktiviranjem nekog linka ili tipke na ekranu.

Prvi dio aplikacije napravljen je uz pomoć biblioteke od Tensorflowa za treniranje neuronske mreže, te prepoznavanje treniranih objekata u Android aplikaciji. Prepoznavanje slike je akcija kojom uređaj, koji izvrši akviziciju slike, u ovom slučaju preko kamere kao ulaza, prepozna koji je objekat na istoj, te ispiše kojoj klasi pripada. Moderni načini prepoznavanja slike su korištenjem dubokog učenja, odnosno, treniranjem konvolutivnih neuronskih mreža na ogromnim bazama podataka. Tensorflow, korišten za izradu programa, je *open-source* platforma, odnosno platforma otvorenog kôda, za mašinsko učenje koja sadrži ogroman broj alata i biblioteka za olakšani razvoj aplikacija. Ovu platformu je razvio Google Brain, Googleov tim za umjetnu inteligenciju, prvotno samo za potrebe kompanije Google, a u novembru 2015. godine objavljen je za široku populaciju pod Apache 2.0 licencom. Može se koristiti za kreiranje algoritama za mašinsko učenje, te za pokretanje istih, a bez puno promjena može se pokrenuti na mnogim uređajima, od mobilnih telefona i tableta do velikih distribuiranih sistema sastavljenih od stotina ili hiljada uređaja poput GPU kartica [10].

Klasifikator za prepoznavanje slika u ovom projektu napravljen je treniranjem jednostavnog modela uz pomoć dubokog učenja sa online alatom kojeg je razvio Google, a koji se zove Teachable Machine. Treniranje modela prikazano je na slici 4.1, a isti je eksportovan u TensorFlow lite datoteku, koja se može koristiti na Android uređajima.



Slika 4.1: Treniranje neuronske mreže

Model se sastoji od slika ulaznica za lunapark, koje su kreirane u softveru Adobe Photoshop. Za projekat su napravljene 4 ulaznice, a njihove fotografije raspoređene su u 4 direktorija pomoću kojih je neuronska mreža istrenirana. TensorFlow nudi vlastitu aplikaciju otvorenog kôda za korištenje modela. Ista je izmijenjena da se prilagodi potrebama projekta.

Prvo što se korisniku prikaže je izlaz iz kamere njegovog mobitela, te poruka da pokaže ulaznicu. Bitan dio kôda je:

```
1 protected void showResultsInBottomSheet(List<Recognition> results)
2 {
3     if (results != null && results.size() >= 3) {
```

```

4     Recognition recognition = results.get(0);
5     if (recognition != null) {
6
7         if (recognition.getTitle() != null)
8             recognitionTextView.setText(recognition.getTitle());
9             System.out.println("name"+recognition.getTitle());
10
11         System.out.println("namevalue"+recognition.getConfidence());
12         if(recognition.getConfidence()<0.90f){
13             result_text.setText("Pokazite ulaznicu");
14             // ako je stepen prepoznavanja manji od 0.9, korisniku se
15             // ispisuje poruka da pokaze ulaznicu
16         }
17         if(recognition.getConfidence()>0.90f){
18             result_text.setText(recognition.getTitle());
19             Intent intent = new Intent(getApplicationContext(),
20                 BtActivity.class);
21             startActivity(intent);
22             // ako je stepen prepoznavanja veci od 0.9, korisnika se
23             // prebacuje na Bluetooth aktivnost
24         }
25     }

```

Kad sistem prepozna ulaznicu, pokrene se sljedeća aktivnost, a to je izbornik za traženje dostupnih Bluetooth uređaja, te povezivanje na odgovarajući. Sastoji se od dva buttona, odnosno dugmeta, tipke, te liste Bluetooth uređaja koja nije vidljiva dok korisnik ne pokrene pretragu.

