

IDServices BOTG-SDK User Guide

Android Programming

Version 1.00



Copyright © 2017, IDServices

All rights reserved

RELEASE NOTES

<u>Version</u>	<u>Date</u>	<u>Notes</u>
1.00	Nov. 03, 2017	First Release

Contents

RELEASE NOTES	2
INTRODUCTION	4
INSTRUCTIONS.....	5
I. Import Library in Android Studio.....	5
II. Add Library to your app.....	7
III. Add an external dependency	9
WARNING	11
JAVA DOCUMENTATION.....	12
BOTGDriver.....	12
I. Overview.....	12
II. Constants.....	12
III. Methods	12
Connection	13
I. Overview.....	13
II. Methods	13
SlaveConnection	14
I. Overview.....	14
II. Methods	14
MasterConnection.....	14
I. Overview.....	14
II. Methods	14
SAMPLE CODE.....	15
I. Manifest.xml.....	15
II. MainActivity.java.....	15
III. Activity_layout.xml.....	17

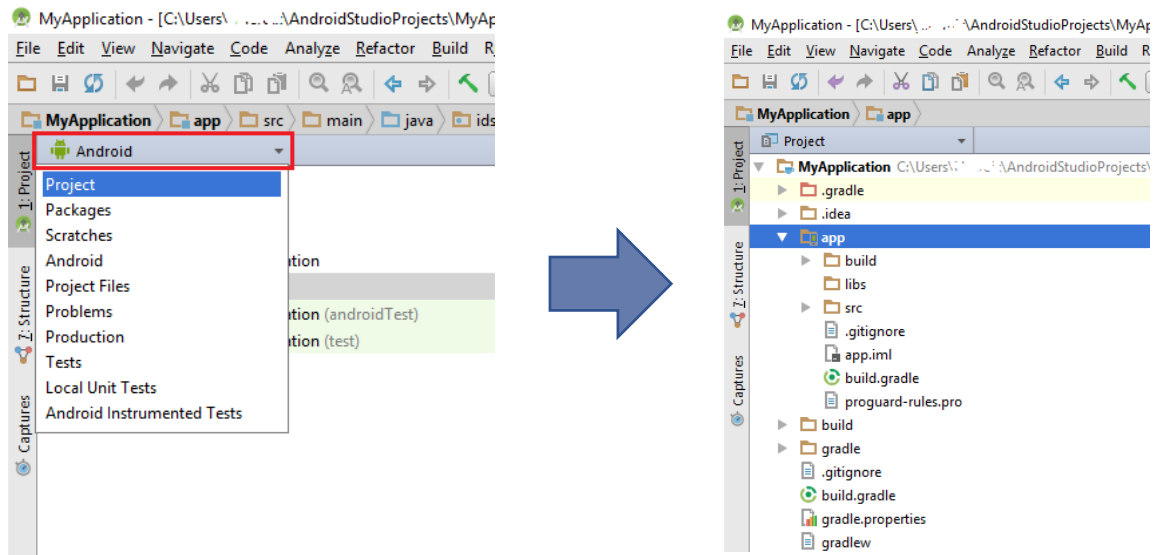
INTRODUCTION

This User Guide contains necessary information for building Android applications that can easily manage Bluetooth connections with BOTGs, receive and send data through Serial Port Profile (SPP) connections.

