TABLE OF CONTENTS

Android App

The Android app is a heavily modified version of the Bluetooth low energy sample app provided on the Android developers website. Currently, it connects to an available Gatt client and sends notifications at a sample rate determined by the user of the app. By default, the sample rate is 50 Hz, but can be set to be anywhere from 5 – 90 Hz.

To save data, the app automatically caches all data it receives into an array fixed at 65,000 in size. At 90 Hz, this results in the array getting full after about 12 minutes. After these 12 minutes elapse, a text file is created with all the data received so far. Each consecutive duodecimal time period has the data appended to the original text file created, so only one file contains all required data.

The app is also capable of instructing the hardware to perform frequency sweeps. The speed of the frequency sweep is entirely dependent on the sample rate chosen as the frequency is incremented at precisely the same rate as the sample rate. So at a frequency of 50 Hz, the frequency is incremented every 20 milliseconds, and etcetera.

The following sections cover each class in the app. This includes all services, activities, fragments and helper classes.

Services

The app has 2 services in total: BluetoothLeService and ServiceBinder.

BluetoothLeService

This service is the crux of the entire app. It is responsible for the entire BLE connection. It connects, maintains the BLE connection and performs all necessary requests on BLE characteristics, be it writing to, or reading from, notifying or indicating on a characteristic. Therefore, it parses any information coming in to the connection, before handing the data off to whatever app component needs it.

Android provides 2 kinds of services depending on the methods they are invoked with, which also determines their lifecycle. This service in particular is a **bound service**, meaning that it needs another android component to be bound to at all times for it to function as intended. Without a component to bind to, it can be destroyed anytime at the Android's system's discretion. When the connection is first established, the bound component is the activity DeviceControlActivity. For long-term connections for continuous monitoring however, the bound component is the **started service** ServiceBinder. Please refer to the source code for detailed descriptions of the methods used and what they do.

ServiceBinder

As mentioned earlier, this service exists to maintain long-term connections to the BLE client. Since the method used to create an instance of this service is the StartService(intent) command, the Android system has no authority to kill it except in dire situations such as when memory is constrained. Therefore, the connection can be maintained indefinitely in this state. To stop the service, the user of the app must expressly call StopService(intent).

Due to this service's function, it shares many identical methods to that of DeviceControlActivity, however, since it is a service and provides no UI to the user, these methods must be called from an activity. This activity is aptly named LongTerm and will be discussed in the following sections.

Activities

The app has 6 activities. They are:

1. MainActivity
2. ActAndPair
3. Scan:
4. DeviceControlActivity
5. LongTerm
6. ButtonNo

MainActivity

This is the app's splash screen.

ActAndPair

This requests to turn on Bluetooth and start scanning for BLE devices.

Scan

Upon successfully turning on Bluetooth, this activity scans for and displays BLE devices found.

DeviceControlActivity

The first activity component BluetoothLeService binds to. It is launched when an item in the list provided by Scan is tapped. When launched, it automatically loops through all the available GATT services and characteristics made available by the GATT client.

Of note are the sample rate, AC frequency parameters and bio-impedance data characteristics. The activity immediately reads from both the sample rate and AC frequency parameter characteristics to get a complete picture of the GATT client's current state. Information like the current sample rate, start frequency, frequency step size and number of frequency increments are cached, provided to the user of the app and propagated to all app components that may need it.

This activity also provides the methods to set the sample rate, frequency sweep parameters, begin the acquisition of data and exportation of data to a text file at any

time during the connection, independent of the 12 minute automatic exporting discussed earlier.

Should a user hit the button to start sampling, this activity also automatically starts the ServiceBinder activity, so if the user were to leave the app, the BLE connection is maintained and data acquisition remains ongoing. The activity also lets the user turn on and off BLE notifications allowing him / her to adjust settings as required during set up, clear the saved text file data and begin sampling again.

This activity also displays a live graph of the data, a summary table of the current state of the GATT client, as well as the raw figures just in case.

LongTerm

This activity acts as a host for a single fragment (BioimpFragment) that can have multiple states. The fragment's states will all have different functions, but as of right now only two states are fully operational. The proposed list of states is:

1. Live Data
2. Past Hour
3. Past 12 Hours
4. Export Data
5. Back to Scanning

Live data (fully operational): This state is identical to DeviceControlActivity in terms of function and form. It hosts identical methods and provide identical functionality, with the exception of he ability to turn on and off notifications. It is assumed that DeviceControlActivity was already used to set up the device, and therefore replicating the functionality here is redundant. It also presents a live graph of data, summary table of the GATT client but lacks raw figures.

Past Hour (Incomplete): Intended to read past data and provide a cohesive summary of key details from said data.

Past 12 hours (incomplete): Same as above, but for 12 hours.

Export Data (incomplete): Currently app data is stored in phone external storage (SD card); ultimately it should be stored in internal storage. This fragment should provide options to move from internal storage to external storage.

Back to scanning (complete): Terminates the current connection, and boots the user back to the scanning activity which then starts scanning once more.

ButtonNo

If a user decides to hit the no button on the splash screen, this activity provides a sad face and says: “That’s no fun”.

Fragments

The fragments as the name implies, are pieces of the application that do not stand alone, but are hosted within an activity to supplement the activity. They also have their own lifecycles, so they can be created and destroyed independent of their hosting activities. These attributes make them valuable as reusable app components, and are used to perform repetitive short-term tasks.

There are a total of 4 fragments utilized globally in the app, they are:

1. BioimpFragment
2. NameTextFileFragment
3. SampleRateFragment
4. FrequencySweepFragment

A fifth fragment, NavigationDrawerFragment exists, but it is used exclusively by the LongTerm activity to switch the states of BioimpFragment.

BioimpFragment

As discussed under the LongTerm activity, this fragment is intended to perform 5 functions, switching states programmatically as necessary. For details on these states, please refer back to the discussion on LongTerm Activity.

NameTextFileFragment

Used both in DeviceControlActivity and LongTerm activities, this fragment provides a text field where the app user can name the text file to be next exported.

SampleRateFragment

Used both in DeviceControlActivity and LongTerm activities, this fragment provides a UI where the app user can change the sample rate of the GATT client.

FrequencySweepFragment

Used both in DeviceControlActivity and LongTerm activities, this fragment provides a UI where the app user can change the frequency sweep parameters of the GATT client. Things that can be changed include whether the frequency sweep is on or not, the start frequency, the size of frequency increments and the number of increments.

Helper Classes

As the name implies, these classes don't provide any UI or even do any background work. They exist solely to separate and trim otherwise bulky code / classes. This application contains only one: SampleGattAttributes.

SampleGattAttributes

This provides a lookup table for DeviceControlActivity to store string hashmaps of GATT service or characteristic human readable names, and their respective 16 or 128 bit universally unique identifiers (UUIDS).