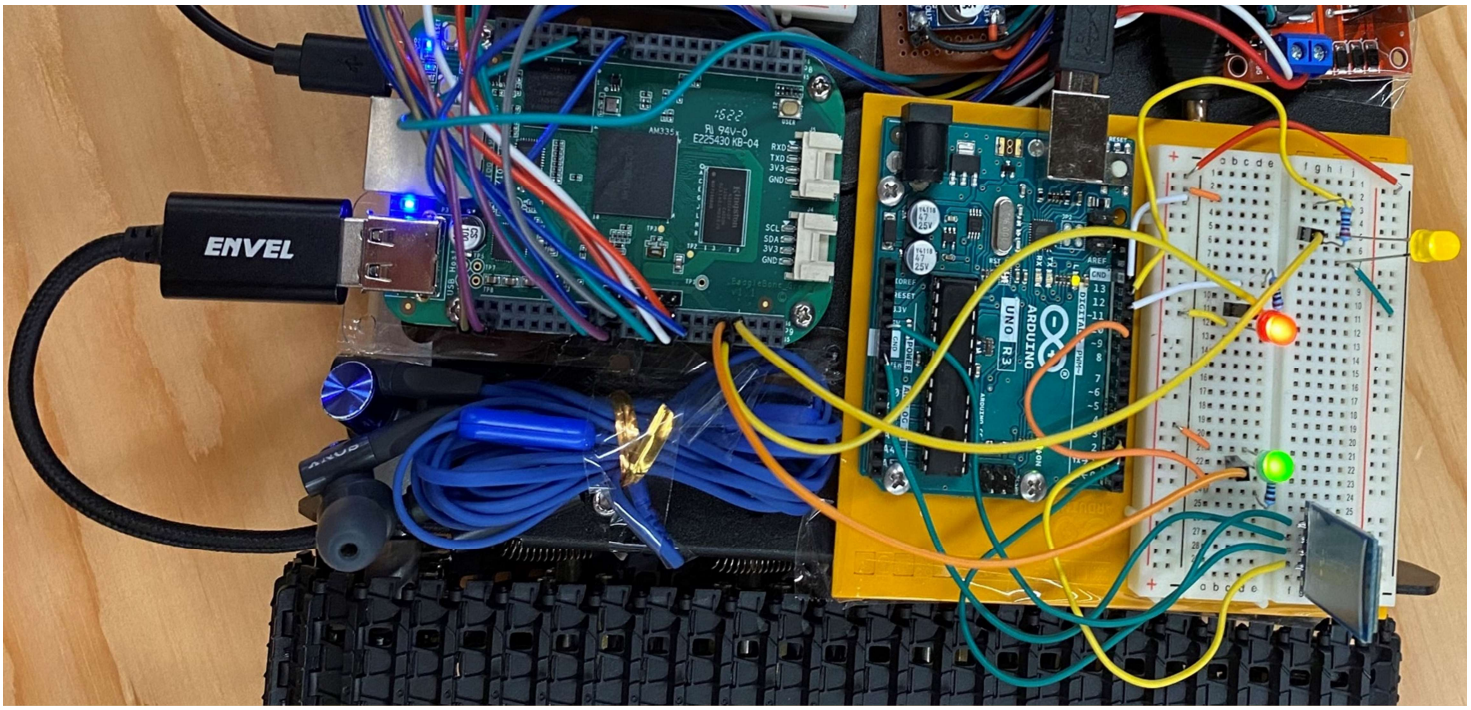
Android App to Bluetooth Connection guide



DECEMBER 7, 2022

ENSC 351

Android App to Bluetooth Connection guide

By Nasim Akbari, Ruth Desta, Mercygold Msaki, Aneet Kaur
Last update: Dec 7, 2022

This document guides the user through

1. Implementing the Android BLE Connection ---------- page 3
2. Sample Implementation (Arduino) ---------- page 6

1. Parts List

- An Android device
- Bluetooth 2.0 Slave Module HC-06 (https://leeselectronic.com/en/product/20201-bluetooth-slave-module.html)
- Breadboard
- LEDs
- 220-ohm Resistors
- Male to Male Jumpers

