SLEEP-LAB SLEEP MONITORING APPLICATION

By

Owen Flannery

Supervisor(s): Mr. Stephan Sheridan

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
B.SC. IN COMPUTING
AT
INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN
DUBLIN, IRELAND
3RD MAY 2018

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of **B.Sc. in Computing** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfillment of the requirements of that stated above.

Dated: 3rd may 2018

Author: Owen Flannery

Abstract

This paper is a talks about the process of making a android application to monitor a users sleep using a bluetooth connected heart-rate device and analysing the received information, some of the technologies involved in making a Firebase connected android application, the following chapters are a description my attempt to create publishable application that i could also use on my Curriculum vitae.

Acknowledgements

I would like to thank Stephen Sheridan my project supervisor for all his help during this project and to Sinead Flannery for being part of the test phase.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 Introduction

The field of study for this project are vast and cover a large array of technologies from android development to medical research, This means that a lot of the project is new to me as i start this project. Some previous work i would recommend for reading before this paper would be.

- Sleep Analysis for Wearable Devices Applying Autoregressive Parametric Models By M. O. Mendez, O. P. Villantieri, A. M. Bianchi, S. Cerutti.
- Non-invasive sensors based human state in nightlong sleep analysis for home-care. By M. Smolen, K. Czopek, P. Augustyniak.
- An unobtrusive sleep monitoring system for the human sleep behaviour understanding. By P. Barsocchi, M. Bianchini, A. Crivello, D. L. Rosa, F. Palumbo, F. Scarselli.

These papers are a good introduction to the problems and technologies involved in sleep analysis without being to hard to read.

1.2 background

A large amount of work has been done in the area of sleep analysis using at the medical level but there is not a large about of information on heart-rate alone all the information is taken using medical grade devices that are out of reach for most people, The studies done on sleep analysis using heart-rate and other metrics all indicated that a lower heart-rate is better but a heart-rate can also get to low for normal bodily functions and at this point the body will force the person awake or causing health problems that require more delicate equipment and and wont be discussed in this paper.

The area of Bluetooth and android is an ever changing environment as the android operating system is growing so fast and is constantly updating to include new standards and systems, while there is learning resources available for connecting android devices through Bluetooth the amount of android to third party sensor resources is almost zero.

Chapter 2

Literature Review

2.1 Abstract

There is a large amount of papers talking about sleep analysis using hospital equipment and other papers describing smartphone inbuilt sensors to analyse but there is not many that talk about the crossover of the two genres this review tries to create a bridge between the two without being too confusing for a person who has never studied the subject before.

2.2 Introduction

This literature review will be concentrating on the area of heart-rate and sleep analysis, the papers that will be discussed are papers conserving previous work on sleep analysis done in the last ten years and heart-rate analysis also in the same time frame. Other work in this paper will use more specialized equipment that are not available too small institutes.

2.3 Literature

2.3.1 How to gather data?

One of the problems associated with monitoring a users sleep will be how to gather data for a reliable source of usable data. After this literature review is finished we should have a

much better understanding of the problem and possible solutions, a solution described in An unobtrusive sleep monitoring system for the human sleep behaviour understanding (from references Barsocchi et al. [2016]) is to use a sensor that is mounted inside the mattress that stores information on the users movements during the night that is then used to find irregularities, another solution would be to use the sensors in the device while the device is placed on the bed with while the person is asleep as described in Monitoring sleep and detecting irregular nights through unconstrained smartphone sensing (from references Sun et al. [2017]).
The second solution would be more useful in this project because it doesnt need an additional specialised device, but some users may not feel comfortable leaving their phone on their bed while it is charging. Other forms of gathering data could be to use external sensors like heart-rate sensors (from references Smolen et al. [2010]) that could monitor the user without the need to have a smartphone lying on the users bed, but this method would also require a dedicated device and the user to wear some form of heart-rate monitor that they may find uncomfortable, disturbing their sleep further.

2.3.2 How to store the data?

The next problem that needs to be solved is the problem of storing the data taken from the device, because assuming the user is going to using this application for an extended period there will be too much data to store on the local device. The most realistic solution to this will be to use a cloud based database that the device can call when either the sleep session is finished or at regular intervals while the session is still running(from references Smolen et al. [2010]). Sending the data after the session will be a better solution as it will not need a constant internet connection lowering the amount of battery usage that will be used during the session. If an external device is going to be used there would have to be a wireless connection to the device that, for this method a constant transfer to the database would also be a worse choose because it would involve a larger battery usage as well as a constant load being put on the smartphone CPU that may lead to a shorter life.

When the data is being processed it will be downloaded or streamed to reduce the memory usage on the smartphone (from references Merilahti et al. [2007]).

2.3.3 How to summarise to data into value?

The system will need to display a simple to understand value for each session that the user will be able to use when reviewing their sleep for a period, one way this could be done is to store a variable that represents the way the users sleep is processing meaning the higher the value the better quality of sleep (from references Barsocchi et al. [2016]). Another way of displaying this will be to use a written representation as the identifier but this would make it harder for the user to understand the differences between to each session if there arent enough identifiers for i.e. poor, good, excellent.

2.3.4 How to analysis a uses sleep?

There are multiple ways to monitor a users sleep but with the technologies that we are going to be using they are limited to movement analysis unless there is an external sensor used with a smartphone but using this configuration would mean that the entire project will have to changed, so it is preferable to use a known algorithm that operates on movement data that can be read using a smartphone internal sensor.
A solution using a smartphone internal sensor would be to use the similar algorithm used in by P. Barsocchi and his team (from references Barsocchi et al. [2016]), in this paper they use an algorithm after reading data form a grid of pressure sensor built into the mattress, the algorithm used in this paper uses the values taken from the grid but only pay attention to the sensors that have the most drastic changes and does not pay attention to the sensors that are not changing because it assumes that these either dont have the person directly over them or are do not have any weight on top of them, this means that the algorithm is only affected by the sensors that are directly under the user and have the most important data.

2.3.5 What is the outcome of the data?

The most desirable outcome of this project will be an application that can record analysis and display data, the display of data to the user will need to an easy to understand display of values that summaries past session into a list or graph that the user can understand at a glance but still gives enough information to the user that they will be able to make an accurate judgement how they can improve their sleep without alienating the user with confusing statistics (from references Mendez et al. [2005]).

2.3.6 What type of sleep is best?

There are two major types of sleep that can be observed without the use of Electroencephalography machines (EEG), the first is non-rapid eye movement (non-REM) sleep and the second is Rapid eye movement (REM), these two types have their own characteristics REM sleep can be described as active sleep while non-REM is quite sleep these too types differ in more than just eye movement, but the most notable difference is the movement levels of the body but the body movement characteristics of each type are the opposite to the eye movements because while there will be lots of major body movements i.e. turning and rolling in bed during non-REM sleep the body during REM sleep is almost paralysed but the brain is still working i.e. dreaming. Because of these large differences (from references Ref [2012]) described non-REM sleep as idling brain in a moving body and REM sleep as actively hallucinating brain in a paralysed body (from references Ref [2012]).
These large differences make it easier to pick to categorise each type into a better and worse quality for refreshing the user

2.4 Conclusion

In conclusion each paper tries to use different variables from heart-rate to brain waves, but almost all the papers produce a similar result with different levels of accuracy and reliability, some of the papers where helpful in understanding the expected result from each item while other papers only state the found results without talking about the expected results.

Chapter 3

Methodology

3.1 Analysis

3.1.1 How will this project be useful

The main function of this project is to help people that maybe having sleep problems but don't need help form medical professional, but want to understand and improve the sleep that they are getting without the need of expensive medical equipment.
With the availability of portable heart-rate monitors for sport and fitness uses this project will targeting this area of users that already have a heart-rate monitor or are open to getting a device.

3.1.2 Who will this project help

The person this project is targeting are users that maybe have or already have a Bluetooth heart-rate monitor, this project will not be a solution too a users sleep problems but a management system for them to understand the problems and work around find solutions, Users that are using the app are maybe normal people that do not have a medical or technical background and are looking to improve the sleep pattern without being swamped with hard to understand data.

3.2 Is this project viable

3.2.1 Problem 1: Bluetooth

The first problem in this project is how too connect a heart-rate monitor to the android device in a reliable way as to manage the input and output stream without using all the android devices battery and storage, the solution to this is to use the android Bluetooth API that manages all the connections to Bluetooth devices and the android Bluetooth profiles to manage any connected devices.

3.2.2 Problem 2: file storage

The second problem presented in this project is managing the data after it has been collected the solution being used for this problems is to use google firebase for storing the heart-rate files and user data, the authentication of the users will also be done through firebase, the users email and password will be used for authentication and any files or tags that they creae will be added to there storage only.

3.2.3 Problem 3: sleep analysis

The sleep analysis is one of the most important parts of this project and will need the most attention and testing to get an accurate and usable result for the program, the problem with heart-rate is the amount it can vary from night to night and person to person, from the literature review we found out that a persons heart-rate can vary for many reasons from after activity recovery to daily diet these external influences don't only raise the heart-rate they can sometimes lower a persons heart-rate for example after a person dose some form of cardio activity like running their heart-rate will be raised by about 10-15BPM but on the opposite side a person sleeping listening to calming music while they fall asleep a user may have a lower heart-rate then normal, as well as having large differences between a single

users heart-rate two different people can have very different heart-rates depending on fitness, lifestyle, age and other variables.
The solution that is being used to solve this problem is to have a dynamic average heart-rate that the program uses instead of setting a fixed normal heart-rate that would not fit all users,This means that while the activity is running the program is calculating the average heart-rate and using that as the nightly normal value this means that the program can be more used by a larger variety of users.

3.3 Design

3.3.1 User Interface

The user interface for the application has been arranged in a simple to understand and easy to use way, each page of the app has a white background and with a blue toolbar on the top and buttons or text areas for any functions that might be needed on that page.
This approach was taken over a more complex layout to save time in the development cycle and to allow the application to be easy to use and navigate for the user.

