Design and Development of Fall Analyzer

A Thesis
Presented to the Faculty of the
Department of Electronics and Communications Engineering
Gokongwei College of Engineering
De La Salle University

In Partial Fulfillment of the
Requirements for the Degree of
Bachelor of Science in Electronics and Communications Engineering

by

AMBION, Paulo Miguel M.
CUNANAN, Aenon Mari Viel T.
HARRIS, Edward Brian L.

March, 2018

De La Salle University

THESIS PROPOSAL APPROVAL SHEET

SAMPLE ONLY

This thesis proposal entitled **Electrical, Electromagnetic, and Optical Characterization of the InP/InGaAs Alloy System**, prepared and submitted by:

DELA CRUZ, Juan Z.
FRANCO, Nat Y.
GARCIA, Sebastian X.
MARTINEZ, Isabella W.
RIANZARES, Max V.

with group number ESG-04 in partial fulfillment of the requirements for the degree of **Bachelor of Science in Electronics and Communications Engineering** (**BS-ECE**) has been examined and is recommended for acceptance and approval.

PANEL OF EXAMINERS

Dr. Amado Z. Hernandez
Chair

Dr. Jose Y. Alonzo
Member

Dr. Mariana X. Mercado
Member

Dr. Francisco D. Baltasar
Adviser

Date: April 8, 2017

De La Salle University

ACKNOWLEDGMENT

ABSTRACT

Accidents is the 4th leading cause of death for all ages in the Philippines. One of the top five leading causes of death due to accidents in the Philippines for all ages is accidental falls. From worldwide statistics, 138 children, aged 0 to 18 years die daily due to falls. This translates to around 50,000 children dying each year due to accidental falls. It counts as the fifth leading cause of unintentional deaths of children below 14 years old in the Philippines. Falls were also found to be the leading cause of morbidity and lifelong disability among children. [Cri, 2014]

The goal of this project is to design and develop a device that can detect motion of user such as standing, sitting, lying, walking and falling with direction detection of fall whether the fall is in forward, sideward or backward motion. This project produced the said device and was successfully developed with 95% of the devices accuracy of detection. When falling motion was established, the notification system of the device was activated, and message was sent to the recorded responder/s with details such as location of fall, time of fall, name of user who has fallen and the date of fall.

The use of two accelerometers located at the hip and thigh resulted as a good component in making a motion detection device with 92.22% reliability. The development of falling device that can recognize direction of fall with the use of the same two modules resulted with 92.22% of accuracy. Motion detection such as standing, sitting, walking and lying that are sub-feature of this device was successfully detected with 97.5% overall accuracy rate. In general view of the devices performance, the design and development of fall analyzer with specific motion detection which can be personalized with the function of send notification message to recorded responder/s during the fall was achieved with above 80% acceptable accuracy rate.

Index Terms—.

TABLE OF CONTENTS

De La Salle University

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

GLOSSARY

Fall

- To come and go down suddenly from a standing position
- To let yourself come or go down to a lower position
- To decend freely by the force of gravity

Synonyms: trip, stumble

Chapter 1

INTRODUCTION

Contents

1.1 Background of the Study

Accidents which are external causes of death and injury is the 4th leading cause of death for all ages in the Philippines. Based on the statistics from 1975 to 2002, injury mortality rates have more than doubled from 19.1 in 1975 to 41.9 in 2003 per 100,000 population. One of the top five leading causes of death due to accidents in the Philippines for all ages is accidental falls. From the 4, 947 patients admitted to the Division of Trauma during a study period, 231 (4.7%) deaths were recorded and 205 (88.7%) of these were males. 135 (58.4%) from death were victims of penetrating injuries, and 96 (41.6%) had blunt trauma (vehicular injury in 66, falls in 15, mauling in 5, and other blunt injuries in 10). Intentional causes of injury (stab wound, gunshot wounds, and blunt assault) led to 151 (65%) deaths, while unintentional causes (vehicular crashes and falls) caused 80 (35%) deaths. [Mar, 2011]

From worldwide statistics, 138 children, aged 0 to 18 years die daily due to falls. This translates to around 50,000 children dying each year due to accidental falls. It counts as the fifth leading cause of unintentional deaths of children below 14 years old in the Philippines. Falls were also found to be the leading cause of morbidity and lifelong disability among children. [Cri, 2014]

From the Filipino elders in a nursing home facility and at the rehabilitation medicine out-patient department of the Philippine General Hospital (PGH), falls are considered one of the most serious health concerns encountered by the elderly. About 30 to 40% of individuals living in the community aged 65 and above fall each year. These accidents are associated with increased morbidity and mortality, and as much as 20 to 30% of those who fall suffer from serious hip fracture and head trauma. Current data show that falls comprise the single largest cause of death due to injury in the elderly. Recovery from falls is often poor because of restricted mobility and functional decline. Most falls have multiple causes, and are usually due to dynamic interplay of predisposing and precipitating factors. [Guevarra and Evangelista, 2010]

1.2 Prior Studies

TABLE 1.1 PRIOR STUDIES

Existing Study	Description
Design and Development of Fall Detector Using Fall Acceleration [Sudarshan et al., 2013]	This study aims to design and develop a prototype of an electronic gadget which is used to detect fall among elderly and the patients who are prone to it. A triaxial accelerometer (adx1335) is used in this study to measure the change of acceleration in three axes which are integrated from the body posture. To study the tilt angle, sensors are placed on the lumbar region of the body. To reduce the false alarms, the acceleration values in each axis are compared twice with threshold and a delay of 20 secs between two comparisons. Values of the threshold voltage are selected by experimental methods. The algorithm is executed by microcontroller (PIC16F877A). A GPS receiver uses the location of fall which is programmed to track the subject continuously. On detection of fall, the device sends a text message through GSM modem, and communicates it to computer through ZigBee transceivers. The device can also be switched to alarm only if a text message is not required. The prototype developed is tested on many subjects and on volunteers who simulated fall. Out of 50 trials 96% of accuracy is achieved with zero false alarms for daily activities like jogging, skipping, walking on stairs, and picking up objects.

Continued on next page

Continued from previous page

Existing Work	**Lacking Approaches**
Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information [Li et al., 2009a]	As prevalent methods only use accelerometers to isolate falls from activities of ADL, certain fall-like activities such as sitting down quickly and jumping result in many false positives which make it difficult to distinguish real falls. In some other non-horizontal ending position like falls on the stairs, the use of body orientation is not very useful to detect falls. This study has created a novel fall detection system using both accelerometers and gyroscopes. Human activities are divided into two categories: static postures and dynamic transitions. By using two tri-axial accelerometers at separate body locations, our system can recognize four kinds of static postures: standing, bending, sitting, and lying. Motions between these static postures are considered as dynamic transitions. Linear acceleration and angular velocity are measured to determine whether motion transitions are intentional. If the transition before a lying posture is not intentional, a fall event is detected. The algorithm, coupled with accelerometers, reduces both false positives and false negatives, while improving fall detection accuracy. In addition, the solution features low computational cost and real-time response.

