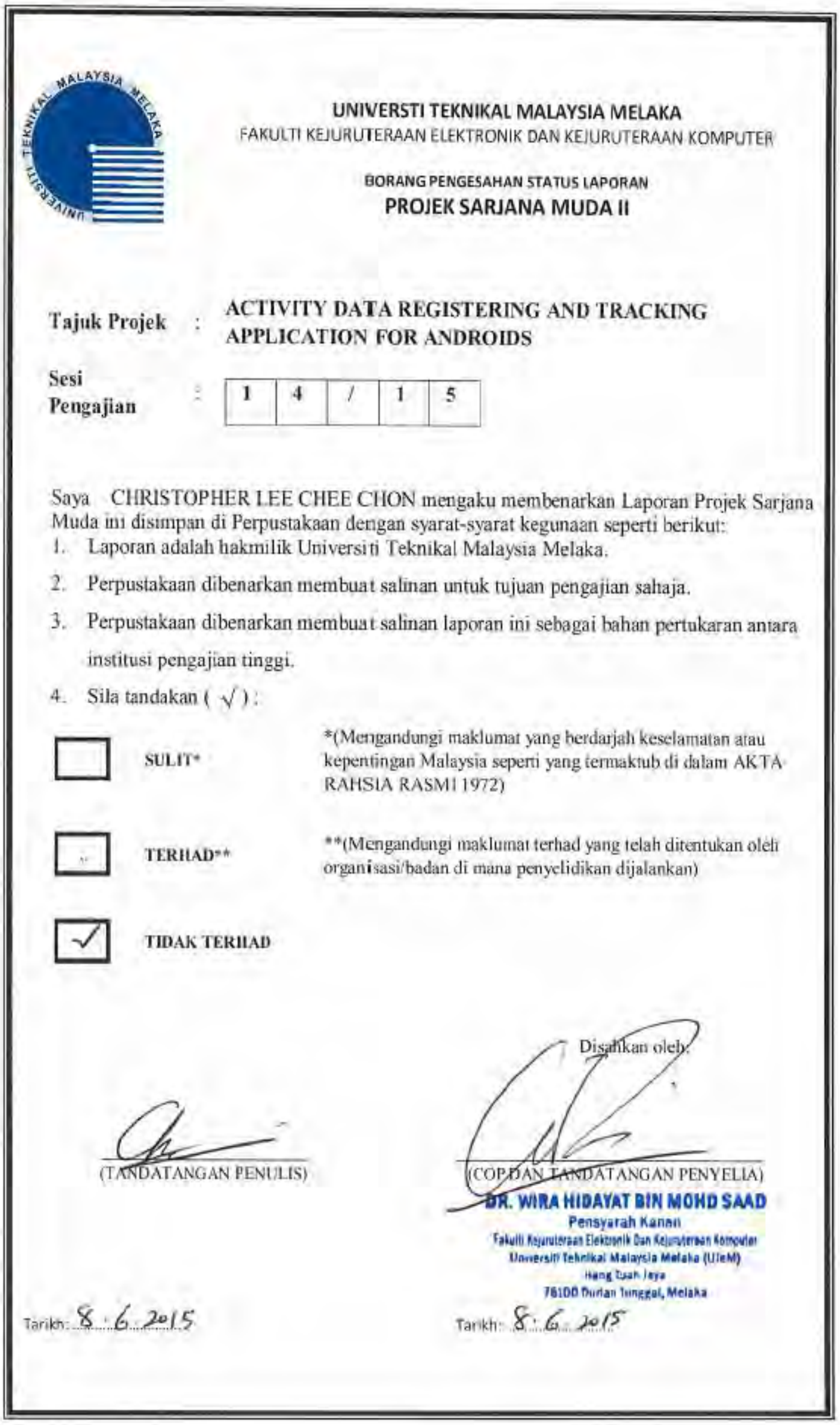
ACTIVITY DATA REGISTERING AND TRACKING APPLICATION FOR ANDROIDS

CHRISTOPHER LEE CHEE CHON

This Report Is Submitted in Partial Fulfillment of Requirements for the Bachelor Degree in Electronic Engineering (Computer Engineering)

Faculty of Electronics and Computer Engineering
University Teknikal Malaysia Melaka

JUNE 20



UNIVERSTI TEKNIKAL MALAYSIA MELAKA
FAKULTI KEJURUTERAAN ELEKTRONIK DAN KEJURUTERAAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek	:	**ACTIVITY DATA REGISTERING AND TRACKING APPLICATION FOR ANDROIDS**
Sesi Pengajian	:	1 4 / 1 5

Saya CHRISTOPHER LEE CHEE CHON mengaku membenarkan Laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (√) :

☐	SULIT*	*(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	TERHAD**	**(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
✓	TIDAK TERHAD	

Disahkan oleh:

[illegible]
(TANDATANGAN PENULIS)

[illegible]
(COP DAN TANDATANGAN PENYELIA)

DR. WIRA HIDAYAT BIN MOHD SAAD
Pensyarah Kanan
Fakulti Kejuruteraan Elektronik Dan Kejuruteraan Komputer
Universiti Teknikal Malaysia Melaka (UTeM)
Hang Tuah Jaya
76100 Durian Tunggal, Melaka

Tarikh: 8.6.2015

Tarikh: 8.6.2015

"I hereby declare that I have read this report and in my opinion this report is sufficient in terms of the scope and quality for the award the Bachelor of Electronics Engineering (Computer Engineering) With Honours"

Signature : [illegible]

Name : DR. WIRA HIDAYAT BIN MOHD SAAD

Date : 8-6-2015

"I hereby declared that this report entitled Activity Data Registering and Tracking Application for Androids is a result of my own work except for notes that have been cited clearly in the references"

Signature	: [illegible]................................
Student Name	: CHRISTOPHER LEE CHEE CHON
Date	: 8-6-2015

DEDICATION

First, I would like to specially dedicate to my project supervisor Dr. Wira Hidayat Bin Mohd Saad, who gives me a lot of guidance and advices throughout this project until successfully. Under his guidance and help, I have successfully developed and achieved the completion of this project.

Besides that, I would also like to thank my second project supervisor En. Khairul Muzzammil Bin Saipullah who gave me an idea and advices to do this project. In addition, I would also like to thanks my beloved family and friends who have given me their encouragement and support for me to complete this project.

"THANK YOU"

ACKNOWLEDGEMENT

First of all, I would like to thank my main project supervisor Dr. Wira Hidayat Bin Mohd Saad, who has given me a lot of guidance and advices throughout this project. He also gives me a lot of support and taking care of me from the very beginning of PSM 1 until I complete this project. He has given me the awareness to find my problem solution and with his kindness and tolerance that give me spirit to be more serious and focus on finishing the project. Without his help and his contribution to this project, I do not think this project can be completely successful.

Besides that, I also want to contribute my acknowledgement to my second supervisor En Khairul Muzzammil Bin Saipullah who also gave me a lot of supports and ideas in term to help me finished this project.

Lastly, I would also like to thank my beloved parent who fully support me and provided me everything that they could do for me in order for me to finish this project. I really appreciate their help that they gave to me. Also not to forget my friends who helping me by providing me some idea and suggestion for this project. Thank you very much and I really appreciate with all of your help.

"THANK YOU ALL"

ABSTRACT

In this modern world, smartphone is already becoming one of the essential elements in everyone's life. It is hard for someone to leave their smartphone from their hand, even when they are eating. They are busying on chatting with friends, uploading their photo, updating their status and more. This is a trend of the new generation called ‘Gen Y’. They can do a lot of things with their smartphone and in the other perspective to see it; we can say that they cannot live without their smartphone in their daily life. This shows that how smartphone is important for us in our everyday life. This study is proposed an Android Smartphone application to provide a facility for the researcher to do a data collection on the basic data available from the mobile device for different type of activity. This data can be used in later to develop a best machine learning algorithm for a better activity detection. In addition, the project was implement on the Eclipse development software in order to develop the Bluetooth communication between the apps with the heart rate monitor device, data mining for 9DOF acceleration sensor data, graphic user interface and so on. Lastly, when this application has been developed completely, it will publish on the google play store to allow user to download it for free.

ABSTRAK

Dalam dunia moden ini, smartphone sudah menjadi salah satu bahagian yang penting dalam kehidupan untuk semua orang. Adalah sukar bagi seseorang untuk meninggalkan telefon pintar mereka daripada tangan mereka walaupun mereka sedang makan. Mereka sangat sibuk pada berbual dengan rakan, memuat naik gambar mereka, mengemas kini status mereka dan banyak lagi degan smartphone mereka. Ini adalah trend generasi baru yang dikenali sebagai Gen Y. Projek ini membincangkan mengenai membangunkan program aplikasi latihan kecergasan untuk android. Fokus utama mengenai perkara ini permohonan latihan kecergasan adalah untuk tujuan perlumbaan marathon. Dengan melaksanakan permohonan itu dengan sistem pemantauan kadar jantung untuk membiarkan permohonan itu dapat memantau kadar jantung pengguna semasa menjalankan latihan mereka. Dengan sistem pemantauan kadar jantung, permohonan itu dapat menjana satu program latihan yang paling sesuai untuk pengguna berdasarkan menguatkan keupayaan hati mereka. Kadar jantung memantau permohonan pengguna dengan menggunakan peranti monitor kadar jantung yang merupakan hasil HxM Smart oleh Zephyr Syarikat. Di samping itu, projek itu dilaksanakan pada perisian pembangunan Eclipse dalam usaha untuk membangunkan komunikasi Bluetooth antara aplikasi dengan peranti monitor kadar jantung, antara muka pengguna grafik dan sebagainya. Akhirnya, aplikasi ini adalah pembangunan dan diterbitkan di google play untuk membiarkan daripada muat turun dan menggunakannya.

TABLE OF CONTENT

CONTENTS

LIST OF FIGURE

LIST OF APPENDIXES

CHAPTER I

INTRODUCTION

1.1 Project Introduction

In this modern world, smartphone is already becoming one of the essential elements in everyone's life. It is hard for someone to leave their smartphone from their hand, even when they are eating. They are busying on chatting with friends, uploading their photo, updating their status and more. This is a trend of the new generation called Gen Y. They can do a lot of things with their smartphone and in the other perspective to see it; we can say that they cannot live without their smartphone in their daily life. This proves that how smartphone is important for us in our daily life.

Besides that, not entirely for entertainment purpose, there are likewise a bunch of parts that our smart phone. For example, in the field of education, business, finance, health and fitness and so on. On that point are a bunch of parts that our smartphone can serve when the technology become more modern. In summation, the health science and data mining researchers also use their smartphone to collect some data for their work. For instance, they can track motion sensor data such as accelerometer, gyroscope and gravity sensor data for different type of activity on their smartphone to develop their own algorithm for their project to automatically detecting current user activity by using neural network system.

This work proposes an Android Smartphone application to provide a facility for the researcher to answer a data collection of the basic available data from the mobile device for different type of natural action. This information can be applied subsequently to get a best machine learning algorithm for a better activity detection. Why this application was developed is because not all the researchers have such knowledge to develop their own application to help them to collect data. Thus by applying this application they can acquire what they demand. Lastly, when this application has been developed completely, it is published on the google play store to allow user to download it for free.

In overall, this application involves a software implementation and hardware configuration. The hardware use is the heart rate monitor device model HxM Smart from Zephyr. This device is able to assess the user's heart rate and transmit the signal to the application by using Bluetooth Smart communication. For the software implementation on developing the application, it is developed by using Eclipse software.

1.2 Problem Statement

This application provides nine degrees of freedom (9DOF) accelerometer data. This data will represent an information on the prediction of what are the current activity that the user is doing while he or she carry the smart phone. TYPE_SIGNIFICANT_MOTION is a build in function to automatically extract the natural action of the user, but it is limited to running, cycling and some other simple activity only. Thus by getting 9DOF of accelerometer data, developers can utilize their own algorithm for the detection for more flexibility compare by using the built in mapping. These sensors are often available on the Android device such as android smartphone/tablet, android watch and so on. These devices are carrying a huge market value nowadays and that this is the cause why we prefer this platform.

1.3 Objective

The objective of this task is to produce an Android application to offer an easy facility for the health science and data mining researchers to meet their research information from the application for their inquiry. In order to achieve that, the objective is written as below:

- To integrate the open API Bluetooth heart rate device with the developed application.
- To logged the accelerometer, gyroscope, gravity sensors and GPS positioning data in the smartphone memory.
- To create a GUI to display all the data.

1.4 Scope of Project

These tasks consist of preparing an Android application to offer an easy facility for the health science and data processing analyzers to collect their analysis data from the appliance for his or her research. This application can measure the user heart rate while carry out the activity by using a heart rate monitor–HxM Smart from Zephyr [1]. Besides that, user can get the accelerometer, gyroscope, gravity sensor data and GPS positioning by using this application. There have three categories of sensors that we can use in application such as motion sensors, environmental sensors and position sensors, but this application is only using the motion sensors data only that had been provided by the android. The device is using an ECG monitoring system to assess the user heart rate while training. Electrocardiogram (ECG) is a method to record the electrical activity of the heart. By using this way, the device can measure the heart rate and record it [2]. In addition, the heart rate monitoring device is connected to the application on our smartphone via Bluetooth while the training is in session. The embedded system for the application will be written based on the Java [3], C and C++ high-level computing language on Android 4 application [4, 5].

1.5 Importance of the project

This study is being developed with the objective to provide a simple program for the health science and data mining researchers to accumulate their examination data from the application for their exploration. Besides that, user also can monitor their heart rate when using this application. In addition, the application will be published for free after it is completely developed so that Android users can download the application and utilizing it. By publishing this application on Google market, it can help in user data collection for future development purpose. When the software is matured enough, charging for the cost of the software is plausible.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

This chapter describes about some previous study for using motion sensor that inside our Android smartphone in any research purpose and some activity tracking purpose. By these research we can conclude that these motion sensor data are very useful for us to develop our own algorithm to develop our project. Besides that, there have also some research discuss about the heart rate monitoring on android smartphone device that will implement to this project.

2.2 Previous research for using motion sensors on android smartphone device

On that point are a great deal of subroutines can be performed by using motion detectors on our Android smartphone device. For instance, using motion sensors to predict the activity carry out by the user when they extend the device, location tracking by using accelerometer sensor, step counter to count the distance travel for the user and so along. This is also the motivation for me to develop this application. This application can provide the facility for these researchers to collect these sensor data to train their own project by applying their own algorithm.

2.2.1 Physical Activity Recognition by Using Smartphone Sensors

In this paper, *Muhammad Shoaib et.al.* Investigate the accelerometer, gyroscope and magnetometer sensors in our smartphone for the activity recognition purpose. They evaluate four parts of body positions by using seven classifiers for recognizing six physical activities by applying these three sensors [6]. They indicate that the accelerometer and gyroscope sensors can well performance for making the recognition the physical actions. Nevertheless, most of the events, the gyroscope sensors do not better the recognition accuracy when using it in concert with the accelerometer sensors, but it can supply a reasonable result when using it entirely. On the other hand, the results from using a magnetometer are not well encouraging when they are utilizing it to train the classifiers depend on the orientation of the smartphone device. At the final stage, they conclude that in that respect are a difficult to make a closing statement to state that which sensor are performs well compared with others. This is because the identification performance of these sensors is different depending on smartphone’s orientation. So by reading this report, we can see that how important about the motion sensor data for these researchers to utilize it to get their own algorithm to accomplish their aim.

2.2.2 The Statistical Recognition of Walking, Jogging, and Running Using Smartphone Accelerometers

On that point is another similar field that discusses about the advantages for using smartphone accelerometers for the statistical recognition of physical activity's purpose. *Ekachai Thammasat et.al.* Was discuss about the statistically recognition for walking, trotting, and running activities by using accelerometers that had enforced in our smartphone [7]. By applying the sensors on the smartphone, the user will no need to carry out another device anymore to pass over their activity. The advantages of this study are to observe the remotely for any cause of the user by utilizing this application at all the time, so that the user can handle the appropriate plan of practice, monitors their energy use and evaluate their calorie compensation. Figure 2.1 shows the graphical results for the application to track the different type of activities carry out by the user.

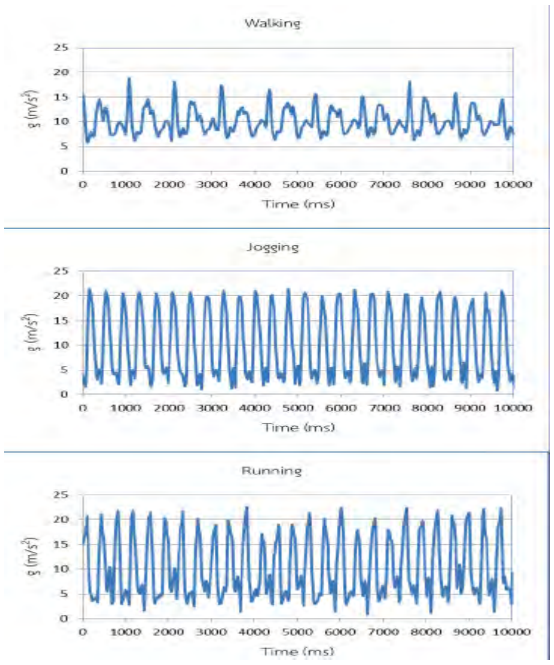


Figure 2.1: Acceleration of walking, jogging, and running against time.

2.2.3 Fall Detection System Using Accelerometer and Gyroscope Based on Smartphone

Beside that the physical activity recognition, accelerometer and gyroscope sensor data on our smartphone also can use to develop a fall detection system. *Arkham Zahri Rakhman et.al*. develop an fall detection system by using these motion sensors [8]. They stated that the accelerometer and gyro sensors that inside our smartphone can offer a more accurate effect of fall detection motion. There will have an automatic call for user's family members when the user's had any fatal condition when using this application. Besides that, this study also describes the condition of people between falls and activity daily living. By taking the sensor data for accelerometer and gyroscope, they get their own algorithm for detecting a failing condition for the user. With this fall detection system, it can be avoided, many cases of accidentally happened in the household when there is just the older people alone in the mansion. This arrangement can save their spirits.

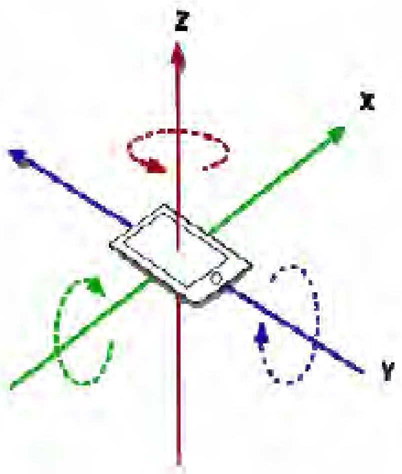


Figure 2.2: Axis of the gyroscope and accelerometer.

2.2.4 Movement Pattern Recognition through Smartphone's Accelerometer

In addition, presently there experienced an investigation suggest that sensor allowed smartphone's have grown to be the popular system with regard to scientists because of their capability to gather as well as procedure big amounts associated with information, therefore making brand new possibilities with regard to revolutionary android application, through *Armir Bujari et.al. 2012* [9]. This may be the primary inspiration with this application which experienced created. In the following paragraphs, additionally they talk about concerning the motion design acknowledgement within daily depending on city road conduct. Like an example, these people limit from realizing a scenario whenever a pedestrian cease, traversing the road dominated with a visitors gentle. They make use of these types of information in the accelerometer sensor in the smartphone gadget to accomplish their own research.

2.2.5 Smartphone Indoor Localization with Accelerometer and Gyroscope

Other that physical activity recognition and fall detection system, accelerometer and gyroscope sensor data also can use to develop an indoor localization system. *Hui-Huang Hsu et.al, 2014* develop a smartphone indoor positioning application by using accelerometer and gyroscope sensor [10]. They build a pedestrian dead reckoning method in their application. The calibration points for the localized application are checked off both on the ground floor and on the map of the application. The user first need to get a calibration mark and then stand on it and then face to the correct direction. After that the user need to place the Android smartphone on the top of the calibration point. When the user start to strike, the android icon on the application will also run on the map following with the real time estimates of the orientation change for each step from accelerometer and gyroscope data respectively. The preliminary outcomes for their application in walking distance and orientation estimation show high accuracy solutions. This shows that another perspective of the vantages of the motion sensor data on our smartphone device.

2.2.6 Initial Test on the Use of GPS and Sensor Data of Modern Smartphones for Vehicle Tracking in Dense High Rise Environments

There's an additional research which is designed to explore the actual mixing of the actual built-in GPS NAVIGATION as well as sensor information upon mobile phones with regard to localization within thick city conditions exactly where frequently satellite television indicators tend to be blocked through high structures or even bigger buildings, leading to the actual in adequate amount in the event that GNSS dimension information with regard to the prosperous placement dedication [11]. The actual document additionally talks about touching the features from the information results from the electronic compass as well as accelerometer with regards to the actual alignment and also the telephone actions offered with this text file. Side by side, they'll contain the actual GPS NAVIGATION, an electronic compass as well as accelerometer for his or her automobile monitoring programs. This particular document exhibits an additional way of while using movement sensor which within smartphone to build up a good Google android application.

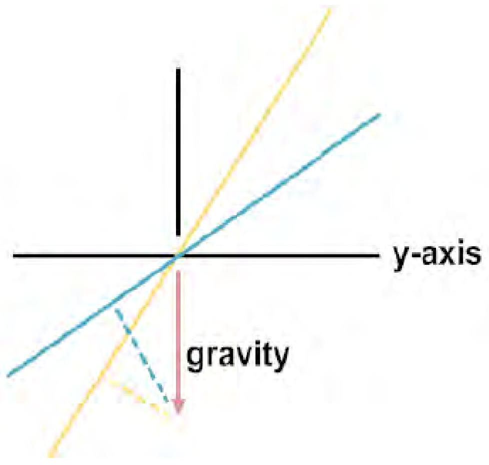


Figure 2.3: The gravity effect on the accelerometer output when the y-axis is at different elevation angles.

2.3 Previous research for heart rate while exercising

This project not only tracking about the motion sensors, data in our smartphone, it also tracks the user's heart rate when user using this application connected with the HxM heart rate monitor device. Our heartbeat frequency or pulse is measured by counting the number of times in our heart beating per minute, normally we called it BPM. Difference person will have a different normal BPM and the BPM is also differed while resting or exercising. Other than that, BPM is also depending on the gender and age.

A professor of medicine and cardiology at the New York University School of Medicine in New York City and a volunteer for the American Heart Association, Richard Stein, M.D., said that "When our age increases, by monitor the changing of our heart rate we can observe our heart condition and then we can do some health prevention," [12].

The value of the heart rate can be measured by counting the number of our heartbeat frequency in one minute by just placing our finger over our pulse. There are four general places to let us find our pulse which are at wrists, inside out elbow, side of our neck and the top of the foot. While resting, our heart wills pumping our blood at the lowest amount of blood that we need, this is our resting heart rate. It is because while we are not exercising, the heart just pumps enough to support us for our normal activity. If we are exercising, we will need more oxygen to generate more energy and more oxygen that is needed by our brain. This can cause the BPM to be increased. Normally our BPM is in between 60 to 100 while we are resting, said by Stein [12]. Then, during physical activity, it is usually does not change the resting BPM much as it normally changes within 40 BPM if we exercise regularly. For a person who does less exercise, the BPM will increase between 60 to 100 while they are doing physical activity. This knowledge of this behavior is very important to develop the heart rate monitoring application for Android. Besides that, by exercising with the target heart rate zone (THR), it helps us to burn more calories even quickly. So below are some research study that related about the information with heart rate.

2.3.1 Heart Rate and SpO2 Level Monitoring System

In this paper, *Sornanathan et.al.* Discuss about measure and monitor the user's heart rate and user's oxygen saturation level by using a pulse oximeter during exercising. In year 2010, there are a team at RMIT University, Australia doing a research with monitoring and analyzing physiological signal during fitness activity. In that study, they monitor and measure the user's heart rate and oxygen saturation by using a pulse oximeter by implementing CaszOxiSys [13]. CaszOxiSys is a hardware which is consists of a pair of physiological sensor which is developed by using Light-Emitting Diode (LED) and photodiode. The CaszOxiSys is using a wireless communication method which is Bluetooth communication technique to communicate with user laptop or computer to help analyze the physiological signal data and convert it into the heart rate and the oxygen saturation value for the user during their fitness activity. Besides that, there is a discussion on how to calculate the targeted heart rate zone depended on target's gender, resting heart rate maximum heart rate, age and strength of exercising level for the user during exercising [13]. This information is very important in this study to make sure that the method that we develop in this project was proven.

2.3.2 Wearable Heart Rate Monitoring System

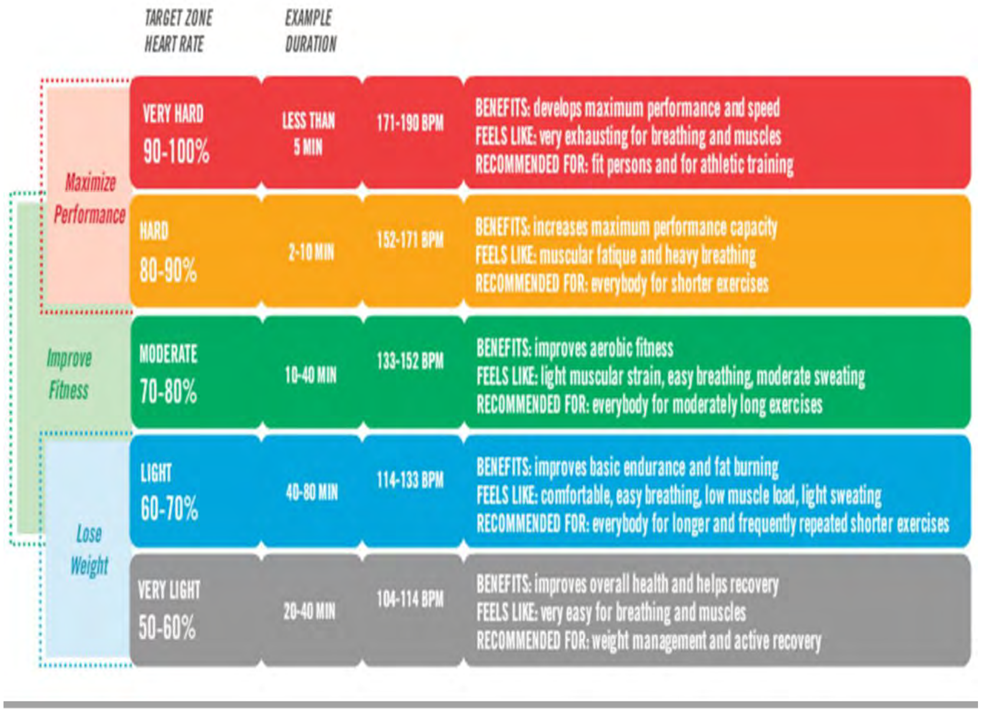
In addition, there is another study done by *Yaofeng Wen et.al*. that discuss about a heart rate monitoring system in dynamic movements for example running. This wearable system was developed by using a wireless heart rate monitor sensor with a single waistline-wear triaxial wireless accelerometer sensor to build out a wearable system with the purpose to measure the user's heart rate. This design is depending on the speedup thresholds and heart rate average value to measure the user's heart status during fitness training. This method is tested with ten continuous physical activities of the user which are sitting, standing, walking, running, falling and composite exercising [14]. Based on their finding, they suggest that the heart rate reading is depending on difference physical activity instead of depending on running activity only.

2.3.3 Heart Rate Zone

Difference equation will provide us with a different target heart rate zone for resting and exercising. Based on the THR that we had calculated, we can monitor our heart rate condition during exercising. In year 1970, there is a young physician which is Dr. William Haskell and his mentor Dr. Samuel Fox that has conducted a service program on heart disease and by that they manage to generate a common formula to calculate the maximum heart rates of patients [15]. The maximum heart rate can be divided into five categories of heart rate zone and this five heart rate zone had been used up to until now very widely. Training in different heart rate zone will help us in improving our performance in many different purposes. There is five color zone for our heart rate, which is a very light zone in gray color, light shone in blue color, moderate zone in green color, hard zone in orange color and the last one is a very hard zone in red color. For grey and blue zone are considered to lose weight zone. We can carry out our physical training based on this two zone to improve our basic endurance and help us to burn our fat during the training. Besides that, training in these zones will help us to improve overall health, it will give us to feel comfortable, easy breathing and light sweating. Normally, training in these two heart rate zone is for losing weight purpose and it should be conducted in 40 to 80 minute duration to achieve the purpose and because of it is not very hard for our heart to handle it, it can be done in the longer duration.

The next heart rate zone is a moderate zone which is in green color. Training in this zone can help us improve our aerobic fitness performance. It feels like a light muscular strain and moderate sweating when we carry out the training. In addition, training in this zone is for fitness purpose which means the person who want to build out their muscles or want to be looking more energetic then they should be trained in this zone with the duration 10 to 40 minutes. Training in this zone should not be more than 40 minutes, this is because in this zone, our heart rate BPM is maintained at 133 to 152 per minute, so if we carry out the training in this zone more than 40 minutes then it would injure our heart. To avoid overloading our heart, we should aware with the duration and should not over-training.

The following two heart rate zone is the hard zone and a very hard zone which is in orange and red in color. Training in these zones is to optimize our performance and it is recommended for athletic training and shorter exercise. This is because training in these zones should not be more than five minutes and then it will cause us feel like very exhausting to breathing. If our training in this zone more than five minutes and if we does not exercise regularly, training in these two zones will cause fatal injuries and it is very dangerous to do training in this zone alone.



