ANDROID APLIKACIJA ZA UPRAVLJANJE MBOT ROBOTOM PUTEM BLUETOOTH VEZE I SENZORA AKCELERACIJE

Purgar, Borna

VELEUČILIŠTE U RIJECI

Borna Purgar

ANDROID APLIKACIJA ZA UPRAVLJANJE MBOT ROBOTOM PUTEM BLUETOOTH VEZE I SENZORA AKCELERACIJE

(Završni rad)

Rijeka, 2018.

VELEUČILIŠTE U RIJECI

Stručni studij Telematika

ANDROID APLIKACIJA ZA UPRAVLJANJE MBOT ROBOTOM PUTEM BLUETOOTH VEZE I SENZORA AKCELERACIJE

(Završni rad)

MENTOR

Damir Malnar, mag. ing. el.

STUDENT

Borna Purgar

MBS: 2427000004/15

Rijeka, rujan 2018.

VELEUČILIŠTE U RIJECI

Studij Telematike

Rijeka, 17.6.2018.

ZADATAK
za završni rad

Pristupniku Borni Purgar MBS: 2427000004/15

Studentu stručnoga studija telematike izdaje se zadatak za završni rad – tema završnog rada pod nazivom:

ANDROID APLIKACIJA ZA UPRAVLJANJE mBot ROBOTOM PUTEM BLUETOOTH VEZE I SENZORA AKCELERACIJE

Sadržaj zadatka: Uspostaviti osnovni terminološki sustav koji uključuje Arduino i Android Studio IDE, pametni telefon, mikrokontroler, senzor akceleracije, Bluetooth i aktuatore na mCore razvojnoj pločici iz Makeblock mBot robotske edukacijske platforme. Provesti analizu postojećih rješenja. Na osnovu provedene analize dizajnirati i realizirati aplikaciju za daljinsko upravljanje robotskim sustavom putem Bluetooth veze koristeći senzor akceleracije pametnog telefona za generiranje upravljačkih naredbi. Prikazati glavne programske algoritme. Opisati uporabu sustava.

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 17.6.2018. **Predati do: 15.9.2018.**

Mentor: **Voditelj studija:**

Damir Malnar
(Damir Malnar, mag. ing. el.)

(izv.prof.dr.sc. Alen Jakupović)

Zadatak primio dana: 17.6.2018.

Purgar
(Borna Purgar)

IZJAVA

Izjavljujem da sam završni rad pod naslovom "Android aplikacija za upravljanje mBot robotom putem Bluetooth veze i senzora akceleracije" izradio samostalno pod nadzorom i uz stručnu pomoć mentora Damira Malnara.

Ime i prezime

Borna Purgar

(potpis studenta)

SAŽETAK

Tema ovog završnog rada je izrada Android aplikacije za upravljanje robotskim vozilom putem akcelerometra u Bluetooth mreži i Arduino programa koji upravlja mikrokontrolerom na pločici robota. Rad se sastoji od dva programa od kojih je jedan razvijen u alatu Android Studio, a drugi u Arduino razvojnom okruženju. Kao robotsko vozilo, odabran je *Makeblock mBot* iz razloga što je dostupan i pogodan za edukacijske svrhe, a ima mogućnost programiranja prema vlastitim potrebama u Arduino okruženju. Android aplikacija ima svrhu uključivanja Bluetooth-a, uspostave, održavanja te prekida veze prema robotu i samog upravljanja istim putem akcelerometra. Nadalje, Arduino program na mikrokontroleru robota ima svrhu čitanja, dekodiranja i izvršenja naredbi koje šalje mobilni uređaj putem mreže prema unaprijed zadanim vrijednostima. Sve naredbe koje korisnik šalje sa mobilnog uređaja prema robotu su u obliku vrijednosti generiranih od strane akcelerometra koje su zatim kodirane i prilagođene za slanje.

Ključne riječi: Android, Bluetooth, akcelerometar, mBot, upravljanje

SADRŽAJ

1. UVOD

Daljinsko upravljanje vozilima u svijetu nalazi mnoge svrhe; od igračaka kojima je svrha zabava, do vojnih robota koji se koriste za točno određene zadatke. Osim toga, uz permanentni napredak tehnologije i pristupačnost iste civilnom društvu, danas većina ljudi posjeduje mobilne uređaje koji se temelje na operacijskom sustavu iOS ili Android. Stoga je proizašla ideja za izradom projekta kojem je cilj upravljanje robotskim vozilom putem Bluetooth mreže i mobilnog uređaja.

Sustav se sastoji od upravljačkog dijela u obliku Android mobilnog uređaja i aplikacije na istom; te programa i robotskog vozila kojim korisnik upravlja generirajući vrijednosti akcelerometra pomakom mobilnog uređaja.

2. ANALIZA

Kao što je spomenuto u uvodu, projekt se može podijeliti na dva dijela od kojih prvi dio čini Android aplikacija, a drugi Arduino program. Naime Android aplikacija ima svrhu uključivanja Bluetooth-a, pretrage uređaja u dometu, spajanje na odabrani uređaj te održavanje veze i upravljanje robotskim vozilom. Korisnik robotskim vozilom upravlja na način da pomakom mobilnog uređaja generira vrijednosti x, y i z osi akcelerometra u uređaju i šalje ih prema robotu. S obzirom da se radi o dvodimenzionalnoj okolini, nema potrebe za slanjem vrijednosti osi z sa mobilnog uređaja. Razvojno okruženje u kojem je izrađena aplikacija je Android Studio IDE *(eng Integrated Development Environment).*

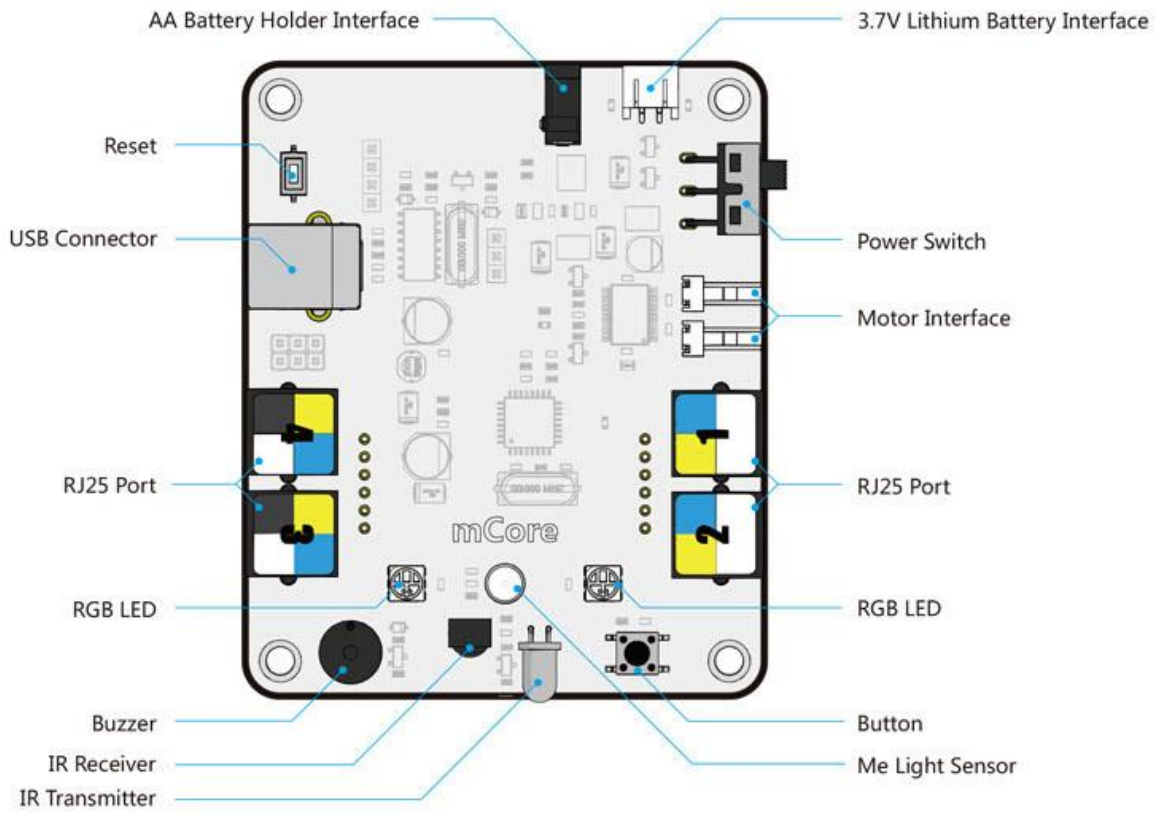
Drugi dio projekta se sastoji od Arduino programa koji čita i dekodira naredbe primljene od strane mobilnog uređaja te na temelju njih upravlja motorima robota. Razvojno okruženje u kojem program razvijen je Arduino IDE.