1.3 Problem Statement

Many simple fall related accidents that lead to death or irreversible damage due to slow dissemination or lack of knowledge of responders (e.g. family members, friends or rescuers), can be prevented if information regarding the incident can be sent immediately to a responder.

Fall incidents are enumerated into different circumstances that defines itself as accident. Fall in this study is defined as an act of falling or collapsing or sudden uncontrollable descent or to suddenly go down onto the ground or towards the ground without intending to or by accident [Press, 2017]. It can also be defined as a sudden unintentional and uncontrollable descent which can also be classified as to stumble or to trip.

This study would like to give a solution to accidents which can be prevented if knowledge or information regarding the fall incident can be identified and sent quickly. It also aims to give aid to the innovation of the past technologies that has been done in relation to these problems.

Many different studies and technologies are made from the past to detect fall, but these studies differ in accordance on how fall is being focused. The observation of Raul Igual et. al in their fall detection analysis trends states that, in the study of Li, the use of gyroscope and accelerometer in analyzation of fall types such as falling forward, backward and sideways is a success. Compared to the other technologies used to detect fall like image processing, the use of accelerometer is similar to this study. The advantage of this study over Lis is that in her study, she located the sensors on the waist and chest, whereas this study will locate the sensors on the hip and on the thigh as based from the past studies these are better locations to achieve a higher success rate. [Igual et al., 2013]

1.4 Objectives

1.4.1 General Objective(s)

To design a device that can detect users motion and can send a notification to responder/s via Global System for Mobile (GSM) when fall is detected.

1.4.2 Specific Objectives

1. To develop a system that can analyze the duration and direction of the fall whether the user has been fallen backward, forward, or sideward with 80% of accuracy upon detection.
2. To create an algorithm that will store data collected from the user to a Secure Digital Card (SD Card) that has user-friendly functions.
3. To collect data that will be stored in the database from the user such as location and certain motions using Global Positioning System (GPS) and accelerometer respectively.
4. To develop a system that will allow the device to send information to the responder/s.
5. To develop an algorithm that can detect specific motions of the user such as standing, walking and lying.

1.5 Significance of the Study

Imagine the advantages of having a device that can detect and acknowledge a persons motion specifically when they unintentionally fell or tripped. As fall is one of the leading causes of death, it is an advantage to know how and when to help the victims after the incident.

This project aims to create a device that will innovate the detection of such incident. It also aims to notify certain responders when the said incident happened. By using this device, the goal to respond to fall related accidents will be achieved. It will be a solution to the fast-growing statistics of deaths, and other injuries that fall related accidents give.

Also, as the device is done, it can be used for further development of fall analyzer with the use of accelerometer. It can also be used in the medical field and through innovation, the detection and notification of a fallen patient can be recognized easily which may eventually lead to the prevention of fall related accidents.

1.5.1 Scope

1. The device can only detect limited fall directions such as forward, backward and sideward.

2. The device will gather the data for the motion and GPS coordinates in real time.

3. This project will only require 30 people with an age range of 8 to 24 years old as test subjects. These subjects will perform activities for raw data gathering and would be translated into digital information eventually.

4. A text-based database will be created to store the given data into the SD card. It could only store a limited amount of data so an Automatic delete and save algorithm will be implemented to save important data. The device will make a beeping sound to notify the user if it reached 90% of its memory storage. It will automatically clear the

data that has been saved and the unnecessary data will be deleted to allocate memory storage.

5. The Graphical User Interface (GUI) must initialize important information of the user such as name, age, address, gender, personal contact, and contact numbers of the responder with their name.

6. After the device has detected that the user needs an assistance, the set responder/s will be notified through Short Message Service (SMS). A False alarm button is included in the device so that user has an option to stop the assistance.

7. Fall detection is the focus of this project. The hip and thigh will have an individual accelerometer to determine basic body orientations such as sitting, standing, walking and lying.

1.5.2 Delimitations

1. The communication ability of the device will only be limited to the Philippines.

2. SMS are system-generated. The user cannot modify nor manually send SMS however the user can interrupt SMS that will be sent.

3. The current location of the user will be retrieved as GPS coordinates and not as the exact address of the user.

4. Two accelerometers will be used for the project, one for the upper body and one for the lower body. Two accelerometers are being used in accordance to the effectivity of the same number of sensors from the past related studies. By these, limited motions and body orientation of the user will be detected.

1.6 Description of the Thesis

This project is a device that can detect if a user had a sudden fall and the direction on how the user had fallen. It can detect fall directions like forward, backward and sideward. Other motions such as sitting, standing, walking and lying can also be detected as a sub-feature of the device. It is user-friendly because the device can be set by accessing the GUI. This device is designed to detect pre-fall, during the fall and post-fall motions. It can detect pre-fall motions which the user is possibly doing before the fall like sitting, standing or walking. It can also recognize whether the user had fallen and post-fall detection can also be acknowledged in which after the fall GPS coordinates.

When the device detects the said fall related incidents, it will also send a notification through GSM communication to responders like family members, friends, and other people that the user nominates. The notification also includes the GPS coordinates of the user, with the help of GPS module.

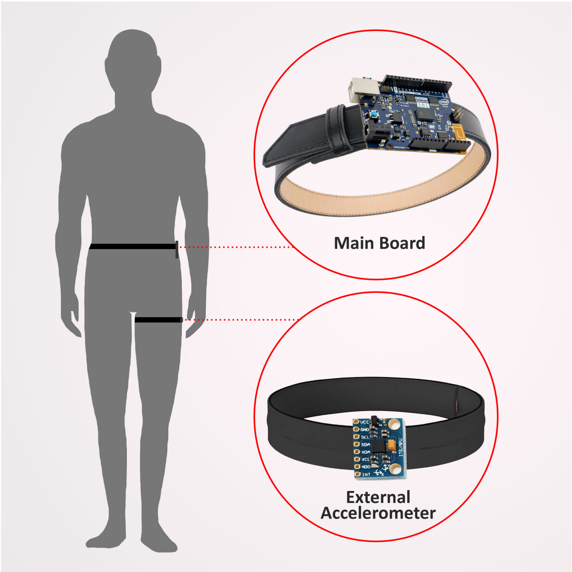


Fig. 1.1 Location of the device on the user

Shown in Figure 1.1 is the way the device will be worn by the user. The Main Board will be located at the hip of the user and can be attached on its belt. The Main Board has a built-in accelerometer. GSM Module and SD Card Module will also be attached on the Main Board. The External Accelerometer will be placed on the thigh of the user which can be attached on a belt or a garter.

