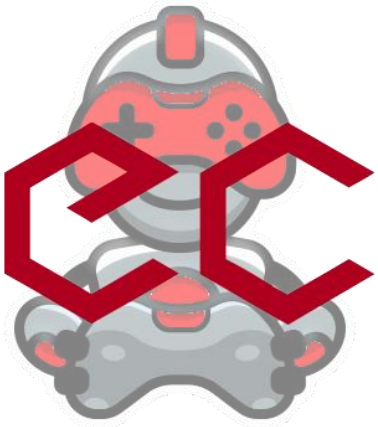
ŽUPANIJA BRODSKO-POSAVSKA

ELEKTROTEHNIČKA I EKONOMSKA ŠKOLA NOVA GRADIŠKA



Autori: Ivan Došlić, Ivan Trobić

Razred: 3.G

Godina: 2020./2021.

Mentor: Zoran Ivančić

Nova Gradiška, Prosinac 2020.

SADRŽAJ

1. UVOD

Kontroliranje robota upravljanih mikroupravljačima poput Arduina poprilično je zanimljivo, ali ponekad može biti naporno mijenjati kod samo kako bismo rekli robotu da se kreće malo drukčije nego prošli put.

„EMoRo Control“ aplikacija je koja rješava taj problem. „EMoRO Control“ nudi multimodularno upravljanje robotom putem Bluetooth komunikacije između android uređaja i bluetooth modula koji se nalazi na robotu.

Unutar aplikacije nalaze se 4 moda upravljanja robotom, a to su: Joystick, Žiroskop, Glasovno upravljanje i „D-Pad“ upravljanje.

Korištenje je vrlo jednostavno potrebno se samo povezati s robotom putem Bluetootha i unutar aplikacije izabrati vašeg robota kao robota kojim želite upravljati.

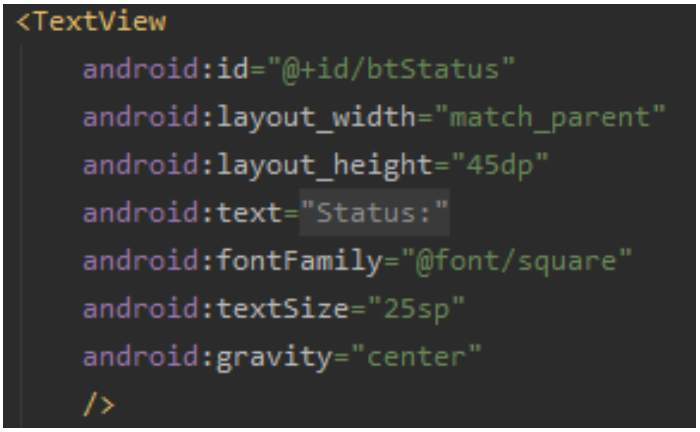
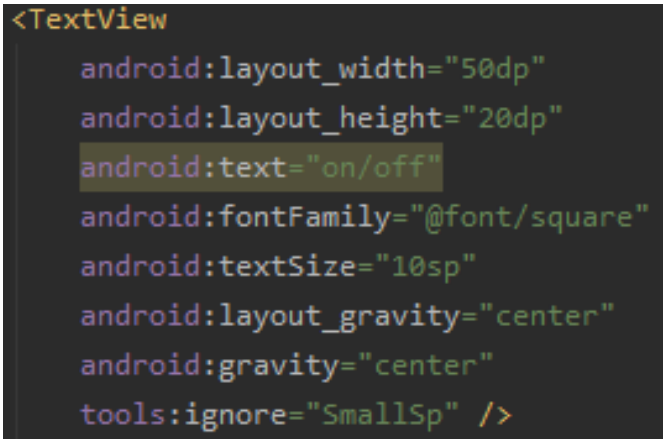
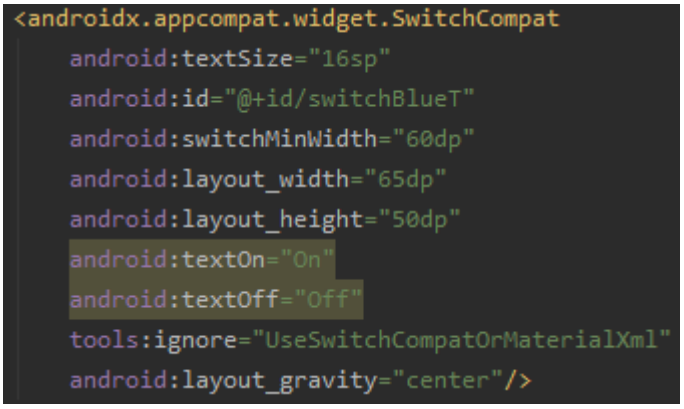
2. Connection Screen

2.1. Bluetooth Layout

Slika 2.1 – *„Izgled zaslona za osposobljavanje bluetootha“*

1. Dinamična slika Bluetootha koja indicira na to dali je bluetooth uključen ili isključen.
2. Tekst koji označava ima li dostupnih uređaja na koje se možemo povezati, neće moći reći ima li dostupnih uređaja ukoliko bluetooth nije uključen.
3. Interaktivna sklopka za uključivanje i uključivanje/isključivanje bluetootha unutar aplikacije, ali i na samom uređaju.
4. Ukoliko je bluetooth uključen i ima dostupnih uređaja ovaj gumb pokreće pretragu i dohvaća sve dostupne uređaje i otvara skočni prozor sa popisom uređaja na koje se možemo povezati

2.2. XML kod bluetooth zaslona

XML kod	Opis koda
	Prikaz bluetooth logotipa s dimenzijama 150x150, uz orijentaciju na sredinu ekrana i s ID oznakom „btIco“ putem koje pristupamo slici u Java kodu
	Prikaz statusnog teksta koji indicira na dostupnost uređaja za povezivanje uze dimenzije takve da ima visi od 45dp i uzima širinu zaslona ovisnu o uređaju i posjeduje ID „btStatus“ i orijentiran je na sredinu ekrana.
	Prikaz on/off teksta koji označava da sklopka ispod njega služi za uključivanje i isključivanje bluetootha.
	Prikaz sklopke za uključivanje i isključivanje bluetootha. Sklopka je iz androidx biblioteke koja dolazi u sklopu Android Studija. ID sklopke je „switchBlueT“, dimenzije su 65x50, orijentacija je na sred ekrana.

```
<ImageView
    android:id="@+id/btIco"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:src="@drawable/bluetooth"
    android:layout_gravity="center"/>
```

```
<TextView
    android:id="@+id/btStatus"
    android:layout_width="match_parent"
    android:layout_height="45dp"
    android:text="Status:"
    android:fontFamily="@font/square"
    android:textSize="25sp"
    android:gravity="center"
    />
```

```
<TextView
    android:layout_width="50dp"
    android:layout_height="20dp"
    android:text="on/off"
    android:fontFamily="@font/square"
    android:textSize="10sp"
    android:layout_gravity="center"
    android:gravity="center"
    tools:ignore="SmallSp" />
```

```
<androidx.appcompat.widget.SwitchCompat
    android:textSize="16sp"
    android:id="@+id/switchBlueT"
    android:switchMinWidth="60dp"
    android:layout_width="65dp"
    android:layout_height="50dp"
    android:textOn="On"
    android:textOff="Off"
    tools:ignore="UseSwitchCompatOrMaterialXml"
    android:layout_gravity="center"/>
```

