HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING

BBM 460 FINAL PROJECT REPORT

Arduino & Bluetooth Based Home Automation

Group Name : Make Wifi Great Again

Mehmet Taha USTA – 21527472

1) Introduction

Using wireless network technology to facilitate daily life. You will see how to automate various gadgets of your home and room. You will be able to control the lights, fan, air conditioning, curtains, locks and TV.

2) Technologies to be Used

Hardware

Arduino UNO
Arduino Bluetooth Modül HC05
Arduino 4 Way 5V Relay Module
Breadboard
Connecting wires
Bluetooth enabled smartphone

Software

Arduino 1.8.13 compiler
Android Studio 4.1
Android Application

2.1) Arduino UNO

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.[1] The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.

Advantages

It has an Atmel brand microcontroller on it and it is ready to use with the necessary connections made.

It can be directly connected to a computer via USB.

Easily adaptable to kits -> so-called plug-in circuits are widely available in the market.

It makes it simpler to build circuits together with the peripheral modules (Shield).

It does not need the programmer required for microcontrollers.

Its programming is more understandable and easier than other development kits.

With its wide library support, it has simplified even very complex projects as much as possible.

Its cost is more affordable than similar systems.

It is an easily available product in the market.

2.2) Arduino Bluetooth Modüle HC05

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module,designed for transparent wireless serial connection setup.The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication.This serial port bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Rluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

2.3) Arduino 4 Way 5V Relay Module

The 4 Channel Relay Module is a convenient board which can be used to control high voltage, high current load such as motor, solenoid valves, lamps and AC load. It is designed to interface with microcontroller such as Arduino, PIC and etc. The relays terminal (COM, NO and NC) is being brought out with screw terminal. It also comes with a LED to indicate the status of relay.

2.4) Breadboard

A breadboard is a construction base for prototyping of electronics.

2.5) Connecting Wires

Connecting wires allows an electrical current to travel from one point on a circuit to another because electricity needs a medium through which it can move.

2.6) Bluetooth Enabled Smartphone

Source that can send bluetooth signal.

2.7) Arduino 1.8.13 Compiler

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

2.8) Android Studio 4.1

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

2.9) Android Application

Android App is a software designed to run on an Android device or emulator.Android App is a software designed to run on an Android device or emulator.

3) Details of Application

3.1) Arduino Compiler Codes

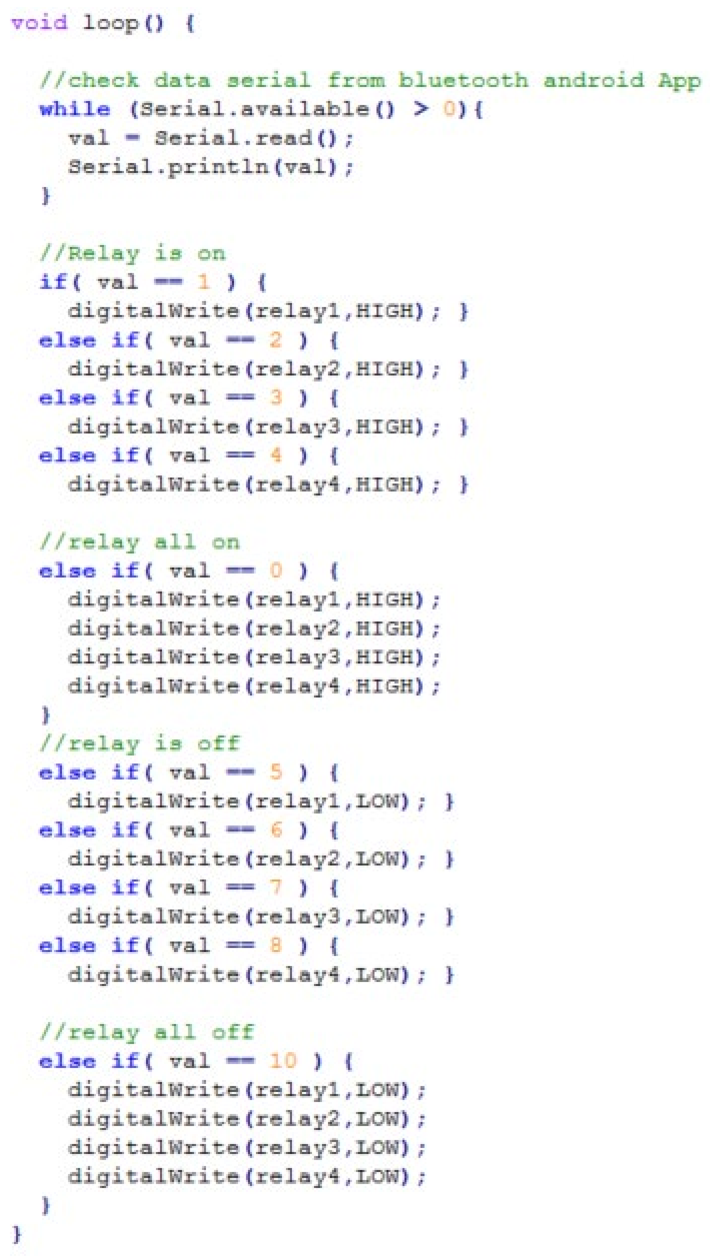
```
//using ports 10, 11, 12, 13
int relay1 = 10;
int relay2 = 11;
int relay3 = 12;
int relay4 = 13;
int val;

void setup() {

  Serial.begin(9600);
  pinMode(relay1,OUTPUT);
  pinMode(relay2,OUTPUT);
  pinMode(relay3,OUTPUT);
  pinMode(relay4,OUTPUT);
  digitalWrite(relay1,LOW);
  digitalWrite(relay2,LOW);
  digitalWrite(relay3,LOW);
  digitalWrite(relay4,LOW);

}
```