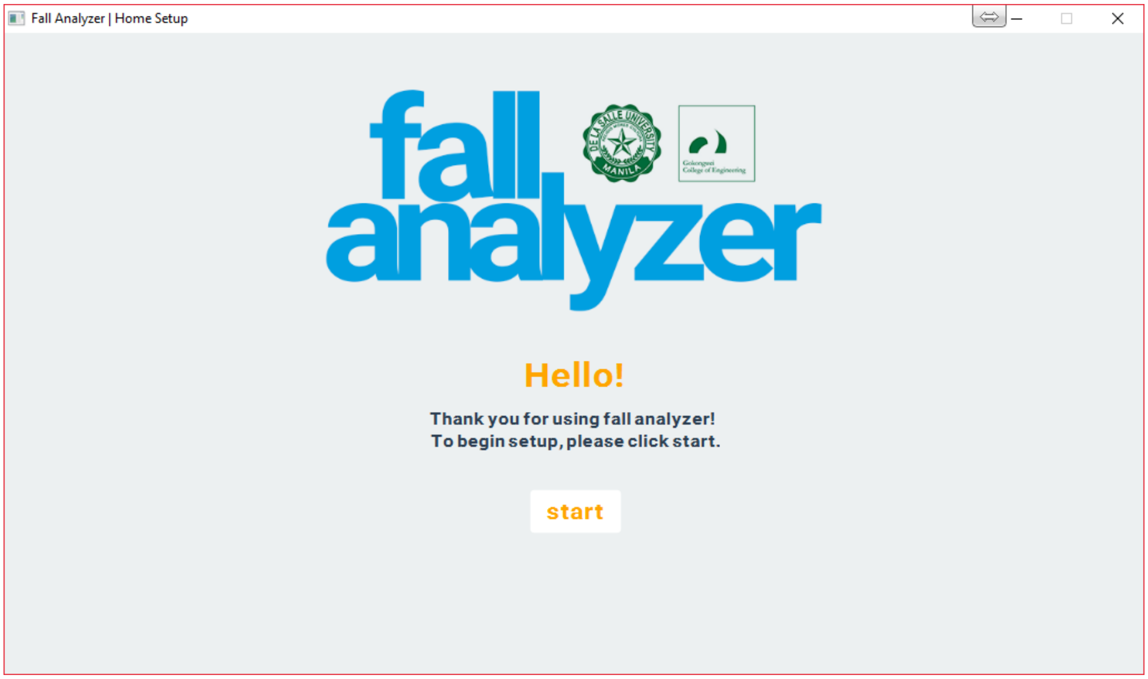


Fig. 1.2 Initial User Setup

Figure 1.2 to figure 1.7 shows the GUI interface of the device set-up. Through these windows, the users information will be recorded wherein the said data may only be edited or changed by the user. Through this setup, responder/s can only view the data recordings. As shown in Figure 1.2, the simplicity of the interface can be seen as the device is being initialized, which makes the GUI user-friendly. In figure 1.3, the device is being setup by asking for a username and a password. Profile set up on Figure 1.4 initialized the users

