

## Bluetooth Low Energy Product Development Kit

### *1) Abstract*

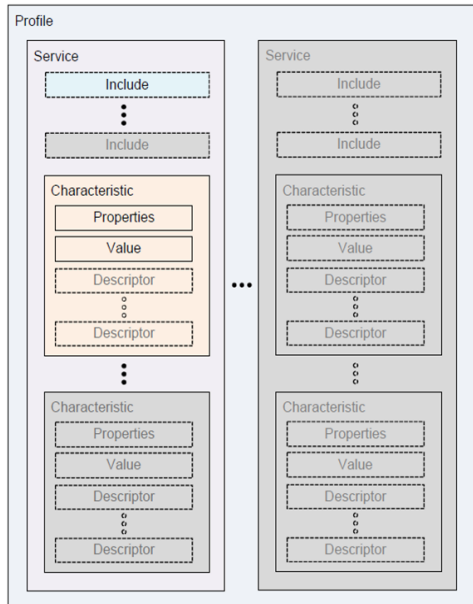
This project is a development kit to allow an engineer to quickly add a Bluetooth Low Energy interface to a product. The kit uses an Android application to establish a connection with and manipulate an external Bluetooth Low Energy Module.

### *2) Introduction*

With the rapid development of embedded software and hardware, new technologies, especially mobile devices, are increasingly expected to have the ability to interface with a wide variety of technologies. For a company designing a product, adding these functionalities takes time and delays the product's arrival on the marketplace. This means either your competitor could beat you to the marketplace or you simply must wait longer to profit from your product. The following project is a development kit to add a Bluetooth Low Energy interface to a product. This kit will attach the interface quickly and with minimal configuration, and it will get your product to the marketplace quicker.

### *3) Background*

Bluetooth Low Energy (BLE), or Bluetooth Smart, is a Bluetooth technology first released by the Bluetooth Special Interest group in 2010. Bluetooth Smart offers considerably lower power consumption and cost while maintaining a similar communication range as standard Bluetooth. This technology then finds a niche in embedded systems, as it is very appealing to a developer that needs his module to last up to years on something of little power, like that of a button cell. Bluetooth Smart establishes a framework for data transported and stored via BLE connections with the Generic Attribute Profile (GATT). GATT defines two roles: a server and a client. The server will broadcast its available services to the client, and each of these services will have characteristics associated with them. If a client wishes to read or write to the server, it must first discover the services that the server offers, and then it must extract the characteristics associated with that service.

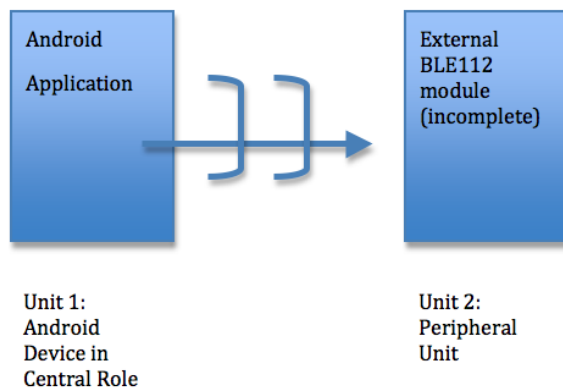


**Figure : GATT Profile**

These characteristics will have values that are capable of being manipulated. However, the current Android SDK does not support profile level abstraction, so it requires that the program provide a unique identifier, or a UUID, for each of the characteristics that you wish to manipulate on the system. There are many industry standard characteristic UUIDs (such as for a heartbeat monitor), but often they are custom defined in the application code for the BLE enabled module.

#### 4) Implementation

The project runs an Android application on a mobile device, and this Android application establishes a Bluetooth Low Energy (BLE) connection with a peripheral device. The Bluetooth Low Energy stack implements a central-peripheral model of communication, very similar to the usual client-server model. The Android device in the central (or client) role will scan for Bluetooth Low Energy advertisements, and the Bluetooth module on the peripheral unit will advertise itself as a Bluetooth device. The Android device will list any nearby BLE devices, and the user will pick which device to add as a peripheral to the Android application.



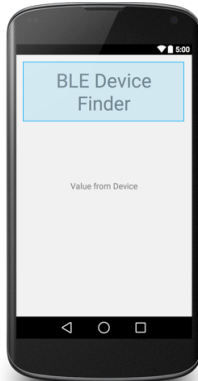
**Figure : Block Diagram**

In the Android application, the main activity first starts by establishing the main page layout. On the boot up, it also uses Android APIs to extract the Bluetooth Adapter, which can be used to run various Bluetooth operations. The program then continues to check the mobile device for three things: that it can run Bluetooth, that it has Bluetooth enabled, and that it has Bluetooth Smart capabilities. The program will then wait for the user to click the scan button on the top of the options menu. If it is clicked, the application will automatically call the `onOptionsMenuSelected` callback method. This callback method will then start a scan for available BLE devices within range. This is accomplished by calling the `startLeScan` method. Any Bluetooth GATT operation is handled in background threads, and if information needs to be sent to the main thread of operation, then it must be done through a handler. A handler then tells this `startLeScan` method that it should only scan for 5 seconds, in order to make sure it does not scan indefinitely. Once the `startLeScan` method is called, it automatically triggers the `onLeScan` callback method. This method simply just redraws the options menu to include the discovered BLE enabled devices.

