Bluetooth

The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices.

In order to help your development, I have created a class that you only need to put on your project that encapsulate the main functions of a Bluetooth. This class is called BluetoothManager, that is responsible for all the interaction with the bluetooth and it has an interface to use it that is called BluetoothCallback.

You can include this two classes in your project (maybe you need to change the package in order to work it properly).

The process of working the bluetooth is kind of complex. First we check if there is a bluetooth available and turn it on (if it is off). After we need to put one device in a discovery mode, with a specific name, and another in the listening mode, looking for the name of the other device. If it finds the connection, it creates a socket, that will be able to exchange messages. So in order to help you, I have create a class that encapsulate all of that, which is named BluetoothManager and is a singleton (i.e. there is only one object of each shared in all of the APP). The main functions of the BluetoothManager can be seen bellow:

- public static BluetoothManager **getInstance**() -> used to access the Singleton class, in order to use the functions;
- public boolean **initializeBluetooth**(Activity activity,String uuid,String name)-> used to initialize the bluetooth class, should be the first thing to do when you want to use the class. You need to pass the activity that is using it, an unique identifier that should be unique for every pair of devices (you can create one here: https://www.uuidgenerator.net/) and a unique name for the pair of devices;
- public void **closeBluetooth**() -> used to close the bluetooth. Should be done when you are not going to use the Bluetooth connection anymore.
- public int **turnOnBluetooth**(Activity activity)-> turn on the bluetooth on the device;
- public void **turnOffBluetooth**()-> turn off the bluetooth on the device;
- public boolean **isBluetoothOn**()-> check if the bluetooth is on;
- public void **makeBluetoothDiscoverable**(Activity activity,int time,BluetoothCallback bluetoothCallback)-> active the bluetooth as discoverable for time seconds.
- public void **startDiscover**(BluetoothCallback bluetoothCallback)-> tries to discover the device that is discoverable with the name that was used in the **initiazeBluetooth**;
- public int **CheckActivityResult**(int requestCode, int resultCode)-> used to check the messages that get back from the activity. Should be implemented in the onActivityResult of the Activity;
- public void **sendData**(BluetoothCallback bluetoothCallback,String data) -> used to send a string data to the other device;
- public void **startReadingData**(BluetoothCallback bluetoothCallback)-> used to start a threat that will keep listening to any read messages that come, and send it though the callback **onReceiveData;**
- public void **stopReadingData**()-> stop the thread that is responsible for listening to the bluetooth connection.

We also need a interface for having a callback on the activity of whats has happened in the bluetooth. This interface is called BluetoothCallback and has the following functions:

- abstract void **onBluetoothConnection**(int returnCode)-> The onBluetoothConnection is responsible to receive messages about how the Connection between the devices has been. It can be an BLUETOOTH_CONNECTED, which is when the devices connected, or a BLUETOOTH_CONNECTED_ERROR, when there where some errors between the connection.

- abstract void **onBluetoothDiscovery**(int returnCode)-> The onBluetoothDiscovery is responsible to receive messages about the discovery state of the bluetooth. It can receive BLUETOOTH_DISCOVERABLE when it is in a discoverable mode, BLUETOOTH_CONNECTABLE when it is in a connectable mode (ie it can be connected by devices that already know this device) or BLUETOOTH_NOT_CONNECTABLE (that it cannot be connected by any device);

- abstract void **onReceiveData**(String data)-> used to receive messages from the bluetooth. When implementing in an activity, if you need to change thing on the UI, you need to put on runOnUiThread in order to avoid errors on the execution, since the messages are received in a different threads;

Now we are going to start building our application.

In our application we are going to have 2 Activities, one that is responsible for the connection between the devices (BluetoothConnectionActivity), and another to exchange the messages between then (BluetoothCallback).

Then in order to use the Bluetooth you need the permission to do it so in your manifest. So we have the following XML.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.markjoselli2015.bluetoothdemo" >

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".BluetoothConnectionActivity"
            android:label="@string/title_activity_bluetooth" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ConnectedActivity"
            android:label="@string/title_activity_connected" >
        </activity>
```