Kad korisnik dodirne dugme za pretragu, ispiše mu se lista dostupnih bluetooth uređaja:

```

1  pretraga.setOnClickListener(new View.OnClickListener() {
2
3     @Override
4     public void onClick(View arg0) {
5         mBTAdapter = BluetoothAdapter.getDefaultAdapter();
6         // povezivanje na bluetooth adapter
7
8         if (mBTAdapter == null) {
9             Toast.makeText(getApplicationContext(), "
10                 Bluetooth_uredajii_nisu_pronadeni", Toast.
11                 LENGTH_SHORT).show(); // poruka ako nije
12                 pronaden ni jedan uredaj
13         } else if (!mBTAdapter.isEnabled()) {
14             Intent enableBT = new Intent(BluetoothAdapter.
15                 ACTION_REQUEST_ENABLE);
16             startActivityForResult(enableBT,
17                 BT_ENABLE_REQUEST);
18         } else {
19             new SearchDevices().execute(); // pretraga
20             uredeja
21         }
22     }
23 });

```

Ako korisnik nije povezan na bluetooth, otvori se poruka da ga je potrebno uključiti. Pretraga uređaja vrši se sljedećim dijelom kôda:

```

1  private class SearchDevices extends AsyncTask<Void, Void, List<
2      BluetoothDevice>> {
3
4     @Override

```

```

4     protected List<BluetoothDevice> doInBackground(Void...
5         params) {
6             Set<BluetoothDevice> pairedDevices = mBTAdapter.
7                 getBondedDevices();
8             List<BluetoothDevice> listDevices = new ArrayList<
9                 BluetoothDevice>(); // kreiranje liste
10            for (BluetoothDevice device : pairedDevices) {
11                listDevices.add(device); // stavljanje uređaja u
12                    listu
13            }
14            return listDevices;
15        }
16    }

```

Kad korisnik klikne na dugme za povezivanje, pokrene se povezivanje na odabrani bluetooth uređaj:

```

1 connect.setOnClickListener(new View.OnClickListener() {
2
3     @Override
4     public void onClick(View arg0) {
5         BluetoothDevice device = ((MyAdapter) (listView.
6             getAdapter())).getSelectedItem();
7         Intent intent = new Intent(getApplicationContext(),
8             ControllingActivity.class); // novi intent
9         intent.putExtra(DEVICE_EXTRA, device); // stavljanje
10            podataka o uređaju u intent
11            intent.putExtra(DEVICE_UUID, mDeviceUUID.toString())
12                ;
13            intent.putExtra(BUFFER_SIZE, mBufferSize);
14            startActivity(intent); // zapocinjanje aktivnosti
15        }
16    });

```

Također se izvršio prijelaz u sljedeću aktivnost, kreiranjem takozvanog intenta, pomoću kojeg se započinje aktivnost *ControllingActivity*, koja je zadužena za samo upravljanje lunaparkom. U toj aplikaciji nalazi se izbornik za pokretanje i zaustavljanje sistema, te pokretanje i zaustavljanje muzike, sastavljen od buttona.

```

1 connect.setOnClickListener(new View.OnClickListener() {
2 btnMuzika.setOnClickListener(new View.OnClickListener()
3     {
4
5         @Override
6         public void onClick(View v) {
7
8             try {
9                 mBTSocket.getOutputStream().write(pokreni.
10                    getBytes());
11
12             } catch (IOException e) {
13                 // TODO Auto-generated catch block
14                 e.printStackTrace();
15             }
16         } });

```

Osim funkcionalnosti, svaki dio je potrebno definisati u XML datoteci zaduženoj za dizajn same aplikacije, odnosno, pozicioniranje njenih dijelova i stavljanje određene slike kao poza-

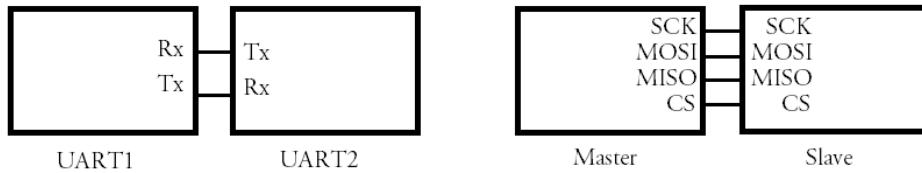
dine.

```
1 connect.setOnClickListener(new View.OnClickListener() {  
2     <Button  
3         android:id="@+id/muzika"  
4         android:layout_width="150dp"  
5         android:layout_height="100dp"  
6         android:text="Muzika"  
7         android:layout_marginTop="420dp"  
8         android:layout_marginLeft="20dp"  
9         android:background="@drawable/muzika"  
10    />
```