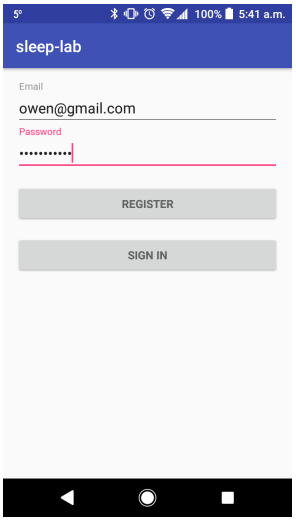

Figure 3.1: Screen Shot of login page

3.3.2 How the user sees there results

The results of the application will be displayed as a graph for each activity showing the heart-rate over time for each activity, this will allow the user to easily gage a understanding of that activity while still having the option of looking at the information in more detail.

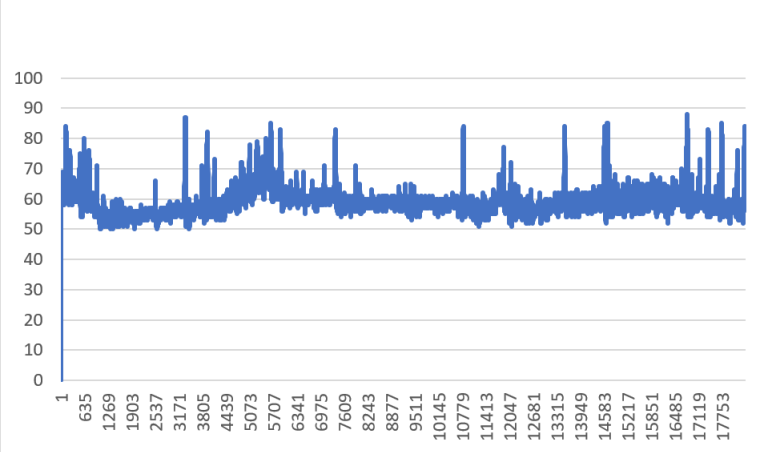

Figure 3.2: Graph of user heart-rate logged against sample rate

Chapter 4

System analysis and design

4.1 Required specification

These are the requirement expected at the end of the development cycle the application will success if all of these requirements are met, The requirements are aimed at a ready to publish application that could be rolled out to users and sold.

- Bluetooth connectivity between the heart rate monitor and the android device
- Cloud database that can store files and user profile information
- Algorithm for analysing the data for the user
- Android user interface

4.2 What is this project for

The purpose of this project is too make a sleep monitoring application for user with sleep problems, users with sleeping conditions could use this application too help their conditions by having clear data on their sleep habits while they are not able to monitor without the use of expensive equipment and and expert skills.

4.3 Use cases diagrams

4.3.1 Sign in page

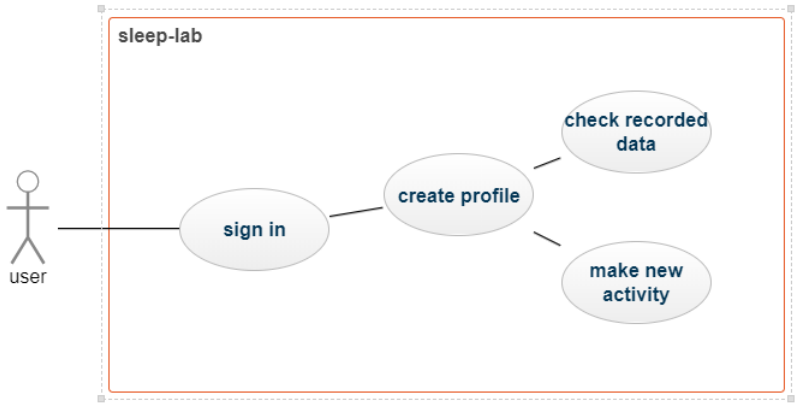

Figure 4.1: Sign In page use case diagram

When a user first opens the application they will have to sign up if they are not already logged in, the process involved in this is

1. open application
2. sign in to application if not already logged in.
3. the application will then send the email and password to firebase to create a new user.
4. the user will input their information for example(nick names, profile image, date of birth, age).
5. this data is sent to firebase to make up a user profile page.
6. the user is the brought to the profile page.

4.3.2 Login

1. Open application
2. Login to application using email and password.
3. The application will navigate to the statistics page.
4. The user can then choose to start a new activity or expand previously recorded data.
5. When the user open a previously record activity it a graph will open up allowing the user to see the data.

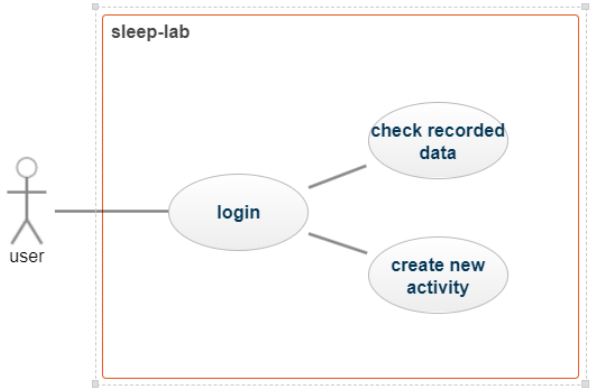

Figure 4.2: Login page use case diagram

4.3.3 start new activity

1. When the user starts a new activity them will be prompted to check if bluetooth is enabled, if yes a scan will start.
2. Users can then connect a device and start the activity.
3. The user will stop the activity and the data will be analysed and sent to firebase.
4. The results of the activity will be displayed as well as the the progress

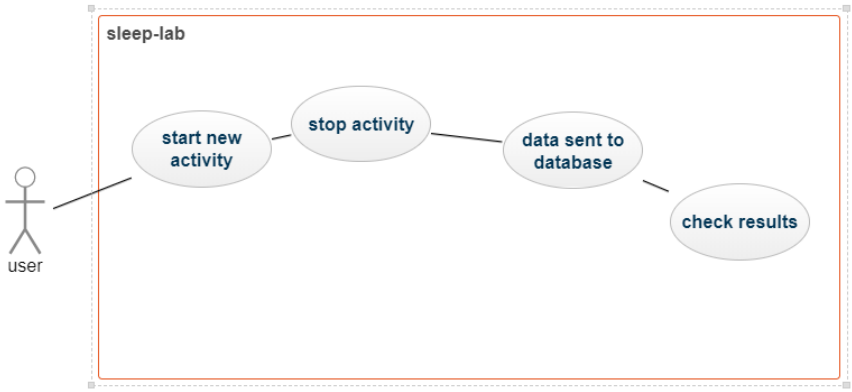

Figure 4.3: Start activity use case

4.4 Program architecture

This program will involve a heart-rate monitor connected to a smartphone device via bluetooth and a firebase cloud storage, the cloud storage will be handled using google's firebase service this will also handle the user authentication, the connection between the device and the heart-rate monitor will be using a bluetooth adapter class instead as the google sensor api that was also a possibility.

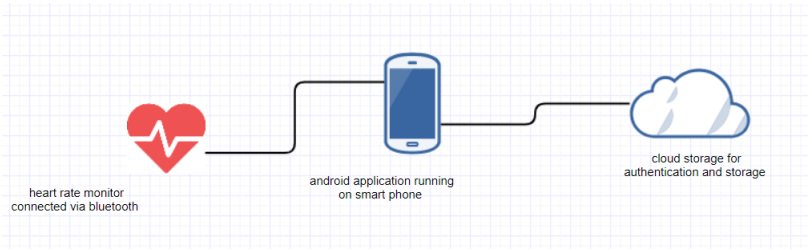

Figure 4.4: Program Architecture

4.5 Wire Frame Design

4.5.1 Login page

This page will be the first page any user sees it contains two input fields and two buttons that allow the user to login to their profile or create an account.

Figure 4.5: Login page wire frame diagram

4.5.2 Profile page

The profile activity allows users to add images to there profile as well as uploading text to firebase real time storage, the users can also able to move to the new activity page and the their statistics page from the profile page

Figure 4.6: Profile page wire frame diagram

4.5.3 New recording page

This page is where the user will start all activities and connect the monitor to the device, it also controls when the activity is stopped The new recording page it is where the user will start a new recording and connect a Bluetooth device fig 6

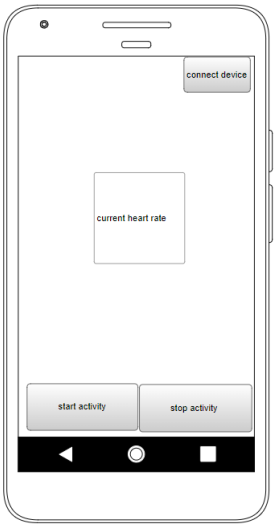

Figure 4.7: Start activity wire frame diagram

Chapter 5

Implementation

5.1 Main features

The main features of the application are the user authentication, cloud storage, real time database. the first draft of the application had all of the above features these features where all done through google Firebase using the Firebase android studio add on tools, Because of these tool connecting an application to fire base can be done fast and with much trouble thanks to androids naming conventions.

Thanks to the simple method of connecting it allows all cloud database and storage to be managed through Firebase and using their API to control the user actions.

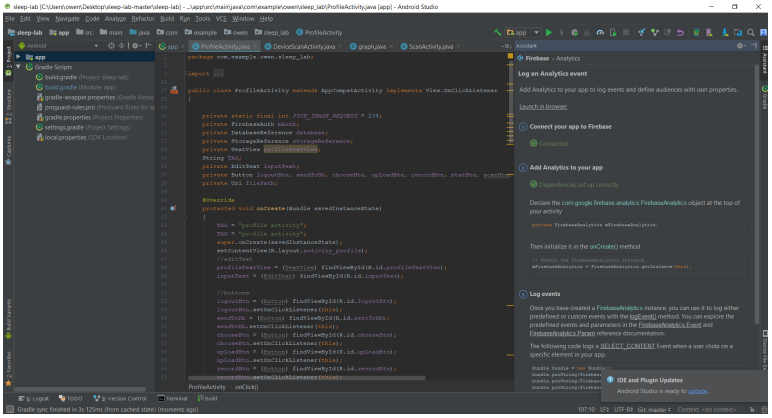

Figure 5.1: Screen shot of Firebase Tool

5.1.1 Authentication

The authentication of the users are done on the server side through Firebase, Firebase uses an object orientated style to allow each entry to easily to added to a user, the reason Firebase was used was because of the amount of documentation and success it has previously had allowing it to be quickly added to any android application after apk level 26.