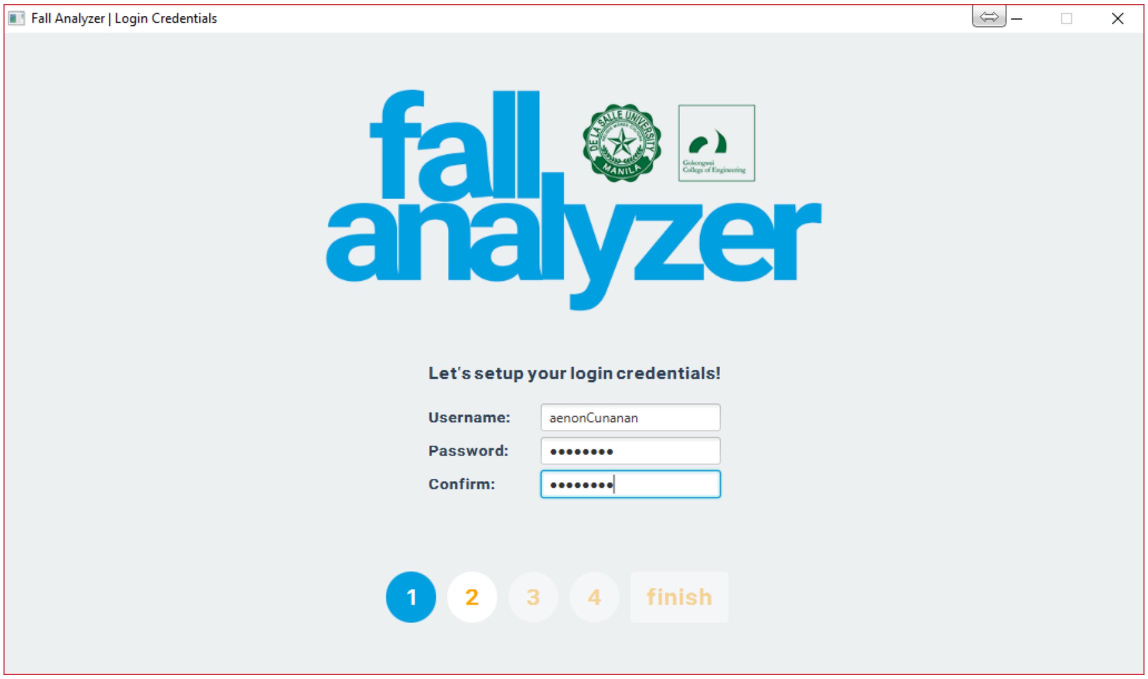


Fig. 1.3 User Setup, logging in and password setting

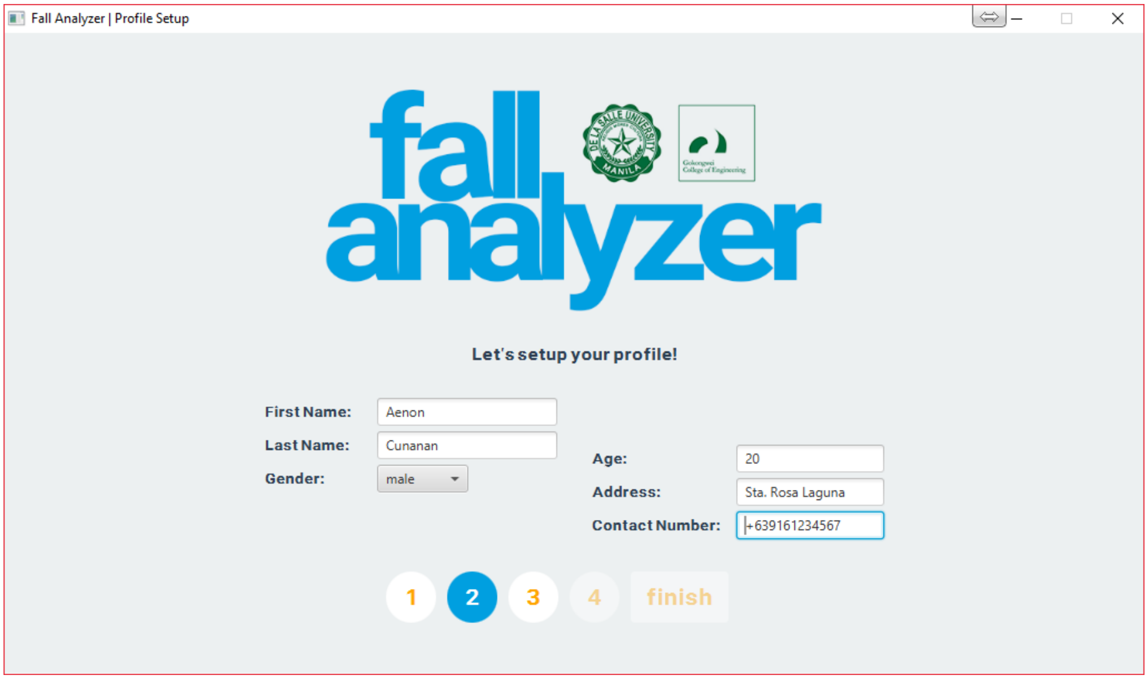


Fig. 1.4 Profile Setup

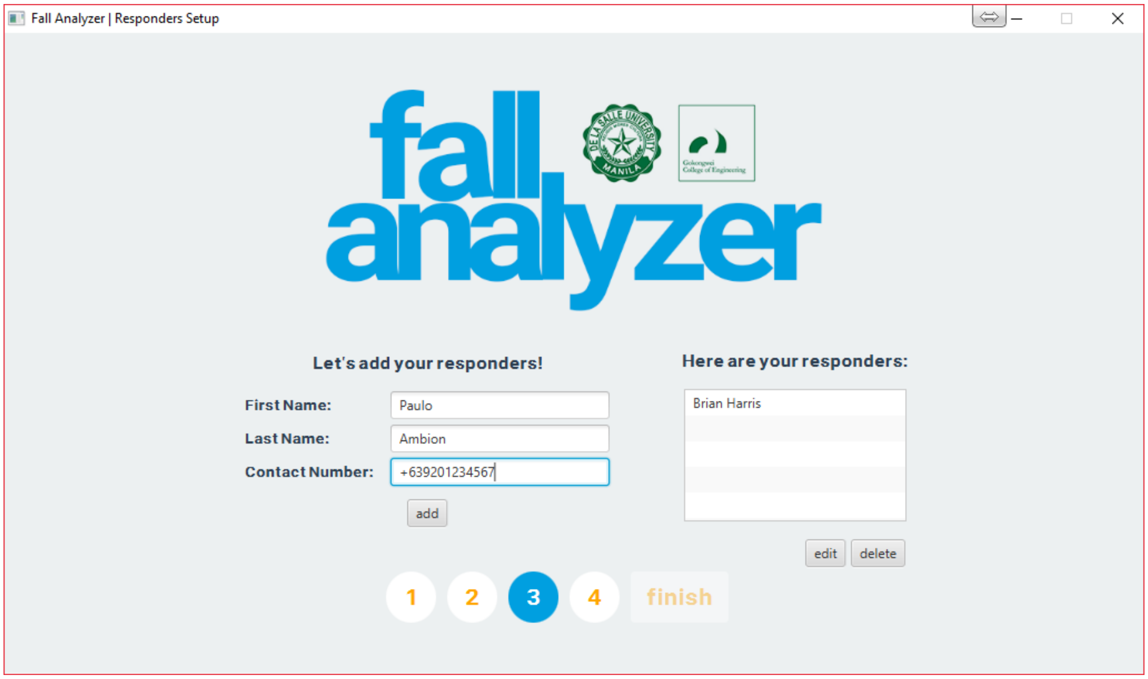


Fig. 1.5 Responder's Setting

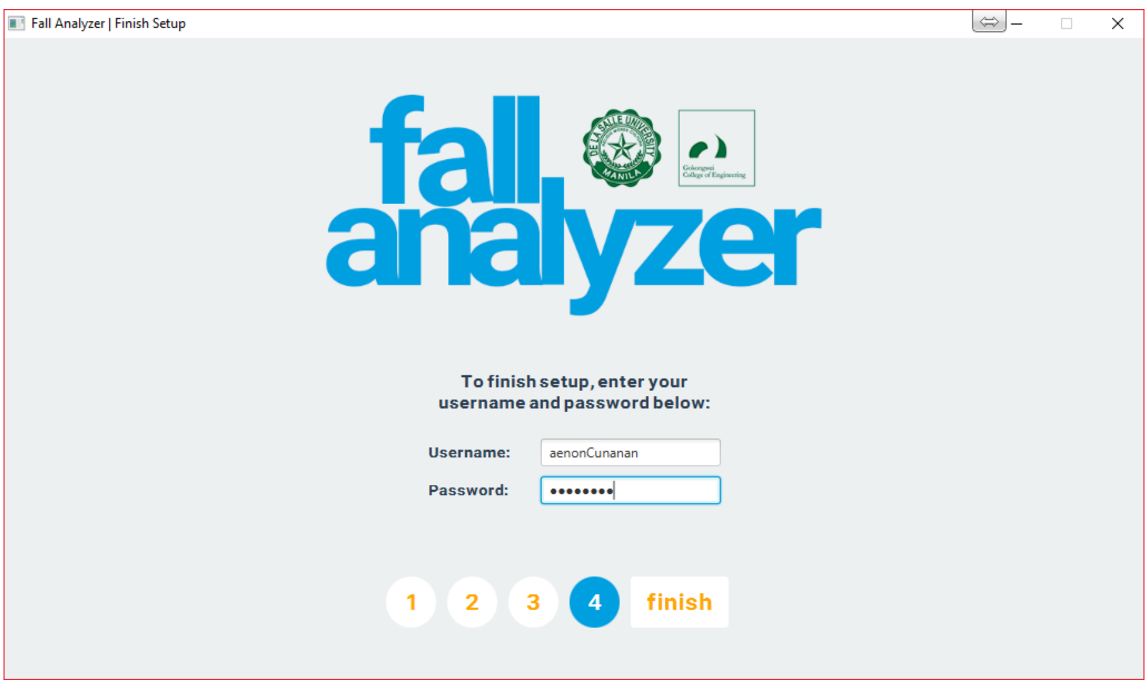


Fig. 1.6 Confirmation of Identification and password

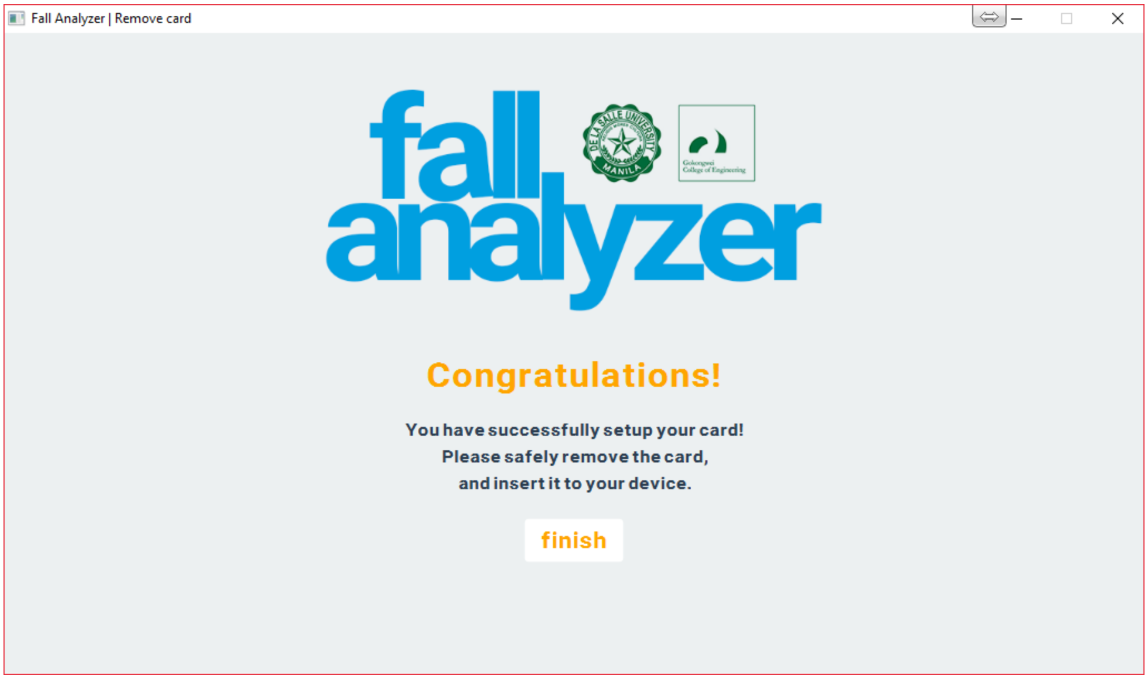


Fig. 1.7 Setup success page

details. In this tab the complete name, address where they are staying, contact number and gender is being registered. In Figure 1.5, the profile of the responder/s is being setup which includes the name and immediate contact (mobile) number. Confirmation of username and password can be seen in Figure 1.6. Lastly, the success page in Figure 1.7 is shown which concludes the users setup. Only the user has the access to override and change the data from the device. Responders have limited access on the device where the data has been recorded.