The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each powerup or reset of the Arduino board.

Serial.begin () sets the communication rate in bits per second.

The pinMode() function is used to configure a specific pin to behave either as an input or an output.

The digitalWrite() function is used to write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.



```
void loop() {

  //check data serial from bluetooth android App
  while (Serial.available() > 0){
    val = Serial.read();
    Serial.println(val);
  }

  //Relay is on
  if( val == 1 ) {
    digitalWrite(relay1,HIGH); }
  else if( val == 2 ) {
    digitalWrite(relay2,HIGH); }
  else if( val == 3 ) {
    digitalWrite(relay3,HIGH); }
  else if( val == 4 ) {
    digitalWrite(relay4,HIGH); }

  //relay all on
  else if( val == 0 ) {
    digitalWrite(relay1,HIGH);
    digitalWrite(relay2,HIGH);
    digitalWrite(relay3,HIGH);
    digitalWrite(relay4,HIGH);
  }
  //relay is off
  else if( val == 5 ) {
    digitalWrite(relay1,LOW); }
  else if( val == 6 ) {
    digitalWrite(relay2,LOW); }
  else if( val == 7 ) {
    digitalWrite(relay3,LOW); }
  else if( val == 8 ) {
    digitalWrite(relay4,LOW); }

  //relay all off
  else if( val == 10 ) {
    digitalWrite(relay1,LOW);
    digitalWrite(relay2,LOW);
    digitalWrite(relay3,LOW);
    digitalWrite(relay4,LOW);
  }
}
```

Serial.read() reads incoming serial data.