	TARGET ZONE HEART RATE	EXAMPLE DURATION		
Maximize Performance	VERY HARD 90-100%	LESS THAN 5 MIN	171-190 BPM	BENEFITS: develops maximum performance and speed FEELS LIKE: very exhausting for breathing and muscles RECOMMENDED FOR: fit persons and for athletic training
Maximize Performance	HARD 80-90%	2-10 MIN	152-171 BPM	BENEFITS: increases maximum performance capacity FEELS LIKE: muscular fatique and heavy breathing RECOMMENDED FOR: everybody for shorter exercises
Improve Fitness	MODERATE 70-80%	10-40 MIN	133-152 BPM	BENEFITS: improves aerobic fitness FEELS LIKE: light muscular strain, easy breathing, moderate sweating RECOMMENDED FOR: everybody for moderately long exercises
Lose Weight	LIGHT 60-70%	40-80 MIN	114-133 BPM	BENEFITS: improves basic endurance and fat burning FEELS LIKE: comfortable, easy breathing, low muscle load, light sweating RECOMMENDED FOR: everybody for longer and frequently repeated shorter exercises
Lose Weight	VERY LIGHT 50-60%	20-40 MIN	104-114 BPM	BENEFITS: improves overall health and helps recovery FEELS LIKE: very easy for breathing and muscles RECOMMENDED FOR: weight management and active recovery

Figure 2.4: Heart rate zones with training suggestion.

CHAPTER III

METHODOLOGY

3.1 Introduction

This study consists of two parts of sub-project which is hardware implementation and software implementation. For hardware implementation is the Bluetooth heart rate monitor device by HxM Smart from Zephys whereas the software implementation is my android application development. For hardware implementation I need to implement a Bluetooth communication between the device and my apps and then the software implementation is discussed about the GUI for the apps for how to read the sensor data reading and GPS position points and then export it into a text file in the memory of the device.

3.2 Hardware Implementation

For hardware implementation, I would wish to discuss about how the hardware was gone through. The hardware for this study is a Bluetooth heart rate monitor device produce by HxM Smart from Zephys. Before implement it, in that respect are some specification for the device that we should know that, for example the frequency of the Bluetooth using for the device so that I can pass it with the application via Bluetooth communication. The Zephyr HxM Smart Heart Rate Monitor device using a Bluetooth 4.0 to communicate with our smartphone and it is usable for both iPhone and certain Android based Smartphones. The device does not have any button or shift to allow user to twist it on and off, it will turn on automatically when we assume the device on our chest with the provided chest strap. Later on we put on the device, the sensor of the device will detect the ECG from our muscle and then it will turn it on the device. Below are the specification for the device:

HR Range:	25 – 240 BPM
Battery Life:	150 hours
Transmit Range:	10m
Frequency:	2.4 – 2.4835G Hz

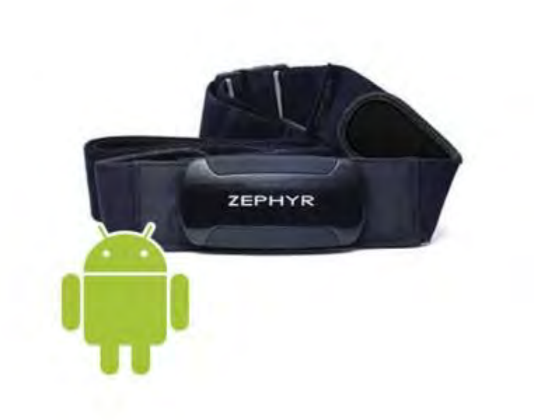


Figure 3.1.: Heart Rate Monitor Device

When the hardware implementation was completed, we will proceed it to the coding implementation part which is to develop the Bluetooth communication between the device and the application, it will discuss in more detail in the chapter four.

3.2.1 Development flow chart for hardware implementation

The development flow chart for the Bluetooth communication for the device are shown as follow:

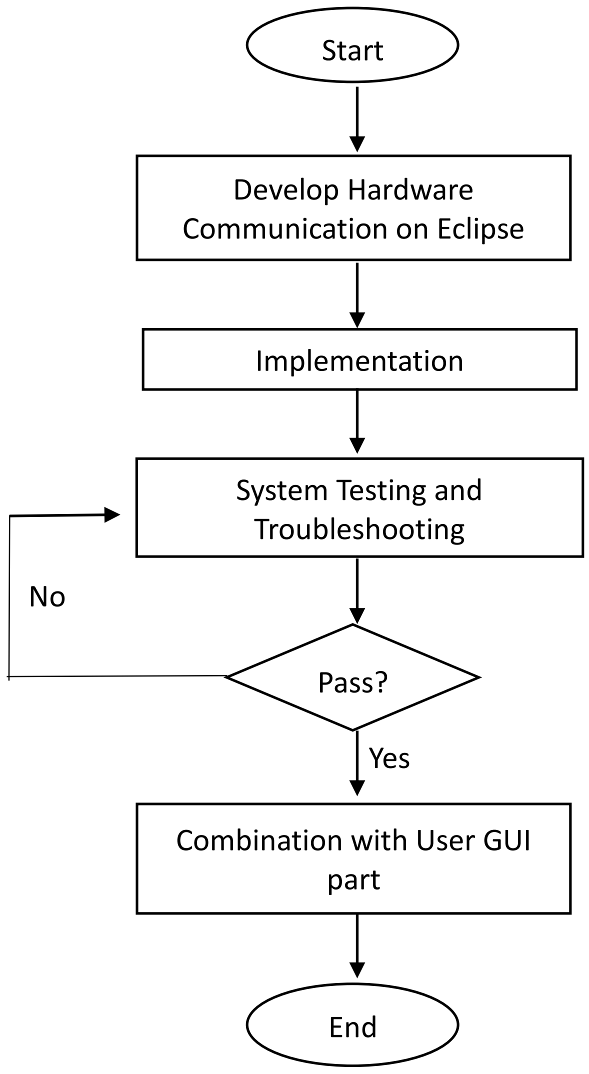
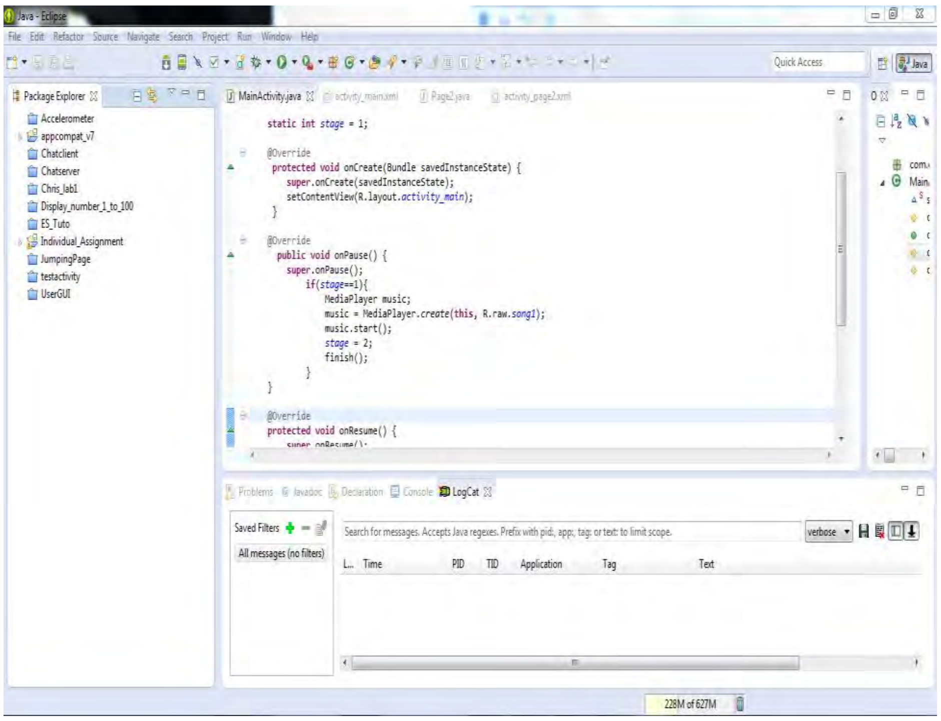

Start
Develop Hardware Communication on Eclipse
Implementation
System Testing and Troubleshooting
No
Pass?
Yes
Combination with User GUI part
End

Figure 3.2: Development flow chart of heart rate monitor device.

Figure 3.2 above is the flow chart of developing the heart rate monitor device on Eclipse software. Starting the Bluetooth communication for the heart rate monitor device was developed on Eclipse. After completing the coding development part then it only move to implementation part and then will test the system on the smartphone or virtual device on laptop until it does not contain any error and can work perfectly then only will combine it with the user GUI part and it has only become my application on android application.

3.3 Software Implementation

For software implementation part, Eclipse Juno developing software was chosen to develop the training program android application. This is because Eclipse is a very powerful and commercially-friendly, open source develops software that can help us to develop our own software. It provides us an open development underlying computer system on which application programs can run consists of extendable structure, tools and runtimes intern to help us in building and deploying software across the lifecycles [16]. By using eclipse Juno, it can help me in order to develop both hardware and software implementation for the application.



```
    static int stage = 1;

    @Override
     protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
     }

    @Override
      public void onPause() {
        super.onPause();
            if(stage==1){
                MediaPlayer music;
                music = MediaPlayer.create(this, R.raw.song1);
                music.start();
                stage = 2;
                finish();
            }
    }

    @Override
    protected void onResume() {
```

Figure 3.3: Eclipse Juno Interface

3.3.1 Development flow chart for software implementation

The development flow chart for software implementation are shown as follow:

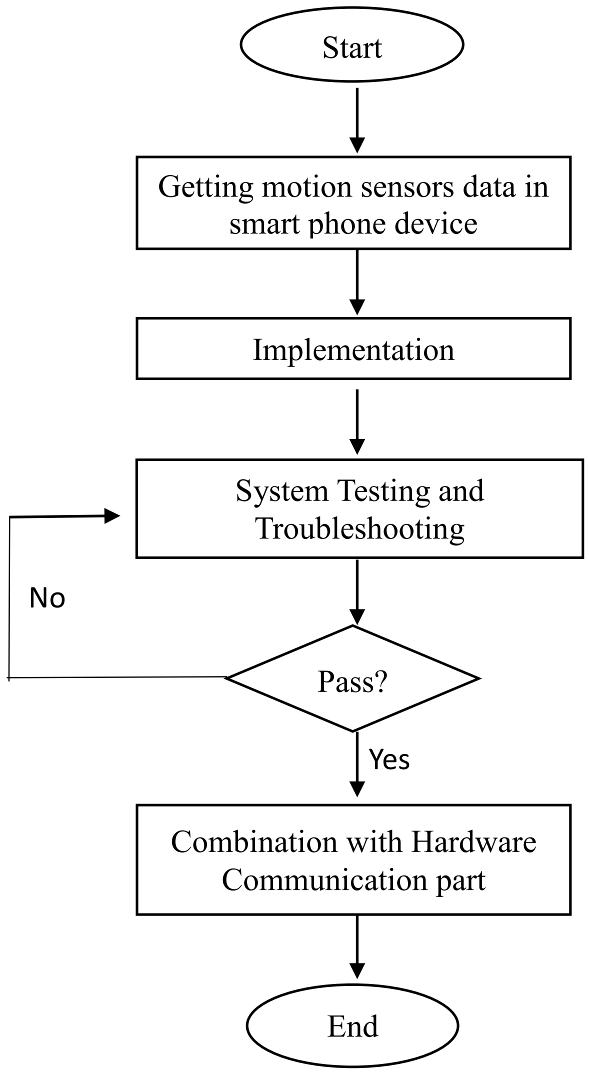

Start
Getting motion sensors data in smart phone device
Implementation
System Testing and Troubleshooting
No
Pass?
Yes
Combination with Hardware Communication part
End

Figure 3.4: Development flow chart of software implementation

For software implementation, a GUI was designed to show user's heart rate and instant speed when it had successfully connected with the HxM heart rate device. After that, we try to add the sensor data collecting function into the GUI and then implement the Google map function. Lastly, we try to save it into a text file in the device memory so that user can use it for research purpose.

3.3.2 UML Class Diagram for Application GUI Design

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes. UML class diagrams are the mainstay of the object-oriented analysis and design tools. UML class diagrams show the classes of the system, their interrelationships which are included inheritance, aggregation, and association and the operations and attributes of the classes [17]. Class diagrams are used for a wide variety of purposes, including both conceptual and domain modelling and the detailed design modelling. Figure 3.5 shows the UML class diagram that I had use for design my application GUI:

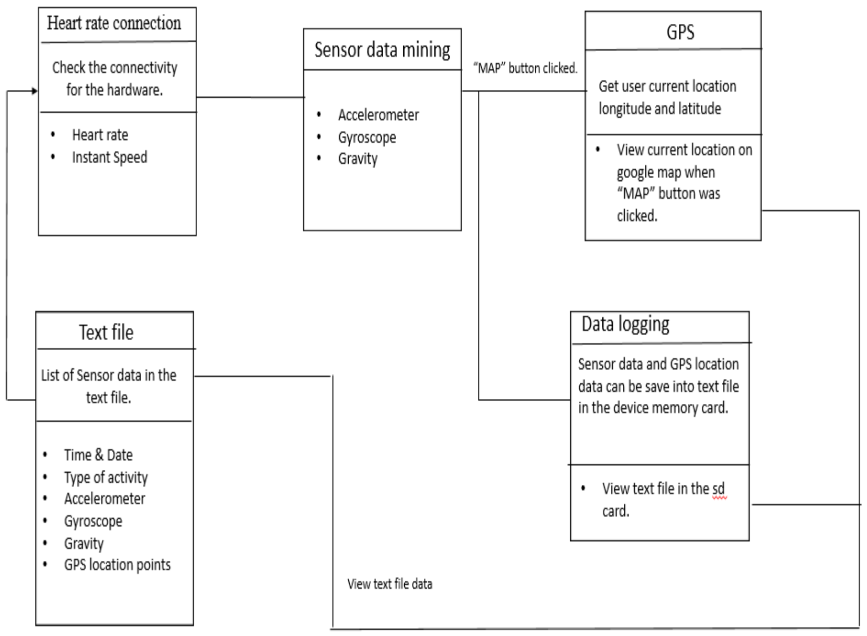

Heart rate connection
Check the connectivity for the hardware.
• Heart rate
• Instant Speed
Sensor data mining
• Accelerometer
• Gyroscope
• Gravity
"MAP" button clicked.
GPS
Get user current location longitude and latitude
• View current location on google map when "MAP" button was clicked.
Text file
List of Sensor data in the text file.
• Time & Date
• Type of activity
• Accelerometer
• Gyroscope
• Gravity
• GPS location points
Data logging
Sensor data and GPS location data can be save into text file in the device memory card.
• View text file in the sd card.
View text file data

Figure 3.5: UML class diagram.

The UML class diagram show that the relationship between all the classes of the application. Firstly, there is a heart rate connection class for the application to use to check the connectivity between the HxM heart rate monitor devices with the application. If they are connected, the value of the user's heart rate and instant speed will be displayed on the GUI layout of the application. After that, there is also have a sensors, data mining class used to collect the motion sensors on the smartphone device such as Accelerometer, Gyroscope and Gravity sensors. When the “MAP” button was clicked, the GUI of the application will jump go to the google map display layout and then the user is able to see their current location on the google map. Besides that, they also can get their location longitude and latitude points on the GUI layout. There also have other classes for users to record their data. When “Start record” was clicked, all the data will save in a text file on the memory card on the device. This function lets these researchers can get used all the data in term to develop their own algorithm.

3.4 Google Map API V2

Google provides us a library via google play for using google maps in our application. Google map API V2 allow us to place the locations with custom markers, augment the map data with image overlays and embed one or more maps as fragments in our application. On that point are a bunch of parts that we can implement in our application by using the google map android api v2. For instance, add maps to our apps, customize the map, check the user’s position and add street view in our android application [18]. I am using this google map android api v2 to help me implement the Google map function in my application to become the user’s current location and then display it with a marker on the Google map. It will discuss in detail in the discussion part later.

3.5 Android Sensor EventListener

Most of the android devices have built-in sensors that measure motion, orientation and various environmental condition. The android platform provides several sensors that allow us to monitor our motion of the device such as motion detectors, environmental sensors and attitude sensors. Besides that, Android provides sensor manager and sensor classes to apply the sensors in our application. In order to use these sensors, we need to instantiate the object of the sensormanager class in our coding [19]. This will discuss in detail in the discussion part later. In this case, this application only using the motion sensors only and it had been declared early on the scope part.

3.6 Overall Project Flow

In this part I would like to discuss about the flow of our study. It is discussed how this application has been done and how I work with step by step to complete my job. The flow chart below is the overall project flow of this final year project. Firstly, I divided the application into two mini part of the project. The first part is for Bluetooth communication between the heart rate monitor device and the android application and then another part is the software implementation which is user GUI and data mining development. For the first part, I focus on the Bluetooth communication method which is simple developing an Android application on eclipse software and then develop the Bluetooth communication. After finishing the first part I only move on the second part which is a data mining development and GUI designed part. This part is also done by using eclipse, developing software. After completing the two parts of the mini part of the application, then it only can combine it together to become a one complete training program application and then only can install it on the smartphone device and then doing the testing and troubleshooting of the application. Figure 3.6 are the flow chart about the overall project flow.

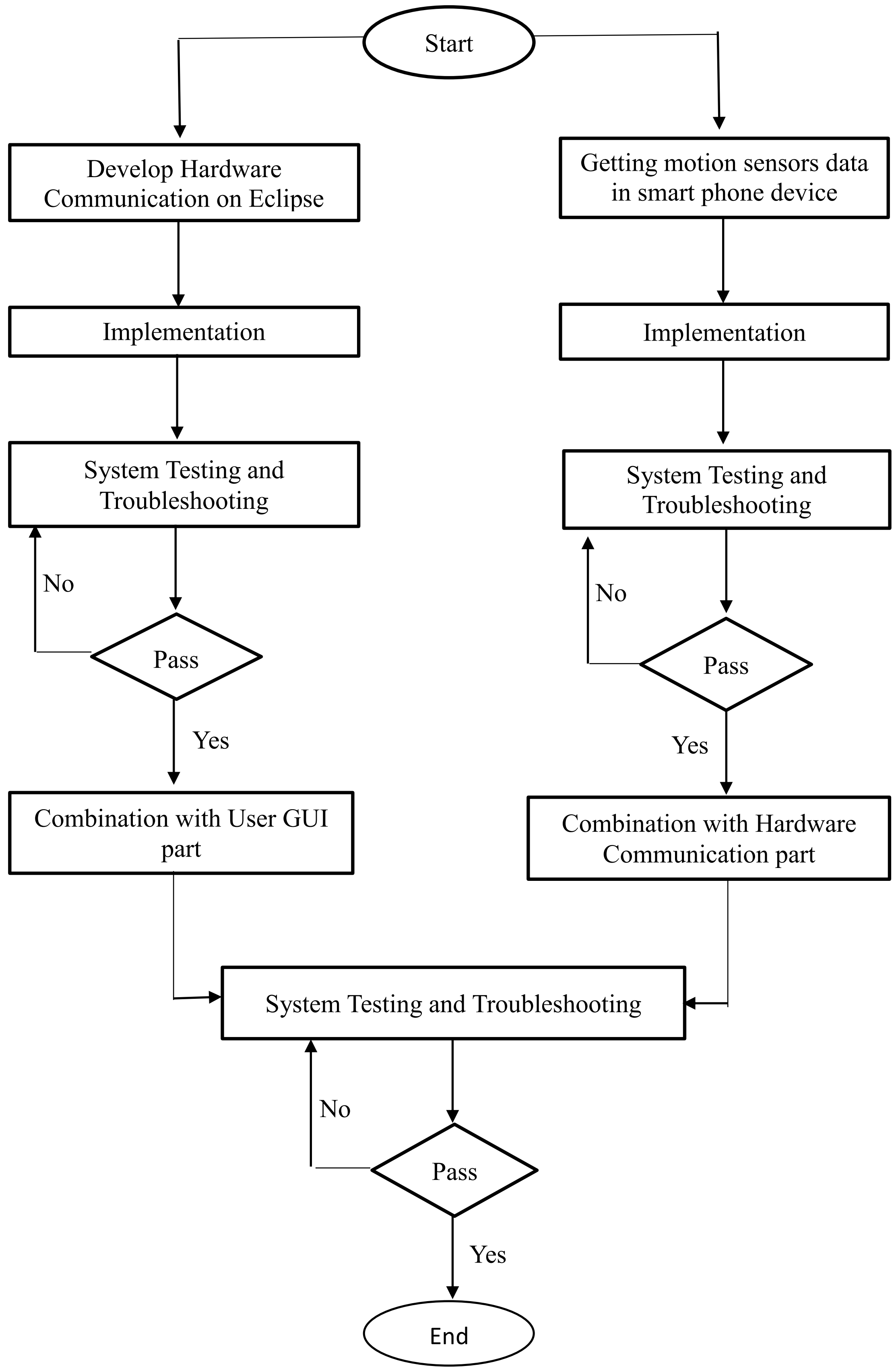

Start
Develop Hardware Communication on Eclipse
Getting motion sensors data in smart phone device
Implementation
Implementation
System Testing and Troubleshooting
System Testing and Troubleshooting
No
Pass
Yes
No
Pass
Yes
Combination with User GUI part
Combination with Hardware Communication part
System Testing and Troubleshooting
No
Pass
Yes
End

Figure 3.6: Overall Project flow

3.7 System Flow

This study is discussed about developing an Android application for the data mining and heart rate tracking system. So by using a heartbeats frequency rate monitor device to measure the heart rate of the users, then the device will send to value to the application by using Bluetooth communication. After that the heart rate of the user will display on the application GUI on our smartphone. Besides that, the motion sensor data also will be collecting and displayed it on the GUI. All of these data are able to be recorded into a text file and then save it into memory card in our smartphone device. The system flow of the application represented in the block diagram form can be seen in Figure 3.7.

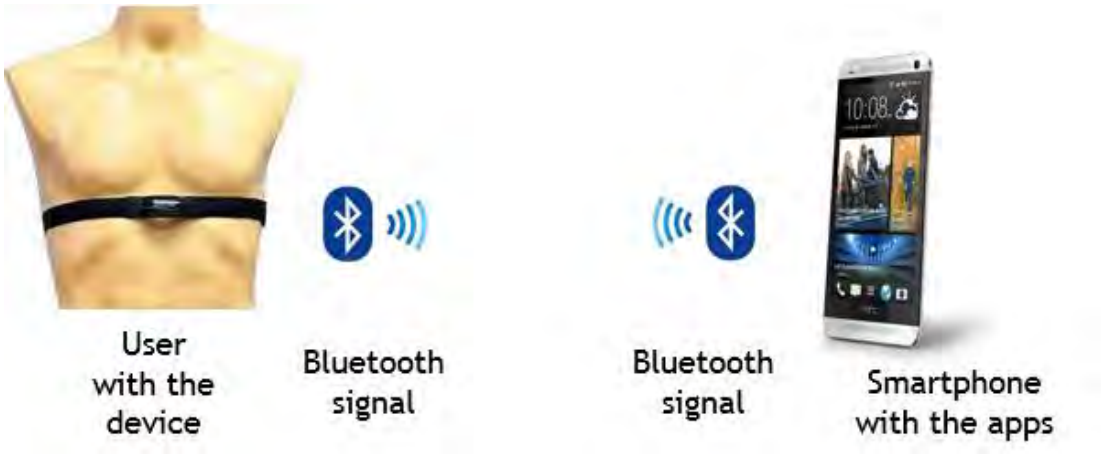


Figure 3.7: Block diagram for the system flow.

From the block diagram above shows that the heart rate signal is measured by the heart rate monitoring device which is a chest belt and user need to wear it on the chest position. After the device reads the signal then it will send to the apps on a smartphone with Bluetooth communication.

CHAPTER IV

RESULT AND DISCUSSION

4.1 Introduction

This chapter is discussing about the explanation and analysis of the result of the project. In term to get the results, the smartphone device must installed the application. This is because some of the features are not able to run in the software emulator, so we must install it on the device and run it so that we only able to manage the outcome. This chapter will divide into two parts, the first part is discussed about the hardware using for this project and then the second part discusses about the android application for this project. This application is required Bluetooth signal processing to communicate between the heart rate monitor device and our smartphone device. Besides that, the application also able to get the accelerometer, gyroscope, gravity sensor reading and GPS location. So the smartphone also requires these sensors so that the application can get all the results.

4.2 Project Device and Hardware Analysis

The Zephyr HxM Smart Heart Rate Monitor device from Zephyr Company are able to track the user's heart rate range between 25 to 240 beats per minute (BPM). The heart rate signal was generated by the hardware. It is operated in the Bluetooth range between 2.4 to 2.4835 GHz frequencies. Besides that, the transmit range for the heart rate monitor device depends on the Bluetooth transmit range, which is 10 meters. This means if the heart rate monitor device is 10 meters far away from your devices, the heart rate signal will not be success to send to your smartphone, and then the heart rate is unable to detect.

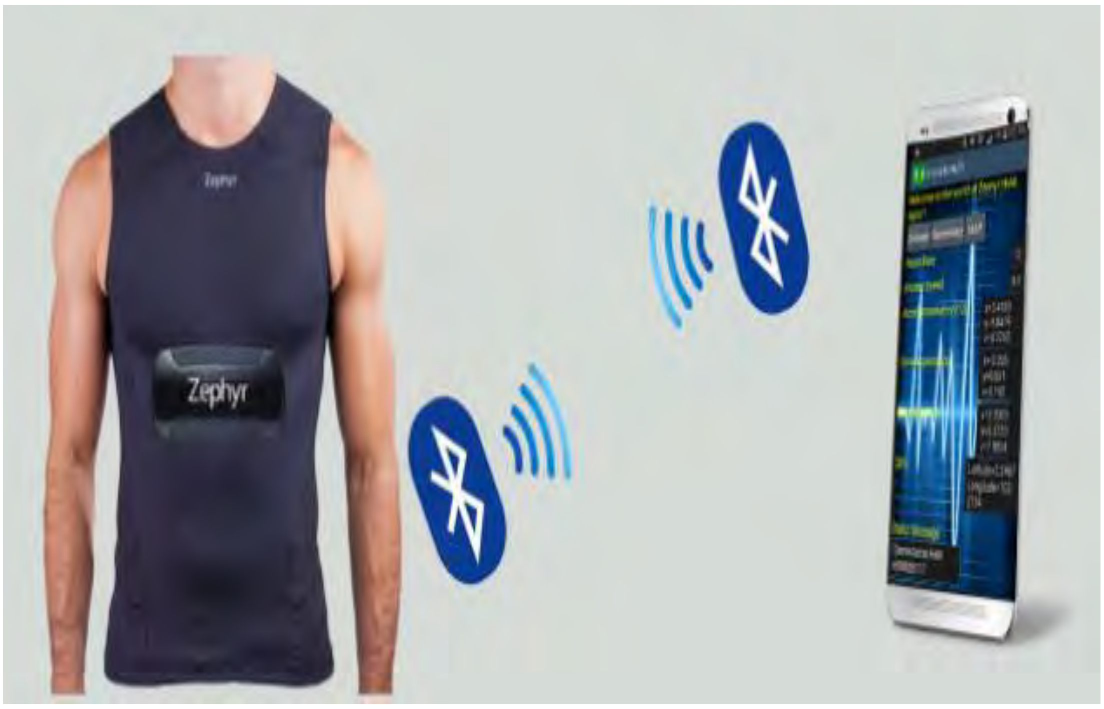


Figure 4.1: Hardware using Bluetooth communication with smartphone device.

In addition, the battery life of the hardware is 26 hours. It only took 1 hour to charge 90% of the battery life and 3 hours for 100% battery life charged. The charge cycles for the hardware it only 300 cycles. So to extend the battery life of the hardware, we can try to just charging to the 90% of the battery life of the hardware. There is no any button or switch to turn the heart rate device on or off. We only need to wear it on our chest position and take a deep breath, then the device will automatically turn on. There are two LEDs on the heart rate device to let user know the device is charging or fully charged. When the device was charging, the two LEDs will light up with one yellow and one red. When it is fully charged, the red led will turn to yellow color.

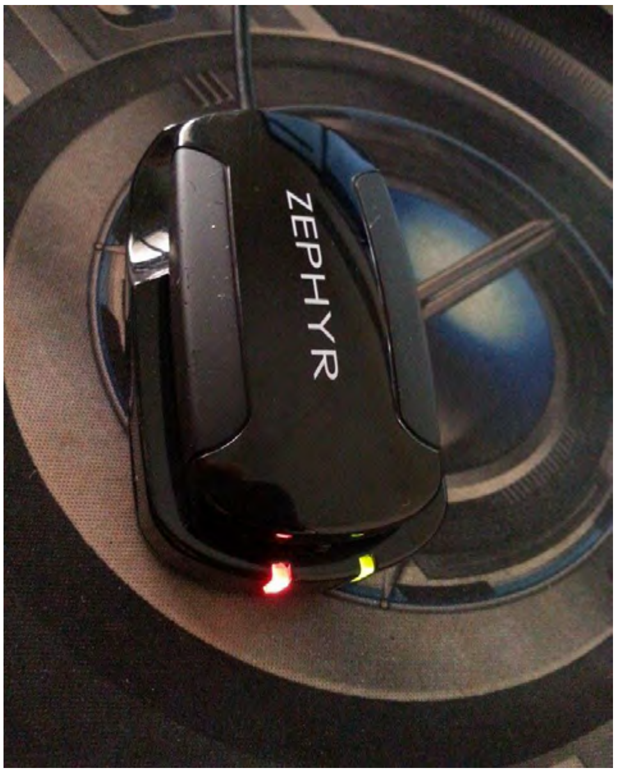


Figure4.2: Zephyr HxM BT hardware was charging

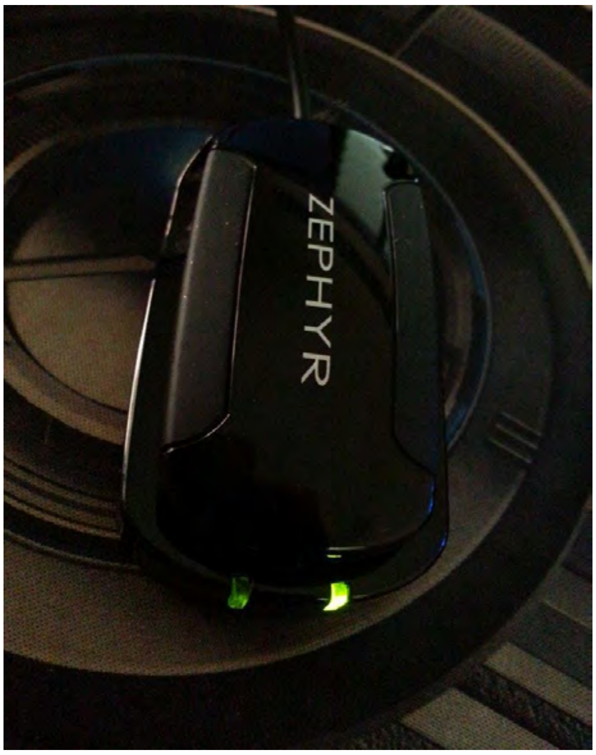


Figure 4. 3: Zephyr HxM BT hardware was fully charged

There is a very important part when using this hardware, the Zephyr HxM BT hardware is turned on automatically when the user wears it and take a deep breath. This is because the hardware needs enough conductivity between user's skin and the device itself. So if a user wears the device outside the shirt, even user takes a deep breath, but the device may not be able to get the ECG signal and send it to our app. Sometimes, it may need to moisten the sensor pads on the chest strap with little water to allow the HxM turn on.