After the device has been setup, the user must first insert the SD card to the device to gather and save data. The user or responder can review the data by inserting the said SD card in a computer then launching the GUI. The GUI will first ask the user or responder to select the proper storage device as shown in Figure 1.8.

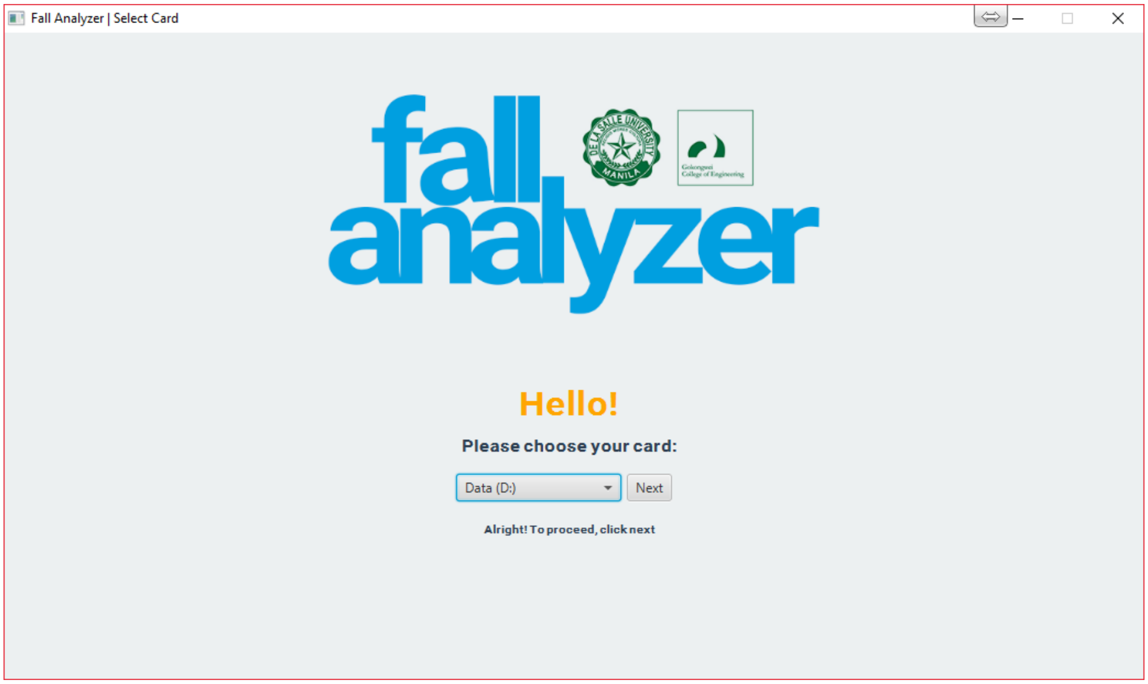


Fig. 1.8 Storage Selection

Shown in Figures 1.9 to 1.11 is the explore window where the data records can be found. Figure 1.9 shows the profile tab which includes the name, gender, contact number, and address of the user. Responders tab can be seen in Figure 1.10, from which the responders details such as name, and contact number can be seen. On the next tab shown in Figure 1.11, the activity log where the activities and motions of the user can be found. These tabs from the explore window can be seen by all but only the user has the access to edit the recorded data. If the user wishes to modify some data, a prompt will show up as shown in Figure 1.12. The prompt will request the users username and password to continue the modification of the recorded data.

