MIMO LED Service Android HTTP App Instructions And Library In apk

8th May, 2019

Version: 1.2

Table of Contents

1. Overview 3

2. Starting the App 3

3. Home screen 4

4. Controlling the LEDs 4

4.1 Controlling Internal LEDs 5

4.2 Controlling External RGB LED units 5

5. Enabling Auto-start 6

6. Overview 7

7. Adding the library 7

8. Added Code 9

9. Controlling the LEDs 10

1. Overview

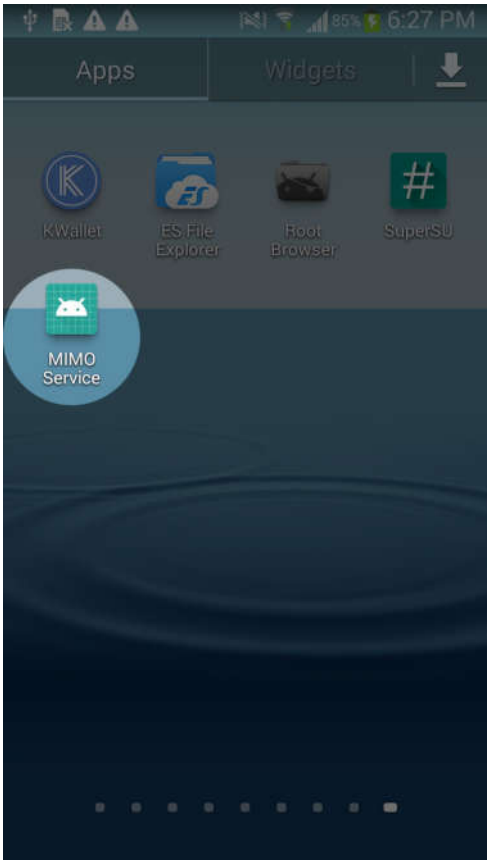
MIMO service is an app that runs on supported MIMO android tablet devices and allows controlling devices' internal LEDs or external RGB LED units by sending HTTP GET commands to the android device over the network.

The app automatically detects whether external RGB LEDs are connected or not. If external RGB LED units are not connected then the device's internal LEDs will be controlled as usual, otherwise, the external RGB LED units will be controlled.

The app can support a maximum of three external RGB LED units connected to the left, right and/or bottom USB ports of the device.

2. Starting the App

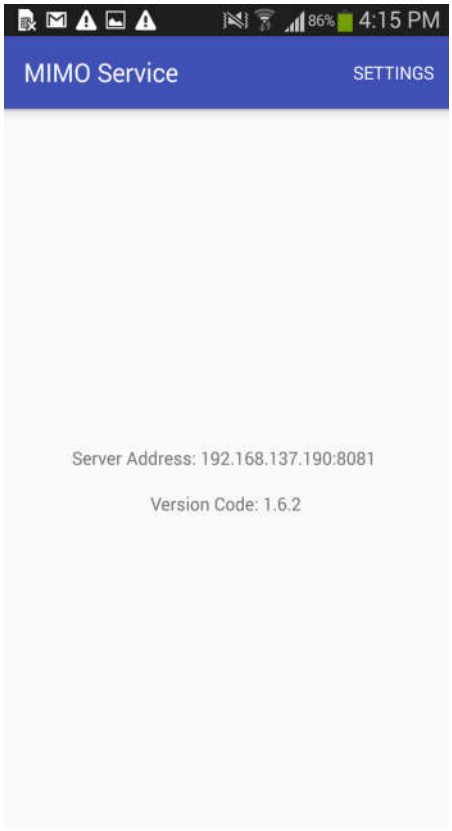
MIMO LED service android app can be launched by tapping the "MIMO Service" icon on the android application launcher screen.



3. Home screen

When the app is started, it displays the home screen. The home screen shows all the IPs on each network interface cards available on the device. If the device is not connected to a network, then “No Connection” will be displayed instead of the IP address.