2.1 M-BOT

Mbot je edukacijski programibilni robot koji je sastavljen od metalnog kučišta, *mCore* pločice koja se temelji na Arduino *Uno* modelu sa mikrokontrolerom ATmega328, dva motora te dva kotača koje motori pokreću. Pločica je otvorenog tipa zbog jednostavnijeg razumijevanja i spajanja sklopovlja te principa rada istih, a na njoj se nalaze sljedeće komponente:

- 4 porta RJ25
- 2 porta za spajanje motora
- 2 RGB LED
- Programibilna tipka
- Zvučnik
- IR transmitter i IR receiver
- Sensor svjetlosti
- USB konektor

Slika 1 - mCore pločica



Izvor: https://www.robotshop.com/media/files/images2/mcore-control-board-mbot-desc-spec.jpg, (10.8 2018.)

Daljinsko upravljanje *mBot-om* je moguće implementirati u obliku Bluetooth ili 2.4GHz komunikacije.Osim toga, moguće je i daljinsko upravljanje putem upravljača na temelju IR *(eng. Infrared)* senzora. Napajanje se vrši putem računala, a u slučaju daljinskog upravljanja se koriste 4 AA baterije spojene u seriju (4 x 1.5 VDC) ili litijska baterija (3.7 VDC). U ovom radu se koristi Bluetooth komunikacija između mobilnog uređaja i robota.

Mbot je moguće programirati u njegovom originalnom *Scratch* okruženju *mblock* koje omogućuje mlađim populacijama jednostavno i grafičko programiranje metodom *"drag and drop"* bez potrebe za pisanjem koda iako ga je moguće kalibrirati u Arduino okruženju. Osim navedenog, mBot je moguće i u potpunosti programirati u Arduino IDE uz službenu *Makeblock* biblioteku što je ujedno i jedan od razloga odabira tog robota.

2.2 BLUETOOTH

Bluetooth je naziv za standard IEEE *(eng. Institute of Electrical and Electronic Engineers)* 802.15 i tehnologiju bežičnog povezivanja i razmjene podataka putem radiovalova kao medija na frekvenciji od 2.4 GHz – 2.480 GHz (https://www.bluetooth.com/bluetooth-technology/radio-versions). Primarna namjena navedene tehnologije je u PAN mreži zbog prilično kratkog dometa koji seže do 10 m. Implementacija tehnologije se temelji na *master-slave* principu u *ad hoc*[1] mreži.

U ovom projektu se u implementaciji bežične komunikacije koristi mobilni uređaj i *Makeblock* Bluetooth modul. Navedeni modul ima radni frekvencijski pojas od 2.4 GHz do 2.48 GHz; radni napon iznosi 5V, kompatibilan je sa verzijama od v2.1 do v4.0, ima sedam pinova od kojih su prva četiri potrebna za komunikaciju sa mikrokontrolerom na pločici *(Makeblock Co., Dual-mode Bluetooth module, pdf).*

Tablica 1 - Raspored pinova Bluetooth modula prema funkcijama

Naziv	**Funkcija**
VCC	Napajanje
GND	Uzemljenje
TX	Odašiljanje signala
RX	Primanje signala
DTR	Data terminal ready
NC	Slobodan pin (*Not Connected)*
RST	Pin za resetiranje modula

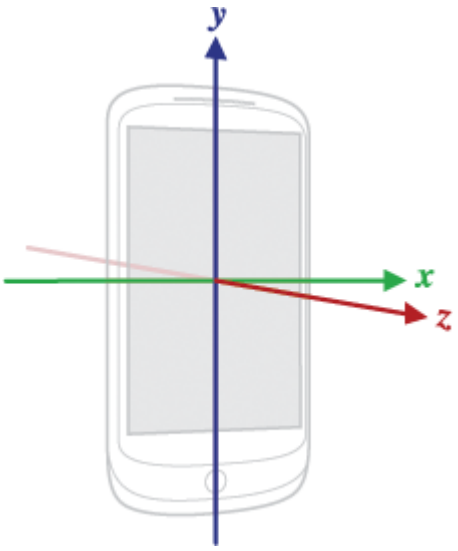
Izvor: autor

[1] Ad hoc – Lokalna mreža formirana od bežičnih uređaja u kojoj se čvorovi mogu lako dodavati ili ukloniti

2.3 AKCELEROMETAR

Akcelerometar je elektromehanički uređaj koji mjeri ubrzanje ili nagib objekta na temelju vrijednosti x, y i z osi. Osi x i y se koriste za potrebe u dvodimenzionalnoj okolini, dok se z os koristi u trodimenzionalnoj i zadana je u obliku sile gravitacije ($g \approx 9.81\ m/s^2$). Uređaj može mjeriti statičke (npr. gravitacija) i dinamičke (npr. pokret mobilnog uređaja, vibracija) sile. Nadalje, uređaj treba biti iznimno osjetljiv jer je potrebno izmjeriti svaku promjenu brzine ili pomak objekta. Slika 2 prikazuje koordinatni sustav na mobilnom uređaju.

Slika 2 - Koordinatni sustav akcelerometra u mobilnom uređaju



Izvor: https://github.com/imthexie/GestureLearner/wiki/Android-Accelerometer, (12.8. 2018.)

Akcelerometar se koristi u mnogim uređajima, a u mobilnim telefonima ima ulogu detekcije orijentacije zaslona uređaja, vibracije, pokreta i ubrzanja. Napon na kojem radi iznosi 5V dok se jakost struje izražava u mikroamperima (μA) ili miliamperima (mA) (https://learn.sparkfun.com/tutorials/accelerometer-basics). U ovom projektu se koristi akcelerometar modela BMC150 koji je ugrađen u Android mobilnom uređaju te ima svrhu detekcije pomaka uređaja. Generirane vrijednosti mogu biti u intervalu od -19 do 19, međutim u algoritmu će se koristiti interval od -10 do 10 te se prema njemu određuje smjer kretanja mBot-a.

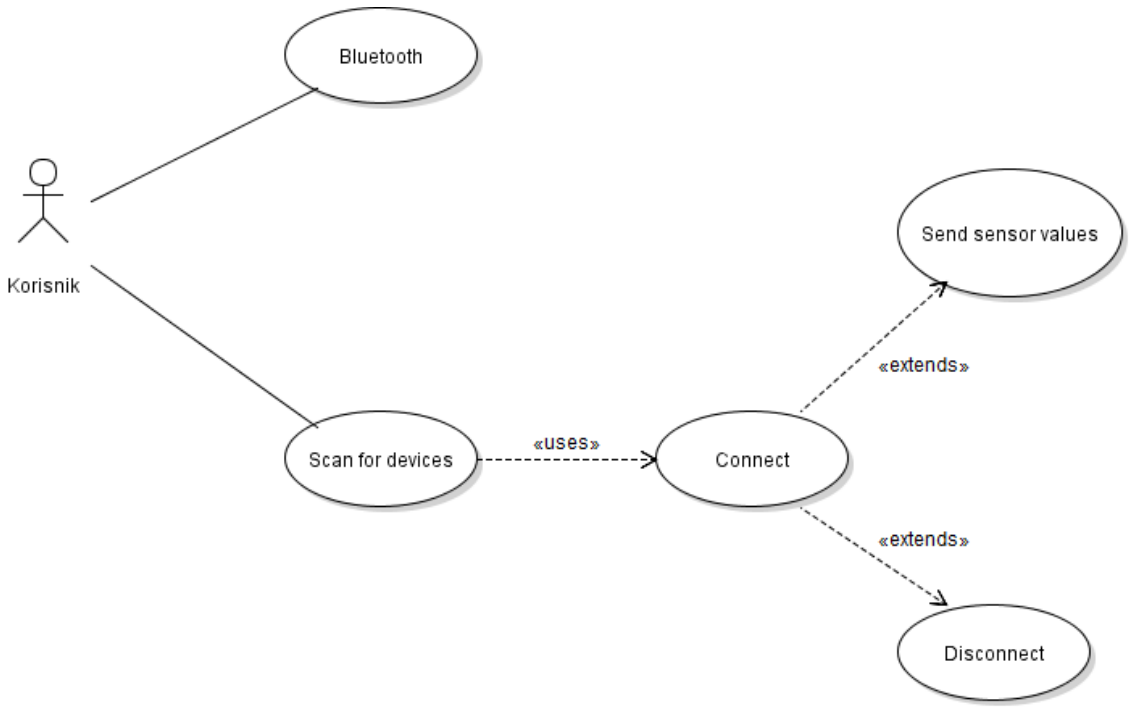
3. DIZAJN

Nakon analize radnji koje sustav treba raditi i komponenti potrebnih za izradu sustava, u dizajnu se određuje kako sustav radi. U ovoj cjelini će se opisati proces razvoja objektno-orijentiranog sustava. Dakle korisnik najprije uključuje Bluetooth na mobilnom uređaju i prikazuje mu se sučelje za pretraživanje uređaja u dometu. U slučaju pronalaska uređaja u blizini, isti se pojavljuje na zaslonu. Odabirom na željeni uređaj se pokreće spajanje na taj uređaj i prikazuje sučelje za upravljanje vozilom. Nakom uspješne uspostave veze, korisnik može upravljati robotskim vozilom. Princip rada sustava će se opisati UML *(eng. Unified Modeling Language)* dijagramima koji su izrađeni u alatima *Violet UML Editor* i *Draw IO*.