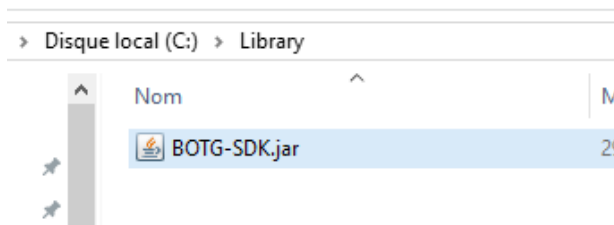
INSTRUCTIONS

I. Import Library in Android Studio

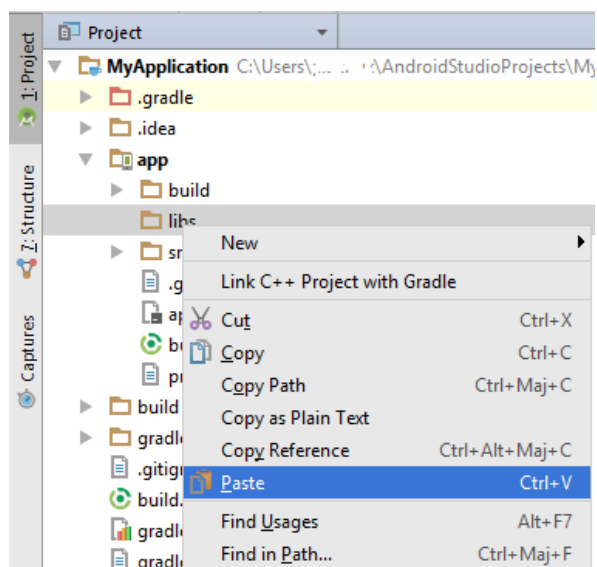
- 1) After creating an Android Studio project, click the **Android** project view icon to switch to the Traditional project view.



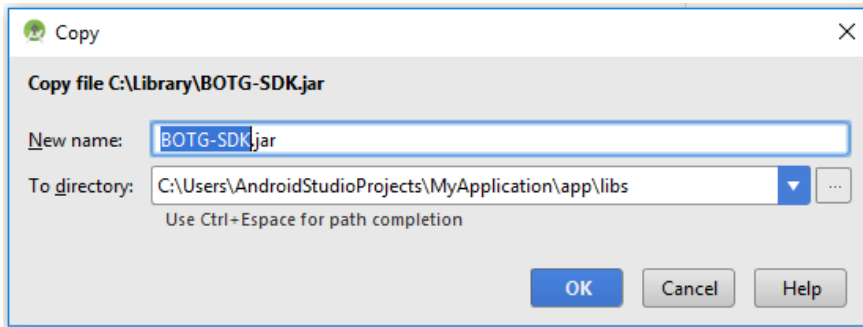
- 2) Locate the **"BOTG-SDK.jar"** library file in your system and **copy** it



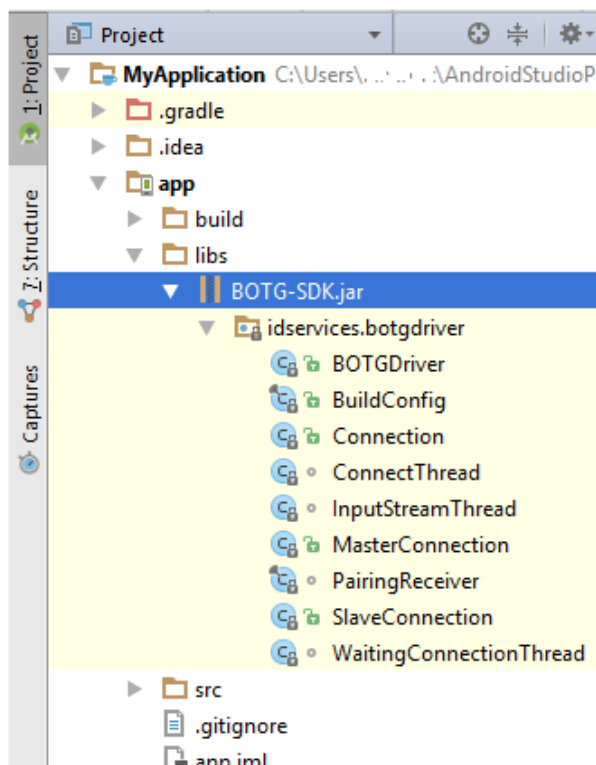
- 3) Right click on the **libs** folder and select **Paste**.