2. Software Lists

- Android Studio

3. Implementing the Android BLE Connection

1. Add the following permissions to your AndroidManifest.xml file

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

2. Add this line in your build.gradle (:app) to import a BLE permissions library
   implementation 'pub.devrel:easypermissions:3.0.0'
3. We are using the Bluetooth Adapter built into AndroidStudio so declare the Bluetooth Adapter as variable of Bluetooth Adapter type and call the built in functions to get the Default Adapter of the Android device. Implement the following init(), checkPermissions() and override onRequestPermissionsResult()

```
private void initBluetooth() {
    mBTAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBTAdapter == null) {
        Toast.makeText(this, "No Bluetooth found", Toast.LENGTH_SHORT).show();
    }
    if (mBTAdapter.isEnabled()) {
        Toast.makeText(this, "Bluetooth already enabled", Toast.LENGTH_SHORT).show();
    } else {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.PERMISSION_GRANTED) {
            checkPermissions();
            return;
        }
        mBTAdapter.enable();
    }
}

private void checkPermissions() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        if (!EasyPermissions.hasPermissions(this, BLUETOOTH_PERMISSIONS_S)) {
            EasyPermissions.requestPermissions(this, "message", LOCATION_PERMISSION_REQUEST, BLUETOOTH_PERMISSIONS_S);

        }
    }
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        requestPermissions(BLUETOOTH_PERMISSIONS_S, 1);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if (requestCode == LOCATION_PERMISSION_REQUEST) {
        if (!(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED)) {
            new AlertDialog.Builder(this)
                    .setCancelable(false)
                    .setMessage("Location permission is required for this app.")
                    .setPositiveButton("Grant", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            checkPermissions();
                        }
                    })
                    .setNegativeButton("Deny", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            MainActivity.this.finish();
                        }
                    }).show();
        }
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