The home screen also shows the current version of the MIMO LED service app.



4. Controlling the LEDs

When the MIMO LED service app is started, then a lightweight HTTP web server is also automatically started in the background. This HTTP web server listens for HTTP GET commands on port 8081. Any third party service can send an HTTP GET to control the internal LEDs. The format of the URL that needs to be issued to the device is as follows:

`http://[host]:8081/light?led=[rrggbb]`

where, [host] is the hostname or IP address of the device, and [rrggbb] is a 6 digit hex number, which represents the intensities of the red, green and blue components of the LEDs. It is passed as a value of the `led` query parameter in the URL.

The first two digits represent the intensity of the red component of the LED, the second two digits represent the intensity of the green component of the LED and the third two digits represent the intensity of the blue component of the LED.

4.1 Controlling Internal LEDs

The application will control the Internal LEDs only if no external RGB LED units are connected to any of the USB ports on the device, otherwise, the application will work for the external RGB LED units only.

For internal LEDs, the values of the intensities of each of the red, green and blue components of the LED can either be 00 or ff. It cannot be any value between 00 and ff.

For example, for internal LEDs, the following is a valid URL and will lit the LED with full intensity for the green component of the LED and 0 intensity for the red and blue components of the LED, on a device that has an IP of 192.168.1.12 assigned to it.

`http://192.168.1.12:8081/light?led=00ff00`

In order to turn off the LED, the value for the `led` parameter needs to be passed as `000000`.

4.2 Controlling External RGB LED units

The application will control the external RGB LED units, if at least one RGB LED unit is connected to any of the USB ports on the device. The application can work for a maximum of three RGB LED units connected to either the left, right or the bottom USB ports of the supported devices.

For external LEDs, the values of the intensities of each of the red, green and blue components of the LED can be any value between 00 and FF. However, it is important to note though, that the external RGB LED units only support the values between 00 and 7F inclusive, therefore, any values above 7F will be treated as 7F and will represent the maximum intensity for any color component. For example, the values 007F00 and 00FF00 will be treated as the same.

As an example, for internal LEDs, the following is a valid URL and will lit the LED with full intensity for the red component of the LED, half intensity for the green component of the LED and 0 intensity for the blue component of the LED, on a device that has an IP of 192.168.1.12 assigned to it.

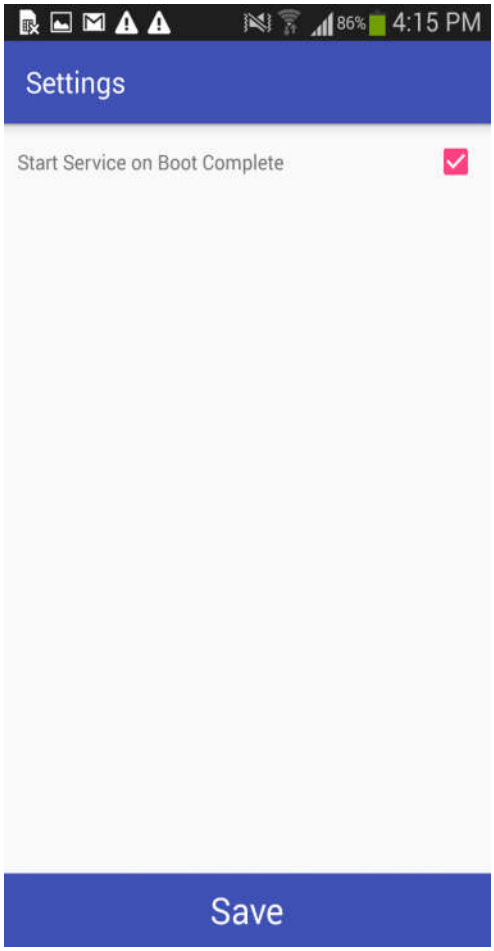
`http://192.168.1.12:8081/light?led=7f3f00`

In order to turn off the LED, the value for the `led` parameter needs to be passed as `000000`.

