***Internet of Things Course***

**LAB 5**

**BLE Heart Rate Sensor App**

**Issue 1.0**

**Contents**

[1 Introduction 3](#_Toc36653016)

[Lab Overview 3](#_Toc36653017)

[2 Requirements 3](#_Toc36653018)

[2.1 Hardware 3](#_Toc36653019)

[2.2 Software 3](#_Toc36653020)

[3 User Interface 4](#_Toc36653021)

[4 Bluetooth Connection 4](#_Toc36653022)

[4.1 Enabling Bluetooth 4](#_Toc36653023)

[4.2 Finding the Heart Rate Sensor 5](#_Toc36653024)

[4.3 Receiving Sensed Heart Rate 6](#_Toc36653025)

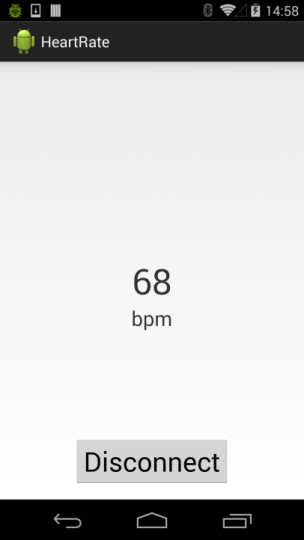
[4.4 Displaying the Heart Rate 7](#_Toc36653026)

[5 References 7](#_Toc36653027)

# Introduction

## Lab Overview

In this exercise, you will design and build an app to connect to the heart rate sensor developed in the previous lab, using Bluetooth Low Energy (BLE), and display the information on-screen as shown in Figure 1.



*Figure 1. Finished app design.*

# Requirements

## Hardware

* Android device for debugging. It should be possible to complete this lab with any Android smartphone, as long as it supports Bluetooth Low Energy and Android 4.3 (API Level 18) or above. It is strongly recommended that you update to the most recent version of Android to ensure compatibility. The lab has been tested with Android 8.
* Micro USB cable type B to type A for attaching the mobile to the PC.
* BLE heart rate monitor. Any monitor should be supported, and you can also use the embedded platform developed in the previous lab to achieve the same functionality.

## Software

* A 64-bit version of Java SE Development Kit (JDK) is required by Android Studio for Java compilation. This can be downloaded free of charge from Oracle’s website.

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

* Android Studio for app development. This is the Android development environment, which includes the Android SDK Tools and an emulator. This can be downloaded for free on the Android Developer Tools page.

<https://developer.android.com/sdk/installing/studio.html>

* USB drivers. These are required for the Android Debug Bridge (adb) to detect your device. The installation guide for these drivers can be found in the simple app development lab notes. They can also be found online on the Android Developer Tools page.

<http://developer.android.com/sdk/win-usb.html>

# User Interface

The user interface is similar to the basic app we created in a previous lab, except there is an added button at the bottom for connecting/disconnecting from the heart rate sensor. The main difference is the addition of a selection dialog, which is displayed when the connect button is pressed. This is known as an AlertDialog. Documentation on creating an AlertDialog from scratch can be found in the Android Developers API Guides section. In our project, the alert is designed to display Bluetooth devices, so a custom list adapter has been made to store BluetoothDevice objects.

# Bluetooth Connection

## Enabling Bluetooth

Before we can connect to our device, we need to discover it using a BLE scan. The Android API has provided a BluetoothAdapter class for this purpose (see android.bluetooth.BluetoothManager). To receive an instance of this class, use the following snippet:

BluetoothManager bluetoothManager= (BluetoothManager)getSystemService(Context.BLUETOOTH\_SERVICE);

mBluetoothAdapter = bluetoothManager.getAdapter();

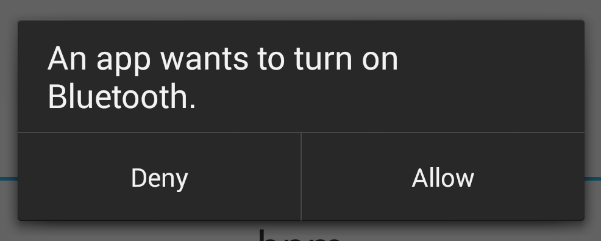
The OS needs to be informed we require permission to access various Bluetooth functions, otherwise a java.lang.SecurityException will be thrown when these functions are called. The following permissions are required, and must be added to the AndroidManifest.xml for the rest of the code to work:

android.permission.BLUETOOTH

android.permission.BLUETOOTH\_ADMIN

android.hardware.bluetooth\_le  
android.permission.ACCESS\_FINE\_LOCATION  
android.permission.ACCESS\_COARSE\_LOCATION

Finally, we are ready to check if Bluetooth is on, and otherwise request to turn it on. It is possible to check if Bluetooth is enabled with the BluetoothAdapter.isEnabled() function. If this returns false, we can fire an Intent with action BluetoothAdapter.ACTION\_REQUEST\_ENABLE, which will trigger the prompt shown in Figure 2.