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/discoverButt"
    android:layout_width="150dp"
    android:layout_height="50dp"
    android:layout_gravity="center"
    android:fontFamily="@font/square"
    android:text="CONNECT"
    android:layout_marginTop="30dp"
    />
```

Prikaz gumba za pretragu i ispis dostupnih uređaja za povezivanje. Dolazi u sklopu androidx biblioteke. ID mu je „discoverButt“ i dimenzije su mu 150x50.

2.3. Java kod bluetooth zaslona

2.3.1. Klasa Bluetooth

```
public class Bluetooth extends AppCompatActivity {

    private static final int REQUEST_ENABLE_BT = 0;
    private static final int REQUEST_DISCOVER_BT = 1;

    // Popup Window Lista
    TextView lista;
    // Bluetooth on/off switch
    SwitchCompat sBt;
    // Status tekst (Available/Unavailable)
    TextView btStat;
    // Bluetooth ikona (on/off)
    ImageView btIco;
    // Button za otkrivanje uređaja
    AppCompatButton discBtn;
    // Bluetooth adapter
    BluetoothAdapter BtAdapter;
```

Slika 2.2 – *„Deklaracija klase bluetooth i varijabli vezani uz nju“*

Klasa Bluetooth je javna klasa što znači da je dostupna svim ostalim klasama unutar aplikacije, ona služi kako bi odradila sav bitan dio za samo dohvaćanje adrese uređaja na koji se spajamo te za osposobljavanje Bluetooth komunikacije.

„private static final int REQUEST_ENABLE_BT i REQUEST_DISCOVER_BT“ su tzv. zastavice koje se koriste kako bismo mogli bluetooth adapteru lakše reći što treba odraditi tj. prvom zastavicom javljamo da treba uključiti/isključiti bluetooth ili s pomoću druge zastavice da treba započeti pretragu dostupnih uređaja.

Deklaracija varijabli pomoću kojih pravimo reference na XML elemente te varijable bluetooth adapter koja svojim imenom govori za što se koristi.

2.3.2. onCreate funkcija

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bluetooh);

    // Definiranje java varijabli za XML elemente
    btStat      = (TextView)findViewById(R.id.btStatus);
    btIco       = (ImageView)findViewById(R.id.btIco);
    discBtn     = (AppCompatButton)findViewById(R.id.discoverButt);
    sBt         = (SwitchCompat)findViewById(R.id.switchBlueT);
    //lista       = (TextView)findViewById(R.id.Lista);

    // Bluetooth adapter
    BtAdapter   = BluetoothAdapter.getDefaultAdapter();
```

Slika 2.3 – *„povezivanje varijabli sa XML-om i BluetoothAdapterom“*

onCreate funkcija je koja se pokreće kada aplikacija započne aktivnost Bluetooth tj. ovaj onCreate se pokreće na samom pokretanju aplikacije jer je ovo prva aktivnost nakon „Splash screena“.

Varijable definirane izvan ove funkcije sada povezujemo sa XML elementima koristeći gore navedene ID oznake svakog elementa tako je varijabla „btStat“ varijabla za referencu na tekst sa ID-om „btStatus“ itd.

Varijabla „BtAdapter“ služi za bluetooth adapter samog uređaja za to koristimo androidovu kalsu BluetoothAdapter i njezinu funkciju getDefaultAdapter() koja dohvati zadani lokalni Bluetooth adapter.

```
// Pri pokretanju aplikacije program provjerava bluetooth adapter
if (BtAdapter.isEnabled()){
    btIco.setImageResource(R.drawable.bluetooth);
    sBt.setChecked(true);
} else {
    btIco.setImageResource(R.drawable.bluetooth_no);
    sBt.setChecked(false);
}

// Provjerava status bluetooth-a
if(BtAdapter == null){
    btStat.setText("Unavailable");
} else {
    btStat.setText("Available");
}
```

Slika 2.4 – *„Uvjeti pri pokretanju aktivnosti Bluetooth“*

Pri pokretanju aplikacije automatski se provjeravaju dva uvjeta. Prvi uvjet provjerava dali je Bluetooth adapter aktiviran putem .isEnabled() funkcije koja izvrati true ukoliko je adapter aktivan što znači da će se izvesti blok naredbi poslije if a prije else, a ukoliko je adapter neaktivan izvratiti će se false i izvest će se blok naredbi poslije else. Prvi blok naredbi postavlja sliku bluetootha kada je uključen te sklopku postavlja u 1, drugi blok naredbi radi obratno.

Drugi uvjet koristi se status bluetootha pošto je moguće da je adapter aktivan, ali bluetooth je neaktivan jer nema dostupnih uređaja ukoliko nema uređaja postavlja se tekst „Unavailable“, a ako ima „Available“ to se provjerava uvjetom ukoliko je „adapter null“ tj. nema nikakve vrijednosti s kojom je povezan, u našem slučaju nema uređaja za povezivanje, izvratit će se true ako nema, a false ako ima princip je isti kao u prvom uvjetu.

```
// Bluetooth on/off switch
sBt.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked){
            if(!BtAdapter.isEnabled()){
                // Ako adapter nije aktiviran
                showToast( msg: "Turning on Bluetooth...");
                Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(intent, REQUEST_ENABLE_BT);
            } else {
                // Ako je adapter vec aktiviran **ovaj slucaj je gotovo nemoguc
                showToast( msg: "Bluetooth is already on...");
            }
        } else {
            // Iskljucuje adapter ako je aktiviran
            if(BtAdapter.isEnabled()){
                BtAdapter.disable();
                showToast( msg: "Turning Bluetooth off...");
                btIco.setImageResource(R.drawable.bluetooth_no);
            } else {
                // Iskljucuje adapter ako je aktiviran **ovaj slucaj je gotovo nemoguc
                showToast( msg: "Bluetooth is already off...");
            }
        }
    }
});
```

Slika 2.5 – *„Funkcija sklopke“*

Na sklopku za bluetooth postavljamo „slušatelja“ koji provjerava kada se stanje sklopke prebaci iz 0 u 1 i obratno. Kad se to dogodi pokreće se funkcija onCheckedChange() koja provodi više uvjeta. Prvi i glavni uvjet provjerava u kojem je stanju skolpka ukoliko je sklopka u stanju 1.

Provjerava se drugi uvjet koji je provjera dali je adapter neaktivan. To se ostvaruje korištenjem funkcije .isEnabled() i logičkog operatora ! (ne) koji invertira rezultat. Ukoliko je adapter neaktivan prikazuje se poruka da se uključuje Bluetooth te se pokreće aktivnost za pokretanje Bluetootha i tu koristimo zastavicu koju smo prethodno deklarirali. Ukoliko je adapter aktivan samo obavijestimo korisnika da je Bluetooth već aktivan. Drugi slučaj je gotovo nemoguć jer mi odrađujemo provjeru aktivnosti u onCreate().