5. Enabling Auto-start

The app can be set to auto-start after the device is rebooted. This can be done by going to the settings screen from within the app and then checking the “Start Service on Boot Complete” option, if it is not already checked, and then clicking the “Save” button.

The app will then be automatically started in the background as a service, each time when the device is rebooted.



6. Overview

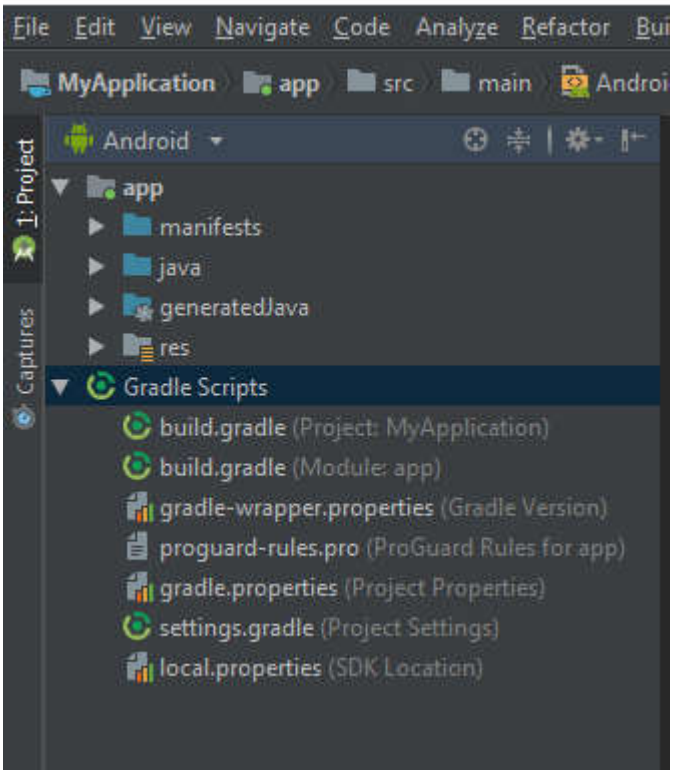
MIMO service is an app that runs on supported MIMO android tablet devices and allows controlling devices' internal LEDs or external RGB LED units by a library built into an application.

The library automatically detects whether external RGB LEDs are connected or not. If external RGB LED units are not connected then the device's internal LEDs will be controlled as usual, otherwise, the external RGB LED units will be controlled.