Moreover, the Bluetooth connection may also cause the Zephyr HxM Bt hardware fail to get the data. So when face this problem, user can try to unpair it between the hardware with our smartphone and then pair it back. If the problem still there, then users can try to turn off the Bluetooth connection of the smarphone and then turn it on again to check there is a connection between it or not.

If the problem still occur, then user may check there is other Bluetooth devices active within close range or not. Sometimes there are a number of Bluetooth devices working close range, then they will interfere each other with the signal from our HxM to smartphone. So try to move away and try connecting it again.

Lastly, we also can check the functionality of the Bluetooth connection between the HxM with our smartphone by using another application. If there is no error for the Bluetooth connection, then they should be the application problem to cause the application malfunction.

4.3 Zephyr HxM BT heart rate monitor device connection with R-Tracker4Life

R-Tracker4Life is designed for the heart rate monitoring purpose. This android app is designed based on Zephyr HxM BT heart rate monitor device, so if the user wants to use the heart rate tracking features in the application, the user must have the HxM hardware and then connecting with their smartphone. For connecting the HxM hardware to smartphone, firstly user need to turn on Bluetooth on their smartphone. After that pairing the hardware with smartphone and then run the application. By following these procedure user are able to connect the HxM hardware with the application.

Procedure:

1. Wearing Zephyr HxM BT heart rate monitor device on the chest position of your body.
2. Take a deep breath to turn on the hardware device.
3. Turn on Bluetooth on your smartphone.
4. Select Zephyr HxM BT device to connect it.
5. It will require a pin number to access it, normally is “1234”. But developer also can change in the coding part.
6. After key in the pin number, now you were pairing the HxM hardware with your smartphone already.
7. Open your application and then start to track your heart rate.

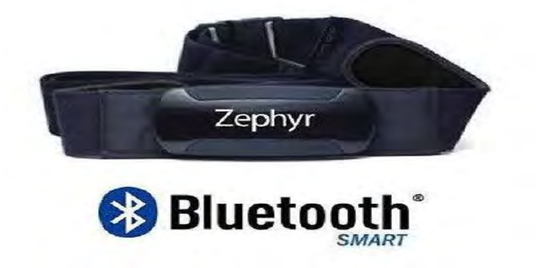


Figure4.4: Zephyr HxM BT hardware device.

I.

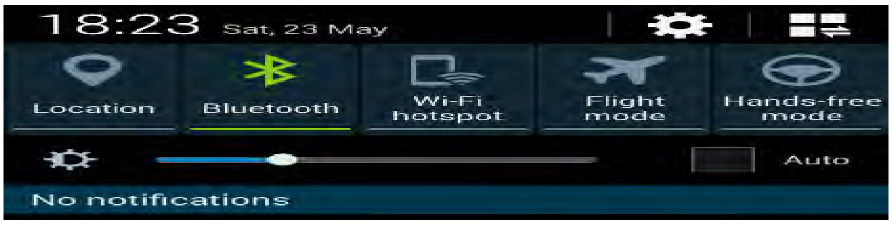
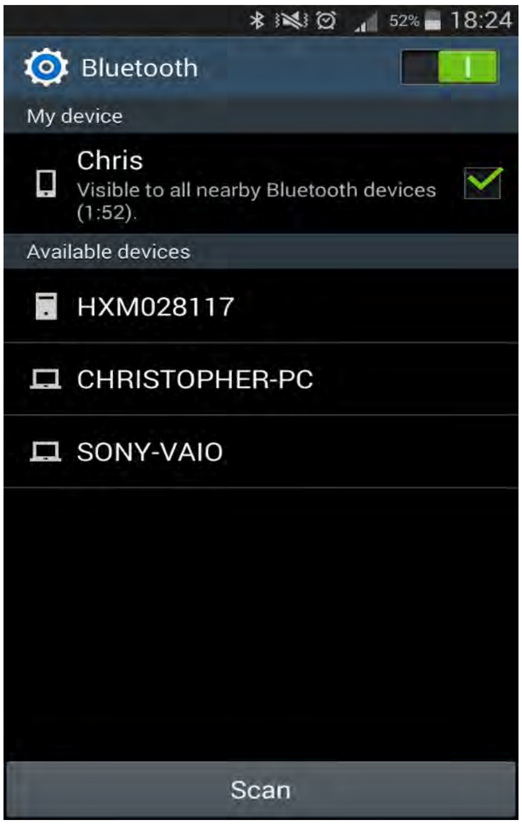


Figure 4.5: Turn on Bluetooth.

II.



My device

Chris
Visible to all nearby Bluetooth devices (1:52).

Available devices

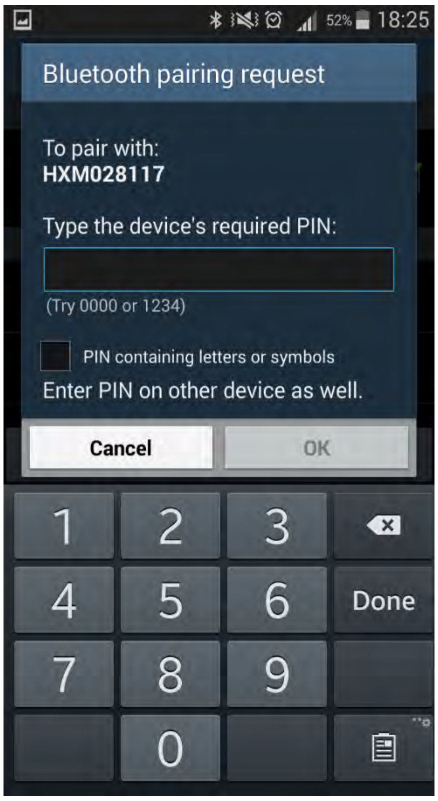
HXM028117

CHRISTOPHER-PC

SONY-VAIO

Figure 4.6: Select Zephyr HxM heart rate device.

III.


18:25

Bluetooth pairing request

To pair with:
HXM028117

Type the device's required PIN:

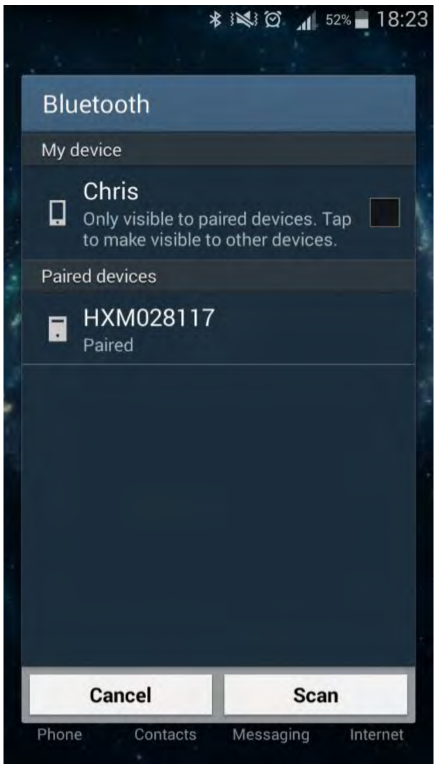
(Try 0000 or 1234)

PIN containing letters or symbols

Enter PIN on other device as well.

Figure 4.7: Key in Pin number to access the connection.

IV.


18:23

Bluetooth

My device

Chris
Only visible to paired devices. Tap to make visible to other devices.

Paired devices

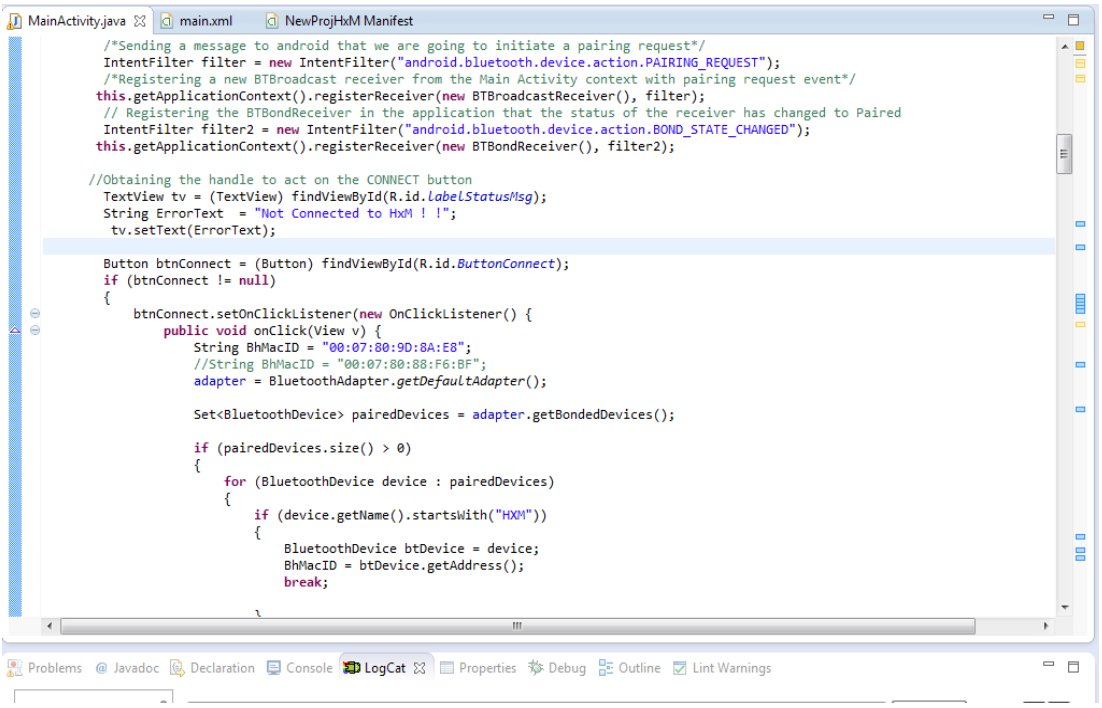
HXM028117
Paired

Figure 4.8: HxM successful to paired and ready to use.

4.4 Application (R-Tracker4Life) Analysis

4.4.1 App features analysis

The first part we discuss about how the hardware connection with the application (software). Now we continue to discuss about the android application which is R-Tracker4Life. Firstly, this android application is developing in the integrated development environment which is Eclipse software. Eclipse is a software which provides a development platform to developer who want to develop an Android application. The most important thing is it is an open source integrated development environment (IDE) to let developers develop their project. Since R-Tracker4Life is developed based on Zephyr HxM BT heart rate device, so the first step to develop this application is to build the link bridge between the hardware and the application. Besides that, we also need to design a layout to display the heart rate result, we get it from the HxM device and then to display for the user know either the application has connected with the heart rate device or not.



```
    /*Sending a message to android that we are going to initiate a pairing request*/
    IntentFilter filter = new IntentFilter("android.bluetooth.device.action.PAIRING_REQUEST");
    /*Registering a new BTBroadcast receiver from the Main Activity context with pairing request event*/
   this.getApplicationContext().registerReceiver(new BTBroadcastReceiver(), filter);
    // Registering the BTBondReceiver in the application that the status of the receiver has changed to Paired
    IntentFilter filter2 = new IntentFilter("android.bluetooth.device.action.BOND_STATE_CHANGED");
   this.getApplicationContext().registerReceiver(new BTBondReceiver(), filter2);

  //Obtaining the handle to act on the CONNECT button
    TextView tv = (TextView) findViewById(R.id.labelStatusMsg);
    String ErrorText  = "Not Connected to HxM ! !";
     tv.setText(ErrorText);

    Button btnConnect = (Button) findViewById(R.id.ButtonConnect);
    if (btnConnect != null)
    {
        btnConnect.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                String BhMacID = "00:07:80:9D:8A:E8";
                //String BhMacID = "00:07:80:88:F6:BF";
                adapter = BluetoothAdapter.getDefaultAdapter();

                Set<BluetoothDevice> pairedDevices = adapter.getBondedDevices();

                if (pairedDevices.size() > 0)
                {
                    for (BluetoothDevice device : pairedDevices)
                    {
                        if (device.getName().startsWith("HXM"))
                        {
                            BluetoothDevice btDevice = device;
                            BhMacID = btDevice.getAddress();
                            break;
```

Figure 4.9: Coding for communication between the hardware and the application.

Figure 4.9 above shows that the coding used to communicate between the hardware and the app. First, we need to import Bluetooth Adapter into the project so that we only can use the Bluetooth function in the application. After that we can write the operation that we want to do when we get the Bluetooth signal.

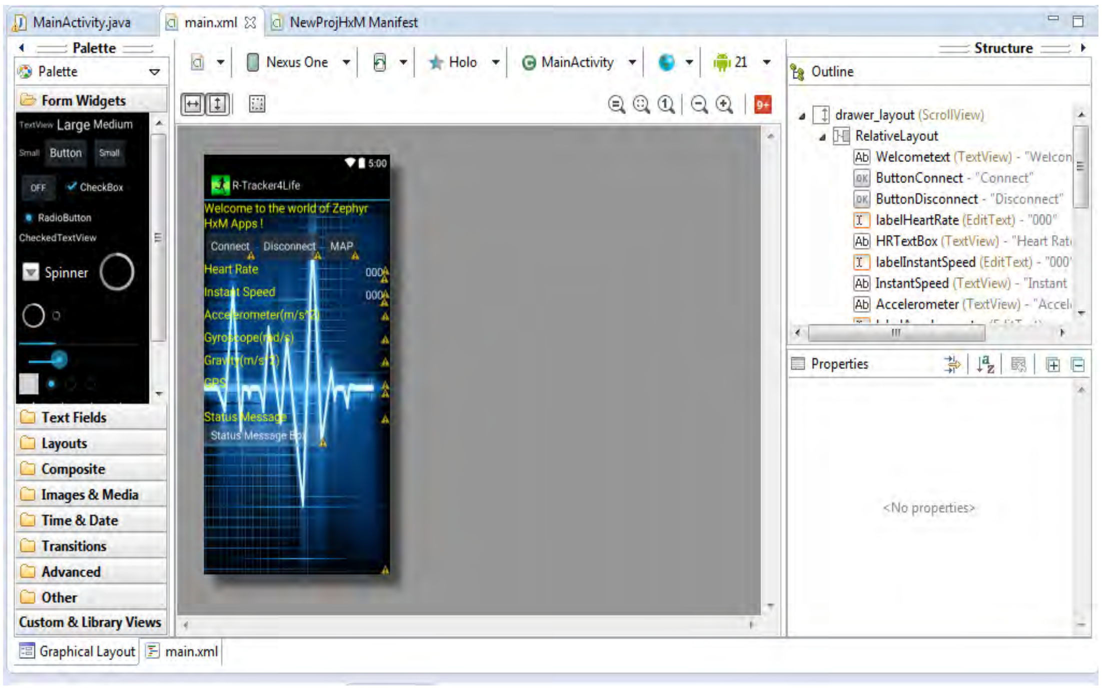


Figure 4.10: GUI layout design display on eclipse software.

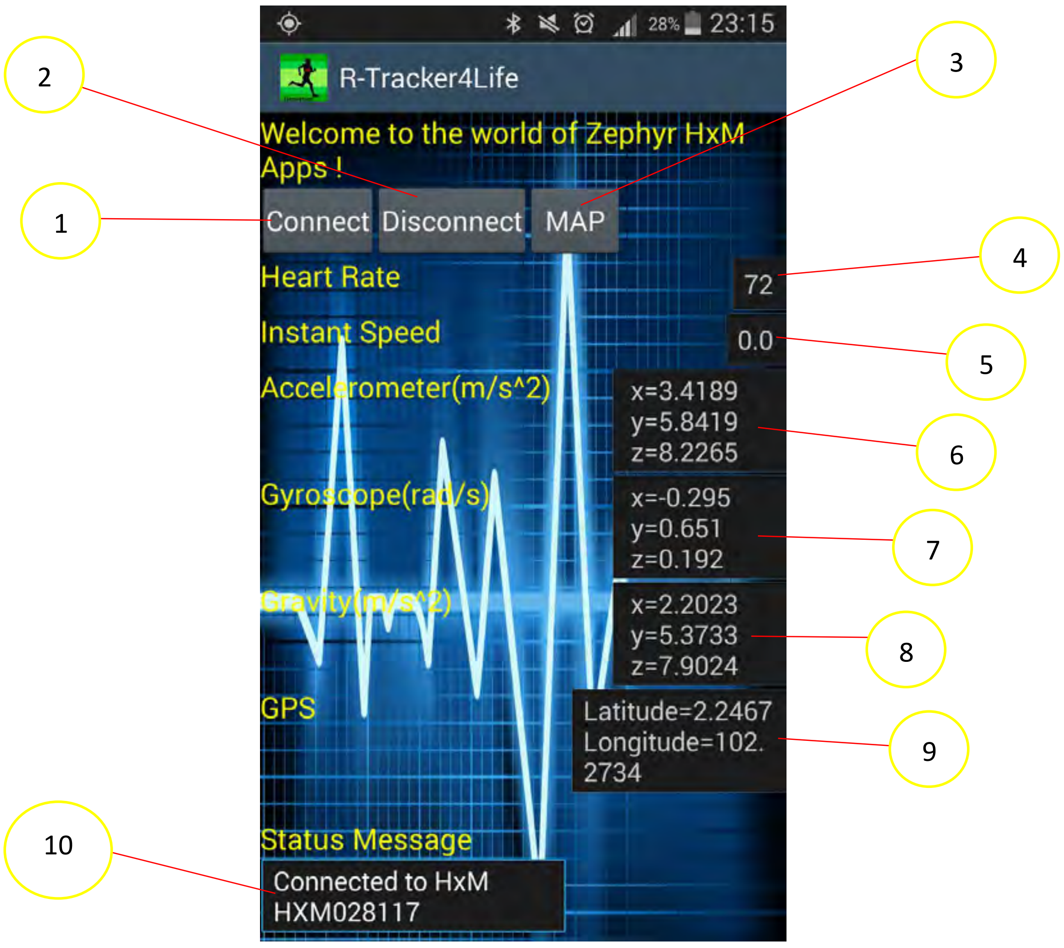


Figure 4.11: Actual layout on device.

The figure 4.11 above is the application GUI layout design of the eclipse software and then for figure 4.12 are the actual GUI layout design on real smartphone device. There have total ten components in figure 4.11. The list below is the description for each of the components.

1. "Connect" button to let user connect the HxM hardware with the app.
2. "Disconnect" button for the user to disconnect it the connection.
3. "Map" button to like the Mainactivity page to the map layout page.
4. Heart beat per minute (BPM) reading display in textview.
5. The instant speed for the HxM device.
6. Accelerometer sensor reading display in textview.
7. Gyroscope sensor reading display in textview.
8. Gravity sensor reading display in textview.
9. GPS Longitude and Latitude reading display in textview.

10. The status message about the connection for the HxM device and the application. If they are connected, the status will show “Connected to HxM123456” message. If not it will display “Not Connected to HxM!”

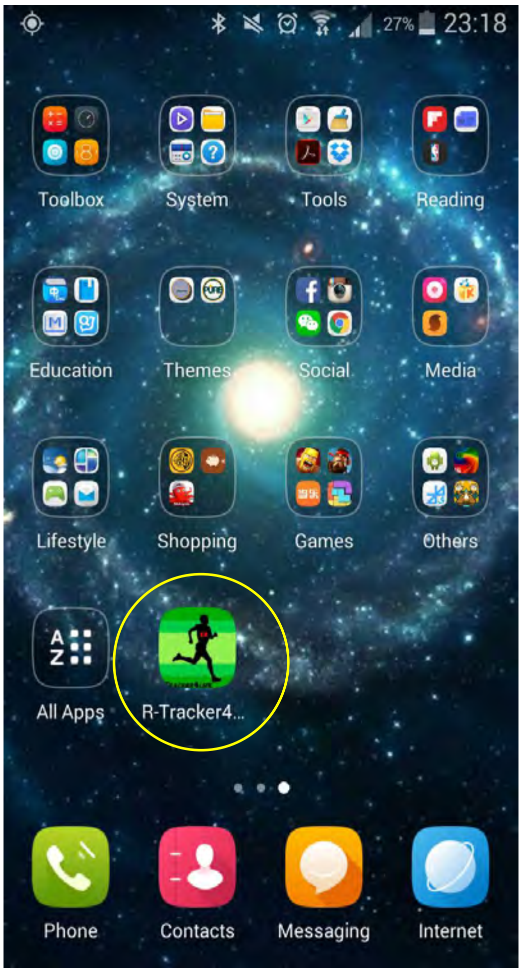
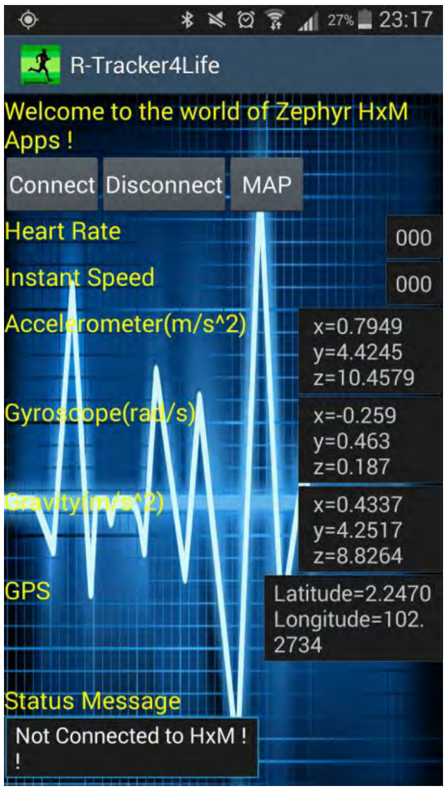


Figure 4.12: “R-Tracker4Life” apps layout has been installed on the smartphone device.



Welcome to the world of Zephyr HxM Apps !

Heart Rate 000

Instant Speed 000

Accelerometer(m/s^2) x=0.7949 y=4.4245 z=10.4579

Gyroscope(rad/s) x=-0.259 y=0.463 z=0.187

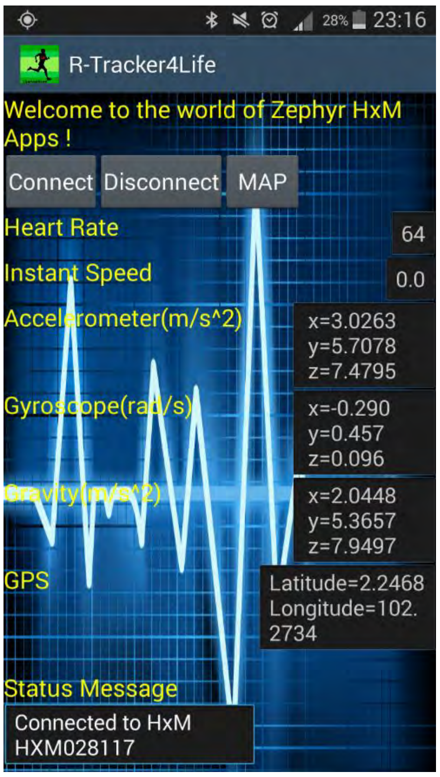
Gravity(m/s^2) x=0.4337 y=4.2517 z=8.8264

GPS Latitude=2.2470 Longitude=102.2734

Status Message

Not Connected to HxM ! !

Figure 4.13: GUI layout before connection.



Welcome to the world of Zephyr HxM Apps !

Heart Rate 64

Instant Speed 0.0

Accelerometer(m/s^2) x=3.0263 y=5.7078 z=7.4795

Gyroscope(rad/s) x=-0.290 y=0.457 z=0.096

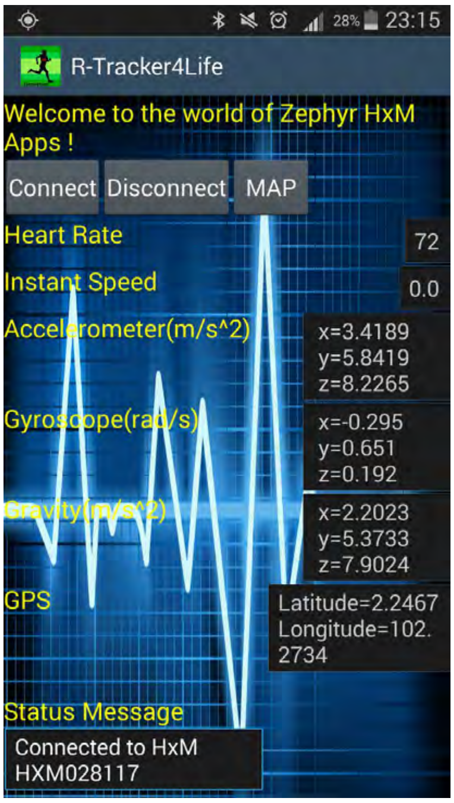
Gravity(m/s^2) x=2.0448 y=5.3657 z=7.9497

GPS Latitude=2.2468 Longitude=102.2734

Status Message

Connected to HxM HXM028117

Figure 4.14: GUI layout after connection.



72

0.0

x=3.4189
y=5.8419
z=8.2265

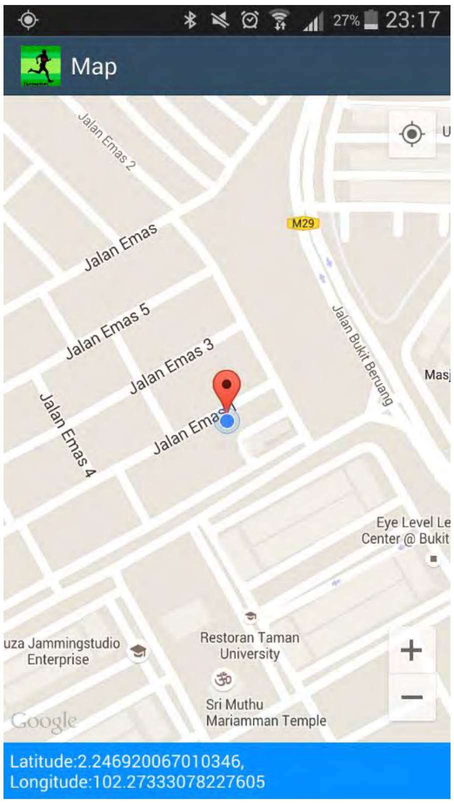
x=-0.295
y=0.651
z=0.192

x=2.2023
y=5.3733
z=7.9024

Latitude=2.2467
Longitude=102.
2734

Connected to HxM
HXM028117

Figure 4.15: Heart rate reading increase compare with figure above.



Latitude:2.246920067010346,
Longitude:102.27333078227605

Figure 4.16: Google map layout after user click "Map" button.

4.4.2 App features analysis with coding discussion

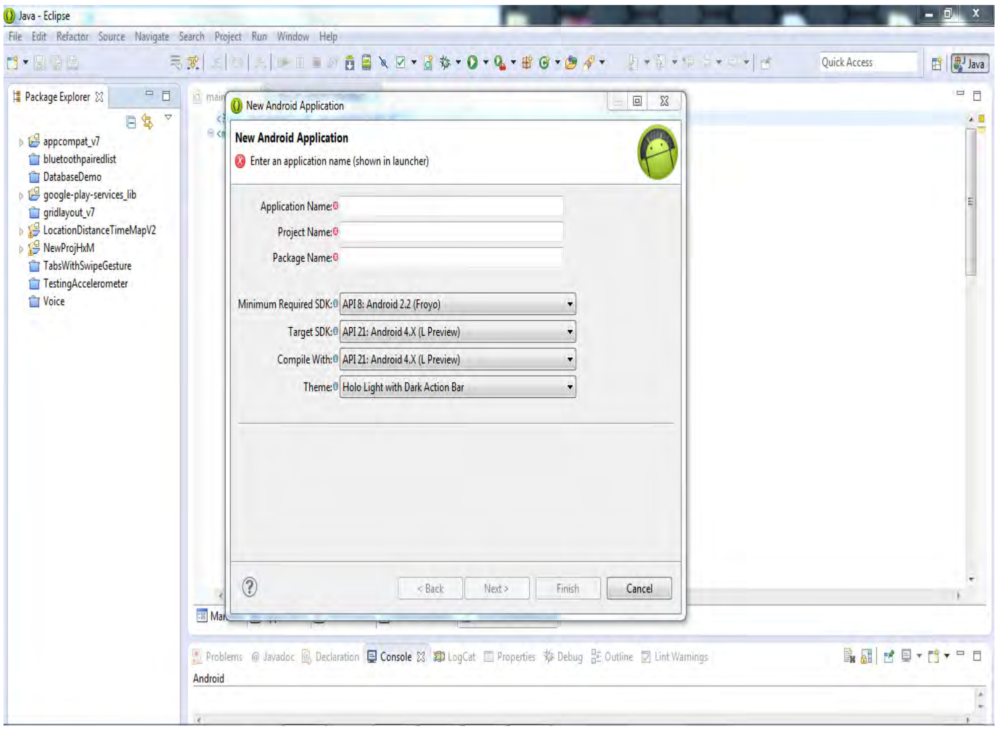
This section is use to discussion about the features for the application R-Tracker4Life. For this app there have three sensors using which is Accelerometer, Gyroscope and Gravity sensors and the last one is a GPS tracking feature.