Ukoliko je sklopka u stanju 0. Provjerava se dali je adapter aktivan i ukoliko je aktivan obavijestimo korisnika da se Bluetooth isključuje i pokreće se funkcija za onesposobljavanje Bluetootha. Ukoliko je Bluetooth već isključen o tome obavijestimo korisnika iako je iz istih razloga i to nemoguća situacija.

```
discBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!BtAdapter.isDiscovering()){
            // Posavlja uređaj da bude vidljiv i počinje israživanje
            showToast( msg: "Making your device discoverable!");
            Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
            startActivityForResult(intent, REQUEST_DISCOVER_BT);
        }
        if(BtAdapter.isEnabled()){
            // Otvara popup s popisom uređaja koji su dostupni
            startActivity(new Intent( packageContext: Bluetooh.this, popUp.class));
        } else {
            // Ako bluetooth nije uključen daje upozorenje da se prvo uključi bluetooth
            showToast( msg: "First turn on your Bluetooth!");
        }

    }
});
```

Slika 2.6 – *„Funkcija gumba“*

Posljednji dio onCreate() funkcije je postavljanje „slušatelja“ na gumb „connect“ koji na pritisak gumba pokreće funkciju onClick() unutar koje se dešavaju dvije provjere. Prvo provjerimo otkriva li adapter uređaje ukoliko ih ne otkriva prikazuje se obavijest da se uređaj otkriva tj. da ga drugi uređaji mogu prepoznati i da naš uređaj može prepoznati ostale. Za to koristimo drugu zastavicu koju smo deklarirali na početku i pokrećemo aktivnost za rezultat s pomoću nje.

Drugi uvjet provjerava aktivnost adaptera kako ne bi došli do greške prilikom pokušavanja spajanja s robotom. Ukoliko je adapter neaktivan obavijesti se korisnik da prvo mora uključiti Bluetooth. Ukoliko je adapter aktivan pokreće se nova aktivnost „Popup“ koja ispisuje listu dostupnih uređaja.

2.3.3. showToast() funkcija

```
private void showToast(String msg){ // ova funkcija samo olakšava stvaranje toast poruke
    Toast.makeText( context: this, msg, Toast.LENGTH_SHORT);
}
```

Slika 2.7 – *„showToast() funkcija“*

Ova funkcija koristi se kroz cijelu aplikaciju za lakši ispis poruka za obavještavanje tzv. „Toast Message“

2.3.4 onActivityResult() funkcija

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Bluetooth is ON
            btIco.setImageResource(R.drawable.bluetooth);
            showToast( msg: "Bluetooth is ON");
        } else {
            // Upozorava da nije uspjelo uključivanje bluetootha
            showToast( msg: "Couldn't turn bluetooth ON");
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

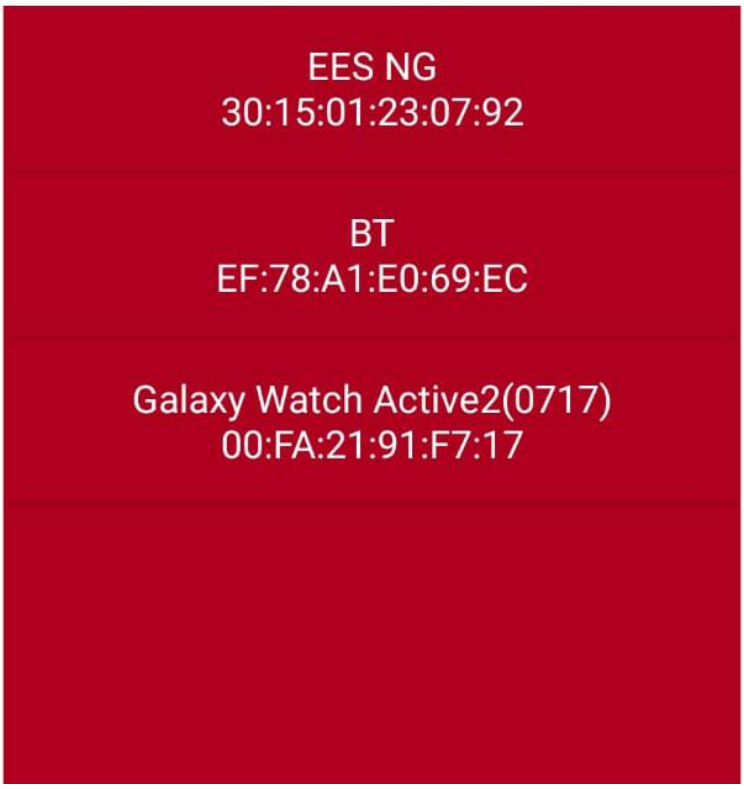
Slika 2.8 – *„onActivityResult() funkcija“*

Funkcija koju nazivamo i aktivnost za rezultat. Uz nju koristimo zastavice ovisno o zahtjevu koji smo napravili kao uključivanje Bluetootha i sl.

Unutar onActivityResult() kod Bluetooth klase imamo samo uvjet koji provjerava dali je zahtjev za uključivanje Bluetootha uspješan ukoliko je obavijesti se korisnik da je uspješno uključen Bluetooth ako nije obavijesti se da nije uspjelo. Ta provjera se odrađuje sa zastavicom RESULT_OK. Ukoliko je resultCode jednak toj zastavici znači da je uspjelo inače nije.

3. POPUP Screen

3.1. Popup Layout



Slika 3.1 – *„Izgled zaslona Popup“*

Zaslon Popup je vrlo jednostavan sastoji se samo od liste koju program stvara od svih uparenih uređaja putem Bluetootha.

3.2. XML kod Popup zaslona

```
<ListView
    android:id="@+id/Lista"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

ListView je element za prikazivanje liste širina i visina su mu jednake dimenzijama ekrana te mu je dodijeljen ID „Lista“

3.3. Java kod Popup zaslona

3.3.1. Klasa Popup

```
public class popUp extends Activity {
    ListView lista;
    BluetoothAdapter BtAdapter;
    TextView infor, adr;
    public static String address;
```

Slika 3.2 – *„Klasa popUp“*

Klasa Popup javna je klasa koja se koristi za skočni prozor koji nudi uređaje na koje se moguće spojiti. Unutar nje prvo deklariramo varijable kao što su lista koja služi za referencu liste u XML-u, BtAdapter koji je bluetooth adapter kao u Bluetooth klasi i javni string adress koji se koristi kroz cijelu aplikaciju jer se u njega spremi adresa uređaja na koji se spajamo tj. EMoRo robota.

3.3.2. onCreate() funkcija

On create funkcije objašnjena je u prvoj klasi tako da ćemo sada prikazati samo specifične dijelove za ovu klasu.

```
// Bluetooth dohvaća uređaje i dodaje ih na listu koju ispisujemo
ArrayList<String> list = new ArrayList<String>();
Set<BluetoothDevice> pairedDevices = BtAdapter.getBondedDevices();
for(BluetoothDevice bt: pairedDevices){
    list.add(bt.getName()+"\n"+bt.getAddress());
}
ArrayAdapter adapter = new ArrayAdapter( context: this, R.layout.row, list);
// Listu uređaja koju smo dohvatili povezujemo sa listom u XML-u
lista.setAdapter(adapter);
lista.setOnItemClickListener(customClickListener);
```

Slika 3.3 – *„Kreiranje liste uređaja“*

Stvaramo listu pod nazivom „list“ u koju ćemo upisati sve uređaje. U Set pod nazivom „pairedDevices“ spremamo uređaje koje je adapter prepoznao pomoću funkcije .getBondeddevices() zatim s pomoću petlje prolazimo kroz set i dodajemo ime i adresu uređaja. Zatim stvaramo ArrayAdapter tj. adapter za listu kako bismo tu listu postavili u XML i na kraju postavljamo adapter na listu u XML-u te na svaku stvar u listi postavljamo „slušatelja“ za pritisak te stvari.