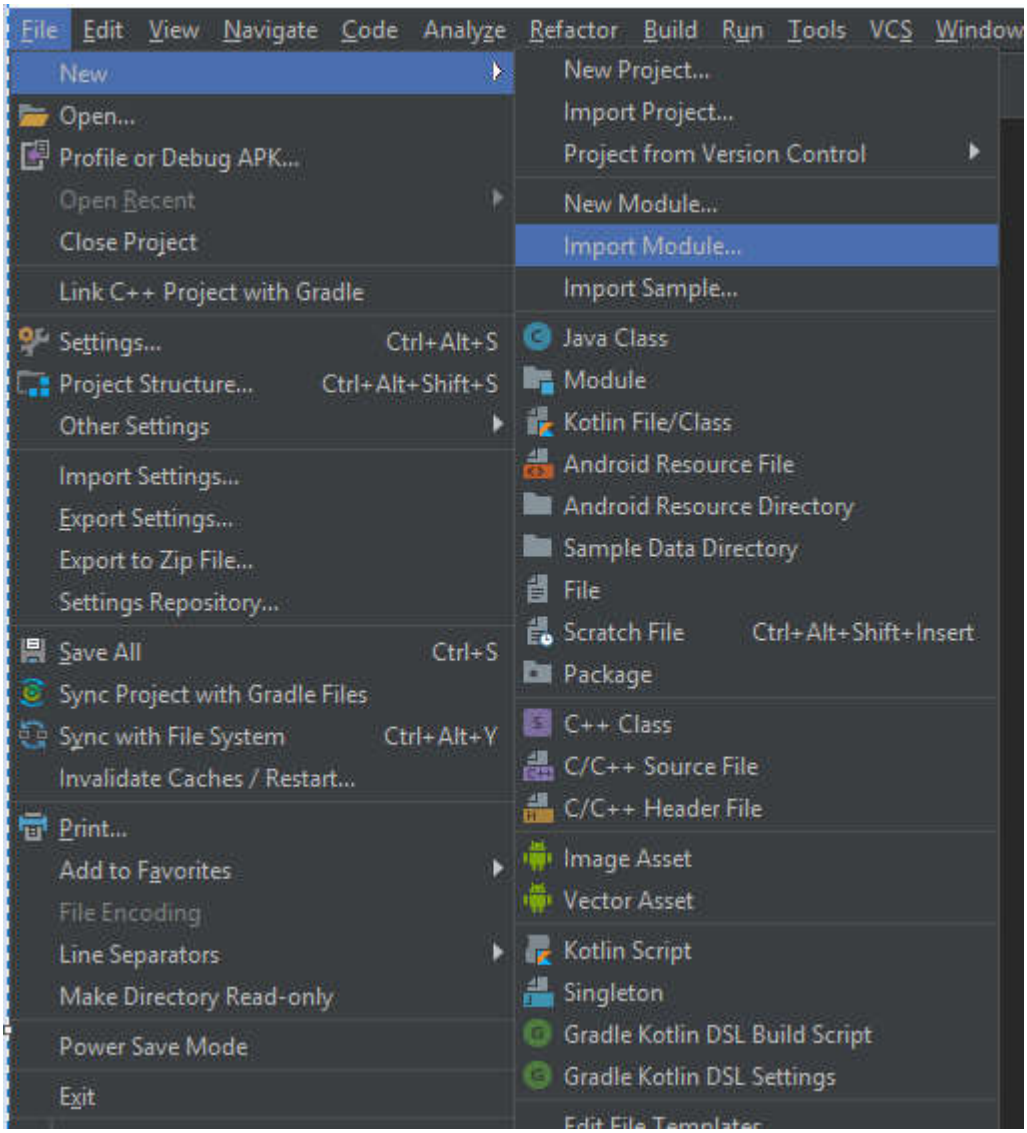
The library can support a maximum of three external RGB LED units connected to the left, right and/or bottom USB ports of the device.

7. Adding the library

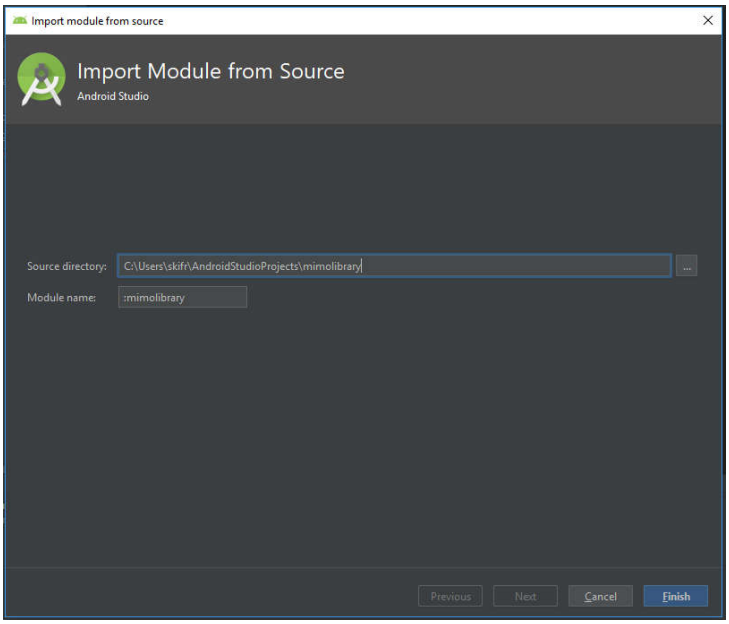
Add the mimolibrary to a folder location you know. Then open up your application in Android Studio.