The program then remains idle until the user picks a device to connect to from the list of available BLE devices. The program then calls the `connectGatt` method, which finally establishes the GATT Bluetooth LE connection with the external BLE module. This `connectGatt` method takes a `BLEcallback` as one of its arguments, which is where the program will automatically go next. In addition, the handler is used to send information back to the UI about the connection progress. If the connection is successfully established, the program automatically enters the `onConnectionStateChange` method inside of the previously defined `BLEcallback` in the `connectGatt` call, where it begins to discover the services associated with the selected BLE device. The services that are discovered are sent up the UI via the handler. The discovery of services triggers the `onServicesDiscovered` callback, where the value of the characteristics associated with these services are read and sent up to the UI via the handler. When the initial value for the characteristic is read, it triggers the `onCharacteristicRead` callback, where the program enables notifications from this characteristic. This means that the program will continually update the value being read from the BLE module, and it will also continue to use the handler to send this value up to the UI. The program will then repeatedly call the `onCharacteristicChanged` callback every time it receives a notification from the BLE device.

### *5) Experiments and Results*

Developing mobile apps always comes with much overhead, as one must write and configure all of the layout, manifest and style files in addition to the java code. I then began this project by first just building a simple UI. I was able to view the UI via the preview tool built into Android studio. Due to the integrated tools in Android Studio, it took me about a week to learn Android Studio and put together all of the layout, style, and manifest files for my program.



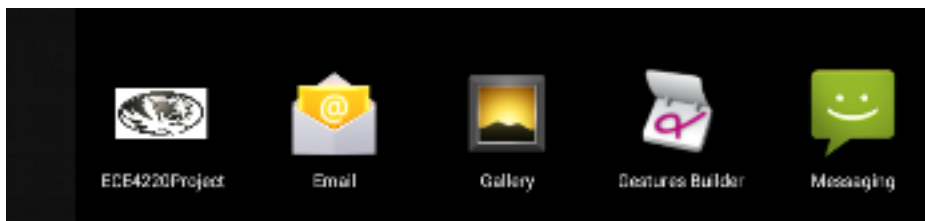
**Figure : My Application's UI**

After I got my UI to my desired look, I began to code the main Android activity. Unfortunately, there is not a lot of documentation on the Android Bluetooth Low Energy API, assumedly because it is such a new technology. Thus, there was a large amount of trial and error in debugging my code, and this significantly slowed down my development time. This is then part of the project that took several weeks.

For the external BLE module, I ordered the BLE112 BlueGiga chip. The specs for this module indicated that it had a USB host interface, and that I could thus upload my own application code for it. However, once I ordered the device, I discovered that I needed more hardware to do anything with it. I found that my only solution was to purchase a BLE development kit that would include all of the necessary supporting hardware to upload my application code to this module. However, this development kit was way out of my price range (\$100+), and also I did not discover the inadequacy of the hardware that I did order until about a week before the project was due. This was because I had begun my project with the Android application (as indicated by my timeline on my proposal), and this Android application took much longer to develop than anticipated.

#### *6) Discussion and Conclusions*

Overall, this project is still incomplete. I did not have an Android mobile device to test my Android program on, so I set up an emulator to simulate a Nexus 7 tablet. The Android program is able to compile and load successfully onto an Android emulator, but it gets a common error when I attempt to run it.



**Figure : ECE4220Project Loaded onto Android Device**

This is a very common error for running an application on an emulator in Android Studio, and it does not seem to have a clear fix.

```
HAX is working and emulator runs in fast virt mode  
setWindowSurfaceColorBuffer: bad color buffer handle 0
```

**Figure : Emulator Error Message**

I believe it may be because the emulator does not have BLE capabilities, but the error is so broad that it is difficult to tell. In the future, I hope to acquire an actual Android device with BLE capabilities in which to run my application. However, I still gained valuable experience with both Android programming and the new Bluetooth Low Energy technology. I had always wanted to work with Bluetooth, and this project gave me a nice medium for accomplishing that task. Though it was a little daunting at first, it was not too hard to quickly learn how data is exchanged via Bluetooth Low Energy. The implementation of the GATT profile made it relatively intuitive, and I hope to find a future project that I could work with Bluetooth Low Energy. However, I may migrate over to the iOS implementation of BLE in the future. Working with Android and Android Studio was a little frustrating to me. There is always a learning curve when working with a new IDE, but it seemed oddly difficult to find my around Android Studio when compared to a more commonly used IDE like Eclipse. In addition, I often found that modifying code somewhere in an Android program could produce a seemingly unrelated error in a totally different spot in the program. Thus, if I could acquire a license, I would like to extend this project over to be iOS compatible in the future.

Once completed, I believe this project could be a valuable tool for modern developers. Bluetooth Low Energy is a very exciting and modern technology. Many products could benefit from adding a potential Bluetooth interface, and this kit could allow a company or developer to quickly add this interface without losing much time in development. This means getting the product to the market faster, which means both increased profit and increased innovation for the company or developer.