```
// Prilagodba prozora da bude kao popup
DisplayMetrics dm = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(dm);

int width = dm.widthPixels;
int height = dm.heightPixels;

getWindow().setLayout((int) (width * .8), (int) (height * .9));
```

Slika 3.4 – *„Prilagodba prozora popup“*

Pošto je Popup skočni prozor želimo da se iza njega vidi stari prozor tj. Bluetooth zato koristimo ovaj kod gdje samo smanjujemo dimenzije prozora tj. širinu na 80% od originalne, a visinu na 90%.

3.3.3. customClickListener

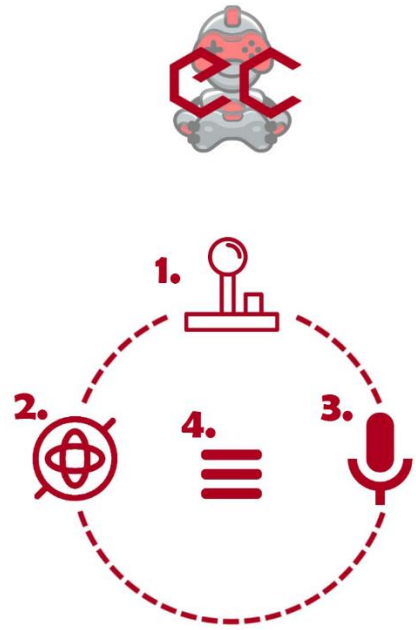
```
// Ovisno i pristisnutom uređaj daje nam njegovu adresu i postavlja je u public string te pokreće main activity
public AdapterView.OnItemClickListener customClickListener = new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Object item = parent.getItemAtPosition(position);
        String info = item.toString();
        address = info.substring(info.length()-17);
        Intent intent = new Intent( packageContext: popUp.this, MainActivity.class);
        intent.putExtra( name: "StrValue", info);
        startActivity(intent);
    }
};
```

Slika 3.5 – *„Funkcija customClickListenera“*

Pomoću ove funkcije na klik jednog od uređaja na listi uzimamo njegove informacije preko njegove pozicije u listi i spremamo ih u string info. Zatim iz tog stringa uzimamo samo adresu uređaja i spremamo to u javnu varijablu koju smo na početku deklarirali i nakon toga pokrećemo glavnu aktivnost i time zapravo ulazimo u glavni zaslon aplikacije.

4. Glavni izbornik (MainActivity)

4.1. MainActivity layout



Slika 4.1 – *„Glavni izbornik“*

1. Brzi pristup Joystick modu upravljanja.
2. Brzi pristup Žiroskop modu upravljanja.
3. Brzi pristup Glasovnom modu upravljanja.
4. Otvaranje velikog izbornika.

4.2. XML kod glavnog izbornika

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bgmainmenu"
    tools:context=".MainActivity">
```

Glavni izbornik napravljen je unutar jednog FrameLayouta i sastoji se od 2 LinearLayouta i jednog ConstraintLayouta

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="top"
    android:layout_marginBottom="0dp">

    <ImageView
        android:layout_width="0dp"
        android:layout_height="150dp"
        android:src="@drawable/beta1"
        android:layout_weight="1"
        android:gravity="center"
        tools:layout_editor_absoluteX="130dp"
        tools:layout_editor_absoluteY="49dp" />

</LinearLayout>
```

Unutar prvog LinearLayouta nalazi se ikona aplikacije EMoRo Control

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="429dp"
    android:layout_gravity="center">

    <ImageView
        android:id="@+id/imageView14"
        android:layout_width="300dp"
        android:layout_height="301dp"
        android:src="@drawable/littlecircle"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.495"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.695" />

    <ImageView
        android:layout_width="75dp"
        android:layout_height="75dp"
        android:src="@drawable/icomicred"
        app:layout_constraintBottom_toBottomOf="@+id/imageView13"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.685"
        app:layout_constraintStart_toEndOf="@+id/imageView13"
        app:layout_constraintTop_toTopOf="@+id/imageView13"
        app:layout_constraintVertical_bias="1.0"
        android:onClick="openVoice"/>
```

Unutar ConstraintLayouta nalazi se 5 slika od kojih je jedna pozadinska slika, a ostale 4 su ikone modula rada i velikog izbornika koje su ujedno i gumbovi za pokretanje modula s pomoću onClick:funkcija

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:layout_gravity="bottom">

    <ImageView
        android:layout_width="0dp"
        android:layout_height="25dp"
        android:layout_gravity="bottom"
        android:layout_weight="1"
        android:src="@drawable/juanonjuantxt"
        tools:layout_editor_absoluteX="105dp"
        tools:layout_editor_absoluteY="668dp" />
</LinearLayout>
```

Unutar drugog LinearLayouta nalazi se slika koja sadrži tekst s imenom kreatora aplikacije.

4.3. Java kod glavnog izbornika

4.3.1. Klasa MainActivity

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

Slika 4.2 – *„MainActivity klasa“*

MainActivity javna je klasa koja sadrži samo funkcije za otvaranje velikog izbornika i brzi pristup nekim od modula rada.

4.3.2. Open funkcije

Sve funkcije unutar glavnog izbornika su jako slične jer im je svrha samo da pokrenu aktivnost koju korisnik želi tj. da pokrenu modul upravljanja ili veliki izbornik.

```
public void openMenu(View view){ // Otvaranje glavnog menu-a sa svim ostalim podacima
    Intent intent = new Intent( packageContext: this, Menu.class);
    startActivity(intent);
    overridePendingTransition(R.anim.bottom_up, R.anim.bottom_down);
}

public void OpenJoy(View view){...}

public void OpenGyro(View view){...}

public void openVoice(View view){...}

public void OpenNormal(View view){...}
```

Slika 4.3 – *„Open funkcije“*

Unutar svih funkcija nalazi se isti primjer koda samo što se mijenja ciljana aktivnost koju želimo pokrenuti i uz to dodaje se overridePendingTransition() funkcija koja postavlja animaciju za tranziciju iz funkcije u funkciju za bolju estetiku aplikacije.

5. Veliki Izbornik

5.1. Menu layout

Slika 5.1 – *„Veliki izbornik“*

1. Povratak u zadnju korištenu aktivnost
2. Otvaranje glavnog izbornika
3. Otvaranje Joystick modula rada
4. Otvaranje Žiroskop modula rada
5. Otvaranje Glasovnog modula rada
6. Otvaranje „D-Pad“ modula rada
7. Otvaranje prozora za informacije o autorima
8. Otvaranje prozora za informacije o aplikaciji

5.2. XML kod Velikog Izbornika

XML kod velikog izbornika sastoji se od FrameLayouta unutar kojeg se za svaku od modula rada nalazi po jedan LinearLayout koji sadrži ikonu i naziv modula te u sebi sadrži onClick:funkcija gdje je funkcija jedna od varijanti open funkcija navedenih u MainActivity-u.

5.3. Java kod Velikog Izbornika

5.3.1. Klasa Menu i open funkcije

```
public class Menu extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);
    }