Kôd za svaki button je sličan, pa nema potrebe da se svaki stavi u rad. Kad se dodirne button, preko bluetootha se šalje određen niz bajtova povezanim uređaju. Arduinov bluetooth modul prima te nizove bajtova, a u kôdu aplikacije specifirano je koja se radnja izvrši nakon primanja kojeg niza bajtova.

### 4.3 Komunikacija mikrokontrolera i vanjskih uređaja

Mikrokontroler koristi različite načine komunikacije s vanjskim uređajima. U sklopu ovog sistema, vršeno je direktno čitanje izlaznih signala, serijska i master-slave komunikacija.



Slika 4.2: Serijska i master-slave komunikacija

Kod čitanja stanja push buttona, te kontrole LED dioda, te motora, komunikacija je ostvarena direktnim čitanjem stanja elemenata sistema. Stanje može biti niski napon, odnosno logička nula, ili visoki napon, logička jedinicna. Naprimjer, ako je push button pritisnut, on na povezani pin šalje logičku jedinicu, te prema tome sistem započinje kretanje servo motora i očitavanje boje.

Komunikacija sa mobilnim uređajem ostvarena je uz pomoć bluetooth modula koji je povezan serijski s mikrokontrolerom preko UART modula, kao na lijevoj strani slike 4.2. UART modul pretvara paralelni prikazane podatke sa registara u serijski prikazane podatke, odnosno niz bajtova. HC-05 modul omogućuje full duplex komunikaciju, odnosno, oba učesnika u komunikaciji mogu biti i prijemnici, i predajnici, te su linije za prijenos i primanje podataka odvojene. Ovaj modul također omogućuje i sinhronu, i asinhronu komunikaciju - i u odgovarajućim, i u nasumičnim intervalima.

Komunikacija sa modulom za SD karticu ostvarena je preko SPI interfejsa, kao na desnoj strani slike 4.2, koji je sinhronog tipa. Kod ovog interfejsa, jedan uređaj je master, odnosno, onaj koji inicira početak prijenosa komunikacije. U ovom slučaju to je mikrokontroler. Modul za SD karticu je slave, te se s njega preko linija za prijenos podataka isti prenose, onda kada master mikrokontroler preko upravljačkih linija to inicira.

## 5. Eksperimentalna analiza rada sistema

Krerani sistem napravljen je od nekoliko podsistema koji su usklađeni u jednu cjelinu.

Prvi podsistem je osvjetljenje, prikazano na slici 5.1, postignuto pojedinačnim LED diodama, te LED svjetlima iz novogodišnjeg ukrasa. Uključuje se i isključuje po korisnikovoj želji uz pomoć aplikacije.



Slika 5.1: Osvjetljenje sistema

Drugi podsistem je igra, prikazana izbliza na slici 5.2, u kojoj korisnik pogađa boju loptice koja izade iz aparata. Prvo pusti lopticu da uđe u podsistem, odabere na mobitelu da će igrati igru, te onda pogađa boju loptice pritiskom push buttona, koji je smješten u dizajn igre "pogodi balon", koja često bude u zabavnim parkovima. Ako pogodi boju, zasvijetli zelena dioda, a ako promaši, zasvijeli crvena. Boja se čita senzorom za boju, a servo motorima se loptice prebacuju u određenu posudu. U posudama su loptice sortirane po boji.

Treći podsistem je vračara, također vidljiva na slici 5.2, čija RGB dioda se uključuje se i isključuje po korisnikovoj želji uz pomoć aplikacije.