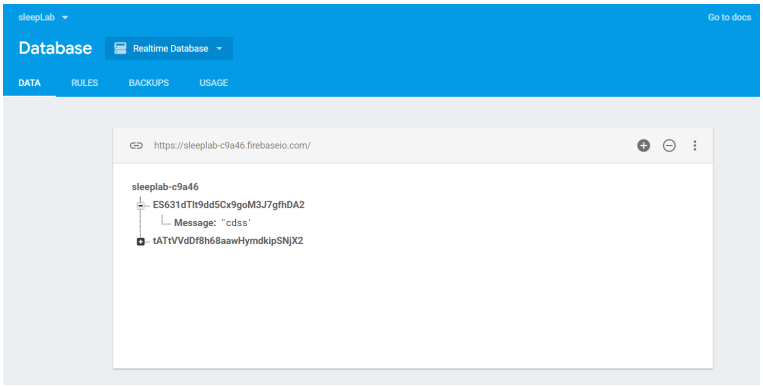

Figure 5.2: Firebase database in browser

5.1.2 Database

Te database side of the cloud storage solution is also run on Firebase, the tag system will be using the real time database as well as the profile descriptions. Each item that is added using the database has a code that lead to the user and then the item allowing Firebase to find the right item in a large database fast and without pulling lots of resources from the device its running on make it an ideal database for a small application that wont need a dedicated server or had lots of resources to use.

5.1.3 Storage

Storage in Firebase is handled in a different way then a database, the storage managers items by folders meaning it may take longer for each item to be reached and processes but item in storage are less likely to be used then the items in the database, this means that the it will require a lower amount of memory on the android device but ill mean a longer load time.

5.1.4 Bluetooth

Bluetooth presented a large problem for this project as the speed of the constantly updating standards and the lack of learning resources has meant that the method of connecting the Bluetooth device has changed twice during the development cycle of this application, To try and keep up with the standard expected form the application and so the application can stay relevant even after the next android update. The way the Bluetooth adapter is accessed on a android device is unique compared to accessing the internal sensor adapter, The internal sensor adapter only required three lines of code to use any of the internal sensors but for Bluetooth the developer will need to use a minimum of two different classes depending on which method you use but the method I choose required two external classes as well as any code in the activity file. The large difference between standards for sensor adapter and Bluetooth adapter, the lack of learning resources and the frequency of updates was one of the greatest problems in the development causing several set backs.

5.1.5 Analysis

The analysis section of this project was not completed but the planned method for analysing heart-rate was to break the activity into ten sections and get the average of each of those sections, and compare it against a base value and as the application learnt more about the user it would change that base value either up of down to till it was a more suitable value for the user, for example a super fit person would have a low heart-rate during sleep but a overweight person would have a higher heart-rate during sleep. The negative side of this method is that a person that has an irregular heart beat or dose some physical activity during the day their heart-rate will appear higher then the true value would be lowering the accuracy of the system.

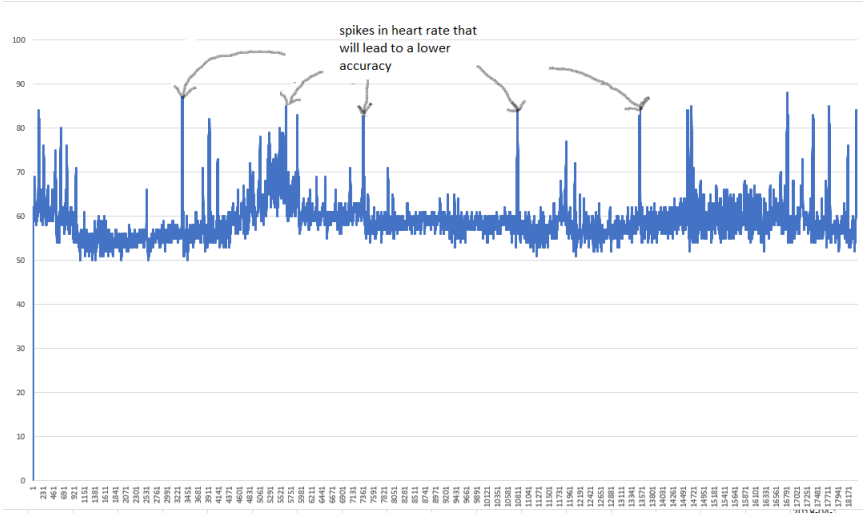

Figure 5.3: heart-rate peak

Chapter 6

Testing and Evaluation

6.1 Aspects of tested

6.1.1 Sleep analysis

The main focus on testing for this project was on the sleep analysis, the feature did not make it too the final version but there was still tests run to see the best method for analysing heart-rate and the best algorithm to use after the data has been collected. Two different method where tested for analysing the data after collecting the data, The two methods tested where the average method and the constant monitoring method. The two method where trialled against a number of different heart-rate recording one form a 24 year old male, 27 year old female and a semi complete recording from the 24 year old male. The expected results of the experiment where that the the average method would be more accurate overall but less sensitive to change then the constant monitoring method.

6.1.2 Results of the tests

Fig 8 shows the recording of the female subject, in the recording its visible that this person has a higher resting heart-rate then the male and has more variation in the heart-rate then the male, while fig 7 and 9 show the male heart-rate averages at 62BPM and peaks in both recordings at 93BPM.

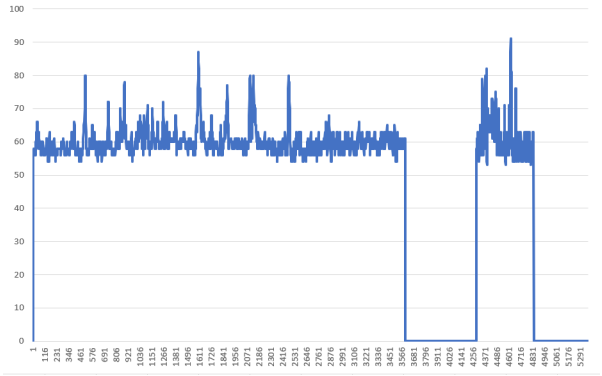

Figure 6.1: Incomplete file due to Bluetooth reliability

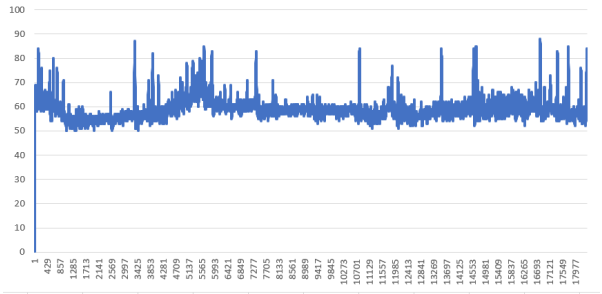

Figure 6.2: Recording of female heart-rate

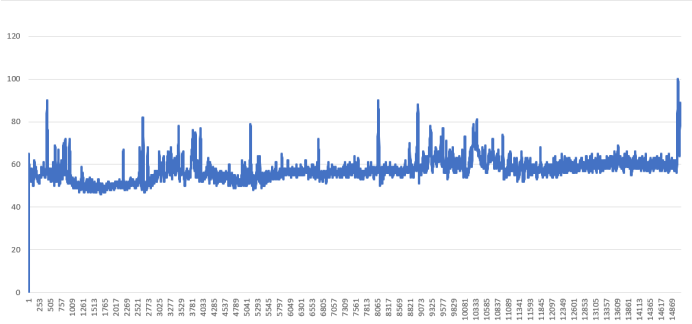

Figure 6.3: Recording of 24 year old male

The result of testing the two method against the three different files showed that the average method was better at dealing with peaks but could not detect small peak but would just lead to a slightly higher average foe that section, the constant monitoring method had the opposite effect the small peak had a large influence on the result and the average heart-rate for the entire file was not taken in to consideration.

6.1.3 Evaluation

To produce the best the most accurate method a hybrid of the average and th constant method is need, the constant method should be used for detecting fast but short lasting jumps in the heart-rate that could mean that the person is unsettled in bed while the average method is used to produce a model of the entire activity. Using this method it would be possible to detect peaks in the heart-rate while still keeping the accuracy of the average method, a notification could be setup to notify to notify the user of there events without waking them.

Chapter 7

Conclusion

In conclusion the development of this application was a disappointment but the test and theory side of the project where completed, this means that the the project was not a complete failure but the project did not meet the goals that where set out at the start of the project meaning that there is a lot of work to be done before this project could be considered a success.

7.0.1 Future work

Future work that i would like to get completed on this project would be to complete the development to of the application and get it to a usable state that it could be relied upon, including finishing the analysis algorithm and fixing the bluetooth problems.

Bibliography

Basic sleep mechanisms: An integrative review. *Central Nervous System Agents in Medicinal Chemistry*, 12(1):38–54, Mar 2012. URL http://www.eurekaselect.com/openurl/content.php?genre=article&issn=18715249&volume=12&issue=1&spage=38.

P. Barsocchi, M. Bianchini, A. Crivello, D. L. Rosa, F. Palumbo, and F. Scarselli. An unobtrusive sleep monitoring system for the human sleep behaviour understanding. In *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, page 91, 2016. doi: 10.1109/CogInfoCom.2016.7804531. ID: 1.

M. O. Mendez, O. P. Villantieri, A. M. Bianchi, and S. Cerutti. Sleep analysis for wearable devices applying autoregressive parametric models. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 7353–7356, 2005. ISBN 1094-687X. doi: 10.1109/IEMBS.2005.1616210. ID: 1.

J. Merilahti, A. Saarinen, J. Parkka, K. Antila, E. Mattila, and I. Korhonen. Long-term subjective and objective sleep analysis of total sleep time and sleep quality in real life settings. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5202–5205, 2007. ISBN 1094-687X. doi: 10.1109/IEMBS.2007.4353514. ID: 1.

M. Smolen, K. Czopek, and P. Augustyniak. Non-invasive sensors based human state in nightlong sleep analysis for home-care. In *2010 Computing in Cardiology*, pages 45–48, 2010. ISBN 0276-6574. ID: 1.