    // Cjeli/Veliki Menu

    public void homeSc(View view){ // Povratak na početni zaslon
        Intent intent = new Intent( packageContext: this, MainActivity.class);
        startActivity(intent);
        overridePendingTransition( enterAnim: 0, R.anim.bottom_down);
    }

    public void OpenJoy(View view){...}

    public void OpenGyro(View view){...}

    public void openVoice(View view){...}

    public void OpenNormal(View view) {...}
```

Slika 5.2 – *„Klasa i funkcije“*

Klasa Menu javna je klasa koja u sebi sadrži funkcije za pokretanje svih aktivnosti koje aplikacija nudi s pomoću funkcija navedenih kod glavnog izbornika.

5.3.2. Funkcija finish()

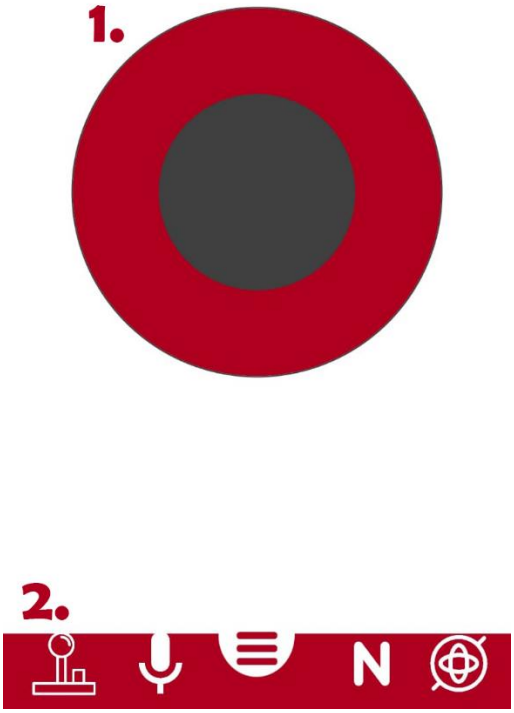
```
@Override
public void finish() {
    super.finish();
    overridePendingTransition(R.anim.bottom_up, R.anim.bottom_down);
}
```

Slika 5.3 – *„Funkcija finish()“*

Ova funkcija izvršava se prilikom završavanja aktivnosti Menu i unutar nje je postavljena animacija zalaska i izlaska na isti način kao kod open funkcija.

6. Joystick Modul

6.1. Joystick layout



Slika 6.1 – *„Joystick“*

1. Joystick upravljač kojim se usmjerava EMoRo
2. Brzi izbornik za otvaranje velikog izbornika i brzi pristup ostalim modulima rada

6.2. XML kod Joysticka

Joystick se sastoji od FrameLayouta unutar kojeg imamo jedan LinearLayout i jedan ConstraintLayout.

```
<io.github.controlwear.virtual.joystick.android.JoystickView
    android:id="@+id/joystick"
    android:layout_width="350dp"
    android:layout_height="350dp"
    app:JV_backgroundColor="@color/colorPrimaryDark"
    app:JV_borderColor="@color/Grey"
    app:JV_borderWidth="1dp"
    app:JV_buttonColor="@color/Grey"
    app:JV_buttonSizeRatio="40%"
    app:JV_fixedCenter="false"
    android:background="#FFFFFF"/>
```

Za joystick upravljač koristi se „opensource“ joystick kojemu je dodijeljen ID za pristup u Java kodu te sve ostale vizualne stavke.

Joystick je moguće pronaći na GitHub-u i koristiti ga u svojoj aplikaciji. [1]

ConstraintLayout sadrži ikone za brzi pristup i koristi open funkcije kao i ostale aktivnosti u aplikaciji.

6.3. Java kod Joysticka

6.3.1. Joystick klasa

```
public class Joystick extends AppCompatActivity {
    // Stvari za komunikaciju sa eMoro
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private ProgressDialog progress;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        new ConnectBT().execute();
        setContentView(R.layout.activity_joystick);
```

Slika 6.2 – *„Joystick klasa“*

Unutar Joystick klase ponovno se pojavljuje bluetooth adapter koji će se pojavljivati i u svim ostalim aktivnostima jer je potreban za komunikaciju sa EMoRo Robotom, ali uz njega se nalaze još neke ključne stvari za Bluetooth komunikaciju kao što su BluetoothSocket koji se koristi kako bismo slali signale Bluetooth modulu na robotu, boolean koji nam govori dali je ostvaren konekcija sa robotom i UUID za identifikaciju.

6.3.2. Joystick upravljanje

```
// Stvaranje joysticka i povezivanje s joystickom unutar XML-a
JoystickView joystick = (JoystickView) findViewById(R.id.joystick);
joystick.setOnMoveListener(new JoystickView.OnMoveListener() {
    @Override
    public void onMove(int angle, int strength) {
        /*
            Provjerava kut tj. smjer i snagu joystica tj. koliko smo daleko povukli
            i na osnovu toga šalje signal za pokretanje robota u određenom smjeru
        */
        if(strength<70) {if (btSocket!=null)
        {
            try
            {
                btSocket.getOutputStream().write("S".toString().getBytes());
            }
            catch (IOException e)
            {
                msg( s: "Error");
            }
        }}
        else {
            if ((angle < 30 && angle >= 0) || (angle < 360 && angle >= 330))
            {if (btSocket!=null)
            {
                try
                {
                    btSocket.getOutputStream().write("R".toString().getBytes());
                }
                catch (IOException e)
                {
                    msg( s: "Error");
                }
            }}
```

Slika 6.3 – *„Kod za čitanje pokreta Joysticka“*

Prvo povežemo Joystick varijablu sa joystick elementom u XML-u i na nju postavimo „slušatelja“ pokreta koji nam govori pod kojim kutom se kreće joystick i koliko daleko ga je korisnik pomaknuo i ovisno o tome program pokuša robotu poslati signal u kojem se smjeru kretati, ukoliko ne uspije ispiše se da je došlo do greške. Za slanje signala koristi se BluetoothSocket i funkcija .getOutputStream().write()

6.3.3. Općenito slanje signala EMoRo

EMoRo u sebi sadrži program koji na određene znakove izvršava željene radnje mi te znakove šaljemo bluetooth modulu na robotu sa našeg uređaja putem BluetoothSocketa i output streama.

Osnovni signali za upravljanje:

Znak	Funkcija
S	Zaustavljanje
F	Naprijed
B	Nazad
R	Desno
L	Lijevo
I	Gore desno
G	Gore lijevo
J	Dolje desno
H	Dolje lijevo

Tablica 6.1 – *„Znakovi za upravljanje EMoRo-m“*

Prilikom slanja tih znakova mi te znakove pretvaramo u bajtove kako bi ih se moglo poslati i kako bi ih bluetooth modul mogao iščitati to odrađujemo sa .getBytes() npr. „F“.getBytes();

6.3.4. Privatna klasa ConnectBT

Ovu klasu koristimo u svim modulima rada jer je ona zapravo ta koja ostvaruje komunikaciju sa EMoRo-m, a u početku klasa Bluetooth nam samo pomogne dohvatiti adresu uređaja na koji se spajamo.

```
private class ConnectBT extends AsyncTask<Void, Void, Void>
{
    private boolean ConnectSuccess = true;
    @Override
    protected void onPreExecute()
    {
        progress = ProgressDialog.show( context: Joystick.this, title: "Connecting...", message: "Please wait!!!");
    }
```

Slika 6.4 – *„Privatna klasa ConnectBT“*

Unutar ConnectBT klase nalazi se boolean ConnectSuccess koji označava da je spajanje uspješno u početku ga postavljamo u true, a onda ukoliko dođe do problema postavlja se u false. Također započinjemo funkciju onPreExecute() koja se odvija prije samog izvođenja spajanja.