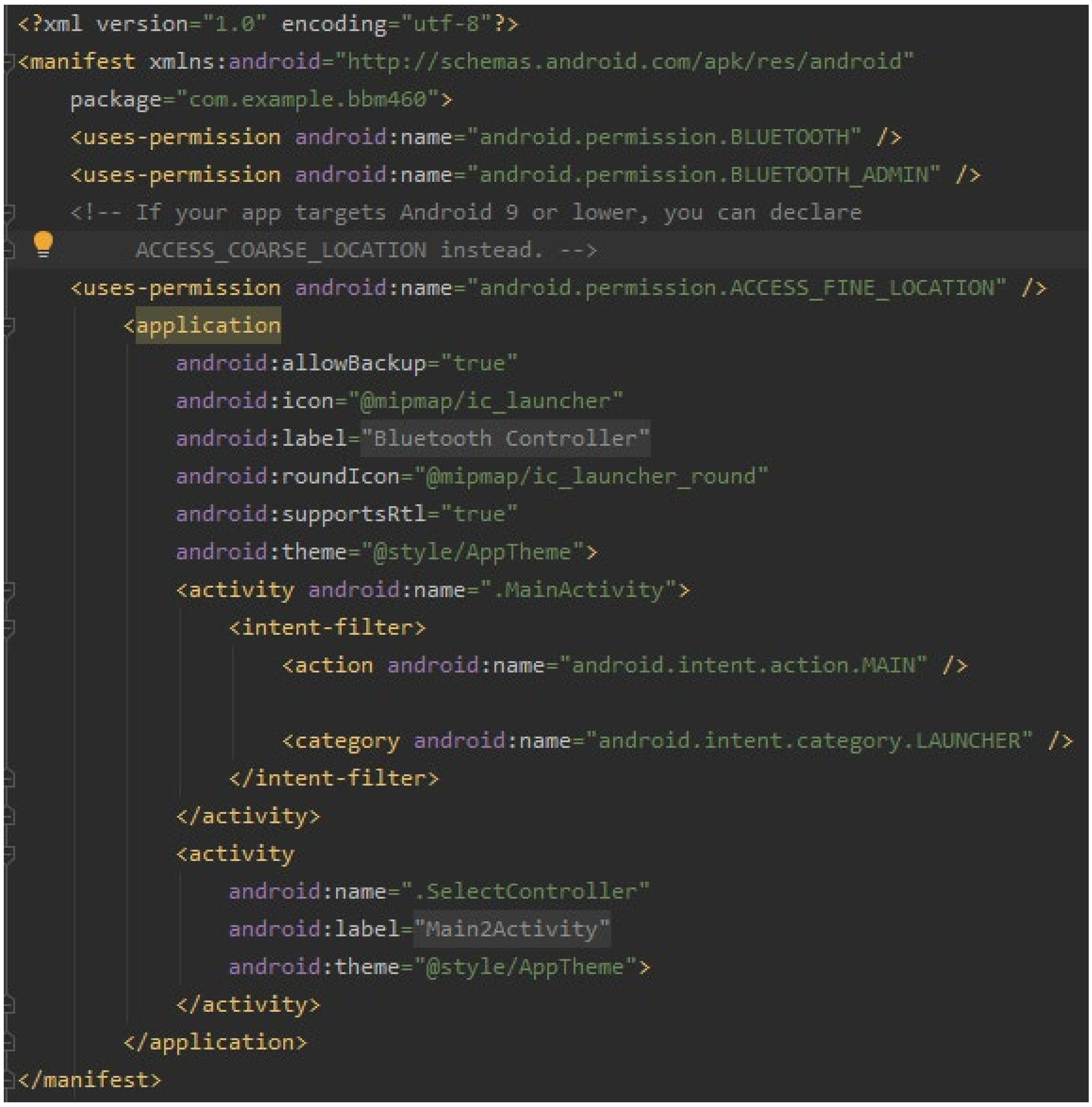
Serial.available() get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).

The loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond.

First While loop check data serial from Bluetooth Androip App. If the application sends data, it reads it with the Serial.read() function and saves in val variable. After the data is saved, the pins are adjusted according to the value in the data.