```
    </application>

</manifest>
```

We can see highlighter two lines, this lines are responsible for the permission to use the bluetooth by the application.

For the first activity, we are going to have the following screen:

One toogle to change the on/off stage of the bluetooth, one toogle to put the device on discoverable mode, and a button to discover devices that were on a discoverable mode. That screen has the following xml.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:paddingLeft="@dimen/activity_horizontal_margin"
            android:paddingRight="@dimen/activity_horizontal_margin"
            android:paddingTop="@dimen/activity_vertical_margin"
            android:paddingBottom="@dimen/activity_vertical_margin"

tools:context="com.example.markjoselli2015.bluetoothdemo.BluetoothConnectionActivity"
            android:orientation="vertical">


    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Bluetooth:"
            android:id="@+id/textView"/>

        <ToggleButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New ToggleButton"
            android:id="@+id/toggleButtonBluetooth"/>
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Visible:"
            />

        <ToggleButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New ToggleButton"
            android:id="@+id/toggleButtonVisible"/>

        <ProgressBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/progressBarDiscover"
            android:visibility="invisible"/>
    </LinearLayout>

    <Button
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Connect"
        android:id="@+id/buttonConnect"
        android:layout_gravity="center_horizontal"
        android:visibility="visible"/>
</LinearLayout>
```

In our first activity, BluetoothConnectionActivity is responsible for the connection between the devices, and we have the following code, with comments of what is it doing:

```
//class activity that implements the Bluetooth callback
public class BluetoothConnectionActivity extends Activity implements BluetoothCallback{
    //declaration of the UI components
    ToggleButton toggleButtonBluetooth,toggleButtonVisible;
    ProgressBar spinnerDiscovering;
    Button buttonConnect;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bluetooth);

        //initialize the bluetooth with the activity,
        // an UUID THAT YOU MUST GENERATE IN https://www.uuidgenerator.net/
        //and a unique name (that you also must change it!!!
        BluetoothManager.getInstance().initializeBluetooth(this,
                "321cb8fa-9066-4f58-935e-ef55d1ae06ec","bluetooth.communicator");
        //all the functions from BluetoothManager must be accessible by using getInstance in
        // order to use the singleton


        //pick up the UI components
        buttonConnect = (Button)findViewById(R.id.buttonConnect);
        toggleButtonBluetooth = (ToggleButton)findViewById(R.id.toggleButtonBluetooth);
        spinnerDiscovering = (ProgressBar)findViewById(R.id.progressBarDiscover);
        toggleButtonVisible = (ToggleButton)findViewById(R.id.toggleButtonVisible);

        //set the toogleButton that will show how is the state of the bluetooth to the correct state
        toggleButtonBluetooth.setChecked(BluetoothManager.getInstance().isBluetoothOn());

        //on the click will turn on or off the bluetooth device
        toggleButtonBluetooth.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                if(isChecked){
                    BluetoothManager.getInstance().turnOnBluetooth(BluetoothConnectionActivity.this);
                }else{
                    BluetoothManager.getInstance().turnOffBluetooth();
                }
            }
        });

        //on the click will turn on the discovery mode on the device for 10 seconds
        toggleButtonVisible.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
```

```
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if(isChecked){
                if(!BluetoothManager.getInstance().isBluetoothOn()){
                    Toast.makeText(BluetoothConnectionActivity.this,"The BT device is OFF!",
                            Toast.LENGTH_SHORT).show();
                    toggleButtonVisible.setChecked(false);
                    return;
                }

BluetoothManager.getInstance().makeBluetoothDiscoverable(BluetoothConnectionActivity
                        .this,10,BluetoothConnectionActivity
                        .this);
                toggleButtonVisible.setEnabled(false);
                spinnerDiscovering.setVisibility(View.VISIBLE);
                buttonConnect.setVisibility(View.INVISIBLE);
            }else{

            }
        }
    });

    //on the click will start the connect mode
    //one device must be on the Discovey mode... and another in this connect mode!
    buttonConnect.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(!BluetoothManager.getInstance().isBluetoothOn()) {
                Toast.makeText(BluetoothConnectionActivity.this, "The BT device is OFF!",
                        Toast.LENGTH_SHORT).show();
                return;
            }
            BluetoothManager.getInstance().startDiscover(BluetoothConnectionActivity.this);

        }
    });
}

//on the destroy we make sure to close the connections that we made.
@Override
protected void onDestroy() {
    BluetoothManager.getInstance().closeBluetooth(this);
    super.onDestroy();
}

//on onActivityResult we must implement the CheckActivityResult of the BluetoothManager that
// will check results about some request that were made about discovering and on/off
// relations to bluetooth.
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    int resultBluetooth = BluetoothManager.getInstance().CheckActivityResult(requestCode,
            resultCode);

    if(resultBluetooth == BluetoothManager.BLUETOOTH_ON){
        toggleButtonBluetooth.setChecked(true);
    }else if(resultBluetooth == BluetoothManager.BLUETOOTH_OFF){
        toggleButtonBluetooth.setChecked(false);
```

```
        }else if(resultBluetooth == BluetoothManager.BLUETOOTH_DISCOVERY_LISTEN){
            toggleButtonVisible.setEnabled(false);
            spinnerDiscovering.setVisibility(View.VISIBLE);
            buttonConnect.setVisibility(View.INVISIBLE);
        }else if(resultBluetooth == BluetoothManager.BLUETOOTH_DISCOVERY_CANCELED){
            toggleButtonVisible.setEnabled(true);
            toggleButtonVisible.setChecked(false);
            spinnerDiscovering.setVisibility(View.INVISIBLE);
            buttonConnect.setVisibility(View.VISIBLE);
        }

        super.onActivityResult(requestCode,resultCode,data);
    }

    //callback for when the device is connected or an error occurred. When the connection os ok,
    // it starts a new Activity for the message exchange
    @Override
    public void onBluetoothConnection(int returnCode) {
        if(returnCode == BluetoothManager.BLUETOOTH_CONNECTED){
            Toast.makeText(BluetoothConnectionActivity.this, "Connected",
                    Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(this, ConnectedActivity.class);

            startActivity(intent);
        }else if(returnCode == BluetoothManager.BLUETOOTH_CONNECTED_ERROR){
            Toast.makeText(BluetoothConnectionActivity.this, "ConnectionError",
                    Toast.LENGTH_SHORT).show();
        }
    }

    //callback for when the device is discoverable or if it ended the discoverable mode.
    @Override
    public void onBluetoothDiscovery(int returnCode) {
        if(returnCode == BluetoothManager.BLUETOOTH_DISCOVERABLE){
            toggleButtonVisible.setEnabled(false);
            spinnerDiscovering.setVisibility(View.VISIBLE);
            buttonConnect.setVisibility(View.INVISIBLE);

        }else if(returnCode == BluetoothManager.BLUETOOTH_CONNECTABLE ||
                returnCode == BluetoothManager.BLUETOOTH_NOT_CONNECTABLE){
            toggleButtonVisible.setEnabled(true);
            toggleButtonVisible.setChecked(false);
            spinnerDiscovering.setVisibility(View.INVISIBLE);
            buttonConnect.setVisibility(View.VISIBLE);
        }
    }

    //callback for when the bluetooth receive some that, that we are going to deal in a separated
    // activity
    @Override
    public void onReceiveData(String data) {

    }
}
```