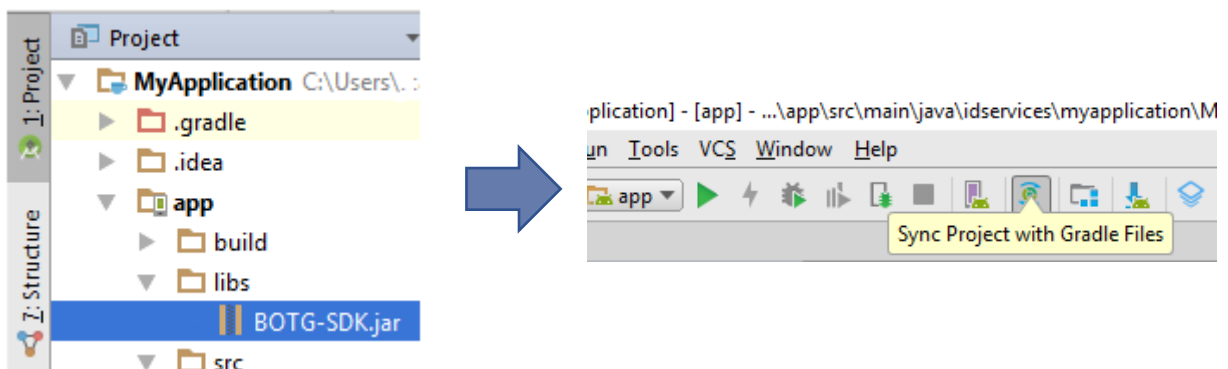
- 4) A dialog will show up indicating the file name and the destination directory to be copied. Click the **OK** button to confirm the importation.



- 5) In the project view, you can see the library is imported

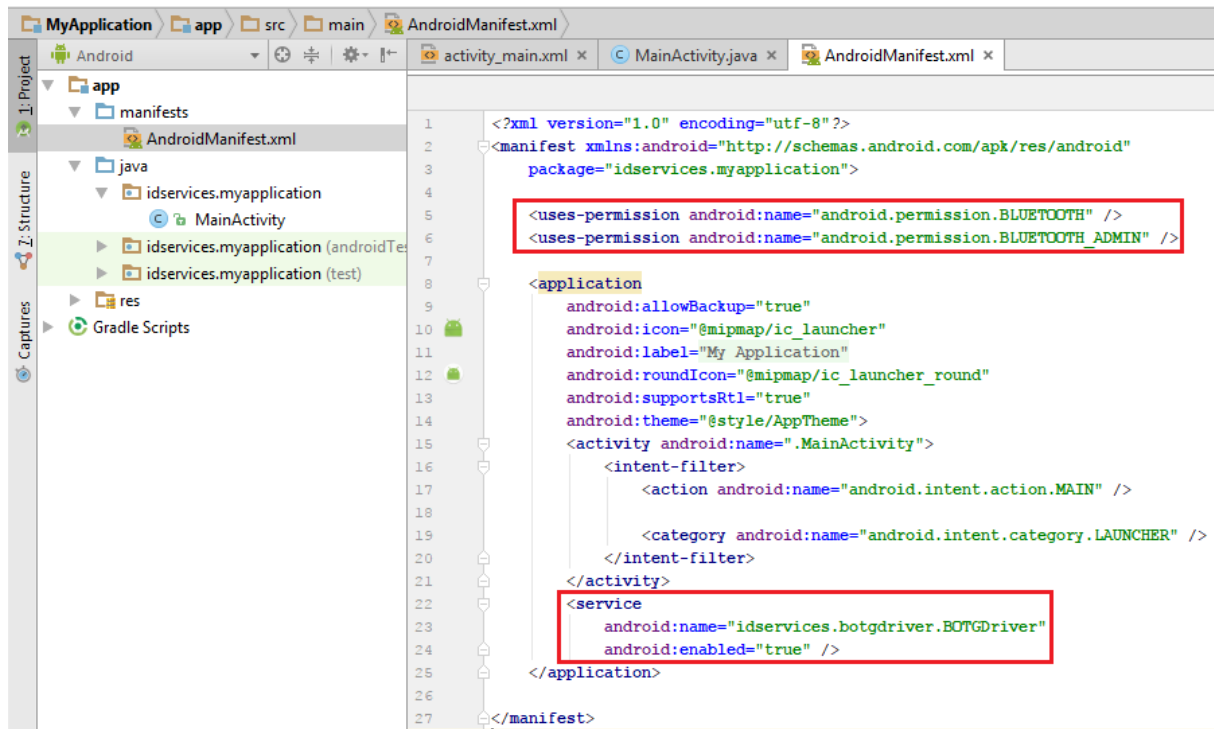


If you don't see any files listed under the **BOTG-SDK.jar** item, please click the **Sync Project with Gradle Files** button in the toolbar.