3.1 MODEL UPORABE

Prvi korak objektno-orijentirane analize je model uporabe. Korisnik je u interakciji sa sustavom u slučajevima kada odabire uključivanje Bluetooth-a, pretragu uređaja u dometu, upravlja robotskim vozilom i prekida vezu prema istom. Na slici 3 je vidljiv dijagram uporabe sustava.

Slika 3 - Dijagram uporabe sustava



Izvor: autor

Scenarij uporabe:

1. Korisnik uključuje Bluetooth na mobilnom uređaju
2. Aplikacija šalje upit za potvrdom uključivanja Bluetooth-a na uređaju
3. Uključivanje Bluetooth-a nakon potvrde
4. Korisnik odabire pretragu uređaja u dometu
5. Pretraga uređaja u dometu
6. Korisnik odabire robotsko vozilo za spajanje
7. Spajanje sa robotskim vozilom
8. Registriranje senzora na mobilnom uređaju
9. Korisnik šalje podatke sa akcelerometra mobilnog uređaja (upravlja vozilom)
10. Korisnik prekida vezu sa robotskim vozilom

3.2 MODEL KLASA

Sljedeći korak objektno-orijentirane analize je model klasa. Kroz model klasa se mogu vidjeti klase i objekti koji se spominju u modelu uporabe. Kroz izgradnju sustava se dijagram klasa konstantno nadopunjuje, a na slici 4 je prikazan dijagram klasa sustava u ranoj fazi objektno-orijentirane analize.

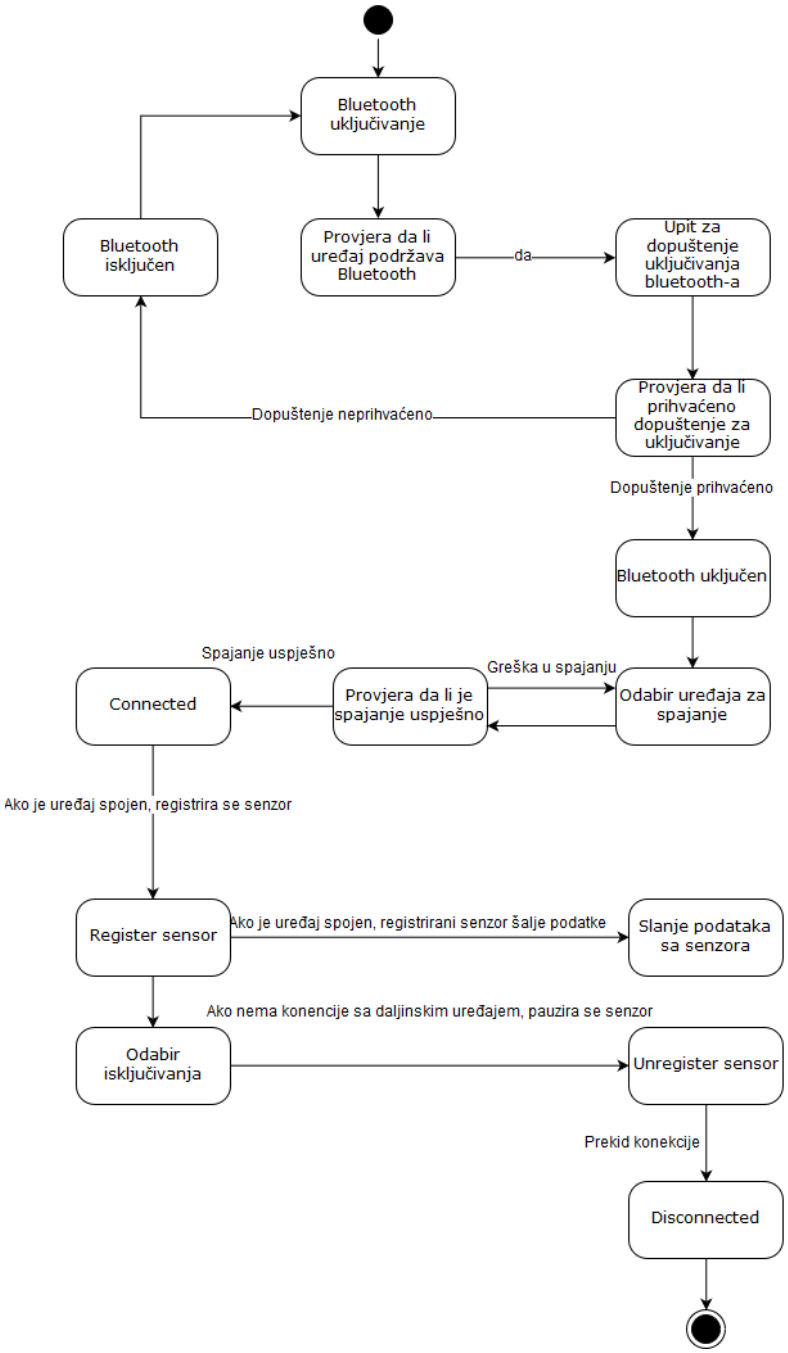
Slika 4 - Dijagram klasa u ranoj fazi objektno-orijentirane analize

Izvor: autor

3.3 MODEL DINAMIKE

Kroz dijagram dinamike sustava se može uočiti korisnička interakcija sa sustavom kroz promjene stanja sustava nakon određenih događaja. Aplikacija ne može imati mnogo stanja i stoga je dovoljan jedan dijagram stanja sustava.