Četvrti podsistem je karusel, odnosno vrtuljak sa konjićima. Napravljen je od kartona po šablonu s Interneta. U sebi sadrži DC motor sa zupčanicima koji usporava njegovo okretanje kako ne bi bio prebrz. Pokreće se i zaustavlja po korisnikovoj želji uz pomoć aplikacije.

Peti podsistem je vrtuljak. Napravljen je od jako detaljnog šablona koji je preuzet s Interneta i modificiran kako bi odgovarao sistemu. Ima sjedala koja se mogu pomjerati. U njemu se nalazi jači DC motor, od 12 V, pošto ga slabiji ne mogu pokrenuti. Pokreće se i zaustavlja po korisnikovoj želji uz pomoć aplikacije.

Šesti podsistem su šoljice, koje se vrte uz pomoć DC motora ispod njih. Pokreću se i zaustavljaju po korisnikovoj želji uz pomoć aplikacije. Kako bi svi DC motori imali dovoljno



**Slika 5.2:** Podsistemi igra i vračara

snage da se pokrenu, samo za njih je osiguran ulazni napon od 6 baterija po 1,5 V, koji je puno bolji za DC motore od standardnih baterija od 9 V.



**Slika 5.3:** Podsistemi karusel, vrtuljak i šoljice

Sedmi podsistem je muzika koja se pušta s SD kartice preko zvučnika. SD kartica je morala biti povezana na posljednja 4 pina Arduino Mega pločice (prije 5 V pinova), pošto su to pinovi namijenjeni za asinhronu komunikaciju u kojoj je Arduino Mega pločica master element. Zvučnik je morao biti povezan na pin 5, 11 ili 46, inače ne bi radio. Pjesma za puštanje morala je biti konvertovana u .wav datoteku, sa frekvencijom 16 KHz, da bi zvuk bio neometano pušten.

Pjesma se pokreće i zaustavlja po korisnikovoj želji uz pomoć aplikacije.

Svaki dio sistema prvotno je zasebno testiran, odnosno, sistem je rađen postepeno. Tek u završnoj fazi projektovanja sistema, svi su dijelovi sklopljeni u jednu cjelinu i usklađen je međusoban rad svih njegovih komponenti.

Motori koji pokreću određene dijelove prvo su testirani spajanjem direktno na napon, a tek onda povezani na modul, te na Arduino. Možda najzahtjevniji dio projektovanja sistema je bio usklađivanje napajanja, brzine vrtnje motora, te težine objekata koji se trebaju pokretati motorom. Kad je brzina motora premalena ili napon previše nizak, dijelovi se ne mogu vrtjeti, a prevelika brzina može oštetiti neki dio. Također, L298N driver za motore, jedini na lokalnom tržištu, nije savršen.

Rad sistema je jako teško prikazati na jednoj slici, puno bolji način je animacija ili video, s obzirom da se većina komponenti vrti, svijetli ili okreće, te se muzika sluša.



**Slika 5.4:** Maketa sistema

Prednosti ovog sistema su da je potpuno jedinstven, stabilan, detaljan, te, zbog pozicioniranja USB kabla od Arduino pločice na lagano dohvataljivo mjesto, podložan nadogradnji i promjeni. Mane sistema su materijal, karton nije dobar materijal za gradnju, te nesavršenost komponenti.

## 6. Zaključak

U ovom seminarskom radu prikazano je koji su to koraci potrebni za projektovanje jednog mikroprocesorskog sistema koji funkcioniše u realnom vremenu. Sistem koji je napravljen je lunapark.

Lunapark ili park zabave je ograđeni prostor sa nizom elemenata namijenjenih zabavi većeg broja ljudi, te zaradi vlasnika. Pojava određenih ograđenih prostora samo za zabavu postoji još od početka prošlog milenija, a smatra se da su moderni parkovi zabave nastali u 19. stoljeću. Danas su veoma popularni takozvani tematski parkovi, odnosno lunaparkovi koji su napravljeni na neku temu.