3.1) Android Studio Codes

AndroidManifest.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bbm460">
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <!-- If your app targets Android 9 or lower, you can declare
         ACCESS_COARSE_LOCATION instead. -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
        <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="Bluetooth Controller"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/AppTheme">
            <activity android:name=".MainActivity">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
            <activity
                android:name=".SelectController"
                android:label="Main2Activity"
                android:theme="@style/AppTheme">
            </activity>
        </application>
</manifest>
```

The AndroidManifest.xml file contains information of package, including components of the application such as activities, services, broadcast receivers, content providers etc.

It is responsible to protect the application to access any protected parts by providing the permissions.

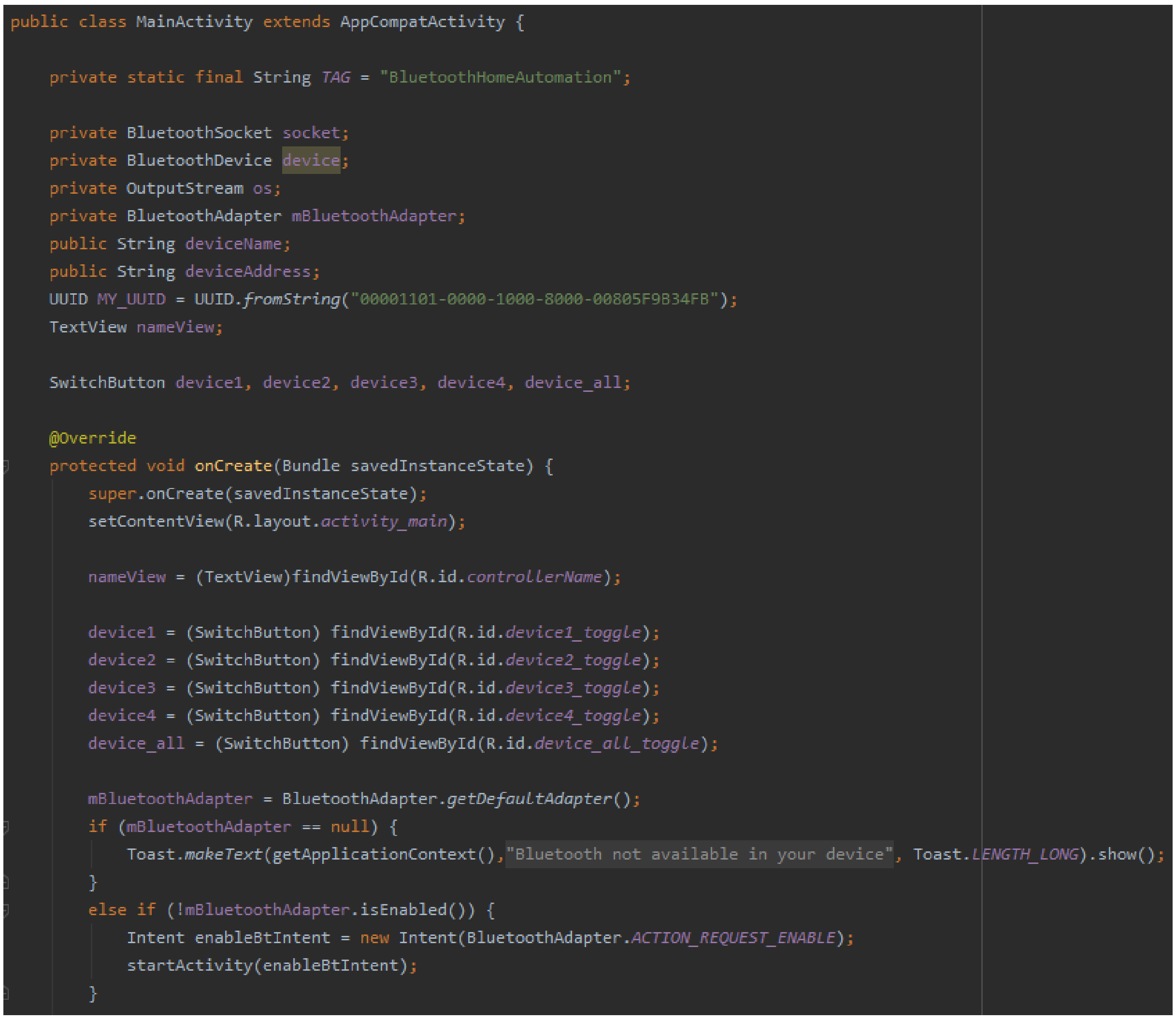
It also declares the android api that the application is going to use.

It lists the instrumentation classes. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

This is the required xml file for all the android application and located inside the root directory.

AndroidManifest has 2 activity. The first is MainActivity, the second is SelectControls. MainActivity is the first class that will run when our android application is started. SelectController is for bluetooth connection

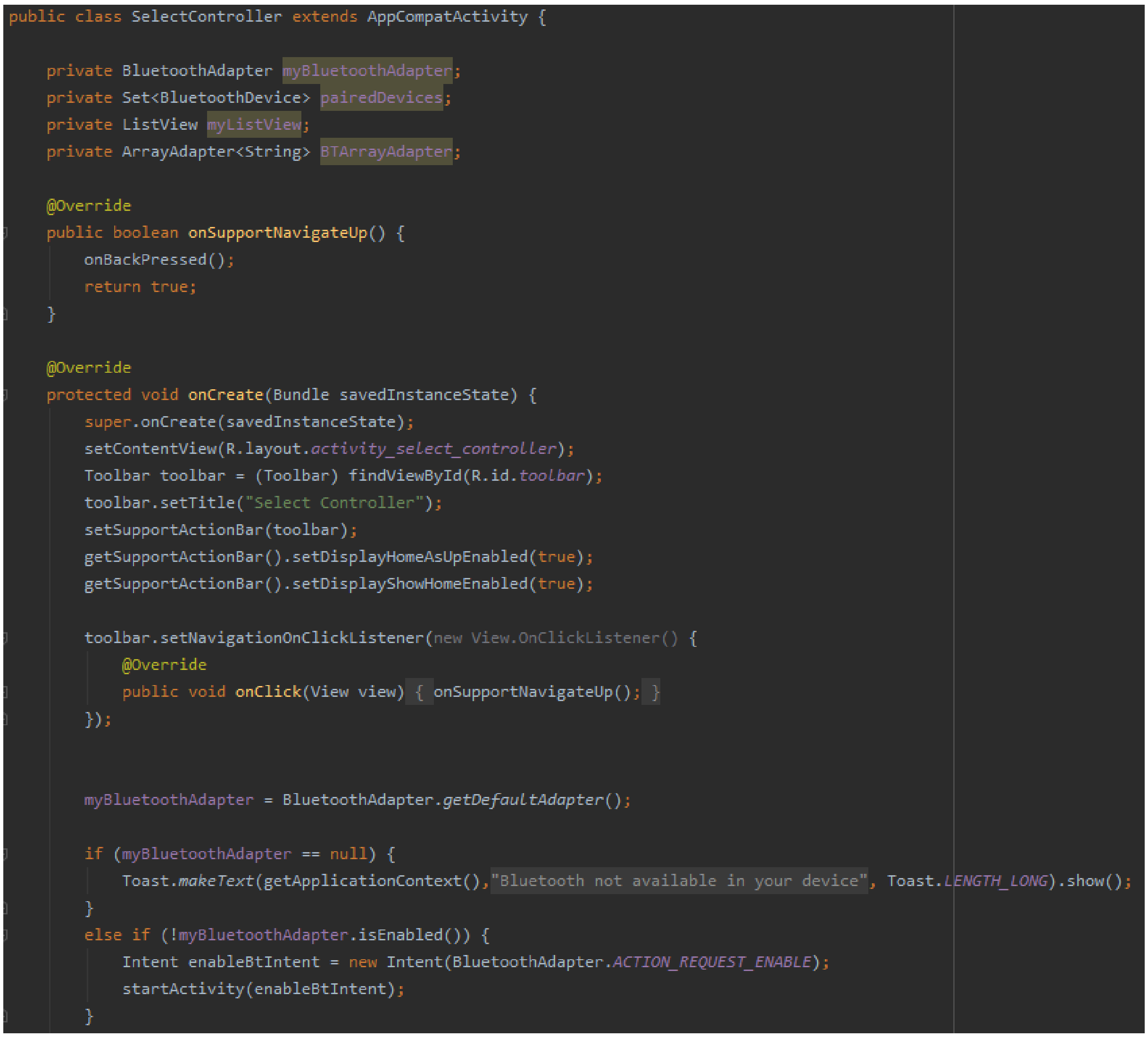
MainActivity.java



