SENSATION SMART WATCH

Circuit Design, Sensors Software, Server and iOS App

FYP Final Report

Project I.D.: Design of Wearable Sensors

Project Number: YJ1-16

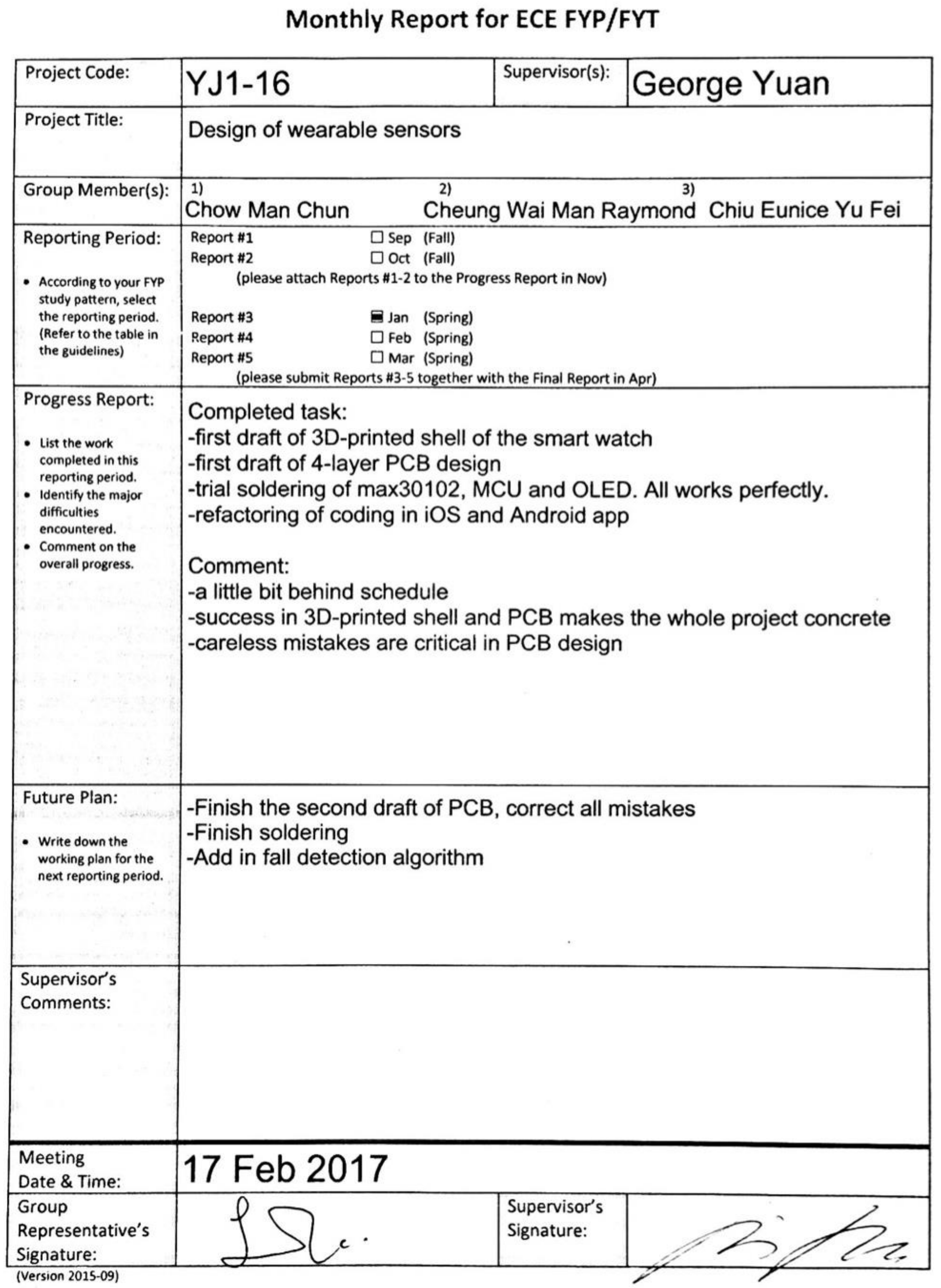
Project Supervisor: George Yuan

Signature of Project Supervisor:

Name of Author: CHOW Man Chun

Date: April 18, 2017

Monthly Report #3

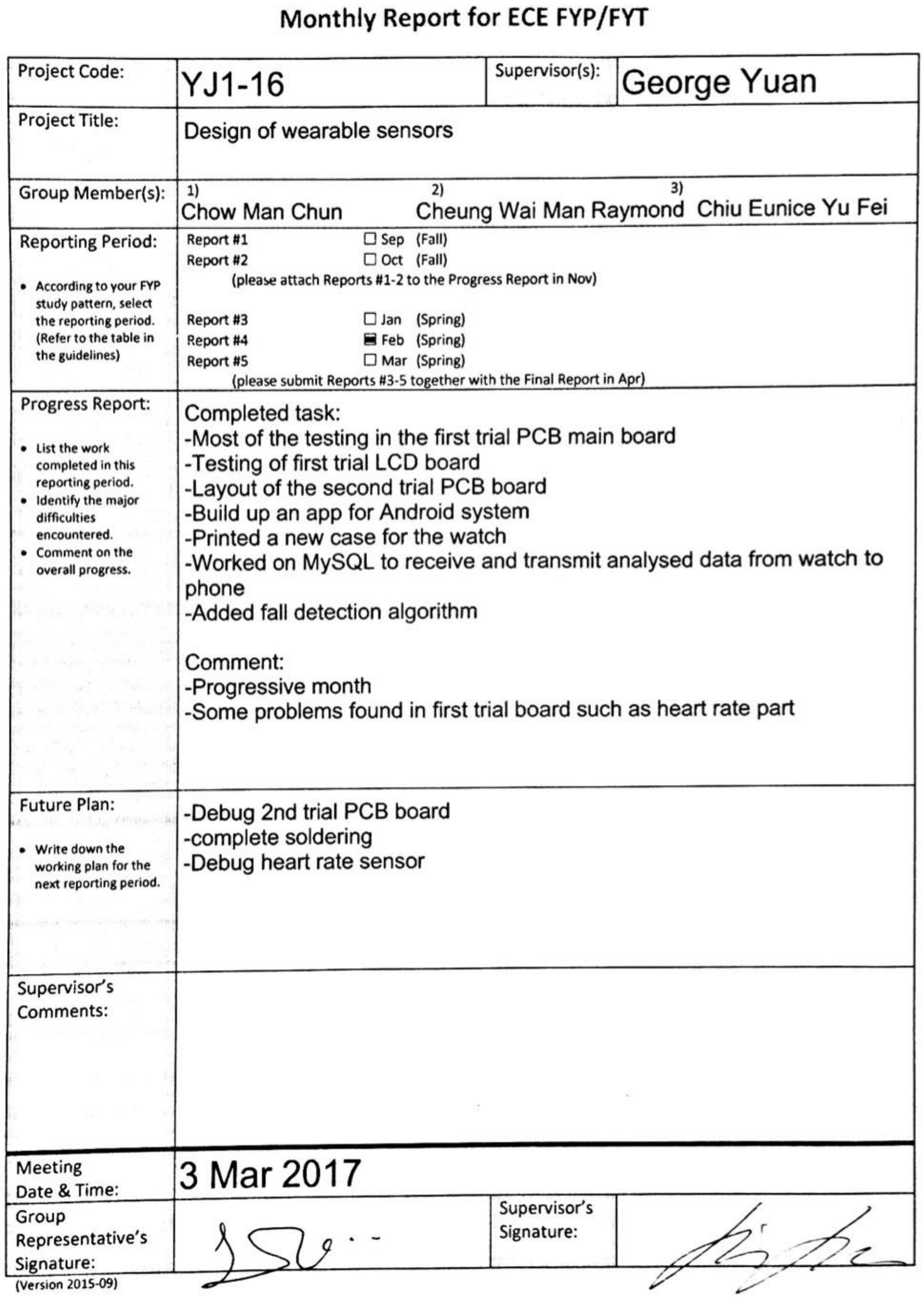


Monthly Report for ECE FYP/FYT

Project Code:	YJ1-16	Supervisor(s):	George Yuan
Project Title:	Design of wearable sensors		
Group Member(s):	1) Chow Man Chun	2) Cheung Wai Man Raymond	3) Chiu Eunice Yu Fei
Reporting Period: • According to your FYP study pattern, select the reporting period. (Refer to the table in the guidelines)	Report #1 □ Sep (Fall) Report #2 □ Oct (Fall) (please attach Reports #1-2 to the Progress Report in Nov) Report #3 ■ Jan (Spring) Report #4 □ Feb (Spring) Report #5 □ Mar (Spring) (please submit Reports #3-5 together with the Final Report in Apr)		
Progress Report: • List the work completed in this reporting period. • Identify the major difficulties encountered. • Comment on the overall progress.	Completed task: -first draft of 3D-printed shell of the smart watch -first draft of 4-layer PCB design -trial soldering of max30102, MCU and OLED. All works perfectly. -refactoring of coding in iOS and Android app Comment: -a little bit behind schedule -success in 3D-printed shell and PCB makes the whole project concrete -careless mistakes are critical in PCB design		
Future Plan: • Write down the working plan for the next reporting period.	-Finish the second draft of PCB, correct all mistakes -Finish soldering -Add in fall detection algorithm		
Supervisor's Comments:			
Meeting Date & Time:	17 Feb 2017		
Group Representative's Signature:		Supervisor's Signature:	

(Version 2015-09)

Monthly Report #4

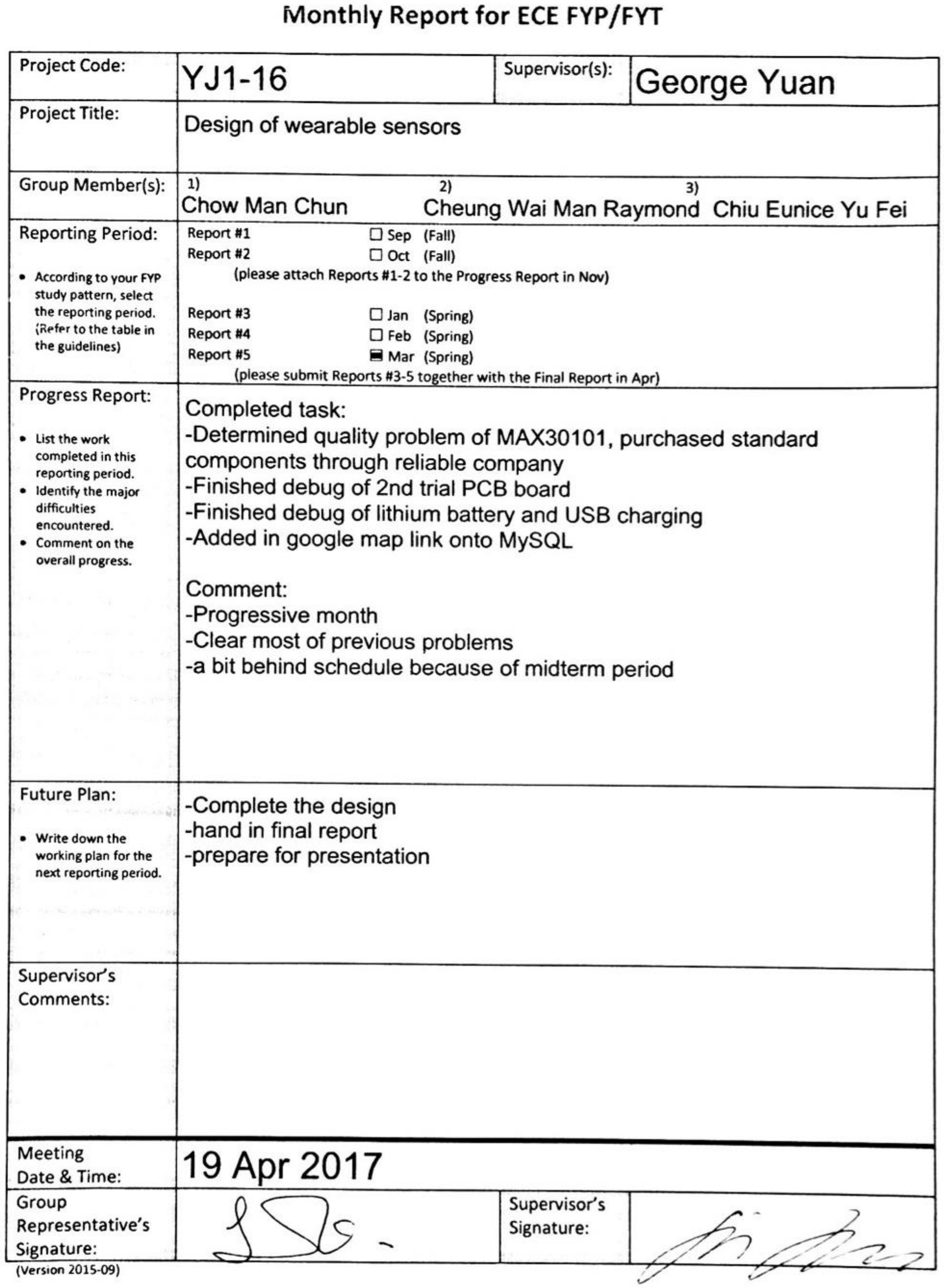


Monthly Report for ECE FYP/FYT

Project Code:	YJ1-16	Supervisor(s):	George Yuan
Project Title:	Design of wearable sensors		
Group Member(s):	1) Chow Man Chun	2) Cheung Wai Man Raymond	3) Chiu Eunice Yu Fei
Reporting Period: • According to your FYP study pattern, select the reporting period. (Refer to the table in the guidelines)	Report #1 □ Sep (Fall) Report #2 □ Oct (Fall) (please attach Reports #1-2 to the Progress Report in Nov) Report #3 □ Jan (Spring) Report #4 ■ Feb (Spring) Report #5 □ Mar (Spring) (please submit Reports #3-5 together with the Final Report in Apr)		
Progress Report: • List the work completed in this reporting period. • Identify the major difficulties encountered. • Comment on the overall progress.	Completed task: -Most of the testing in the first trial PCB main board -Testing of first trial LCD board -Layout of the second trial PCB board -Build up an app for Android system -Printed a new case for the watch -Worked on MySQL to receive and transmit analysed data from watch to phone -Added fall detection algorithm Comment: -Progressive month -Some problems found in first trial board such as heart rate part		
Future Plan: • Write down the working plan for the next reporting period.	-Debug 2nd trial PCB board -complete soldering -Debug heart rate sensor		
Supervisor's Comments:			
Meeting Date & Time:	3 Mar 2017		
Group Representative's Signature:		Supervisor's Signature:	

(Version 2015-09)

Monthly Report #5



Monthly Report for ECE FYP/FYT

Project Code:	YJ1-16	Supervisor(s):	George Yuan
Project Title:	Design of wearable sensors		
Group Member(s):	1) Chow Man Chun	2) Cheung Wai Man Raymond	3) Chiu Eunice Yu Fei
Reporting Period: • According to your FYP study pattern, select the reporting period. (Refer to the table in the guidelines)	Report #1 ☐ Sep (Fall) Report #2 ☐ Oct (Fall) (please attach Reports #1-2 to the Progress Report in Nov) Report #3 ☐ Jan (Spring) Report #4 ☐ Feb (Spring) Report #5 ☒ Mar (Spring) (please submit Reports #3-5 together with the Final Report in Apr)		
Progress Report: • List the work completed in this reporting period. • Identify the major difficulties encountered. • Comment on the overall progress.	Completed task: -Determined quality problem of MAX30101, purchased standard components through reliable company -Finished debug of 2nd trial PCB board -Finished debug of lithium battery and USB charging -Added in google map link onto MySQL Comment: -Progressive month -Clear most of previous problems -a bit behind schedule because of midterm period		
Future Plan: • Write down the working plan for the next reporting period.	-Complete the design -hand in final report -prepare for presentation		
Supervisor's Comments:			
Meeting Date & Time:	19 Apr 2017		
Group Representative's Signature:		Supervisor's Signature:	

(Version 2015-09)

Acknowledgement

I would like to express my sincere gratitude to Dr. George Yuan for supervising me and providing me a valuable opportunity to do this project. Dr. Yuan provided many useful resources and supports such as his own laboratory and funding. His advice to the project are also extremely beneficial to the project.

The completion of this project could not have been possible without the effort and co-operation of all team Sensation's members, including Raymond Cheung Wai Man and Eunice Yu Fei Chiu.

I would also like to express my appreciations to Mr. Chiye Zhang who teaches my group members how to use the BGA Rework Station and helped us to find PCB fabricators in China.

A debt of gratitude is also owed to Mr. Anthony Ka Shing Yuen, an experienced electronic engineer and an alumnus of HKUST, for his professional advice and assistance on creating PCB Layouts.

Abstract

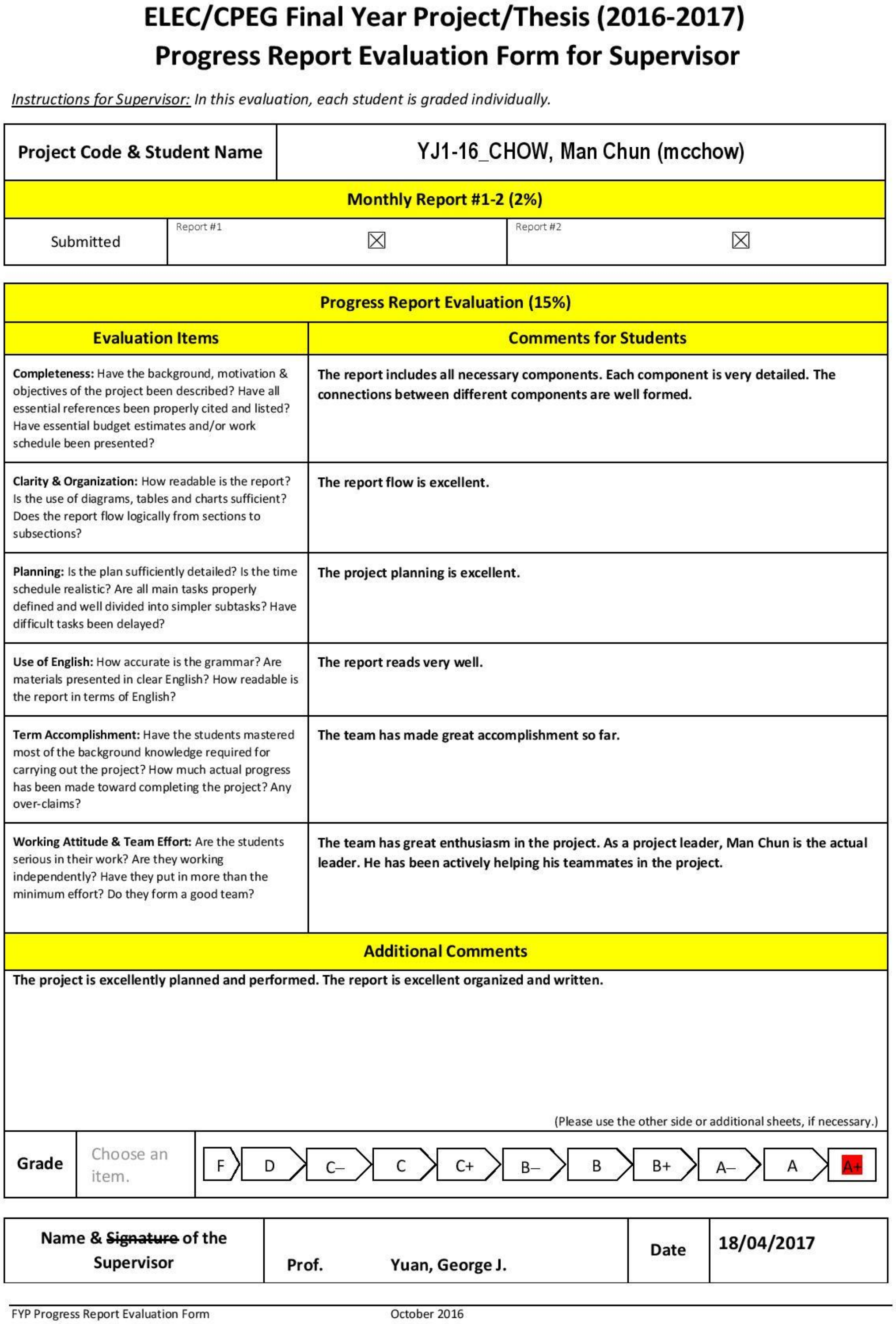
The objective of this project is to create a new wearable sensors system suitable for continuous health monitoring such as elderly monitoring. It records vital signs such as heart rate, step counts and fall detection of the user. In case of emergency, it can send a message to the central server indicating the location of the user, such that the user can be found if he/she is unconscious.

The motivation of creating this system is to explore the market of cost-effective health monitoring. In fact, most of the activity trackers in the market did not use the data collected to measure body movements other than step counts. Although some smartwatches provide more features, they are generally more expensive.

Our system consists of a smartwatch-like wearable sensor, a smartphone and a central server. The watch will connect with the smartphone, while the smartphone will connect with the central server. Both Android and iOS are supported.

The result of this project has been successful: the whole ecosystem including a web interface, two mobile applications, a watch shell and a watch circuit board are developed and tested.

Evaluation Form



ELEC/CPEG Final Year Project/Thesis (2016-2017)
Progress Report Evaluation Form for Supervisor

Instructions for Supervisor: In this evaluation, each student is graded individually.

Project Code & Student Name	YJ1-16_CHOW, Man Chun (mcchow)				
Monthly Report #1-2 (2%)					
Submitted	Report #1	☒	Report #2	☒	

Progress Report Evaluation (15%)	
Evaluation Items	**Comments for Students**
Completeness: Have the background, motivation & objectives of the project been described? Have all essential references been properly cited and listed? Have essential budget estimates and/or work schedule been presented?	**The report includes all necessary components. Each component is very detailed. The connections between different components are well formed.**
Clarity & Organization: How readable is the report? Is the use of diagrams, tables and charts sufficient? Does the report flow logically from sections to subsections?	**The report flow is excellent.**
Planning: Is the plan sufficiently detailed? Is the time schedule realistic? Are all main tasks properly defined and well divided into simpler subtasks? Have difficult tasks been delayed?	**The project planning is excellent.**
Use of English: How accurate is the grammar? Are materials presented in clear English? How readable is the report in terms of English?	**The report reads very well.**
Term Accomplishment: Have the students mastered most of the background knowledge required for carrying out the project? How much actual progress has been made toward completing the project? Any over-claims?	**The team has made great accomplishment so far.**
Working Attitude & Team Effort: Are the students serious in their work? Are they working independently? Have they put in more than the minimum effort? Do they form a good team?	**The team has great enthusiasm in the project. As a project leader, Man Chun is the actual leader. He has been actively helping his teammates in the project.**

Additional Comments

The project is excellently planned and performed. The report is excellent organized and written.

(Please use the other side or additional sheets, if necessary.)

Grade	Choose an item.	F D C– C C+ B– B B+ A– A [✓ A+]

Name & ~~Signature~~ of the Supervisor	**Prof. Yuan, George J.**	**Date**	**18/04/2017**

FYP Progress Report Evaluation Form October 2016

Table of Contents

List of Illustrations

List of figures

List of Tables:

Chapter 1 – Introduction

Section 1.1: Project Objectives and Introduction

1.1.1 Introduction

In the past few decades, bio-medical sensors have usually been cumbersome and could only be operated by professional physicians. However, with the rapid development of bio-sensing technology and micro-electro-mechanical (MEMS) sensors, tiny sensors measuring vital signs such as heart rate, respiratory rate and daily step counts of human have been popularized. This technological advancement has been enabled the realization of wearable sensors systems that are compact, inexpensive and user-friendly. Therefore, sensors monitoring health status nowadays can easily be worn by anybody without prior medical knowledge.

In fact, many people are interested in utilizing wearable sensors for different purposes. For example, clinicians are interested in using wearable sensors to monitor their patients in different environments over a long period [1]. Other general users are also interested in using wearable sensors as personal fitness trainers. It is predicted that the market in activity-sensing will be worth about $975 million by 2017 [2]. By exploring new usages of sensors and their measurement data, it is believed that the needs of wearable sensors will increase steadily.

While wearable sensors continue to gain popularity, their prolific use is currently hampered by the usage of data and the cost of the devices. Low-price activity trackers in the market such as Xiaomi Mi-Band could only provide basic functions, while the multi-function, programmable activity trackers like Apple Watch or Microsoft Band are usually expensive because of the costs in combining multiple sensors components and the general-purpose operating systems.

1.1.2 Literature Review

A review of the current technology in wearable sensors highlights the need for a comprehensive, user-friendly but low cost wearable sensor system for healthcare. A wider market would be captured ranging from not only the fitness and weight conscious, but to those in need of constant health monitoring, including the ill and elderly.

Component Investigation of Mi-Band

Mi-Band is a low-cost activity tracking wristband produced by Xiaomi in 2014 [19]. Mi-Band monitors user's daily step-counts continuously. It also provides alarms and notification reminders by vibrating the wristband. By investigating major components used in Mi-Band, the basic structure of a wearable sensor can be understood. According to ARM Connected Community [4], Mi-Band consists of only 6 major components, which are Bluetooth Processor, DC-DC converter, 3-axis accelerometer, flash memory, battery charger and LED driver. Using fewer components, the complexity of this system has been notably minimized. It could be beneficial since it helps Mi-Band to be very light (5.0 gram) and battery-saving (30 days) [5]. However, it also brings an issue of lacking multiple types of sensor data, as there is only one sensor, the accelerometer, equipped in the system.

Current Technology in Heart Rate Sensing

For heart rate sensing, there are two major non-invasive methods to record users' heartbeat, namely PPG (Photo-plethysmography) and ECG (Electrocardiography). PPG is a sensing technology that uses a single optical sensor with a near-infrared emitter and a detector to measure the changes in blood vessels. PPG sensors are usually comfortable to wear on forefinger [6]. These sensors are currently produced and supported by many hardware manufacturers such as Texas Instruments [7], Maxim Integrated [8], Silicon Labs [9] and more.

Therefore, it is a considerably mature sensing technology in market. ECG is another sensing technology that uses at least 3 electrodes to attach on body to measure electrical signals generated by human body [6]. Experiments showed that the heartbeat measurement in ECG is generally more accurate than in PPG because the motion artefacts generated in PPG sensing are difficult to remove [6]. However, as PPG sensors are relatively more comfortable to wear and easier to be found in market, PPG sensors are more feasible for developing low-cost wearable sensors products that is easy and comfortable to wear.

Current Technology in Body Movement Sensing

For body movement sensing, an accelerometer is usually chosen because of its popularity. An accelerometer is a widely-used sensor in in smartphones to measure linear acceleration of the device along multiple axis [10]. Mi-Band has equipped with it for counting users' footsteps. In fact, there are still other kinds of data can be extracted with accelerometer: with appropriate software algorithms, accelerometers can also facilitate the detections of many body movements such as falling. According to [11], since there are observable unique patterns for different body movements, fall events can be reliably detected with accelerometers. However, current wearable sensors systems equipped with accelerometer and sophisticated algorithms record only step counts and not falling and other body movements of the users, which are useful features for more comprehensive monitoring and for wider user market such as elderly healthcare.