J. Sun, Y. Ye, L. Chang, J. Jiang, and X. Ji. Sleep monitoring approach based on belief rule-based systems with pulse oxygen saturation and heart rate. In *2017 29th Chinese*

Control And Decision Conference (CCDC), pages 1335–1340, 2017. doi: 10.1109/CCDC.2017.7978724. ID: 1.

Appendices

Appendix A

code

Appendix A.1

build.gradle:module

```
apply plugin: 'com.android.application'

android {
   compileSdkVersion 26
   defaultConfig {
      applicationId "com.example.owen.sleep_lab"
      minSdkVersion 23
      targetSdkVersion 26
      versionCode 1
      versionName "1.0"
      testInstrumentationRunner "android.support.test.runner.
         AndroidJUnitRunner"
      multiDexEnabled true
   }
   buildTypes {
      release {
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android.txt'),
             'proguard-rules.pro'
      }
   }

   packagingOptions {
      exclude 'META-INF/DEPENDENCIES'
      exclude 'META-INF/NOTICE'
      exclude 'META-INF/NOTICE.txt'
      exclude 'META-INF/LICENSE'
      exclude 'META-INF/LICENSE.txt'
   }
```

```
}

dependencies {
   implementation fileTree(include: ['*.jar'], dir: 'libs')
   implementation 'com.android.support:appcompat-v7:26.1.0'
   implementation 'com.android.support:design:26.1.0'
   implementation 'com.google.firebase:firebase-core:15.0.0'
   implementation 'com.google.firebase:firebase-crash:15.0.0'
   compile 'com.google.firebase:firebase-storage:15.0.0'
   compile 'com.google.firebase:firebase-auth:15.0.0'
   compile 'com.google.firebase:firebase-database:15.0.0'
   compile 'com.jjoe64:graphview:4.2.1'
   implementation 'com.android.support.constraint:constraint-layout
      :1.1.0'
   testImplementation 'junit:junit:4.12'
   androidTestImplementation 'com.android.support.test:runner:1.0.1'
   androidTestImplementation 'com.android.support.test.espresso:
      espresso-core:3.0.1'
}

apply plugin: 'com.google.gms.google-services'
```

Appendix A.2

build.gradle:project

```
// Top-level build file where you can add configuration options common
    to all sub-projects/modules.

buildscript {

repositories {
google()
jcenter()
}
dependencies {
classpath 'com.android.tools.build:gradle:3.1.1'
classpath 'com.google.gms:google-services:3.1.1'



// NOTE: Do not place your application dependencies here; they belong
// in the individual module build.gradle files
}
}

allprojects {
repositories {
google()
jcenter()
```

```
maven {
url "https://maven.google.com"
}
}
}

task clean(type: Delete) {
delete rootProject.buildDir
}
```

Appendix A.3

manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.owen.sleep_lab">

<!-- To auto-complete the email text field in the login form with the
    user's emails -->
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_PROFILE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
    />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity
android:name=".LoginActivity"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".ProfileActivity" />
<activity android:name=".RecordActivity" />
<activity android:name=".StatsActivity" />
<activity android:name=".DeviceScanActivity"/>
<activity android:name=".ScanActivity"/>
</application>
```

```
</manifest>
```

Appendix A.4

BluetoothLeService.java

```
package com.example.owen.sleep_lab;

import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothGattCallback;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattDescriptor;
import android.bluetooth.BluetoothGattService;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothProfile;
import android.content.Context;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;

import java.util.List;
import java.util.UUID;

public class BluetoothLeService extends Service
{
private final static String TAG = BluetoothLeService.class.getSimpleName
    ();

private BluetoothManager mBluetoothManager;
private BluetoothAdapter mBluetoothAdapter;
private String mBluetoothDeviceAddress;
private BluetoothGatt mBluetoothGatt;
private int mConnectionState = STATE_DISCONNECTED;

private static final int STATE_DISCONNECTED = 0;
private static final int STATE_CONNECTING = 1;
private static final int STATE_CONNECTED = 2;

public final static String ACTION_GATT_CONNECTED = "com.example.bluetooth
    .le.ACTION_GATT_CONNECTED";
public final static String ACTION_GATT_DISCONNECTED = "com.example.
    bluetooth.le.ACTION_GATT_DISCONNECTED";
public final static String ACTION_GATT_SERVICES_DISCOVERED = "com.example
    .bluetooth.le.ACTION_GATT_SERVICES_DISCOVERED";
public final static String ACTION_DATA_AVAILABLE = "com.example.bluetooth
    .le.ACTION_DATA_AVAILABLE";
```

```
public final static String EXTRA_DATA = "com.example.bluetooth.le.
    EXTRA_DATA";

public final static UUID UUID_HEART_RATE_MEASUREMENT = UUID.fromString(
    SampleGattAttributes.HEART_RATE_MEASUREMENT);

private final BluetoothGattCallback mGattCallback = new
    BluetoothGattCallback()
{
@Override
public void onConnectionStateChange(BluetoothGatt gatt, int status, int
    newState)
{
String intentAction;
if(newState == BluetoothProfile.STATE_CONNECTED)
{
intentAction = ACTION_GATT_CONNECTED;
mConnectionState = STATE_CONNECTED;
broadcastUpdate(intentAction);
Log.i(TAG, "Connected to GATT server.");
// Attempts to discover services after successful connection.
Log.i(TAG, "Attempting to start service discovery:" + mBluetoothGatt.
    discoverServices());

}
else if(newState == BluetoothProfile.STATE_DISCONNECTED)
{
intentAction = ACTION_GATT_DISCONNECTED;
mConnectionState = STATE_DISCONNECTED;
Log.i(TAG, "Disconnected from GATT server.");
broadcastUpdate(intentAction);
}
}

@Override
public void onServicesDiscovered(BluetoothGatt gatt, int status)
{
if(status == BluetoothGatt.GATT_SUCCESS)
{
broadcastUpdate(ACTION_GATT_SERVICES_DISCOVERED);
}
else
{
Log.w(TAG, "onServicesDiscovered received: " + status);
}
}

@Override
public void onCharacteristicRead(BluetoothGatt gatt,
    BluetoothGattCharacteristic characteristic, int status)
{
```

```
if(status == BluetoothGatt.GATT_SUCCESS)
{
broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
}
}

@Override
public void onCharacteristicChanged(BluetoothGatt gatt,
    BluetoothGattCharacteristic characteristic)
{
broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
}
};

private void broadcastUpdate(final String action)
{
final Intent intent = new Intent(action);
sendBroadcast(intent);
}

private void broadcastUpdate(final String action, final
    BluetoothGattCharacteristic characteristic)
{
final Intent intent = new Intent(action);

if(UUID_HEART_RATE_MEASUREMENT.equals(characteristic.getUuid()))
{
int flag = characteristic.getProperties();
int format = -1;
if((flag & 0x01) != 0)
{
format = BluetoothGattCharacteristic.FORMAT_UINT16;
Log.d(TAG, "Heart rate format UINT16.");
}
else
{
format = BluetoothGattCharacteristic.FORMAT_UINT8;
Log.d(TAG, "Heart rate format UINT8.");
}
final int heartRate = characteristic.getIntValue(format, 1);
Log.d(TAG, String.format("Received heart rate: %d", heartRate));
intent.putExtra(EXTRA_DATA, String.valueOf(heartRate));
}
else
{
// For all other profiles, writes the data formatted in HEX.
final byte[] data = characteristic.getValue();
if(data != null && data.length > 0)
{
final StringBuilder stringBuilder = new StringBuilder(data.length);
for(byte byteChar : data)
```

```
{
stringBuilder.append(String.format("%02X ", byteChar));
}
intent.putExtra(EXTRA_DATA, new String(data) + "\n" + stringBuilder.
   toString());
}
}
sendBroadcast(intent);
}

public class LocalBinder extends Binder
{
BluetoothLeService getService()
{
return BluetoothLeService.this;
}
}

@Override
public IBinder onBind(Intent intent)
{
return mBinder;
}

@Override
public boolean onUnbind(Intent intent)
{
close();
return super.onUnbind(intent);
}

private final IBinder mBinder = new LocalBinder();

public boolean initialize()
{

if(mBluetoothManager == null)
{
mBluetoothManager = (BluetoothManager) getSystemService(Context.
   BLUETOOTH_SERVICE);
if(mBluetoothManager == null)
{
Log.e(TAG, "Unable to initialize BluetoothManager.");
return false;
}
}

mBluetoothAdapter = mBluetoothManager.getAdapter();
if(mBluetoothAdapter == null)
{
Log.e(TAG, "Unable to obtain a BluetoothAdapter.");
```

```
return false;
}

return true;
}

public boolean connect(final String address)
{
if(mBluetoothAdapter == null || address == null)
{
Log.w(TAG, "BluetoothAdapter not initialized or unspecified address.");
return false;
}

if(mBluetoothDeviceAddress != null && address.equals(
   mBluetoothDeviceAddress) && mBluetoothGatt != null)
{
Log.d(TAG, "Trying to use an existing mBluetoothGatt for connection.");
if(mBluetoothGatt.connect())
{
mConnectionState = STATE_CONNECTING;
return true;
}
else
{
return false;
}
}

final BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address)
   ;
if(device == null)
{
Log.w(TAG, "Device not found.  Unable to connect.");
return false;
}
mBluetoothGatt = device.connectGatt(this, false, mGattCallback);
Log.d(TAG, "Trying to create a new connection.");
mBluetoothDeviceAddress = address;
mConnectionState = STATE_CONNECTING;
return true;
}

public void disconnect()
{
if(mBluetoothAdapter == null || mBluetoothGatt == null)
{
Log.w(TAG, "BluetoothAdapter not initialized");
return;
}
mBluetoothGatt.disconnect();
```