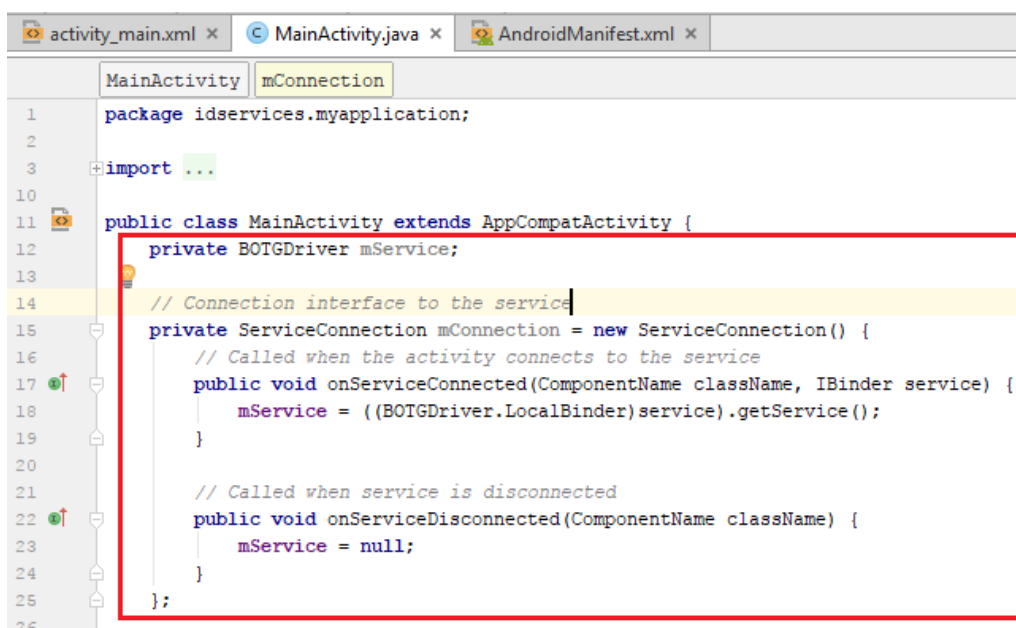
II. Add Library to your app

- 1) First, you need to ask for the permissions needed and declare the service by add the following surrounded lines in your app **AndroidManifest.xml**.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 package="idservices.myapplication">
4
5 <uses-permission android:name="android.permission.BLUETOOTH" />
6 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
7
8 <application
9 android:allowBackup="true"
10 android:icon="@mipmap/ic_launcher"
11 android:label="My Application"
12 android:roundIcon="@mipmap/ic_launcher_round"
13 android:supportsRtl="true"
14 android:theme="@style/AppTheme">
15 <activity android:name=".MainActivity">
16 <intent-filter>
17 <action android:name="android.intent.action.MAIN" />
18 <category android:name="android.intent.category.LAUNCHER" />
19 </intent-filter>
20 </activity>
21 <service
22 android:name="idservices.botgdriver.BOTGDriver"
23 android:enabled="true" />
24 </application>
25 </manifest>
```

- 2) Then you must declare the service and the connection interface. Please go to the activity you want to bind the service and add these lines.



```
activity_main.xml x MainActivity.java x AndroidManifest.xml x
MainActivity mConnection
1 package idservices.myapplication;
2
3 import ...
10
11 public class MainActivity extends AppCompatActivity {
12     private BOTGDriver mService;
13
14     // Connection interface to the service
15     private ServiceConnection mConnection = new ServiceConnection() {
16         // Called when the activity connects to the service
17         public void onServiceConnected(ComponentName className, IBinder service) {
18             mService = ((BOTGDriver.LocalBinder) service).getService();
19         }
20
21         // Called when service is disconnected
22         public void onServiceDisconnected(ComponentName className) {
23             mService = null;
24         }
25     };
26 }
```


3) Bind the service to your activity in the `onCreate()` method.

```
28     private Intent mIntent;
29     private boolean mBounded;
30
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_main);
35
36         mIntent = new Intent(this, BOTGDriver.class);
37         bindService(mIntent, mConnection, BIND_AUTO_CREATE);
38         mBounded = true;
39     }
```

4) Use the BOTGDriver methods as you want.

Note: You may handle time process until the service has bound to your activity.

If you are calling BOTGDriver's methods asynchronously (e.g. calling method in an OnClickListener) you don't have to wait until the service has bound.