```
private BluetoothAdapter bluetoothAdapter;
private final int LOCATION_PERMISSION_REQUEST = 101;
public static final String[] BLUETOOTH_PERMISSIONS_S =
{Manifest.permission.BLUETOOTH_SCAN, Manifest.permission.BLUETOOTH_CONNECT,
Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION};
```

4. To get all the BLE devices on your network, add this code to your Activity.
   Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();
5. Initialize your Bluetooth Device address for your BLE Module and create a socket device to connect your BLE module to your phone
   BluetoothDevice HC06BluetoothModule = bluetoothAdapter.getRemoteDevice(*your module address*);
   private UUID deviceUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
   //universal UUID
   BluetoothSocket bluetoothSocket = NULL;
   bluetoothSocket = HC068BluetoothModule.createRfcommSocketToServiceRecord(deviceUUID);
   bluetoothSocket.connect();
6. To send data from the app, use the following lines (example of sending data as byte array, make sure to add newline!)
   byte[] data = (“*your data”* + “\r\n”).getBytes();
   bluetoothSocket.getOutputStream().write(data);
7. When you’re done and closing your app/activity (and ONLY then!), add this line to close the BluetoothSocket you created.
   bluetoothSocket.close();

Troubleshooting

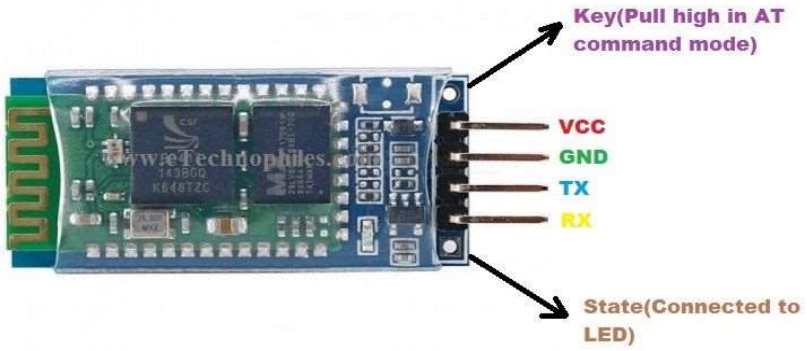
If the delay between your app sending data and the Arduino receiving it is too high, check that you are adding a new line (“\r\n”) at the end and that should fix it!

There are recent updates in AndroidStudio that give errors for checking permissions. If you do, call the necessary overrides and just add the checkPermissions() function provided above.

If you get any errors beginning with “Missing permissions required by BluetoothAdapter”, make sure you add the required permissions in your Manifest file as provided on step 1 of this section of the guide.

Bluetooth Module

1. Supply Voltage for module: +3.3 VDC 50mA
2. The Structure of the pins are shown below:



Bluetooth 2.0 Slave Module HC-06

PIN #	PIN Name	Description
1	VCC	Connect to a 3.3V pin of the microcontroller to power the module
2	GND	Connect it to the common ground pin of the circuit
3	TXD	Connect to the RXD pin of the Microcontroller. Transmits Serial data (wireless signals received by the Bluetooth module are converted + transmitted out serially on this pin)
4	RXD	Connect to the TXD pin of the Microcontroller. Module receives the data from this pin and then transmits it wirelessly.

Recommendation: We recommend HC-05 since it can be used to initiate connection or accept connection while HC-06 only accepts connection (cannot initiate).

4. Sample Implementation (Arduino)

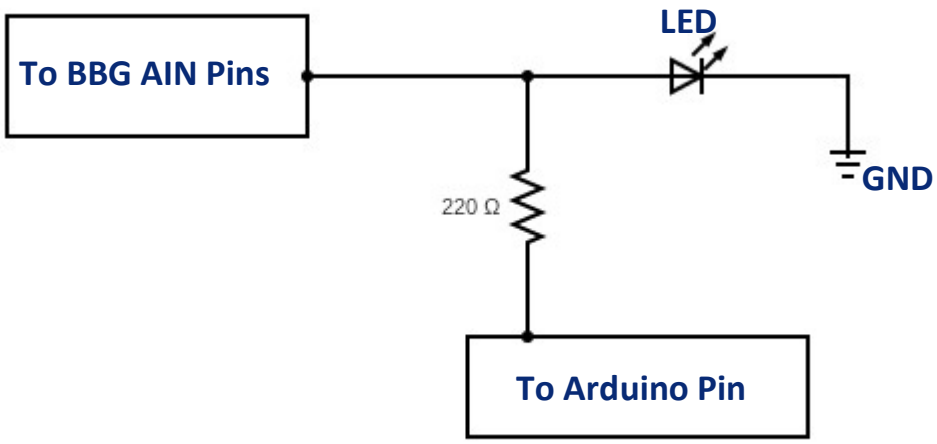