Investigation of a Programmable Wearable Sensor: Apple Watch

Apple Watch is an activity tracking smartwatch produced by Apple in 2015 [20]. It is equipped with S1 System-in-Package technology that contains an accelerometer, a gyroscope and a heart-rate sensor [21]. Its proprietary operating system, watchOS, has provided many libraries such as CoreMotion [22] for software developers to use all sensors data to build their health monitoring applications. Therefore, developers are able use Apple Watch to detect different kinds of movement of the user such as falling and so on. However, the downsides of Apple Watch are the short battery life (the stand-by time around only 22 hours) [21] and its unfavorable high price [23], which make Apple Watch to be less suitable for continuous monitoring and cost-effective wearable sensing.

To sum up, two wearable sensors systems, the **Xiaomi Mi-Band** and **Apple Watch**, are analyzed. Mi-Band is a wearable sensors system that has only 6 major components. It can only provide information of step-counting. Its simplicity in hardware design has made the device cheap and power-conservative. However, the function of Mi-Band is quite limited because of the lack of multiple sensors. Apple Watch is the opposite of Mi-Band, it combines three types of sensors and allows software developers to create their own algorithms to detect different movements. However, it is far more expensive than Mi-Band but its battery life is much shorter.

Besides, two kinds of sensing technology, **heart-rate sensing** and **body movement sensing**, are also analyzed. The technological advancement in PPG (Photo-plethysmography) sensors has made heart-rate sensing become convenient and comfortable. PPG sensors are also increasingly accessible because of the active support of manufacturers. When it comes to body movement detection, enhanced software algorithm has also enabled the detection of different body movements with existing accelerometers. Thus, the sensing technology nowadays is ready to for us to develop a new wearable sensors system that could monitor users' vital signs with non-invasive means.

1.1.3 Group Project Objectives

In this paper, we develop a new wearable sensors system monitoring user's vital signs continuously and transmitting health data to smartphone using Bluetooth. Our new wearable sensors system, the "Sensation Smart Watch" is a cost-effective sensing systems intended for users that needs continuous monitoring such as elderly. It combines a heart rate sensor and an accelerometer to collect vital signs data including step counts, heart rate and fall event.

Smartphone application for both iOS and Android will use Bluetooth to connect with the device. It allows users synchronize device time, visualize health data and report fall location to a central server immediately when the watch detects that the user has fallen. Therefore, our wearable sensors system will be not only a simple activity tracker, but also a safeguard to detect if the user has encountered any accidents like fainting.

To make our wearable sensor device be more user-friendly, users can check the time and read their health information directly with an OLED screen and a button on the device. Therefore, users can also treat our device as a smart watch.

1.1.4 Individual Sub-Project Objectives

In this project, I will mainly focus on logic design, hardware design and project management.

Logic designs refers the software and hardware logic inside the Sensation Smart Watch and the smartphone application. First, I am responsible for choosing and purchasing components that can implements features mentioned in our objectives. Then, I need to combine these components and create software to make them work together. Therefore, I am also responsible for creating driver code for all sensors and creating the schematic that puts all components together systematically. I also need to write codes to allow the watch to interact with users with a button or Bluetooth. Therefore, I am also responsible for programming the Bluetooth Stack in smartwatch and the backend code related to Bluetooth in both Android and iOS apps.

Hardware designs refers to creating a customized PCB (printed circuit board) that implements our schematic. I am responsible for making use of the components libraries (which are created by Eunice) to put all components and wirings into the limited area space (which is defined by Raymond). Then, I need to contact with PCB fabricators to create the PCB. Finally, I also need to test all hardware connections of the customized PCB board to make sure no error has been created during the fabrication process.

Section 1.2: Project Description and Job Distribution

1.2.1 Group Project Description

The workflow of the system

Sensation Smart Watch is wearable sensors solution integrated with a smartwatch, a smartphone application and a cloud server. Our watch measures step counts, heart rate and falling of the user continuously with low power consumption. All measured data will be stored inside the watch and transmitted to the smartphone using Bluetooth with our smartphone app, so user can view their health status on the both devices. If the watch detects the user has fallen, a notification will immediately be sent to the smartphone. After that, the smartphone will send the falling time and falling location to a cloud server. Finally, the server will update automatically and be able to display all real-time fall records.

Technical Requirements of the system

To make the system monitors the user continuously, the estimated standby time (with mixed use) of the watch should be longer than one day (24 hours). Besides, to make the real-time fall event monitoring server, the delay time of the fall time and the fall location reported by the system should be accurate without much delays (at most 1 minute delay). The watch should also be as compact as possible such that most people can wear the watch comfortably.

Block diagram of the system

For the smartwatch, we will be using a Bluetooth Processor (MCU) to connect with an accelerometer, a heart-rate sensor, an OLED screen and a button. The watch will be powered by a lithium-ion battery that can be charged when USB is attached to the device. An Android App and an iOS App will also be developed for transmitting and visualizing the data collected from the smart watch. For the server, we will be using mainly PHP and MySQL to get and store all fall records. HTTP will also be used for receiving all data sent by smartphone apps.

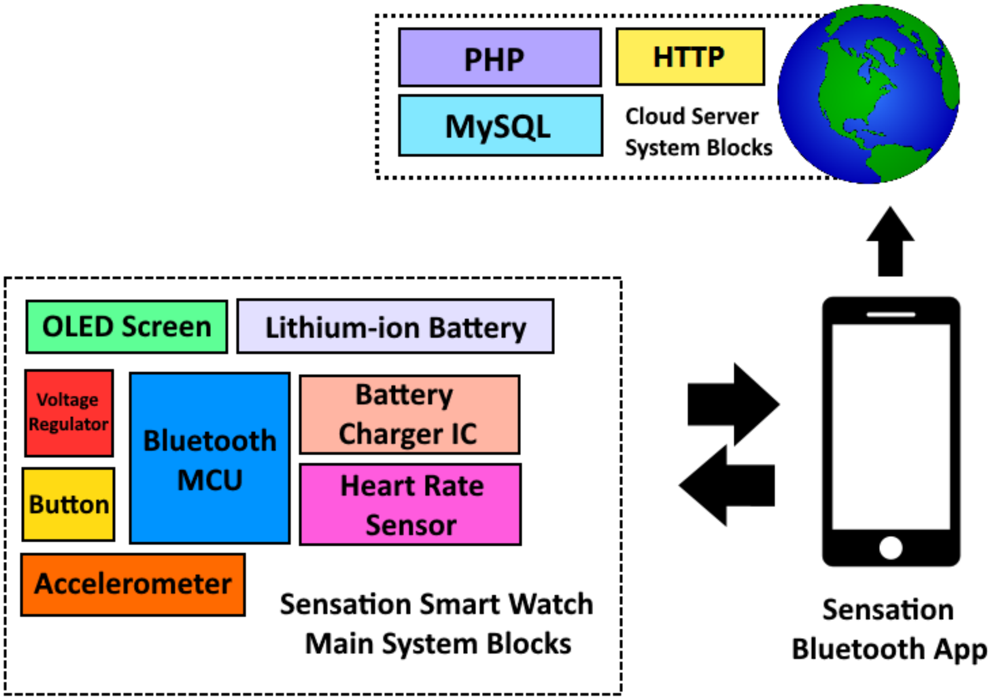


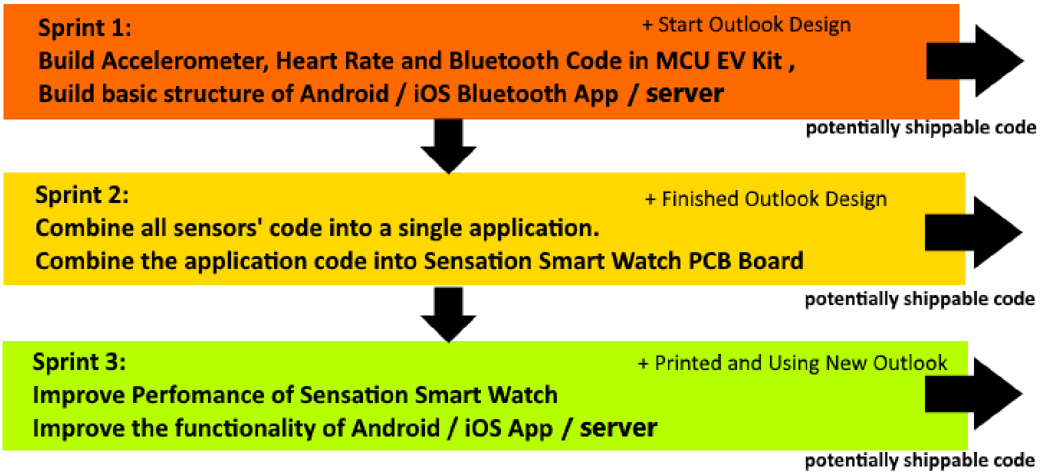
Figure 1 - System Block Diagram of Sensation Smart Watch

1.2.2 Schedule of Responsibilities

Project Structure and Gantt Chart

This project is using “Scrum” Project Management. Scrum is a management framework for incremental product development, in each fixed length iterations (called “Sprint”), Scrum team attempts to build a potentially shippable product increment [12].

Our project involves 3 Sprints. The objectives of each sprint are shown below:



Sprint 1: + Start Outlook Design
Build Accelerometer, Heart Rate and Bluetooth Code in MCU EV Kit ,
Build basic structure of Android / iOS Bluetooth App / server
potentially shippable code

Sprint 2: + Finished Outlook Design
Combine all sensors' code into a single application.
Combine the application code into Sensation Smart Watch PCB Board
potentially shippable code

Sprint 3: + Printed and Using New Outlook
Improve Perfomance of Sensation Smart Watch
Improve the functionality of Android / iOS App / server
potentially shippable code

Figure 2 - Sprints of our Project

The Gantt Chart of the project is shown below:

Task	Sprint 1				Sprint 2				Sprint 3			
Week Number	1	3	5	7	1	3	5	7	1	3	5	7
Choosing hardware platform	■											
Purchase of components	■	■										
Pedometer Programming			■	■								
Testing Pedometer Performance									■	■		
Fall Detection code										■	■	
Heart Rate Programming			■	■					■	■		
Testing Heart Rate Performance										■	■	■
User Interface Programming (watch)			■	■	■	■	■					
Bluetooth Programming (watch)			■	■								
iOS and Android Backend Programming	■	■							■	■		
iOS and Android Frontend Programming										■	■	
iOS and Android Testing											■	■
Server Backend Programing			■	■					■			
Server Frontend Programming				■						■		
Server Testing										■	■	
Schematic Drawing			■	■	■							
PCB Layout Drawing						■	■					
Printing PCB							■	■				
Outer shell Drawing					■	■	■					
Outer shell Printing							■	■				
Soldering and Assembling								■	■			
Writing Final Report											■	■

Figure 3 - Gantt Chart of our Project

Job Division and Responsibilities

This project has been divided into 7 sub-categories and several sub-projects to provide a clear job-distribution among group members. The sub-projects include:

Table 1 - Job Division Diagram

Sub-Project	**CHOW Man Chun**	CHEUNG Wai Man Raymond	CHIU Eunice Yu Fei
Category A: Programming of Sensation Smart Watch (Embedded System Code)			
Accelerometer Related Code	Responsible		
Heart Rate Related Code	Assist	Responsible	
Buttons and User Interface Code	Responsible		
Bluetooth Stack Code	Responsible	Assist	
Category B: Programming of Smartphone Application (Java/Swift Code)			
Android Backend Programming	Assist	Responsible	
Android Frontend Design		Responsible	
iOS Backend Programming	Responsible		
iOS Frontend Design	Responsible		
Category C: Programming of Server Side Application (PHP/MySQL/HTML/HTTP Code)			
Fall Server Backend Programming	Responsible	Assist	
Fall Server Frontend Programming	Assist	Responsible	
Category D: Selection and Purchase of Components			
Find components available in market	Responsible	Responsible	Assist
Buy components and evaluation kits	Responsible	Assist	Assist
Handle Budget Reimbursement	Assist		Responsible
Category E: Design of Hardware Platform			
Schematic Design	Responsible		Assist
Schematic Library Drawing	Assist		Responsible
PCB Layout Design	Responsible		Responsible
PCB Library Drawing			Responsible
PCB Gerber Generation and Printing	Assist		Responsible
Soldering and Rework	Assist		Responsible
Category F: Outlook Design			
Design the outer shell of the watch		Responsible	
Print the shell using 3D Printer		Responsible	
Choosing Wristband	Assist	Responsible	
Purchase of Wristband and materials	Assist	Responsible	
Category G: Paperwork and Project Management Related			
Drafting Monthly Report		Assist	Responsible
Managing Scrum board	Responsible	Assist	Assist

Explanation of the table

- "Responsible" refers to
  "most of the work of this sub-project will be done by that person".
- "Assist" refers to "that person is involved in this sub-project,
  but he/she should not be responsible for the completion of the sub-project".

Section 1.3: Individual Sub-Project Details

I am involved in 4 sub-projects. They are circuit design, sensors software programming, server programming and iOS application programming.

1.3.1 Sub-Project 1 – Circuit Design

1.3.1.1 Sub-Project Description

Circuit Design sub-project is to decide what components to be included in the watch, create a schematic diagram to show how the components are connected, and create a customized PCB that is compact enough to fit inside the 3D-printed smart watch case.

By the end of this sub-project, a schematic diagram, a PCB Layout diagram and a customized PCB will be created. The circuit should provide a low noise environment such that noise-sensitive components such as sensors and the main processor can work in different voltages and different power source.

1.3.1.2 Components

1.3.1.2.1 Hardware List

The following hardware components are used in this sub-project:

Table 2 - Components List of Sensation Smart Watch Circuit

Component ID	Description and Technical Specifications	Count
JDY-08	Texas Instrument CC2541 MCU Module • PCB Antenna Attached • 32.768K and 32M Crystal Included	1
ADXL362	Analog Device ADXL362 Accelerometer and Gyroscope • Ultra-Low Power Accelerometer	1
MAX30101	Maxim MAX30101 Heart-Rate Sensor • Operating Voltage is 1.8V • 3 LEDs with IR/Red/Green light embedded	1
LTC4054L	Linear Technology LTC4054L Li-ion Battery Charger IC • Automatic CC/CV control • Overcharge protection	1
XC6206P332MR	Torex XC6206 LDO Power Regulator • Input Voltage: 3.3V to 6.0V • Output Voltage: 3.3V (± 2% accuracy)	3
XC6026P182MR	Torex XC6026 LDO Power Regulator • Input Voltage: 1.8V to 6.0V • Output Voltage: 1.8V (± 2% accuracy)	1
NCP1402-5V	ON Semiconductor NCP1402 Step-up Regulator • Input Voltage: 2.0V to 6.0V • Output Voltage: 5.0V (± 2.5% accuracy)	1
VGM128064C0W01	OLED Screen • 0.96 inch, 128*64 pixels • Solomon SSD1306 Controller	1
SS24	Schottky Diode • Low Forward Voltage (~0.4V in 3V)	1
SS14	Schottky Diode • Use with NCP1402-5V	1

1.3.1.2.2 Software List

The following software is used in this sub-project:

Table 3 - Software List of Sensation Smart Watch Circuit

Software Name	Description	Version
Altium Designer	Schematic Design and PCB Layout Design Tool	16.1.12

1.3.1.3 System Block Diagrams

The block diagrams of the sub-project procedure are shown below:

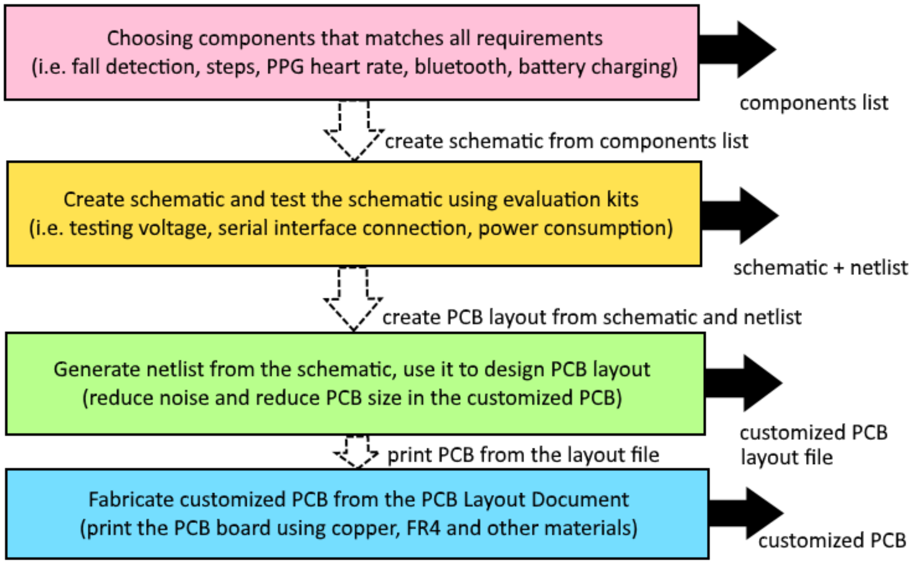

Choosing components that matches all requirements
(i.e. fall detection, steps, PPG heart rate, bluetooth, battery charging)
components list
create schematic from components list
Create schematic and test the schematic using evaluation kits
(i.e. testing voltage, serial interface connection, power consumption)
schematic + netlist
create PCB layout from schematic and netlist
Generate netlist from the schematic, use it to design PCB layout
(reduce noise and reduce PCB size in the customized PCB)
customized PCB
layout file
print PCB from the layout file
Fabricate customized PCB from the PCB Layout Document
(print the PCB board using copper, FR4 and other materials)
customized PCB

Figure 4 - Design Flow of the Circuit

The system block diagram of the watch circuit designed in this sub-project is shown below:

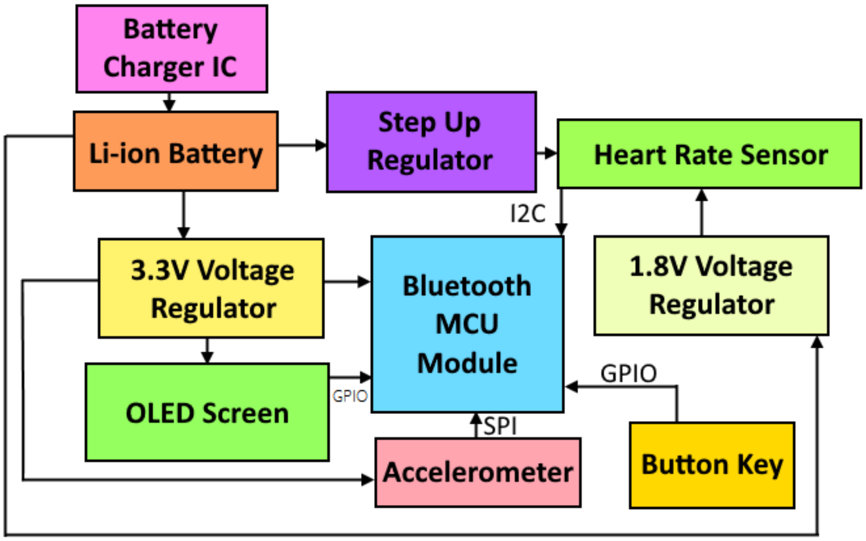


Figure 5 - System Block Diagram of Sensation Smart Watch Circuit

1.3.1.4 Sub-Project Tasks

These are tasks related to this sub-project:

1. I/O Testing on accelerometer (ADXL362)
2. I/O Testing on heart rate monitor (MAX30101)
3. Voltage Testing on regulators (XC6206)
4. Voltage Testing on step-up regulator (NCP1402-5V)
5. Characteristic Testing on Charger IC (LTC4054L)
6. I/O Testing on OLED (VGM128064C0W01)
7. Drawing Schematic Diagram
8. Drawing Optimized PCB Layout
9. PCB Rules and ERC Checking
10. Contacting Fabrication Plants

1.3.1.5 Technical Challenges

There are some expected technical challenges in this sub-project, which includes:

- ADXL362 and MAX30101 are noise-sensitive sensors. To reduce the fluctuation of their readings, the PCB design should try to reduce noises from power line. Also, the operating voltage of these two sensors should be as stable as possible to avoid improper unconditional reset of the sensors.
- MAX30101 requires 1.8V operating voltage and 5V LEDs voltage. However, all other components in the schematic are using 3.3V. That means we would need extra space to put both 1.8V, 3.3V and 5V regulators. It would increase the size of the PCB board.
- JDY-08 (the MCU CC2541) is using PCB Antenna. We need to leave a copper-free window in our customized PCB to avoid changing the radiation pattern of the 2.4GHz Bluetooth signal.

1.3.1.6 Budget

The budget of this sub-project is listed below:

Table 4 - Budget Used in Creation and Fabrication of Sensation Smart Watch Circuit

Item	Item Price	Qty.	Total Price
Fabrication Fee – First PCB Prototype (2 layers)	RMB 40.00	1	RMB 40.00
Fabrication Fee – Second PCB Prototype (4 layers)	RMB 140.00	1	RMB 140.00
Shipping Charge from Shenzhen to Hong Kong	RMB 30.00	2	RMB 60.00
Component - JDY-08	RMB 12.00	1	RMB 12.00
Component - ADXL362	RMB 15.00	1	RMB 15.00
Component - MAX30101	RMB 32.00	1	RMB 32.00
Component - LTC4054L	RMB 2.00	1	RMB 2.00
Component - XC6206P332MR	RMB 0.20	3	RMB 0.60
Component - XC6026P182MR	RMB 0.20	1	RMB 0.20
Component - NCP1402-5V	RMB 1.00	1	RMB 1.00
Component - VGM128064C0W01	RMB 24.00	1	RMB 24.00
Component - SS24	RMB 0.01	1	RMB 0.01
Component - SS14	RMB 0.01	1	RMB 0.01
		Total Price	RMB 326.82

1.3.2 Sub-Project 2 – Sensors Software Programming

1.3.2.1 Sub-Project Description

Sensors software programming is to write codes to make all sensors including ADXL362 and MAX30101 to communicate with the Bluetooth processor CC2541. In this sub-project, I am mainly focusing on creating driver code of ADXL362, fall detection algorithm and pedometer (step counting) algorithm. Also, I am also focusing on creating basic UI (user interface) in the smart watch.

At the end of this sub-project, user can read their step counts using the OLED screen on the smartwatch. Also, user can choose to show different data including heart rate number, date and time, or even turn off the screen.

1.3.2.2 Components

1.3.2.2.1 Hardware List

The following hardware components are used in this sub-project:

Table 5 - Hardware used in Programming Sensors Software

Component ID	Description and Technical Specifications	Count
CC254xEK	Evaluation Board for CC254X Module It must be used with CC254xEM Evaluation Module	1
CC2541EM	CC2541 Evaluation Module • CC2541 MCU • PCB Antenna Attached • 32.768K + 32M Crystal	1
GY-ADXL362	Accelerometer Module • Analog Device ADXL362	1
VGM128064C0W01	OLED Screen • 0.96 inch, 128*64 pixels • Solomon SSD1306 Controller	1
CC-DEBUGGER	CC Debugger • Flashing firmware of CC2541 • Provide line-by-line debugging	1

1.3.2.2.2 Software List

The following software is used in the sub-projects:

Table 6 - Software Used in Programming Sensors Software

Software Name	Description	Version
IAR Embedded Workbench for 8051	**8051 Compiler, IDE and C Library** It is used to program the firmware of CC2541 inside Sensation Smart Watch.	8.10.3
TI BLE Stack	**TI Proprietary Bluetooth Library and Examples** The firmware code of Sensation Smart Watch is modified from the examples provided here.	1.3.2
Zimo221	**Tools for translating bitmaps to C arrays** This is useful for translating bitmaps in .BMP to the C array that can be displayed in the watch.	V2.2

1.3.2.3 System Block Diagrams

The system block diagrams of the sub-project are shown below:
(Dotted-lines in these diagrams imply the data needs to be converted before sending it.)

<u>Accelerometer Sensor Software: Programming the pedometer and fall detection</u>

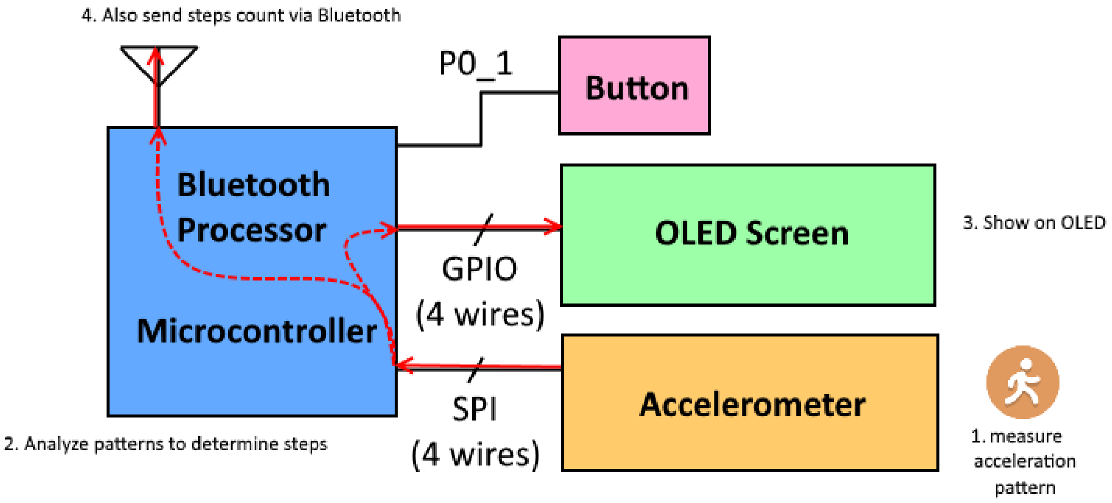


Figure 6 - System Block Diagram of Pedometer and Fall Sensing

<u>Wearable Sensors System Software: Programming the UI and buttons</u>

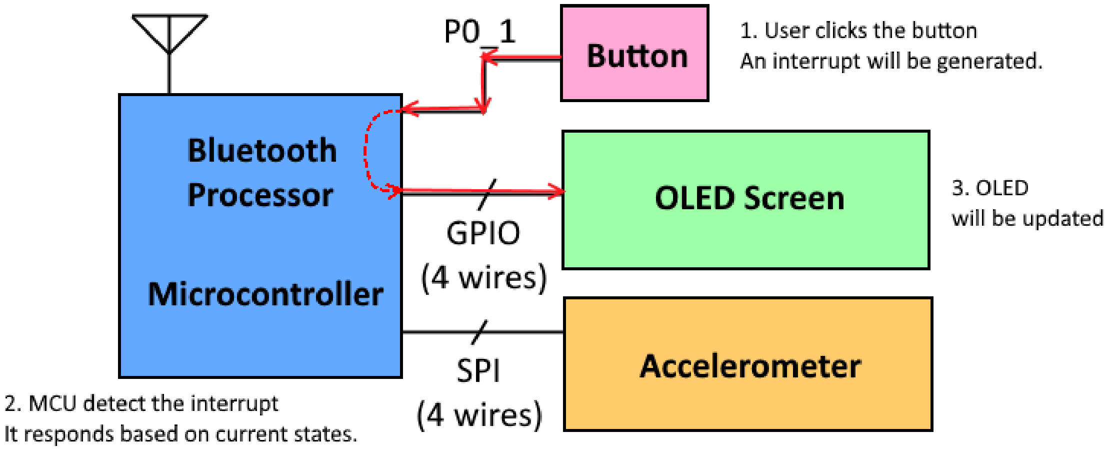

P0_1
Button
1. User clicks the button
An interrupt will be generated.
Bluetooth Processor
Microcontroller
OLED Screen
3. OLED will be updated
GPIO (4 wires)
Accelerometer
SPI (4 wires)
2. MCU detect the interrupt
It responds based on current states.