```
@Override
protected Void doInBackground(Void... devices)
{
    try
    {
        if (btSocket == null || !isBtConnected)
        {
            myBluetooth = BluetoothAdapter.getDefaultAdapter();
            BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(popUp.address);
            btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);
            BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
            btSocket.connect();
        }
    }
    catch (IOException e)
    {
        ConnectSuccess = false;
    }
    return null;
}
```

Slika 6.5 – *„Funkcija spajanja sa robotom“*

U pozadini se pokreće funkcija za spajanje s EMoRo-m gdje se pokuša spojiti sa robotom ponovno se dohvaća adapter i pokuša s prepoznati robot ukoliko se uspije odradi se .connect() putem BluetoothSocketa, ako ne uspije ConnectSuccess se postavi u false.

```
@Override
protected void onPostExecute(Void result)
{
    super.onPostExecute(result);
    if (!ConnectSuccess)
    {
        msg( s: "Connection Failed. Is it a SPP Bluetooth? Try again.");
        finish();
    }
    else
    {
        msg( s: "Connected.");
        isBtConnected = true;
    }
    progress.dismiss();
}
```

Slika 6.6 – *„onPostExecute“*

Ukoliko konekcija nije ostvarena ispisuje se poruka da povezivanje nije uspjelo i da treba pokušati ponovo. To se radi s pomoću funkcije msg koja je identična kao već navedena funkcija showToast(). I pokreće se funkcija finish() koja nas vrati na početni zaslon. Ukoliko konekcija uspije ispiše se obavijest da je konekcija uspješna i boolean isBtConnected postavlja se u true. Na kraju se u oba slučaja prekida progress.

7. Žiroskop

7.1. Gyroscope layout

Slika 7.1 – *„Gyroscope layout“*

Gyroscope layout jako je sličan Joystick layoutu samo što umjesto joysticka sadrži sliku na sredini koja se mijenja ovisno u koji smjer usmjerimo ili pokazuje P(park) ukoliko držimo uređaj u mjestu.

7.2. XML kod Žiroskopa

Žiroskop se sastoji od FrameLayouta kao i Joystick(od jednog LinearLayouta i jednog ConstraintLayouta) u LinearLayout postavljena je slika s ID-om „gyro_pointer“ koja se mijenja putem Java koda. A u ConstraintLayoutu je traka za brzi pristup modulima i otvaranje velikog izbornika.

7.3. Java kod Žiroskopa

7.3.1. Klasa Gyroscope

```
public class Gyroscope extends AppCompatActivity {
    private Gyro gyroscope;
    // Stvari za komunikaciju sa eMoro
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket mySocket = null;
    private boolean isBluetoothConnected = false;
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private ProgressDialog progress;

    // stvari za rad Žiroskopa
    public ImageView gyropointer;
```

Slika 7.2 – *„Gyroscope klasa“*

Na početku klase deklariran je objekt gyroscope iz klase Gyro koju smo napravili za objekte tipa žiroskopa, zatim su deklarirane sve varijable potrebne za bluetooth komunikaciju kao i kod Joysticka i na kraju je deklarirana varijabla za referencu na sliku u XML koja pokazuje smjer.

7.3.2. Klasa Gyro

```
public class Gyro {
    public interface Listener{
        void onRotation(float rx, float ry, float rz);
    }

    private Listener listener;

    public void setListener(Listener l){
        listener = l;
    }

    private SensorManager sensorManager;
    private Sensor sensor;
    private SensorEventListener sensorEventListener;
```

Slika 7.3 – *„Gyro klasa“*

Gyro klasa je javna klasa koju smo kreirali kako bismo olakšali rad sa žiroskopom, ali i kako bismo poboljšali estetiku koda. Unutar nje nalaze se SensorManager, Sensor i SensorEventListener koji se koriste u radu sa senzorima unutar uređaja. Stvoren je i interface Listener koji se koristi za dobivanje rotacije po x, y i z osima.

```
Gyro(Context context){

    sensorManager = (SensorManager) context.getSystemService(Context.SENSOR_SERVICE);
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
    sensorEventListener = new SensorEventListener(){
        @Override
        public void onSensorChanged(SensorEvent sensorEvent){
            if(listener != null){
                listener.onRotation(sensorEvent.values[0], sensorEvent.values[1], sensorEvent.values[2]);
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int i){

        }
    };
}
```

Slika 7.4 – *„Gyro Context“*

Ovdje je stvoren kontekst Gyro klase u kojemu se definiraju sensorManager dohvaćanjem iz sistemskih servisa te senzor koji se dohvaća putem sensorManagera iz servisa za senzore.

Zatim imamo dvije funkcije nama je bitna samo onSensorChanged() koja detektira pokrete senzora i dohvaća rotacije po x, y i z osi koje zatim sprema u listener.onRotation što smo deklarirali iznad.

```
public void register(){
    sensorManager.registerListener(sensorEventListener, sensor, SensorManager.SENSOR_DELAY_NORMAL);
}

public void unregister(){
    sensorManager.unregisterListener(sensorEventListener);
}
```

Slika 7.5 – *„Register i Unregister funkcije“*

Posljednje dvije funkcije koriste se za registriranje i „odjavu“ listenera kako bi se mogao postaviti delay a ne konstantno čitati vrijednost sa senzora.

7.3.3. Funkcioniranje žiroskopa

```
gyroscope.setListener(new Gyro.Listener() {
    @Override
    public void onRotation(float rx, float ry, float rz) {
        if(rx > 1f){
            if(ry > 1f){
                try
                {
                    mySocket.getOutputStream().write("H".toString().getBytes());
                    gyropointer.setImageResource(R.drawable.arrow_southwest);
                }
                catch (IOException e)
                {
                    msg( s: "Error");
                }
            } else if (ry < -1f){
```

Slika 7.6 – *„Kontroliranje EMoRo preko žiroskopa“*

Na gyroscope postavljamo „slušatelja“ kojeg smo definirali u klasi Gyro i iščitavamo rotaciju na x i y osi kako bismo znali slati signale robotu i postavljati slike unutar aktivnosti. 1f i -1f su pragovi koje treba preći kako bi se robot pomaknuo. Prag se može mijenjati ovisno o tome koliko želimo zakretati uređaj da se ostvari pomak robota. Ukoliko se ne uspije poslati signal kao i kod Joysticka ispisuje se poruka greške.

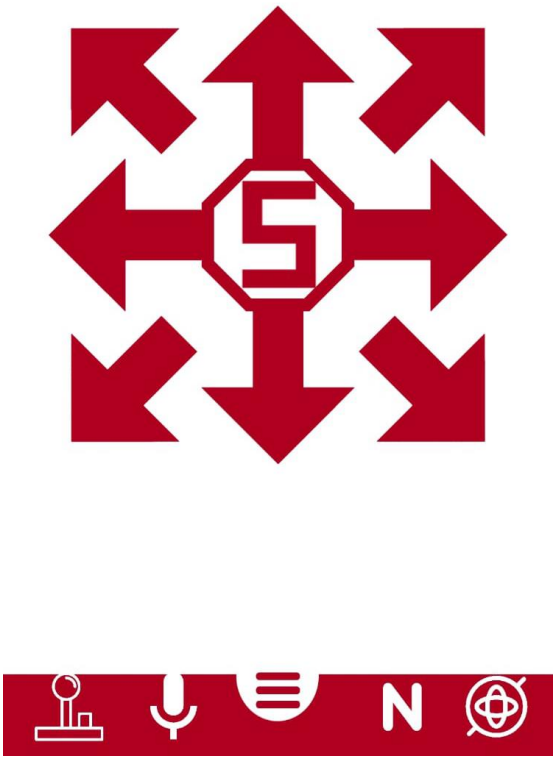
Za komunikaciju s robotom i ovdje je korištena klasa ConnectBT.

Za poruke se koristi showToast() tj. msg().

Za navigaciju koriste se open funkcije.

8. „D-Pad“ modul

8.1. „D-Pad“/Normal layout



Slika 8.1 – *„D-Pad layout“*

„D-Pad“ layout napravljen je od FrameLayouta unutar kojeg se nalaze dva ConstraintLayouta jedan za „D-Pad“ i jedan za traku za brzi pristup.

8.2. XML kod „D-Pada“