Za kreiranje ovog sistema, korištena je Arduino Mega pločica, velik broj komponenti, Arduino programsko okruženje, Android Studio programsko okruženje, te Tensorflow Lite neuronska mreža. Ukrasni, nefunkcionalni dio, pravljen je od kartona, te su slike uređivane uz pomoć Adobe Photoshopa. Dobijen je lijep i funkcionalan sistem, ali nesavršen zbog materijala - kartona, te nesavršenih komponenti.

Pokretanje sistema realizirano je kao sistem sa ulaznicama. Korisnik treba pokrenuti Android aplikaciju, te preko kamere izvršiti akviziciju slike. Ako je na slici jedna od ulaznica sistema, korisniku se dopušta da se poveže na bluetooth i upravlja pojedinim dijelovima sistema. Prepoznavanje ulaznice istrenirano je pomoću neuronske mreže. Radnje koje korisnik kontrolira su pokretanje i zaustavljanje vrteški, uključivanje i isključivanje svjetala, pokretanje igre, te pokretanje i zaustavljanje muzike.

# Literatura

- [1] Clavé, S. A., *The global theme park industry*. Cabi, 2007.
- [2] Rhodes, E., “These were the world’s most popular theme parks in 2019 (juli 2020.)”, dostupno na: [travelandleisure.com/travel-news/most-visited-theme-parks-2019](http://travelandleisure.com/travel-news/most-visited-theme-parks-2019) (pristupljeno: oktobar 2020. godine).
- [3] Moskowitz, D. B., “The amusement park: 900 years of thrills and spills, and the dreamers and schemers who built them”, 2019.
- [4] Wroth, W. W., Wroth, A. E., *The London pleasure gardens of the eighteenth century*. Macmillan, 1896.
- [5] Plaisance, M., “World’s fairs (1853–1897): A new idea”, dostupno na: [web.archive.org/web/20071209022544/http://www.icewind.net/themepark/History/h\\_worldfairs.htm](http://web.archive.org/web/20071209022544/http://www.icewind.net/themepark/History/h_worldfairs.htm) 2007. (pristupljeno: oktobar 2020. godine).
- [6] “Royal collection trust”, dostupno na: [rct.uk](http://rct.uk) (pristupljeno: oktobar 2020. godine).
- [7] Levine, A., “Zašto su neki zabavni parki poznati kao parkovi kolica?”, dostupno na: [bs.traasgpu.com/zasto-su-neki-zabavni-parki-poznati-kao-parkovi-kolica/](http://bs.traasgpu.com/zasto-su-neki-zabavni-parki-poznati-kao-parkovi-kolica/) (pristupljeno: oktobar 2020. godine).
- [8] Ricciulli, V., “Coney island’s luna park wants you to help name its three new rides (2019.)”, dostupno na: [ny.curbed.com/2019/8/16/20808552/luna-park-coney-island-rides-naming-contest-nyc-brooklyn](http://ny.curbed.com/2019/8/16/20808552/luna-park-coney-island-rides-naming-contest-nyc-brooklyn) (pristupljeno: oktobar 2020. godine).
- [9] Aaron, C. S., “The american amusement park industry: A history of technology and thrills”, *Business History Review*, Vol. 66, No. 2, 1992.
- [10] Tensorflow, “Službena stranica od tensorflowa”, dostupno na: [tensorflow.org](http://tensorflow.org) (pristupljeno: maj 2020. godine).