Slika 5 - Dijagram stanja sustava kroz korisničku interakciju

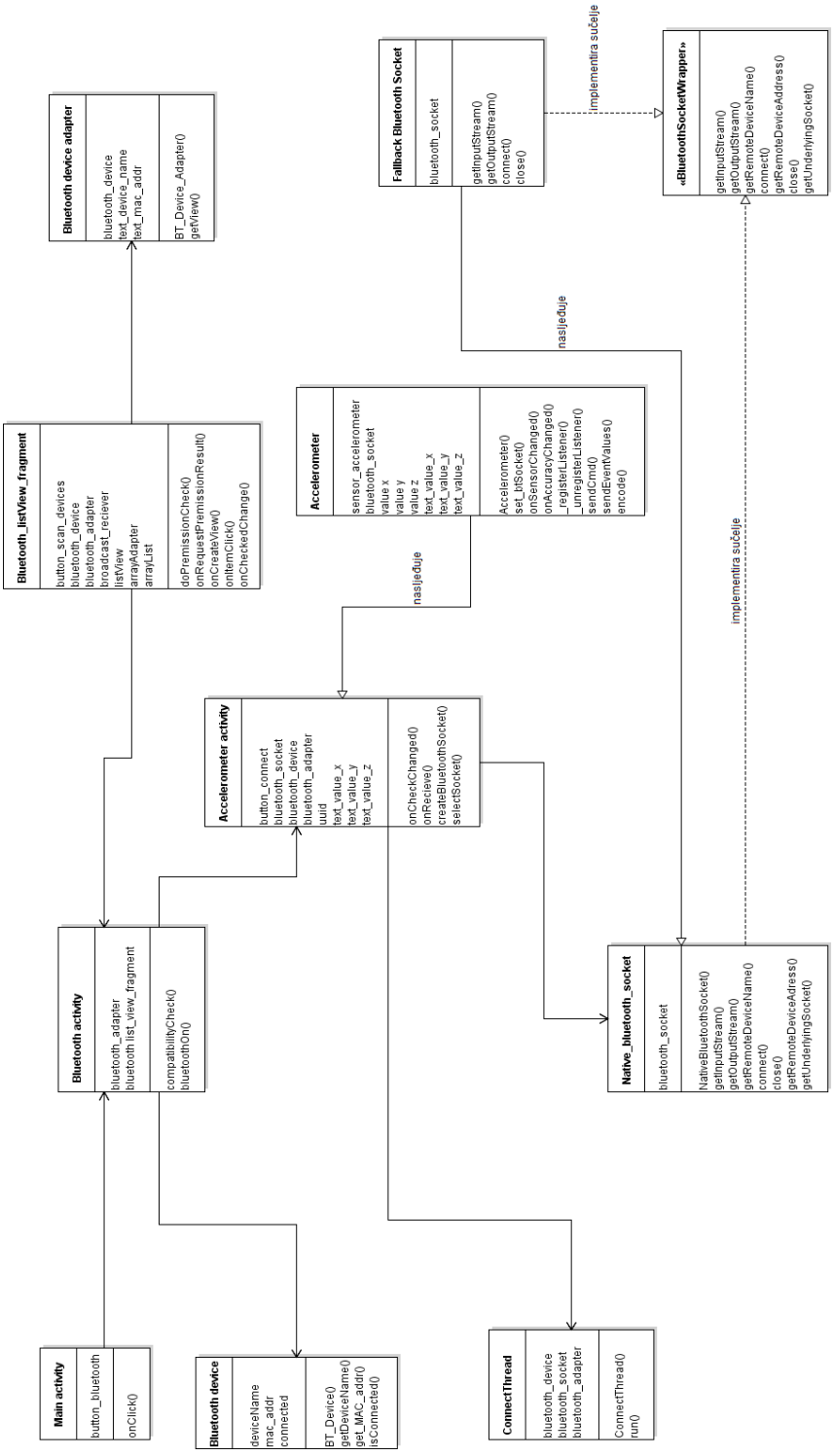


Izvor: autor

3.4 DETALJAN MODEL KLASA

Cilj objektno-orijentiranog dizajna je planiranje sustava, razgradnja logičke strukture aplikacije uz uporabu objekata koji su identificirani u fazi objektno-orijentirane analize (dr. sc. *A. Jakupović, Veleučilište u Rijeci, Programiranje 2, Predavanja ppt 4*). Sljedeći dijagram, kao prvi korak objektno-orijentiranog dizajna je detaljan dijagram klasa sustava koji se nalazi na slici 6.

Slika 6 - Detaljan dijagram klasa sustava



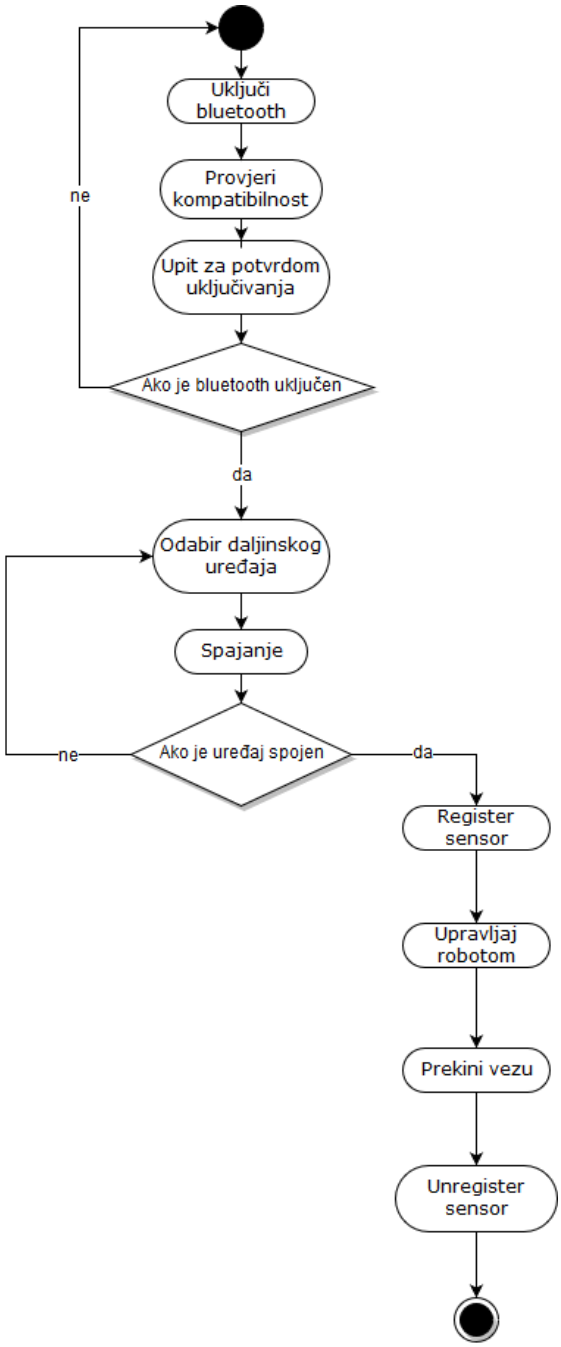
Izvor: autor

Kao što je vidljivo na slici, u detaljnom dijagramu klasa se nalaze i novi objekti koji u prethodnoj fazi nisu bili uočeni, a koji su elementi sustava.

3.5 MODEL AKTIVNOSTI

Sljedeći korak objektno-orijentiranog dizajna je dijagram aktivnosti sustava. Kroz dijagram aktivnosti se može uočiti tok aplikacije i mogući scenariji u određenim slučajevima. Slika 7 prikazuje dijagram aktivnosti sutava.

Slika 7 - Dijagram aktivnosti sustava

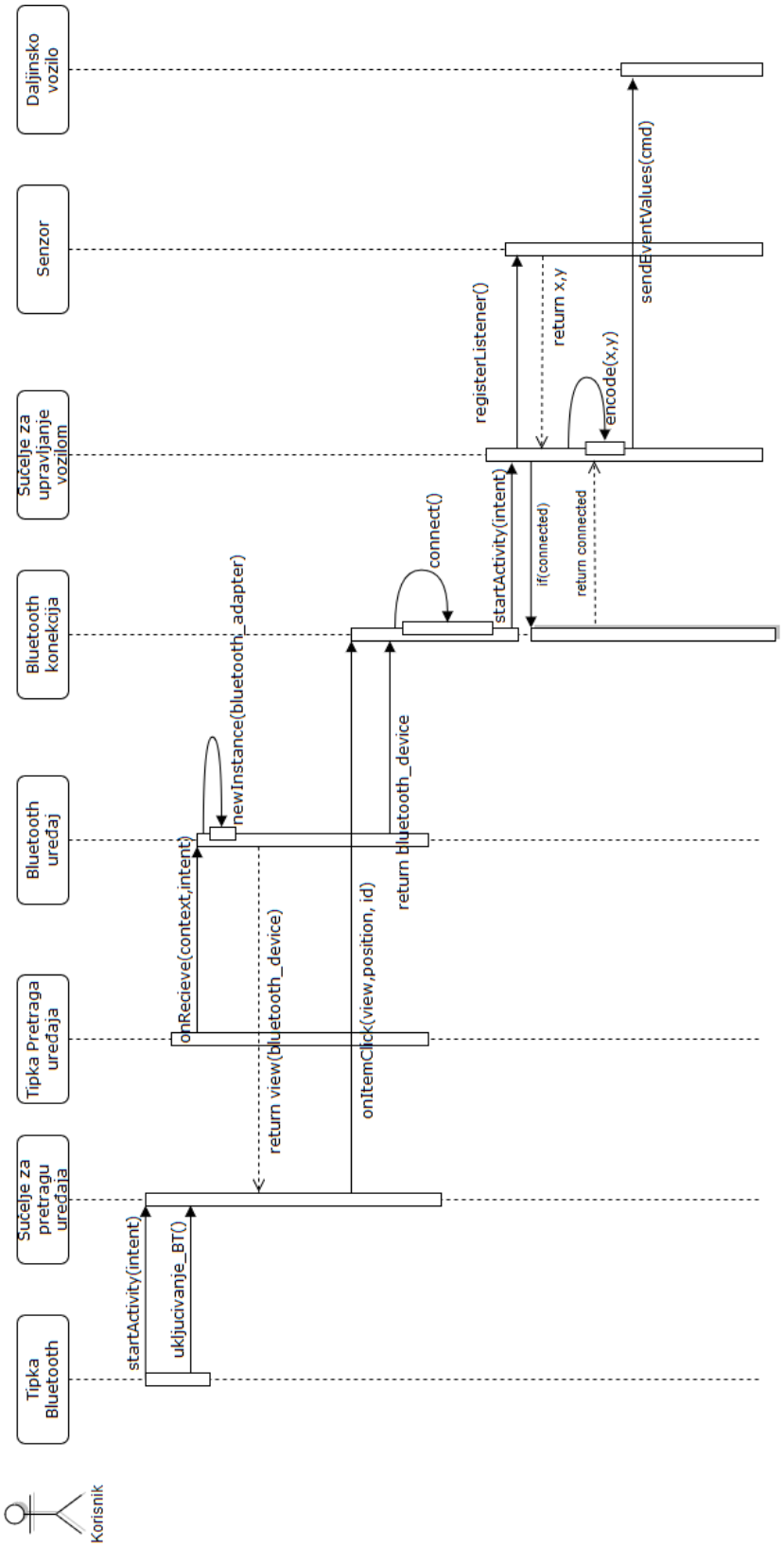


Izvor: autor

3.6 MODEL SLIJEDA AKTIVNOSTI

Nakon dijagrama aktivnosti, zadnji korak objektno-orijentiranog dizajna je dijagram slijeda aktivnosti u programu. On opisuje vremenski-orijentiranu dinamiku sustava i prikazuje tijekove kontrole i scenarije ponašanja programa kroz interakciju sa korisnikom. Pomoću dijagrama slijeda aktivnosti se jasno može uočiti ponašanje programa u realnom vremenu.