4.4.2.1 Android smartphone sensors implemented into the application

The Android platform provides several sensors that to let us monitor our motion of the device. There are three types of sensors can be categorize which is Motion sensors, orientation sensors and environmental sensors. Theses sensors are capable of providing a raw data with high precision and accuracy. But for the aim of the application is to develop a workout training application, so we no need to implement all the sensors in our application. For R-Tracker4Life, we decide to implement three types of motion sensors which is Accelerometer, Gravity and Gyroscope sensors.

First, we need to add new project in our eclipse software. So we need to go File then choose new and select android application project and then a mini new android application window will pop out as figure 4.17. After that, user needs to key in the application name in the project window. For minimum required SDK, user can choose the minimum required SDK for your application and then choose also the target SDK. For target SDK, users should choose the latest version of the application so that it can use for almost all of the Android device.

.



New Android Application

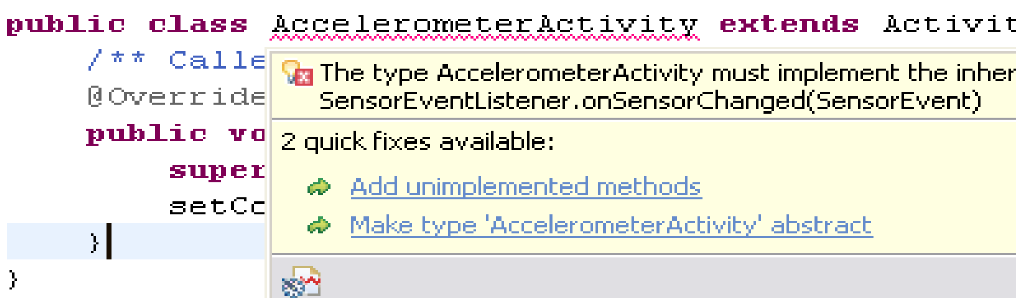
Enter an application name (shown in launcher)

Application Name:
Project Name:
Package Name:

Minimum Required SDK: API 8: Android 2.2 (Froyo)
Target SDK: API 21: Android 4.X (L Preview)
Compile With: API 21: Android 4.X (L Preview)
Theme: Holo Light with Dark Action Bar

Figure 4.17: Create a new android application project.

After creating application, we need to add sensor eventlistener in our application so that our application only able to use the sensor on the devices. The user can click to the .java class and after the “extends Activity”, put “implements SensorEventListener” (The error will occur, then point your cursor to the error and click “Add unimplemented methods”).



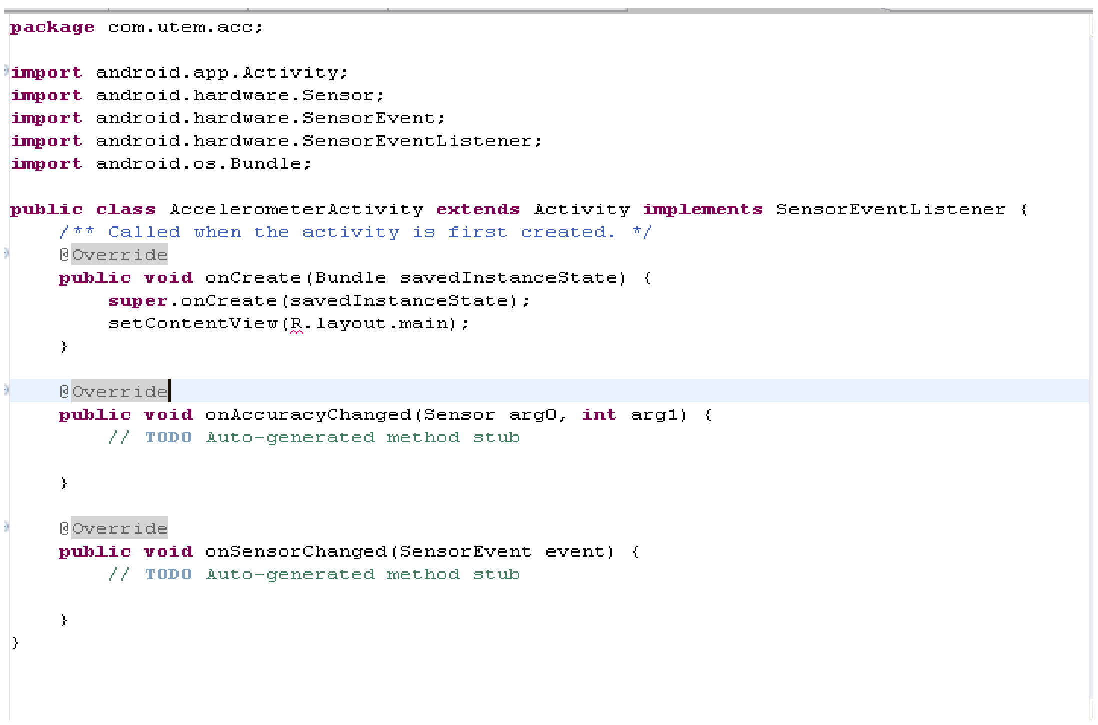
```
public class AccelerometerActivity extends Activit
        /** Calle
        @Override
        public vo
                super
                setCc
        }
}
```

The type AccelerometerActivity must implement the inher
SensorEventListener.onSensorChanged(SensorEvent)

2 quick fixes available:

Add unimplemented methods
Make type 'AccelerometerActivity' abstract

Figure 4.18: Add unimplemented methods to our .java coding.



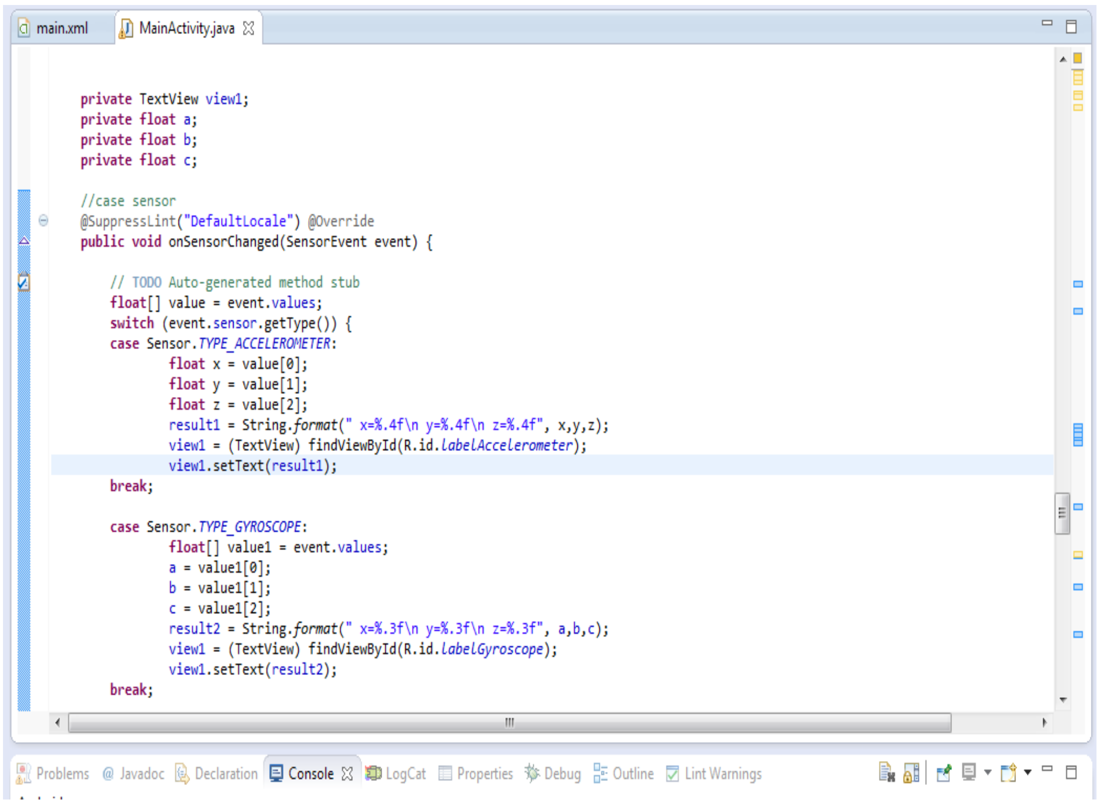
```
package com.utem.acc;

import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.os.Bundle;

public class AccelerometerActivity extends Activity implements SensorEventListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    public void onAccuracyChanged(Sensor arg0, int arg1) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        // TODO Auto-generated method stub

    }
}
```

Figure 4.19: Example of sensor coding after adding in the coding.

From the figure 4.19, we can observe that there will have two autogenerated method stub generate by the eclipse software. For onAccuracyChanged method stub is do about the operation for the sensor accuracy change. For example, we can set the sensor delay for the sensor to several environmental conditions in this method. For example, I set the sensor to user interface environment for this project.

```
sensorManager.registerListener(SensorEventListener,
 sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        SensorManager.SENSOR_DELAY_UI);
```

This is because I want to display the accelerometer sensor in the user interface. So I set the sensor_delay with UI. If not, the sensitivity of the sensor will increase and then we are very hard to read the actual reading from the sensor. Then for the onSensorChanged method stub is the method for us to write the operation comment when the sensor reading was change for the sensorevent. Figure 4.20 shows are the example coding for my application on the onSensorChanged method stub.



```
private TextView view1;
private float a;
private float b;
private float c;

//case sensor
@SuppressLint("DefaultLocale") @Override
public void onSensorChanged(SensorEvent event) {

    // TODO Auto-generated method stub
    float[] value = event.values;
    switch (event.sensor.getType()) {
    case Sensor.TYPE_ACCELEROMETER:
            float x = value[0];
            float y = value[1];
            float z = value[2];
            result1 = String.format(" x=%.4f\n y=%.4f\n z=%.4f", x,y,z);
            view1 = (TextView) findViewById(R.id.labelAccelerometer);
            view1.setText(result1);
    break;

    case Sensor.TYPE_GYROSCOPE:
            float[] value1 = event.values;
            a = value1[0];
            b = value1[1];
            c = value1[2];
            result2 = String.format(" x=%.3f\n y=%.3f\n z=%.3f", a,b,c);
            view1 = (TextView) findViewById(R.id.labelGyroscope);
            view1.setText(result2);
    break;
```

Figure 4.20: Example coding for onSensorChanged method stub.

After done the mainActivity.jave part, now we need go to edit the layout of the application. So go to the main.xml and then add textView's id and modify the string for the layout.

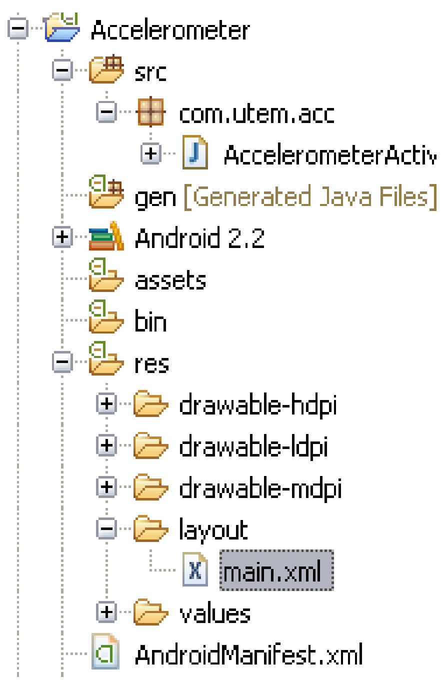
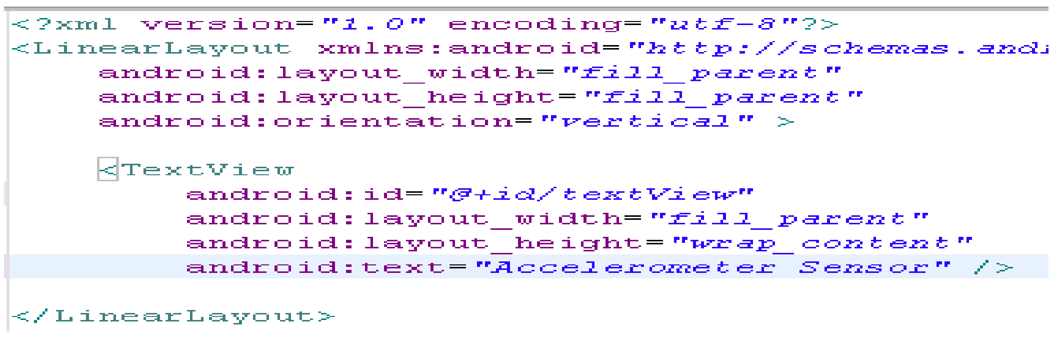


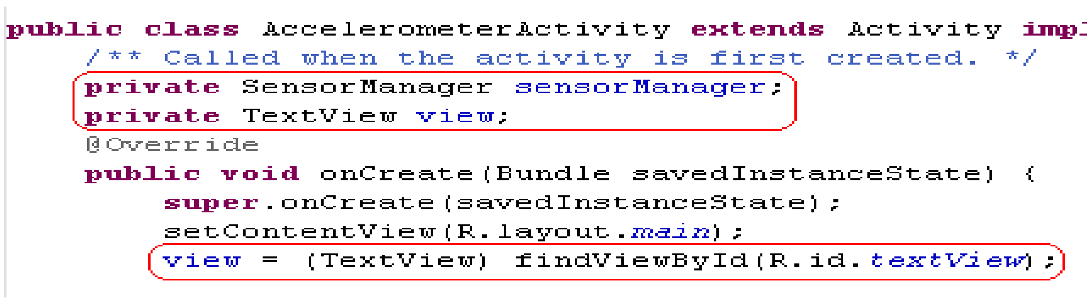
Figure 4.21: Open main.xml under layout folder.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.andi
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Accelerometer Sensor" />

</LinearLayout>
```

Figure 4.22: main.xml coding.

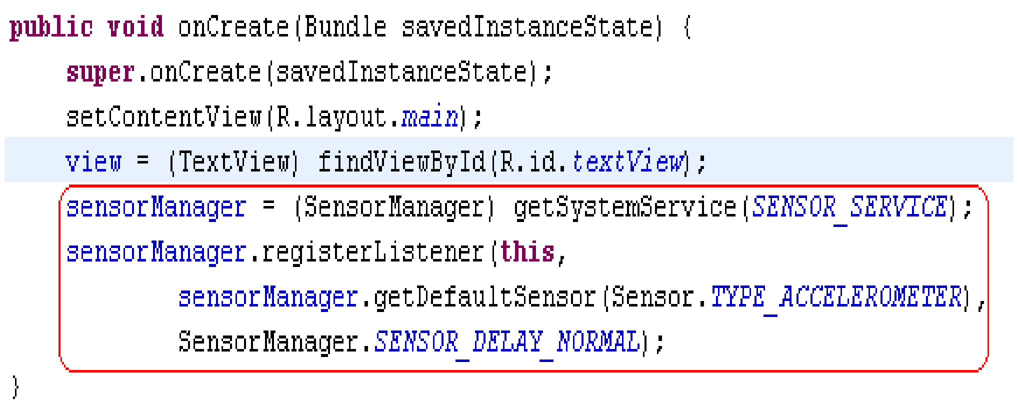
After doing the layout part, now we go back to mainActivity.java to declare SensorManager and TextView in our coding so that the coding only can run smoothly without error.



```
public class AccelerometerActivity extends Activity imp
    /** Called when the activity is first created. */
    private SensorManager sensorManager;
    private TextView view;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        view = (TextView) findViewById(R.id.textView);
```

Figure 4.23: Declare sensormanager and textview.

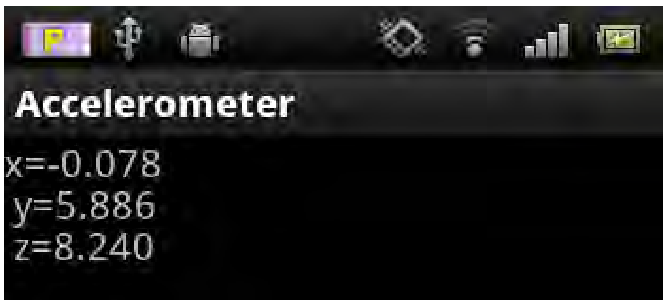
In onCreate method stub, we need to put the initialvaluefor the sensorManager.



```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    view = (TextView) findViewById(R.id.textView);
    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    sensorManager.registerListener(this,
            sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
            SensorManager.SENSOR_DELAY_NORMAL);
}
```

Figure 4.24: onCreate method stub coding.

The result will show the sensor value of x, y and z axis in our .xml layout.



Accelerometer

```
x=-0.078
y=5.886
z=8.240
```

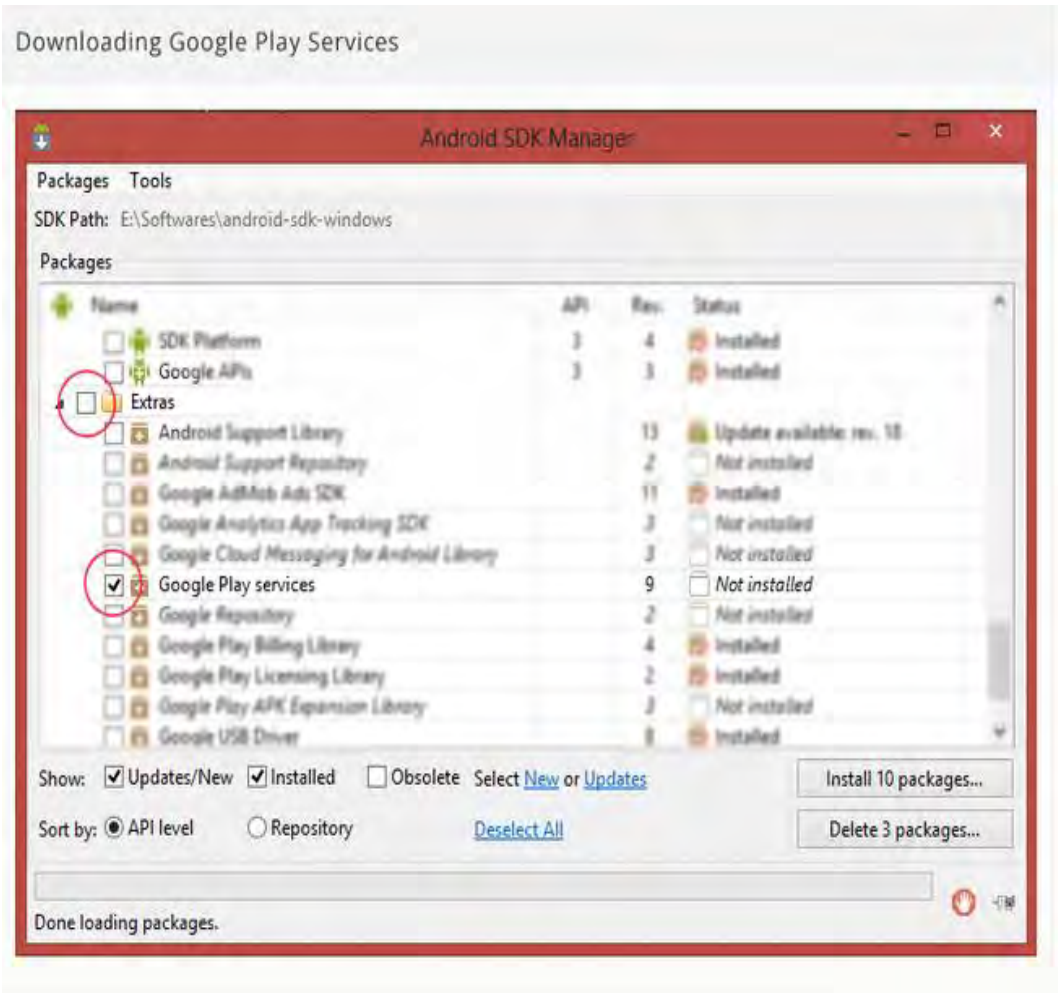
Figure 4.25: Accelerometer results on application layout.

4.4.2.2 Google map implemented into the application

This part will discuss about how this android application had implemented the Google map function. Google via google play service provides us a library for utilizing the Google map function in our application. The following description and procedure is the step how to use google map in our application by using Google Map Android API V2 which provides substantial improvements to older API version. The older version is google map api v1 but that was outdated already by google. The minimum version to use google map in our application is version 2 and the latest edition is version 3. Before starting a fresh project, we have to undergo a few pre needed actions to ascertain that the google map display on our application. These cases of activities include importing the required library, making the actual SHA1 fingerprint and configuring maps in google console.

1. Downloading Google Play Services on android eclipse software

Google made new Maps V2 API as a constituent of a Google Play Services SDK. Hence prior to all of us begin to prepare maps, we bear to download google play services from SDK manger. You are capable to spread out the SDK manager either through Eclipse software or through android SDK folder. The steps required are Open Eclipse ⇒ Windows ⇒ Android SDK Manager and check whether you have already downloaded Google Play Services or not under Extras section. Otherwise, select play services and then install the software.



Downloading Google Play Services

Android SDK Manager

Packages Tools

SDK Path: E:\Softwares\android-sdk-windows

Packages

Name	API	Rev.	Status
☐ SDK Platform	3	4	Installed
☐ Google APIs	3	3	Installed
☐ Extras			
☐ Android Support Library		13	Update available: rev. 18
☐ Android Support Repository		2	Not installed
☐ Google AdMob Ads SDK		11	Installed
☐ Google Analytics App Tracking SDK		3	Not installed
☐ Google Cloud Messaging for Android Library		3	Not installed
✓ Google Play services		9	Not installed
☐ Google Repository		2	Not installed
☐ Google Play Billing Library		4	Installed
☐ Google Play Licensing Library		2	Installed
☐ Google Play APK Expansion Library		3	Not installed
☐ Google USB Driver		8	Installed

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates

Install 10 packages...

Sort by: ● API level ○ Repository Deselect All

Delete 3 packages...

Done loading packages.

Figure 4.26: Google play services had downloading and installed on android sdk.

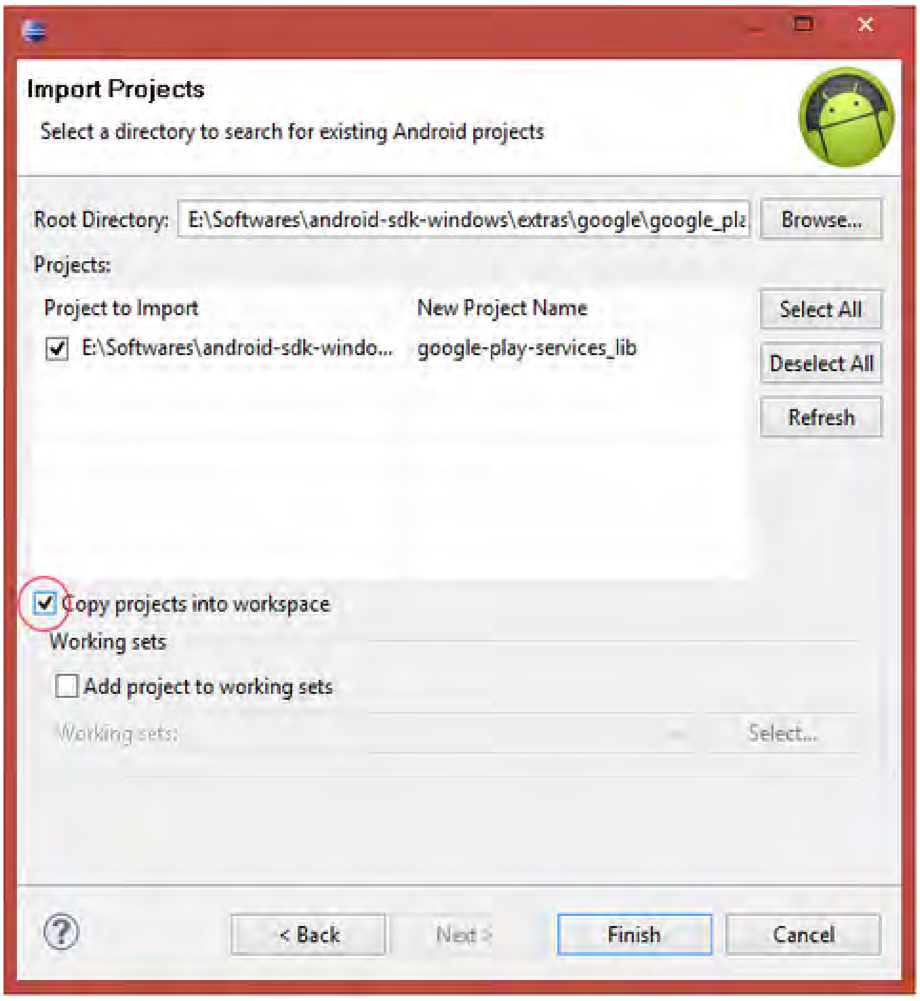
2. Importing Google Play Services into Eclipse Software

After downloading the google play services, we need to import it into Eclipse, which will be utilized as a library for our maps project so that we only able to beat the google map service in our application.

i. In Eclipse go to File ⇒ Import ⇒ Android ⇒ Existing Android Code into Workspace.

ii. Click on Search and choose Google Play Services project from android SDK folder. You are able to locate google play services library project from"android-sdk-windows\extras\google\google_play_services\ libproject \google-play-services_lib".

iii. There is very important to tick up the Copy projects into workspace option in the window as shown in the figure 4.27.


Import Projects

Select a directory to search for existing Android projects

Root Directory: E:\Softwares\android-sdk-windows\extras\google\google_pl

Projects:

Project to Import	New Project Name
✓ E:\Softwares\android-sdk-windo...	google-play-services_lib

[✓] Copy projects into workspace

Working sets

[ ] Add project to working sets

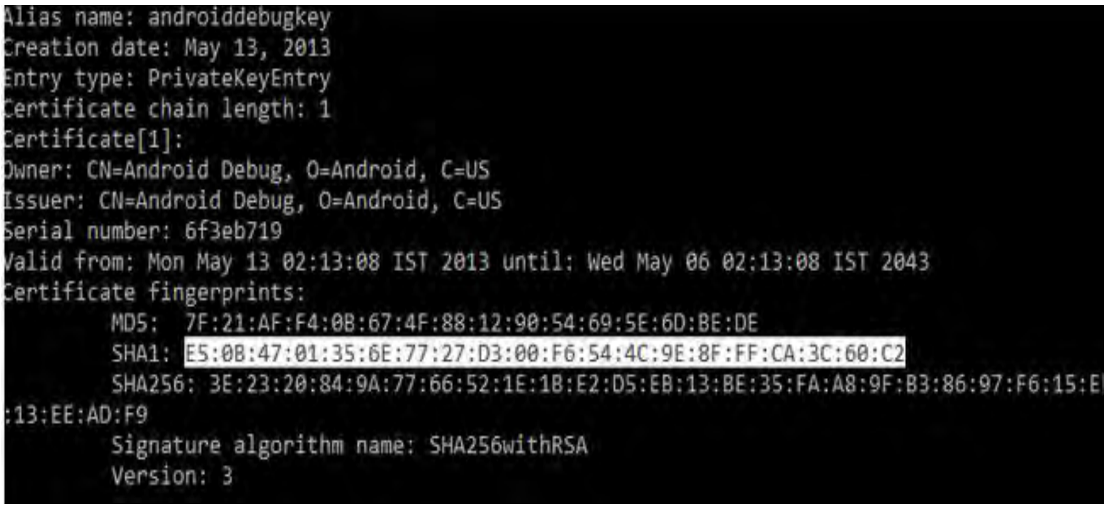
Figure 4.27: Importing Google Play Service into Eclipse IDE.

3. Getting the Google Maps API Key

API Key is a key habit to allow our application can access the google play service and then receive the google map display permission. Before getting the api key, we need first to generate the SHA-1 fingerprint using java keytool. Foremost, we need to open our terminal and then perform the following instruction to generate our SHA-1 fingerprint.

"keytool -list -v -keystore "%USERPROFILE%\. android\ debug.keystore" -alias androiddebugkey -storepass android -keypass android".

Figure 4.28 shows the output that we can get our SHA-1 finger print on the console.

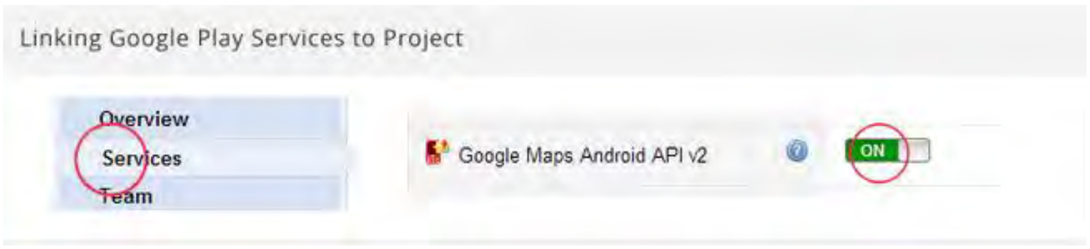