```
}

public void close()
{
if(mBluetoothGatt == null)
{
return;
}
mBluetoothGatt.close();
mBluetoothGatt = null;
}

public void readCharacteristic(BluetoothGattCharacteristic characteristic
    )
{
if(mBluetoothAdapter == null || mBluetoothGatt == null)
{
Log.w(TAG, "BluetoothAdapter not initialized");
return;
}
mBluetoothGatt.readCharacteristic(characteristic);
}

public void setCharacteristicNotification(BluetoothGattCharacteristic
    characteristic, boolean enabled)
{
if(mBluetoothAdapter == null || mBluetoothGatt == null)
{
Log.w(TAG, "BluetoothAdapter not initialized");
return;
}
mBluetoothGatt.setCharacteristicNotification(characteristic, enabled);

// This is specific to Heart Rate Measurement.
if(UUID_HEART_RATE_MEASUREMENT.equals(characteristic.getUuid()))
{
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(UUID.
    fromString(SampleGattAttributes.CLIENT_CHARACTERISTIC_CONFIG));
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
}
}

public List<BluetoothGattService> getSupportedGattServices()
{
if(mBluetoothGatt == null)
{
return null;
}
return mBluetoothGatt.getServices();
}
```

```
}
```

Appendix A.5

DeviceScanActivity.java

```
package com.example.owen.sleep_lab;

import android.Manifest;
import android.app.Activity;
import android.app.ListActivity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;

public class DeviceScanActivity extends ListActivity
{
private LeDeviceListAdapter mLeDeviceListAdapter;
private BluetoothAdapter mBluetoothAdapter;
private boolean mScanning;
private Handler mHandler;

private static final int REQUEST_ENABLE_BT = 1;
// Stops scanning after 10 seconds.
private static final long SCAN_PERIOD = 10000;

@Override
public void onCreate(Bundle savedInstanceState)
{

int permissionCheck = ContextCompat.checkSelfPermission(this, Manifest.
   permission.ACCESS_FINE_LOCATION);
if(permissionCheck != PackageManager.PERMISSION_GRANTED)
```

```
{
if(ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.
    permission.ACCESS_FINE_LOCATION))
{
Toast.makeText(this, "The permission to get BLE location data is required
    ", Toast.LENGTH_SHORT).show();
}
else
{
requestPermissions(new String[]{Manifest.permission.
    ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION}, 1);
}
}
else
{
Toast.makeText(this, "Location permissions already granted", Toast.
    LENGTH_SHORT).show();
}
super.onCreate(savedInstanceState);
mHandler = new Handler();

if(!getPackageManager().hasSystemFeature(PackageManager.
    FEATURE_BLUETOOTH_LE))
{
Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show
    ();
finish();
}

final BluetoothManager bluetoothManager = (BluetoothManager)
    getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();

// Checks if Bluetooth is supported on the device.
if(mBluetoothAdapter == null)
{
Toast.makeText(this, R.string.error_bluetooth_not_supported, Toast.
    LENGTH_SHORT).show();
finish();
return;
}
}

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
getMenuInflater().inflate(R.menu.main, menu);
if(!mScanning)
{
menu.findItem(R.id.menu_stop).setVisible(false);
menu.findItem(R.id.menu_scan).setVisible(true);
```

```
menu.findItem(R.id.menu_refresh).setActionView(null);
}
else
{
menu.findItem(R.id.menu_stop).setVisible(true);
menu.findItem(R.id.menu_scan).setVisible(false);
menu.findItem(R.id.menu_refresh).setActionView(R.layout.
    actionbar_indeterminate_progress);
}
return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item)
{
switch(item.getItemId())
{
case R.id.menu_scan:
mLeDeviceListAdapter.clear();
scanLeDevice(true);
break;
case R.id.menu_stop:
scanLeDevice(false);
break;
}
return true;
}

@Override
protected void onResume()
{
super.onResume();

if(!mBluetoothAdapter.isEnabled())
{
if(!mBluetoothAdapter.isEnabled())
{
Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE
    );
startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
}

// Initializes list view adapter.
mLeDeviceListAdapter = new LeDeviceListAdapter();
setListAdapter(mLeDeviceListAdapter);
scanLeDevice(true);
}

@Override
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent
    data)
{
// User chose not to enable Bluetooth.
if(requestCode == REQUEST_ENABLE_BT && resultCode == Activity.
    RESULT_CANCELED)
{
finish();
return;
}
super.onActivityResult(requestCode, resultCode, data);
}

@Override
protected void onPause()
{
super.onPause();
scanLeDevice(false);
mLeDeviceListAdapter.clear();
}

@Override
protected void onListItemClick(ListView l, View v, int position, long id)
{
final BluetoothDevice device = mLeDeviceListAdapter.getDevice(position);
if(device == null)
{
return;
}
final Intent intent = new Intent(this, ScanActivity.class);
intent.putExtra(ScanActivity.EXTRAS_DEVICE_NAME, device.getName());
intent.putExtra(ScanActivity.EXTRAS_DEVICE_ADDRESS, device.getAddress());
if(mScanning)
{
mBluetoothAdapter.stopLeScan(mLeScanCallback);
mScanning = false;
}
startActivity(intent);
}

private void scanLeDevice(final boolean enable)
{
if(enable)
{
// Stops scanning after a pre-defined scan period.
mHandler.postDelayed(new Runnable()
{
@Override
public void run()
{
mScanning = false;
```

```
mBluetoothAdapter.stopLeScan(mLeScanCallback);
invalidateOptionsMenu();
}
}, SCAN_PERIOD);

mScanning = true;
mBluetoothAdapter.startLeScan(mLeScanCallback);
}
else
{
mScanning = false;
mBluetoothAdapter.stopLeScan(mLeScanCallback);
}
invalidateOptionsMenu();
}

private class LeDeviceListAdapter extends BaseAdapter
{
private ArrayList<BluetoothDevice> mLeDevices;
private LayoutInflater mInflator;

public LeDeviceListAdapter()
{
super();
mLeDevices = new ArrayList<BluetoothDevice>();
mInflator = DeviceScanActivity.this.getLayoutInflater();
}

public void addDevice(BluetoothDevice device)
{
if(!mLeDevices.contains(device))
{
mLeDevices.add(device);
}
}

public BluetoothDevice getDevice(int position)
{
return mLeDevices.get(position);
}

public void clear()
{
mLeDevices.clear();
}

@Override
public int getCount()
{
return mLeDevices.size();
}
```

```
@Override
public Object getItem(int i)
{
return mLeDevices.get(i);
}

@Override
public long getItemId(int i)
{
return i;
}

@Override
public View getView(int i, View view, ViewGroup viewGroup)
{
ViewHolder viewHolder;
// General ListView optimization code.
if(view == null)
{
view = mInflator.inflate(R.layout.listitem_device, null);
viewHolder = new ViewHolder();
viewHolder.deviceAddress = (TextView) view.findViewById(R.id.
    device_address);
viewHolder.deviceName = (TextView) view.findViewById(R.id.device_name);
view.setTag(viewHolder);
}
else
{
viewHolder = (ViewHolder) view.getTag();
}

BluetoothDevice device = mLeDevices.get(i);
final String deviceName = device.getName();
if(deviceName != null && deviceName.length() > 0)
{
viewHolder.deviceName.setText(deviceName);
}
else
{
viewHolder.deviceName.setText(R.string.unknown_device);
}
viewHolder.deviceAddress.setText(device.getAddress());

return view;
}
}

private BluetoothAdapter.LeScanCallback mLeScanCallback = new
    BluetoothAdapter.LeScanCallback()
{
```

```
@Override
public void onLeScan(final BluetoothDevice device, int rssi, byte[]
    scanRecord)
{
runOnUiThread(new Runnable()
{
@Override
public void run()
{
mLeDeviceListAdapter.addDevice(device);
mLeDeviceListAdapter.notifyDataSetChanged();
}
});
}
};

static class ViewHolder
{
TextView deviceName;
TextView deviceAddress;
}
}
```

Appendix A.6

graph

```
package com.example.owen.sleep_lab;

public class graph
{
private double x;
private double y;

public double getX()
{
return x;
}

public void setX(double x)
{
this.x = x;
}

public double getY()
{
return y;
}

public void setY(double y)
```

```
{
this.y = y;
}

public graph(double x, double y)
{
this.x = x;
this.y = y;
}
}
```

Appendix A.7

LoginActivity.java

```
package com.example.owen.sleep_lab;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseApp;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

/**
* A login screen that offers login via email/password.
*/
public class LoginActivity extends AppCompatActivity
{

private static final int REQUEST_READ_CONTACTS = 0;