Note: We tested our app using an Arduino set up using LEDs. We had our app send specific numbers that made the LEDs light up in a certain way. We will explain the wiring and code below in detail but feel free to use a different implementation to test according to what your project is.

1. Wiring

Bluetooth Module To Arduino Uno

Arduino Uno	Bluetooth Module (HC-06)
5V pin	VCC
GND pin	GND
Pin #2	TXD
Pin #3	RXD

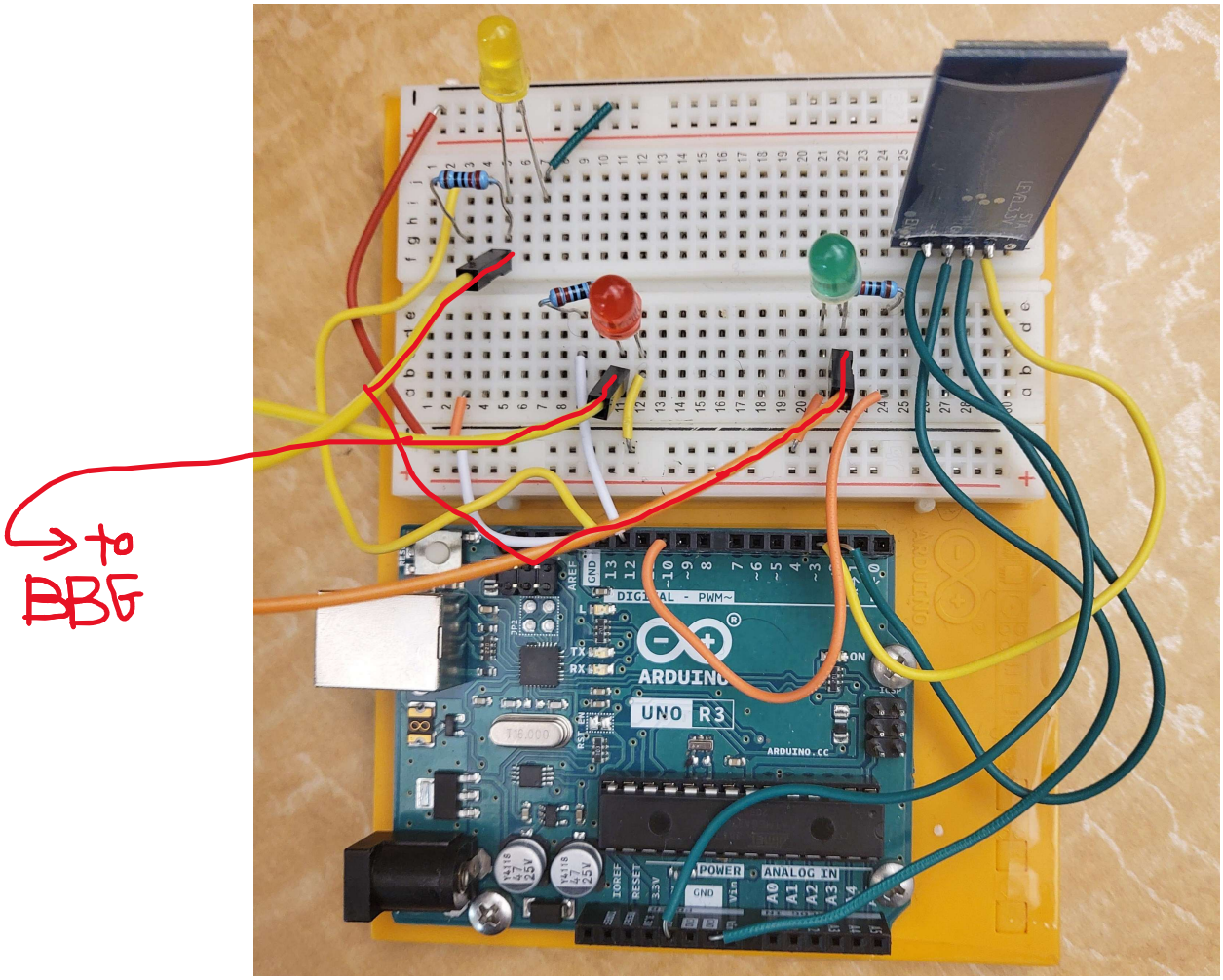
[One LED Connection]



The connectionn between the Arduino and LED is shown above. Make sure that your resistors are in series and the negative pin (shorter pin) is set to ground. We have a beaglebone connection we will explain later (DO NOT ADD NOW).

Using the diagram above, we have wired three LEDs using the configuration shown below

Connection: Arduino → Resistor → LED → GND



Note: highlighted wires went to the BeagleBone. DO NOT CONNECT YET

The table below will show the connections used in our implementation of the Arduino Code to test our bluetooth connection. Feel free to change the pins on your Arduino accordingly. However, don't forget to update your code as well!

Arduino Uno	LED
Pin #13	Yellow
Pin #12	Red
Pin #10	Green

2. Code: Implement the code below in the Arduino software to test BLE Module

[*Code was given in the zip file submitted for the guides. Able to copy and paste with ease.***]**

Troubleshooting

If the code seems like it's not updating make sure you save, compile and upload (in this order!) your changes onto your Arduino Uno.

If you keep getting errors that are not related to your code try unplugging and plugging the Arduino back in before you compile and upload your code.

If the changes are not being implemented after you've uploaded, ask yourself, have you compiled it?

3. Arduino To BeagleBoneGreen

DISCLAIMER: Please DO NOT attempt if you're not comfortable with circuits. There is a HIGH possibility of burning your board. The Arduino outputs a 5V signal and the BBG can only receive a max of 3.3v. We will highlight a method that worked for us, but it has not been extensively tested so proceed at your own risk.

NOTE: We used the BBG AIN pins to read voltage levels from Arduino directly by using 220ohm resistors, but this is not the safest way to do, so you can even use a MOSFET to protect your BeagleBone. Also, you can use higher resistor and measure voltage of each branch, which is going to be connected to the BeagleBone, to make sure that it is less than 3v. TEST, TEST, TEST BEFORE CONNECTING TO THE BEAGLEBONE! You need to measure all the voltages across each resistor to make sure that it is less than 3V after wiring up all the LEDs to make sure that there is no possibility of burning your BealgeBone.

Connection to BBG shown in picture above

BBG	LED
Pin AIN1	Yellow
Pin AIN2	Red
Pin AIN3	Green