Figure 7- System Block Diagram of User Interface Interaction

1.3.2.4 Sub-Project Tasks

These are tasks related to this sub-project:

Logic of accelerometer (Pedometer and Fall-Detection):

1. Basic Testing of ADXL362
   a. Reading Raw Data from accelerometer
2. FIFO Read Accelerometer
3. FIFO Watermark Level Adjustment and Interrupt
4. Developing step-counting algorithm
5. Sending data to PC for further investigations
   a. Using CoreBluetooth Library in XCode
   b. Implementing a tiny code for connecting Sensation Smart Watch
   c. Receive data in the debug window of XCode
6. Developing noise-filtering/averaging algorithm
7. Developing detection of extreme cases (idle/shaking)
8. Developing Fall-Detection algorithm
9. Adjust fall detection trigger threshold

Basic User Interface Design in the watch:

1. Request Driver Code from Manufacturer
2. Creating hardware interrupt for a button
3. Using ISR to Draw simple patterns on OLED

After completing all tasks above, this sub-project can be finished.

1.3.2.5 Technical Challenges

There are some expected technical challenges in implementing sub-projects of the watch, which includes:

- CC2541 is a single core System on Chip microcontroller that Bluetooth RF tasks and usual tasks are running in the same place. Therefore, any improper use of while loop or code with long execution time will break the Bluetooth transmission. This problem is inevitable, but we can make the connection interval to be larger to allow the device handshake less frequent.

- The step count recognition (walking pattern recognition) code should be short and simple such that the Bluetooth RF remains connected during calculations. We have found that the `sqrt()` function used in pattern recognition takes long calculation time, therefore we separate this task into two tasks. This allows Bluetooth task to run before exceeding the connection interval.

- Most of the pins in the CC254xEK Evaluation Kit are connected to some components. Free pins are very limited, which makes using all interrupt pins of ADXL362 becomes impossible in Sprint 1. Customization of PCB board is required.

- There is no official SPI driver for CC2541 in BLE Stack v1.3.2. Extra effort has been spent on developing and debugging SPI driver.

1.3.2.6 Budget

The budget of the sub-projects is listed below:

Table 7- Budget Used in Programming Pedometer and User Interface only

Item	Qty.	Total Amount
CC254xEK Evaluation Kit	1	RMB 168.00
CC2541EM Evaluation Module (CC2541)	1	RMB 40.00
0.96-inch OLED Display Module (VGM128064C0W01)	1	RMB 32.00
CC Debugger	1	RMB 88.00
GY-ADXL362 Accelerometer Module (ADXL362)	1	RMB 18.00
SF-Express Shipping Charge	1	RMB 30.00
Shipping Charge in Shenzhen	1	RMB 9.00
Total		RMB 385.00

1.3.3 Sub-Project 3 – Server Programming

1.3.3.1 Sub-Project Description

To let smartphones to upload the fall time and location to a central server, we need to create a server application written by PHP and MySQL, which are the most common scripting and database language in web applications.

At the end of this sub-project, we should be able to create a central server program that collects and displays all fall records in a webpage. Also, it should only be accessible with a specific username and password such that it can protect the privacy of its users.

1.3.3.2 Components

1.3.2.2.1 Hardware List

There is no hardware involved in this sub-project.

1.3.2.2.2 Software List

The following software is used in the sub-projects:

Table 8 - Software used in Server Programing

Software Name	Description	Version Number
PHP	PHP is the server-side programming language. Although 4.4.7 is an old version of PHP, the server used in this project, which is hosted and managed by HKUST iHome, only supports 4.4.7.	4.4.7
MySQL	MySQL is the database language and server. Although 3.23.58 is a very old version of MySQL, the server used in this project, which is hosted and managed by HKUST iHome, only supports 3.23.58.	3.23.58
Bootstrap	Bootstrap is an open source CSS Template to beautify webpages.	3.3.7-dist
phpMyAdmin	A graphical user interface for controlling MySQL database and generating SQL query.	2.5.7-pl1

1.3.3.3 System Block Diagram

The system block diagram of the sub-project is shown below:

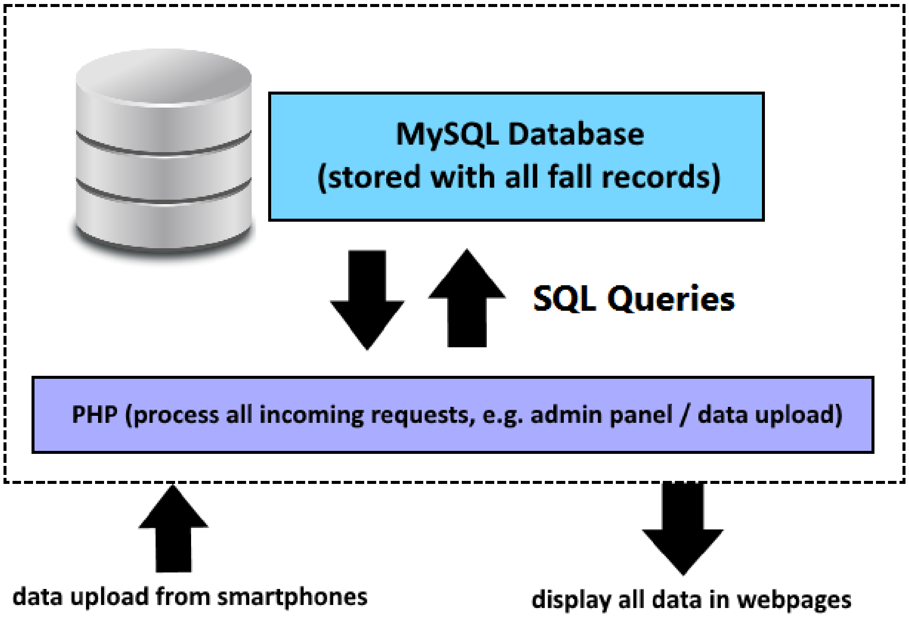

MySQL Database
(stored with all fall records)
SQL Queries
PHP (process all incoming requests, e.g. admin panel / data upload)
data upload from smartphones
display all data in webpages

Figure 8 - System Block Diagram of Sensation Server Program

1.3.3.4 Sub-Project Tasks

These are tasks related to this sub-project:

Backend Designs:

- Login / Logout (PHP Session) Programming (solely done by Wyman)
- SQL Database Initialization
- PHP – MySQL Database Connection Programming
- Add a record to the database using PHP
- Link the latitude and longitude information with Google Maps

Frontend Designs:

- Using Bootstrap to change the layout

1.3.3.5 Technical Challenges

- Using Google Maps API requires mixed use of JavaScript and HTML. This could make the server code becomes lengthy and unmanageable. Therefore, a simpler workaround should be employed instead of using Google Maps API.

1.3.3.6 Budget

Since we are using iHome Web Hosting Service provided by HKUST, there is no server cost induced in this sub-project.

1.3.4 Sub-Project 4 – iOS App Programming

1.3.4.1 Sub-Project Description

iOS is one of the most popular mobile operating systems in the world. Since all iOS devices newer than iPhone 4S has already equipped with Bluetooth Low Energy, we can make our Sensation Smart Watch to be compatible with most iOS devices. Swift 3.0 will be used as the programming language of this iOS App.

At the end of this sub-project, a functional iOS app should be created. This application should be able to find and connect with the watch. Also, it should be able to send and receive all health information, synchronize watch time, and send fall event to the central server when it receives fall alerts from the watch. It should also be able to run in the background, such that user does not need to stay inside the app all the time.

1.3.4.2 Components

1.3.4.2.1 Hardware

There is no hardware involved in this sub-project.

1.3.4.2.2 Software

The following software is used in the sub-projects:

Table 9 - Software used in Programming iOS App

Software Name	Description	Version Number
XCode	XCode is the integrated development environment (IDE) of programming iOS applications.	8.3.1 (8E1000a)

1.3.4.3 System Block Diagram

The system block diagram (program flowchart) of the sub-project is shown below:

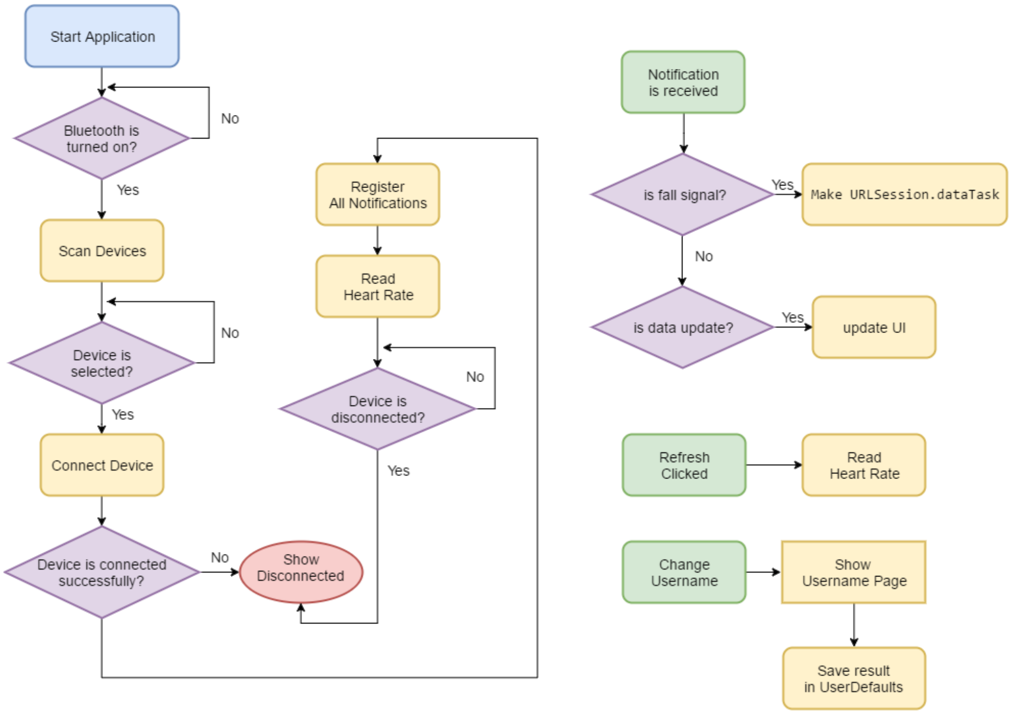


Figure 9 - Flowchart of Sensation Smart Watch iOS App

1.3.4.4 Sub-Project Tasks

These are tasks related to this sub-project:

- Create XCode Project
- Programming of Device Scan View Controller
- Programming of Device Control View Controller
- Programming of Fall Event URL Session Task
- Project Debug

1.3.4.5 Technical Challenges

- iOS Emulator does not have Bluetooth supports.
  Therefore, a real iOS device is required for development.

- Swift 3.0 is a relatively new programming language. There are very few community supports and examples for using CoreBluetooth Library with Swift 3.0.

1.3.4.6 Budget

XCode is a free software can be installed in all Macintosh, and we currently do not have any plans to put this app to App Store, so there is no development cost induced in this sub-project.

Section 1.4: Report Outline

The rest of this paper is organized as follows: Section 2.1 includes the methodology of the work undertaken to design the schematic and PCB layouts of the Sensation Smart Watch. Section 2.2 includes the methodology of sensors software designs including the pedometer, fall detection and UI designs in Sensation Smart Watch. Section 2.3 includes the methodology used for developing server application for Sensation Smart Watch. For Section 2.4, the methodology of creating the iOS application will be included. Part 3 will mainly focus on project results and evaluation. Conclusion will be included in the last part.

Chapter 2 – Methodology

This chapter includes four main sections. Each section describes the design and implementation and testing of one sub-project.

Section 2.1: Circuit Designs and Methodology

2.1.1 Design

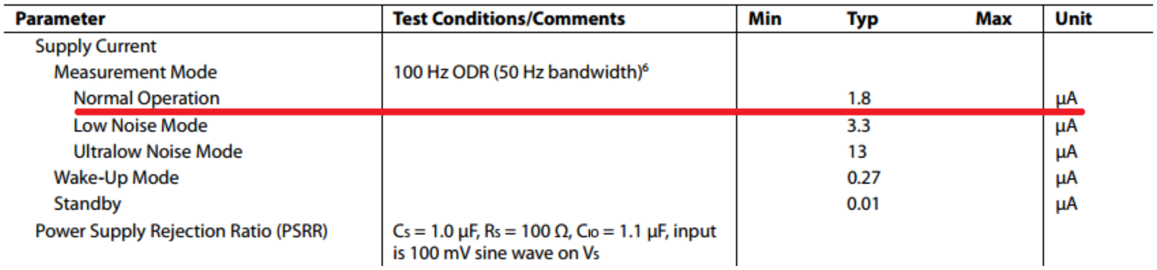
This part describes hardware choices, and the reasons of choosing these components. Schematic and PCB Layout design background will also be covered.

2.1.1.1 Choice of Accelerometer

ADXL362 from Analog Device is chosen for our project.

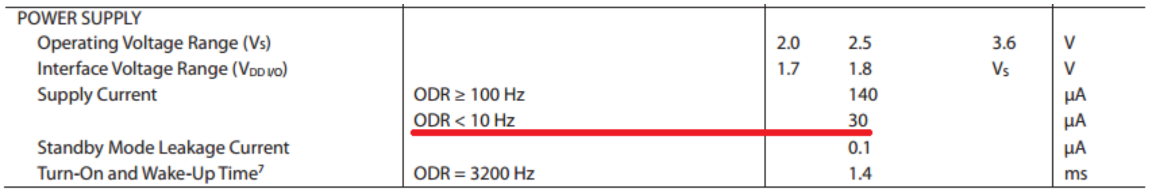
We have compared 4 accelerometers in the market, including ADXL362 from ADI, LSM9DS0 from ST, ADXL345 from ADI and MPU6050 from InvenSense. However, only ADXL362 provide these advantages:

1. **Ultra-low power consumption**
   The figure shows that ADXL362 can measure data continuously (Normal Operation Mode) with only 1.8uA [13]. However, for other sensors, they use much larger current [14], [15], [16]. Power consumption is an important issue because pedometer needs continuous monitoring on acceleration, therefore we could only choose ADXL362.



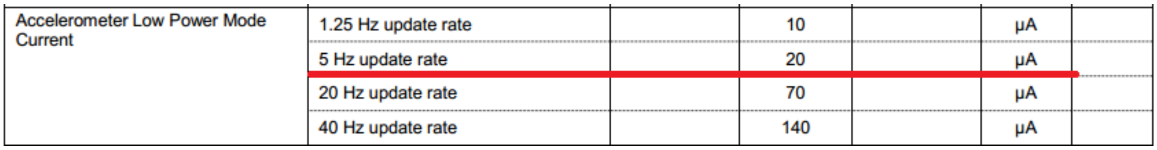
Parameter	Test Conditions/Comments	Min	Typ	Max	Unit
Supply Current					
Measurement Mode	100 Hz ODR (50 Hz bandwidth)[6]				
Normal Operation			1.8		μA
Low Noise Mode			3.3		μA
Ultralow Noise Mode			13		μA
Wake-Up Mode			0.27		μA
Standby			0.01		μA
Power Supply Rejection Ratio (PSRR)	C_S = 1.0 μF, R_S = 100 Ω, C_{IO} = 1.1 μF, input is 100 mV sine wave on V_S				

Figure 10 - Electrical Characteristic of ADXL362



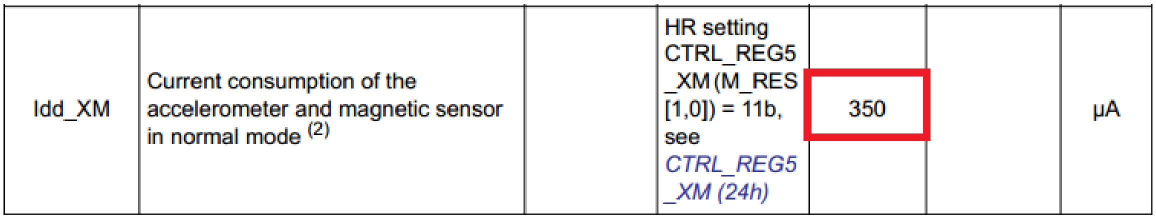
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD I/O}$)		1.7	1.8	V_S	V
Supply Current	ODR ≥ 100 Hz		140		μA
	ODR < 10 Hz		30		μA
Standby Mode Leakage Current			0.1		μA
Turn-On and Wake-Up Time[7]	ODR = 3200 Hz		1.4		ms

Figure 11 - Electrical Characteristic of ADXL345



Accelerometer Low Power Mode Current	1.25 Hz update rate		10		μA	
	5 Hz update rate		20		μA	
	20 Hz update rate		70		μA	
	40 Hz update rate		140		μA	

Figure 12 - Electrical Characteristic of MPU6050



Idd_XM	Current consumption of the accelerometer and magnetic sensor in normal mode [(2)]		HR setting CTRL_REG5 _XM (M_RES [1,0]) = 11b, see *CTRL_REG5 _XM (24h)*	350		µA

Figure 13 - Electrical Characteristic of LSM9DS0

2. **FIFO Buffer Size is big enough**
   The FIFO Buffer in ADXL362 can store 512 samples. Each set of data contains 4 samples, including X-axis, Y-axis, Z-axis and Temperature. Therefore, it can store 512/4 = 128 sets of data. If temperature is not required, it can store at most 512/3 = 170 sets of data. For 12.5Hz, 170 sets of data, it can store around 13 seconds' data [13].

   The FIFO Buffer of LSM9DS0 and ADXL345 are only limited to 32 sets of data [16], [14]. Although MPU6050 provides much larger buffer size (1024 bytes) [15], the RAM in CC2541 is limited to 8KB [17] such that we cannot allocate that much memory for storing data fetched from FIFO buffer.

3. **Two Configurable Interrupt Pins are provided**
   "Configurable Interrupt Pin" means that the functionality of interrupt pins provided by the accelerometer should not be limited to only one function. In fact, all selected accelerometers provide this feature. However, MPU6050 has only 1 interrupt pin. Therefore, using MPU6050 might be more difficult to implement both FIFO and motion-activated interrupts. It will be an issue for implementing "Raise to wake" feature later because the motion can only be identified when the FIFO full event is triggered. That means, interrupt could be generated for "raise to wake" event exclusively.

   After selecting accelerometer, we finished some basic testing using an evaluation module. The details of testing are listed in "Implementation and Testing" part.

2.1.1.2 Choice of PPG Heart Rate Sensor

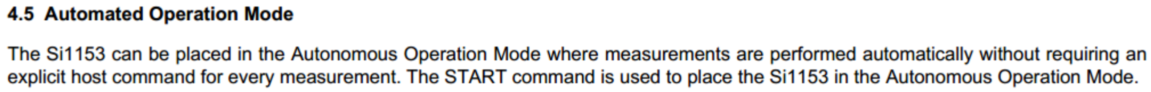
MAX30101 from Maxim Integrated is chosen for our project.

We have compared 3 PPG heart-rate sensors in the market, including Si1153 from Silicon Labs, Arduino Pulse Sensor from WorldFamousElectronics and MAX30101 from Maxim Integrated. MAX30101 is finally chosen because of:

1. **Internally included 3 LEDs**
   MAX30101 included 3 different kinds of LED internally, they are infrared, red and green LED. Using internally embedded LEDs and LED driver can save space in the circuitry of controlling LEDs, which is also easier and more power saving than external LED driver circuit. On the other hand, Si1153 does not have any LED included. Although Arduino Pulse Sensor does have green LED included, it is controlled by an

external LED driver circuit so we also need external power circuit if we want to use it.

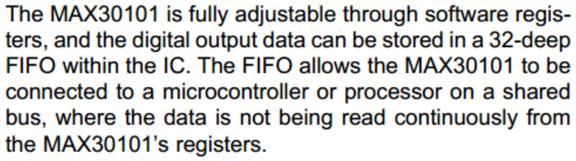
2. **Autonomous Measurement Mode Supports**
   Autonomous measurement mode means the sensor can measure and store the data automatically at a specified interval. When the buffer inside the sensor is almost full, it triggers an interrupt to the microprocessor. Using this mode ensures the microprocessor can get all data in very accurate intervals even if the processor is busy. It is crucial in heart rate sensing because heart rate is calculated using the time intervals between two peaks, if the time intervals between two data are changing, the heart rate calculation would be very difficult. In fact, only MAX30101 and Si1153 have the autonomous mode support.



4.5 Automated Operation Mode

The Si1153 can be placed in the Autonomous Operation Mode where measurements are performed automatically without requiring an explicit host command for every measurement. The START command is used to place the Si1153 in the Autonomous Operation Mode.

Figure 14 - Si1153 has Autonomous Mode



The MAX30101 is fully adjustable through software registers, and the digital output data can be stored in a 32-deep FIFO within the IC. The FIFO allows the MAX30101 to be connected to a microcontroller or processor on a shared bus, where the data is not being read continuously from the MAX30101's registers.

Figure 15 - MAX30101 has Autonomous Mode

After selecting this sensor, we finished some basic testing using an evaluation module. The details of testing are listed in "Implementation and Testing" part.

2.1.1.3 Choice of Bluetooth Microprocessor

The selection criteria of MCU are:

1. **Easily found and supported in market**
   In our project, we would only buy a small number of MCU. If it is too difficult to find and purchase, we might need extra costs in both shipping and production.

2. **Embedded with Bluetooth 4.0 Connectivity**
   To minimize the power consumption and size of our watch, we would like to find a single SoC (System on Chip) that handles both Bluetooth tasks and application. In fact, there are numerous of manufacturers providing this kind of BLE SoC, such as Texas Instruments, Nordic and Dialog.

3. **Complete Usable Code Example**
   To reduce undesirable efforts in programming, we prefer choosing MCUs that has many different examples provided officially. In fact, we found that the BLE Stack Software provided by Texas Instruments has more than 15 example projects.

We have compared three microprocessors including CC2540, CC2541 and CC2640 during our selection. Finally, **CC2541** is chosen because of its simplicity and I2C support. The details of how do we compare among them are listed in the "Implementation and Testing" part.

2.1.1.4 PCB Layout Designs

By designing our customized PCB board, we want to achieve these goals:

1. **Compacts the size of PCB while keeping the solderability.**
   Although we want to make the PCB size as small as possible, as we are using manual soldering and rework during our project, we need to make sure the space between two components should never be too small. Otherwise, it would be extremely difficult to solder all components tightly and safely.

2. **Creates a low-noise environment to the noise-sensitive sensors.**
   ADXL362 and MAX30101 have internal ADCs that are very sensitive to external noises. Therefore, we need independent voltage regulators and extra capacitors near the voltage source to isolate noises produced by the lithium-ion battery, the buzzer and the microprocessor in the circuit. Also, to make sure all components have the same reference ground, we also need to create a ground plane to reduce resistance.

3. **Avoid changing the radiation pattern of Bluetooth PCB antenna.**
   The Bluetooth processor module (JDY-08) we have selected is using PCB antenna. To make sure the radiation pattern of the antenna would not be affected (or even worsen), we need to avoid putting any copper lines or planes under the PCB antenna.

All these rules are followed during the implementation of our own customized PCB. Please refer to the "Implementation and Testing" part to understand more about how these goals are achieved and tested.

2.1.2 Implementation and Testing

2.1.2.1 Accelerometer Implementation and Testing

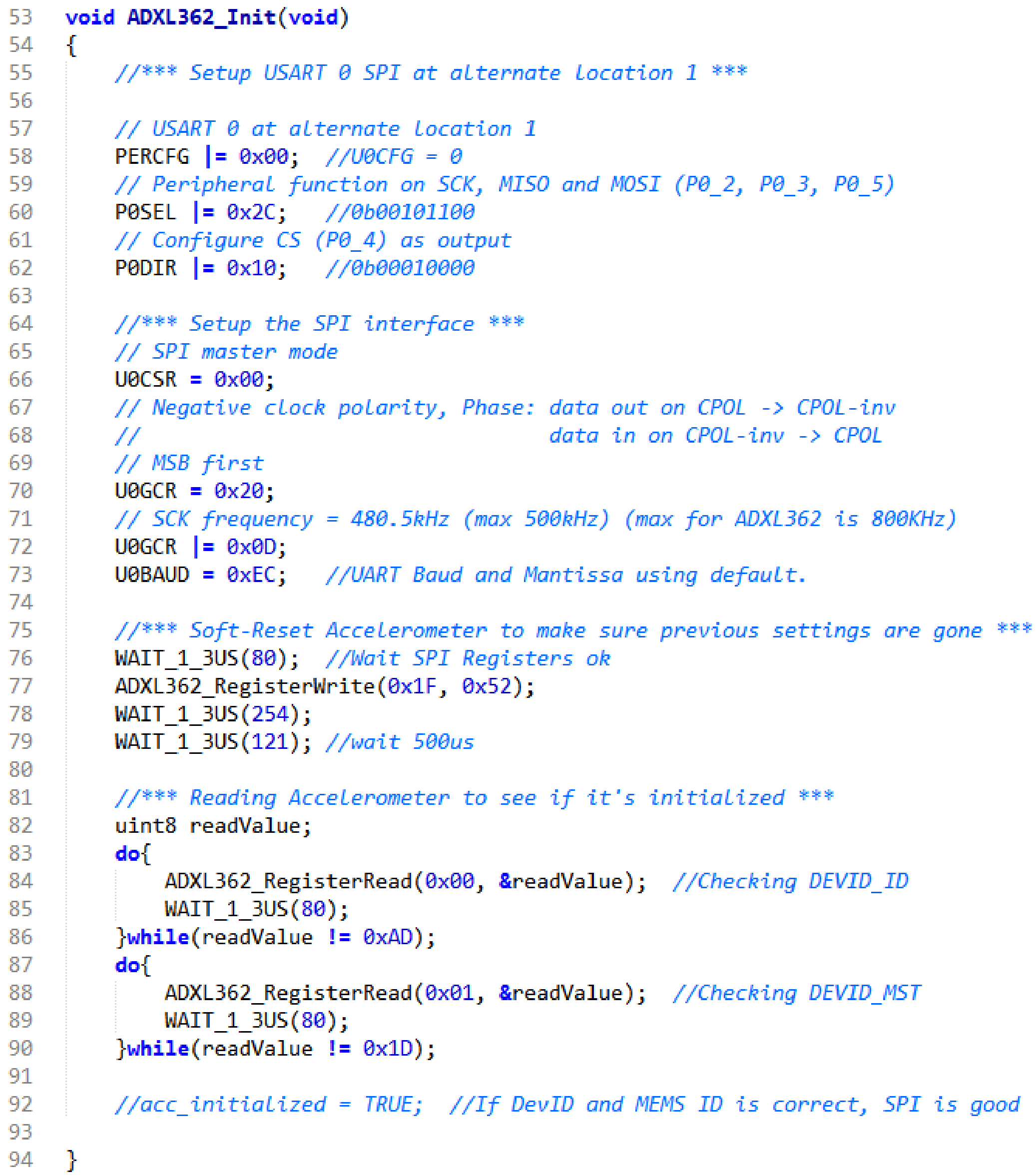
After choosing ADXL362 as our accelerometer, we have implemented basic I/O drivers for communication between ADXL362 and microprocessor. The following parts are the implementation and testing results.

2.1.2.1.1 Implementation

All I/O operations of ADXL362 is using SPI, the serial peripheral interface bus. So, we need to initialize and enable the SPI pins in CC2541 (the Bluetooth microprocessor) first, after that, initialization commands can be issued from CC2541 to ADXL362 to initialize it properly.