Slika 8 - Dijagram slijeda aktivnosti



Izvor: autor

4. IMPLEMENTACIJA

Implementacija objektno-orijentiranog sustava se može podijeliti u dva dijela, a to su upravljački (Android aplikacija) i upravljani (Arduino program). Najprije će se pojasniti implementacija softverskog riješenja upravljanog dijela, a zatim upravljačkog.

4.1 ARDUINO PROGRAM

Kao što je pojašnjeno u cjelini 2.1, *mcore* pločica ima na sebi 4 RJ25 porta koji su namijenjeni za spajanje eksternih komponenti kao što su senzori, LED, mikrofoni i slični. Svim portovima su zajednički pinovi SCL *(Serial Clock Line)*, SDA *(Serial Data Line)*, GND *(Ground)* i 5V, međutim dva porta imaju preostale pinove namijenjene za analogni I/O (A1, A3), a dva porta za digitalni I/O (10, 12). Ostali pinovi (A0, A2, 9, 11) su slobodni, ali nisu izravno dostupni korisniku. Motori su spojeni na pinove 9 i 10.

Program je razvijen uz pomoć službene *Makeblock* biblioteke koja sadrži funkcije potrebne za rad komponenata. Bluetooth komunikacija je implementirana na 115200 Bd *(baud)*. Nadalje, naredbe koje određuju smjer kretanja robotskog vozila su implementirane u obliku jednog znaka (*char - 8 bit)* i svaka naredba mora biti jedinstvena zbog jedinstvenih scenarija. Definiran je 21 slučaj pomaka akcelerometra mobilnog uređaja koji određuju smjer i brzinu kretanja robotskog vozila. Program primljene naredbe dekodira te prema njima pokreće motore, a tablica 2 opisuje smjer i brzinu kretanja robota prema određenim naredbama.

Tablica 2 - Kodirani znakovi prema naredbama

Znak	Naredba – Radni ciklus
q	Naprijed - 100
w	Naprijed – 180
e	Naprijed - 250
r	Natrag - 100
t	Natrag - 180
z	Natrag - 250
a	Lijevo -100
d	Desno – 100
g	Naprijed + lijevo - 100
h	Naprijed + lijevo -180
j	Naprijed + lijevo -250
k	Naprijed + desno - 100
l	Naprijed + desno - 180
y	Naprijed + desno - 250
x	Natrag + lijevo - 100
c	Natrag + lijevo - 180
v	Natrag + lijevo - 250
b	Natrag + desno - 100
n	Natrag + desno - 180
m	Natrag + desno - 250
s	Stop

Izvor: autor

U nastavku se nalazi programski kod za mikrokontroler koji je razvijen u Arduino IDE.

Prije razvoja programa je potrebno uključiti sljedeće biblioteke:

Kod 1 – Uključene biblioteke u Arduino programu

```
#include <MeMCore.h>

#include <SoftwareSerial.h>

#include <Wire.h>

#include <Arduino.h>
```

Izvor: autor

Zatim slijedi inicijalizacija globalnih varijabli i početna funkcija *setup()* koja se izvodi kada se program pokrene, a koristi se za određivanje modova pinova, inicijalizaciju varijabli i ostale funkcije koje se trebaju izvesti jednom.

Kod 2 – Inicijalizacija globalnih varijabli i setup() funkcija

```
MeDCMotor m1(9);
MeDCMotor m2(10);


uint8_t v1 = 100;
uint8_t v2 = 180;
uint8_t v3 = 250;
float x = 0.6;
char cmd;


void setup() {
  Serial.begin(115200);
}
```

Izvor: autor

Sljedeći dio Arduino programa je *loop()* funkcija koja se izvodi permanentno kao petlja, sve dok mikrokontroler ima napajanje. U toj funkciji se izvodi glavni dio programa. Navedena funkcija će biti prikazana u dijelovima zbog veličine programskog koda.

Kod 3 – loop() funkcija u prvi dio

```
void loop() {
  if (Serial.available()) {
    cmd = readCMD();
    switch (cmd) {
      case 'q':               /****  forward  ****/
        m1.run(v1);
        m2.run(-v1);
        break;
      case 'w':
        m1.run(v2);
        m2.run(-v2);
        break;
      case 'e':
        m1.run(v3);
        m2.run(-v3);
        break;
      case 'r':               /****  back ****/
        m1.run(-v1);
        m2.run(v1);
        break;
      case 't':
        m1.run(-v2);
        m2.run(v2);
        break;
      case 'z':
        m1.run(-v3);
        m2.run(v3);
        break;
```

Izvor: autor

Kod 4 – loop() funkcija drugi dio

```
case 'a':          /****  left  ****/
  m1.run(v1);
  m2.run(v1);
  break;
case 'd':                /****  right  ****/
  m1.run(-v1);
  m2.run(-v1);
  break;
case 'f':                /****  forward + left  ****/
  m1.run(v1);
  m2.run(-v1 * x);
  break;
case 'g':
  m1.run(v2);
  m2.run(-v2 * x);
  break;
case 'h':
  m1.run(v3);
  m2.run(-v3 * x);
  break;
case 'j':                /****  forward + right  ****/
  m1.run(v1 * x);
  m2.run(-v1);
  break;
case 'k':
  m1.run(v2 * x);
  m2.run(-v2);
  break;
case 'l':
  m1.run(v3 * x);
  m2.run(-v3);
  break;
```

Izvor: autor

Kod 5 – loop() funkcija treći dio

```
    case 'y':          /****  back + left  ****/
      m1.run(-v1);
      m2.run(v1 * x);
      break;
    case 'x':
      m1.run(-v2);
      m2.run(v2 * x);
      break;
    case 'c':
      m1.run(-v3);
      m2.run(v3 * x);
      break;
    case 'v':               /****  back + right  ****/
      m1.run(-v1 * x);
      m2.run(v1);
      break;
    case 'b':
      m1.run(-v2 * x);
      m2.run(v2);
      break;
    case 'n':
      m1.run(-v3 * x);
      m2.run(v3);
      break;
    case 's':               /****  stop  ****/
      m1.stop();
      m2.stop();
      break;
   }
  }
}
```

Izvor: autor

Osim glavnih funkcija *setup()* i *loop()* je još implementirana funkcija za čitanje podataka primljenih putem Bluetooth mreže. Funkcija je prilično jednostavna te ima svrhu sekvencijalnog čitanja znakova kako bi primljene naredbe bile izvršene redom

Kod 6 – funkcija za čitanje naredbi primljenih sa mreže

```
char readCMD() {
  char bluetoothInput = '\0';
  bluetoothInput = (char) Serial.read();


  return bluetoothInput;
}
```

Izvor: autor

4.2 ANDROID APLIKACIJA

Nakon razrade implementacije Arduino programa, kroz pseudokod i zatim programski kod se opisuje implementacija metoda Android aplikacije koje su relevantne za temu rada. Sljedeći pseudokod opisuje algoritam uključivanja, pretraživanja uređaja, spajanja i održavanja Bluetooth veze prema mikrokontroleru na robotu.

Kod 7 – pseudokod provjeraKompatibilnosti()

```
1. dohvati korisnički uređaj (adapter)
2. if (adapter == null)
       then postavi poruku "Uređaj ne podržava Bluetooth"
3. end if
```

Izvor: autor

Sljedeća metoda je sastavni dio *Broadcast Reciever-a* koji je sastavni dio biblioteke Android Studio IDE, a ima ulogu pretraživanja uređaja u dometu sa uključenim Bluetooth-om.

Kod 8 – pseudokod bluetoothUkljucivanje()

```
1. if (adapter == enabled)
       then ukljuci bluetooth
2. end if
```

Izvor: autor

Kod 9 – pseudokod onRecieve()

```
1. if (bluetoothUređaj == found)
        then kreiraj novi objekt tipa uređaj
                if (bluetoothUređaj != connected)
                        then dodaj pronađeni uređaj na listu
                end if
2. end if
```

Izvor: autor

Nakon što korisnik odabere uređaj sa zaslona, pokreće se nova aktivnost i kreira se objekt tipa akcelerometar koji kasnije nakon uspješne uspostave veze registrira *listener* za senzor i omogućuje upravljanje akcelerometrom. Odabirom na pronađeni uređaj na listi zaslona, korisnik također pokreće nit za uspostavu i održavanje Bluetooth komunikacije prema robotu. Pseudokod u nastavku opisuje algoritam za prethodno navedene aktivnosti.