```java
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "BluetoothHomeAutomation";

    private BluetoothSocket socket;
    private BluetoothDevice device;
    private OutputStream os;
    private BluetoothAdapter mBluetoothAdapter;
    public String deviceName;
    public String deviceAddress;
    UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    TextView nameView;

    SwitchButton device1, device2, device3, device4, device_all;

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

        nameView = (TextView)findViewById(R.id.controllerName);

        device1 = (SwitchButton) findViewById(R.id.device1_toggle);
        device2 = (SwitchButton) findViewById(R.id.device2_toggle);
        device3 = (SwitchButton) findViewById(R.id.device3_toggle);
        device4 = (SwitchButton) findViewById(R.id.device4_toggle);
        device_all = (SwitchButton) findViewById(R.id.device_all_toggle);

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {
            Toast.makeText(getApplicationContext(),"Bluetooth not available in your device", Toast.LENGTH_LONG).show();
        }
        else if (!mBluetoothAdapter.isEnabled()) {
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivity(enableBtIntent);
        }
```

It is the part that runs when the program is opened on the phone. It will not work if the phone does not have Bluetooth. The Bluetooth connection is selected from the SelectController section. After the Bluetooth connection is established, the signal is sent with toggle buttons.

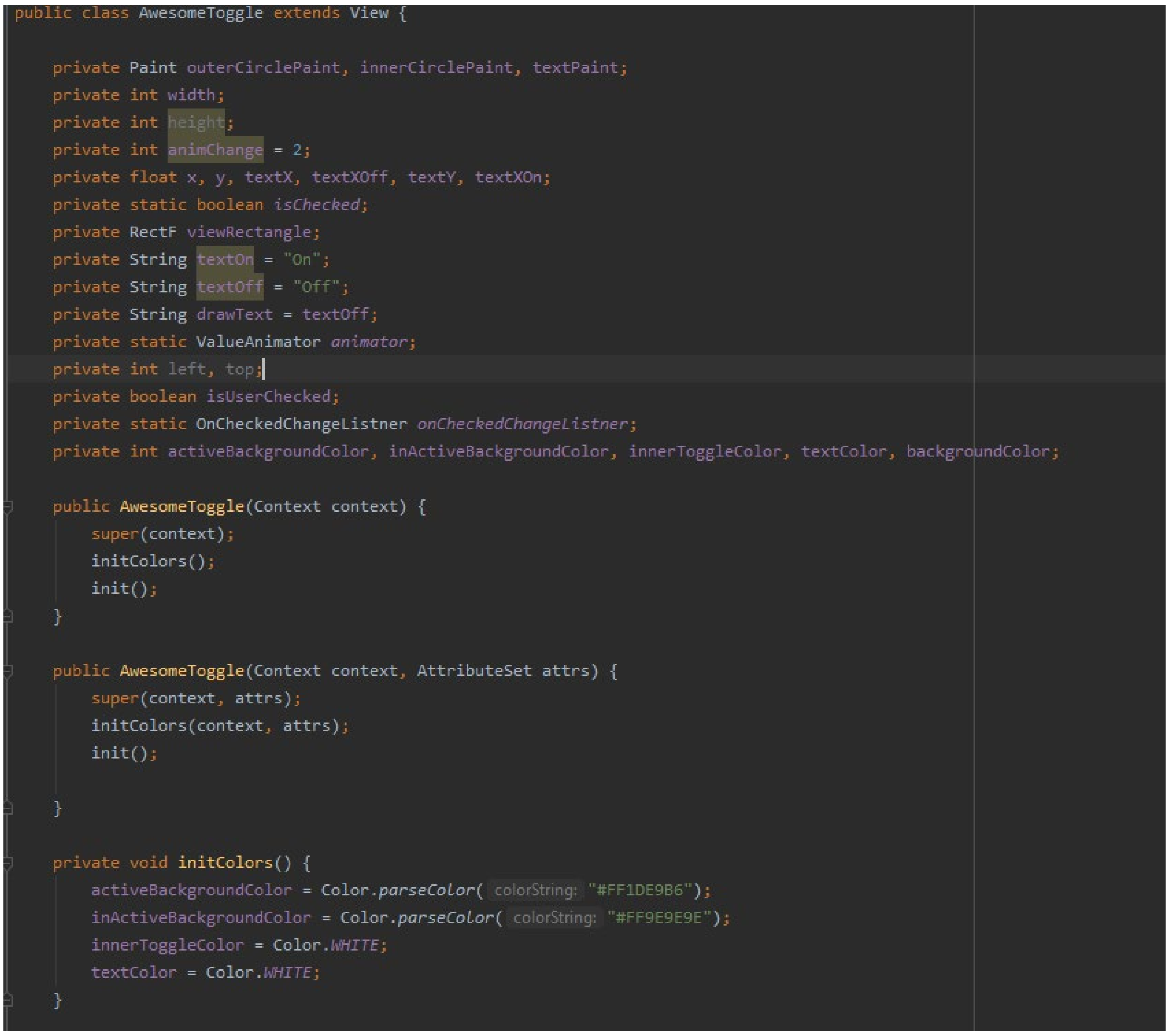
SelectController.java