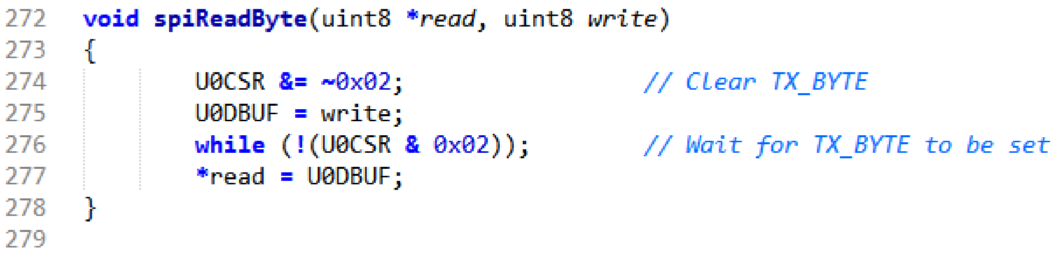
However, even though CC2541 has the SPI pins and SPI supports, there are no official SPI drivers or C APIs. We need to write the SPI driver by ourselves.

We have successfully initialized the accelerometer and read the device ID using the CC2541 SPI driver solely developed by us. Figure 16 shows the code of SPI initialization and ADXL362 existence checking:



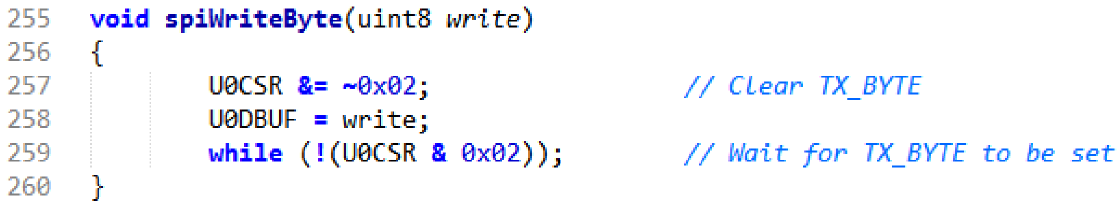
```c
void ADXL362_Init(void)
{
    //*** Setup USART 0 SPI at alternate location 1 ***

    // USART 0 at alternate location 1
    PERCFG |= 0x00;  //U0CFG = 0
    // Peripheral function on SCK, MISO and MOSI (P0_2, P0_3, P0_5)
    P0SEL |= 0x2C;   //0b00101100
    // Configure CS (P0_4) as output
    P0DIR |= 0x10;   //0b00010000

    //*** Setup the SPI interface ***
    // SPI master mode
    U0CSR = 0x00;
    // Negative clock polarity, Phase: data out on CPOL -> CPOL-inv
    //                                 data in on CPOL-inv -> CPOL
    // MSB first
    U0GCR = 0x20;
    // SCK frequency = 480.5kHz (max 500kHz) (max for ADXL362 is 800KHz)
    U0GCR |= 0x0D;
    U0BAUD = 0xEC;   //UART Baud and Mantissa using default.

    //*** Soft-Reset Accelerometer to make sure previous settings are gone ***
    WAIT_1_3US(80);  //Wait SPI Registers ok
    ADXL362_RegisterWrite(0x1F, 0x52);
    WAIT_1_3US(254);
    WAIT_1_3US(121); //wait 500us

    //*** Reading Accelerometer to see if it's initialized ***
    uint8 readValue;
    do{
        ADXL362_RegisterRead(0x00, &readValue);  //Checking DEVID_ID
        WAIT_1_3US(80);
    }while(readValue != 0xAD);
    do{
        ADXL362_RegisterRead(0x01, &readValue);  //Checking DEVID_MST
        WAIT_1_3US(80);
    }while(readValue != 0x1D);

    //acc_initialized = TRUE;  //If DevID and MEMS ID is correct, SPI is good

}
```

Figure 16 - SPI Initialization and ADXL362 Existence Checking



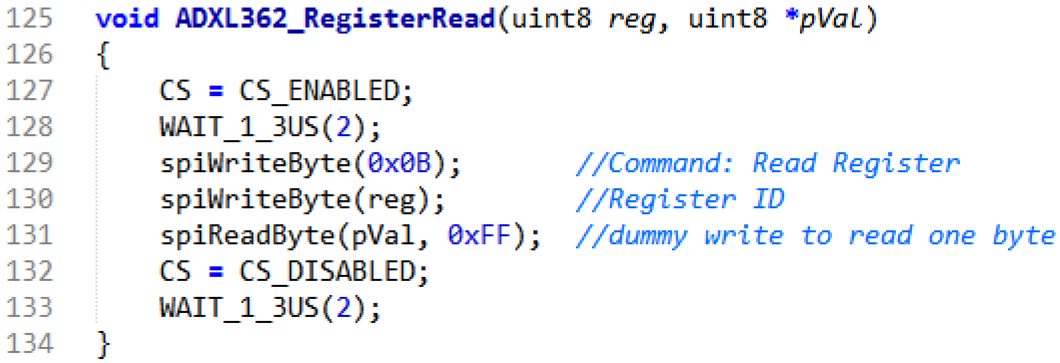
```c
void spiReadByte(uint8 *read, uint8 write)
{
        U0CSR &= ~0x02;                 // Clear TX_BYTE
        U0DBUF = write;
        while (!(U0CSR & 0x02));        // Wait for TX_BYTE to be set
        *read = U0DBUF;
}

```

Figure 17 - SPI Read Byte Driver Function



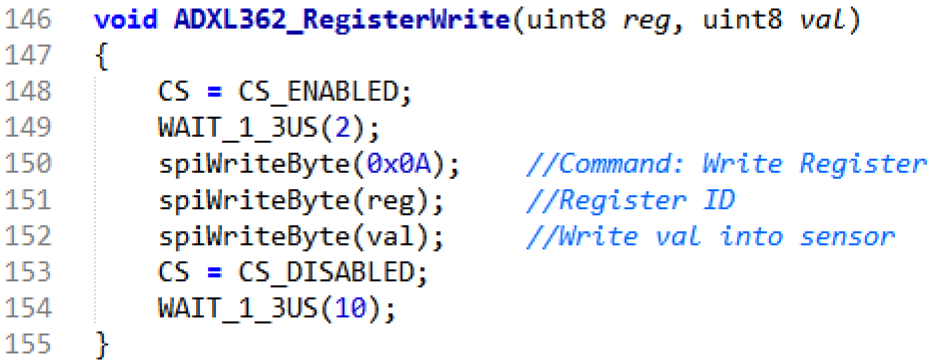
```
void spiWriteByte(uint8 write)
{
        U0CSR &= ~0x02;                 // Clear TX_BYTE
        U0DBUF = write;
        while (!(U0CSR & 0x02));        // Wait for TX_BYTE to be set
}
```

Figure 18 - SPI Write Byte Driver Function



```
void ADXL362_RegisterRead(uint8 reg, uint8 *pVal)
{
    CS = CS_ENABLED;
    WAIT_1_3US(2);
    spiWriteByte(0x0B);        //Command: Read Register
    spiWriteByte(reg);         //Register ID
    spiReadByte(pVal, 0xFF);   //dummy write to read one byte
    CS = CS_DISABLED;
    WAIT_1_3US(2);
}
```

Figure 19 - ADXL362 Driver Function (Read)



```
void ADXL362_RegisterWrite(uint8 reg, uint8 val)
{
    CS = CS_ENABLED;
    WAIT_1_3US(2);
    spiWriteByte(0x0A);     //Command: Write Register
    spiWriteByte(reg);      //Register ID
    spiWriteByte(val);      //Write val into sensor
    CS = CS_DISABLED;
    WAIT_1_3US(10);
}
```

Figure 20 - ADXL362 Driver Function (Write)

2.1.2.1.2 Testing

The self-developed driver code in figure 16, 17,18, 19 and 20 successfully initialize and perform I/O operations from SPI. Device ID can be read successfully.

2.1.2.2 PPG Sensor Implementation and Testing

After choosing MAX30101 as our heart rate sensor, we have implemented basic I/O drivers for communication between MAX30101 and microprocessor. The following parts are the implementation of the driver functions and testing results.

***** Points to note about MAX30100, MAX30101 and MAX30102 ***:**

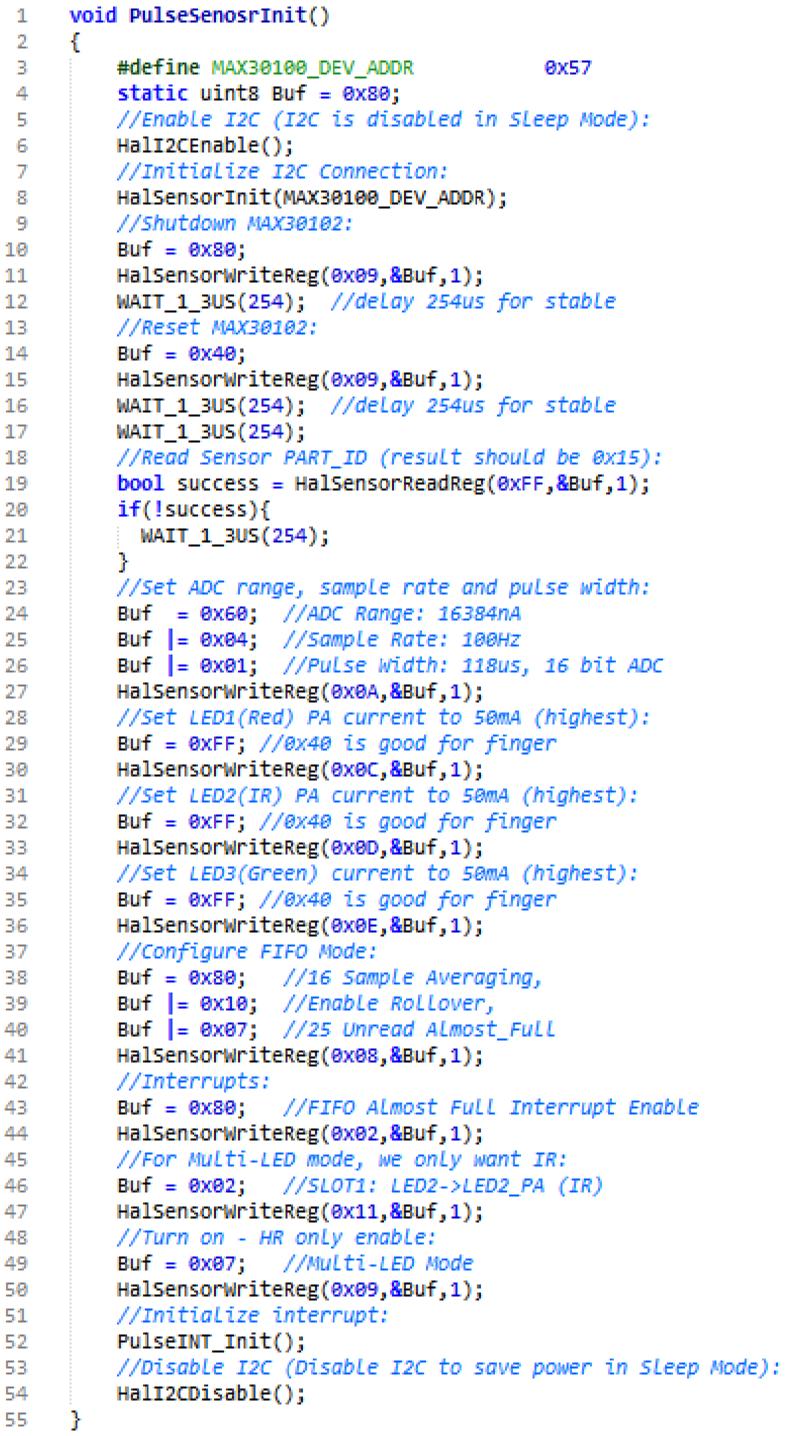
In fact, both MAX30100, MAX30101 and MAX30102 were put into consideration during the selection of PPG sensing. These heart rate sensors share the same register maps, Device ID and footprints. (For register maps of MAX30100, please kindly refer to CHEUNG Wai Man, Raymond's report.) Before making final decision of using MAX30101, MAX30102 and MAX30100 are also evaluated and well tested. However, as MAX30101 and MAX30102 support 5V LED voltage, it provides higher LED brightness than MAX30100 (MAX30100 only supports 3.3V maximum.)

During several times of experiments, we have also found that MAX30100 is less accurate than MAX30101 and MAX30102 due to its lower ADC resolution. Lower ADC resolution would be a serious problem because heart rate sensing in wrists requires generally higher sensitivity than in fingertips. For more details about the ADC resolution provided by MAX30100, please kindly read the datasheet of MAX30100.

The difference between MAX30101 and MAX30102 is the green LED – only MAX30101 provides green LED supports. It could potentially increase the reliability of heart rate sensing because green light has a much shorter wavelength than red and IR. Even if we do not use the green light, we could still use MAX30101 because it also has Red and IR LED. As the unit price of MAX30101 and MAX30102 are same, we decided to use MAX30101.

2.1.2.2.1 Implementation

MAX30101 uses I2C as the serial communication protocol. CC2541 provided I2C HAL (Hardware Abstraction Layer) Library for us. Therefore, we only implemented the driver functions of initializations of MAX30101. The I2C initialization is handled by the HAL.



```
 1  void PulseSenosrInit()
 2  {
 3      #define MAX30100_DEV_ADDR           0x57
 4      static uint8 Buf = 0x80;
 5      //Enable I2C (I2C is disabled in Sleep Mode):
 6      HalI2CEnable();
 7      //Initialize I2C Connection:
 8      HalSensorInit(MAX30100_DEV_ADDR);
 9      //Shutdown MAX30102:
10      Buf = 0x80;
11      HalSensorWriteReg(0x09,&Buf,1);
12      WAIT_1_3US(254);  //delay 254us for stable
13      //Reset MAX30102:
14      Buf = 0x40;
15      HalSensorWriteReg(0x09,&Buf,1);
16      WAIT_1_3US(254);  //delay 254us for stable
17      WAIT_1_3US(254);
18      //Read Sensor PART_ID (result should be 0x15):
19      bool success = HalSensorReadReg(0xFF,&Buf,1);
20      if(!success){
21        WAIT_1_3US(254);
22      }
23      //Set ADC range, sample rate and pulse width:
24      Buf  = 0x60;  //ADC Range: 16384nA
25      Buf |= 0x04;  //Sample Rate: 100Hz
26      Buf |= 0x01;  //Pulse Width: 118us, 16 bit ADC
27      HalSensorWriteReg(0x0A,&Buf,1);
28      //Set LED1(Red) PA current to 50mA (highest):
29      Buf = 0xFF; //0x40 is good for finger
30      HalSensorWriteReg(0x0C,&Buf,1);
31      //Set LED2(IR) PA current to 50mA (highest):
32      Buf = 0xFF; //0x40 is good for finger
33      HalSensorWriteReg(0x0D,&Buf,1);
34      //Set LED3(Green) current to 50mA (highest):
35      Buf = 0xFF; //0x40 is good for finger
36      HalSensorWriteReg(0x0E,&Buf,1);
37      //Configure FIFO Mode:
38      Buf = 0x80;   //16 Sample Averaging,
39      Buf |= 0x10;  //Enable Rollover,
40      Buf |= 0x07;  //25 Unread Almost_Full
41      HalSensorWriteReg(0x08,&Buf,1);
42      //Interrupts:
43      Buf = 0x80;   //FIFO Almost Full Interrupt Enable
44      HalSensorWriteReg(0x02,&Buf,1);
45      //For Multi-LED mode, we only want IR:
46      Buf = 0x02;   //SLOT1: LED2->LED2_PA (IR)
47      HalSensorWriteReg(0x11,&Buf,1);
48      //Turn on - HR only enable:
49      Buf = 0x07;   //Multi-LED Mode
50      HalSensorWriteReg(0x09,&Buf,1);
51      //Initialize interrupt:
52      PulseINT_Init();
53      //Disable I2C (Disable I2C to save power in Sleep Mode):
54      HalI2CDisable();
55  }
```

Figure 21 - Initialization Code for MAX30101

Figure 21 shows the initialization process of MAX30101. The initialization code is well structured such that we could change the status of LED light (e.g. turn on IR only / turn on Red only / turn on Green only / turn off all) very easily by only changing line 46.

2.1.2.2.2 Testing

MAX30101 can be initialized successfully using the code in Figure 21. All I/O operations of MAX30101 can be performed successfully.

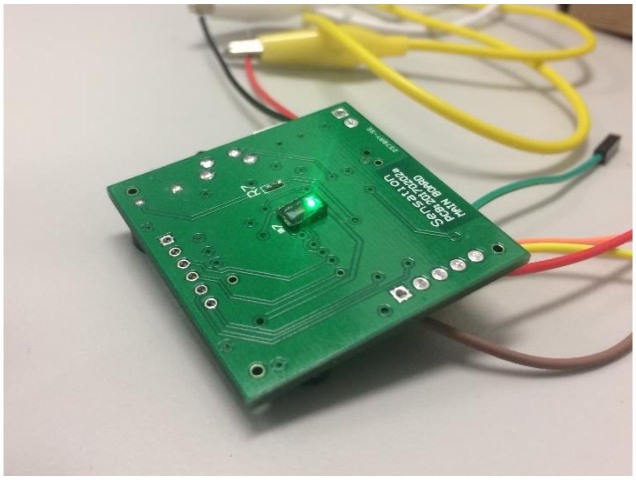


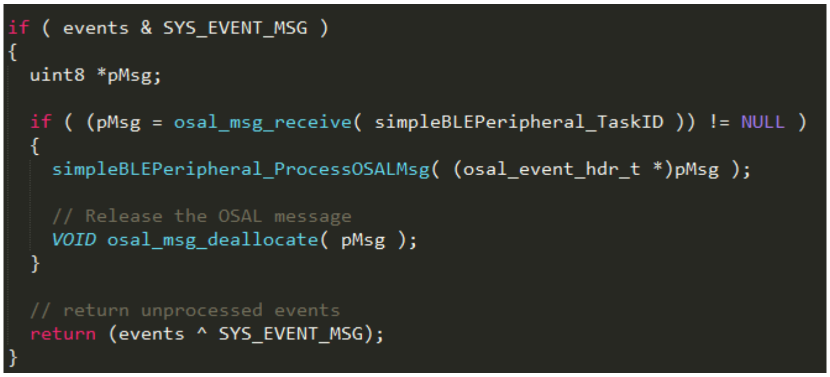
Figure 22 - Green LED in MAX30101 is initialized successfully

2.1.2.3 Bluetooth Processor Implementation and Testing

We have compared 3 MCUs in the market, including CC2540, CC2541 and CC2640. These three MCUs are chosen because they are actively supported by Texas Instruments. Also, there are many community supports provided in Texas Instruments Community and AmoMCU (a hardware manufacturer in Shenzhen). By studying the source code examples and their specifications, we finally choose CC2541 for these reasons:

1. **The source code in CC254x is more manageable than CC2640**
   We have compared the same project "SimpleBLEPeripheral" for both CC254x and CC2640 provided by TI. It is found that the logic in CC254x is more manageable because the operating system used in CC254x is non-preemptive. Hence, all code related to semaphore does not exist in CC254x. For example:

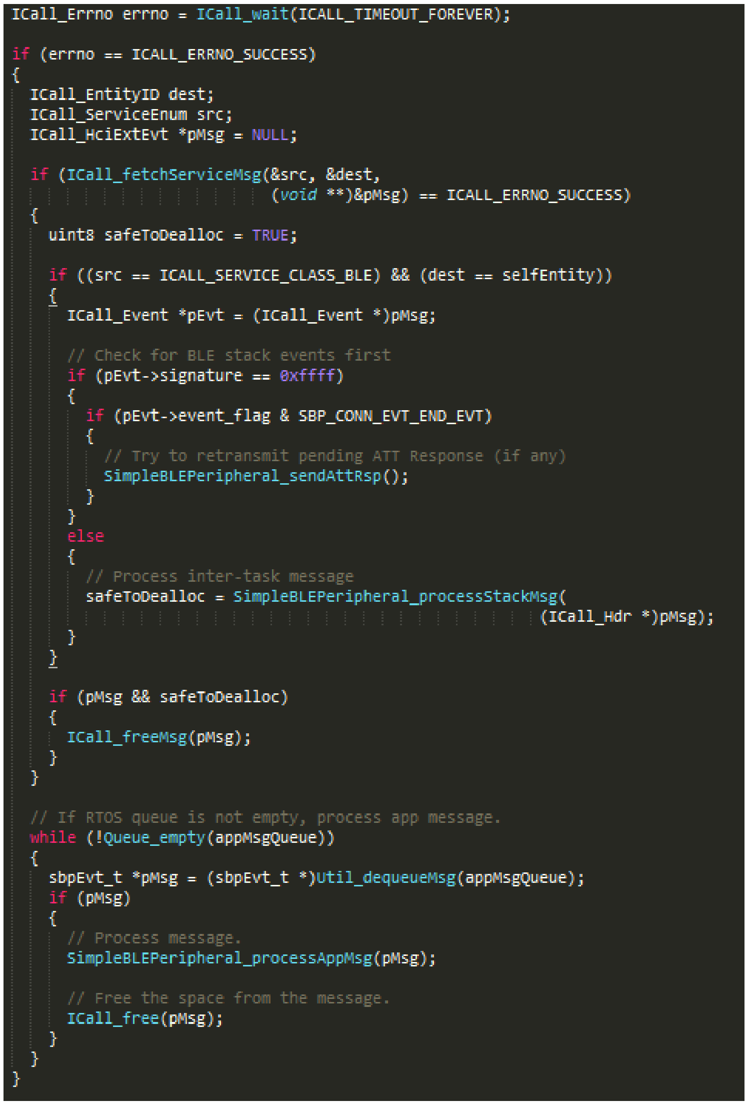
   In CC2540, the code handling messages from other tasks is neat:



```
if ( events & SYS_EVENT_MSG )
{
  uint8 *pMsg;

  if ( (pMsg = osal_msg_receive( simpleBLEPeripheral_TaskID )) != NULL )
  {
    simpleBLEPeripheral_ProcessOSALMsg( (osal_event_hdr_t *)pMsg );

    // Release the OSAL message
    VOID osal_msg_deallocate( pMsg );
  }

  // return unprocessed events
  return (events ^ SYS_EVENT_MSG);
}
```

Figure 23 - Source Code "Message Handling" in CC2541

The code handling messages from other tasks in CC2640:



```
ICall_Errno errno = ICall_wait(ICALL_TIMEOUT_FOREVER);

if (errno == ICALL_ERRNO_SUCCESS)
{
  ICall_EntityID dest;
  ICall_ServiceEnum src;
  ICall_HciExtEvt *pMsg = NULL;

  if (ICall_fetchServiceMsg(&src, &dest,
                            (void **)&pMsg) == ICALL_ERRNO_SUCCESS)
  {
    uint8 safeToDealloc = TRUE;

    if ((src == ICALL_SERVICE_CLASS_BLE) && (dest == selfEntity))
    {
      ICall_Event *pEvt = (ICall_Event *)pMsg;

      // Check for BLE stack events first
      if (pEvt->signature == 0xffff)
      {
        if (pEvt->event_flag & SBP_CONN_EVT_END_EVT)
        {
          // Try to retransmit pending ATT Response (if any)
          SimpleBLEPeripheral_sendAttRsp();
        }
      }
      else
      {
        // Process inter-task message
        safeToDealloc = SimpleBLEPeripheral_processStackMsg(
                                                         (ICall_Hdr *)pMsg);
      }
    }

    if (pMsg && safeToDealloc)
    {
      ICall_freeMsg(pMsg);
    }
  }

  // If RTOS queue is not empty, process app message.
  while (!Queue_empty(appMsgQueue))
  {
    sbpEvt_t *pMsg = (sbpEvt_t *)Util_dequeueMsg(appMsgQueue);
    if (pMsg)
    {
      // Process message.
      SimpleBLEPeripheral_processAppMsg(pMsg);

      // Free the space from the message.
      ICall_free(pMsg);
    }
  }
}
```

Figure 24- Source Code "Message Handling" in CC2640

By observing the length of the code, it shows that the code in CC254x is much shorter than CC264x. Besides, it can also prove that the operating system used in CC254x (as known as OSAL, Operating System Abstraction Layer) is a non-preemptive task scheduler. Using it would be beneficial because preemptive tasks and threads are rare in our project.

2. **CC2541 supports Hardware I2C**
   CC2540 does not support I2C. I2C is a communication protocol used in heart rate sensor. Although we could use GPIO to implement the same function (as known as "bit-banging"), it would be more difficult to debug. Therefore, CC2541 would be a better choice than CC2540 in this project.

The comparison shows that CC2541 from Texas Instruments is chosen for our project because of its manageability in coding, I2C supports and availability of code examples.

2.1.2.4 PCB Layout Designs Implementation and Testing

2.1.2.4.1 Implementation

To achieve goals mentioned in the design section (i.e. compact size, solderability, low-noise and reserving radiation pattern of PCB antenna), we have employed these rules during PCB layout work:

1. **Using 0603 or larger components**
   0603 refers to "the SMD component size is 0.06 inch * 0.03 inch". We are using 0603 components instead of the smaller 0402 in our project. Also, we make the soldering pad size a bit larger than usual, therefore, the solderability can be improved.

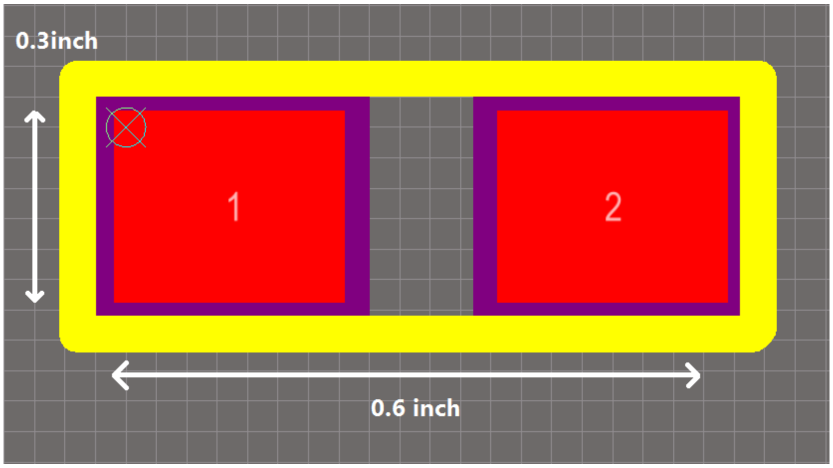


Figure 25 - 0603 Footprint with Larger Pad Size

2. **Deliberately leaving extra space between two components**
   In the PCB layout, we deliberately leave some empty space between two components to avoid short circuits due to excessive soldering tins. This could potentially reduce the difficulty in soldering and debugging work due to short circuit.