Kod 10 – pseudokod selectSocket()

```
1. kreiraj privremeni socket
2. if (konekcija == sigurna)
        then kreiraj siguran socket (UUID)
        else kreiraj nesiguran socket (UUID)
3. postavi vrijednost varijable socket na privremeni socket
4. end if
```

Izvor: autor

Kod 11 – pseudokod ConnectThread

```
1. prekini pretraživanje uređaja
2. kreiraj i pokreni nit za bluetooth komunikaciju
3. connect
4. if (bluetoothSocket != null)
       then postavi zastavicu isConnected = true
       kreiraj i pokreni nit za slanje podataka
5. end if
```

Izvor: autor

U prethodnom algoritmu se može uočiti treći korak (connect) koji pokreće niz procesa koji su opisani koracima u nastavku.

Kod 12 – pseudokod connect()

```
1. while (selectSocket == true)
2. prekini pretraživanje uređaja
3. pokušaj spajanje
4. if (konekcija = true)
       then postavi zastavicu success
       break
5. end if
6. if (konekcija != true)
       kreiraj novi socket
       pričekaj 500 ms
       pokušaj spajanje
       break
7. return socket
```

Izvor: autor

Nakon uspješnog spajanja sa daljinskim uređajem se registrira *listener* za akcelerometar na mobilnom uređaju. Zadaća *listener-a* je pokretanje metode *onSensorChanged()* detektiranjem promjene pomaka mobilnog uređaja. Unutar navedene metode se generiraju, kodiraju i šalju vrijednosti x,y i z osi; što će biti opisano sljedećim algoritmom:

Kod 13 – pseudokod upravljanje robotom

```
1. register listener
2. if (bluetoothSocket != null)
        postavi vrijednost x na vrijednost dobivenu pomakom senzora
        postavi vrijednost y na vrijednost dobivenu pomakom senzora
        postavi vrijednost z na vrijednost dobivenu pomakom senzora
        kodiraj vrijednosti u komandu
        pošalji vrijednost na spojeni uređaj u formatu znakova
3. else unregister listener
4. end if
```

Izvor: autor

Valja napomenuti da se vrijednost osi z ne šalje, već samo prikazuje na zaslonu jer su za upravljanje vozilom potrebne samo osi x i y.

Implementacija prethodno navedenih radnji će se bolje pojasniti programskim kodom u nastavku. Aplikacija je razvijena u programskom jeziku *Java*, u alatu Android Studio IDE, a sastoji se od sljedećih sedam klasa:

- **BT_Activity**
- **BT_Device**
- **BT_Device_Adapter**
- **BT_ListView_Fragment**
- **RC_Main_Activity**
- **Accelerometer_Activity**
- **Accelerometer**

Glavna aktivnost (RC_Main_Activity) ima ulogu jednostavnog sučelja u kojem korisnik pritiskom na tipku uključuje Bluetooth i pokreće novu aktivnost (BT_Activity). Primjer koda 14 prikazuje metode za provjeru kompatibilnosti i uključivanje Bluetooth-a.

Kod 14 – compatibilityCheck()

```
private void compatibilityCheck(){
    bt_adapter = BluetoothAdapter.getDefaultAdapter();

    if (bt_adapter == null) {
        new AlertDialog.Builder(this)
                .setTitle("Error")
                .setMessage("No Bluetooth service")
                .setPositiveButton("Exit", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        System.exit(0);
                    }
                })
                .show();
    }
}
```

Izvor: autor

Kod 15 – bluetoothOn()

```
protected void bluetoothOn(){


   compatibilityCheck();


   if (!bt_adapter.isEnabled()) {
      Intent enableBT = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(enableBT, REQUEST_BLUETOOTH);
   }


}
```

Izvor: autor

Iz primjera koda je vidljivo da se naprije poziva metoda provjere kompatibilnosti, a zatim se uključuje Bluetooth na mobilnom uređaju.

Nakon što se na mobilnom uređaju uključi Bluetooth, aplikacija u novoj aktivnosti pritiskom na tipku pretražuje sve uređaje u dometu sa uključenim Bluetooth-om, što je realizirano putem *Broadcast Reciever-a*. Prvi primjer koda prikazuje tipku za pretraživanje uređaja, a drugi *Broadcast Reciever.*

Kod 16 – button Scan

```
btnScan = (ToggleButton) v.findViewById(R.id.btnScan);



btnScan.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
   @Override
   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
      if (isChecked) {
         mAdapter.clear();
         getActivity().registerReceiver(bReciever, filter);
         bt_adapter.startDiscovery();
      } else {
         getActivity().unregisterReceiver(bReciever);
         bt_adapter.cancelDiscovery();
      }
   }
});
```

Izvor: autor

Kod 17 – Broadcast Reciever

```
private final BroadcastReceiver bReciever = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (BluetoothDevice.ACTION_FOUND.equals(action)) {

            BluetoothDevice _device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

            bt_device = new BT_Device(_device.getName(), _device.getAddress(),false);

            if (!bt_device.isConnected()){
                mAdapter.add(bt_device);
                mAdapter.notifyDataSetChanged();
            }
        }
    }
};
```

Izvor: autor

Nakon odabira uređaja sa liste se pokreće nova aktivnost (Accelerometer_Activity) što opisuje kod 18:

Kod 18 – odabir Bluetooth uređaja

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    if (null != mListener) {
        mListener.onFragmentInteraction(bt_deviceList.get(position).getDeviceName());

        Intent intent = new Intent(getActivity(), Accelerometer_Activity.class);

        intent.putExtra("MAC_addr", bt_deviceList.get(position).get_MAC_addr());

        btnScan.setChecked(false);

        startActivity(intent);
    }
}
```

Izvor: autor

Prilikom pokretanja nove aktivnosti (Accelerometer_Activity) se pokreće nit *ConnectThread* u kojoj se mobilni uređaj spaja sa robotskim vozilom, a u zadanoj metodi *onCreate()* se kreira objekt klase *Accelerometer* pomoću kojeg se može jednostavnije manipulirati senzorom u cijeloj aplikaciji. Programski kod u prilogu opisuje navedene radnje.

Kod 19 – *onCreate()* metoda u Accelerometer_Activity

```
accMeter = new Accelerometer(this, textX, textY, textZ);

btnConn = (ToggleButton) findViewById(R.id.connBtn);

String mac_adr = getIntent().getStringExtra("MAC_addr");

_btAdapter = BluetoothAdapter.getDefaultAdapter();
_btDevice = _btAdapter.getRemoteDevice(mac_adr);



final ConnectThread thr = new ConnectThread(_btDevice, _btAdapter);
   thr.start();
```

Izvor: autor

Nakon što se otvori nova aktivnost (Accelerometer_Activity), pokreće se *run()* metoda *ConnectThread* niti, čija je svrha spajanje sa robotskim vozilom, a koju opisuje kod 20.

Kod 20 – ConnectedThread run() prvi dio

```
public void run() {


   btAdapter.cancelDiscovery();
   try {
      runOnUiThread(new Runnable() {
         public void run() {
             dialog = ProgressDialog.show(Accelerometer_Activity.this, "Connecting to:
"+bTDevice.getName(),
                                  "In progress...", true, false);
         }
      });
      Thread.sleep(1000);
         connect();
         if (ConnectThread.currentThread().isInterrupted()) {
            btSocket.close();
            return;
         }
      if(btSocket != null){
            try {
               accMeter._registerListener();


            } catch (Exception e) {
               e.printStackTrace();
            }
      }
      dialog.dismiss();
      runOnUiThread(new Runnable() {
```

Izvor: autor

Kod je zbog veličine morao biti podijeljen na dva dijela, a drugi dio metode *run()* se nalazi u nastavku.

Kod 21 – ConnectedThread run() drugi dio

```
        runOnUiThread(new Runnable() {
            public void run() {


                Toast msg = Toast.makeText(Accelerometer_Activity.this, "Connected",
Toast.LENGTH_SHORT);
                msg.setGravity(Gravity.CENTER, 0, 0);
                msg.show();
            }
        });


    } catch (IOException e) {
        e.printStackTrace();
        try {
            btSocket.close();


        } catch (IOException e1) {
            Log.d("Bluetooth_device", "Could not close the client socket");


        }
    } catch (InterruptedException e) {
        e.printStackTrace();
        Log.d("Connect_thread", "INTERRUPTED");
    }


}
```

Izvor: autor

Metoda *connect()* u prethodnom primjeru pokreće niz sljedećih procesa:

Kod 22 – connect() prvi dio