When the devices connect they start a second activity, that has the following screen:

That has a TextView, and EditText(that will be data that will be send to the other device) and a button (to send the data). The that that is received will appear as a Toast message. This interface has the following XML:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:paddingBottom="@dimen/activity_vertical_margin"
        tools:context="com.example.markjoselli2015.bluetoothdemo.ConnectedActivity">
```

```
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">

        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Send:"/>

            <EditText
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/editTextToSend"/>

        </LinearLayout>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Send"
            android:id="@+id/buttonSend"
            android:layout_centerHorizontal="true"/>

    </LinearLayout>

</RelativeLayout>
```

Then we have the class that is responsible for sending and receiving the messages from the bluetooth connection.

```
//class activity that implements the Bluetooth callback
public class ConnectedActivity extends Activity implements BluetoothCallback {
    //declaration of the UI components
    Button buttonSend;
    EditText editTextToSend;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_connected);

        //start the thread responsible of receiving the data from the other device
        BluetoothManager.getInstance().startReadingData(this);

        //pick up the UI components
        editTextToSend = (EditText)findViewById(R.id.editTextToSend);
        buttonSend = (Button)findViewById(R.id.buttonSend);

        //when the button is clicked send the data from the EditText
        buttonSend.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                BluetoothManager.getInstance().sendData(ConnectedActivity.this,
                        editTextToSend.getText().toString());
            }
        });


    }

    @Override
    protected void onDestroy() {
        //stop the thread responsible for reading the data.
        BluetoothManager.getInstance().stopReadingData();
        super.onDestroy();
    }


    @Override
    public void onBluetoothConnection(int returnCode) {

    }

    @Override
    public void onBluetoothDiscovery(int returnCode) {

    }

    @Override
    public void onReceiveData(String data) {
        //if it receives a new data, put on a toast on the UIThread
        final String finalData = data;
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(getApplicationContext(), finalData,
                        Toast.LENGTH_LONG).show();
            }
        });

    }
}
```

If you test the application it will connect and receive/send messages between devices. One observation is that bluetooth does not work on the emulator, so you will need a real device to test it.