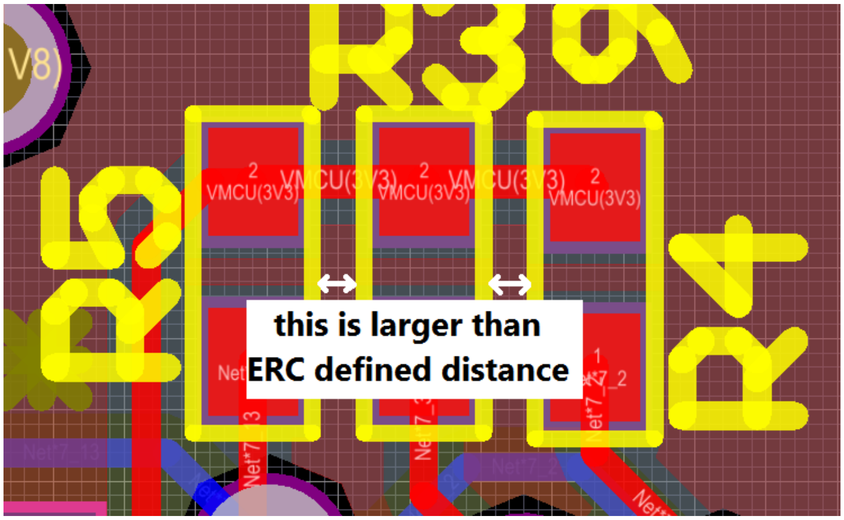
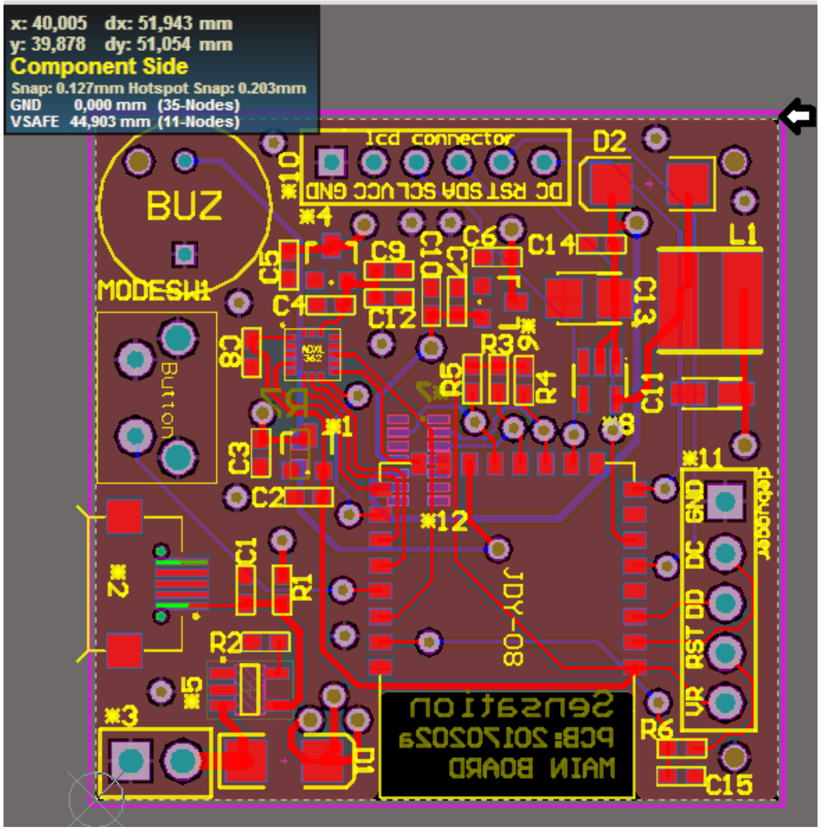


Figure 26 - Extra space between two components

3. **Size of PCB is fixed to be smaller than 41mm * 41mm**
   41mm is the one of the very common size of men's watch. The PCB size should never be larger than 41mm * 41mm. Otherwise it would be uncomfortable to wear.



x: 40,005 dx: 51,943 mm
y: 39,878 dy: 51,054 mm
Component Side
Snap: 0.127mm Hotspot Snap: 0.203mm
GND 0,000 mm (35-Nodes)
VSAFE 44,903 mm (11-Nodes)

*Figure 27- Size of PCB is around 41mm * 41mm*

4. **Open Window is created for PCB Antenna**
   We did not put any copper plane below the PCB Antenna. This can avoid changing the radiation pattern of the antenna or even shielding it. It can be done simply with "Polygon" feature in Altium Designer.

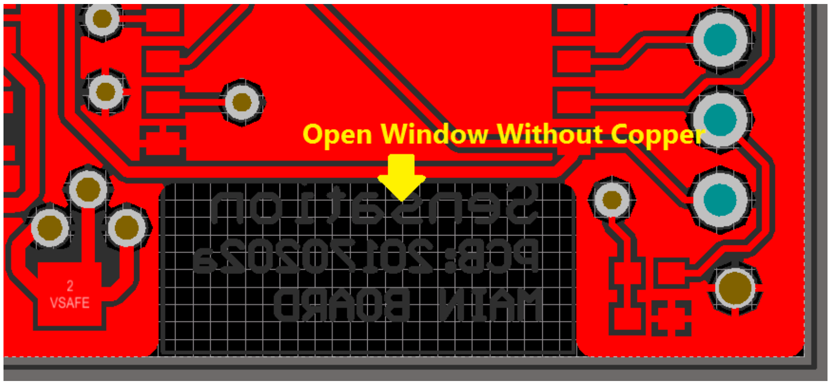


Figure 28 - Open Window for JDY-08 PCB Antenna

2.1.2.4.2 Testing

After finishing the schematic designs and PCB layout, the customized PCB board is printed and soldered properly. One error in USB positioning is found in the latest prototype board, but it can also be fixed with jumping wires easily.

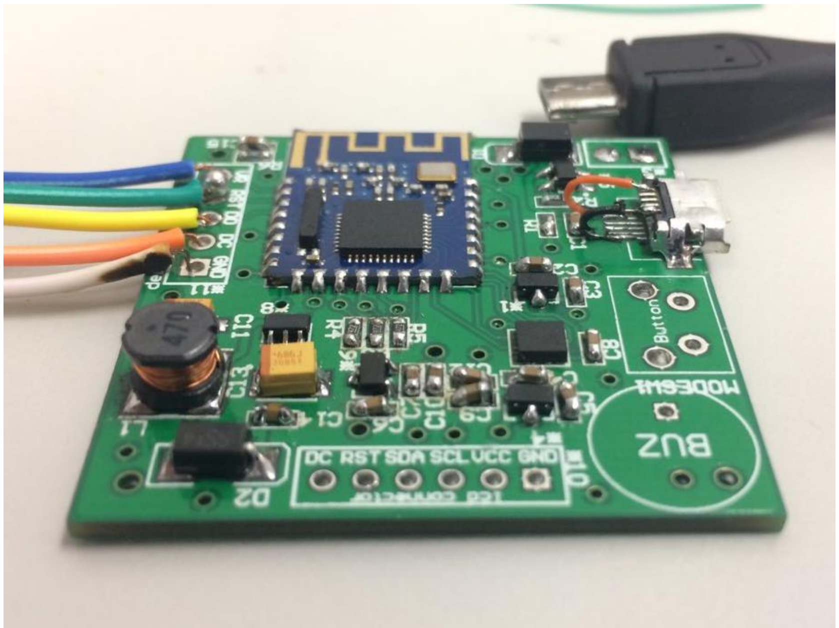


Figure 29 – Customized Sensation Smart Watch PCB (Front Side)

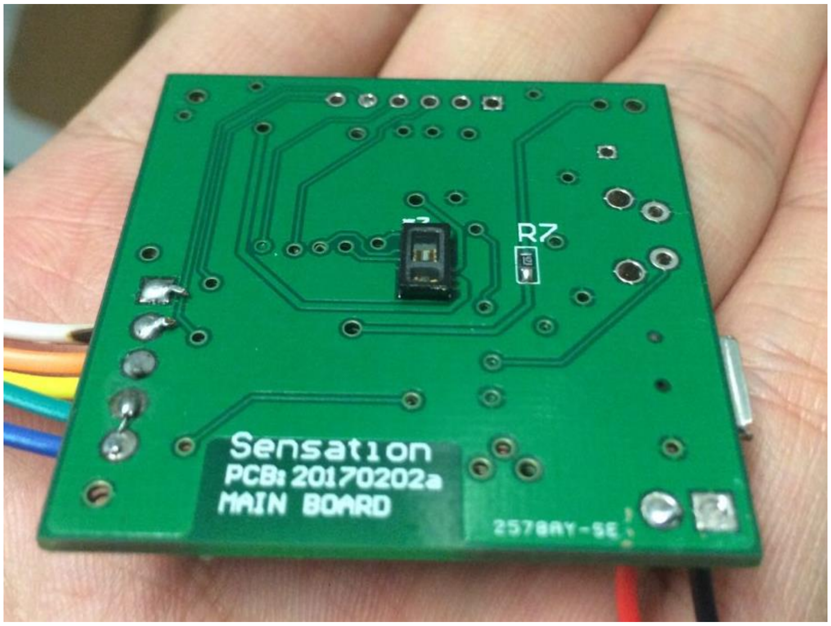


Figure 30 - Customized Sensation Smart Watch PCB (Back Side)

To summarize, all sensors and components including accelerometer, heart rate sensors, microprocessor, Bluetooth antenna, buzzer, step up converters, voltage regulators and battery charger IC are operating normally in the customized PCB. Therefore, this sub-project has been finished and tested with satisfactory testing results.

Section 2.2: Sensors Software Programming

2.2.1 Design

This part describes the sensors software and embedded system software designs, which includes the design of pedometer algorithm, fall detection algorithm and embedded user interface designs.

2.2.1.1 Design of Pedometer Algorithm and Fall Algorithm

As ADXL362 accelerometer has two interrupt pins and a 512-byte buffer, we are making use of all its interrupt pins and its buffer to make the pedometer and fall detection:

- **512-byte Buffer will be used to store the continuous walking patterns:**
  We will use the 512-byte buffer to store the x-axis, y-axis and z-axis data continuously, when the buffer is almost full, the accelerometer will issue an interrupt signal, then, the microprocessor will copy all content inside the 512-byte buffer to microprocessor's pre-allocated RAM for processing. After that, the buffer can be freed to store the new records.

- **INT1 (Interrupt Pin 1) on ADXL362 will be used for buffer almost full interrupt:**
  INT1 in ADXL362 is configured to issue an interrupt signal when the buffer reaches a certain threshold (above half, i.e. 50% in our project). When the microprocessor receives this interrupt, it will read all content of the FIFO as soon as possible.

- **INT2 (Interrupt Pin 2) on ADXL362 will be used for fall detection interrupt:**
  ADXL362 can be configured to issue an interrupt when the total acceleration reaches or below a constant. When the user / the watch falls, the total acceleration of the watch will be lower than the gravitation acceleration. Therefore, we can set an interrupt to be triggered when the total acceleration is lower than 0.6g, where 0.6 is an arbitrary number can be found with several experiments.

We need to take the readings from the MCU to a computer to analyze it before developing the pedometer algorithm, the "Implementation and Testing" part shows all the code, procedures and testing results of taking readings out from the MCU to a computer.

2.2.1.2 Design of User Interface in the watch

Users should be able to switch different modes to view different kinds of data of the watch. For example, the user should be able to view his/her heart-rate, step counts, current time or even switch off the screen for power saving.

To provide the "mode switching" function, a Moore state-machine code exclusively for the button should be created. The flowchart should also be simple such that it would not delays the ongoing tasks in the microprocessor:

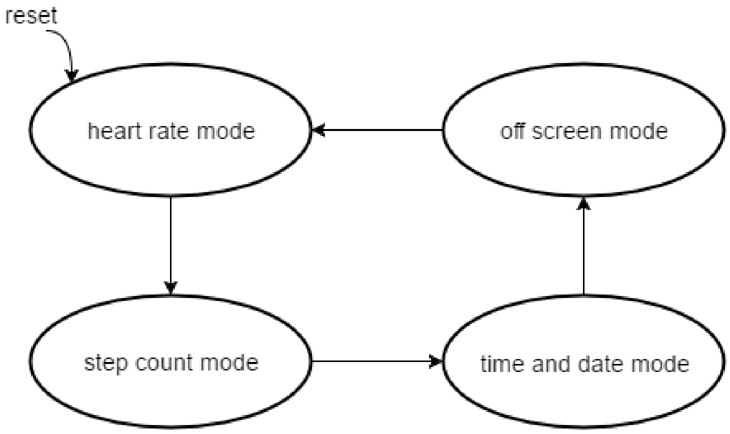


Figure 31 - Moore State Machine of the Button and UI

For fall triggered, there should also be another mechanism for displaying the proper UI:

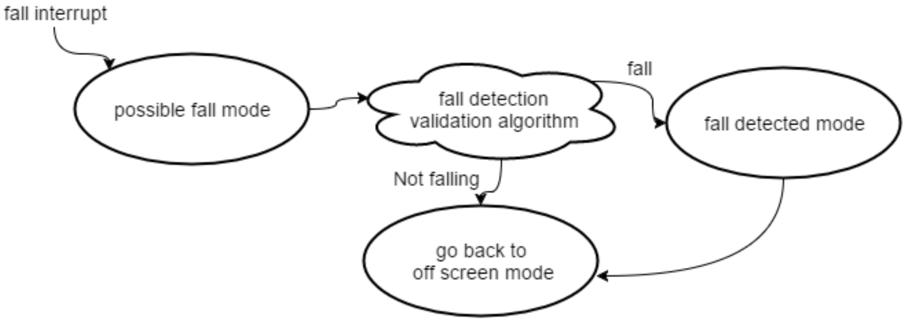


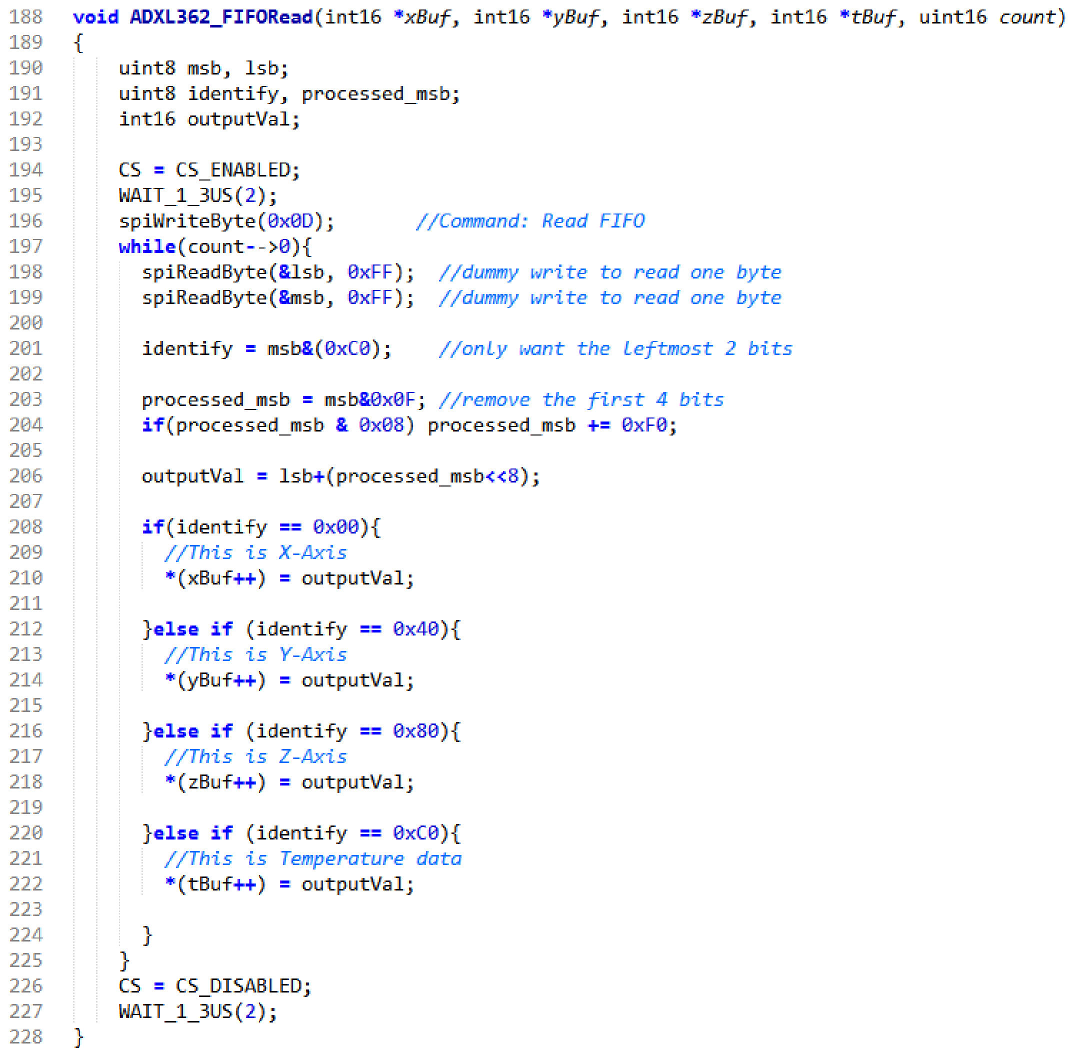
Figure 32 - State Machine for Fall Interrupt UI

2.2.2 Implementation and Testing

2.2.2.1 Pedometer Implementation and Testing

Extracting the acceleration data from ADXL362 FIFO buffer to MCU, then transmit the data from MCU to computer via Bluetooth, can helps us understand the fixed pattern of acceleration during walking. The following part shows the procedures of obtaining the example.

First, we have been successfully written the FIFO MCU Code to read the buffer in ADXL362:



```
void ADXL362_FIFORead(int16 *xBuf, int16 *yBuf, int16 *zBuf, int16 *tBuf, uint16 count)
{
    uint8 msb, lsb;
    uint8 identify, processed_msb;
    int16 outputVal;

    CS = CS_ENABLED;
    WAIT_1_3US(2);
    spiWriteByte(0x0D);         //Command: Read FIFO
    while(count-->0){
      spiReadByte(&lsb, 0xFF);  //dummy write to read one byte
      spiReadByte(&msb, 0xFF);  //dummy write to read one byte

      identify = msb&(0xC0);    //only want the leftmost 2 bits

      processed_msb = msb&0x0F; //remove the first 4 bits
      if(processed_msb & 0x08) processed_msb += 0xF0;

      outputVal = lsb+(processed_msb<<8);

      if(identify == 0x00){
        //This is X-Axis
        *(xBuf++) = outputVal;

      }else if (identify == 0x40){
        //This is Y-Axis
        *(yBuf++) = outputVal;

      }else if (identify == 0x80){
        //This is Z-Axis
        *(zBuf++) = outputVal;

      }else if (identify == 0xC0){
        //This is Temperature data
        *(tBuf++) = outputVal;

      }
    }
    CS = CS_DISABLED;
    WAIT_1_3US(2);
}
```

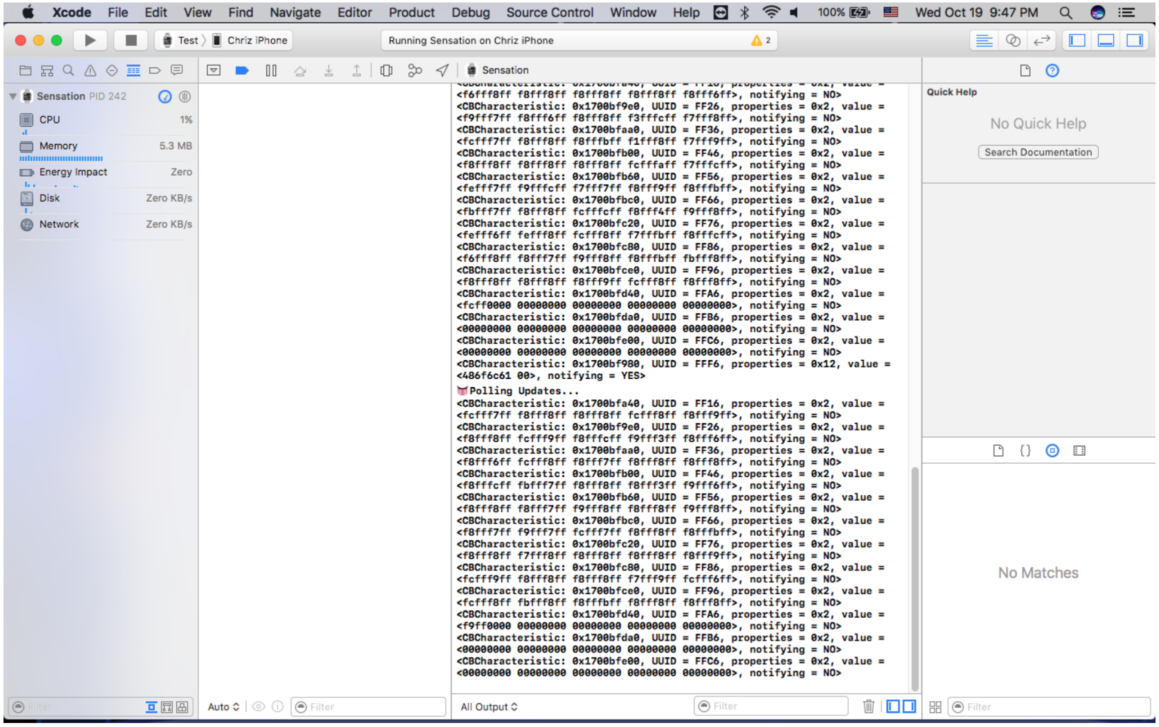
Figure 33 - FIFO Read of ADXL362

We have also successfully extract the data using Bluetooth LE.

We have created many characteristics to store the data fetched from the FIFO buffer of ADXL362. Each characteristic has 20 bytes. Totally 13 characteristics are created (FF16, FF26, FF36, FF46, FF56, FF66, FF76, FF86, FF96, FFA6, FFB6, FFC6).

Then, we create a characteristic with “Notify” and register notification with iOS Swift Code. When notification is received, XCode can read all characteristic once to take the numbers.

Finally, all data inside the RAM of MCU can be extracted using Bluetooth LE:



```
<CBCharacteristic: 0x1700bfa40, UUID = FF16, properties = 0x2, value =
<f6fff8ff f8fff8ff f8fff8ff f8fff8ff f8fff6ff>, notifying = NO>
<CBCharacteristic: 0x1700bf9e0, UUID = FF26, properties = 0x2, value =
<f9fff7ff f8fff6ff f8fff8ff f3fffcff f7fff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfaa0, UUID = FF36, properties = 0x2, value =
<fcfff7ff f8fff8ff f8fffbff f1fff8ff f7fff9ff>, notifying = NO>
<CBCharacteristic: 0x1700bfb00, UUID = FF46, properties = 0x2, value =
<f8fff8ff f8fff8ff f8fff8ff fcfffaff f7fffcff>, notifying = NO>
<CBCharacteristic: 0x1700bfb60, UUID = FF56, properties = 0x2, value =
<fefff7ff f9fffcff f7fff7ff f8fff9ff f8fffbff>, notifying = NO>
<CBCharacteristic: 0x1700bfbc0, UUID = FF66, properties = 0x2, value =
<fbfff7ff f8fff8ff fcfffcff f8fff4ff f9fff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfc20, UUID = FF76, properties = 0x2, value =
<fefff6ff fefff8ff fcfff8ff f7fffbff f8fffcff>, notifying = NO>
<CBCharacteristic: 0x1700bfc80, UUID = FF86, properties = 0x2, value =
<f6fff8ff f8fff7ff f9fff8ff f8fffbff fbfff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfce0, UUID = FF96, properties = 0x2, value =
<f8fff8ff f8fff8ff f8fff9ff fcfff8ff f8fff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfd40, UUID = FFA6, properties = 0x2, value =
<fcff0000 00000000 00000000 00000000 00000000>, notifying = NO>
<CBCharacteristic: 0x1700bfda0, UUID = FFB6, properties = 0x2, value =
<00000000 00000000 00000000 00000000 00000000>, notifying = NO>
<CBCharacteristic: 0x1700bfe00, UUID = FFC6, properties = 0x2, value =
<00000000 00000000 00000000 00000000 00000000>, notifying = NO>
<CBCharacteristic: 0x1700bf980, UUID = FFF6, properties = 0x12, value =
<486f6c61 00>, notifying = YES>
👅Polling Updates...
<CBCharacteristic: 0x1700bfa40, UUID = FF16, properties = 0x2, value =
<fcfff7ff f8fff8ff f8fff8ff fcfff8ff f8fff9ff>, notifying = NO>
<CBCharacteristic: 0x1700bf9e0, UUID = FF26, properties = 0x2, value =
<f8fff8ff fcfff9ff f8fffcff f9fff3ff f8fff6ff>, notifying = NO>
<CBCharacteristic: 0x1700bfaa0, UUID = FF36, properties = 0x2, value =
<f8fff6ff fcfff8ff f8fff7ff f8fff8ff f8fff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfb00, UUID = FF46, properties = 0x2, value =
<f8fffcff fbfff7ff f8fff8ff f8fff3ff f9fff6ff>, notifying = NO>
<CBCharacteristic: 0x1700bfb60, UUID = FF56, properties = 0x2, value =
<f8fff8ff f8fff7ff f9fff8ff f8fff8ff f9fff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfbc0, UUID = FF66, properties = 0x2, value =
<f8fff7ff f9fff7ff fcfff7ff f8fff8ff f8fffbff>, notifying = NO>
<CBCharacteristic: 0x1700bfc20, UUID = FF76, properties = 0x2, value =
<f8fff8ff f7fff8ff f8fff8ff f8fff8ff f8fff9ff>, notifying = NO>
<CBCharacteristic: 0x1700bfc80, UUID = FF86, properties = 0x2, value =
<fcfff9ff f8fff8ff f8fff8ff f7fff9ff fcfff6ff>, notifying = NO>
<CBCharacteristic: 0x1700bfce0, UUID = FF96, properties = 0x2, value =
<fcfff8ff fbfff8ff f8fffbff f8fff8ff f8fff8ff>, notifying = NO>
<CBCharacteristic: 0x1700bfd40, UUID = FFA6, properties = 0x2, value =
<f9ff0000 00000000 00000000 00000000 00000000>, notifying = NO>
<CBCharacteristic: 0x1700bfda0, UUID = FFB6, properties = 0x2, value =
<00000000 00000000 00000000 00000000 00000000>, notifying = NO>
<CBCharacteristic: 0x1700bfe00, UUID = FFC6, properties = 0x2, value =
<00000000 00000000 00000000 00000000 00000000>, notifying = NO>
```

Figure 34 - Showing All Characteristics in Debug Window

After taking all numbers using XCode, we put them to excel to plot these graphs:

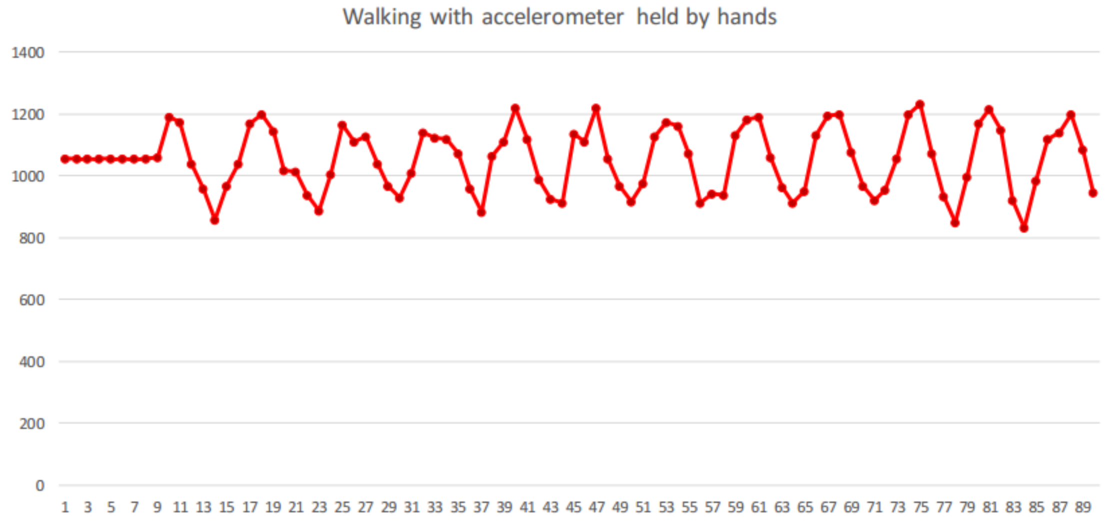


Figure 35 - Acceleration of X-axis: handheld steadily

The Y-axis of this graph is the x-axis acceleration of the accelerometer (in ms-2), the X-axis is the sample number. The sampling frequency is 12.5Hz, that means each sample is 80ms.