```
Alias name: androiddebugkey
Creation date: May 13, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 6f3eb719
Valid from: Mon May 13 02:13:08 IST 2013 until: Wed May 06 02:13:08 IST 2043
Certificate fingerprints:
         MD5:  7F:21:AF:F4:0B:67:4F:88:12:90:54:69:5E:6D:BE:DE
         SHA1: E5:0B:47:01:35:6E:77:27:D3:00:F6:54:4C:9E:8F:FF:CA:3C:60:C2
         SHA256: 3E:23:20:84:9A:77:66:52:1E:1B:E2:D5:EB:13:BE:35:FA:A8:9F:B3:86:97:F6:15:E
:13:EE:AD:F9
         Signature algorithm name: SHA256withRSA
         Version: 3
```

Figure 4.28: Generating SHA-1 fingerprint using comment prompt in window.

4. Now go to the google api console to activate our google api key and get the permission with our application.

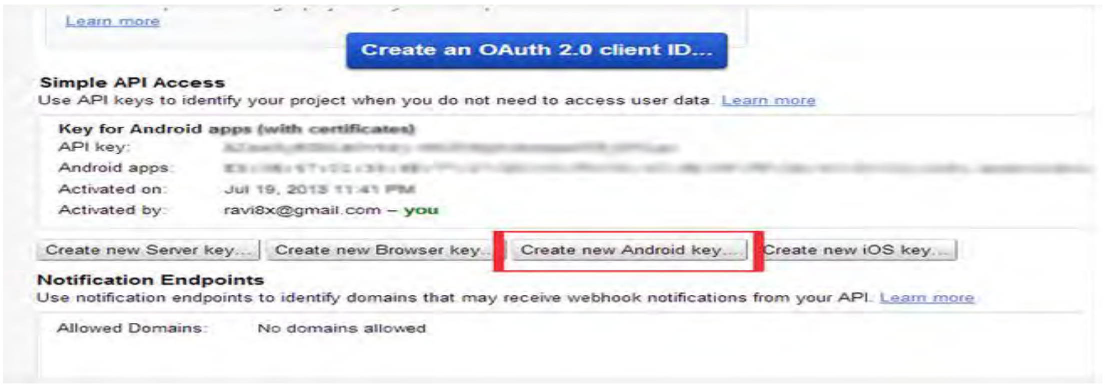
5. Select Services on the left hand side window and then turn on the google maps android api v2 as figure 4.29.



Linking Google Play Services to Project

Figure 4.29: Turn on the google map android api v2 on google api console.

6. Select the API Access on the left hand side and click on create a new android key at the right side.



Learn more

Create an OAuth 2.0 client ID...

Simple API Access
Use API keys to identify your project when you do not need to access user data. Learn more

Key for Android apps (with certificates)

API key:	[redacted]
Android apps:	[redacted]
Activated on:	Jul 19, 2013 11:41 PM
Activated by:	ravi8x@gmail.com – you

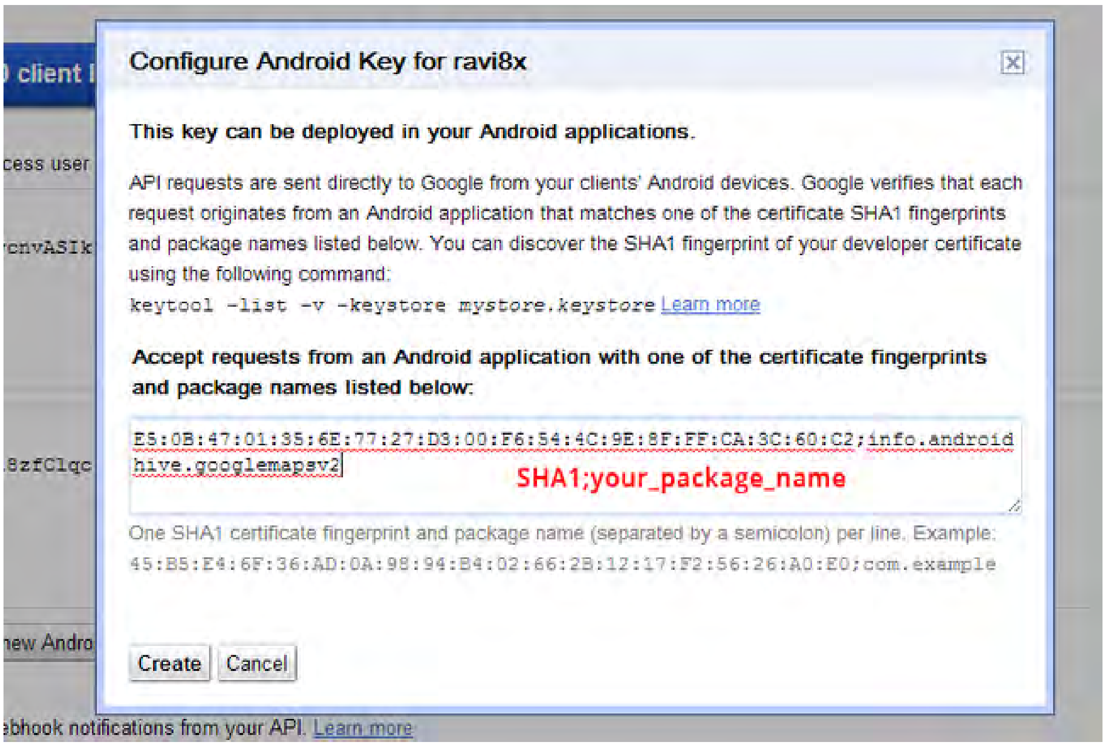
Create new Server key... | Create new Browser key... | Create new Android key... | Create new iOS key...

Notification Endpoints
Use notification endpoints to identify domains that may receive webhook notifications from your API. Learn more

Allowed Domains: No domains allowed

Figure 4.30: Obtaining google android map api key on google console.

7. It will pop up a window for asking the SHA-1 that we had generated early and the package name of our application. So enter our SHA-1 key with our application project package name, separated by a semicolon; and then click on create.



Configure Android Key for ravi8x

This key can be deployed in your Android applications.

API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches one of the certificate SHA1 fingerprints and package names listed below. You can discover the SHA1 fingerprint of your developer certificate using the following command:

```
keytool -list -v -keystore mystore.keystore
```

Learn more

Accept requests from an Android application with one of the certificate fingerprints and package names listed below:

```
E5:0B:47:01:35:6E:77:27:D3:00:F6:54:4C:9E:8F:FF:CA:3C:60:C2;info.android
hive.googlemapsv2
```

SHA1;your_package_name

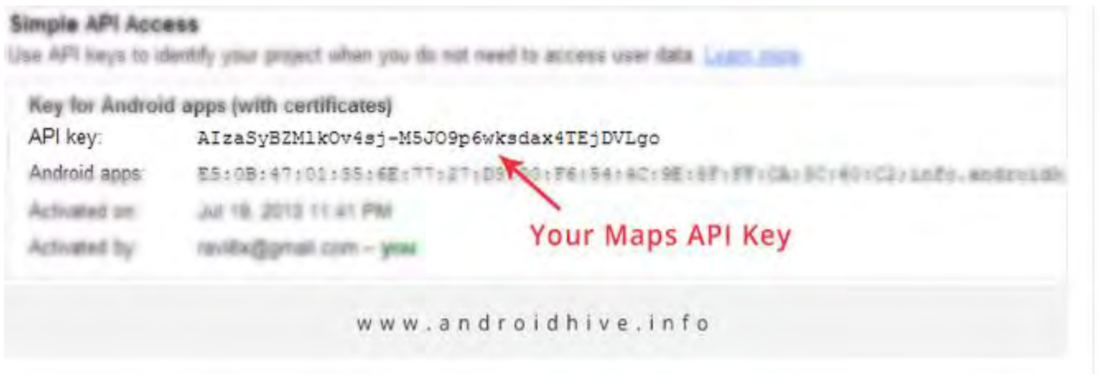
One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:

```
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example
```

Create | Cancel

Figure4.31: Generating the API Key on google console.

8. Note down your google api key which are required to include in our project.


Simple API Access

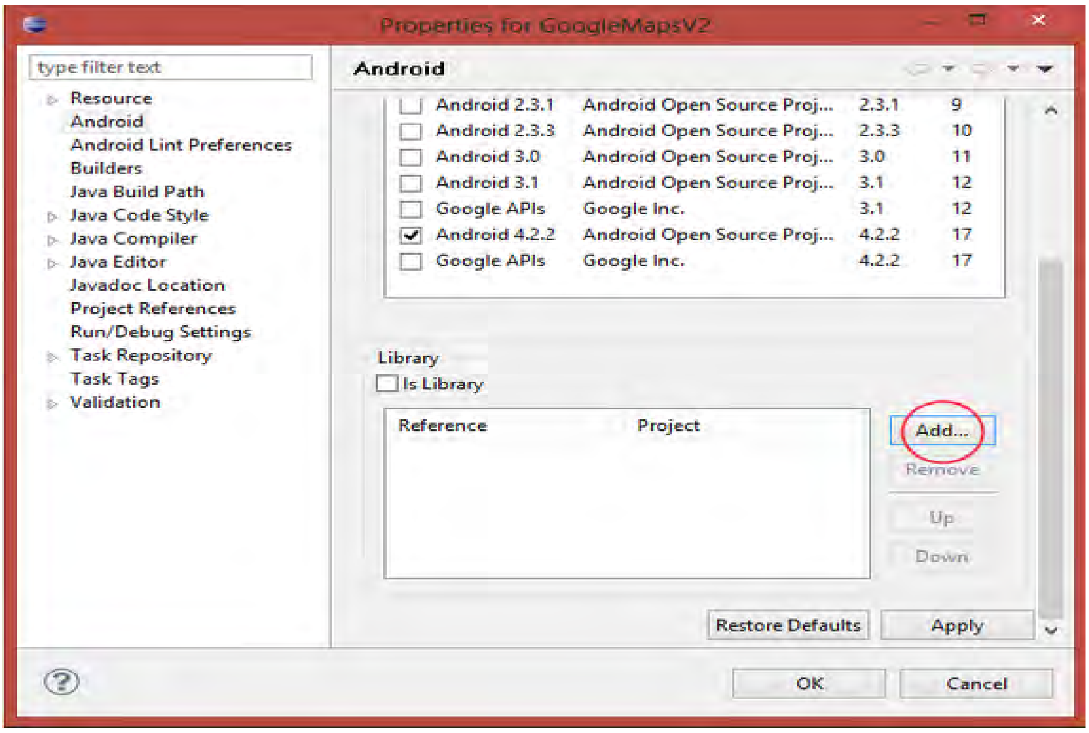
Use API keys to identify your project when you do not need to access user data. Learn more

Key for Android apps (with certificates)

API key:	AIzaSyBZM1kOv4sj-M5JO9p6wksdax4TEjDVLgo
Android apps:	E5:0B:47:01:55:6E:77:27:D9:[illegible]:F6:54:4C:9E:[illegible];info.androidh
Activated on:	Jul 19, 2013 11:41 PM
Activated by:	ravi[illegible]@gmail.com – you

Figure 4.32: Note down the API Key that need to write into our manifest.xml coding file.

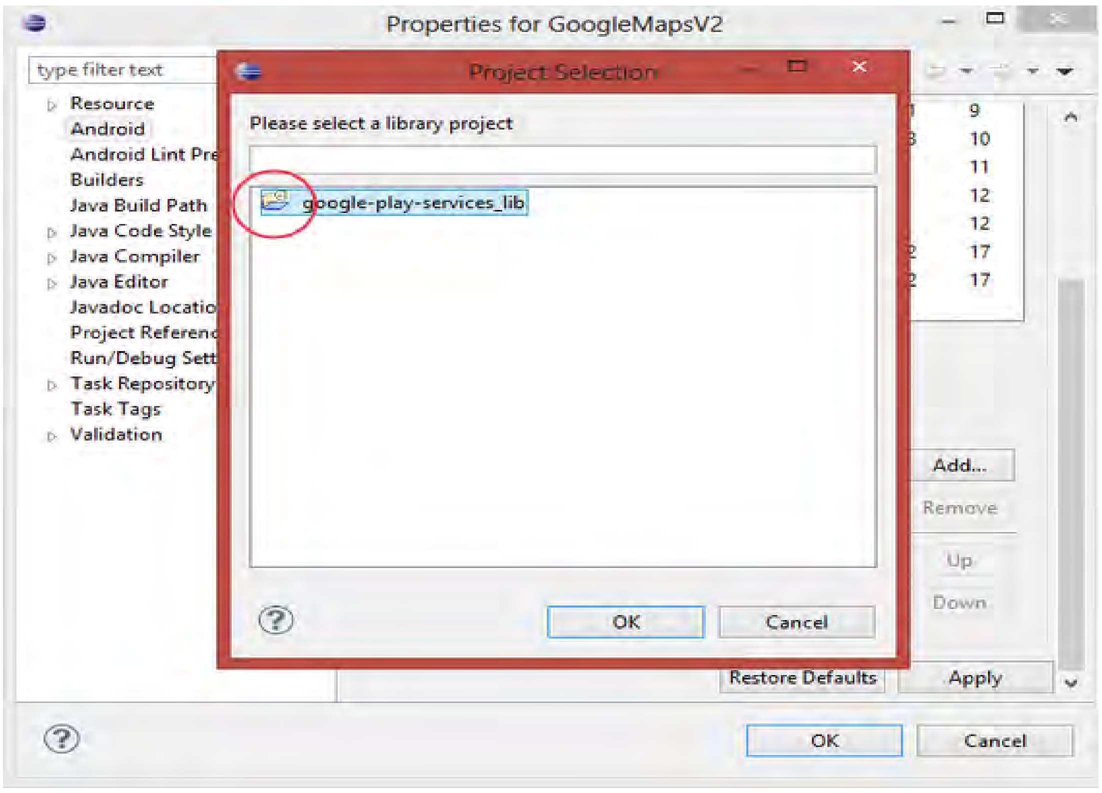
9. Now we have to import the google play service project as a library in our project to support us. Therefore through right click on our project and then select properties option. After that in the properties window on the left hand side select Android. On the right hand side you can see an Add button under library section. Click on this button and then select the google play service project, which we had imported previously.



☐	Android 2.3.1	Android Open Source Proj...	2.3.1	9
☐	Android 2.3.3	Android Open Source Proj...	2.3.3	10
☐	Android 3.0	Android Open Source Proj...	3.0	11
☐	Android 3.1	Android Open Source Proj...	3.1	12
☐	Google APIs	Google Inc.	3.1	12
✓	Android 4.2.2	Android Open Source Proj...	4.2.2	17
☐	Google APIs	Google Inc.	4.2.2	17

☐ Is Library

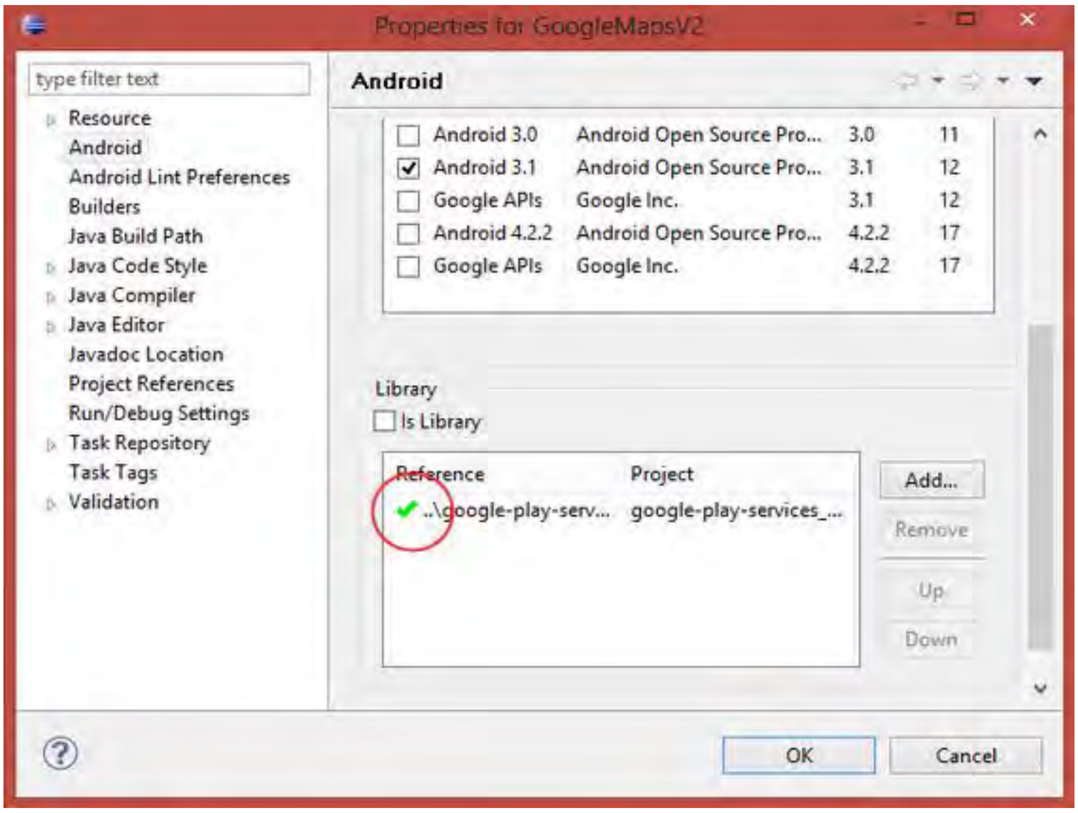
Figure 4.33: Import google play service as library into our project.



Please select a library project

google-play-services_lib

Figure 4.34: Link google play services to the project.



☐ Android 3.0	Android Open Source Pro...	3.0	11
☑ Android 3.1	Android Open Source Pro...	3.1	12
☐ Google APIs	Google Inc.	3.1	12
☐ Android 4.2.2	Android Open Source Pro...	4.2.2	17
☐ Google APIs	Google Inc.	4.2.2	17

Library
☐ Is Library

Reference	Project
✓ ..\google-play-serv...	google-play-services_...

Figure 4.35: Link google play services to the project.

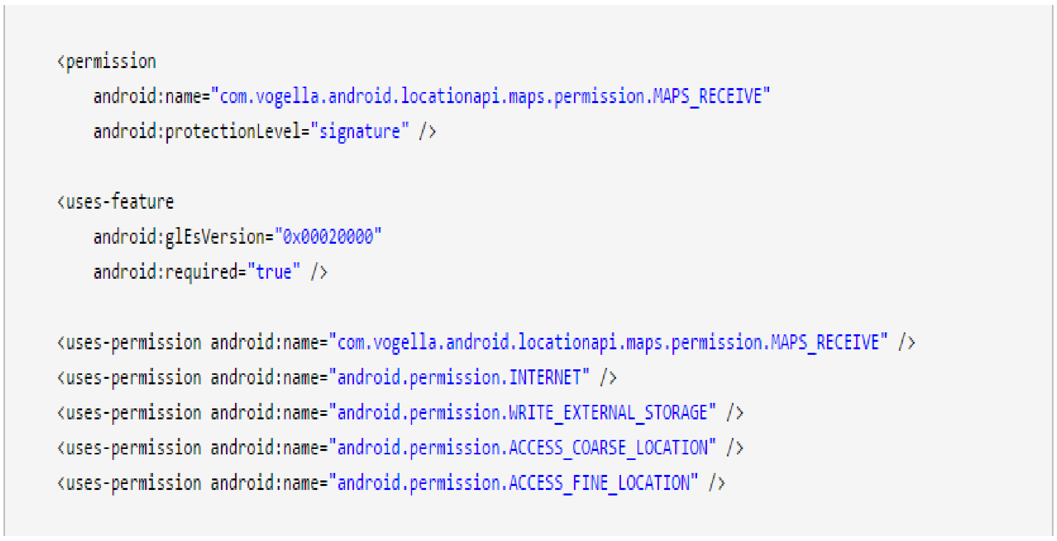
10. After we importing the google play services into our project, now this is a very important step to do so that we can get the google map display on our application. Now we need to open our project's manifest file in eclipse software and then add the google map api key we generate just now.

```
<!-- Goolge Maps API Key -->
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyBZMlkOv4sj-M5JO9p6wksdax4TEjDVLgo" />
```

Figure 4.36: Replace the google api key on the android value.

11. The library provides the "com.google.android.gms.maps. MapFragment" class and the "MapView" class for displaying the map component in our application. Firstly, we need to add some additional information in our "AndroidManifest.xml" file so that we only can use google maps in our application. The figure 4.37 shows the

"AndroidManifest .xml" coding file. To access current location information through location providers, we need to set permissions with android manifest file.



```
<permission
    android:name="com.vogella.android.locationapi.maps.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />

<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

<uses-permission android:name="com.vogella.android.locationapi.maps.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figure 4.37: Project manifest.xml coding file with permission.

ACCESS_COARSE_LOCATION can be used to look for the user's current location by utilizing the network organization and equally well as mobile data. Besides that, **ACCESS_FINE_LOCATION** is actually using of supplying the actual user's current position by utilizing GPS navigation. **INTERNET** authorization is really should within our html coding with regard to using of network provider so that we only able to display google map with a user's current location in our application. **WRITE_EXTERNAL_STORAGE** can be utilized to determine the authorization for the apps to make toward the exterior storage as google maps to store the map data in our external memory. This will be used in the data mining part in later.

12. New google maps are implemented using *MapFragments* which is a subclass of *Fragments* class. First, we need to open our main activity layout file which is activity_main.xml file and then add the following code. I used a RelativeLayout as a parent element. You can also remove it and use MapFragment directly.

```
activity_main.xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</RelativeLayout>
```

Figure 4.38: activity_main.xml coding for google map display on GUI Layout.

13. There are several functions that we can work with google map api v2. For example, we can place a marker on the map to display the user's current location on the google map.

```
// Latitude and longitude
double latitude = ;
double longitude = ;

MarkerOptions marker = new MarkerOptions().position(new
LatLng(latitude, longitude)).title("Hello Maps ");
```

Figure 4.39: Display markers on google map coding.

14. In addition, we also can move the camera on google map to zoom in and zoom out the google map with an animation. Below are the example coding to do this function.

```
googleMap.getUiSettings().setZoomControlsEnabled(false); // true to enable
```

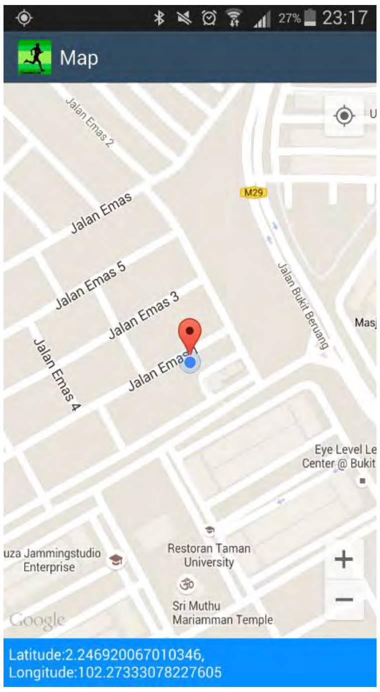
Figure 4.40: Moving camera function coding on google map.

15. The last part of coding that I had used in my application is that the showing current location function for the google map. By adding this function we are able to display the user's current location on google map in our application.

```
googleMap.setMyLocationEnabled(true); // false to disable
```

Figure 4.41: Showing current location example coding.

16. Google map display on our application GUI layout.



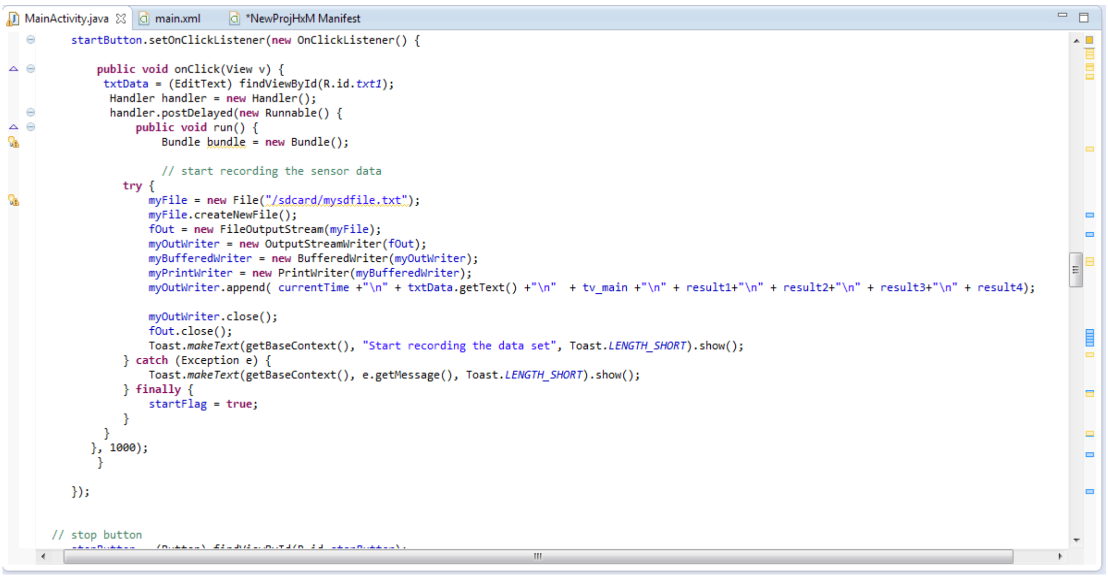
Latitude:2.246920067010346,
Longitude:102.27333078227605

Figure 4.42: Google map display on application layout.

4.4.2.3 Data logging and save into memory of the smartphone device

After all the data had been displayed on the application layout, user can save these data for research purpose. User can get these data in a text file on their SD memory card. First of all, we need to add the android permission for our application so that we only able to write data into an external storage such as SD card in our smartphone device. First, go to our project and then select the file android manifest, in the project manifest file, add the coding *<uses-permission android: name=" android. Permission. WRITE_EXTERNAL_STORAGE" />* into the file.

After that, we need to add some button to our layout so that the application only knows when we want to record our data. So just simply go to the project.xml file to add button on our application layout. After doing the adding button part, now we go to the mainActivity.java file for the project to write our function in our coding. Figure 4.43 show the mainActivity coding to save our data in a file in our SD memory card.



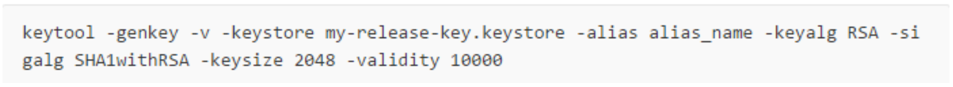
```
MainActivity.java    main.xml    *NewProjHxM Manifest

    startButton.setOnClickListener(new OnClickListener() {

        public void onClick(View v) {
         txtData = (EditText) findViewById(R.id.txt1);
          Handler handler = new Handler();
          handler.postDelayed(new Runnable() {
              public void run() {
                  Bundle bundle = new Bundle();

                  // start recording the sensor data
              try {
                  myFile = new File("/sdcard/mysdfile.txt");
                  myFile.createNewFile();
                  fOut = new FileOutputStream(myFile);
                  myOutWriter = new OutputStreamWriter(fOut);
                  myBufferedWriter = new BufferedWriter(myOutWriter);
                  myPrintWriter = new PrintWriter(myBufferedWriter);
                  myOutWriter.append( currentTime +"\n" + txtData.getText() +"\n"  + tv_main +"\n" + result1+"\n" + result2+"\n" + result3+"\n" + result4);

                  myOutWriter.close();
                  fOut.close();
                  Toast.makeText(getBaseContext(), "Start recording the data set", Toast.LENGTH_SHORT).show();
              } catch (Exception e) {
                  Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
              } finally {
                  startFlag = true;
              }
            }
         }, 1000);
          }

    });

 // stop button
```

Figure 4.43 MainActivity.java coding file to write data into a text file.

4.4.2.4 Publish Android Application to Google Play Store

This segment will discuss about how to publish our android application to the google play store. There are various things we need to take short letter so that our application only able published. It will this discuss clearly part by part later.

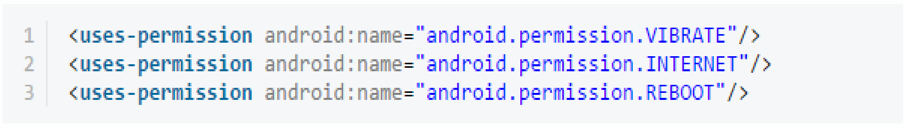
I. Foremost, we need to get our own keystore file on our eclipse software. This is the most important measure, then we need to act so that our application only able publish on google play store later. Keystore is a key used in android platform development to maintain the keys needed to sign in our application prior to distribution. We can get our own keystore by writing the following command as shown in the figure 4.44 in the command prompt to bring forth our own secret key. After we generate it, the keystore file will be saved in our computer. To be make sure that the keystore folder save in a location that you can access and remember. This is we always to need it when we want to publish our application.



```
keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -si
galg SHA1withRSA -keysize 2048 -validity 10000
```

Figure 4.44: Command used to generate our private keystore.

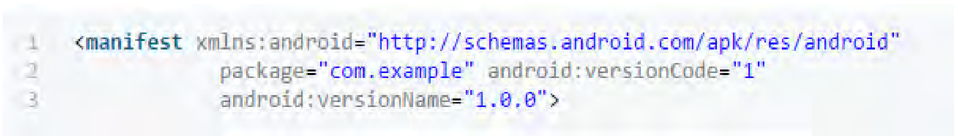
II. After we generate the keystore, now we need to request the necessary android permission in our application so that our application only can be published. First, go to the androidManifest.xml folder by our project and then added the coding in fugure 4.4.2.4.2 in the coding.



```
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.REBOOT"/>
```

Figure 4.45: The permission required to publish our application.

III. After that, we need to configure the version of our application on the androidManifest.xml folder. This is very important step for us because if we skip this step, although we edit a lot of function in our application but we dost not change the version of the application, the android console will not figure out the different for our application compare with the previous application. It will pop out a message with “The apk version as same as the previous version” and this will cause us not able to publish our new application apk file. Figure 4.46 shows the example of the android version name that we need to change in androidManifest.xml folder.