# **Prilozi**

# A. Programski kôd

Programski kôd korišten za rad sistema dan je u nastavku.

```
1 #include <Servo.h>
2 #include "SD.h"
3 #define SD_ChipSelectPin 53
4 #include "TMRpcm.h"
5 #include "SPI.h"
6
7 // SD kartica
8 TMRpcm tmrpcm;
9 // Senzor za boju
10 #define S0 2
11 #define S1 3
12 #define S2 4
13 #define S3 5
14 #define sensorOut 6
15 const int crveniButton = 28;
16 const int plaviButton = 26;
17 Servo gornjiServo;
18 Servo donjiServo;
19
20 // svjetla
21 const int crvenaDioda = 29;
22 const int zelenaDioda = 30;
23
24 const int dioda1 = 32;
25 const int dioda2 = 31;
26 const int dioda3 = 33;
27 const int dioda4 = 34;
28 const int svjetla = 36;
29
30 // aplikacija
31 long int data;
32
33 // motori
34 #define enA 10
35 int motor1pin1 = 40;
36 int motor1pin2 = 41;
37
38 int motor2pin1 = 42;
39 int motor2pin2 = 43;
40
41 int motor3pin1 = 44;
42 int motor3pin2 = 45;
43
44 int motor4pin1 = 46;
45 int motor4pin2 = 47;
```

```
46 // vracara
47 int crveni_pin= 11;
48 int zeleni_pin = 12;
49 int plavi_pin = 13;
50
51 // senzor za boju
52 int frekvencija = 0;
53 int color = 0;
54 int crveniButtonState = 0;
55 int plaviButtonState = 0;
56 // citanje buttona
57
58
59 void setup(){
60 // pokretanje SD kartice
61 tmrpcm.speakerPin = 46;
62 Serial.begin(9600);
63 if(!SD.begin(SD_ChipSelectPin)){
64 Serial.println("SD_fail");
65 return;
66 }
67 tmrpcm.setVolume(6);
68
69 //bluetooth pokretanje na serijskom portu 1
70 Serial1.begin(9600);
71
72 // senzor za boju
73 pinMode(crveniButton, INPUT);
74 pinMode(plaviButton, INPUT);
75 pinMode(zutiButton, INPUT);
76 pinMode(S0, OUTPUT);
77 pinMode(S1, OUTPUT);
78 pinMode(S2, OUTPUT);
79 pinMode(S3, OUTPUT);
80 pinMode(sensorOut, INPUT);
81
82 // skaliranje
83 digitalWrite(S0, HIGH);
84 digitalWrite(S1, LOW);
85
86 gornjiServo.attach(22);
87 donjiServo.attach(24);
88
89 // motori
90 pinMode(motor1pin1, OUTPUT);
91 pinMode(motor1pin2, OUTPUT);
92 pinMode(motor2pin1, OUTPUT);
93 pinMode(motor2pin2, OUTPUT);
94 pinMode(motor3pin1, OUTPUT);
95 pinMode(motor3pin2, OUTPUT);
96 pinMode(motor4pin1, OUTPUT);
97 pinMode(motor4pin2, OUTPUT);
98
99 // vracara
100 pinMode(crveni_pin, OUTPUT);
101 pinMode(zeleni_pin, OUTPUT);
102 pinMode(plavi_pin, OUTPUT);
103
```

## Programski kôd

---

```
104 // DIODE
105 pinMode(zelenaDioda, OUTPUT);
106 pinMode(crvenaDioda, OUTPUT);
107 pinMode(dioda1, OUTPUT);
108 pinMode(dioda2, OUTPUT);
109 pinMode(dioda3, OUTPUT);
110 pinMode(dioda4, OUTPUT);
111 pinMode(svjetla, OUTPUT);
112
113 void loop() {
114
115 while(Serial1.available()==0) ;
116
117 donjiServo.write(0);
118 if(Serial1.available()>0) {
119 data = Serial1.parseInt();
120 }
121
122 delay(400);
123 Serial.print(data);
124
125 if (data == 67) {
126     diodePali();
127 }
128 if (data == 76) {
129     diodeGasi();
130 }
131 if (data == 23) {
132     vracara();
133 }
134 if (data == 32) {
135     vracaraUgasi();
136 }
137 if (data == 56) {
138     analogWrite(enA, 160);
139     digitalWrite(motor4pin1, HIGH);
140     digitalWrite(motor4pin2, LOW);
141 }
142 if (data == 65) {
143     digitalWrite(motor4pin1, LOW);
144     digitalWrite(motor4pin2, LOW);
145 }
146 if (data == 34) {
147     digitalWrite(motor1pin1, LOW);
148     digitalWrite(motor1pin2, HIGH);
149 }
150 if (data == 43) {
151     digitalWrite(motor1pin1, LOW);
152     digitalWrite(motor1pin2, LOW);
153 }
154 if (data == 11) {
155     digitalWrite(motor3pin1, HIGH);
156     digitalWrite(motor3pin2, LOW);
157     digitalWrite(motor2pin1, LOW);
158     digitalWrite(motor2pin2, HIGH);
159 }
160 if (data == 21) {
```