Another walking style will also produce another graph, but the pattern is similar:

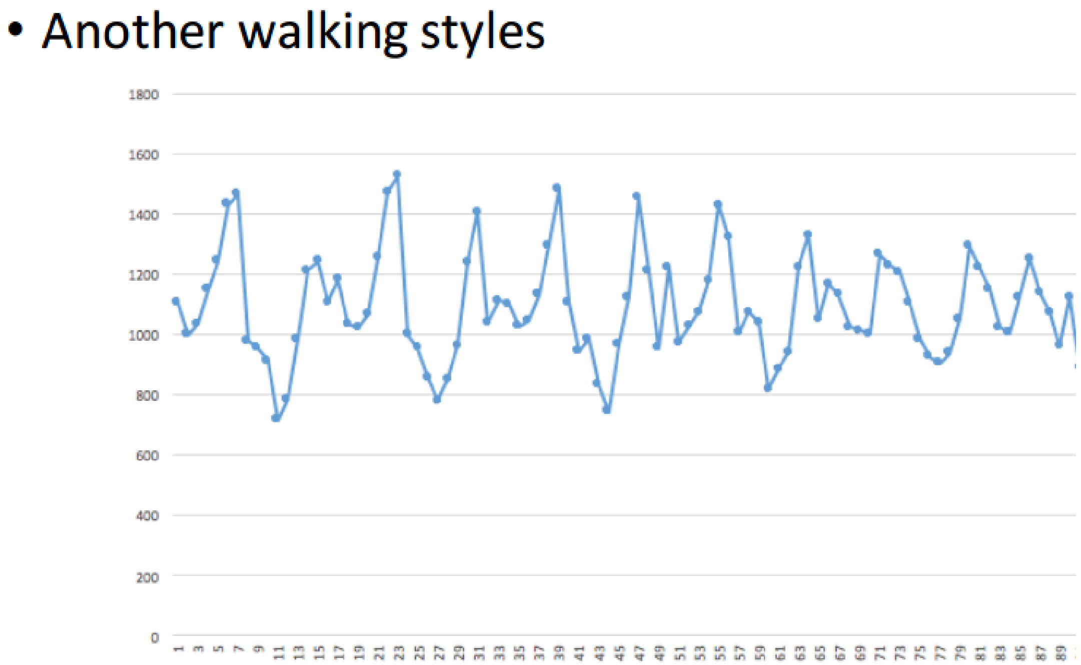

- Another walking styles

Figure 36 - Acceleration of X-axis: walk differently

The Y-axis of this graph is the x-axis acceleration of the accelerometer (in ms-2), the X-axis is the sample number. The sampling frequency is 12.5Hz, that means each sample is 80ms.

We also draw a 5-data moving average line on the same graph, it looks satisfactory for finding average of the walking data, and therefore, we use this way to find steps:

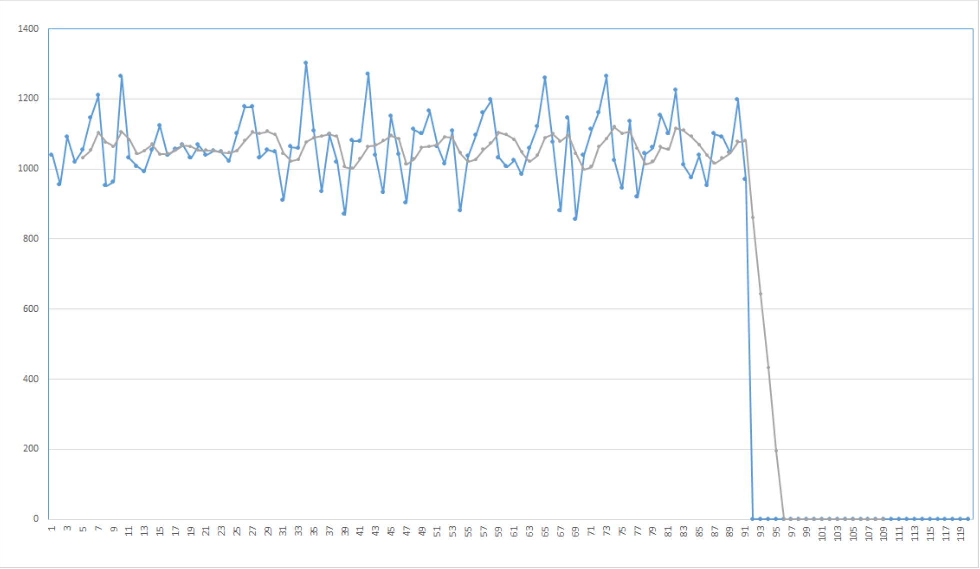
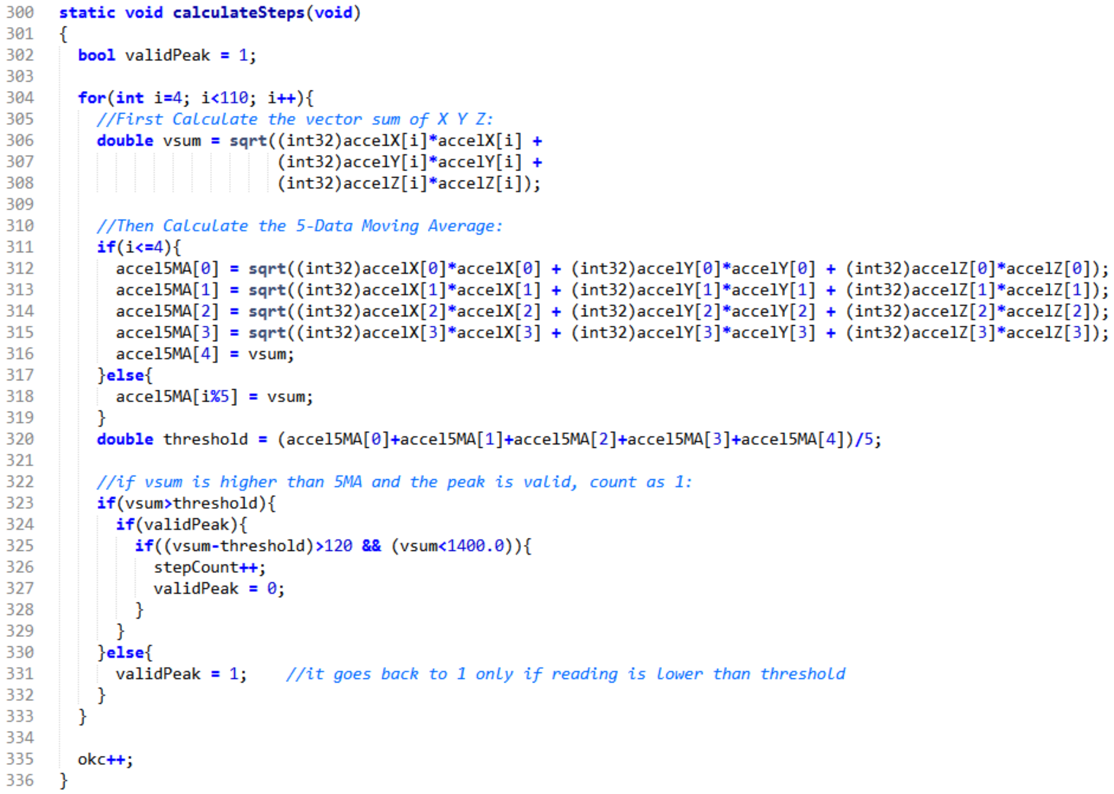


Figure 37 - 5-Data Moving Average

The testing results above is satisfactory. Therefore, we convert it into embedded C code as follows:



```c
static void calculateSteps(void)
{
  bool validPeak = 1;

  for(int i=4; i<110; i++){
    //First Calculate the vector sum of X Y Z:
    double vsum = sqrt((int32)accelX[i]*accelX[i] +
                       (int32)accelY[i]*accelY[i] +
                       (int32)accelZ[i]*accelZ[i]);

    //Then Calculate the 5-Data Moving Average:
    if(i<=4){
      accel5MA[0] = sqrt((int32)accelX[0]*accelX[0] + (int32)accelY[0]*accelY[0] + (int32)accelZ[0]*accelZ[0]);
      accel5MA[1] = sqrt((int32)accelX[1]*accelX[1] + (int32)accelY[1]*accelY[1] + (int32)accelZ[1]*accelZ[1]);
      accel5MA[2] = sqrt((int32)accelX[2]*accelX[2] + (int32)accelY[2]*accelY[2] + (int32)accelZ[2]*accelZ[2]);
      accel5MA[3] = sqrt((int32)accelX[3]*accelX[3] + (int32)accelY[3]*accelY[3] + (int32)accelZ[3]*accelZ[3]);
      accel5MA[4] = vsum;
    }else{
      accel5MA[i%5] = vsum;
    }
    double threshold = (accel5MA[0]+accel5MA[1]+accel5MA[2]+accel5MA[3]+accel5MA[4])/5;

    //if vsum is higher than 5MA and the peak is valid, count as 1:
    if(vsum>threshold){
      if(validPeak){
        if((vsum-threshold)>120 && (vsum<1400.0)){
          stepCount++;
          validPeak = 0;
        }
      }
    }else{
      validPeak = 1;    //it goes back to 1 only if reading is lower than threshold
    }
  }

  okc++;
}
```

Figure 38 - calculateStep: the step counting algorithm in Sensation Smart Watch

The calculateStep() will be run as a new task instead of running inside the interrupt function routine. This solves the problem that sqrt() takes long execution time. We performed experiments of wearing the prototype to walk from CYT Building in HKUST to Library in HKUST few times. It records around 300 steps each time. As the result is consistent every time and the number of steps count makes sense, we can conclude that the result of this step counter is satisfactory.

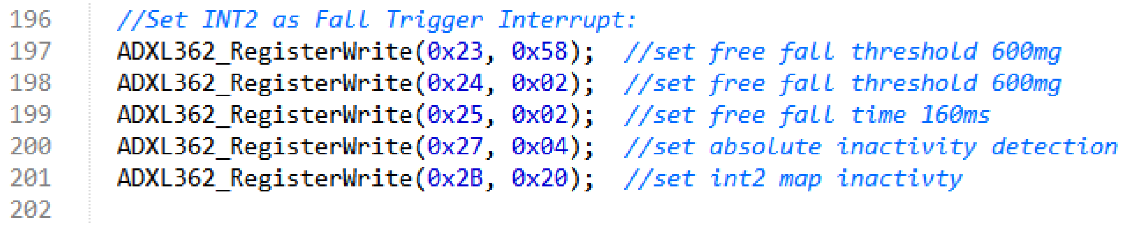
2.2.2.2 Fall Detection Implementation and Testing

Fall detection is implemented using a fixed interrupt threshold configured in ADXL362.

Normally, the resultant acceleration of a walking/standing/sitting person should be close to g (the gravity constant 9.81ms-2). If the resultant acceleration decreases significantly, it is believed that the watch has encountered a strong acceleration towards the ground, eliminating the gravitational acceleration. In this case, we can treat the user or the watch has fallen.

We set the threshold to be around 0.6g (600mg, g is the gravity constant), and the falling time is 160ms. In fact, the number 0.6 and 160 are arbitrary numbers found by our own experiments and these values could be further be reduced/increased to reduce false-alarms like waving hands and jumping.

The threshold configuration code is shown in the following graph:

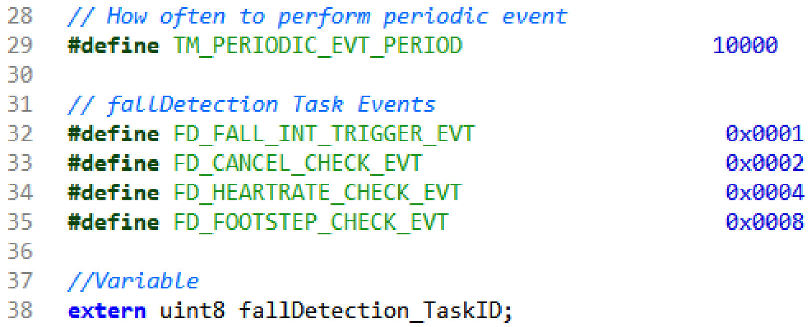


```
//Set INT2 as Fall Trigger Interrupt:
ADXL362_RegisterWrite(0x23, 0x58);  //set free fall threshold 600mg
ADXL362_RegisterWrite(0x24, 0x02);  //set free fall threshold 600mg
ADXL362_RegisterWrite(0x25, 0x02);  //set free fall time 160ms
ADXL362_RegisterWrite(0x27, 0x04);  //set absolute inactivity detection
ADXL362_RegisterWrite(0x2B, 0x20);  //set int2 map inactivty

```

Figure 39 - Fall Interrupt Parameters for ADXL362

To reduce false-alarms, we need to distinguish if the user is wearing the watch during the fall event. We measure the step count difference and heart rate 10 seconds after experiencing fall interrupt. If the step count does not change or the heart rate is invalid, it is very likely that the user has taken off the watch and thrown it accidentally.

There is a OSAL task, Fall Detection Task, inside our MCU responsible for this checking:



```
// How often to perform periodic event
#define TM_PERIODIC_EVT_PERIOD                   10000

// fallDetection Task Events
#define FD_FALL_INT_TRIGGER_EVT                   0x0001
#define FD_CANCEL_CHECK_EVT                       0x0002
#define FD_HEARTRATE_CHECK_EVT                    0x0004
#define FD_FOOTSTEP_CHECK_EVT                     0x0008

//Variable
extern uint8 fallDetection_TaskID;
```

Figure 40 - fallDetection.h - the OSAL fall detection checking task

We have performed several tests regarding the fall detection algorithm. It shows that if the heart rate reading is not affected by motion artifacts, the accuracy of the fall detection algorithm would be higher. Overall, the result of fall detection itself is satisfying because it does prevent false-alarms occurred by shaking hands and waving hands during our experiment.

2.2.2.3 User Interface Implementation and Testing

We are using OLEDs that has two different colors (yellow and cyan). In fact, the "two colors" OLED is controlled by a mono-color OLED controller, that means the first two page (page 0 and page 1) will always show yellow, other pages will always show cyan (page 2-7).

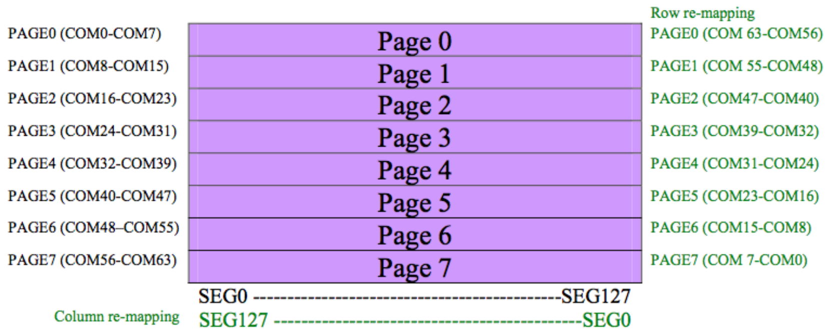
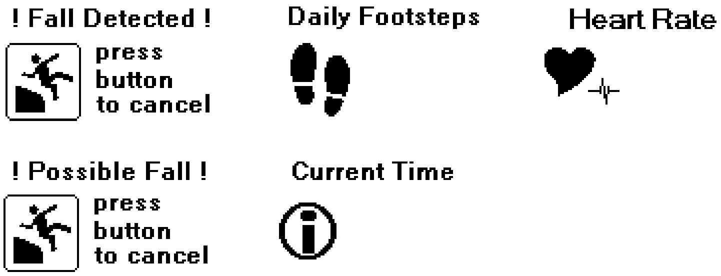
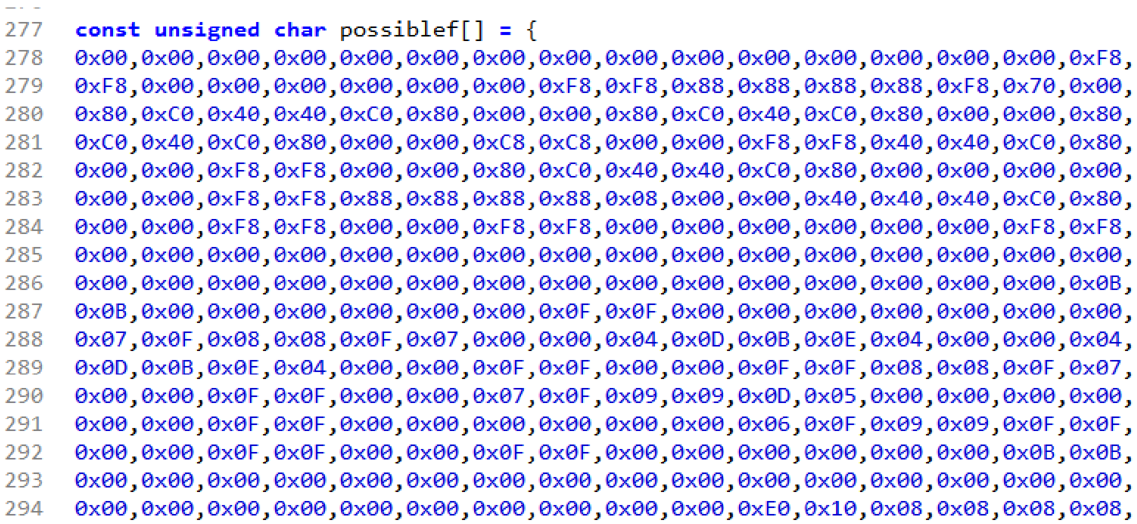


Figure 41 - Memory Architecture of SSD1306

We have prepared some bitmaps for different modes:



After using Zimo221, these bitmaps are converted from BMP into C arrays like this:



```c
const unsigned char possiblef[] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xF8,
0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0xF8,0xF8,0x88,0x88,0x88,0x88,0xF8,0x70,0x00,
0x80,0xC0,0x40,0x40,0xC0,0x80,0x00,0x00,0x80,0xC0,0x40,0xC0,0x80,0x00,0x00,0x80,
0xC0,0x40,0xC0,0x80,0x00,0x00,0xC8,0xC8,0x00,0x00,0xF8,0xF8,0x40,0x40,0xC0,0x80,
0x00,0x00,0xF8,0xF8,0x00,0x00,0x80,0xC0,0x40,0x40,0xC0,0x80,0x00,0x00,0x00,0x00,
0x00,0x00,0xF8,0xF8,0x88,0x88,0x88,0x88,0x08,0x00,0x00,0x40,0x40,0x40,0xC0,0x80,
0x00,0x00,0xF8,0xF8,0x00,0x00,0xF8,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0xF8,0xF8,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0B,
0x0B,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x07,0x0F,0x08,0x08,0x0F,0x07,0x00,0x00,0x04,0x0D,0x0B,0x0E,0x04,0x00,0x00,0x04,
0x0D,0x0B,0x0E,0x04,0x00,0x00,0x0F,0x0F,0x00,0x00,0x0F,0x0F,0x08,0x08,0x0F,0x07,
0x00,0x00,0x0F,0x0F,0x00,0x00,0x07,0x0F,0x09,0x09,0x0D,0x05,0x00,0x00,0x00,0x00,
0x00,0x00,0x0F,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,0x06,0x0F,0x09,0x09,0x0F,0x0F,
0x00,0x00,0x0F,0x0F,0x00,0x00,0x0F,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,0x0B,0x0B,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xE0,0x10,0x08,0x08,0x08,0x08,
```

These are the some of the testing results of the OLED screen displays, it shows that the state machine code and the bitmaps can be displayed successfully:

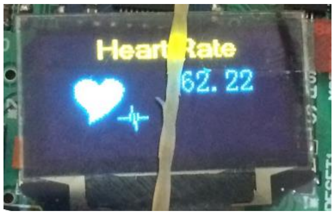


Figure 42 - Heart Rate OLED

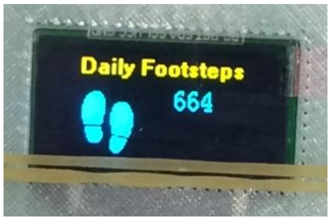


Figure 43 - Daily Footstep OLED

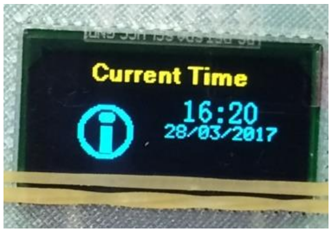


Figure 44 - Current Time OLED

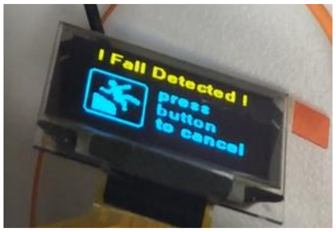


Figure 45 - Fall Detected OLED

To summarize, sensors software programming (especially in pedometer, fall detection and user interface designs) has been implemented and tested with satisfactory results.

Section 2.3: Server Programming

2.3.1 Design

2.3.1.1 Backend Design

PHP and MySQL is selected to be the programming language because of its popularity. In fact, most of the cloud-based services are compatible with PHP and MySQL. The structure of the whole system should be as follows:

1. Authentication is required for logging in the system.
   Only authorized persons can view the data inside our database.

2. Smartphones can send information to the server using a simple HTTP function.

3. Smartphones can know if the data insertion is successful or not by reading the content returned by HTTP. It should be formatted with JSON such that it would be easy for smartphone applications to parse it.

The detailed workflow of the server program would be:

Login / Logout Workflow (mostly done by Raymond):

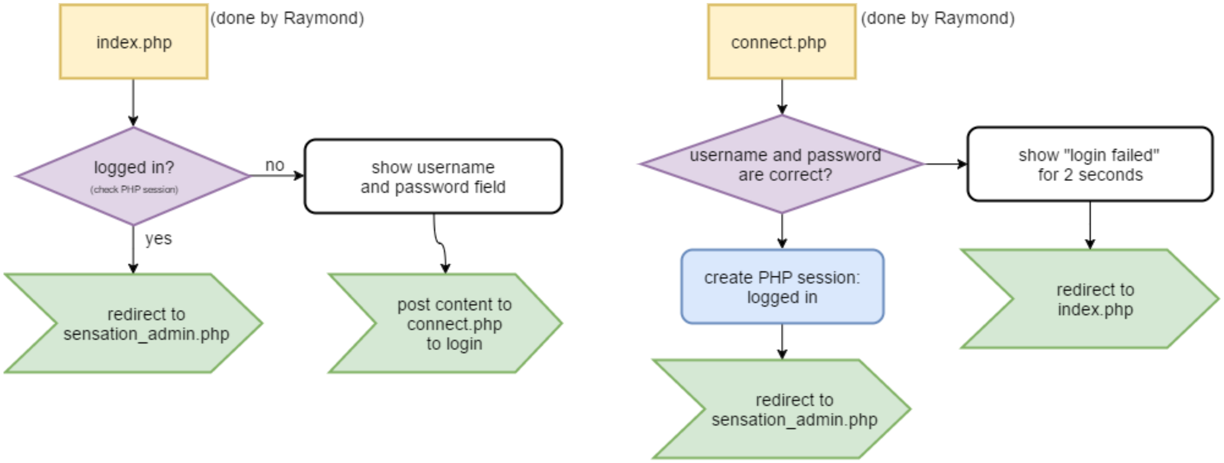

index.php
(done by Raymond)
logged in?
(check PHP session)
no
show username and password field
yes
redirect to sensation_admin.php
post content to connect.php to login
connect.php
(done by Raymond)
username and password are correct?
show "login failed" for 2 seconds
create PHP session: logged in
redirect to index.php
redirect to sensation_admin.php

Figure 46 - Login Workflow of Server

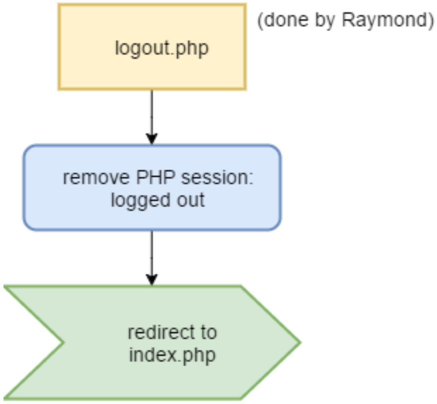


Figure 47- Logout Workflow of Server

Fall Record Query and Processing (by me):

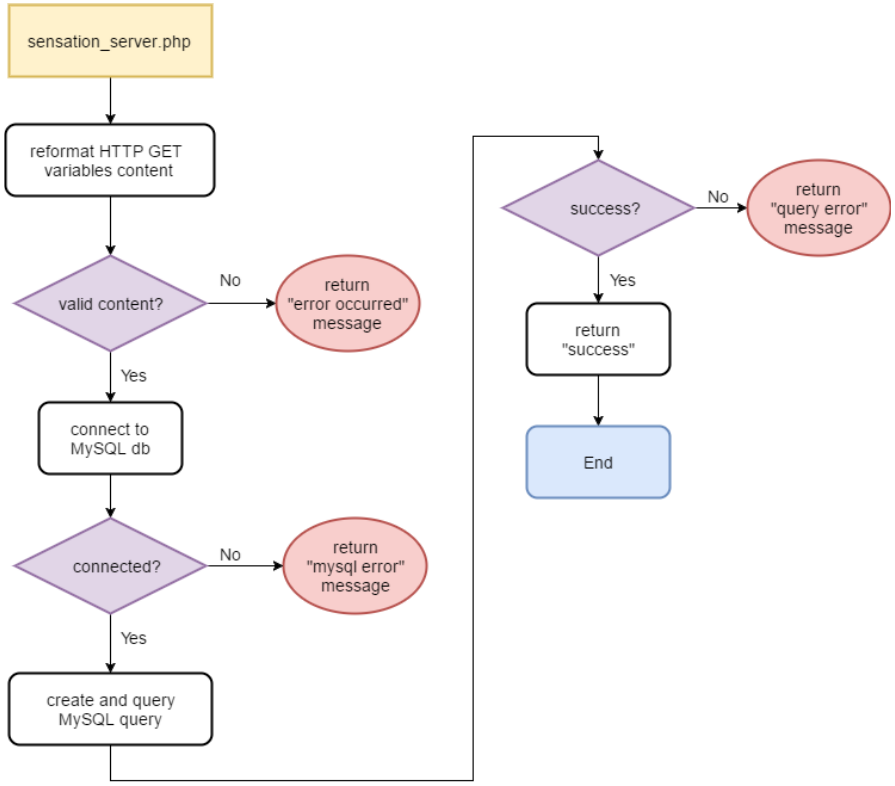

sensation_server.php
reformat HTTP GET variables content
valid content?
No
return "error occurred" message
Yes
connect to MySQL db
connected?
No
return "mysql error" message
Yes
create and query MySQL query
success?
No
return "query error" message
Yes
return "success"
End

Figure 48 - Insert new record to database in Server

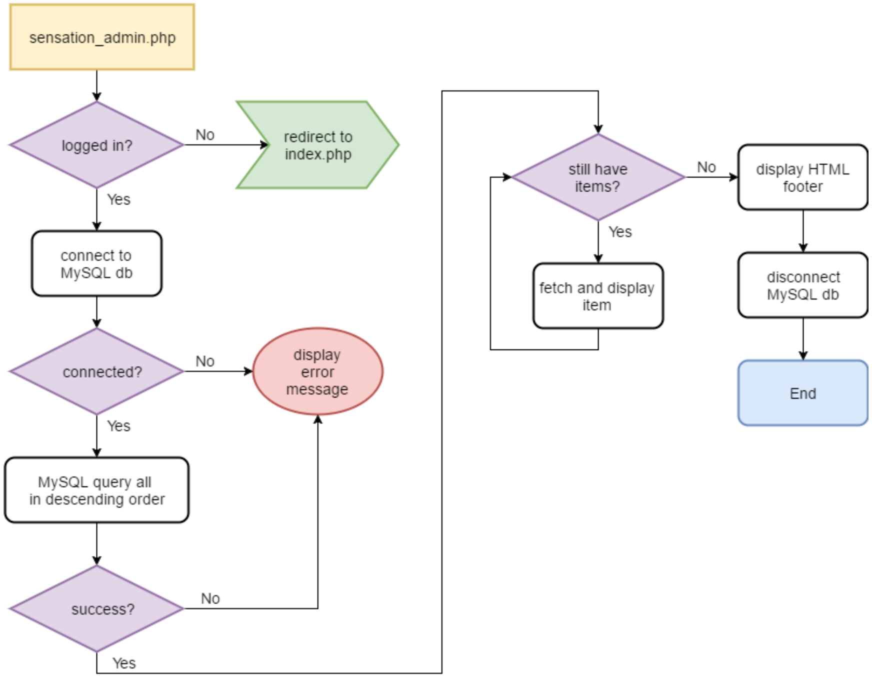


Figure 49 - View all records from database in Server

Following the flowchart, there will be totally 5 PHP files generated.