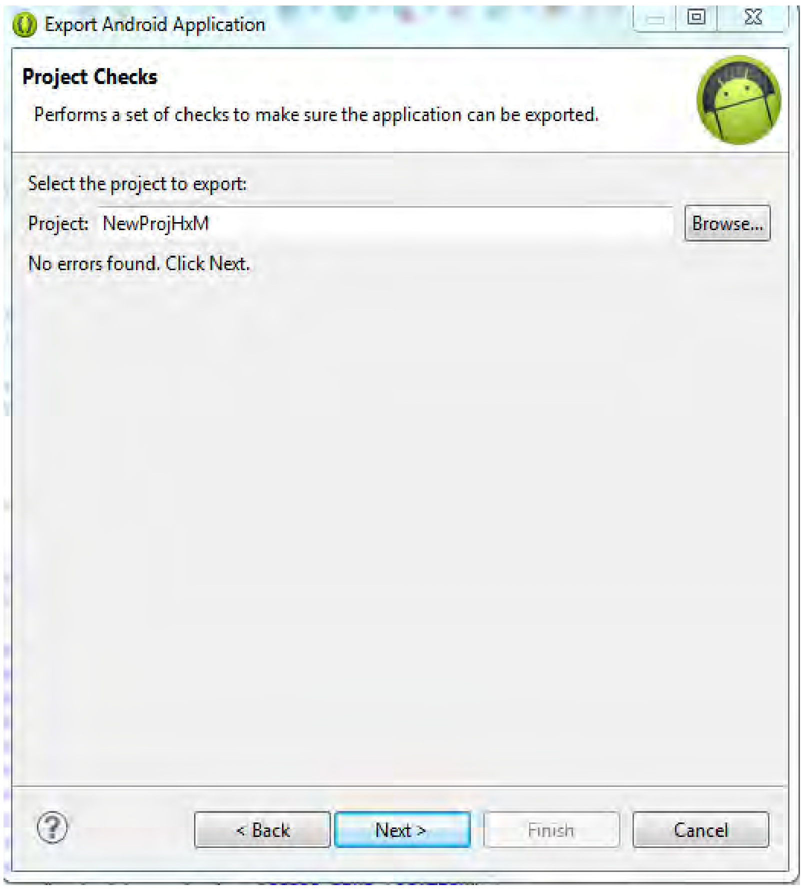


```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example" android:versionCode="1"
          android:versionName="1.0.0">
```

Figure 4.46: Android version Name for android apk version.

IV. When we have done all adding the permission on our application on the eclipse software, now is the time we export our application into an APK file and then prepare for publishing. Android application package (APK) is the package file format that we used to distribute and install the android application software and middleware onto google's android operating system. Our android application must be digitally signed with a certificate that the developer holds so that we only able to publish it. This is why we need to do these steps previously such as generate a keystore and adding the epic version. This is used to ensure the authenticity of the application from the developer.

V. Now we select the project that we want to export it, by selecting the File > Export and then select the export android application in the pop out window. After that we need to compile, sign and zip-align our application before publishing.


Export Android Application

Project Checks

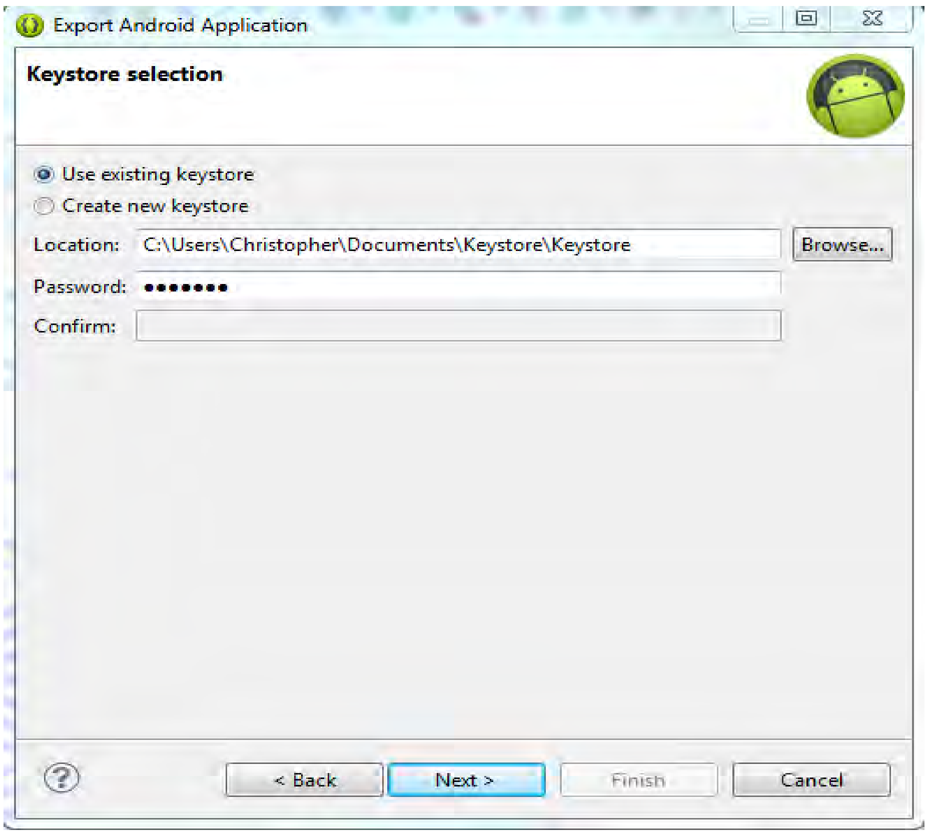
Performs a set of checks to make sure the application can be exported.

Select the project to export:

Project: NewProjHxM

No errors found. Click Next.

Figure 4.47: Select the project that wants to publish.



Keystore selection

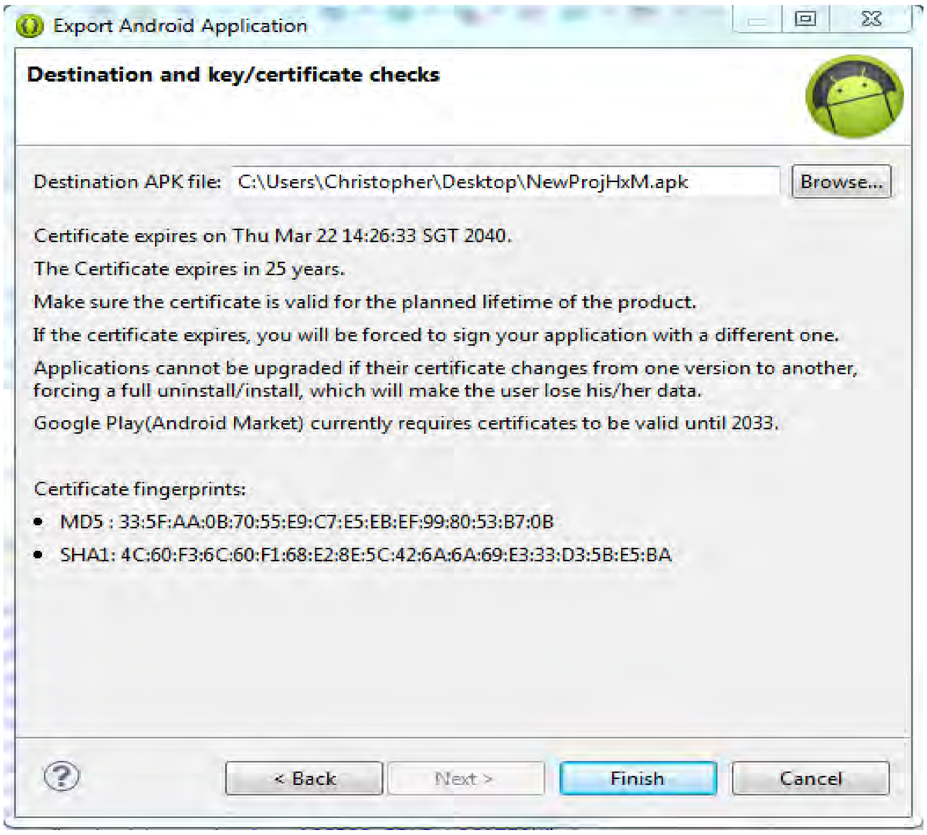
(● Use existing keystore) Create new keystore

Location: C:\Users\Christopher\Documents\Keystore\Keystore

Password: •••••••

Confirm:

Figure 4.48: Using the keystore that we had generated early.



Destination and key/certificate checks

Destination APK file: C:\Users\Christopher\Desktop\NewProjHxM.apk

Certificate expires on Thu Mar 22 14:26:33 SGT 2040.

The Certificate expires in 25 years.

Make sure the certificate is valid for the planned lifetime of the product.

If the certificate expires, you will be forced to sign your application with a different one.

Applications cannot be upgraded if their certificate changes from one version to another, forcing a full uninstall/install, which will make the user lose his/her data.

Google Play(Android Market) currently requires certificates to be valid until 2033.

Certificate fingerprints:

- MD5 : 33:5F:AA:0B:70:55:E9:C7:E5:EB:EF:99:80:53:B7:0B
- SHA1: 4C:60:F3:6C:60:F1:68:E2:8E:5C:42:6A:6A:69:E3:33:D3:5B:E5:BA

Figure 4.49: Select the destination for the apk file has been exported.

VI. Now we only left the final step for us to publish our own app into google play store on google play developer console. First of all, we need to register as a publisher and then setup our own profile, and then read and agree the android market developer distribution agreement and then pay for a registration fee of $25 USD via google checkout. This is a must pay fee and one-time registration fee only for us to publish our application.

VII. After we login our publisher account, click on the "upload new APK to production" to upload our application apk file that we have exported previously. Besides that, we also need to provide our title for our application, short description and some screen shoot for our application before we publish it. Fill in all the information required and then click "save and submit" button to publish our apps. Normally it needs to wait few hours to publish it. And finally we had published our own application on google play store.

4.4.3 Discussion

This segment will discuss about some problem and solution that I had encountered when preparing this application. These are the minor error that possibly occurs when using eclipse software, but it also is a really serious thing we need to take short letter so that we only can able to bear on our task. Eclipse software is a very powerful software tool to assist us develop an Android application, but it also is a very case sensitive error software tool. A small tiny error will cause our application to shut down and not able to function. Normally it will cause the error shut down automatically and pop out a message "Unfortunately (your app) has stopped." This error will force our app not able to unfold, it will push to close down in the background on our application. Thus we cannot neglect any one of the faults in our application.

1. When there is a connection error occurs between eclipse software with our smartphone, we can solve this problem by typing “adb kill-server” and “adb start-server” and restart eclipse software to solve this problem.

2. We can modify our application name manually after we had made an android application project. So by doing this we need to go to the res file on our project and then choosing res > values > strings.xml and then change the value for our package name.

3. We can convert our project software program name by only converting our project package name in AndroidManifest.xml file and deliver it. After that go to scr file and then right click it and select refactor > rename selection. A window will pop out and then type the package name we want for our application in the window and click ok. Why we want to do this is because the default for package name for Android application in eclipse software will be abc.example.com. But if we want to publish our application into google play store we need to eliminate the example in our application package name.

4. We need keystore to update the same application on google play store and it already explain in detail in the part 4.4.2.4. Besides that, the version of the application also needs to be taken note before publishing the application.

5. There may will have missing jar file in our application. So to solve this application, we can go to the build path part and select configure, build path to check there is a missing jar file or not in our project. If all the jar files are there, but the problem still occurs then we can move the missing jar on the top of the order and export folder.

6. Java.null error will occur when there is an empty value that has taken from the coding to do some function. It will cause our application to shut down automatically and show out the message with “Unfortunately (your app)

has stopped.” So by solving this problem you need to go through all your coding, especially for some looping function or some getting value function to make sure there is no empty value to be called and used it. This also is the most common error that beginner for android developer facing it.

CHAPTER V

CONCLUSION & RECOMMENDATION

5.1 Introduction

This project aim to develop an Android application that provides a facility to the researcher to do a data collection of the basic data available from the mobile device for different type of activity. By using this application, the user is able to get the user's heart rate and instant speed reading display on the GUI of the application. Besides that, user also able to get the motion sensors data such as accelerometer, gyroscope and gravity sensors data in the application layout. In addition, users also can see their current location on google map and get their current location point from the application. Lastly, user also can get all of these data in a text file on their device memory card for other purpose

Besides that, for using the heart rate monitoring feature for this application, user need to wear the Zephyr Hxm BT device so that the application only able to display the user's heart rate. By using this feature, there have many possibility feature that can be done with heart rate monitoring features. So this also is an important motivation that this application had been developed. All of these data can be used to develop a lot of applications, there is still a lot of possibilities that can be added into this project in the future. It will be discussed in detail in the project recommendation part.

5.2 Project Recommendation and Sustainability

There is a lot of possibilities can be added into this project in the future. For example, when we gather enough data and manage to detect the activity by using a machine learning, we can automatically predict the daily activity for headphone user and estimate their calorie burn and many other applications. This kind of additional feature can be part of the update on the published app in Google Store. There are some suggestion features that can be added into this application, such as:

a. Step counter by using accelerometer and GPS data collected from this application.
b. Activity tracking system by using 9DOF accelerometer data getting from this application.
c. Running training program with heart rate monitoring function
d. Cardio workout progress tracking such as distance, duration and calories burned calculator
e. Other activity monitoring such as running, jogging, biking, swimming and walking.

5.3 Conclusion

As a conclusion, this application has been successfully developed and published on the google play store to let user download and use it. Besides that, we can conclude that 9DOF of accelerometer data are very useful to develop our own algorithm and then use it to develop our own project.

In addition, this project has increased my knowledge in Android Development by using Eclipse Software. Google Android has a very large market value as Apple's IOS operating system. They are two of the main top operating systems that apply to the most of the smartphone device are using nowadays. I believe that with these android application development knowledge it will give me a lot of opportunity in my future career. Lastly, it can be concluded that this final year project has been successfully completed and all of the objectives have been achieved.

REFERENCE

[1] *"Zephyr HxMTM Smart Heart Rate Monitor"*. [Online] Available: http://zephyranywhere.com [Accessed: Sept 18, 2014].

[2] Praveen Kumar Diwakar, Young Keun Oh, Seung-Hun Park, Young-Ro Yoon. "*Personal Digital Exercise Trainer for Managing, Monitoring and Recording the exercise", 2005.*

[3] Wei-Meng Lee. "*Beginning Android Application Development",* 2011.

[4] Onur Cinar. "*Pro Android C++ with the NDK",* 2012.

[5] Wallance Jackson. *"Android Apps for Absolute Beginneers",* 2012.

[6] Muhammad Shoaib, Hans Scholten, P.J.M. Havinga. "*Towards Physical Activity Recognition Using Smartphone Sensors"*, 2013.

[7] Ekachai Thammasat. "*The Statistical Recognition of Walking, Jogging, and Running Using Smartphone Accelerometers",* 2013.

[8] Arkham Zahri Rakhman, Lukito Edi Nugroho, Widyawan, Kurnianingsih. "*Fall Detection System Using Accelerometer and Gyroscope Based on Smartphone",*2014.

[9] Armir Bujari, Bogdan Licar, Claudio E.Palazzi. "*Movement Pattern Recognition through Smartphone's Accelerometer",* 2012.

[10] Hui-Huang Hsu, Wei-Jan Peng, Timothy K.Shih, Tun-Wen Pai, Ka Lok Man. "*Smartphone Indoor Localization with Accelerometer and Gyroscope",* 2014.

[11] Esmond Mok, Guenther Retscher and Chen Wen. *"Initial Test on the Use of GPS and Sensor Data of Modern Smartphones for Vehicle Tracking in Dense High Rise Environments"*, 2012.

[12] American Heart Association. "*All about Heart Rate (Pulse)",* [Online] Available: *http://www.heart.org/HEARTORG/Conditions/More/.* [Accessed: Sept 17, 2014].

[13] Lakshmanan Sornanathan and Ibrahim Khalil. "*Fitness Monitoring System Based on Heart Rate and SpO2 Level",* 2010.

[14] Yaofeng Wen, Rong Yang, Yu Quan Chen. "*Heart Rate Monitoring in Dynamic Movements from a Wearable System"*, 2008.

[15] Jelena Nikolic-Popovic, Rafik Goubran. "*Measuring Heart Rate, Breathing Rate and Skin Conductance during Exercise"*, 2011.

[16] "*What is Eclipse"*, [Online], and Available: http://www.eclipse.org/org/. [Accessed: Nov, 25, 2014].

[17] "*UML Class Diagrams: An Agile Introduction",* Available: http://www.agilemodeling.com /classDiagram. [Accessed: May, 30, 2015].

[18] *"Google map android api",* Available: *http://www.developers.google/.* [Accessed: Feb, 15, 2015].

[19] *Motion Sensor,* Available: http://www.developers.android.com/sensors/. [Accessed: Jan, 7, 2015].

APPENDIX A

Zephyr™ *HRM BT*™

USER GUIDE

PACKAGE CONTENTS

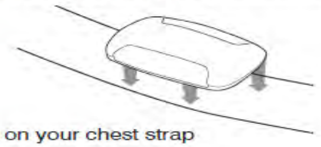
Congratulations on purchasing a Zephyr™ Bluetooth™ Heart Rate Monitor. This package contains the following:

i. Zephyr™ Bluetooth™ enabled Heart Rate Monitor.
ii. Zephyr™ USB charging cradle.
iii. Zephyr™ patented Smart-Fabric Heart Rate sensing chest strap.

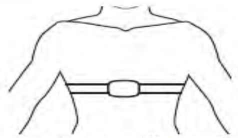
OPERATING INSTRUCTIONS

Wearing your device

i. Clip your HxM device to your chest strap



ii. Put on your chest strap



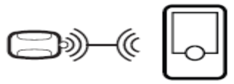
[1] *Putting the strap on will turn your HxM on.*
[2] *Taking the chest strap off will turn your HxM off.*

Note: In order to ensure a good connection, it might be necessary to sprinkle a small amount of water on the fabric sensors before wearing.

Connecting to your Mobile Phone

i. Make sure you are wearing your HxM (so that it is turned on).

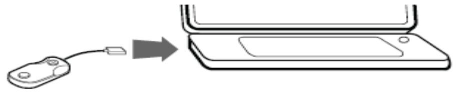
ii. You will need to first 'pair' your HxM with your Mobile Phone with passcode "1234". (See your phone manual for BlueTooth pairing instructions) Following this, you'll need to follow the instructions from the provider of the application you are using on your Phone. You should find this information on their website.



Charging the battery

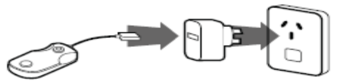
i. Clip your device into the cradle.

¹ *Connect your cradle to a USB port on your PC,*



or;

² *Connect your cradle to a USB wall charger.*



ii. The green LED indicates that the charger is connected to a power source.

iii. The red LED indicates that the HxM is charging.

¹ *The red light will turn off when the device is fully charged.*

iv. 90% charge in 1 hour.

v. 100% charge in 3 hours.

vi. A full battery will give approximately 26 hours of use.

TROUBLE SHOOTING

You cannot connect to your Mobile Phone Application.

i. Have you charged the battery in your HxM?

ii. If you have very dry skin, you may need to moisten the sensors on your chest strap to allow the HxM to turn on. Try connecting again.

iii. Have you installed the fitness application on your Mobile Phone correctly?

You lose the connection to your phone during a workout.

i. Is the HxM charged?

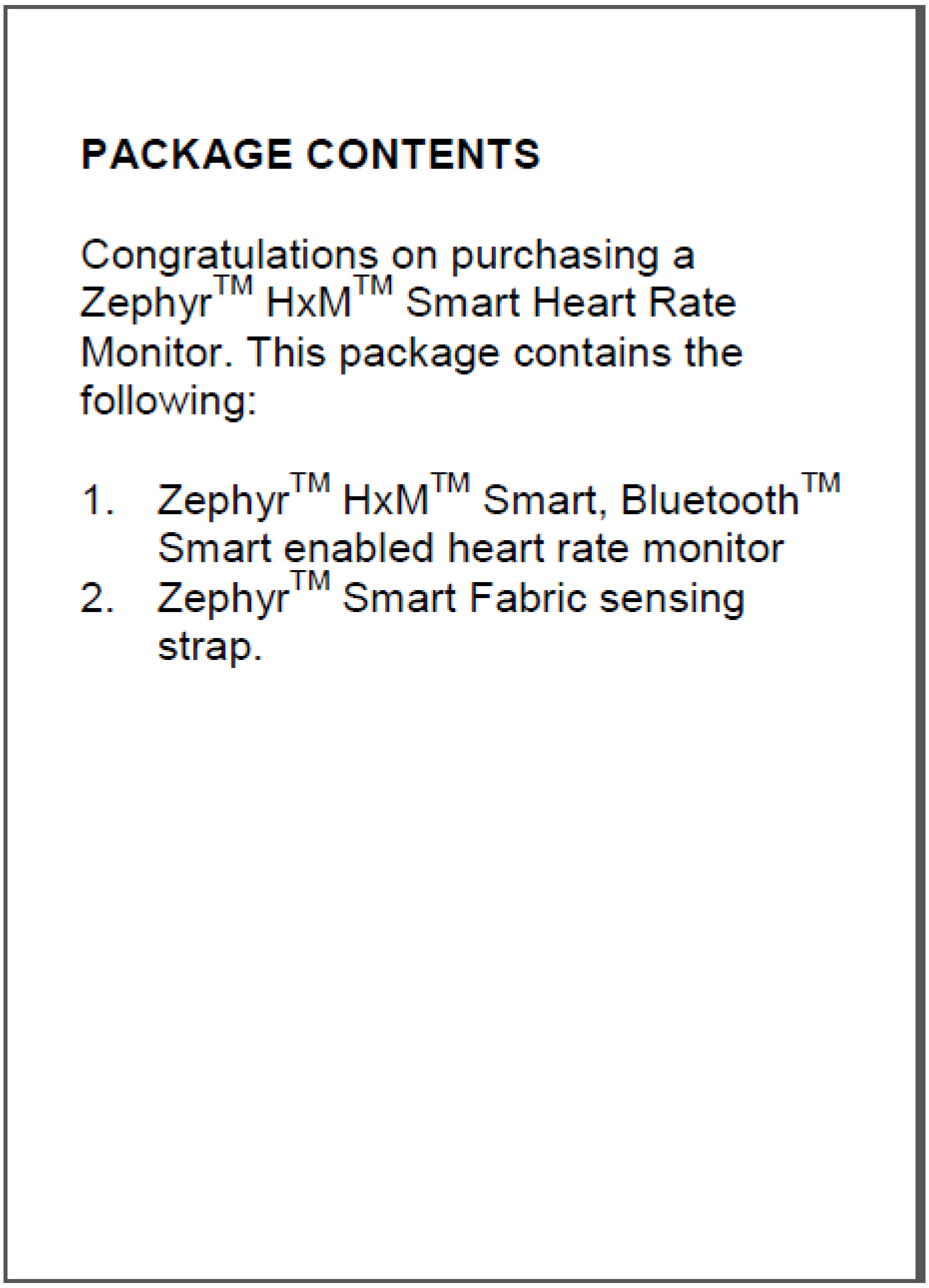
ii. Are there other Bluetooth™ devices active within close range?

¹ *Sometimes if there are a number of Bluetooth™ devices working within close range, they can interfere with the signal from your HxM to your PC. Try moving away, and connect again.*

APPENDIX B

Zephyr ^{TM} HxM^{TM} Smart

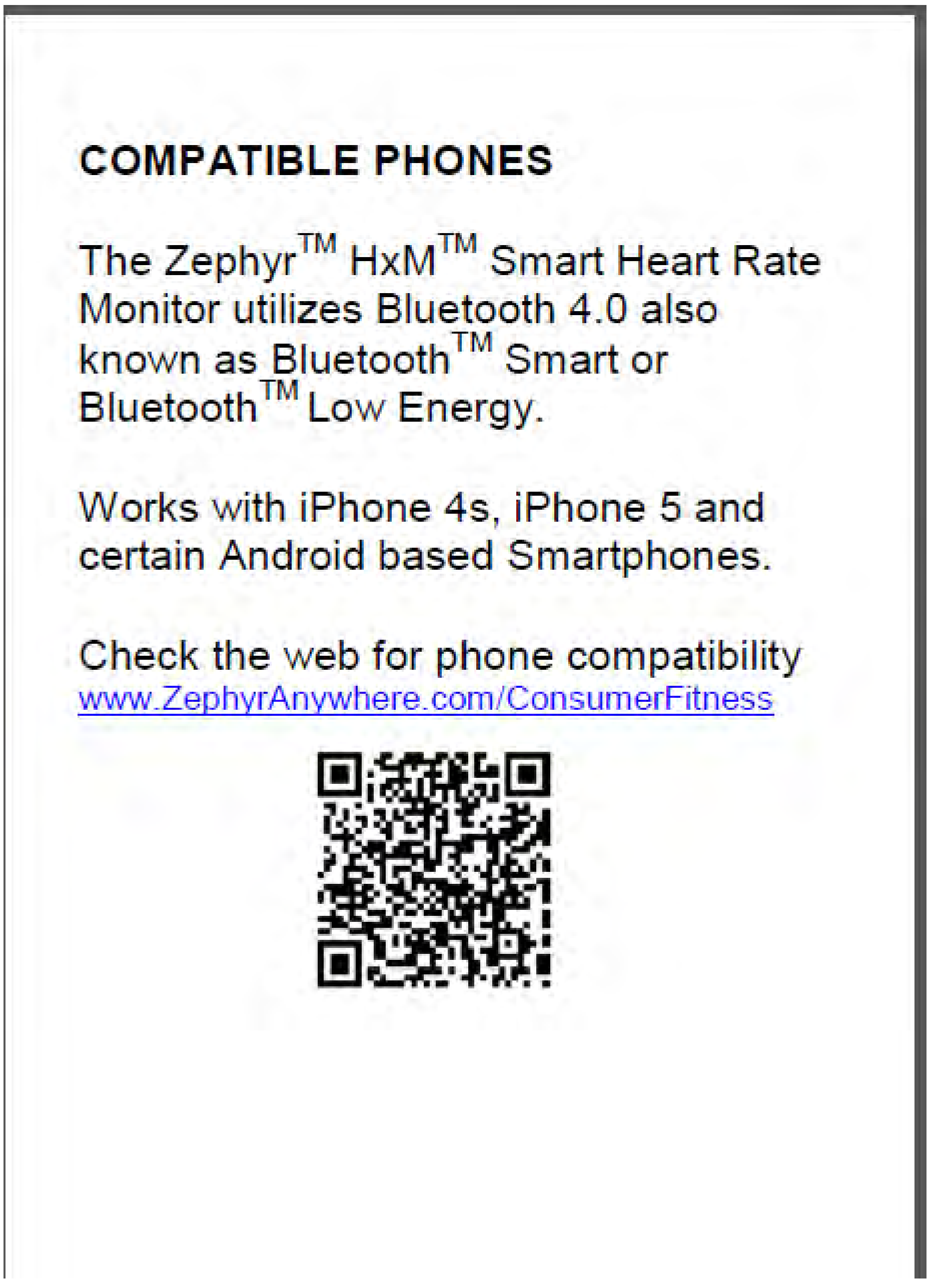
USER GUIDE



PACKAGE CONTENTS

Congratulations on purchasing a Zephyr^{TM} HxM^{TM} Smart Heart Rate Monitor. This package contains the following:

1. Zephyr^{TM} HxM^{TM} Smart, Bluetooth^{TM} Smart enabled heart rate monitor
2. Zephyr^{TM} Smart Fabric sensing strap.

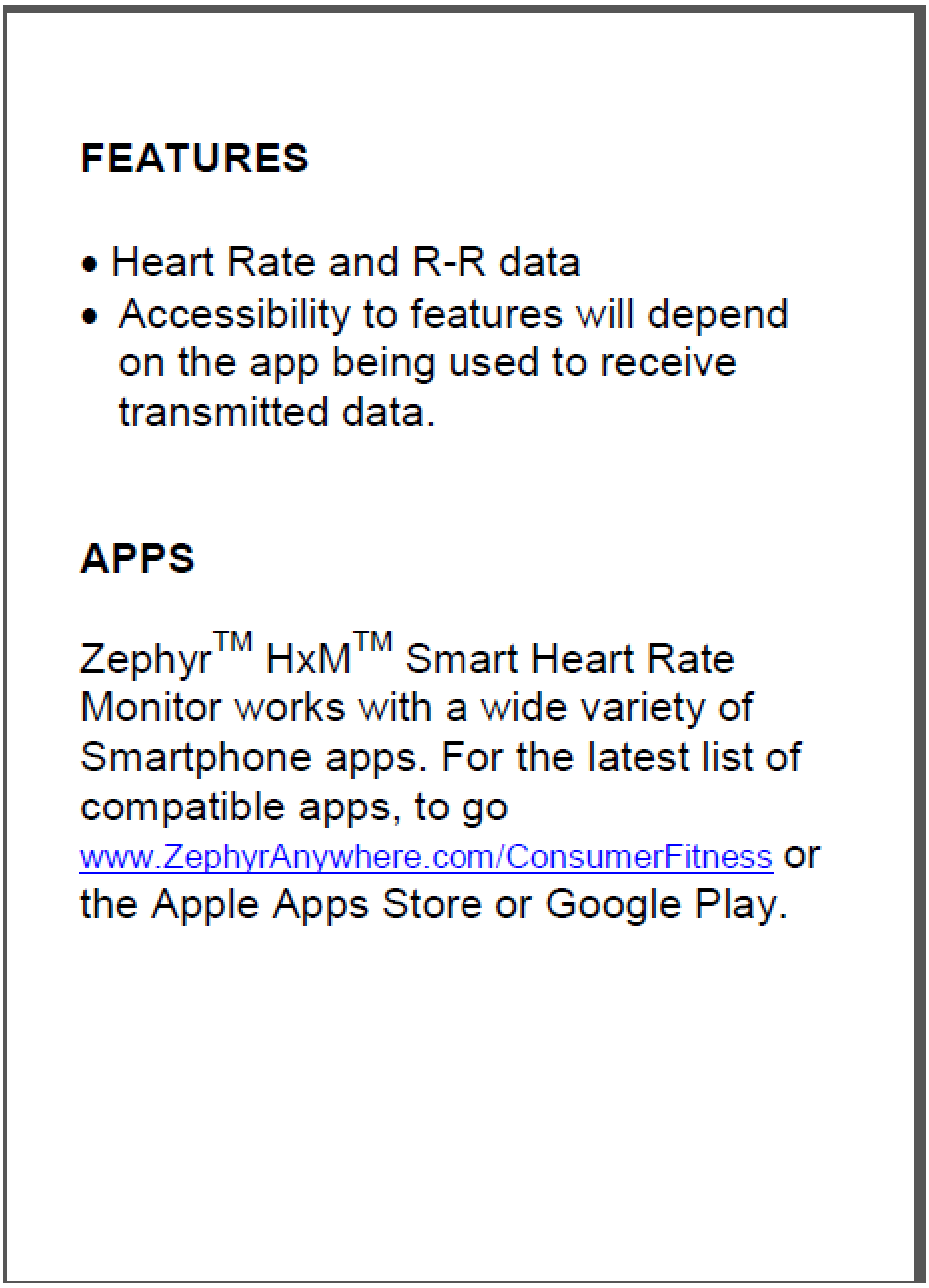


COMPATIBLE PHONES

The Zephyr^{TM} HxM^{TM} Smart Heart Rate Monitor utilizes Bluetooth 4.0 also known as Bluetooth^{TM} Smart or Bluetooth^{TM} Low Energy.