```
<ImageButton
    android:id="@+id/ul"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:background="@drawable/ul"
    app:layout_constraintBottom_toTopOf="@+id/l"
    app:layout_constraintRight_toLeftOf="@+id/u" />

<ImageButton
    android:id="@+id/u"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:background="@drawable/u"
    app:layout_constraintBottom_toTopOf="@+id/stop"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

<ImageButton
    android:id="@+id/ur"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:background="@drawable/ur"
    app:layout_constraintBottom_toTopOf="@+id/r"
    app:layout_constraintLeft_toRightOf="@+id/u" />
```

Slika 8.2 – *„Gumbovi za kretanje“*

„D-Pad“ se sastoji od 9 gumbova tj. slika pomoću kojih se upravlja EMoRo-m svakom gumbu/slici dodijeljen je ID preko kojeg mu pristupamo u Java kodu i postavljamo mu funkciju na klik.

8.3 Java kod „D-Pada“

8.3.1. Klasa Normal

```
public class Normall extends AppCompatActivity {
    // Stvari za komunikaciju sa eMoro
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private ProgressDialog progress;

    public ImageButton ul;
    public ImageButton u;
    public ImageButton ur;
    public ImageButton l;
    public ImageButton stop;
    public ImageButton r;
    public ImageButton bl;
    public ImageButton b;
    public ImageButton br;
    public TextView text;
```

Slika 8.3 – *„Klasa Normal“*

Klasa Normal javna je klasa koja u sebi sadrži varijable za bluetooth konekciju i klasu ConnectBT, uz to sadrži i varijable koje referiraju na gumbove u XML-u.

8.3.2. Funkcioniranje „D-Pada“

```
ul.setOnClickListener((v) → {
        try
        {
            btSocket.getOutputStream().write("G".toString().getBytes());
        }
        catch (IOException e)
        {
            msg( s: "Error");
        }
});
u.setOnClickListener((v) → {
        try
        {
            btSocket.getOutputStream().write("F".toString().getBytes());
        }
        catch (IOException e)
        {
            msg( s: "Error");
        }
});
```

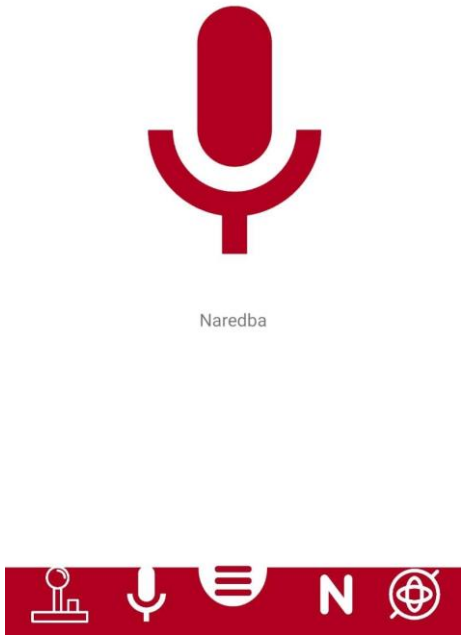
Slika 8.4 – *„Princip funkcioniranja D-Pada“*

Na svaki od gumbova postavljen je „slušatelj“ koji prepoznaje klik na gumb i šalje signal za usmjeravanje robota koji želimo. Princip je isti kao i u ostalim modulima.

Od ostalih funkcija opet se koriste open funkcije i showToast tj. msg.

9. Glasovno upravljanje

9.1. Voice layout



Slika 9.1 – *„Voice layout“*

Voice layout je kao i Joystick i Žiroskop FrameLayout koji se sastoji od jednog LinearLayouta za sliku/gumb mikrofona i jednog ConstarintLayouta za traku za brzi pristup.

9.2. XML kod Glasovnog Upravljanja

Unutar LinearLayouta nalazi se slika mikrofona koja služi kao gumb za uključivanje mikrofona i ispod slike nalazi se tekst kojemu je zadana vrijednost „Naredba“, a mijenja se ovisno o tome što korisnik izgovori. Unutar ConstraintLayouta je samo traka za brzi pristup i veliki menu.

9.3. Java kod Glasovnog Upravljanja

9.3.1. Klasa Voice

```
public class Voice extends AppCompatActivity {
    // Stvari za komunikaciju sa eMoro
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private ProgressDialog progress;

    // Ispis naredbe koju smo izgovorili
    public TextView naredba;
```

Slika 9.2 – *„Klasa Voice“*

Klasa Voice javna je klasa koja u sebi sadrži također klasu ConnectBT i funkciju za prepoznavanje riječi te funkciju tj. aktivnost za rezultat koja putem Googleovog API-a pretvara zvuk u tekst. [2]

9.3.2. Funkcija getSpeech()

```
public void getSpeech(View view){
    // Funkcija koja dohvaća govor
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());

    if(intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, requestCode: 10);
    } else {
        Toast.makeText( context: this, text: "Vaš uređaj Ne Podržava Glasovno Upravljanje", Toast.LENGTH_SHORT).show();
        // Treba vratiti korisnika na home screen zato što ne može koristiti ovu funkciju
    }
}
```

Slika 9.3 – *„getSpeech() funkcija“*

Prilikom pritiska na mikrofon pokreće se funkcija getSpeech() koja počinje aktivnost za rezultat, ukoliko korisnik nije dopustio korištenje mikrofona ili uređaj ne podržava korištenje mikrofona unutar aplikacije ispisat će se obavijest da uređaj ne podržava glasovno upravljanje. Za pristup prepoznavanju govor koristi se zastavica na istom principu kao i kod Bluetooth klase.

9.3.3. onActivityResult() funkcija

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // Dohvaća riječi koje smo izgovorili
    if (requestCode == 10) {
        if (resultCode == RESULT_OK && data != null) {
            ArrayList<String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            recognizeAction(result.get(0));
        }
    }
}
```

Slika 9.4 – *„onActivityResult() funkcija“*

Funkcionira na isti princip kao i aktivnost u Bluetooth klasi izvraća nam rezultat tj. riječi koje su prepoznate iz našeg govora mi zatim taj rezultat šaljemo u recognizeAction() funkciju za prepoznavanje riječi i slanje signala robotu ovisno o riječi.