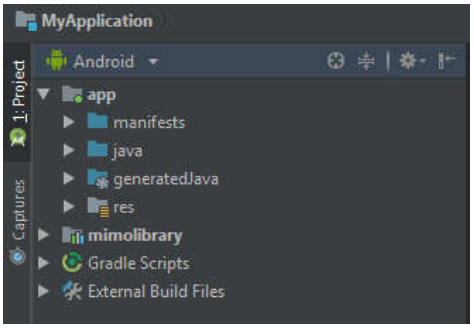
Highlight your app folder. Then go to File -> New -> Import Module

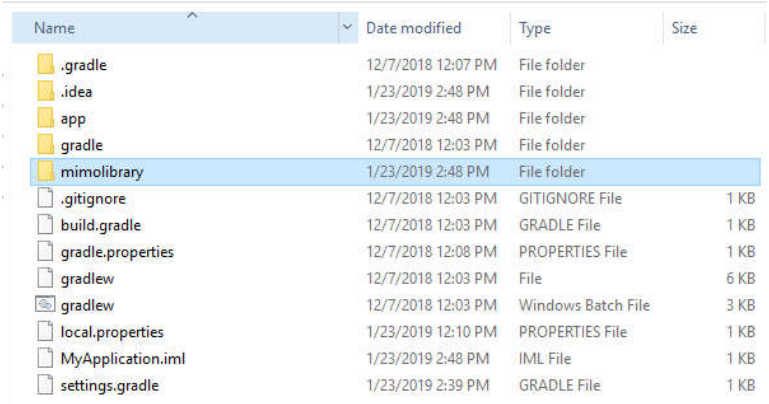


Select the destination that the mimolibrary folder is and import it into your project



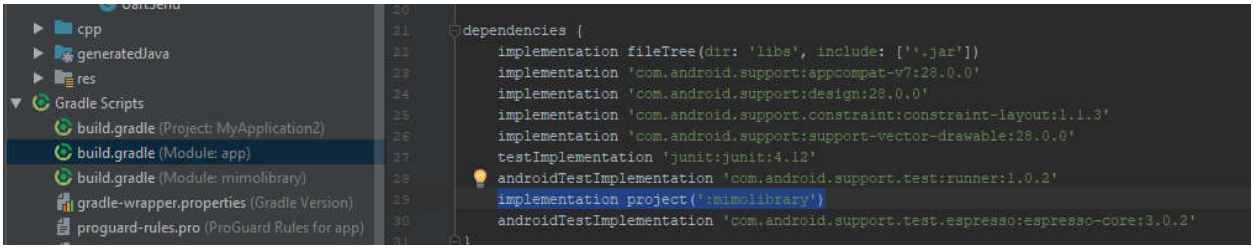
The folder will now be added to your project and can be viewed in the file explorer of your project.





8. Added Code

To use the mimolibrary, code needs to be inserted into the main application first. First add the following code the build.gradle(Module app): implementation project(':mimolibrary')



Afterwards Rebuild the project. Build -> Rebuild Project

Add the following code to the Java class that will control the LEDs:

import mimodemo.com.mimolibrary.MIMOService;

```
package com.example.skifr.myapplication3;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import mimodemo.com.mimolibrary.MIMOService;
```

9. Controlling the LEDs

To control the LEDs, create a variable for the MIMOService to be called from. In the example below, mimo is the variable used. Once created it can be used to call out the ledscontroller for which will then be used to make changes to the LEDs. The mimolibrary will automatically detect the LEDs and change them.

To send a color to the LEDs, use mimo.ledsController("######") Where # are the Hex values for the colors. The Mimo External LEDs are from 00 – 7F for RGB. Mimo Internal LEDs are 00 or FF for RG.

```
public class MainActivity extends AppCompatActivity {

    MIMOService mimo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        mimo=new MIMOService();
        mimo.ledsController( value: "FF0000");
    }
}
```