## Programski kôd

---

```
162     digitalWrite(motor3pin1, LOW);
163     digitalWrite(motor3pin2, LOW);
164     digitalWrite(motor2pin1, LOW);
165     digitalWrite(motor2pin2, LOW);
166 }
167 if (data == 91) {
168     tmrpcm.play("test.wav");
169 }
170 if (data == 19) {
171     tmrpcm.pause();
172 }
173 if (data == 89) {
174     crveniButtonState = digitalRead(crveniButton);
175     plaviButtonState = digitalRead(plaviButton);
176
177 if (crveniButtonState == HIGH || zutiButtonState == HIGH ||
178     plaviButtonState == HIGH) {
179     senzorZaBoje(crveniButtonState, zutiButtonState,
180                   plaviButtonState);
181 }
182 }
183 if (data == 98) {
184     donjiServo.write(110);
185 }
186 if (data == 99) {
187     crveniButtonState = digitalRead(crveniButton);
188     plaviButtonState = digitalRead(plaviButton);
189
190 if (crveniButtonState == HIGH || zutiButtonState == HIGH ||
191     plaviButtonState == HIGH) {
192     senzorZaBoje(crveniButtonState, zutiButtonState,
193                   plaviButtonState);
194 }
195 diodePali();
196     digitalWrite(motor1pin1, LOW);
197     digitalWrite(motor1pin2, HIGH);
198     digitalWrite(motor3pin1, HIGH);
199     digitalWrite(motor3pin2, LOW);
200     digitalWrite(motor2pin1, LOW);
201     digitalWrite(motor2pin2, HIGH);
202     analogWrite(enA, 160);
203     digitalWrite(motor4pin1, HIGH);
204     digitalWrite(motor4pin2, LOW);
205     vracara();
206     tmrpcm.play("test.wav");
207 }
208 if (data == 88) {
209     diodeGasi();
210     donjiServo.write(110);
211     digitalWrite(motor1pin1, LOW);
212     digitalWrite(motor1pin2, LOW);
213     digitalWrite(motor3pin1, LOW);
214     digitalWrite(motor3pin2, LOW);
215     digitalWrite(motor2pin1, LOW);
216     digitalWrite(motor2pin2, LOW);
217     analogWrite(enA, 160);
218     digitalWrite(motor4pin1, LOW);
219     digitalWrite(motor4pin2, LOW);
```

```
216     vracaraUgasi();
217     tmrpcm.pause();
218 }
219 }
220 }
221 void vracara() {
222     RGB_color(255, 0, 0); // crvena
223     delay(1000);
224     RGB_color(0, 255, 0); // zelena
225     delay(1000);
226     RGB_color(0, 0, 255); // plava
227     delay(1000);
228     RGB_color(255, 255, 125); // roza
229     delay(1000);
230     RGB_color(0, 255, 255); // cijan
231     delay(1000);
232     RGB_color(255, 0, 255); // magenta
233     delay(1000);
234     RGB_color(255, 255, 0); // zuta
235     delay(1000);
236     RGB_color(255, 255, 255); // bijela
237     delay(1000);
238 }
239
240
241 void RGB_color(int crvena, int zelena, int plava) {
242     analogWrite(crveni_pin, crvena);
243     analogWrite(zeleni_pin, zelena);
244     analogWrite(plavi_pin, plava);
245 }
246
247 int readColor() {
248     // Citanje crvenih fotodioda
249     digitalWrite(S2, LOW);
250     digitalWrite(S3, LOW);
251     frekvencija = pulseIn(sensorOut, LOW);
252     int R = frekvencija;
253     delay(50);
254
255     // Citanje zelenih fotodioda
256     digitalWrite(S2, HIGH);
257     digitalWrite(S3, HIGH);
258     frekvencija = pulseIn(sensorOut, LOW);
259     int G = frekvencija;
260     delay(50);
261
262     // Citanje plavih
263     digitalWrite(S2, LOW);
264     digitalWrite(S3, HIGH);
265     frekvencija = pulseIn(sensorOut, LOW);