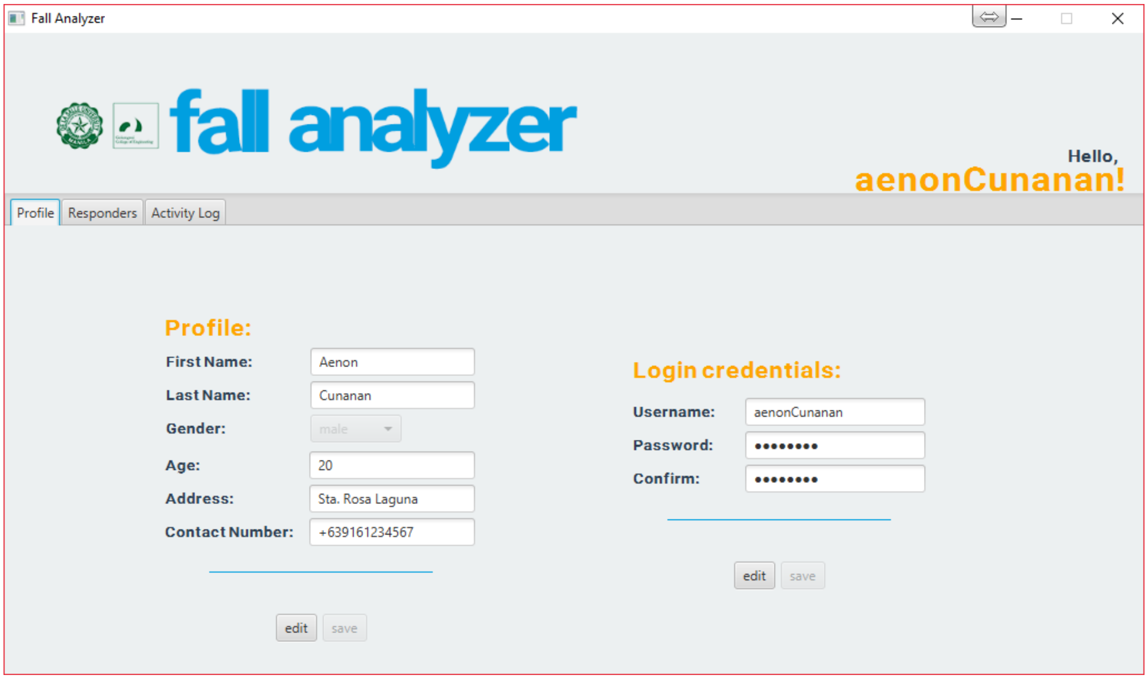


Fig. 1.9 Profile Tab

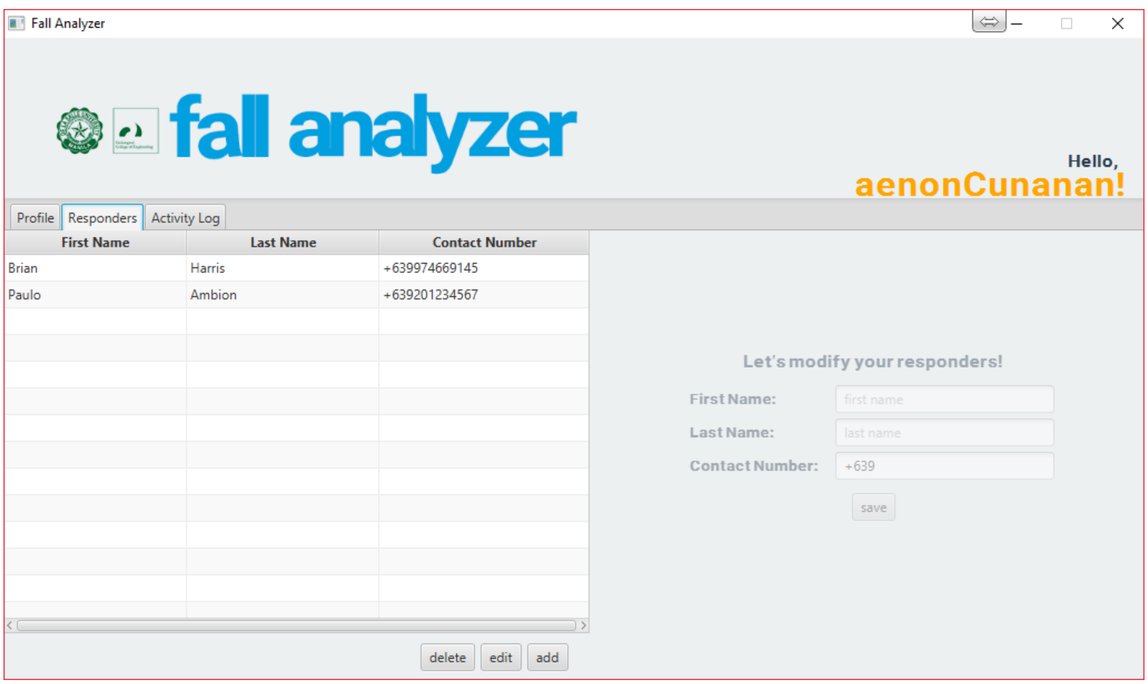


Fig. 1.10 Responder's Tab

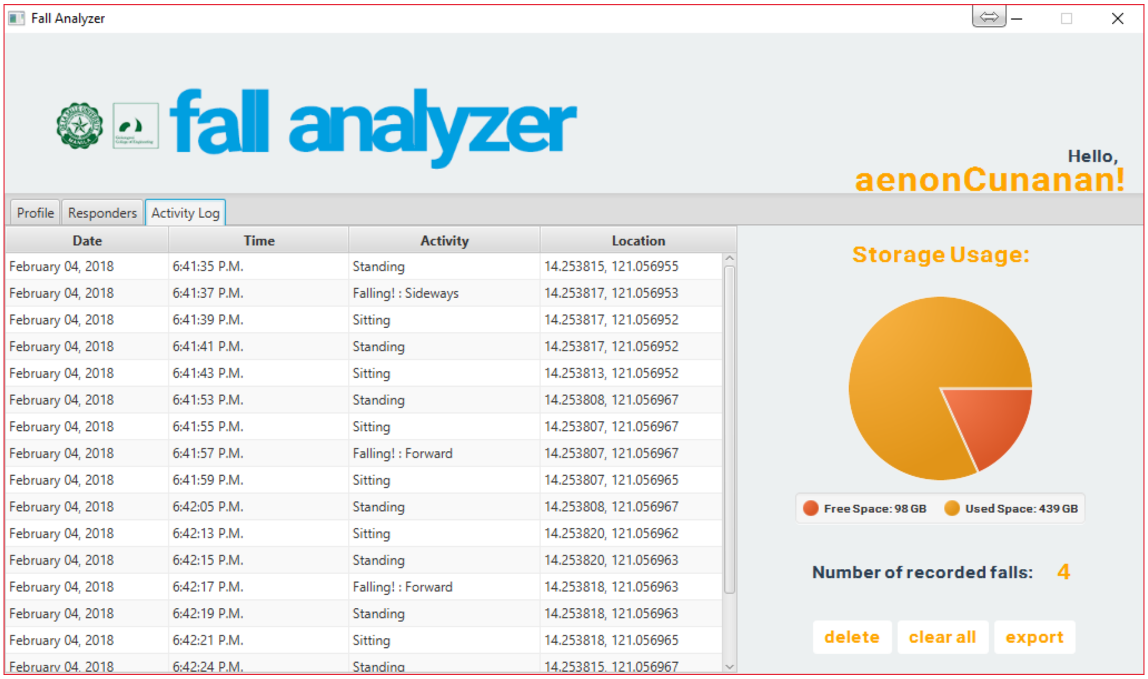


Fig. 1.11 Activity Log