```
35     @Override
36     protected void onCreate(Bundle savedInstanceState) {
37         super.onCreate(savedInstanceState);
38         setContentView(R.layout.activity_main);
39
40         mIntent = new Intent(this, BOTGDriver.class);
41         bindService(mIntent, mConnection, BIND_AUTO_CREATE);
42         mBounded = true;
43
44         // Asynchronous call
45         Button btn = (Button) findViewById(R.id.btn);
46         btn.setOnClickListener(new View.OnClickListener() {
47             @Override
48             public void onClick(View v) {
49                 mService.disconnectToMaster();
50             }
51         });
52     }
53 }
```

Otherwise you have to put your code in `onServiceConnected()` function.

```
20     private ServiceConnection mConnection = new ServiceConnection() {
21         // Called when the activity connects to the service
22         public void onServiceConnected(ComponentName className, IBinder service) {
23             mService = ((BOTGDriver.LocalBinder) service).getService();
24
25             // Code here
26             mService.connectTo("00:00:00:00:00:02");
27     }
```

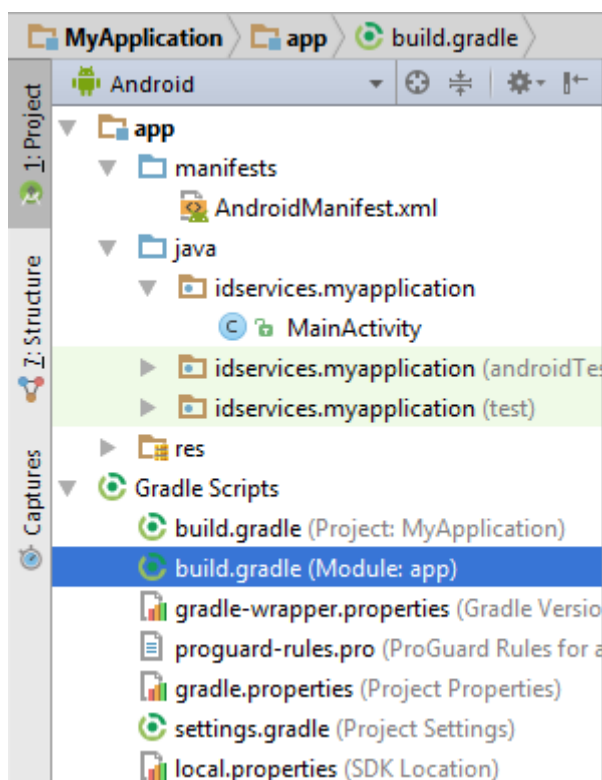
5) When finishing the activity, unbind the service (or stop it).

```
41         @Override
42         protected void onDestroy() {
43             super.onDestroy();
44             // Unbind the service
45             if (mBounded) {
46                 unbindService(mConnection);
47                 mBounded = false;
48             }
49             // Stop the service
50             stopService(mIntent);
51         }
```

III. Add an external dependency

The BOTG-SDK uses an external dependency to create a QR code as Bitmap. You must add an external dependency to successfully compile the SDK. Following these steps:

1) Open the **build.gradle** file corresponding to your app (annotated by “**(Module: app)**”).



2) Then add **compile 'com.google.zxing:core:3.3.0'** in the **dependencies**.