```
public NativeBluetoothSocket connect() throws IOException {
   boolean success = false;

   while (selectSocket()) {
      _btAdapter.cancelDiscovery();
      try {
         _btSocket.connect();
         accMeter.set_btSocket(_btSocket);
         success = true;
         break;
      } catch (IOException e) {
         try {
            if(_btSocket != null) {
               _btSocket = new FallbackBluetoothSocket(_btSocket.getUnderlyingSocket());
               Thread.sleep(500);
               _btSocket.connect();
               accMeter.set_btSocket(_btSocket);
               success = true;
               break;
            }
            else{
               break;
            }
         } catch (FallbackException e1) {
            Log.w("BT", "Could not initialize FallbackBluetoothSocket classes.", e);
         } catch (InterruptedException e1) {
            Log.w("BT", e1.getMessage(), e1);
         } catch (IOException e1) {
```

Izvor: autor

Također, zbog veličine koda se i ovaj kod morao podijeliti na dva dijela, a se nalazi u prilogu:

Kod 23 – connect() drugi dio

```
} catch (IOException e1) {
            Log.w("BT", "Fallback failed. Cancelling.", e1);
               runOnUiThread(new Runnable() {
                  public void run() {
                     Toast msg = Toast.makeText(Accelerometer_Activity.this, "Connect to remote
device failed", Toast.LENGTH_LONG);
                     msg.setGravity(Gravity.CENTER, 0, 0);
                     msg.show();
                     btnConn.setChecked(true);
                     dialog.dismiss();
                  }
               });
            break;
         }
      }
   }
   if (!success) {
      throw new IOException("Could not connect to device: "+ _btDevice.getAddress());
   }
   return _btSocket;
}
```

Izvor: autor

Nakon što se mobilni uređaj uspješno spoji sa robotskim vozilom, registrira se *listener* za senzor te korisnik može upravljati robotom. Programski kod u prilogu opisuje metodu *onSensorChanged()* koja služi upravljanju robotskim vozilom.

Kod 24 – onSensorChanged()

```
@Override
public void onSensorChanged(SensorEvent event) {
   WindowManager windowMgr = (WindowManager)
_context.getSystemService(Context.WINDOW_SERVICE);
   int rotationIndex = windowMgr.getDefaultDisplay().getRotation();
   if(_btSocket != null){
      if (rotationIndex == 0 || rotationIndex == 2) {
         x = event.values[1];
         y = event.values[0];
         z = event.values[2];
      } else {
         x = -event.values[0];
         y = -event.values[1];
         z = event.values[2];
      }
      ((Accelerometer_Activity) _context).runOnUiThread(new Runnable(){
         @Override
         public void run(){
            txtX.setText(String.format(Locale.getDefault(),"x = %s - [%.4f]",(int)Math.round(x), x));
            txtY.setText(String.format(Locale.getDefault(),"y = %s - [%.4f]",(int)Math.round(y), y));
            txtZ.setText(String.format(Locale.getDefault(),"z = %s - [%.4f]",(int)Math.round(z), z));
         }
      });
      sendEventValues();
   }
   else{
      _unregisterListener();
   }
}
```

Izvor: autor

U prethodnom kodu je moguće vidjeti metodu *sendEventValues()* koja dobivene vrijednosti osi x, y, i z zaokružuje na cijeli broj te šalje putem Bluetooth mreže.

Kod 25 – sendEventValues()

```
private void sendEventValues() {
    if(_btSocket != null) {
        int xx = (int) Math.round(x);
        int yy = (int) Math.round(y);

        char cmd = encode(xx, yy);

        sendCmd("<" + cmd + ">");

        Log.d("Data", xx + "\t" + yy + "\tcmd: " + cmd);
        try {
            Thread.sleep(50);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }
}
```

Izvor: autor

U prethodnoj metodi se može najprije uočiti metoda *encode()*, a zatim *sendCmd()*. Metoda *encode()* ima ulogu vrijednosti dobivene pomakom akcelerometra kodirati u znak koji će kasnije određivati smjer i brzinu kretanja mBot-a. Kod će se zbog veličine podijeliti na tri dijela.

Kod 26 – encode() prvi dio

```
private char encode(int x, int y){
    char cmd = '\0';
    /******************stop******************/
    if(x >= -2 && y <= 2){
        cmd = 's';
    }
    /******************fw******************/
    if((x >= 3 && x <= 5) && (y <= 3 && y >= -3)){
        cmd = 'q';
    }
    if((x >= 5 && x <= 7) && (y <= 3 && y >= -3)){
        cmd = 'w';
    }
    if((x >= 7 && x <= 10) && (y <= 3 && y >= -3)){
        cmd = 'e';
    }
    /******************back******************/
    if((x <= -3 && x >= -5) && (y <= 3 && y >= -3)){
        cmd = 'r';
    }
    if((x <= -5 && x >= -7) && (y <= 3 && y >= -3)){
        cmd = 't';
    }
    if((x <= -7 && x >= -10) && (y <= 3 && y >= -3)){
        cmd = 'z';
    }
```

Izvor: autor

Kod 27 – encode() drugi dio

```
/******************left******************/
if((y >= 3 && y <= 5) && (x <= 3 && x >= -3)){
    cmd = 'a';
}
/******************right******************/
if((y <= -3 && y >= -5) && (x <= 3 && x >= -3)){
    cmd = 'd';
}
/******************fw+l******************/
if((y >= 2 && y >= 5) && (x >= 2 && x <= 4)){
    cmd = 'f';
}
if((y >= 2 && y >= 5) && (x >= 4 && x <= 6)){
    cmd = 'g';
}
if((y >= 2 && y >= 5) && (x >= 6 && x <= 8)){
    cmd = 'h';
}
```

Izvor: autor

Kod 28 – encode() treći dio

```
/******************fw+r******************/
    if((y <= -2 && y >= -5) && (x >= 3 && x <= 5)){
        cmd = 'j';
    }
    if((y <= -2 && y >= -5) && (x >= 5 && x <= 7)){
        cmd = 'k';
    }
    if((y <= -2 && y >= -5) && (x >= 7 && x <= 10)){
        cmd = 'l';
    }
    /******************b+l******************/
    if((y >= 2 && y <= 5) && (x <= -2 && x >= -4)){
        cmd = 'y';
    }
    if((y >= 2 && y <= 5) && (x <= -4 && x >= -6)){
        cmd = 'x';
    }
    if((y >= 2 && y <= 5) && (x <= -6 && x >= -8)){
        cmd = 'c';
    }
    /******************b+r******************/
    if((y <= -2 && y >= -5) && (x <= -3 && x >= -5)){
        cmd = 'v';
    }
    if((y <= -2 && y >= -5) && (x <= -5 && x >= -7)){
        cmd = 'b';
    }
    if((y <= -2 && y >= -5) && (x <= -7 && x >= -10)){
        cmd = 'n';
    }
return cmd;
}
```

Izvor: autor

Nadalje, nakon što se kodiraju vrijednosti u naredbu, šalju se prema robotskom vozilu u metodi *sendCmd()* koju opisuje programski kod u nastavku.

Kod 29 – sendCmd()

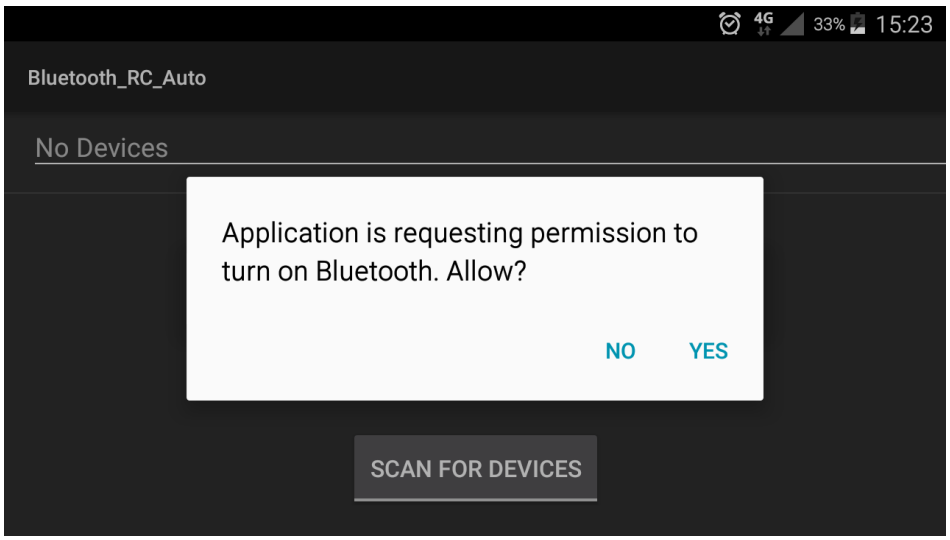
```
private void sendCmd(String cmd) {
    if(_btSocket != null) {
        try {
            _btSocket.getOutputStream().write(cmd.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Izvor: autor