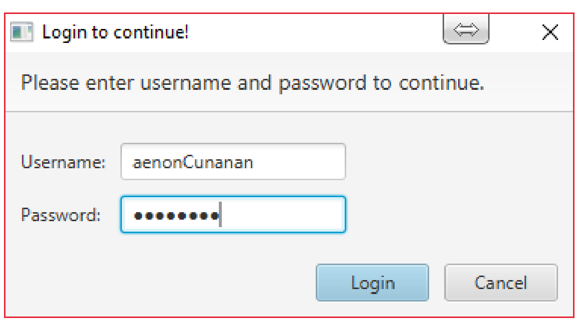


Fig. 1.12 Prompt

De La Salle University

1.7 Estimated Work Schedule and Budget

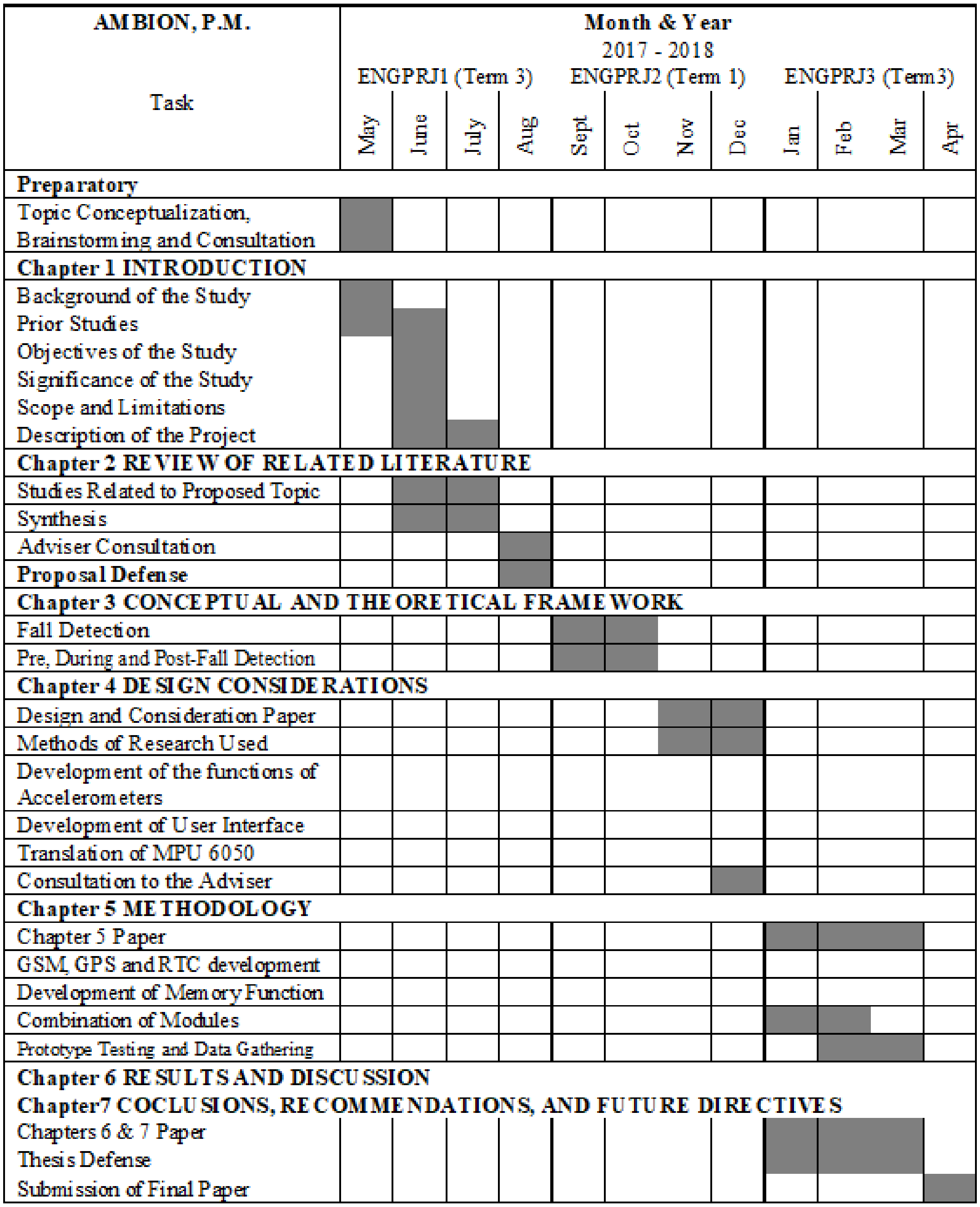


Fig. 1.13 Estimated Work and Schedule (a)