Works with iPhone 4s, iPhone 5 and certain Android based Smartphones.

Check the web for phone compatibility
www.ZephyrAnywhere.com/ConsumerFitness

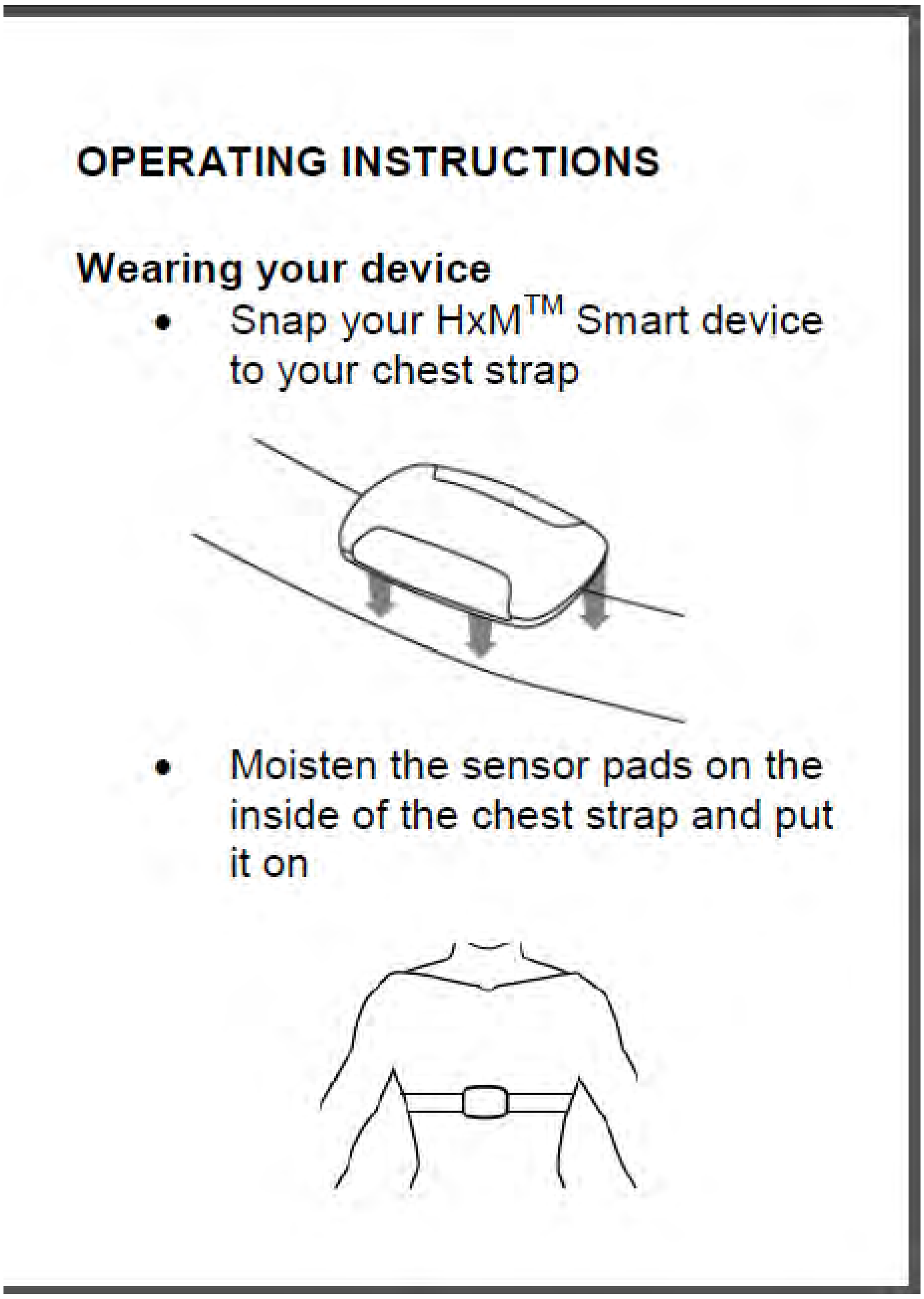


FEATURES

- Heart Rate and R-R data
- Accessibility to features will depend on the app being used to receive transmitted data.

APPS

Zephyr^{TM} HxM^{TM} Smart Heart Rate Monitor works with a wide variety of Smartphone apps. For the latest list of compatible apps, to go www.ZephyrAnywhere.com/ConsumerFitness or the Apple Apps Store or Google Play.

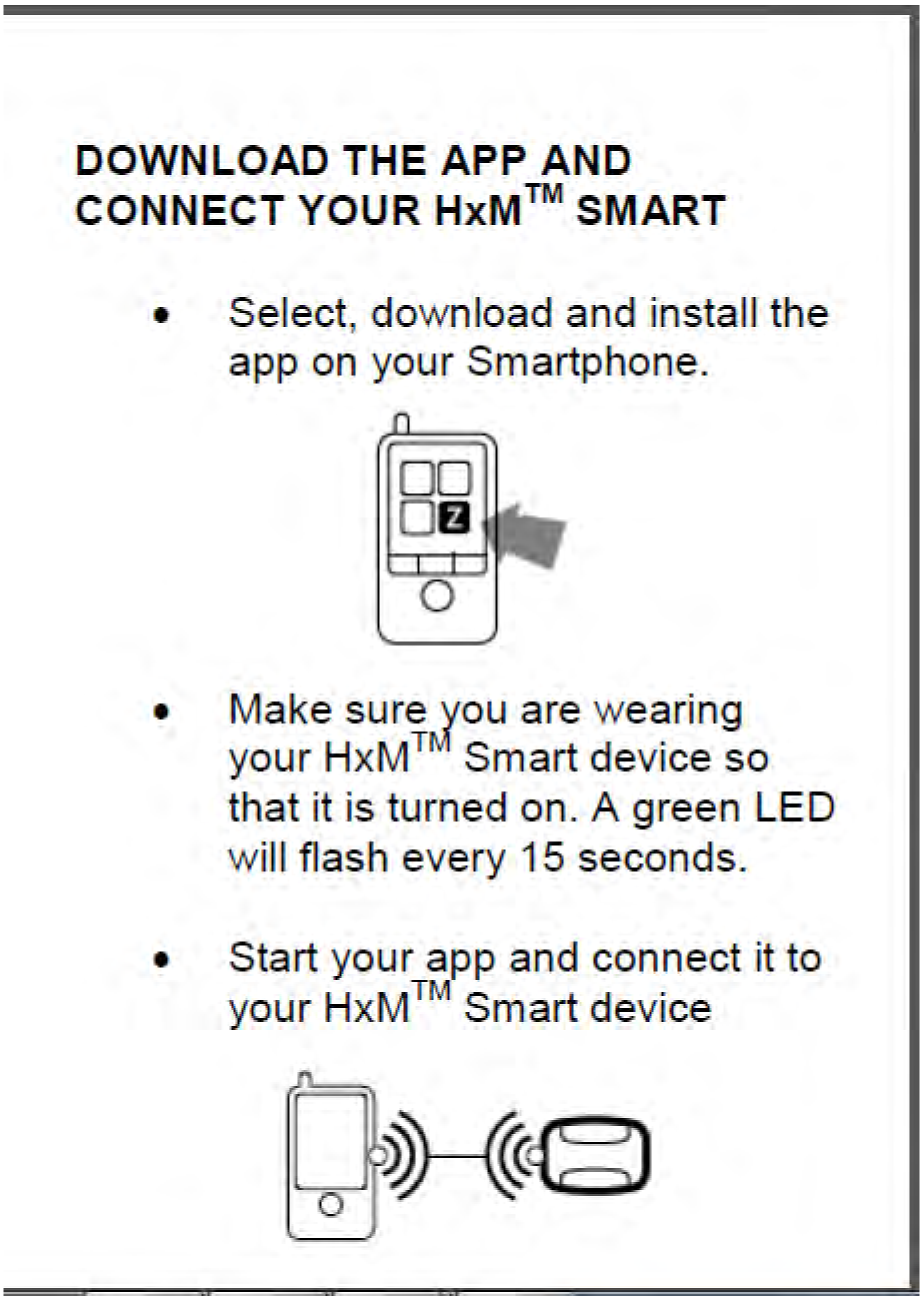


OPERATING INSTRUCTIONS

Wearing your device

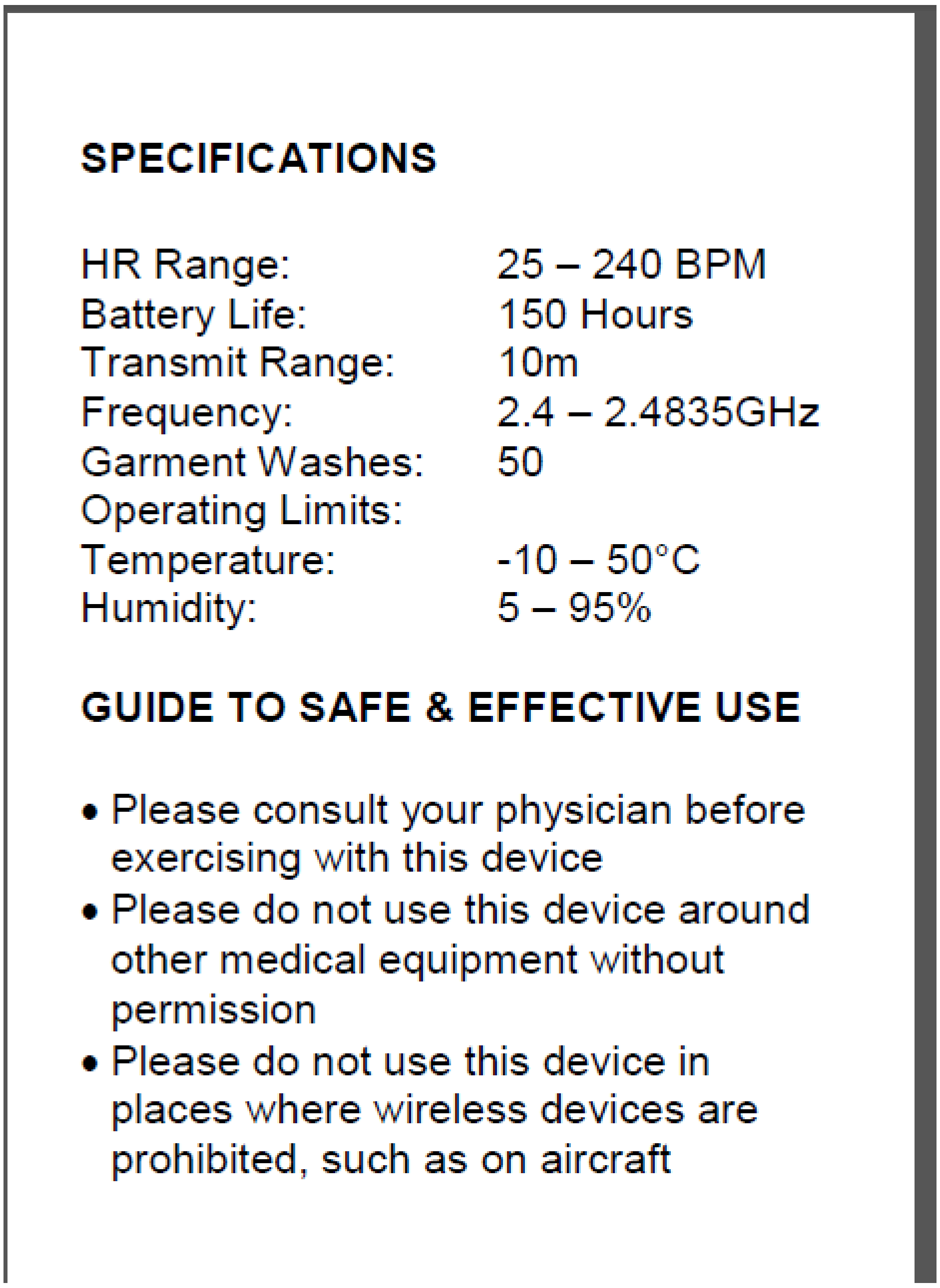
- Snap your HxM^{TM} Smart device to your chest strap
- Moisten the sensor pads on the inside of the chest strap and put it on

APPENDIX C



DOWNLOAD THE APP AND CONNECT YOUR HxM^{TM} SMART

- Select, download and install the app on your Smartphone.
- Make sure you are wearing your HxM^{TM} Smart device so that it is turned on. A green LED will flash every 15 seconds.
- Start your app and connect it to your HxM^{TM} Smart device

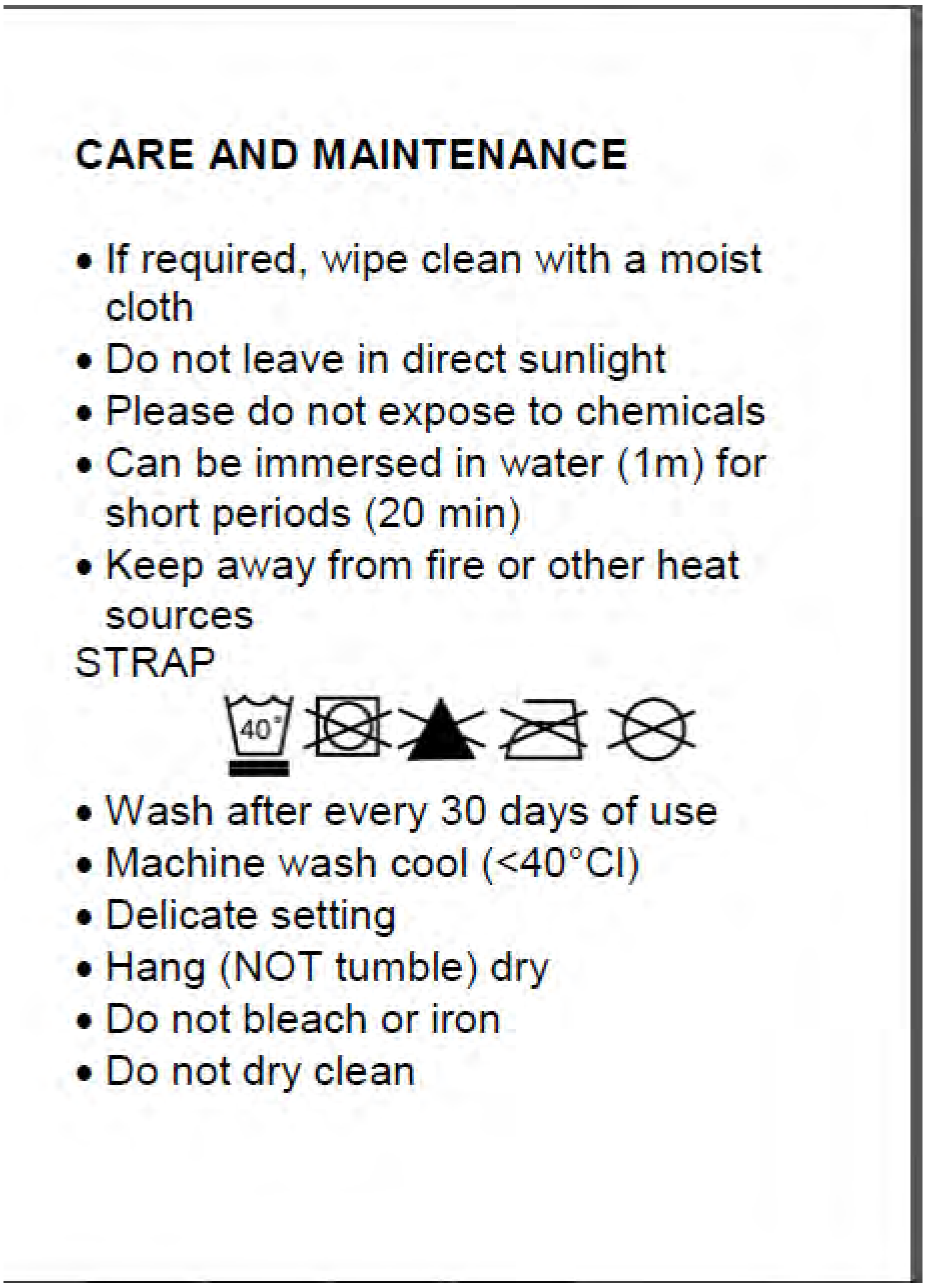


SPECIFICATIONS

HR Range:	25 – 240 BPM
Battery Life:	150 Hours
Transmit Range:	10m
Frequency:	2.4 – 2.4835GHz
Garment Washes:	50
Operating Limits:	
Temperature:	-10 – 50°C
Humidity:	5 – 95%

GUIDE TO SAFE & EFFECTIVE USE

- Please consult your physician before exercising with this device
- Please do not use this device around other medical equipment without permission
- Please do not use this device in places where wireless devices are prohibited, such as on aircraft

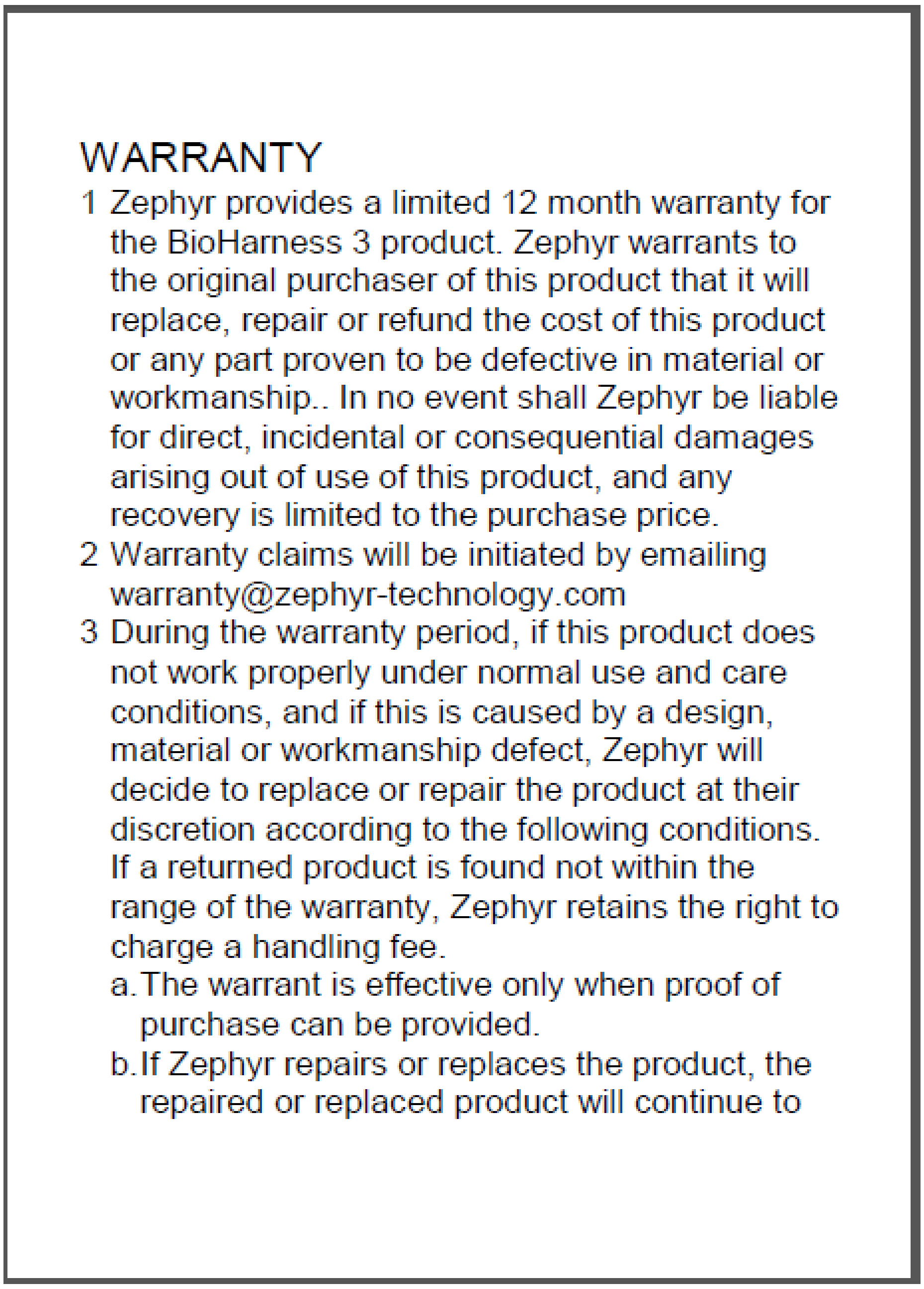


CARE AND MAINTENANCE

- If required, wipe clean with a moist cloth
- Do not leave in direct sunlight
- Please do not expose to chemicals
- Can be immersed in water (1m) for short periods (20 min)
- Keep away from fire or other heat sources

STRAP

- Wash after every 30 days of use
- Machine wash cool (<40°CI)
- Delicate setting
- Hang (NOT tumble) dry
- Do not bleach or iron
- Do not dry clean



WARRANTY

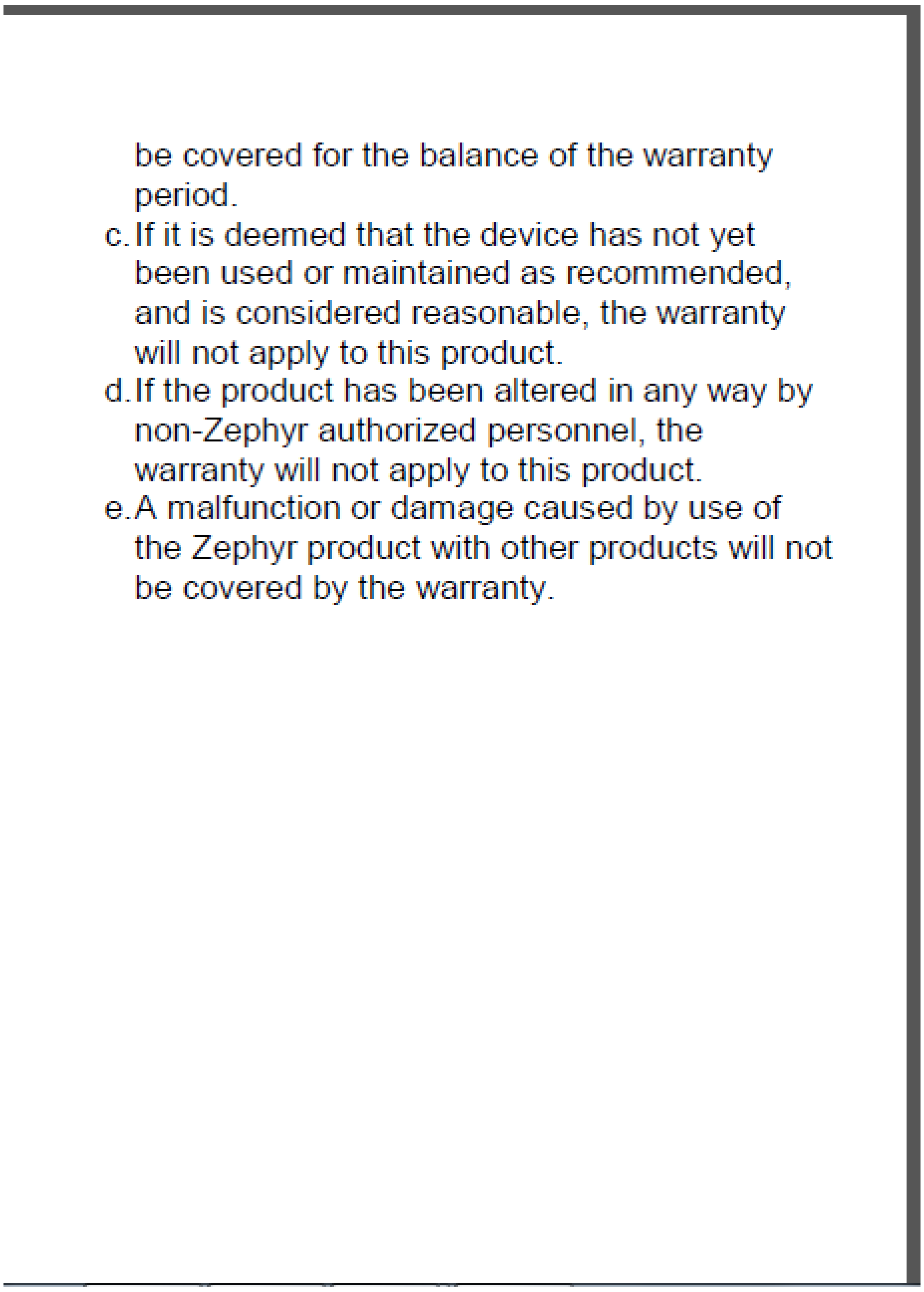
1 Zephyr provides a limited 12 month warranty for the BioHarness 3 product. Zephyr warrants to the original purchaser of this product that it will replace, repair or refund the cost of this product or any part proven to be defective in material or workmanship.. In no event shall Zephyr be liable for direct, incidental or consequential damages arising out of use of this product, and any recovery is limited to the purchase price.

2 Warranty claims will be initiated by emailing warranty@zephyr-technology.com

3 During the warranty period, if this product does not work properly under normal use and care conditions, and if this is caused by a design, material or workmanship defect, Zephyr will decide to replace or repair the product at their discretion according to the following conditions. If a returned product is found not within the range of the warranty, Zephyr retains the right to charge a handling fee.

   a. The warrant is effective only when proof of purchase can be provided.

   b. If Zephyr repairs or replaces the product, the repaired or replaced product will continue to

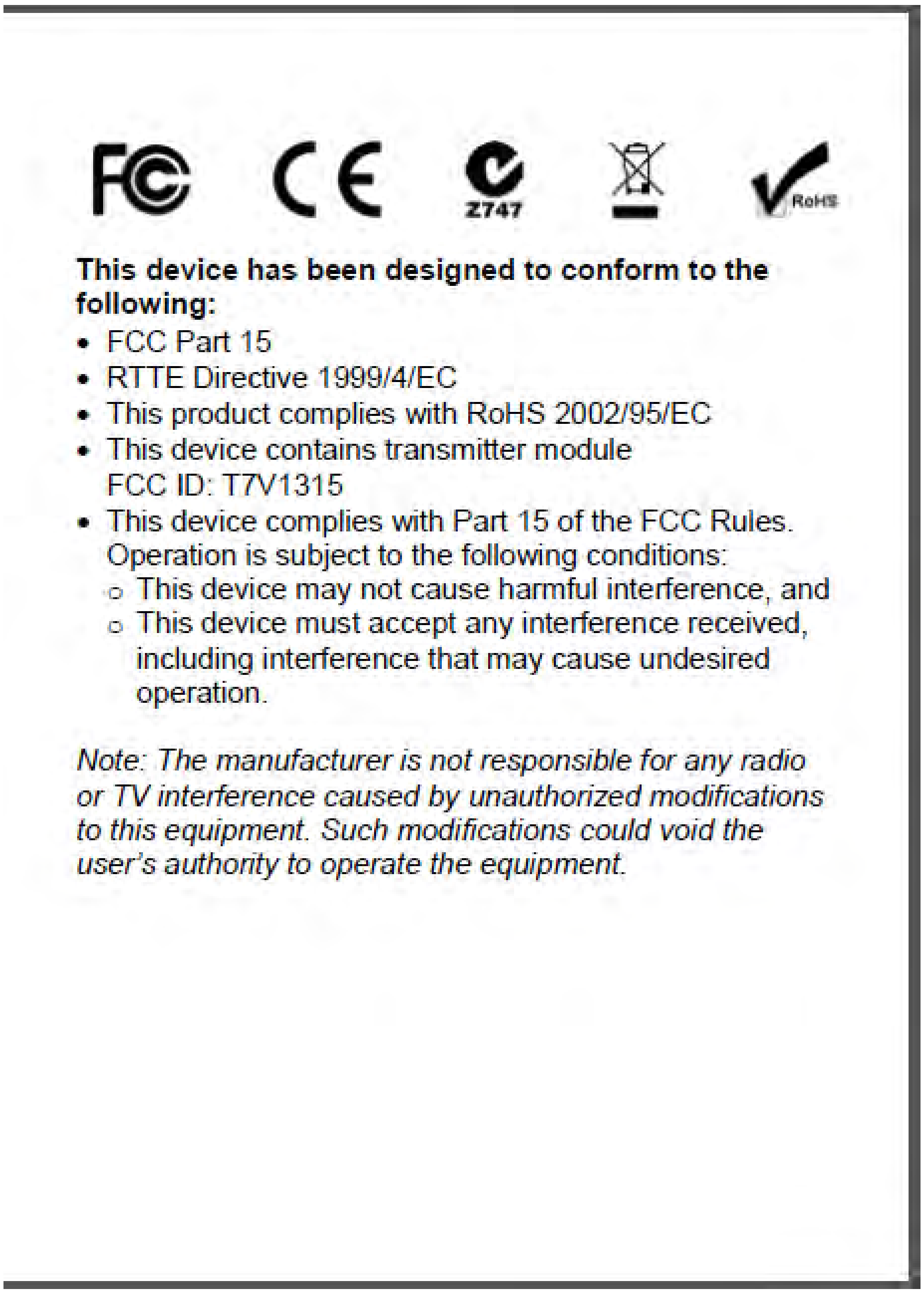


be covered for the balance of the warranty period.

c. If it is deemed that the device has not yet been used or maintained as recommended, and is considered reasonable, the warranty will not apply to this product.

d. If the product has been altered in any way by non-Zephyr authorized personnel, the warranty will not apply to this product.

e. A malfunction or damage caused by use of the Zephyr product with other products will not be covered by the warranty.