private FirebaseAuth mAuth;
private AutoCompleteTextView mEmailView;
private EditText mPasswordView;
private View mProgressView;
private View mLoginFormView;
```

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
FirebaseApp.initializeApp(this);
mAuth = FirebaseAuth.getInstance();
if(mAuth.getCurrentUser() != null)
{
finish();
startActivity(new Intent(getApplicationContext(), ProfileActivity.class))
    ;
}
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_login);
// Set up the login form.
mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
mPasswordView = (EditText) findViewById(R.id.password);

mPasswordView.setOnEditorActionListener(new TextView.
    OnEditorActionListener()
{
@Override
public boolean onEditorAction(TextView textView, int id, KeyEvent
    keyEvent)
{
if(id == EditorInfo.IME_ACTION_DONE || id == EditorInfo.IME_NULL)
{
attemptLogin();
return true;
}
return false;
}
});

Button signInBtn = (Button) findViewById(R.id.sign_in);
Button registerBtn = (Button) findViewById(R.id.register);
signInBtn.setOnClickListener(new OnClickListener()
{
public void onClick(View view)
{
attemptLogin();
}
});

registerBtn.setOnClickListener(new OnClickListener()
{
public void onClick(View view)
{
register();
}
});
```

```
mLoginFormView = findViewById(R.id.login_form);
mProgressView = findViewById(R.id.login_progress);
}

private void attemptLogin()
{
// Reset errors.
mEmailView.setError(null);
mPasswordView.setError(null);

// Store values at the time of the login attempt.
String email = mEmailView.getText().toString();
String password = mPasswordView.getText().toString();

boolean cancel = false;
View focusView = null;

// Check for a valid password, if the user entered one.
if(!TextUtils.isEmpty(password) && !isPasswordValid(password))
{
mPasswordView.setError(getString(R.string.error_invalid_password));
focusView = mPasswordView;
cancel = true;
}

// Check for a valid email address.
if(TextUtils.isEmpty(email))
{
mEmailView.setError(getString(R.string.error_field_required));
focusView = mEmailView;
cancel = true;
}
else if(!isEmailValid(email))
{
mEmailView.setError(getString(R.string.error_invalid_email));
focusView = mEmailView;
cancel = true;
}

if(cancel)
{
focusView.requestFocus();
}
else
{
mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(
    this, new OnCompleteListener<AuthResult>()
{
@Override
public void onComplete(@NonNull Task<AuthResult> task)
```

```
{
if(task.isSuccessful())
{
Toast.makeText(LoginActivity.this, "login sussesful", Toast.LENGTH_SHORT)
    .show();
finish();
startActivity(new Intent(getApplicationContext(), ProfileActivity.class))
    ;
}
Toast.makeText(LoginActivity.this, "Login not successful", Toast.
    LENGTH_SHORT).show();
}
});

}
}

public void register()
{
// Reset errors.
mEmailView.setError(null);
mPasswordView.setError(null);

// Store values at the time of the login attempt.
String email = mEmailView.getText().toString();
String password = mPasswordView.getText().toString();

boolean cancel = false;
View focusView = null;

// Check for a valid password, if the user entered one.
if(!TextUtils.isEmpty(password) && !isPasswordValid(password))
{
mPasswordView.setError(getString(R.string.error_invalid_password));
focusView = mPasswordView;
cancel = true;
}

// Check for a valid email address.
if(TextUtils.isEmpty(email))
{
mEmailView.setError(getString(R.string.error_field_required));
focusView = mEmailView;
cancel = true;
}
else if(!isEmailValid(email))
{
mEmailView.setError(getString(R.string.error_invalid_email));
focusView = mEmailView;
cancel = true;
}
```

```
if(cancel)
{
focusView.requestFocus();
}
else
{
//register here
mAuth.createUserWithEmailAndPassword(email, password).
    addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
{
@Override
public void onComplete(@NonNull Task<AuthResult> task)
{
if(task.isSuccessful())
{
finish();
Toast.makeText(LoginActivity.this, "register sussesful", Toast.
    LENGTH_SHORT).show();
finish();
startActivity(new Intent(getApplicationContext(), ProfileActivity.class))
    ;
}
else
{
Toast.makeText(LoginActivity.this, "Register Not Successful", Toast.
    LENGTH_SHORT).show();
}
}
});
}
}

private boolean isEmailValid(String email)
{
//TODO: Replace this with your own logic
return email.contains("@");
}

private boolean isPasswordValid(String password)
{
//TODO: Replace this with your own logic
return password.length() > 2;
}

}
```

Appendix A.8

ProfileActivity.java

```
package com.example.owen.sleep_lab;

import android.content.Intent;
import android.net.Uri;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import java.time.LocalDate;

public class ProfileActivity extends AppCompatActivity implements View.
    OnClickListener
{

private static final int PICK_IMAGE_REQUEST = 234;
private FirebaseAuth mAuth;
private DatabaseReference database;
private StorageReference storageReference;
private TextView profileTextView;
String TAG;
private EditText inputText;
private Button logoutBtn, sendToDb, chooseBtn, upLoadBtn, recordBtn,
    statBtn, scanBtn;
private Uri filePath;

@Override
protected void onCreate(Bundle savedInstanceState)
{
TAG = "profile activity";
TAG = "profile activity";
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_profile);
//editText
profileTextView = (TextView) findViewById(R.id.profileTextView);
inputText = (EditText) findViewById(R.id.inputText);
```

```
//buttons
logoutBtn = (Button) findViewById(R.id.logoutBtn);
logoutBtn.setOnClickListener(this);
sendToDb = (Button) findViewById(R.id.sentToDb);
sendToDb.setOnClickListener(this);
chooseBtn = (Button) findViewById(R.id.chooseBtn);
chooseBtn.setOnClickListener(this);
upLoadBtn = (Button) findViewById(R.id.upLoadBtn);
upLoadBtn.setOnClickListener(this);
recordBtn = (Button) findViewById(R.id.recordBtn);
recordBtn.setOnClickListener(this);
statBtn = (Button) findViewById(R.id.statBtn);
statBtn.setOnClickListener(this);

//Firebase
mAuth = FirebaseAuth.getInstance();
database = FirebaseDatabase.getInstance().getReference();
storageReference = FirebaseStorage.getInstance().getReference();

//************check if user is logged in
if(mAuth.getCurrentUser() == null)
{
startActivity(new Intent(this, LoginActivity.class));
}
FirebaseUser user = mAuth.getCurrentUser();
profileTextView.setText("Welcome " + user.getEmail());

}

@Override
public void onClick(View view)
{
if(view == logoutBtn)
{
mAuth.signOut();
finish();
startActivity(new Intent(this, LoginActivity.class));
}
if(view == sendToDb)
{
saveData();
}
if(view == chooseBtn)
{
showFileChooser();
}
if(view == upLoadBtn)
{
uploadFile();
}
```

```
if(view == recordBtn)
{
Log.d(TAG, "move to record activity ");
startActivity(new Intent(this, RecordActivity.class));
}
if(view == statBtn)
{
startActivity(new Intent(this, StatsActivity.class));
}
}

private void uploadFile()
{
if(filePath != null)
{
StorageReference riversRef = storageReference.child("images/profileImage.
    jpg");

riversRef.putFile(filePath).addOnSuccessListener(new OnSuccessListener<
    UploadTask.TaskSnapshot>()
{
@Override
public void onSuccess(UploadTask.TaskSnapshot taskSnapshot)
{
Toast.makeText(ProfileActivity.this, "File Uploaded", Toast.LENGTH_SHORT)
    .show();

}
}).addOnFailureListener(new OnFailureListener()
{
@Override
public void onFailure(@NonNull Exception exception)
{
Toast.makeText(ProfileActivity.this, exception.getMessage(), Toast.
    LENGTH_SHORT).show();

}
});
}
else
{
Toast.makeText(this, "Upload Failed", Toast.LENGTH_SHORT).show();
}

}

private void showFileChooser()
{
Intent intent = new Intent();
intent.setType("image/*");
intent.setAction(intent.ACTION_GET_CONTENT);
```

```
startActivityForResult(Intent.createChooser(intent, "select file to
    upload"), PICK_IMAGE_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
    data)
{
super.onActivityResult(requestCode, resultCode, data);

if(requestCode == PICK_IMAGE_REQUEST && data != null && data.getData() !=
     null)
{
filePath = data.getData();
}
}

public void saveData()
{
String message = inputText.getText().toString();

Save save = new Save(message);

FirebaseUser user = mAuth.getCurrentUser();

database.child(user.getUid()).setValue(save);
Toast.makeText(this, "Message saved ", Toast.LENGTH_SHORT).show();

Toast.makeText(this, "message saved", Toast.LENGTH_SHORT).show();
}
}
```

Appendix A.9

RecorActivity.java

```
package com.example.owen.sleep_lab;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothHealth;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

```
public class RecordActivity extends AppCompatActivity implements View.
    OnClickListener
{
//bluetooth
private BluetoothAdapter mBluetoothAdapter;
private Button scanBtn, startRecordBtn, stopRecordBtn;
int REQUEST_ENABLE_BT = 1;
public BluetoothHealth mHeartRateMonitor;
public BluetoothDevice mdevice;
String deviceName, deviceMacAddress;


private static final String TAG = RecordActivity.class.getSimpleName();

protected void onCreate(Bundle savedInstanceState)
{

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_record);
startRecordBtn = (Button) findViewById(R.id.startRecordBtn);
stopRecordBtn = (Button) findViewById(R.id.stopRecordBtn);
scanBtn = (Button) findViewById(R.id.scanBtn);
startRecordBtn.setOnClickListener(this);
stopRecordBtn.setOnClickListener(this);
scanBtn.setOnClickListener(this);

//broadcast receiver
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);

mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if(mBluetoothAdapter == null)
{
Log.d(TAG, "device not compatible with bluetooth");
Toast.makeText(this, "Bluetooth is not available on this device", Toast.
    LENGTH_SHORT).show();
}

if(!mBluetoothAdapter.isEnabled())
{
Intent enableBtIntent = new Intent(mBluetoothAdapter.
    ACTION_REQUEST_ENABLE);
startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
IntentFilter enIntent = new IntentFilter(BluetoothAdapter.
    ACTION_STATE_CHANGED);
registerReceiver(mReceiver, enIntent);
}

}//end of onCreate
```

```
@Override
public void onClick(View v)
{
if (v == scanBtn)
{
startActivity(new Intent(this, DeviceScanActivity.class));
}
}

private final BroadcastReceiver mReceiver = new BroadcastReceiver()
{
@Override
public void onReceive(Context context, Intent intent)
{
Toast.makeText(context, "mReceiver called", Toast.LENGTH_SHORT).show();
Toast.makeText(context, "your device is now discoverable", Toast.
    LENGTH_SHORT).show();

String action = intent.getAction();

if(BluetoothDevice.ACTION_FOUND.equals(action))
{
BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.
    EXTRA_DEVICE);

deviceName = device.getName();
Toast.makeText(RecordActivity.this, "device found"+device.getName(),
    Toast.LENGTH_SHORT).show();
deviceMacAddress = device.getAddress();
}
}
};
}
```

Appendix A.10

SampleGattAttributes.java

```
/*
 * Copyright (C) 2013 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied.
```

```
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.owen.sleep_lab;

import java.util.HashMap;

/**
 * This class includes a small subset of standard GATT attributes for
    demonstration purposes.
 */
public class SampleGattAttributes
{
private static HashMap<String, String> attributes = new HashMap();
public static String HEART_RATE_MEASUREMENT = "00002a37
    -0000-1000-8000-00805f9b34fb";
public static String CLIENT_CHARACTERISTIC_CONFIG = "
    00002902-0000-1000-8000-00805f9b34fb";

static
{
// Sample Services.
attributes.put("0000180d-0000-1000-8000-00805f9b34fb", "Heart Rate
    Service");
attributes.put("0000180a-0000-1000-8000-00805f9b34fb", "Device
    Information Service");
// Sample Characteristics.
attributes.put(HEART_RATE_MEASUREMENT, "Heart Rate Measurement");
attributes.put("00002a29-0000-1000-8000-00805f9b34fb", "Manufacturer
    Name String");
}

public static String lookup(String uuid, String defaultName)
{
String name = attributes.get(uuid);
return name == null ? defaultName : name;
}
}
```

Appendix A.11

ScanActivity.java