```java
public class SelectController extends AppCompatActivity {

    private BluetoothAdapter myBluetoothAdapter;
    private Set<BluetoothDevice> pairedDevices;
    private ListView myListView;
    private ArrayAdapter<String> BTArrayAdapter;

    @Override
    public boolean onSupportNavigateUp() {
        onBackPressed();
        return true;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_select_controller);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        toolbar.setTitle("Select Controller");
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { onSupportNavigateUp(); }
        });


        myBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        if (myBluetoothAdapter == null) {
            Toast.makeText(getApplicationContext(),"Bluetooth not available in your device", Toast.LENGTH_LONG).show();
        }
        else if (!myBluetoothAdapter.isEnabled()) {
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivity(enableBtIntent);
        }
```

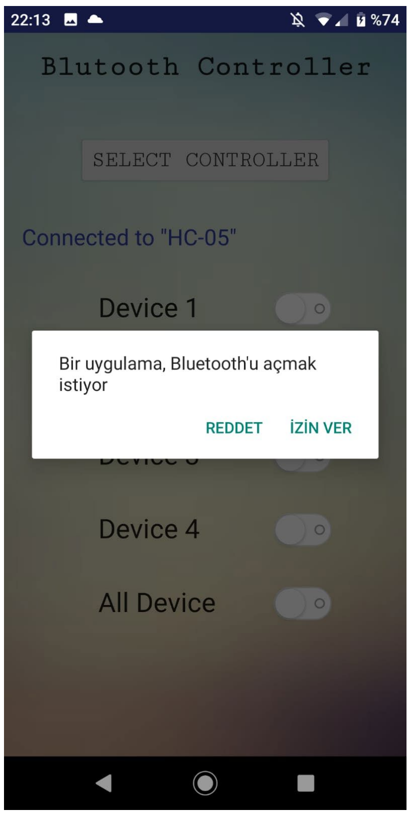
Lists paired devices on bluetooth. Establishes a connection with the device selected through the list.

AwesomeToggle.java



```
public class AwesomeToggle extends View {

    private Paint outerCirclePaint, innerCirclePaint, textPaint;
    private int width;
    private int height;
    private int animChange = 2;
    private float x, y, textX, textXOff, textY, textXOn;
    private static boolean isChecked;
    private RectF viewRectangle;
    private String textOn = "On";
    private String textOff = "Off";
    private String drawText = textOff;
    private static ValueAnimator animator;
    private int left, top;
    private boolean isUserChecked;
    private static OnCheckedChangeListner onCheckedChangeListner;
    private int activeBackgroundColor, inActiveBackgroundColor, innerToggleColor, textColor, backgroundColor;

    public AwesomeToggle(Context context) {
        super(context);
        initColors();
        init();
    }

    public AwesomeToggle(Context context, AttributeSet attrs) {
        super(context, attrs);
        initColors(context, attrs);
        init();