5. UPORABA SUSTAVA

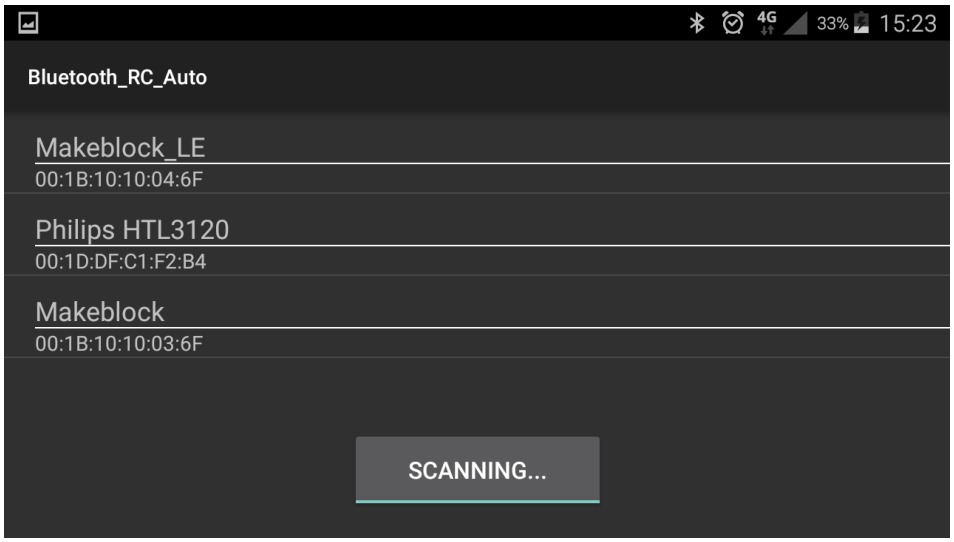
Sustav se koristi na način da korisnik prvo uključi *mBot,* a zatim pokrene aplikaciju na mobilnom uređaju. Najprije je korisnik upitan da li želi uključiti Bluetooth na uređaju. Ukoliko korisnik potvrdi uključivanje Bluetooth-a, prikazuje mu se sučelje za pretragu uređaja u dometu. Nakon što korisnik pokrene opciju pretrage uređaja u blizini, na zaslonu se pojavljuje naziv i MAC adresa svakog uređaja. Korisnik odabire uređaj pod nazivom "Makeblock". Navedeni scenariji su opisani slikama 9 i 10.

Slika 9 - Upit za uključivanje Bluetooth-a



Izvor: autor

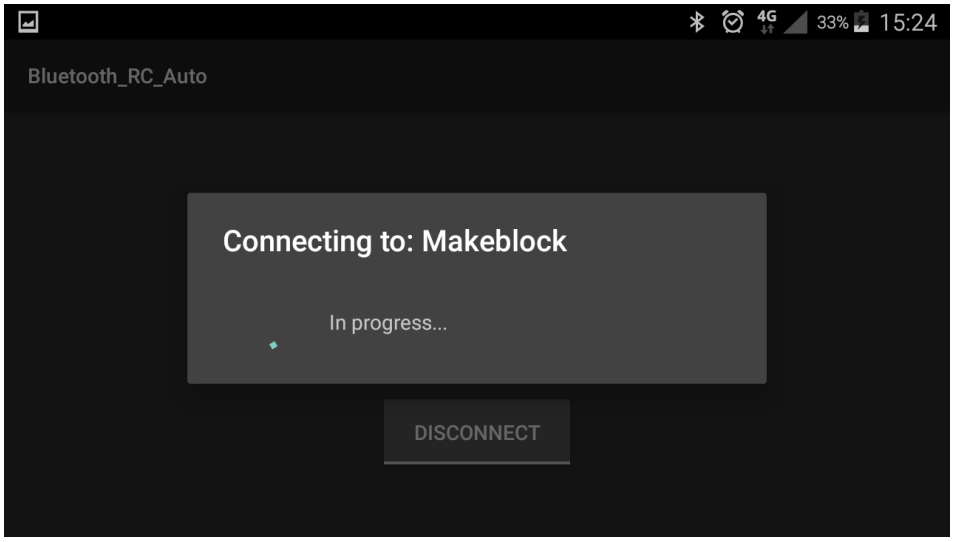
Slika 10 - Sučelje za pretraživanje uređaja



Izvor: autor

Odabirom na robotsko vozilo se pokreće funkcija spajanja. S obzirom da se program na mikrokontroleru pokreće odmah prilikom uključenja, uključuje se i Bluetooth modul kojem LED u tom stanju sporo treperi. Nadalje, ako uređaji međusobno ostvare vezu, na zaslonu se pojavljuje poruka "Connected" i korisnik može upravljati robotskim vozilom. Još jedan indikator uspješnog spajanja je LED na Bluetooth modulu koji u tom stanju permanentno svijetli.

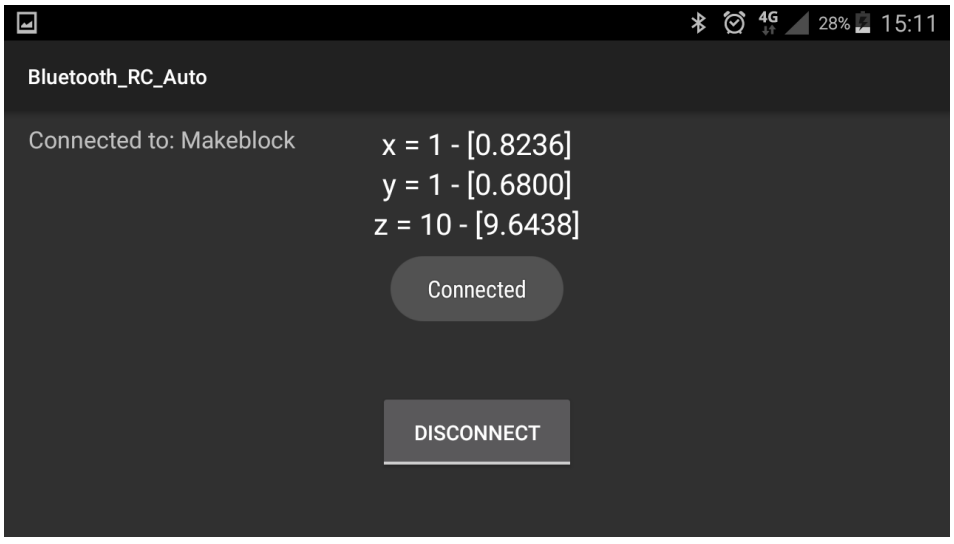
Slika 11 - Spajanje na robotsko vozilo



Izvor: autor

Na zaslonu se vrijednosti x,y i z osi izmjenjuju sukladno pomaku mobilnog uređaja te generiraju određene naredbe.

Slika 12 - Sučelje za upravljanje



Izvor: autor

6. ZAKLJUČAK

Na izradu rada je utrošeno 45 dana od kojih sedam dana na proučavanje dokumentacije vezane za razvojno okruženje Android Studio IDE i učenje principa rada Android platformi; sedam dana na razvoj i optimizaciju Android aplikacije i sedam dana na razvoj i testiranje Arduino programa i simulacije upravljanja. Ostatak vremena je utrošen na pisanje dokumentacije završnog rada.

Prilikom izrade završnog rada se naišlo na niz problema koji su uspješno riješeni. Priroda problema je obično bila vezana uz razvoj Android aplikacije te ispravljanje semantičkih pogreški. Ostali problemi su bili vezani uz razvoj Arduino programa. Rad se pokazao zanimljivim iz razloga što je bilo potrebno steći i softverska i hardverska znanja te ih zajedno primjeniti na način da sustav bude robustan, pouzdan i da je ostvaren postavljeni zadatak.

Sustav je moguće poboljšati primjenom ultrazvučnih senzora čija bi svrha bila detekcija objekata u okolini robota te u skladu s tim i izbjegavanje istih.

POPIS KRATICA I AKRONIMA

IDE – Integrated Development Environment

iOS – iPhone Operating System

IEEE – Institute of Electrical and Electronic Engineers

PAN - Personal Area Network

UML – Universal Modeling Language

LED – Light Emitting Diode

SCL – Serial Clock Line

SDL – Serial Data Line

GND – Ground

VCC – Collector supply voltage

TX - Transmitter

RX - Reciever

DTR – Data Terminal Ready

NC – Not Connected

RST – Resset

IR – Infrared

LITERATURA

1. Android Developers, Developer guides (25.7 2018.)

   https://developer.android.com/guide

2. Bluetooth SIG, Radio versions (10.8.2018.)

   https://www.bluetooth.com/bluetooth-technology/radio-versions

3. T. Corinne, Accelerometer Basics, (10.8. 2018.)

   https://learn.sparkfun.com/tutorials/accelerometer-basics

4. A. Jakupović, Veleučilište u Rijeci, Programiranje 2, Predavanja .ppt 4
5. Makeblock Co., Dual-mode Bluetooth module, pdf

POPIS SLIKA

POPIS TABLICA

POPIS KODOVA