*Figure 2. Request to turn on Bluetooth.*

## Finding the Heart Rate Sensor

Taking a look at the documentation, we find these functions inside the BluetoothAdapter:

boolean startScan(BluetoothAdapter.LeScanCallback callback)

Starts a scan for Bluetooth LE devices.

void stopScan(BluetoothAdapter.LeScanCallback callback)

Stops an ongoing Bluetooth LE device scan.

We call startScan and pass a callback that adds a BLE device to DevicesAdapter.

BluetoothAdapter.LeScanCallback mScanCallback = (device, i, bytes) -> {

runOnUiThread(() -> {

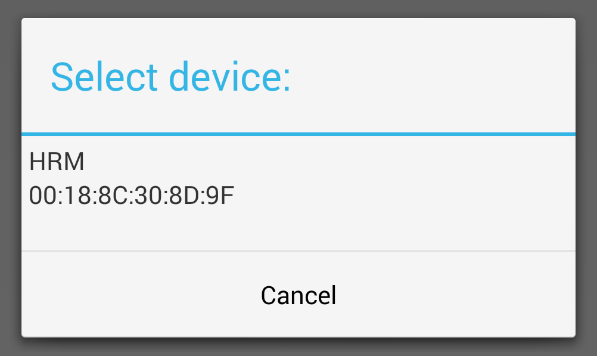
mDevicesAdapter.add(device);

mDevicesAdapter.notifyDataSetChanged();

});

};

Since the DevicesAdapter is bound to a ListView (inside the AlertDialog), we have populated our list of nearby BLE devices as in Figure 3.



*Figure 3.* *List of nearby BLE devices*

## Receiving Sensed Heart Rate

The device has been identified, and now we can discover which services it is broadcasting. This is accomplished with the BluetoothGatt.discoverServices function. Our BluetoothGattCallback.onServicesDiscovered function will be executed when this is completed. It is then possible to find the *Heart Rate* service with BluetoothGatt.getService, and from the service we can find the *Heart Rate Measurement* characteristic (see Figure 4 for a list of services and characteristics) using BluetoothGattService.getCharacteristic.

*Figure 4. Available heart rate monitor services and characteristics.*

Taking a look at the *Heart Rate Measurement* characteristic in detail, we can see further information about its nature. By calling BluetoothGattCharacteristic.getProperties on our instance we see that it supports notification (PROPERTY\_NOTIFY). Along with this, we find a descriptor for the Client Characteristic Configuration [1], which we must modify if we want to receive notifications from the device (set to ENABLE\_NOTIFICATION\_VALUE). Thus, the complete code for subscribing to the notifications is as follows, assuming mGatt is our BluetoothGatt instance:

private void subscribeToNotifications(BluetoothGattCharacteristic characteristic) {

// Enable notifications.

mGatt.setCharacteristicNotification(characteristic, true);

// Tell the HRM we want to receive notifications.

UUID ccc = AssignedNumber.getBleUuid("Client Characteristic Configuration");

BluetoothGattDescriptor descriptor = characteristic.getDescriptor(ccc);

descriptor.setValue(BluetoothGattDescriptor.ENABLE\_NOTIFICATION\_VALUE);

mGatt.writeDescriptor(descriptor);

}

Write the code in the BluetoothGattCallback.onServicesDiscovered function to subscribe to the heart rate measurement characteristic. You can use the UUID of the service to identify it, and the subscribeToNotifications function provided above to subscribe to the notifications of the characteristic broadcasts.

## Displaying the Heart Rate

Now whenever the heart rate sensor broadcasts a notification with an updated heart rate reading, our BluetoothGattCallback.onCharacteristicChanged function will be called. In order to read the actual data, we need to parse the data according to the Heart Rate Measurement specification [2]. We can then update the TextView on our UI to display the updated heart rate. Write a code to this effect. The psuedocode has been provided below.

if changed characteristic is the Heart Rate Measurement then

Check the heart rate format bit in the flags to determine the type

Read the heart rate measurement value from the characteristic

Log the new heart rate

On the UI thread:

Find and update the TextView with the new heart rate.

end

# References

|  |  |
| --- | --- |
| [1] | “Client Characteristic Configuration Descriptor,” Bluetooth, [Online]. Available at: https://developer.bluetooth.org/gatt/descriptors/Pages/DescriptorViewer.aspx?u=org.bluetooth.descriptor.gatt.client\_characteristic\_configuration.xml. |
| [2] | “Heart Rate Measurement Characteristic,” Bluetooth, [Online]. Available at: https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicViewer.aspx?u=org.bluetooth.characteristic.heart\_rate\_measurement.xml. |