The SQL Table structure:

```
CREATE TABLE fallrecord (
  recordid int(11) PRIMARY KEY ASC NOT NULL,
  username varchar(6) NOT NULL,
  unixtime int(11) NOT NULL,
  latitude varchar(64),
  longitude varchar(64)
);
```

2.3.1.2 Frontend Design

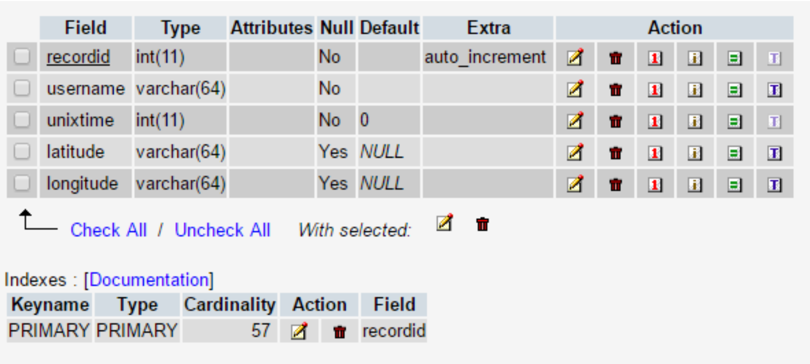
Some requirements of the frontend UI designs:

1. Connect to Google Maps – user can click the location record to show the maps.
2. Simple – the UI should be easy to be understood

Number 2 can be done with Bootstrap, while number 1 can be done without calling Google Maps API. It will further be explained and demonstrated in the "implementation and testing" part.

2.3.2 Implementation and Testing

Following the flowchart and the MySQL table creation query in the design part, we have been successfully creating the SQL table and the whole backend system:

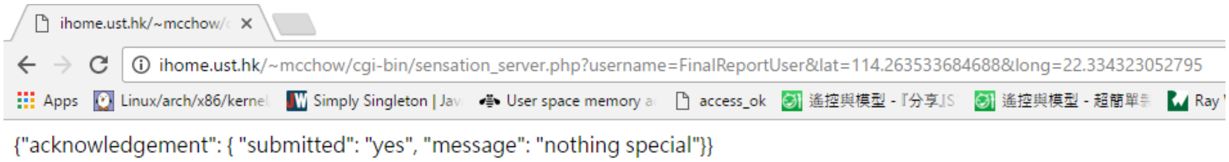


	Field	Type	Attributes	Null	Default	Extra	Action
□	recordid	int(11)		No		auto_increment	
□	username	varchar(64)		No			
□	unixtime	int(11)		No	0		
□	latitude	varchar(64)		Yes	*NULL*		
□	longitude	varchar(64)		Yes	*NULL*		

Indexes : [Documentation]

Keyname	Type	Cardinality	Action	Field
PRIMARY	PRIMARY	57		recordid

Figure 50 - SQL Database Structure - using phpMyAdmin to preview (Backend)



```
{"acknowledgement": { "submitted": "yes", "message": "nothing special"}}
```

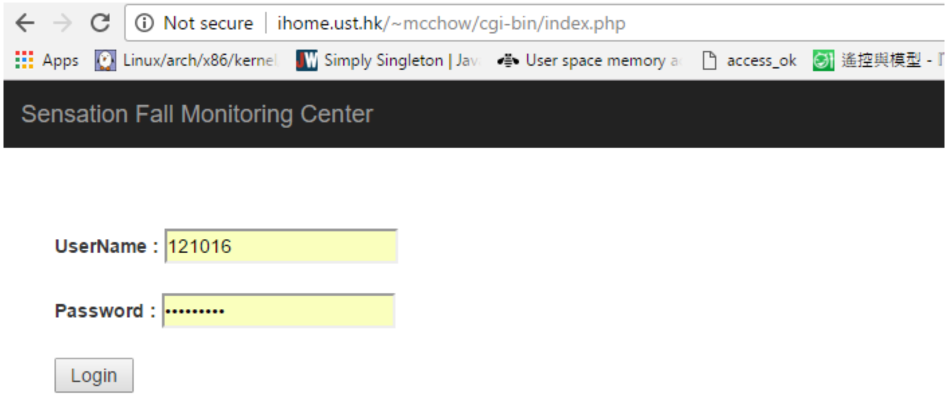
Figure 51 - sensation_server.php Implementation (Backend)

Smartphones can make a simple GET request like this to submit all information:

```
GET /~mcchow/cgi-
bin/sensation_server.php?username=<username>&lat=<latitude>&lng=<longi
tude> HTTP/1.1
Host: ihome.ust.hk
```

If all inputs are valid and username is not null, a new entry will be inserted to the SQL database automatically. Our server also prevents SQL injection, so users should have no read access to the database without password.

The frontend page for viewing all records are also created successfully using Bootstrap:

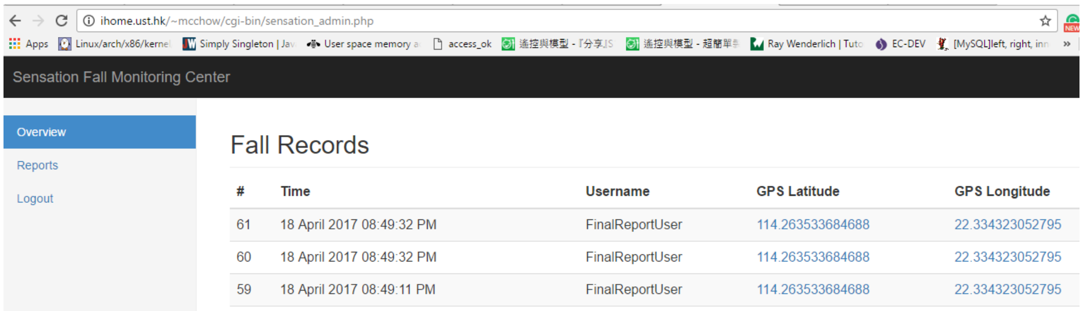


Sensation Fall Monitoring Center

UserName : 121016

Password : •••••••••

Figure 52 - Login Page of Admin Frontend



Sensation Fall Monitoring Center

Fall Records

#	Time	Username	GPS Latitude	GPS Longitude
61	18 April 2017 08:49:32 PM	FinalReportUser	114.263533684688	22.334323052795
60	18 April 2017 08:49:32 PM	FinalReportUser	114.263533684688	22.334323052795
59	18 April 2017 08:49:11 PM	FinalReportUser	114.263533684688	22.334323052795

Figure 53 - Admin Frontend of Fall Records

When the user clicks the GPS Latitude or GPS Longitude, it will automatically redirect users to Google Maps. It helps users understand what is the exact location of the latitude and the longitude in the record:

In fact, Google Maps provided a convenient way to find the latitude and longitude location without using Google API. We can simply use the following links, and we have also generated this hyperlink in our frontend:

```
https://www.google.com/maps?q=<Latitude>,<Longitude>
```

To summarize, the testing and implementation above shows that the server programming result is satisfactory. The server can be used safely and stably.

Section 2.4: iOS Programming

2.4.1 Design

Using our Sensation Smart Watch iOS application, users should be able to synchronize the time of the watch, check heart rate, step counts and calculate the calories burnt automatically. The app should also be able to send a fall notification if the watch detects falling.

The design of this application follows the flowchart shown in Figure 9 (which is in chapter 1.3.4). There are some extra requirements of this application:

- The app should scan all Bluetooth LE devices nearby. However, only Sensation Smart Watch would be highlighted. Other devices should show a question mark, indicating our app may not be compatible with that device.

- If the user connects an incompatible device with our app, we should show him/her a warning. It ensures that they know that they are connecting a device which may be incompatible.

- In some special case, our device may not send device name properly. Although it is an “unknown device” or “incompatible device”, we should still allow it to be connected.

- “Calories Burnt” is using the equation as same as the “calories burnt” in Android version of Sensation Smart Watch application.

- When a fall notification is sent, the app should trigger a notification. It helps users to know that the app is still running in background.

- There should be a page to allow users change their username (fall record upload) and weight (for calories calculation).

- Multi-language support should be provided. It helps broaden the audience of Sensation Smart Watch. For demonstration, English, Chinese and Spanish should be provided.

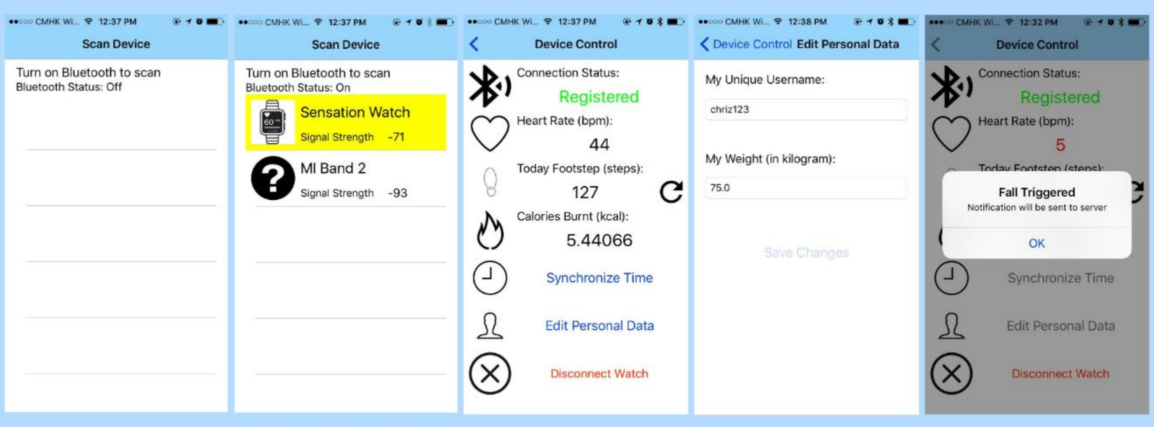
- Time Synchronization should be able to adapt to different time-zones configured in iOS.

- Same color style (sky blue) should be used in the app to make the app UI to be consistent with our Android version.

In the next chapter of “Implementation and Testing”, the details of implementing all requirements above will be mentioned. Also, there will be some screenshots for readers to verify the implementation results.

2.4.2 Implementation and Testing

We have successfully developed and demonstrated the iOS App. The following figure is the screenshots showing that the app workflow is implementing the flowchart in Figure 9:



Scan Device

Turn on Bluetooth to scan
Bluetooth Status: Off

Scan Device

Turn on Bluetooth to scan
Bluetooth Status: On

Sensation Watch
Signal Strength -71

MI Band 2
Signal Strength -93

Device Control

Connection Status: Registered
Heart Rate (bpm): 44
Today Footstep (steps): 127
Calories Burnt (kcal): 5.44066

Device Control Edit Personal Data

My Unique Username: chriz123
My Weight (in kilogram): 75.0

Device Control

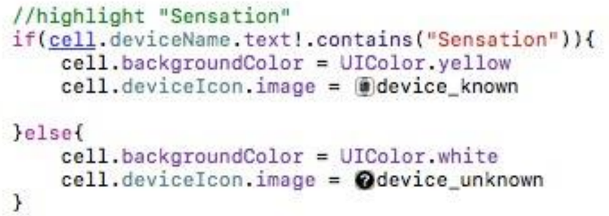
Connection Status: Registered
Heart Rate (bpm): 5

Fall Triggered
Notification will be sent to server

Figure 54 - Workflow Screenshot of iOS App

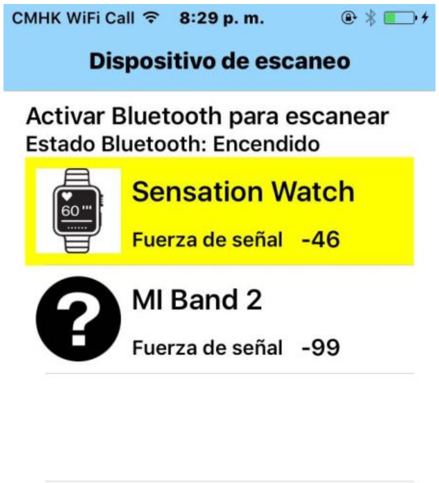
There are the detailed implementations of the extra requirements stated in “Design” chapter:

- **Highlight Sensation Smart Watch**
  This function can be implemented by highlighting the device which has the name “Sensation”. The swift code of that cell is:



```
//highlight "Sensation"
if(cell.deviceName.text!.contains("Sensation")){
    cell.backgroundColor = UIColor.yellow
    cell.deviceIcon.image = device_known

}else{
    cell.backgroundColor = UIColor.white
    cell.deviceIcon.image = device_unknown
}
```

Testing Screenshot (in Spanish interface):



Dispositivo de escaneo

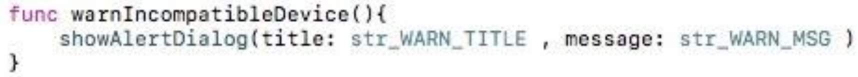
Activar Bluetooth para escanear
Estado Bluetooth: Encendido

Sensation Watch
Fuerza de señal -46

MI Band 2
Fuerza de señal -99

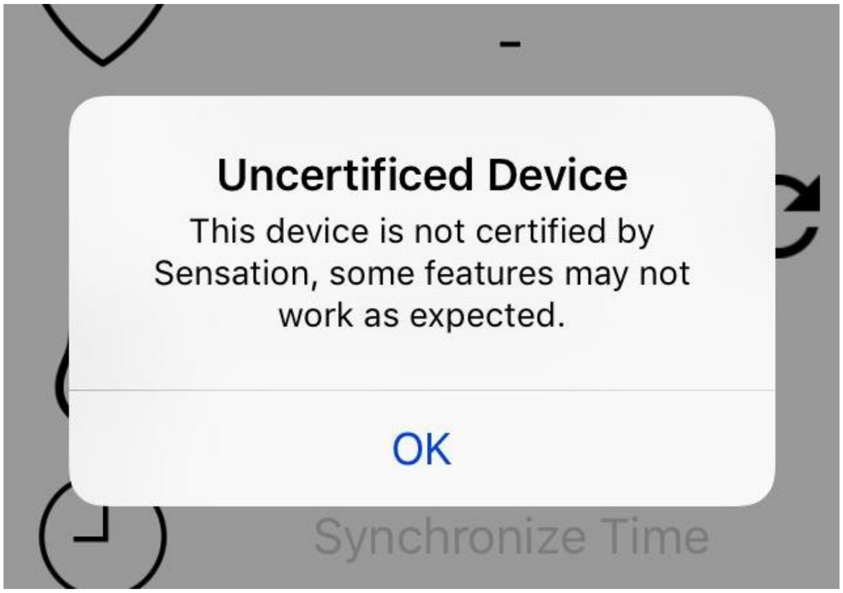
Figure 55 - Highlighted Sensation in the device list

- **Show incompatible prompt** if user connects to an incompatible device:
  Related Swift Code:



```
func warnIncompatibleDevice(){
    showAlertDialog(title: str_WARN_TITLE , message: str_WARN_MSG )
}
```

Testing Screenshot:

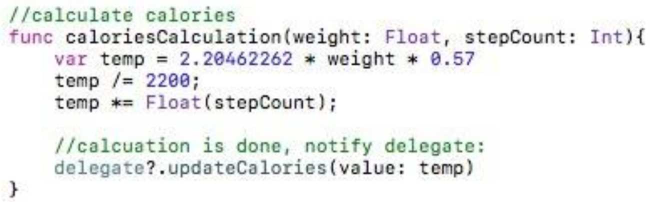


Uncertificed Device

This device is not certified by Sensation, some features may not work as expected.

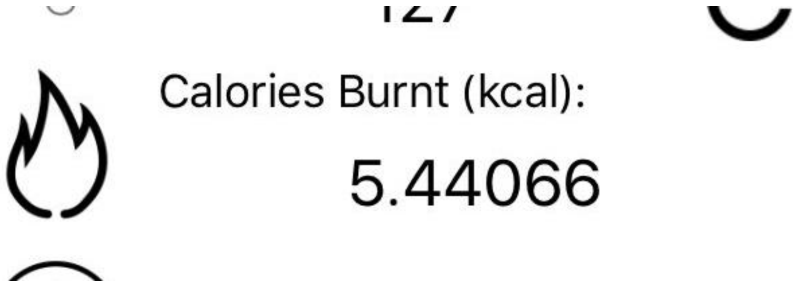
Figure 56 - Uncertified Device Warning

- **Calories Burnt Equation** from Android version of Sensation Smart Watch App
  Related Swift Code:



```
//calculate calories
func caloriesCalculation(weight: Float, stepCount: Int){
    var temp = 2.20462262 * weight * 0.57
    temp /= 2200;
    temp *= Float(stepCount);

    //calcuation is done, notify delegate:
    delegate?.updateCalories(value: temp)
}
```

Testing Screenshot:

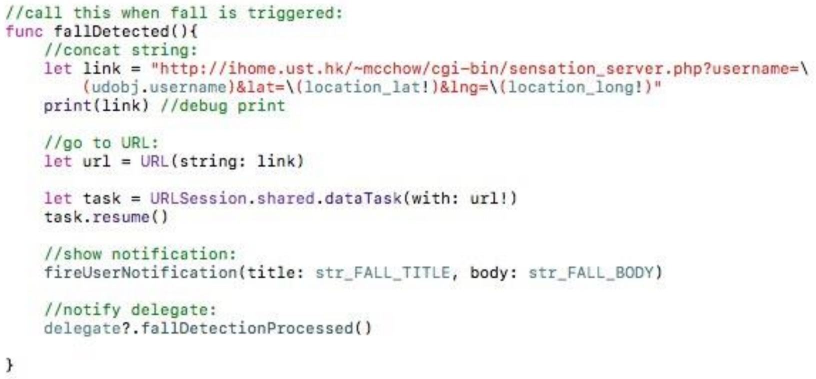


Calories Burnt (kcal):

5.44066

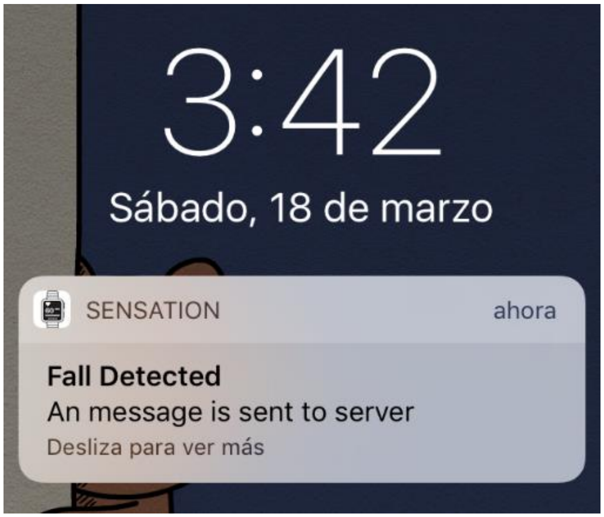
Figure 57 - Calories Calculation on UI

- Triggering notifications after receiving fall notification
  Related Swift Code:



```
//call this when fall is triggered:
func fallDetected(){
    //concat string:
    let link = "http://ihome.ust.hk/~mcchow/cgi-bin/sensation_server.php?username=\
        (udobj.username)&lat=\(location_lat!)&lng=\(location_long!)"
    print(link) //debug print

    //go to URL:
    let url = URL(string: link)

    let task = URLSession.shared.dataTask(with: url!)
    task.resume()

    //show notification:
    fireUserNotification(title: str_FALL_TITLE, body: str_FALL_BODY)

    //notify delegate:
    delegate?.fallDetectionProcessed()

}
```

Testing Screenshot:



3:42

Sábado, 18 de marzo

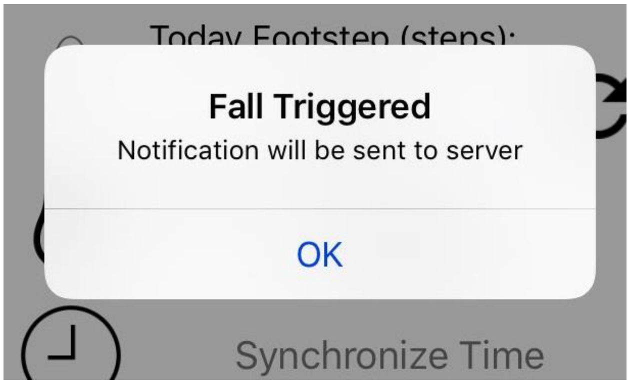
SENSATION ahora

Fall Detected

An message is sent to server

Desliza para ver más

Figure 58 - Background Notification (iOS)



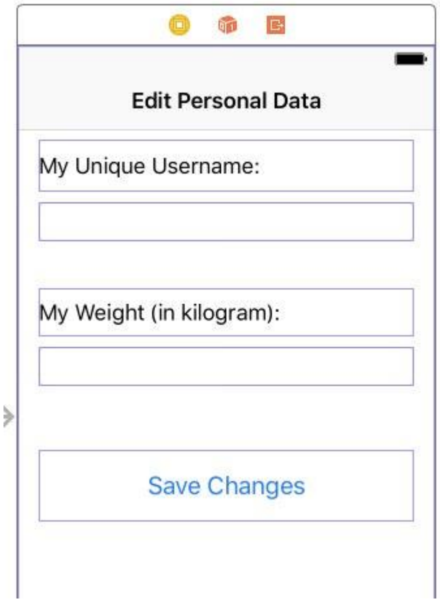
Today Footstep (steps):

Fall Triggered

Notification will be sent to server

Figure 59 - Foreground Alert Dialog (iOS)

- **A Page Specialized for changing username and weight**
  Related Storyboard and Screenshot:



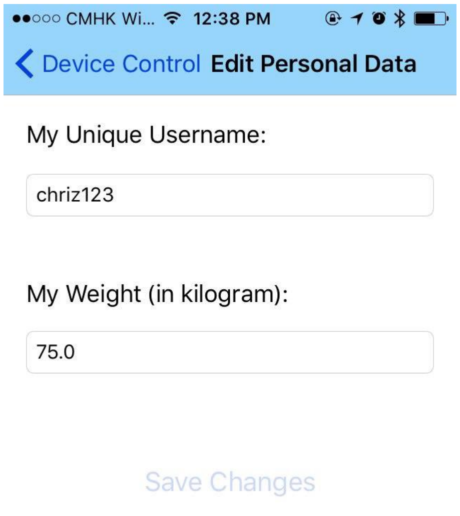
Edit Personal Data

My Unique Username:

My Weight (in kilogram):

Save Changes

Figure 60- Storyboard of "Edit Personal Data"



Edit Personal Data

My Unique Username:

chriz123

My Weight (in kilogram):

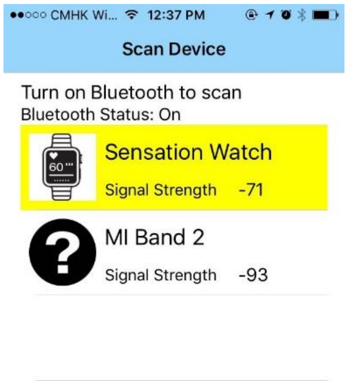
75.0

Save Changes

Figure 61 - Actual Screen of "Edit Personal Data"

- **Multi-Language Supports (App Internationalization)**
  This iOS app successfully provided three languages support (English, Traditional Chinese and Spanish) using `Localizable.Strings` in XCode.

  Testing Screenshots:



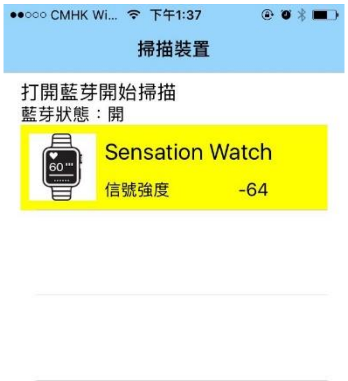
Scan Device

Turn on Bluetooth to scan
Bluetooth Status: On

Sensation Watch
Signal Strength -71

MI Band 2
Signal Strength -93

Figure 62 - English iOS UI

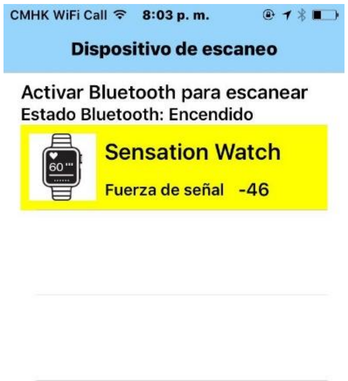


掃描裝置

打開藍芽開始掃描
藍芽狀態：開

Sensation Watch
信號強度 -64

Figure 63 - Chinese iOS UI



Dispositivo de escaneo

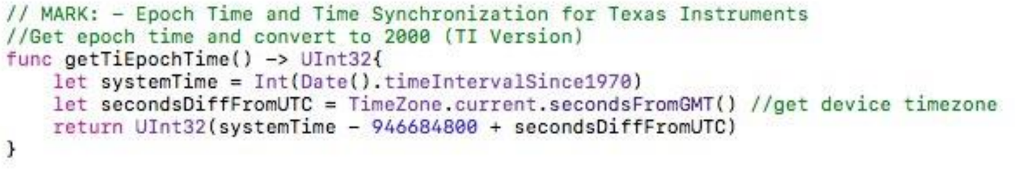
Activar Bluetooth para escanear
Estado Bluetooth: Encendido

Sensation Watch
Fuerza de señal -46

Figure 64 - Spanish iOS UI

- Automatic Time-zone adaption

  Thanks to "**`Timezone.current`**" variable in Swift, this iOS app successfully achieved automatic time-zone adaption. The related Swift code:



```
// MARK: - Epoch Time and Time Synchronization for Texas Instruments
//Get epoch time and convert to 2000 (TI Version)
func getTiEpochTime() -> UInt32{
    let systemTime = Int(Date().timeIntervalSince1970)
    let secondsDiffFromUTC = TimeZone.current.secondsFromGMT() //get device timezone
    return UInt32(systemTime - 946684800 + secondsDiffFromUTC)
}
```

To summarize, This iOS App sub-project has successfully achieved all design requirements. It can be used to connect with our Sensation Smart Watch continuously and smoothly during our experiment. Also, as the language of the interface would change according to the system language, as well as the time synchronization will work in different countries, it would be extremely convenient for international travelers or people using different languages to use our watch comfortably.