This device has been designed to conform to the following:

- FCC Part 15
- RTTE Directive 1999/4/EC
- This product complies with RoHS 2002/95/EC
- This device contains transmitter module
  FCC ID: T7V1315
- This device complies with Part 15 of the FCC Rules. Operation is subject to the following conditions:
  - This device may not cause harmful interference, and
  - This device must accept any interference received, including interference that may cause undesired operation.

Note: The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. Such modifications could void the user's authority to operate the equipment.

Contact:

Zephyr Technology
1 Annapolis St
Suite 200
Annapolis
MD 21401
USA

Tel:	+1 (443) 569-3603
Fax:	+1 (443) 926-9402
Email:	support@zephytech.zendesk.com info@zephyranywhere.com sales@zephyranywhere.com
Web:	www.zephyranywhere.com

9700.0191 2012-09-20

APPENDIX C

```
@SuppressLint("NewApi") @SuppressWarnings("deprecation")
public class MainActivity<DrawerLayout> extends Activity implements
                           SensorEventListener, LocationListener,
ListView.OnItemClickListener{

       private static final String SensorDelay = null;
       private static final SensorEventListener SensorEventListener =
null;
       BluetoothAdapter adapter = null;
       BTClient _bt;
       ZephyrProtocol _protocol;
       NewConnectedListener _NConnListener;
       private final int HEART_RATE = 0x100;
       private final int INSTANT_SPEED = 0x101;
       private TextView view;
       private TextView tv1;
       private TextView tv2;
    protected LocationManager locationManager;
       protected LocationListener locationListener;
       protected Context context;
       protected String latitude,longitude;
       protected boolean gps_enabled,network_enabled;
       private SensorManager mSensorManager;
       private Sensor mAccelerometer;
       private FileWriter writer;
       private Handler handler = new Handler();
       EditText txtData;
    Button startButton;
    Button stopButton;
    float[] acceleration = new float[3];

    File myFile;
    FileOutputStream fOut;
    OutputStreamWriter myOutWriter;
    BufferedWriter myBufferedWriter;
    PrintWriter myPrintWriter;
    boolean stopFlag = false;
    boolean startFlag = false;
    boolean isFirstSet = true;
       TextView txtLat;
       String lat;
       String provider;
       SensorManager sensorManager;
       String currentTime = (String) DateFormat.format("yyyy-MM-dd
hh:mm:ss", new Date());
       static String exits = "No";

       /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //for sensor
        view = (TextView) findViewById(R.id.Accelerometer);
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        sensorManager.registerListener(this,

       sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                    SensorManager.SENSOR_DELAY_NORMAL);

sensorManager.registerListener(this,sensorManager.getDefaultSensor(Sensor.
),SensorManager.SENSOR_DELAY_NORMAL);

        view = (TextView) findViewById(R.id.Gyroscope);
        sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
                    SensorManager.SENSOR_DELAY_NORMAL);
```

```
sensorManager.registerListener(this,sensorManager.getDefaultSensor(Sensor.
),SensorManager.SENSOR_DELAY_UI);

        view = (TextView) findViewById(R.id.Gravity);
        sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY),
                    SensorManager.SENSOR_DELAY_NORMAL);

sensorManager.registerListener(this,sensorManager.getDefaultSensor(Sensor.
),SensorManager.SENSOR_DELAY_NORMAL);

        setContentView(R.layout.main);
        txtLat = (TextView) findViewById(R.id.GPS);
        locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
this);

        /*Sending a message to android that we are going to initiate a
pairing request*/
        IntentFilter filter = new
IntentFilter("android.bluetooth.device.action.PAIRING_REQUEST");
        /*Registering a new BTBroadcast receiver from the Main Activity
context with pairing request event*/
       this.getApplicationContext().registerReceiver(new
BTBroadcastReceiver(), filter);
        // Registering the BTBondReceiver in the application that the
status of the receiver has changed to Paired
        IntentFilter filter2 = new
IntentFilter("android.bluetooth.device.action.BOND_STATE_CHANGED");
       this.getApplicationContext().registerReceiver(new BTBondReceiver(),
filter2);

      //Obtaining the handle to act on the CONNECT button
        TextView tv = (TextView) findViewById(R.id.labelStatusMsg);
             String ErrorText  = "Not Connected to HxM ! !";
              tv.setText(ErrorText);

        Button btnConnect = (Button) findViewById(R.id.ButtonConnect);
        if (btnConnect != null)
        {
             btnConnect.setOnClickListener(new OnClickListener() {
                    private Runnable r;

                           public void onClick(View v) {
                           String BhMacID = "00:07:80:9D:8A:E8";
                           //String BhMacID = "00:07:80:88:F6:BF";
                           adapter = BluetoothAdapter.getDefaultAdapter();

                           Set<BluetoothDevice> pairedDevices =
adapter.getBondedDevices();

                           if (pairedDevices.size() > 0)
                           {
                        for (BluetoothDevice device : pairedDevices)
                        {
                           if (device.getName().startsWith("HXM"))
                           {
                                  BluetoothDevice btDevice = device;
                                  BhMacID = btDevice.getAddress();
                                break;

                           }
                        }
                           }
```

```
                              //BhMacID = btDevice.getAddress();
                              BluetoothDevice Device =
adapter.getRemoteDevice(BhMacID);
                              String DeviceName = Device.getName();
                              _bt = new BTClient(adapter, BhMacID);
                              _NConnListener = new
NewConnectedListener(Newhandler,Newhandler);
                              _bt.addConnectedEventListener(_NConnListener);

                              TextView tv1 =
(EditText)findViewById(R.id.labelHeartRate);
                              tv1.setText("000");

                               tv1 =
(EditText)findViewById(R.id.labelInstantSpeed);
                               tv1.setText("0.0");

                              if(_bt.IsConnected())
                              {
                                     _bt.start();
                                     TextView tv = (TextView)
findViewById(R.id.labelStatusMsg);
                                     String ErrorText  = "Connected to HxM
"+DeviceName;
                                            tv.setText(ErrorText);

                                            //Reset all the values to 0s
                              }
                              else
                              {
                                     TextView tv = (TextView)
findViewById(R.id.labelStatusMsg);
                                     String ErrorText  = "Unable to
Connect !";
                                            tv.setText(ErrorText);
                              }
                       }
                });
        }

        /*Obtaining the handle to act on the DISCONNECT button*/
        Button btnDisconnect = (Button)
findViewById(R.id.ButtonDisconnect);
        if (btnDisconnect != null)
        {
               btnDisconnect.setOnClickListener(new OnClickListener() {
                              @Override
                              /*Functionality to act if the button DISCONNECT
is touched*/
                              public void onClick(View v) {
                                     // TODO Auto-generated method stub
                                     /*Reset the global variables*/
                                     TextView tv = (TextView)
findViewById(R.id.labelStatusMsg);
                              String ErrorText  = "Disconnected from HxM!";
                                      tv.setText(ErrorText);

                                     /*This disconnects listener from acting
on received messages*/

       _bt.removeConnectedEventListener(_NConnListener);
                                     /*Close the communication with the
device & throw an exception if failure*/
                                     _bt.Close();

                              }
               });
        }
        Button button = (Button) findViewById(R.id.map);
        button.setOnClickListener(new View.OnClickListener(){
            @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    if(exits.equals("YES")){
                        finish();}
                    else{
                        Intent Intent = new
Intent(MainActivity.this,Map.class);
                        startActivity(Intent);
                    }
                }

        });


    // start button
       startButton = (Button) findViewById(R.id.startButton);
       startButton.setOnClickListener(new OnClickListener() {

           public void onClick(View v) {
              txtData = (EditText) findViewById(R.id.txt1);
               Handler handler = new Handler();
             handler.postDelayed(new Runnable() {
               public void run() {
                       Bundle bundle = new Bundle();

                       // start recording the sensor data
                try {
                    myFile = new File("/sdcard/mysdfile.txt");
                    myFile.createNewFile();
                    fOut = new FileOutputStream(myFile);
                    myOutWriter = new OutputStreamWriter(fOut);
                    myBufferedWriter = new BufferedWriter(myOutWriter);
                    myPrintWriter = new PrintWriter(myBufferedWriter);
                    myOutWriter.append( currentTime +"\n" +
txtData.getText() +"\n"  + tv_main +"\n" + result1+"\n" + result2+"\n" +
result3+"\n" + result4);

                    myOutWriter.close();
                    fOut.close();
                    Toast.makeText(getBaseContext(), "Start recording the
data set", Toast.LENGTH_SHORT).show();
                } catch (Exception e) {
                    Toast.makeText(getBaseContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
                } finally {
                    startFlag = true;
                }
             }
           }, 1000);
            }

       });


    // stop button
       stopButton = (Button) findViewById(R.id.stopButton);
       stopButton.setOnClickListener(new OnClickListener() {

           public void onClick(View v) {
                // stop recording the sensor data
                try {
                    stopFlag = true;
                    Toast.makeText(getBaseContext(), "Done recording the
data set", Toast.LENGTH_SHORT).show();
```

```
                } catch (Exception e) {
                    Toast.makeText(getBaseContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
                }
            }
        });

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    }

        private class BTBondReceiver extends BroadcastReceiver {
                @Override
                public void onReceive(Context context, Intent intent) {
                        Bundle b = intent.getExtras();
                        BluetoothDevice device =
adapter.getRemoteDevice(b.get("android.bluetooth.device.extra.DEVICE").toS
tring());
                        Log.d("Bond state", "BOND_STATED = " +
device.getBondState());
                }
     }

    private class BTBroadcastReceiver extends BroadcastReceiver {
                @Override
                public void onReceive(Context context, Intent intent) {
                        Log.d("BTIntent", intent.getAction());
                        Bundle b = intent.getExtras();
                        Log.d("BTIntent",
b.get("android.bluetooth.device.extra.DEVICE").toString());
                        Log.d("BTIntent",
b.get("android.bluetooth.device.extra.PAIRING_VARIANT").toString());
                        try {
                                BluetoothDevice device =
adapter.getRemoteDevice(b.get("android.bluetooth.device.extra.DEVICE").toS
tring());
                                Method m =
BluetoothDevice.class.getMethod("convertPinToBytes", new Class[]
{String.class} );
                                byte[] pin = (byte[])m.invoke(device, "1234");
                                m = device.getClass().getMethod("setPin", new
Class [] {pin.getClass()});
                                Object result = m.invoke(device, pin);
                                Log.d("BTTest", result.toString());
                        } catch (SecurityException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        } catch (NoSuchMethodException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        } catch (IllegalArgumentException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (IllegalAccessException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (InvocationTargetException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
     }

    private String tv_main;
       private String result1;
       private String result2;
       private String result3;
       private String result4;
    final  Handler Newhandler = new Handler(){
        public void handleMessage(Message msg)
        {
                TextView tv;
                switch (msg.what)
                {
                case HEART_RATE:
                        String HeartRatetext =
msg.getData().getString("HeartRate");
                        tv = (EditText)findViewById(R.id.labelHeartRate);
                        System.out.println("Heart Rate Info is "+
HeartRatetext);
                        if (tv != null) tv.setText(HeartRatetext);
                        tv_main = HeartRatetext;
                break;

                case INSTANT_SPEED:
                        String InstantSpeedtext =
msg.getData().getString("InstantSpeed");
                        tv = (EditText)findViewById(R.id.labelInstantSpeed);
                        if (tv != null)tv.setText(InstantSpeedtext);

                break;

                }
        }

    };



        private TextView view1;
        private float a;
        private float b;
        private float c;

        //case sensor
        @SuppressLint("DefaultLocale") @Override
        public void onSensorChanged(SensorEvent event) {

                // TODO Auto-generated method stub
                float[] value = event.values;
                float[][] cached_values;
                switch (event.sensor.getType()) {
                case Sensor.TYPE_ACCELEROMETER:
                                float x = value[0];
                                float y = value[1];
                                float z = value[2];
                             result1 = String.format(" x=%.4f\n y=%.4f\n
z=%.4f", x,y,z);
                                view1 = (TextView)
findViewById(R.id.labelAccelerometer);
                                view1.setText(result1);
                break;

                case Sensor.TYPE_GYROSCOPE:
                        float[] value1 = event.values;
                        a = value1[0];
                                b = value1[1];
                                c = value1[2];
                          result2 = String.format(" x=%.3f\n y=%.3f\n z=%.3f",
a,b,c);
                          view1 = (TextView)
findViewById(R.id.labelGyroscope);
                                view1.setText(result2);
                break;

                case Sensor.TYPE_GRAVITY:
                        float[] value2 = event.values;
```

```
                    a = value2[0];
                        b = value2[1];
                        c = value2[2];
                     result3 = String.format(" x=%.4f\n y=%.4f\n z=%.4f",
a,b,c);
                     view1 = (TextView) findViewById(R.id.labelGravity);
                        view1.setText(result3);
            break;



            }
        }


        private void save() {
            // TODO Auto-generated method stub
        }
         @Override
           protected void onResume() {
               super.onResume();
               // register this class as a listener for the sensors
               sensorManager.registerListener(this,
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
SensorManager.SENSOR_DELAY_NORMAL);
               sensorManager.registerListener(this,
sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
SensorManager.SENSOR_DELAY_NORMAL);
               sensorManager.registerListener(this,
sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
SensorManager.SENSOR_DELAY_NORMAL);
           }

           @Override
           protected void onPause() {
               // unregister listener
               super.onPause();
               sensorManager.unregisterListener(this);
           }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
            // TODO Auto-generated method stub
            sensorManager.registerListener(SensorEventListener,
              sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
              SensorManager.SENSOR_DELAY_UI);
         sensorManager.registerListener(SensorEventListener,

sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
              SensorManager.SENSOR_DELAY_UI);
        }

        @Override
        public void onLocationChanged(Location location) {
        txtLat = (TextView) findViewById(R.id.labelGPS);
        result4 = String.format("Latitude=%.4f
Longitude=%.4f",location.getLatitude(), location.getLongitude() );
        txtLat.setText(result4);
        }

        @Override
        public void onProviderDisabled(String provider) {
        Log.d("Latitude","disable");
        }

        @Override
        public void onProviderEnabled(String provider) {
        Log.d("Latitude","enable");
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle
extras) {
        Log.d("Latitude","status");
        }


        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position,
                        long id) {
                // TODO Auto-generated method stub

        }




    @Override
    public void onBackPressed() {
        new AlertDialog.Builder(this)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setTitle("Closing Tracker4Life")
            .setMessage("Are you sure you want to leave Tracker4Life?")
            .setPositiveButton("Yes", new
DialogInterface.OnClickListener()
        {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                finish();
            }

                        })
                                .setNegativeButton("No", null)
                                .show();
                        }

    public void onStartClick(View view) {
        mSensorManager.registerListener(this, mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
    }

    public void onStopClick(View view) {
        mSensorManager.unregisterListener(this);
    }


}
```

APPENDIX D

```
        <ScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/drawer_layout"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <RelativeLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/heart_rate1" >

            <TextView
                android:id="@+id/Welcometext"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/hello"
                android:textColor="#FFFF00"
                android:textSize="20sp" />

            <Button
                android:id="@+id/ButtonConnect"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_below="@+id/Welcometext"
                android:text="Connect" >
            </Button>

            <Button
                android:id="@+id/ButtonDisconnect"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignTop="@id/ButtonConnect"
                android:layout_toRightOf="@id/ButtonConnect"
                android:text="Disconnect" >
            </Button>

            <EditText
                android:id="@+id/labelHeartRate"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_below="@+id/ButtonConnect"
                android:text="000" >
            </EditText>

            <TextView
                android:id="@+id/HRTextBox"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/ButtonConnect"
                android:text="Heart Rate"
                android:textColor="#FFFF00"
                android:textSize="20sp" />

            <EditText
                android:id="@+id/labelInstantSpeed"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_below="@+id/labelHeartRate"
                android:text="000" >
            </EditText>

            <TextView
                android:id="@+id/InstantSpeed"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
        android:layout_alignTop="@+id/labelInstantSpeed"
        android:text="Instant Speed"
        android:textColor="#FFFF00"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/Accelerometer"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/labelInstantSpeed"
        android:text="Accelerometer(m/s^2)"
        android:textColor="#FFFF00"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/labelAccelerometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignTop="@+id/Accelerometer"
        android:ems="10"
        android:inputType="textMultiLine"
        android:width="120dp"/>

    <TextView
        android:id="@+id/Gyroscope"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/labelAccelerometer"
        android:text="Gyroscope(rad/s)"
        android:textColor="#FFFF00"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/labelGyroscope"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/labelAccelerometer"
        android:ems="10"
        android:inputType="textMultiLine"
        android:width="120dp" />

    <EditText
        android:id="@+id/labelGravity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/labelGyroscope"
        android:ems="10"
        android:inputType="textMultiLine"
        android:width="120dp" />

    <TextView
        android:id="@+id/GPS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/labelGravity"
        android:text="GPS"
        android:textColor="#FFFF00"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/labelGPS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/labelGravity"
        android:layout_centerHorizontal="true"
        android:ems="10"
        android:width="150dp" />

    <Button
        android:id="@+id/map"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/HRTextBox"
        android:layout_toRightOf="@+id/ButtonDisconnect"
        android:text="MAP" />

    <EditText
        android:id="@+id/labelStatusMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/StatusTextBox"
        android:ems="10"
        android:text="Status Message Box" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/Gravity"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/labelGyroscope"
        android:text="Gravity(m/s^2)"
        android:textColor="#FFFF00"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/StatusTextBox"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/labelGPS"
        android:text="Status Message"
        android:textColor="#FFFF00"
        android:textSize="20sp" />

    <Button
        android:id="@+id/stopButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/labelGPS"
        android:layout_below="@+id/labelStatusMsg"
        android:layout_marginTop="68dp"
        android:onClick="onStopClick"
        android:text="Stop Record" />

    <Button
        android:id="@+id/startButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/stopButton"
        android:layout_alignBottom="@+id/stopButton"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="21dp"
        android:onClick="onStartClick"
        android:text="Start Record" />
```

```
                <EditText
                    android:id="@+id/txt1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignParentLeft="true"
                    android:layout_alignRight="@+id/labelStatusMsg"
                    android:layout_below="@+id/labelStatusMsg"
                    android:layout_marginTop="14dp"
                    android:ems="10"
                    android:hint="Enter type of activity here..."
                    android:textSize="15sp" />

            </RelativeLayout>

    </ScrollView>
```

APPENDIX E

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.R_Tracker4Life"
    android:versionCode="4"
    android:versionName="1.0" >

    <supports-screens
        android:anyDensity="true"
        android:largeScreens="true"
        android:normalScreens="true"
        android:resizeable="true"
        android:smallScreens="true" />

    <uses-sdk
        android:minSdkVersion="12"
        android:targetSdkVersion="21" />

    <permission
        android:name="com.R_Tracker4Life.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission
android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.BROADCAST_STICKY" />
       <uses-permission
android:name="in.wptrafficanalyzer.locationdistancetimemapv2.permission.MA
PS_RECEIVE" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/
>


    <!-- Required OpenGL ES 2.0. for Maps V2 -->
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:icon="@drawable/logo2"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:configChanges="keyboardHidden|orientation"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
            android:name="org.apache.cordova.DroidGap"
            android:configChanges="orientation|keyboardHidden"
            android:label="@string/app_name" >
            <intent-filter>
            </intent-filter>
        </activity>
        <activity
            android:name=".DataBasePage"
            android:label="@string/title_activity_data_base_page" >
        </activity>
        <activity
            android:name=".ViewPager"
            android:label="@string/title_activity_view_pager" >
        </activity>
        <activity
            android:name=".Map"
            android:label="@string/title_activity_map" >
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyCBmf5f5QUmXmteNKP5-rcvAkUMHW9qBN0" />

        <activity
            android:name=".Tracking"
            android:label="@string/title_activity_tracking" >
        </activity>
    </application>

</manifest>
```

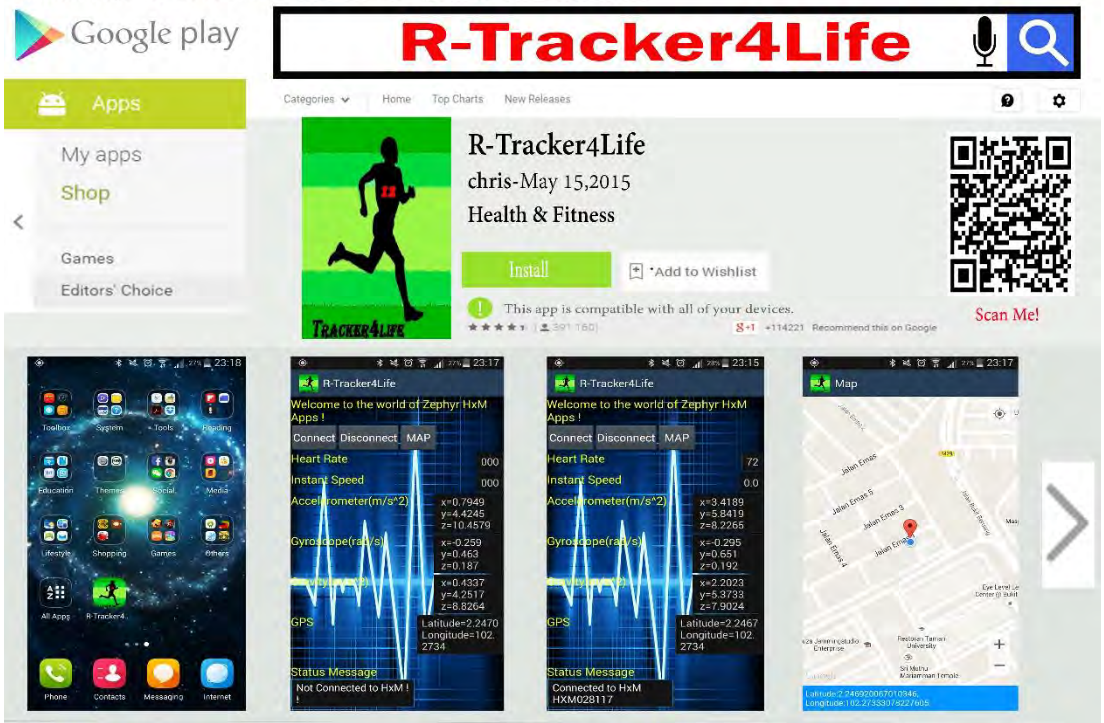
APPENDIX F

iNOTEK UTeM UNIVERSITI TEKNIKAL MALAYSIA MELAKA

HUMAN-TECH INTERACTION

FAKULTI KEJURUTERAAN ELEKTRONIK DAN KEJURUTERAAN KOMPUTER

R-Tracker4Life



R-Tracker4Life
chris-May 15,2015
Health & Fitness
This app is compatible with all of your devices.

R-Tracker4Life
Welcome to the world of Zephyr HxM Apps !
Connect Disconnect MAP
Heart Rate 000
Instant Speed 000
Accelerometer(m/s^2) x=0.7949 y=4.4245 z=10.4579
Gyroscope(rad/s) x=-0.259 y=0.463 z=0.187
Gravity(m/s^2) x=0.4337 y=4.2517 z=8.8264
GPS Latitude=2.2470 Longitude=102.2734
Status Message
Not Connected to HxM ! !

R-Tracker4Life
Welcome to the world of Zephyr HxM Apps !
Connect Disconnect MAP
Heart Rate 72
Instant Speed 0.0
Accelerometer(m/s^2) x=3.4189 y=5.8419 z=8.2265
Gyroscope(rad/s) x=-0.295 y=0.651 z=0.192
Gravity(m/s^2) x=2.2023 y=5.3733 z=7.9024
GPS Latitude=2.2467 Longitude=102.2734
Status Message
Connected to HxM HXM028117

Latitude:2.246920067010346, Longitude:102.27333074227605

DESCRIPTION

This **Android app** is developed to help **researcher** in the field of **health science** and data mining to monitor and **track the heart rate** for different type of activity. Besides that, this app also able to monitor 9 degrees of freedom for Accelerometer, Gyroscope and Gravity that are available in most Android smartphone.

OBJECTIVE

-To **integrate** the heart rate monitoring, accelerometer, gyroscope, gravity sensors and GPS positioning in the activity monitoring app.
-To **relate** the activity types with accelerometer, gyroscope and gravity sensors
-To **study** an **individual heart rate behavior** while doing various activity..

NOVELITY & INVENTIVENESS

A lot of study has been done in developing an accurate activity detector or tracker as well as callory calculation and the research in this area is growing. This app provide an easy facility for the researcher in this area to gather their own information of **Accelerometer, Gyroscope** and **Gravity** sensor**, GPS** and **heart rate** value for their research purpose.

POTENTIAL OF COMMERCIALIZATION

The preliminary version of the app had already been **published** in google play store to be downloaded for **free** and our target user for now is a researcher in health science and artificial intelligent field. Until now, the apps already been updated three time and we try to update the new version every week. In future development, we will **improve** interm of the data representation (speed, callory, activity and etc) and target for sport

REFERENCES

[1] HxM Android API User Guide 2011.
[2] Bluetooth HxM API Guide 2011.
[3] Android Motion Sensor.
http://developer.android.com/guide/topics/sensors/sensors_motion

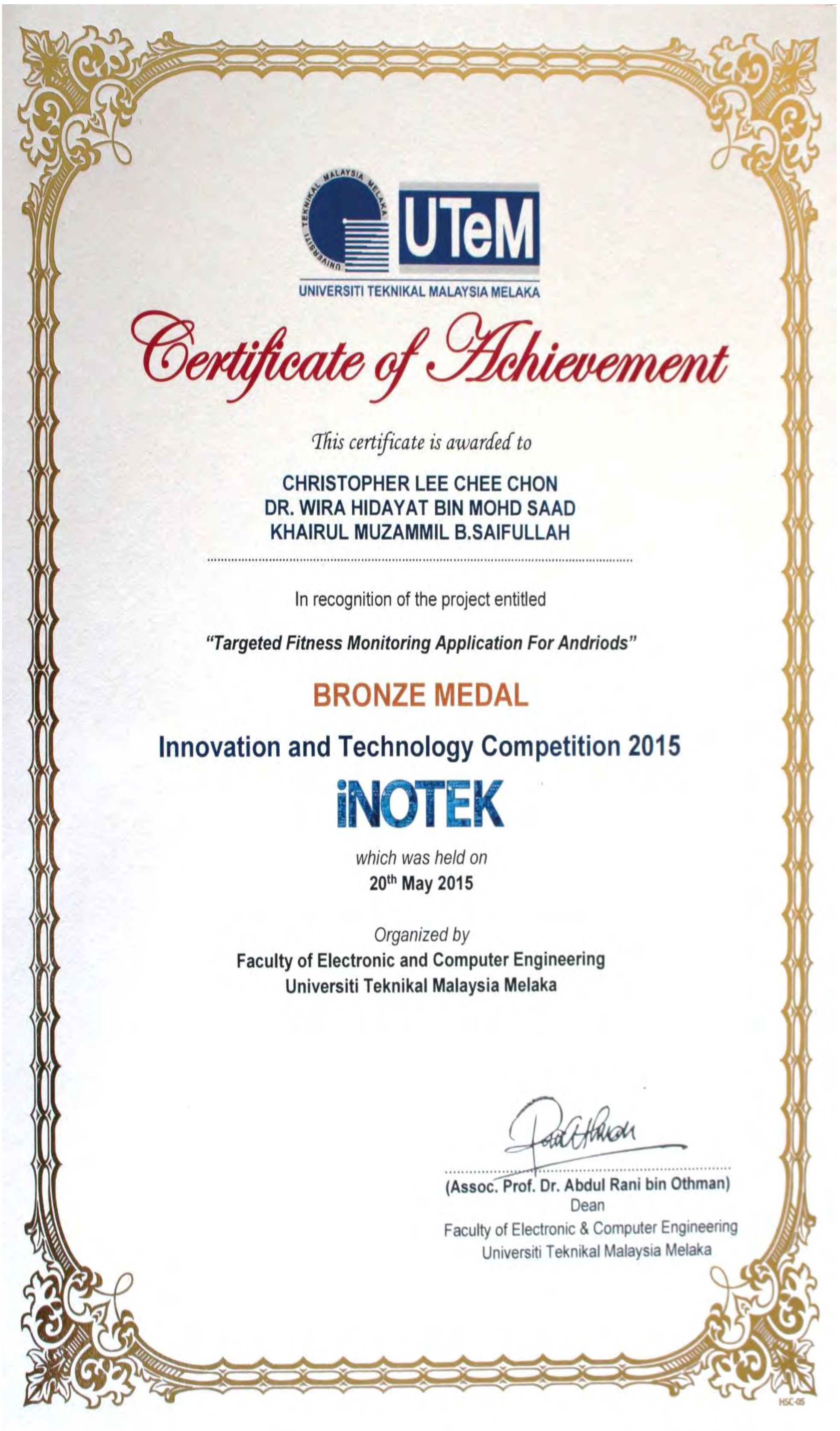
DEVELOPER INFORMATION

CHRISTOPHER LEE CHEE CHON (B021110092)
Bachelor of Electronic Engineering(Computer Engineering) with Honours
Contact Person: b021110092@utem.edu.my
SUPERVISOR: DR.WIRA HIDAYAT BIN MOHD SAAD

DR.WIRA HIDAYAT BIN MOHD SAAD
FACULTY OF ELECTRONICS AND COMPUTER ENGINEERING
Contact Person: wira_yugi@utem.edu.my

ANDROID

APPENDIX G



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Certificate of Achievement

This certificate is awarded to

CHRISTOPHER LEE CHEE CHON
DR. WIRA HIDAYAT BIN MOHD SAAD
KHAIRUL MUZAMMIL B.SAIFULLAH

In recognition of the project entitled

"Targeted Fitness Monitoring Application For Andriods"

BRONZE MEDAL

Innovation and Technology Competition 2015

iNOTEK

which was held on

20th May 2015

Organized by

Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka

[illegible]

(Assoc. Prof. Dr. Abdul Rani bin Othman)
Dean
Faculty of Electronic & Computer Engineering
Universiti Teknikal Malaysia Melaka

HSC-05