9.3.4. recognizeAction() funkcija

```
private void recognizeAction(String s) {
    s = s.toLowerCase();
    // prepoznaje riječi koje smo izgovorili i šalje signal za pokretanje u željenom smjeru
    switch (s) {
        case "go forward":
        case "move forward":
        case "go":
        case "forward":
        case "go ahead":
            try{
                btSocket.getOutputStream().write("F".toString().getBytes());
            } catch (IOException e) {
                msg( s: "Error");
            }
            break;
```

Slika 9.5 – *„recognizeAction() funkcija“*

Funkcija uzima prepoznate riječi i prvo sva slova postavi u mala kako bi program mogao dobro usporediti dva stringa zatim ukoliko se rezultat poklapa s jednom od opcija unutar recognizeAction() funkcije šalje se signal robotu.

9.3.5. Popis glasovnih naredbi

Glasovno upravljanje podržano je na engleskom jeziku, a popis naredbi je:

```
"go forward":
"move forward":
"go":
"forward":
"go ahead":
```

Naprijed

```
"go back":
"come back":
"move back":
"go backward":
"backward":
"backwards":
"go backwards":
"move backwards":
"move backward":
```

Nazad

```
"to the right":
"to right":
"right":
"turn right":
"move right":
"go right":
```

Desno

```
"to the left":
"to left":
"left":
"turn left":
"move left":
"go left":
```

Lijevo

```
"right up":
"up right":
"right and up":
"up and right":
"right forward":
"forward right":
"right and forward":
"forward and right":
"up then right":
"upright":
```

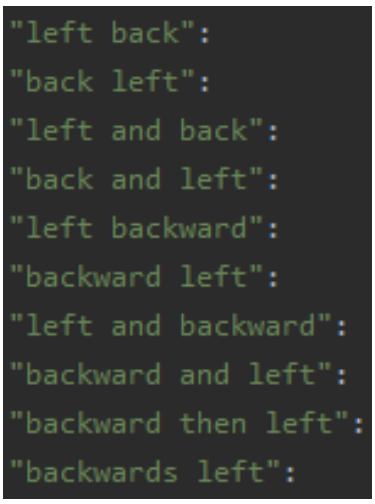
Naprijed Desno

```
"left up":
"up left":
"left and up":
"up and left":
"left forward":
"forward left":
"left and forward":
"forward and left":
"up then left":
"upleft":
```

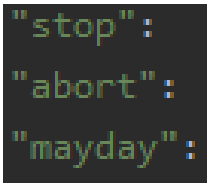
Naprijed Lijevo

```
"right back":
"back right":
"right and back":
"back and right":
"right backward":
"backward right":
"right and backward":
"backward and right":
"backward then right":
"backwards right":
```

Nazad Desno



Nazad Lijevo



Zaustavljanje

10. O autorima i O aplikaciji

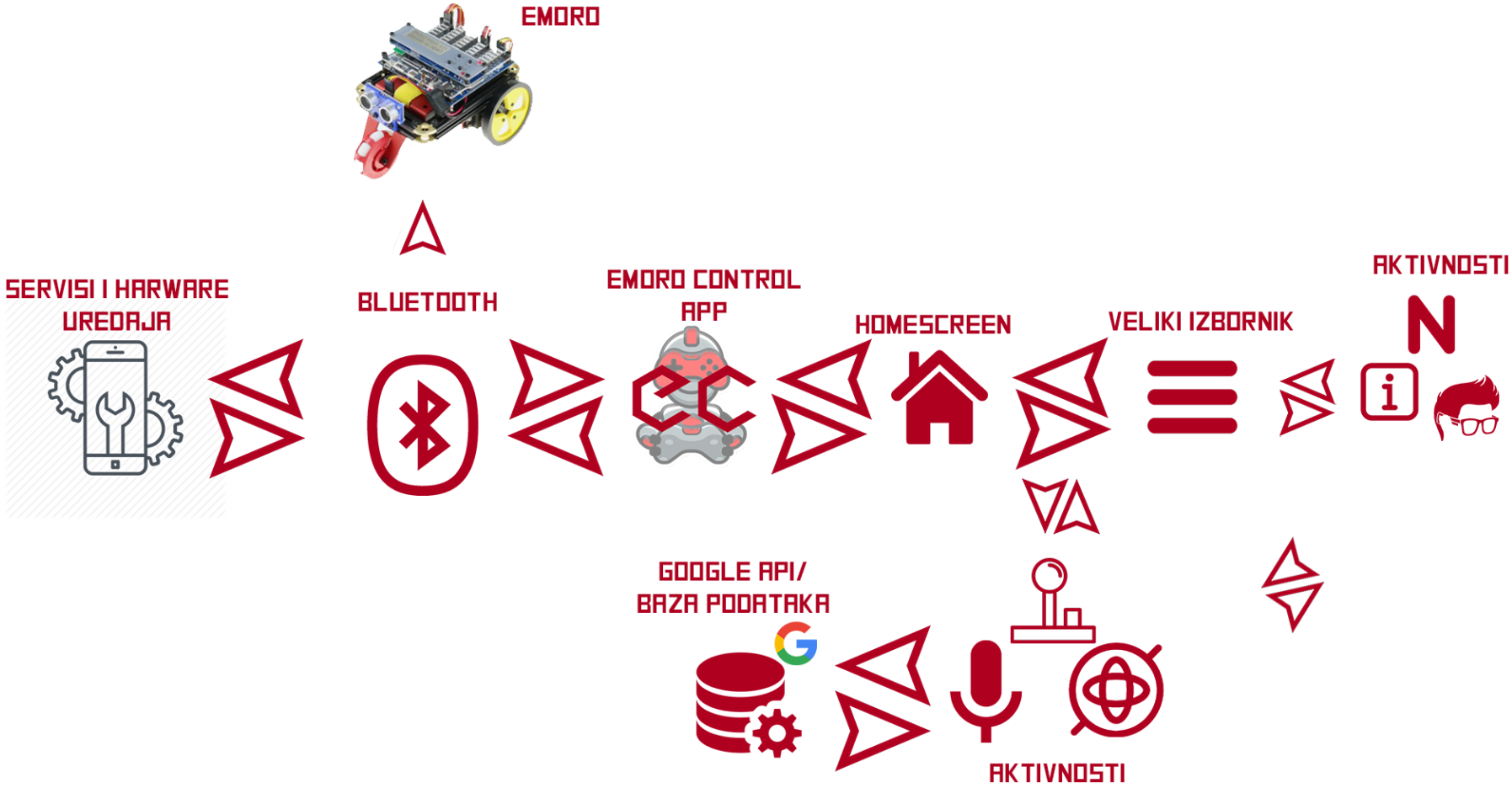
10.1. About Authors/App layout

Ova dva layouta strukturom su identični tj. sastoje se od FrameLayouta unutar kojih se nalaze po jedan LinearLayout u kojemu su slika i tekst te od ConstraintLayouta unutar kojeg je traka za brzi pristup kao i u ostalim aktivnostima.

10.1.1. Java kod aktivnosti

Java kod ovih aktivnosti sadrži samo open funkcije na traci za brzi pristup.

11. Integracijska Shema



11. Literatura

[1] – GitHub Joystick: https://github.com/controlwear/virtual-joystick-android

[2] – Google Speech Recognizer API:
https://developer.android.com/reference/android/speech/SpeechRecognizer