```
package com.example.owen.sleep_lab;

import android.app.Activity;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattService;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
```

```
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ExpandableListView;
import android.widget.SimpleExpandableListAdapter;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class ScanActivity extends Activity implements View.
    OnClickListener
{
private final static String TAG = ScanActivity.class.getSimpleName();

public static final String EXTRAS_DEVICE_NAME = "DEVICE_NAME";
public static final String EXTRAS_DEVICE_ADDRESS = "DEVICE_ADDRESS";

private TextView mConnectionState;
private TextView mDataField;
private String mDeviceName;
private Button disconnect, connect;
private String mDeviceAddress;
private ExpandableListView mGattServicesList;
private BluetoothLeService mBluetoothLeService;
private ArrayList<ArrayList<BluetoothGattCharacteristic>>
    mGattCharacteristics = new ArrayList<ArrayList<
    BluetoothGattCharacteristic>>();
private boolean mConnected = false;
private BluetoothGattCharacteristic mNotifyCharacteristic;

private final String LIST_NAME = "NAME";
private final String LIST_UUID = "UUID";

// Code to manage Service lifecycle.
private final ServiceConnection mServiceConnection = new
    ServiceConnection()
{

@Override
```

```
public void onServiceConnected(ComponentName componentName, IBinder
    service)
{
mBluetoothLeService = ((BluetoothLeService.LocalBinder) service).
    getService();
if(!mBluetoothLeService.initialize())
{
Log.e(TAG, "Unable to initialize Bluetooth");
finish();
}
// Automatically connects to the device upon successful start-up
    initialization.
mBluetoothLeService.connect(mDeviceAddress);
}

@Override
public void onServiceDisconnected(ComponentName componentName)
{
mBluetoothLeService = null;
}
};

private final BroadcastReceiver mGattUpdateReceiver = new
    BroadcastReceiver()
{
@Override
public void onReceive(Context context, Intent intent)
{
final String action = intent.getAction();
if(BluetoothLeService.ACTION_GATT_CONNECTED.equals(action))
{
mConnected = true;
updateConnectionState(R.string.connected);
invalidateOptionsMenu();
}
else if(BluetoothLeService.ACTION_GATT_DISCONNECTED.equals(action))
{
mConnected = false;
updateConnectionState(R.string.disconnected);
invalidateOptionsMenu();
clearUI();
}
else if(BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED.equals(action)
    )
{
// Show all the supported services and characteristics on the user
    interface.
displayGattServices(mBluetoothLeService.getSupportedGattServices());
}
else if(BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action))
{
```

```
displayData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA));
}
}
};

private final ExpandableListView.OnChildClickListener
    servicesListClickListner = new ExpandableListView.OnChildClickListener
    ()
{
@Override
public boolean onChildClick(ExpandableListView parent, View v, int
    groupPosition, int childPosition, long id)
{
if(mGattCharacteristics != null)
{
final BluetoothGattCharacteristic characteristic = mGattCharacteristics.
    get(groupPosition).get(childPosition);
final int charaProp = characteristic.getProperties();
if((charaProp | BluetoothGattCharacteristic.PROPERTY_READ) > 0)
{

if(mNotifyCharacteristic != null)
{
mBluetoothLeService.setCharacteristicNotification(mNotifyCharacteristic,
    false);
mNotifyCharacteristic = null;
}
mBluetoothLeService.readCharacteristic(characteristic);
}
if((charaProp | BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0)
{
mNotifyCharacteristic = characteristic;
mBluetoothLeService.setCharacteristicNotification(characteristic, true);
}
return true;
}
return false;
}
};

private void clearUI()
{
mGattServicesList.setAdapter((SimpleExpandableListAdapter) null);
mDataField.setText(R.string.no_data);
}

@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_scan);
```

```
final Intent intent = getIntent();
mDeviceName = intent.getStringExtra(EXTRAS_DEVICE_NAME);
mDeviceAddress = intent.getStringExtra(EXTRAS_DEVICE_ADDRESS);
disconnect = (Button) findViewById(R.id.menu_disconnect);
connect = findViewById(R.id.menu_connect);
connect.setOnClickListener(this);

// Sets up UI references.
mGattServicesList = (ExpandableListView) findViewById(R.id.
   gatt_services_list);
mGattServicesList.setOnChildClickListener(servicesListClickListner);
mConnectionState = (TextView) findViewById(R.id.connection_state);
mDataField = (TextView) findViewById(R.id.data_value);

Intent gattServiceIntent = new Intent(this, BluetoothLeService.class);
bindService(gattServiceIntent, mServiceConnection, BIND_AUTO_CREATE);
}

@Override
protected void onResume()
{
super.onResume();
registerReceiver(mGattUpdateReceiver, makeGattUpdateIntentFilter());
if(mBluetoothLeService != null)
{
final boolean result = mBluetoothLeService.connect(mDeviceAddress);
Log.d(TAG, "Connect request result=" + result);
}
}

@Override
protected void onPause()
{
super.onPause();
unregisterReceiver(mGattUpdateReceiver);
}

@Override
protected void onDestroy()
{
super.onDestroy();
unbindService(mServiceConnection);
mBluetoothLeService = null;
}

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
getMenuInflater().inflate(R.menu.gatt_services, menu);
if(mConnected)
```

```
{
menu.findItem(R.id.menu_connect).setVisible(false);
menu.findItem(R.id.menu_disconnect).setVisible(true);
}
else
{
menu.findItem(R.id.menu_connect).setVisible(true);
menu.findItem(R.id.menu_disconnect).setVisible(false);
}
return true;
}

@Override
public void onClick(View v)
{
if(v == connect)
{
try
{
Toast.makeText(this, "connected" + mDeviceAddress, Toast.LENGTH_SHORT).
    show();
mBluetoothLeService.connect(mDeviceAddress);
}
catch(Exception e)
{
Toast.makeText(this, "Failed to connect", Toast.LENGTH_SHORT).show();
e.printStackTrace();
}
}
if(v == disconnect)
{
try
{
Toast.makeText(this, "disconnected", Toast.LENGTH_SHORT).show();
mBluetoothLeService.disconnect();
}
catch(Exception e)
{
Toast.makeText(this, "Failed to disconnect", Toast.LENGTH_SHORT).show();
e.printStackTrace();
}
}
}

private void updateConnectionState(final int resourceId)
{
runOnUiThread(new Runnable()
{
@Override
public void run()
{
```

```
mConnectionState.setText(resourceId);
}
});
}

private void displayData(String data)
{
if(data != null)
{
mDataField.setText(data);
}
}

private void displayGattServices(List<BluetoothGattService> gattServices)
{
if(gattServices == null)
{
return;
}
String uuid = null;
String unknownServiceString = getResources().getString(R.string.
    unknown_service);
String unknownCharaString = getResources().getString(R.string.
    unknown_characteristic);
ArrayList<HashMap<String, String>> gattServiceData = new ArrayList<
    HashMap<String, String>>();
ArrayList<ArrayList<HashMap<String, String>>> gattCharacteristicData =
    new ArrayList<ArrayList<HashMap<String, String>>>();
mGattCharacteristics = new ArrayList<ArrayList<
    BluetoothGattCharacteristic>>();

// Loops through available GATT Services.
for(BluetoothGattService gattService : gattServices)
{
HashMap<String, String> currentServiceData = new HashMap<String, String
    >();
uuid = gattService.getUuid().toString();
currentServiceData.put(LIST_NAME, SampleGattAttributes.lookup(uuid,
    unknownServiceString));
currentServiceData.put(LIST_UUID, uuid);
gattServiceData.add(currentServiceData);

ArrayList<HashMap<String, String>> gattCharacteristicGroupData = new
    ArrayList<HashMap<String, String>>();
List<BluetoothGattCharacteristic> gattCharacteristics = gattService.
    getCharacteristics();
ArrayList<BluetoothGattCharacteristic> charas = new ArrayList<
    BluetoothGattCharacteristic>();

// Loops through available Characteristics.
for(BluetoothGattCharacteristic gattCharacteristic : gattCharacteristics)
```

```
{
charas.add(gattCharacteristic);
HashMap<String, String> currentCharaData = new HashMap<String, String>();
uuid = gattCharacteristic.getUuid().toString();
currentCharaData.put(LIST_NAME, SampleGattAttributes.lookup(uuid,
    unknownCharaString));
currentCharaData.put(LIST_UUID, uuid);
gattCharacteristicGroupData.add(currentCharaData);
}
mGattCharacteristics.add(charas);
gattCharacteristicData.add(gattCharacteristicGroupData);
}

SimpleExpandableListAdapter gattServiceAdapter = new
    SimpleExpandableListAdapter(this, gattServiceData, android.R.layout.
    simple_expandable_list_item_2, new String[]{LIST_NAME, LIST_UUID}, new
     int[]{android.R.id.text1, android.R.id.text2}, gattCharacteristicData
    , android.R.layout.simple_expandable_list_item_2, new String[]{
    LIST_NAME, LIST_UUID}, new int[]{android.R.id.text1, android.R.id.
    text2});
mGattServicesList.setAdapter(gattServiceAdapter);
}

private static IntentFilter makeGattUpdateIntentFilter()
{
final IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(BluetoothLeService.ACTION_GATT_CONNECTED);
intentFilter.addAction(BluetoothLeService.ACTION_GATT_DISCONNECTED);
intentFilter.addAction(BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED
    );
intentFilter.addAction(BluetoothLeService.ACTION_DATA_AVAILABLE);
return intentFilter;
}

}
```

Appendix A.12

StatsActivity.java

```
package com.example.owen.sleep_lab;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;
import java.util.function.DoublePredicate;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellValue;
import org.apache.poi.ss.usermodel.FormulaEvaluator;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.*;

public class StatsActivity extends AppCompatActivity
{
private String TAG = "StatsActivity";
GraphView graph1, graph2, graph3;
TextView textView;
int x, y, i;
private String[] path;
private String[] fileName;
private File[] listFile;
ArrayList<graph> uploadData;