```
22 dependencies {
23     compile fileTree(dir: 'libs', include: ['*.jar'])
24     androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
25         exclude group: 'com.android.support', module: 'support-annotations'
26     })
27     compile 'com.android.support:appcompat-v7:25.3.1'
28     compile 'com.android.support.constraint:constraint-layout:1.0.2'
29     compile 'com.google.zxing:core:3.3.0'
30     testCompile 'junit:junit:4.12'
31 }
```

3) Finally click the **Sync Project with Gradle Files** button and your package will be downloaded and added to your project.



Note: To remove a dependency package, just delete the corresponding “*compile ...*” line in the *build.gradle* file and synchronize again the project.

WARNING

It might be impossible to establish Bluetooth connections with [waitForConnections\(\)](#) on some devices (timed out notification will be displayed, but not on API 17).

BOTGDriver

I. Overview

Android Service providing methods to manage Bluetooth connections with BOTGs. Avoid pairing process by setting programmatically the default pin code “1234”.

II. Constants

- **SPP_UUID**: SPP UUID. Value: 00001101-0000-1000-8000-00805F9B34FB.
- **ACTION_BLUETOOTH_BOTG_OUTPUT**: String containing the custom intent action which sends incoming data from the BOTG.
Value: “idservices.botgdriver.intent.action.BLUETOOTH_DATA_IN”.
- **EXTRA_BOTG_OUTPUT**: String to retrieve the Intent’s extra data.
Value: “idservices.botgdriver.extra.BOTG_DATA”.

III. Methods

void **connectTo(String address)**: Connect to *address*. The *address* must be a mac address.

void **disconnectAll()**: Close all connections. Equivalent to calling both *disconnectToMaster()* and *disconnectToSlaves()*.

void **disconnectToMaster()**: Disconnect to the master device (when connection was established with [connectTo\(address\)](#)).

void **disconnectToSlaves()**: Disconnect to all slave devices.

Bitmap **encodeBitmap(String code, int width, int height)**: return a Bitmap object (i.e. an image), which is the QR code representation of the given *code*. The Bitmap is *width* large and *height* high. (Recommended width equals to height to get a square Bitmap).

ArrayList<String> **getConnectedDevices()**: Return the list of the names of the connected devices.

[SlaveConnection](#) **getSlaveConnection()**: Return the [SlaveConnection](#) if a device is connected (with [connectTo\(address\)](#)), else return null.

ArrayList<[MasterConnection](#)> **getMasterConnections()**: Return the list of the MasterConnections if the WaitingConnectionThread is running (with [waitForConnections\(\)](#)), else return null.

boolean **isBOTGConnected()**: Return true if at least one BOTG is connected, false otherwise.

int **nbBOTGPaired()**: Return the number of BOTGs paired to the device.

void **sendCommand(String cmd)**: Send the command *cmd* to all connected devices.

void **waitForConnections()**: Launch a thread waiting for incoming connections. You can scan a connect code to allow the BOTG to initiate the connection.

Connection

I. [Overview](#)

Super class of [MasterConnection](#) and [SlaveConnection](#) managing connection's attributes.

II. [Methods](#)

void **close()**: close the connection (disconnect to the BOTG).

String **getDeviceName()**: return the name of the connected device (e.g.: "BOTG BA:A9:93").

void **write(byte[] command)**: send *command* to the BOTG (you can use *String.getBytes()* to get a byte array of a String).

SlaveConnection

I. Overview

Subclass of [Connection](#) corresponding to a connection created by [connectTo\(address\)](#).

II. Methods

[Inherited methods.](#)

MasterConnection

I. Overview

Subclass of [Connection](#) corresponding to a connection accepted by the android device (i.e. a connection initialized with [waitForConnections\(\)](#)).

II. Methods

[Inherited methods.](#)

SAMPLE CODE

I. Manifest.xml

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

    <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: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>
        <service
            android:name="idservices.botgdriver.BOTGDriver"
            android:enabled="true" />
        </application>
</manifest>
```

II. MainActivity.java

```
package idservices.myapplication;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```



```

import idservices.botgdriver.BOTGDriver;

public class MainActivity extends AppCompatActivity {
    private BOTGDriver mService;
    private Intent mIntent;
    private boolean mBounded;

    // Connection interface to the service
    private ServiceConnection mConnection = new ServiceConnection() {
        // Called when the activity connects to the service
        public void onServiceConnected(ComponentName className, IBinder
service) {
            mService = ((BOTGDriver.LocalBinder) service).getService();

            // TODO: Enter BOTG's mac address below
            String macAddress = "00:00:00:00:00:02";

            mService.connectTo(macAddress);
        }

        // Called when service is disconnected
        public void onServiceDisconnected(ComponentName className) {
            mService = null;
        }
    };

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

        mIntent = new Intent(this, BOTGDriver.class);
        bindService(mIntent, mConnection, BIND_AUTO_CREATE);
        mBounded = true;

        // Asynchronous call
        Button btn = (Button) findViewById(R.id.btn_disconnect);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mService.disconnectToMaster();
            }
        });
    }

    protected void onDestroy(){
        super.onDestroy();
        // Unbind the service
        if(mBounded){
            unbindService(mConnection);
            mBounded = false;
        }
        // Stop the service
        stopService(mIntent);
    }
}

```

III. Activity layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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"
tools:context="idservices.myapplication.MainActivity">

    <Button
        android:id="@+id/btn_disconnect"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click here to disconnect"/>
</RelativeLayout>
```