266     int B = frekvencija;
267     delay(50);
268
269     if((R<45 & R>32 & G<70 & G>50) || (R>70 & G<10 & B <50 & B>30) ) {
270         color = 1; // Crvena
271     }
272     if(R<45 & R>35 & B<40 & B>30) {
273         color = 2; // Roza
```

```
274     }
275     if(R<63 & R>50 & G<55 & G>45 & B<40 & B>29) {
276         color = 3; // Plava
277     }
278     if(R<40 & R>30 & G<50 & G>40 & B<46 & B>38) {
279         color = 4; // Zuta
280     }
281     return color;
282 }
283
284 void senzorZaBoje() {
285     crveniButtonState = digitalRead(crveniButton);
286     plaviButtonState = digitalRead(plaviButton);
287
288 // Ako je pritisnut jedan od buttona, pokrese se dio programa. Ako korisnik pritisne odgovarajuci button, te pogodi koje je boje loptica, on pobijedi, pali se zelena dioda, u suprotnom, pali se crvena.
289     if (crveniButtonState == HIGH || plaviButtonState == HIGH) {
290         gornjiServo.write(120);
291         delay(3000);
292
293         for(int i = 115; i > 55; i--) {
294             gornjiServo.write(i);
295             delay(2);
296         }
297         delay(1000);
298
299         color = readColor();
300         color = readColor();
301         color = readColor();
302         color = readColor();
303
304         delay(1000);
305         switch (color) {
306             case 1: //crvena
307                 if (crveniButtonState == HIGH){
308                     digitalWrite(zelenaDioda, HIGH);
309                     digitalWrite(crvenaDioda, LOW);
310                 }
311                 if (plaviButtonState == HIGH ){
312                     digitalWrite(crvenaDioda, HIGH);
313                     digitalWrite(zelenaDioda, LOW);
314                 }
315                 donjiServo.write(50);
316                 break;
317             case 2: //roza
318                 digitalWrite(crvenaDioda, HIGH);
319                 digitalWrite(zelenaDioda, LOW);
320                 donjiServo.write(75);
321                 break;
322             case 3: // zuta
323                 digitalWrite(crvenaDioda, HIGH);
324                 digitalWrite(zelenaDioda, LOW);
325                 donjiServo.write(100);
326                 break;
327             case 4: // plava
328                 if (plaviButtonState == HIGH) {
```

```
329     digitalWrite(zelenaDioda, HIGH);
330     digitalWrite(crvenaDioda, LOW);
331 }
332 if (crveniButtonState == HIGH ) {
333     digitalWrite(crvenaDioda, HIGH);
334     digitalWrite(zelenaDioda, LOW);
335 }
336 donjiServo.write(125);
337 break;
338 case 0:
339 break;
340 }
341 delay(300);

342 for(int i = 55; i > 20; i--) {
343     gornjiServo.write(i);
344     delay(2);
345 }
346 delay(500);

347 for(int i = 29; i < 115; i++) {
348     gornjiServo.write(i);
349     delay(2);
350 }
351 color=0;
352 }
353 }
354 }

355 }

356 void vrtiMotore() {
357     digitalWrite(motor1pin1, HIGH);
358     digitalWrite(motor1pin2, LOW);

359     digitalWrite(motor2pin1, HIGH);
360     digitalWrite(motor2pin2, LOW);

361     digitalWrite(motor3pin1, HIGH);
362     digitalWrite(motor3pin2, LOW);

363     analogWrite(enA, 160); // upravljanje brzinom
364     digitalWrite(motor4pin1, HIGH);
365     digitalWrite(motor4pin2, LOW);
366 }
367 }
```