@Override
protected void onCreate(Bundle savedInstanceState)
{
Scanner scanner = new Scanner(System.in);
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_stats);
graph1 = (GraphView) findViewById(R.id.graph1);
textView = (TextView) findViewById(R.id.textView);

LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint>();

//path ="C:\\Users\\owen\\Desktop\\sleep-lab-master\\sleep-lab\\app\\src
    \\main\\sampledata\\";
//readExcel(path);
}

/*private void readExcel(String filepath)
{
File inputFile = new File(filepath);

try
{
InputStream input = new FileInputStream(inputFile);
XSSFWorkbook workBook = new XSSFWorkbook(input);
```

```
XSSFSheet sheet = workBook.getSheetAt(0);
int rows = sheet.getPhysicalNumberOfRows();
FormulaEvaluator formulaEvaluator = workBook.getCreationHelper().
    createFormulaEvaluator();
StringBuilder sb = new StringBuilder();

for(int i = 0; i < rows; i++)
{
Row row = sheet.getRow(i);
int cellCount = row.getPhysicalNumberOfCells();

int j = 25;
String value = getCellAssString(row, j, formulaEvaluator);
sb.append(value + ", ");

}

}
catch(FileNotFoundException e)
{
Log.d(TAG, "File not Found");
}
catch(IOException e)
{
e.printStackTrace();
}
}

private String getCellAssString(Row row, int j, FormulaEvaluator
    formulaEvaluator)
{
String value = " ";
try
{
Cell cell = row.getCell(j);
CellValue cellValue = formulaEvaluator.evaluate(cell);
double numericValue = cellValue.getNumberValue();

value = " " + numericValue;
}
catch(NullPointerException e)
{
e.printStackTrace();

}
return value;
}*/

private void parseStringBuilder(StringBuilder sb)
{
```

```
String[] rows = sb.toString().split(":");

for(int d = 0; d < rows.length; d++)
{
String[] columns = rows[d].split(",");

try
{
double x = Double.parseDouble(columns[0]);
double y = Double.parseDouble(columns[1]);

uploadData.add(new graph(x, y));
}
catch(NumberFormatException e)
{
e.printStackTrace();
}
}

}
}
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center_horizontal"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.example.owen.sleep_lab.LoginActivity">

<!-- Login progress -->
<ProgressBar
android:id="@+id/login_progress"
style="?android:attr/progressBarStyleLarge"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="8dp"
android:visibility="gone" />

<ScrollView
android:id="@+id/login_form"
android:layout_width="match_parent"
android:layout_height="match_parent">

<LinearLayout
android:id="@+id/email_login_form"
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:orientation="vertical">

<android.support.design.widget.TextInputLayout
android:layout_width="match_parent"
android:layout_height="wrap_content">

<AutoCompleteTextView
android:id="@+id/email"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="@string/prompt_email"
android:inputType="textEmailAddress"
android:maxLines="1"
android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
android:layout_width="match_parent"
android:layout_height="wrap_content">

<EditText
android:id="@+id/password"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="@string/prompt_password"
android:imeActionId="6"
android:imeActionLabel="@string/action_sign_in_short"
android:imeOptions="actionUnspecified"
android:inputType="textPassword"
android:maxLines="1"
android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<Button
android:id="@+id/register"
style="?android:textAppearanceSmall"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"
android:text="@string/action_register"
android:textStyle="bold" />

<Button
android:id="@+id/sign_in"
style="?android:textAppearanceSmall"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"
```

```
android:text="@string/action_sign_in"
android:textStyle="bold" />

</LinearLayout>
</ScrollView>
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
    schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.owen.sleep_lab.ProfileActivity">

<LinearLayout
android:id="@+id/linear_layout"
android:layout_width="368dp"
android:layout_height="495dp"
android:orientation="vertical"
tools:layout_editor_absoluteX="8dp"
tools:layout_editor_absoluteY="8dp">

<TextView
android:id="@+id/profileTextView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="TextView"
android:textAlignment="center" />

<EditText
android:id="@+id/inputText"
android:layout_width="match_parent"
android:layout_height="80dp"
android:hint="Enter Message to for Database" />

<Button
android:id="@+id/sentToDb"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Send tO Database" />

<Button
android:id="@+id/chooseBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="choose Image" />

<Button
android:id="@+id/upLoadBtn"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Upload Image" />


<Button
android:id="@+id/logoutBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="logout" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal">

<Button
android:id="@+id/recordBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="New Recording" />

<Button
android:id="@+id/statBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Statistics" />

</LinearLayout>

</LinearLayout>
</android.support.constraint.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
    schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.owen.sleep_lab.RecordActivity">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:layout_editor_absoluteX="0dp"
tools:layout_editor_absoluteY="163dp">
```

```
<TextView
android:id="@+id/heartRateDisplay"
android:layout_width="match_parent"
android:layout_height="298dp"
android:layout_weight="1"
android:text="@string/heartRate"
android:textAlignment="center"
tools:layout_editor_absoluteX="158dp"
tools:layout_editor_absoluteY="15dp" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_weight="1"
android:orientation="horizontal">

<Button
android:id="@+id/startRecordBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Start Activity" />

<Button
android:id="@+id/stopRecordBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Stop Activity" />

<Button
android:id="@+id/scanBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="scan for devices "
tools:layout_editor_absoluteX="231dp"
tools:layout_editor_absoluteY="3dp" />
</LinearLayout>

<ListView

android:id="@+id/list"
android:layout_width="368dp"
android:layout_height="495dp"
tools:layout_editor_absoluteX="8dp"
tools:layout_editor_absoluteY="8dp" />

</LinearLayout>

</android.support.constraint.ConstraintLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
    schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.owen.sleep_lab.ProfileActivity">

<LinearLayout
android:id="@+id/linear_layout"
android:layout_width="368dp"
android:layout_height="495dp"
android:orientation="vertical"
tools:layout_editor_absoluteX="8dp"
tools:layout_editor_absoluteY="8dp">

<TextView
android:id="@+id/profileTextView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="TextView"
android:textAlignment="center" />

<EditText
android:id="@+id/inputText"
android:layout_width="match_parent"
android:layout_height="80dp"
android:hint="Enter Message to for Database" />

<Button
android:id="@+id/sentToDb"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Send tO Database" />

<Button
android:id="@+id/chooseBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="choose Image" />

<Button
android:id="@+id/upLoadBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Upload Image" />


<Button
android:id="@+id/logoutBtn"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="logout" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal">

<Button
android:id="@+id/recordBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="New Recording" />

<Button
android:id="@+id/statBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Statistics" />

</LinearLayout>

</LinearLayout>
</android.support.constraint.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
   schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".ScanActivity">

<!--<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="TextView" />
*/-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:orientation="horizontal">

</LinearLayout>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:orientation="vertical">

<Button
android:id="@+id/menu_connect"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="connect" />

<Button
android:id="@+id/menu_disconnect"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="disconnect"
tools:layout_editor_absoluteX="0dp"
tools:layout_editor_absoluteY="0dp" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:orientation="horizontal">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/label_device_address"
android:textSize="18sp" />

<Space
android:layout_width="5dp"
android:layout_height="wrap_content" />

<TextView
android:id="@+id/device_address"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textSize="18sp" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:orientation="horizontal">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:text="@string/label_state"
android:textSize="18sp" />

<Space
android:layout_width="5dp"
android:layout_height="wrap_content" />

<TextView
android:id="@+id/connection_state"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/disconnected"
android:textSize="18sp" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:orientation="horizontal">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/label_data"
android:textSize="18sp" />

<Space
android:layout_width="5dp"
android:layout_height="wrap_content" />

<TextView
android:id="@+id/data_value"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/no_data"
android:textSize="18sp" />
</LinearLayout>

<ExpandableListView
android:id="@+id/gatt_services_list"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</LinearLayout>
</android.support.constraint.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
    schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.owen.sleep_lab.StatsActivity">

<LinearLayout
android:layout_width="368dp"
android:layout_height="495dp"
android:orientation="vertical"
tools:layout_editor_absoluteX="8dp"
tools:layout_editor_absoluteY="8dp">

<com.jjoe64.graphview.GraphView
android:id="@+id/graph1"
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1" />

<TextView
android:id="@+id/textView"
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:text="" />

</LinearLayout>
</android.support.constraint.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<TextView android:id="@+id/device_name"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textSize="24dp"/>
<TextView android:id="@+id/device_address"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textSize="12dp"/>
</LinearLayout>

<resources>
<string name="app_name">sleep-lab</string>

<!-- Strings related to login -->
<string name="prompt_email">Email</string>
<string name="prompt_password">Password</string>
<string name="action_sign_in">Sign in</string>
<string name="action_register">Register</string>
<string name="action_sign_in_short">Sign in</string>
```

```
<string name="error_invalid_email">This email address is invalid</string>
<string name="error_invalid_password">This password is too short</string>
<string name="error_incorrect_password">This password is incorrect</
   string>
<string name="error_field_required">This field is required</string>
<string name="permission_rationale">"Contacts permissions are needed for
   providing email
completions."
</string>
<string name="heartRate">Heartrate Date Displays Here</string>
<string name="startBtn">Start new recording</string>
<string name="scanBtn">Scan bluetooth device</string>
<string name="title_activity_scan">ScanActivity</string>
<string name="ble_not_supported">BLE is not supported</string>
<string name="label_data">Data:</string>
<string name="label_device_address">Device address:</string>
<string name="label_state">State:</string>
<string name="no_data">No data</string>
<string name="title_devices">BLE Device Scan</string>
<string name="error_bluetooth_not_supported">Bluetooth not supported.</
   string>

<string name="unknown_device">Unknown device</string>
<string name="unknown_characteristic">Unknown characteristic</string>
<string name="unknown_service">Unknown service</string>

<string name="connected">Connected</string>
<string name="disconnected">Disconnected</string>

<string name="menu_connect">Connect</string>
<string name="menu_disconnect">Disconnect</string>
<string name="menu_scan">Scan</string>
<string name="menu_stop">Stop</string>
</resources>
```