Chapter 3 – Project Results and Evaluation

Section 3.1: Individual Sub-Projects Results and Evaluation

Four sub-projects have been completed individually. The following paragraphs will be the summary of what the goals are the sub-projects, and what we have successfully achieved in these projects.

3.1.1 Evaluation of Sub-Project 1 – Circuit Design

The aims of this sub-project are to design and fabricate a customized PCB board that is fully functional, compact and low-noise. By using power planes, ground planes and open window and other different strategies, we have successfully overcome all technical challenges.

Firstly, although we have many components such as a step-up converter, inductors and schottky diodes, the size of our final prototype PCB board can still be managed to exactly 41mm*41mm. The size is compact enough for most people to wear. It can also be fitted tightly on the 3D-printed watch case.

Secondly, all noise sensitive components are fully functional and normal. There are no extra noises produced even when the buzzer is on or the USB charging is connected. The extra voltage regulators and capacitors in our circuit has filtered much noises.

Thirdly, using appropriate lithium ion battery, the battery life of our circuit board can last for longer than 24 hours in experiments. The average power consumption of the circuit board is around 12mA. For a 400mAh battery, theoretically it could last for 33 hours. Besides, user can still use the device while charging. It also means that this circuit is suitable for continuous monitoring.

In conclusion, all testing results shows that this sub-project has been completed successfully.

3.1.2 Evaluation of Sub-Project 2 – Sensors Software Programming

The aims of this sub-project are to program the accelerometer properly such that it could perform step counting and fall detection. Also, we need to program a state-machine to make the button work. The button should be used to switch different pages, showing different kinds of information.

This sub-project has demonstrated that accelerometer can be used to measure different kinds of body movement which is not simply steps count. In this sub-project, we have successfully demonstrated that an accelerometer can be used to measure both step counts and fallings of the user. Using a static threshold configured in the accelerometer, with combined use of the heart-rate sensor, we could distinguish if the user has fallen or he/she has taken off the watch.

On the other hand, this sub-project has also demonstrated a very simple yet intuitive user interface – the state machine. User can use this button to switch between different pages or turn off the screen. Therefore, user can treat our Sensation Smart Watch as a usual digital watch.

In conclusion, this sub-project has demonstrated the mixed use of sensors data. Thanks to the intuitive user interface design, this watch can be used as both a digital watch, a simple activity tracker and a continuously monitoring system.

3.1.3 Evaluation of Sub-Project 3 – Server Programming

The aim of this sub-project is to create a server program that receives, store and show all fall records. To protect the privacy of all users being monitored, a login-logout mechanism should also be provided.

Regarding the implementation of this sub-project, PHP and MySQL are used as the server-side programming language and the database server language. With mixed use of PHP Session, it shows that login and logout mechanism can be implemented successfully. On the other hand, the usage of Bootstrap CSS has also minimized the effort in layout programming and maximized the usability and readability of the webpage.

On the other hand, a very simple way of connecting Google Maps – using hyperlink has made the website become easier to use. Although this is not the best way to interact with Google Maps, it is still usable and make the latitude and longitude information on the webpage become more meaningful to users as they can click on it to visualize the location on Maps.

However, there is a little issue in the way of transmitting information – this sub-project is using "HTTP GET" to upload information. Although HTTP GET method can be used to transmit information, it is not the most proper way of doing it because it may create confusion to web developers – the "GET" method should refer to "get information", not to "post data to the website". Therefore, in terms of improvements in coding, "POST" method should be considered, even though it is more difficult to use.

In conclusion, we have achieved the objectives of creating an online server program that receives, store and show fall records using PHP and MySQL. It is also linked to Google Maps such that users can locate the exact location of the fall event. Although some coding styles could be improved, the system is fully functional and be able to respond at real-time.

3.1.4 Evaluation of Sub-Project 4 – iOS App Programming

The aim of this sub-project is to create a native iOS app that connects, transmits and receives data between the Sensation Smart Watch and iOS devices.

To create a native iOS application, all codes in this application are written in Swift 3.0, the new programming language invented and promoted by Apple Inc. In this sub-project, Swift and CoreBluetooth Library are used heavily to deal with all scenarios of Bluetooth connection and Bluetooth state. Apple has provided many resources and explanations about its library, therefore most of the problems in coding can be resolved by reading the documents carefully.

On the other hand, testing have been done to make sure that the app is running in background and it can send fall events to the server successfully.

All other extra requirements are done and well-tested. The interface of this application is also very simple and easy to use because there are only few buttons.

In conclusion, we have successfully achieved the goal of providing iOS platform support for Sensation Smart Watch by creating a native iOS app. This is also one of the main objectives of our project.

Section 3.2: Group Results and Evaluation

From hardware to software, from outer shell design to inner components arrangement, from backend server development to frontend web designs, by combining all different sub-projects, a complete platform of continuous monitoring wearable sensors system can be created. All specifications mentioned in the group project objectives are achieved successfully:

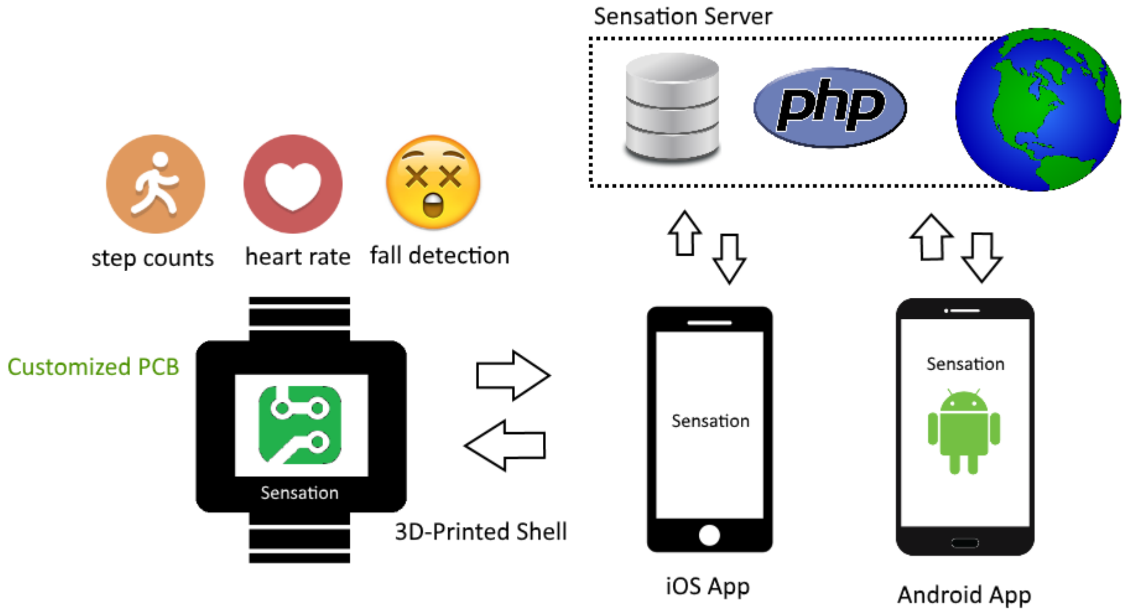


Figure 65 – Visualized Complete Project Objective Review

Firstly, we have created a new wearable sensors system monitoring user’s vital signs (heart rate, step count and fall alert) continuously and transmitting health data to smartphone using Bluetooth. Users can also interact with the device with OLED screen and a button. These functions are implemented with our customized PCB and sensors software.

Secondly, our system is a low-cost sensing systems intended for users that needs continuous monitoring such as elderly. Using all market-available components and 8051 microprocessor (CC2541), the component cost of our circuit is manageable and cost-effective. The following figure shows the complete circuit board of our system:

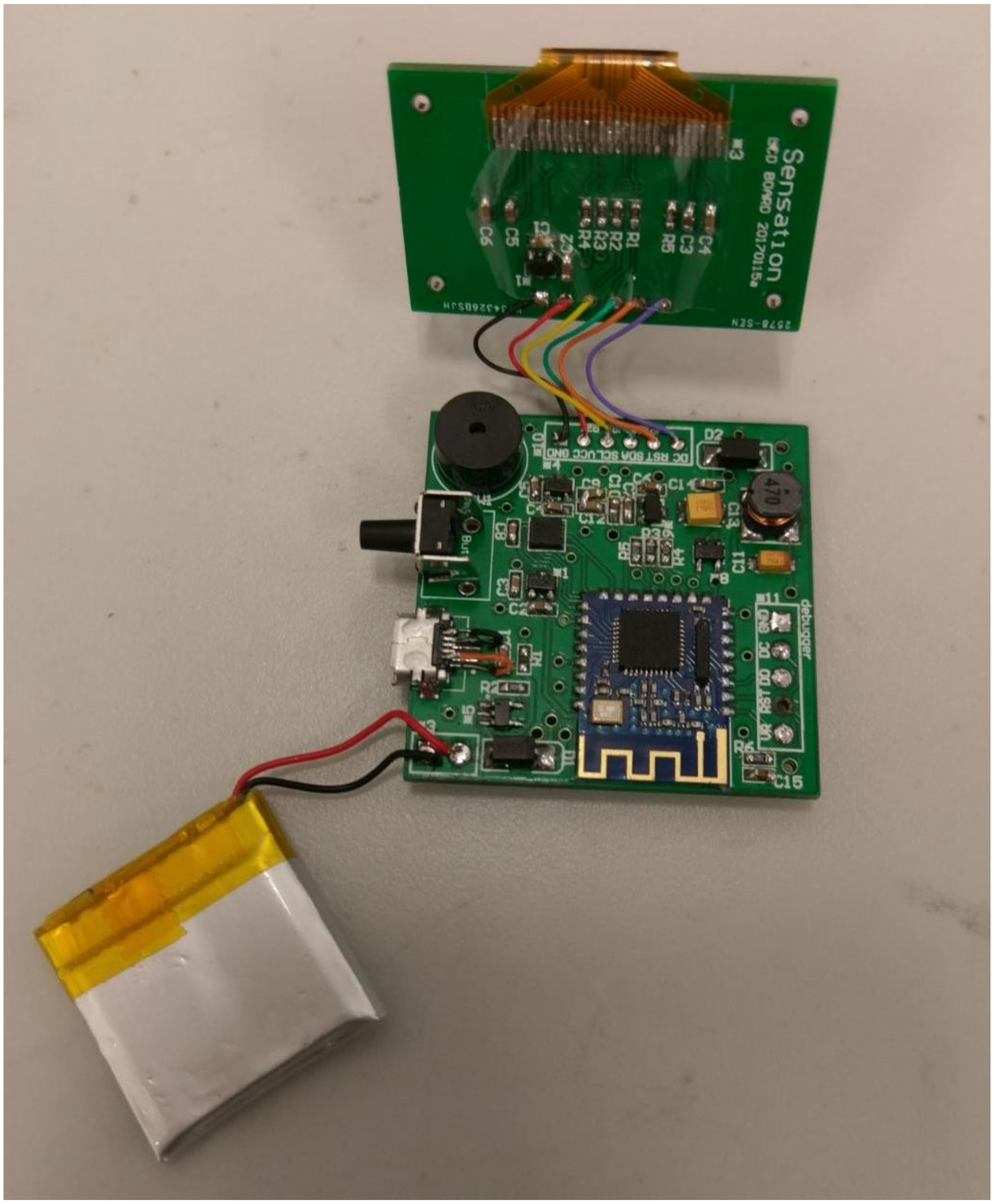
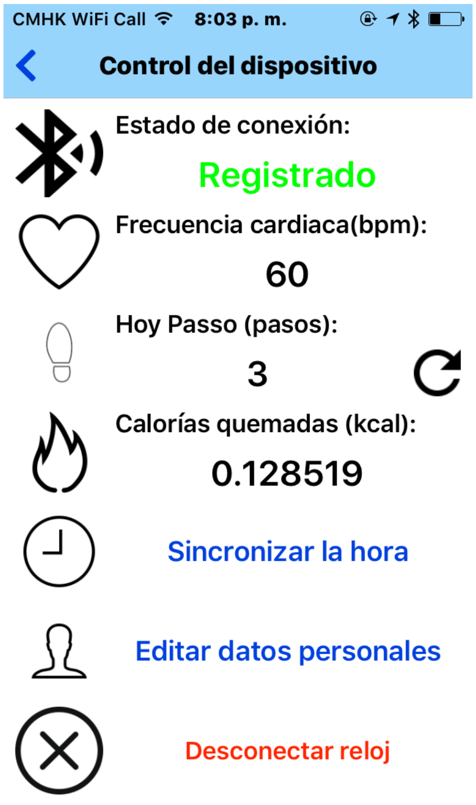


Figure 66 - Complete Circuit of Sensation Smart Watch

Thirdly, smartphone application for both iOS and Android are successfully developed to connect, synchronize, visualize health data and report fall location to a central server immediately when the watch detects that the user has fallen.



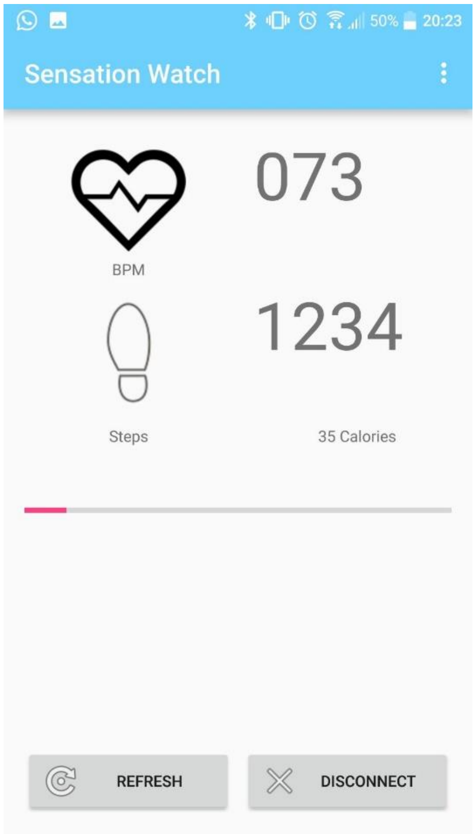
Estado de conexión:
Registrado

Frecuencia cardiaca(bpm):
60

Hoy Passo (pasos):
3

Calorías quemadas (kcal):
0.128519

Figure 67- iOS App Device Control Page

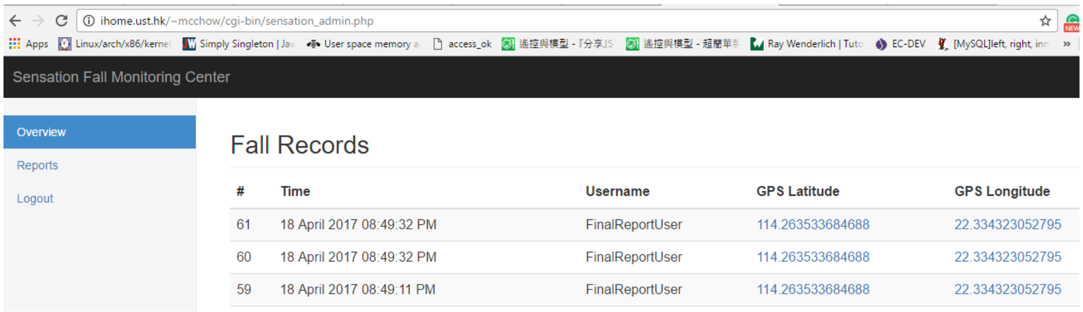


073
BPM

1234
Steps
35 Calories

Figure 68 - Android App Device Control Page

Fourthly, a cloud-based server with written with PHP and MySQL is also successfully developed. The server can store all fall records with username, time and location. A web interface connecting to the database is also created such that administrators or control center users can login to check all fall records of their patients.



Fall Records

#	Time	Username	GPS Latitude	GPS Longitude
61	18 April 2017 08:49:32 PM	FinalReportUser	114.263533684688	22.334323052795
60	18 April 2017 08:49:32 PM	FinalReportUser	114.263533684688	22.334323052795
59	18 April 2017 08:49:11 PM	FinalReportUser	114.263533684688	22.334323052795

Figure 69 - Admininstrative Panel of Sensation Fall Record Server

Fifthly, a 3D-printed case is also created to fit the circuit inside a beautifully crafted watch shell. Therefore, users can really treat our device as a smart watch:

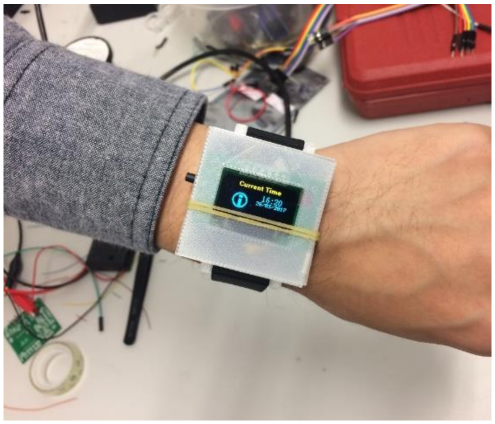


Figure 70 - 3D Printed Shell Prototype 1

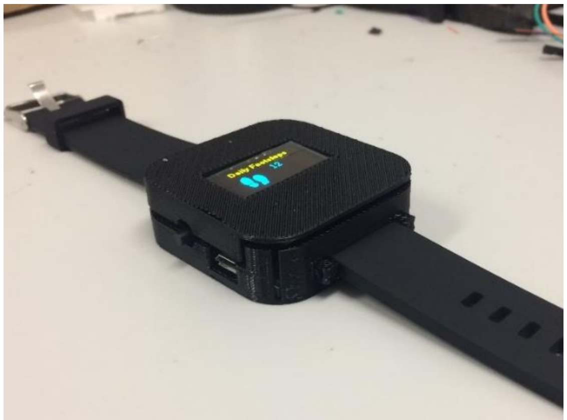


Figure 71 - 3D Printed Shell Prototype 2

Finally, by combining all sub-projects together, a complete wearable sensors system, Sensation Smart Watch, is developed successfully:

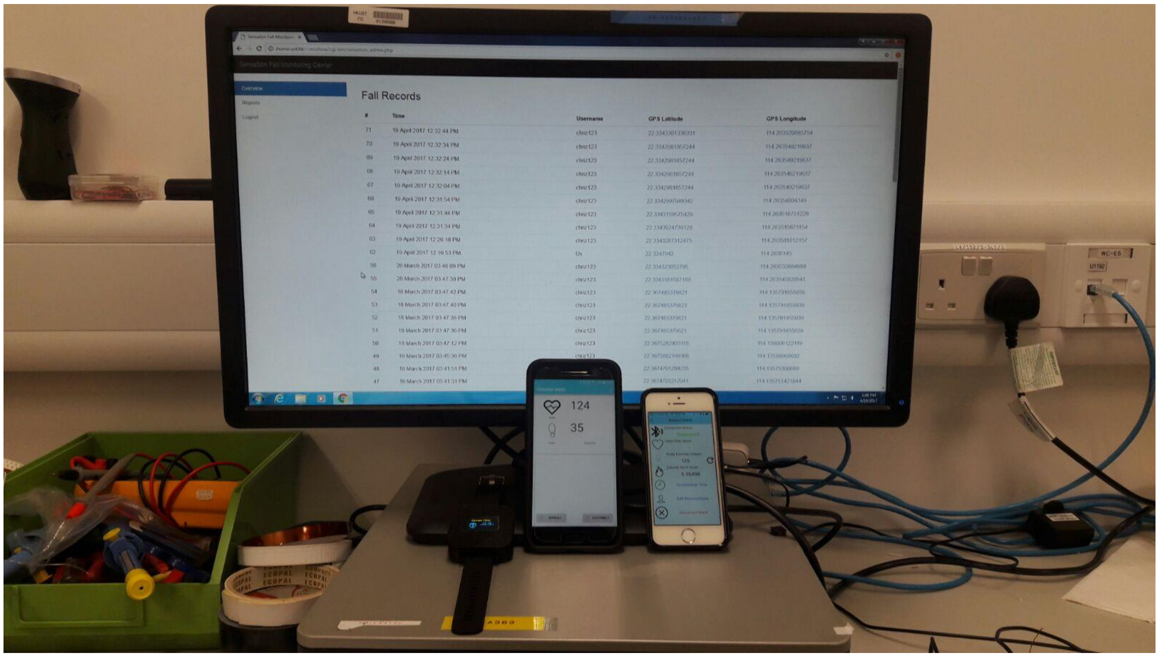


Figure 72 - A complete picture of Sensation Smart Watch, including web, smartwatch, Android app and iOS app

Chapter 4 – Conclusion

In conclusion, this project is to create a cost-effective wearable sensors system that provides continuous health monitoring. The system consists of a customized circuit board that is fully functional and compact, an intuitive sensors software that collects health data and interact with users, a server with web interface that shows all fall data and the location information, and two smartphone applications (Android and iOS) that synchronize the watch and communicate with the central server.

Four sub-projects, including circuit designs (schematic designs and PCB layout designs), sensors software programming, server backend programming and iOS Application programming are finished individually. Experiments and testing shows that all parts of the sub-project is functional: the customized circuit can work under different voltages and power source, the pedometer algorithms and fall-detection algorithm correctly detects the body movements most of the time, the server backend does store and show all fall records correctly with Google Maps redirections, and the iOS app can connect, synchronize, receive notifications and send fall notification to our server.

While combining other groupmates' work, it shows that the testing results are satisfactory. However, it also shows that there is room for improvement. For example, the heart rate detection algorithm and could be further improved to increase the reliability. Also, the size of our customized PCB could be further reduced such that it would be more comfortable to wear. These improvements could potentially make Sensation Smart Watch a better, more reliable health monitoring solution.

References

[1] P. Bonato, "Wearable sensors and systems," IEEE Engineering in Medicine and Biology Magazine, vol. 29, no. 3, pp. 25–36, May 2010.

[2] F. El-Amrawy and M. I. Nounou, "Are currently available Wearable devices for activity tracking and heart rate monitoring accurate, precise, and medically beneficial?" Healthcare Informatics Research, vol. 21, no. 4, p. 315, 2015.

[3] A. Low, "Xiaomi Mi Band review: A bargain price fitness band with great battery life," CNET, 2014. [Online]. Available: https://www.cnet.com/products/xiaomi-mi-band/review/. Accessed: Nov. 13, 2016.

[4] ARM, "Wearables Week teardown #3, Xiaomi Mi Band fitness and sleep tracker," 2014. [Online]. Available: https://community.arm.com/docs/DOC-9392. Accessed: Nov. 13, 2016.

[5] Xiaomi, "Mi Band - Mi Global Home," 2010. [Online]. Available: http://www.mi.com/en/miband/. Accessed: Nov. 13, 2016.

[6] G. Lu, F. Yang, J. A. Taylor, and J. F. Stein, "A comparison of photoplethysmography and ECG recording to analyse heart rate variability in healthy subjects," Journal of Medical Engineering & Technology, vol. 33, no. 8, pp. 634–641, Nov. 2009.

[7] "Integrated Analog Front End for Heart Rate Monitors and Low Cost Pulse Oximeters," in Texas Instruments. [Online]. Available: http://www.ti.com/product/AFE4400. Accessed: Nov. 13, 2016.

[8] Maxim Integrated, "MAX30100 pulse Oximeter and heart-rate sensor IC for Wearable health - maxim," 2016. [Online]. Available: https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/MAX30100.html. Accessed: Nov. 13, 2016.

[9] Silicon Laboratories, "Si1143/44 heart rate monitoring solution,". [Online]. Available: http://www.silabs.com/products/sensors/infraredsensors/Pages/si1144.aspx. Accessed: Nov. 13, 2016.

[10] "Accelerometer and gyroscopes sensors: Operation, sensing, and applications - application note - maxim," 2016. [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/5830. Accessed: Nov. 13, 2016.

[11] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy, "Wearable sensors for reliable fall detection," 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, 2005.

[12] M. James, "Scrum Reference Card," in Scrum Reference Card. [Online]. Available: http://scrumreferencecard.com/scrum-reference-card/. Accessed: Nov. 22, 2016

[13] Analog Device Inc., "ADXL362 Datasheet and Product Info," in Analog Device. [Online]. Available: http://www.analog.com/en/products/mems/accelerometers/adxl362.html#product-overview. Accessed: Nov. 22, 2016.

[14] Analog Device Inc., "ADXL345 Datasheet and Product Info," in Analog Device. [Online]. Available: http://www.analog.com/en/products/mems/accelerometers/adxl345.html#product-overview. Accessed: Nov. 22, 2016.

[15] "MPU-6050 DataSheet," InvenSense Inc., pp. 30–31. [Online]. Available: https://www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf. Accessed: Nov. 22, 2016.

[16] "iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer," in STMicroelectronics, p. 16. [Online]. Available: http://www.st.com/content/c cc/resource/technical/document/datasheet/ab/2a/3b/45/f0/92/41/73/ DM00087365.pdf/files/DM00087365.pdf/jcr:content/translations/en.DM00087365.pdf. Accessed: Nov. 22, 2016.

[17] "CC2541 Wireless Microcontrollers," in Texas Instruments. [Online]. Available: http://www.ti.com/product/CC2541/description. Accessed: Nov. 22, 2016.

[18] Solomon Systech, "SSD1306," in Solomon Systech limited, 2016. [Online]. Available: http://www.solomon-systech.com/en/product/display-ic/oled-driver-controller/ssd1306/. Accessed: Nov. 22, 2016.

[19] "Xiaomi Mi Band", Fitness-trackers.specout.com, 2017. [Online]. Available: http://fitness-trackers.specout.com/l/50/Xiaomi-Mi-Band. [Accessed: 19- Apr- 2017].

[20] "Apple Watch: Release Date, Features & News", iPhone Hacks | #1 iPhone, iPad, iOS Blog, 2017. [Online]. Available: http://www.iphonehacks.com/apple-watch. [Accessed: 19- Apr- 2017].

[21] "Apple Watch 42mm - Full phone specifications", Gsmarena.com, 2017. [Online]. Available: http://www.gsmarena.com/apple_watch_42mm-7696.php. [Accessed: 19- Apr- 2017].

[22] "CMMotionManager - Core Motion | Apple Developer Documentation", Developer.apple.com, 2017. [Online]. Available: https://developer.apple.com/reference/coremotion/cmmotionmanager. [Accessed: 19- Apr- 2017].

[23] "Apple Watch is too expensive for most consumers", BetaNews, 2017. [Online]. Available: https://betanews.com/2015/07/13/apple-watch-is-too-expensive-for-most-consumers/. [Accessed: 19- Apr- 2017].