    }

    private void initColors() {
        activeBackgroundColor = Color.parseColor( colorString: "#FF1DE9B6");
        inActiveBackgroundColor = Color.parseColor( colorString: "#FF9E9E9E");
        innerToggleColor = Color.WHITE;
        textColor = Color.WHITE;
    }
```

It provides graphical image change when the toggle button is pressed.


22:13 %74

Blutooth Controller

SELECT CONTROLLER

Connected to "HC-05"

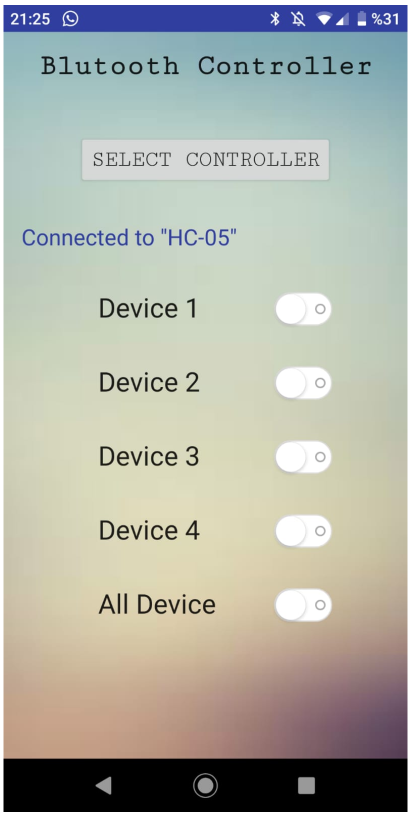
Device 1

Bir uygulama, Bluetooth'u açmak istiyor

REDDET İZİN VER

Device 4

All Device


21:25 %31

Blutooth Controller

SELECT CONTROLLER

Connected to "HC-05"

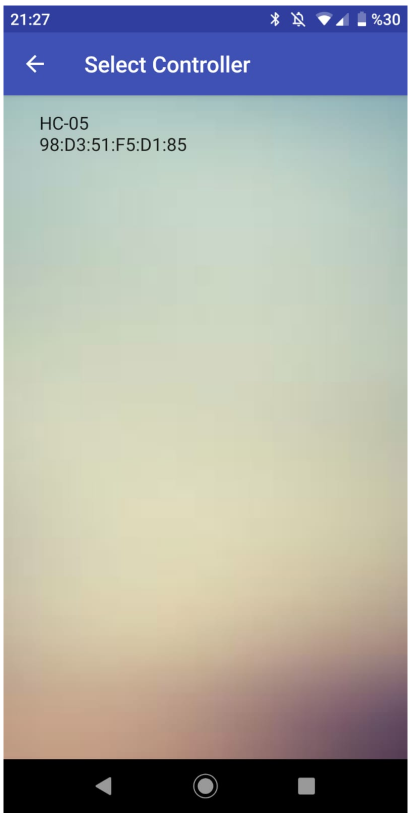
Device 1

Device 2

Device 3

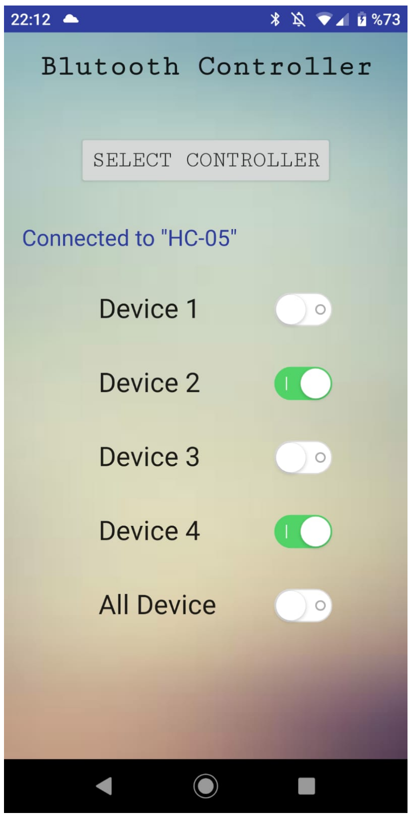
Device 4

All Device


21:27 %30

Select Controller

HC-05
98:D3:51:F5:D1:85


22:12 %73

Blutooth Controller

SELECT CONTROLLER

Connected to "HC-05"

Device 1

Device 2

Device 3

Device 4

All Device