CUNANAN, A.M.	Month & Year											
	2017 - 2018											
	ENGPRJ1 (Term 3)				ENGPRJ2 (Term 1)				ENGPRJ3 (Term 3)			
Task	May	June	July	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Preparatory												
Topic Conceptualization, Brainstorming and Consultation												
Chapter 1 INTRODUCTION												
Background of the Study												
Prior Studies												
Objectives of the Study												
Significance of the Study												
Scope and Limitations												
Description of the Project												
Chapter 2 REVIEW OF RELATED LITERATURE												
Studies Related to Proposed Topic												
Synthesis												
Adviser Consultation												
Proposal Defense												
Chapter 3 CONCEPTUAL AND THEORETICAL FRAMEWORK												
Fall Detection												
Pre, During and Post-Fall Detection												
Chapter 4 DESIGN CONSIDERATIONS												
Design and Consideration Paper												
Methods of Research Used												
Development of the functions of Accelerometers												
Development of User Interface												
Translation of MPU 6050												
Consultation to the Adviser												
Chapter 5 METHODOLOGY												
Chapter 5 Paper												
GSM, GPS and RTC development												
Development of Memory Function												
Combination of Modules												
Prototype Testing and Data Gathering												
Chapter 6 RESULTS AND DISCUSSION												
Chapter7 COCLUSIONS, RECOMMENDATIONS, AND FUTURE DIRECTIVES												
Chapters 6 & 7 Paper												
Thesis Defense												
Submission of Final Paper												

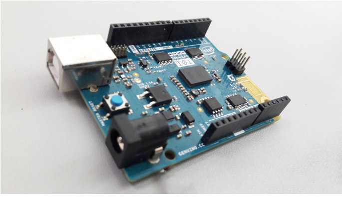
Fig. 1.14 Estimated Work and Schedule (b)

HARRIS, E.B.	Month & Year 2017 - 2018											
	ENGPRJ1 (Term 3)				ENGPRJ2 (Term 1)				ENGPRJ3 (Term3)			
Task	May	June	July	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Preparatory												
Topic Conceptualization, Brainstorming and Consultation												
Chapter 1 INTRODUCTION												
Background of the Study												
Prior Studies												
Objectives of the Study												
Significance of the Study												
Scope and Limitations												
Description of the Project												
Chapter 2 REVIEW OF RELATED LITERATURE												
Studies Related to Proposed Topic												
Synthesis												
Adviser Consultation												
Proposal Defense												
Chapter 3 CONCEPTUAL AND THEORETICAL FRAMEWORK												
Fall Detection												
Pre, During and Post-Fall Detection												
Chapter 4 DESIGN CONSIDERATIONS												
Design and Consideration Paper												
Methods of Research Used												
Development of the functions of Accelerometers												
Development of User Interface												
Translation of MPU 6050												
Consultation to the Adviser												
Chapter 5 METHODOLOGY												
Chapter 5 Paper												
GSM, GPS and RTC development												
Development of Memory Function												
Combination of Modules												
Prototype Testing and Data Gathering												
Chapter 6 RESULTS AND DISCUSSION												
Chapter7 COCLUSIONS, RECOMMENDATIONS, AND FUTURE DIRECTIVES												
Chapters 6 & 7 Paper												
Thesis Defense												
Submission of Final Paper												

Fig. 1.15 Estimated Work and Schedule (c)

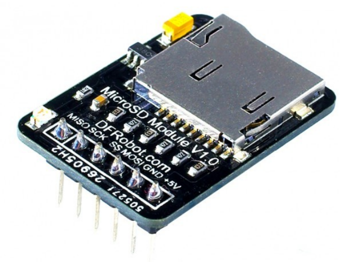
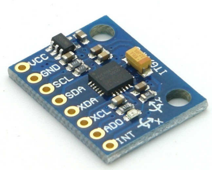
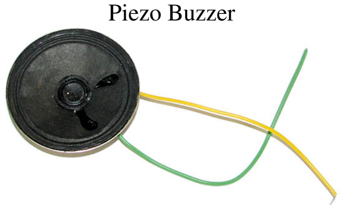
De La Salle University

TABLE 1.2 ESTIMATED BUDGET

Components and Parts	Basic Details	Significance of components to the project	Price
Genuino101 	This is a 32-bit microcontroller, powered by Intel Curie microprocessor that has 196 kb flash memory and 24 kb sram with built in Bluetooth LE and 6-axis accelerometer/-gyro technology. [Intel, 2018]	This microcontroller will be used to detect and assess the motion and orientation of the user using its built-in accelerometer.	P 2,048.00
GSM/GPS/GPRS/SMS SIM808 CROWTAIL and Sim Card 	This is a GSM and GPS two-in-one function module which is called Crowtail- SIM808. It is very small and based on the latest GSM/GPS module SIM808 from SIMCOM, it supports GSM/GPRS Quad-Band network and combines GPS technology for satellite navigation. It has high GPS receive sensitivity with 22 tracking and 66 acquisition receiver channels that will lets you add location-tracking, voice, and text. [Circuitrocks, 2018a]	This GSM/GPS/GPRS module will be used to detect the GPS coordinates of the user. It has also GSM capabilities, so that we can send SMS message to notify responder/s that the user need help and assistance.	P 1,917.00 P 50.00

Continued on next page

Continued from previous page

Components and Parts	**Basic Details**	**Significance of components to the project**	**Price**
SD card socket Module and SD card 	This is an Arduino compatible module and SD card library could be used with this device. [Circuitrocks, 2018c]	The SD card module will be used to read and write to the SD card to create a database. The database will be composed of data collected and the basic information of the user.	P 105.00 P 300.00
MPU6050 Accelerometer Module 	This is an InvenSense MPU-6050 sensor that contains a MEMS accelerometer and MEMS gyro in a single chip. It contains 16-bits analog to digital conversion hardware for each channel. It captures the x, y, and z channel at the same time. [Circuitrocks, 2018b]	This module will be used to detect falling motions and the specific motions of the user. Together with the built-in accelerometer in the Arduino 101 module, acceleration and dynamics can be computed which can be used to analyze the said motions.	P 162.00
Piezo Buzzer 	This is a piezoelectric speaker or buzzer that uses the piezoelectric effect for generating sound. It is a d36 mm Mylar Cone 8ohms 0.5w [e Gizmo Mechatronix Central, 2018d]	This component will be used to alarm as it notifies the user when the device detects a falling motion or to alarm the user if the memory of the device is almost full.	P 22.00

Continued on next page

Continued from previous page

Components and Parts	**Basic Details**	**Significance of components to the project**	**Price**
Push Button 	This is a side tact switch SPST through hole 7x4mm package/image: 77008044. It causes a temporary change in the state of an electronic circuit only while the switch is physically actuated. [e Gizmo Mechatronix Central, 2018f]	This push button is used to terminate the process of the device on sending SMS to the responder recorded.	P 6.00
9v Rechargeable Battery 	This is a 9-volt rechargeable battery with 300 mAh which can be charged into standard charger for 14-15 hours.	This battery will be used to power up the fall analyzer device.	P 493.50
9v Battery Snap 	This is a battery clip for 9v battery attachment with complete red and black power lines. [e Gizmo Mechatronix Central, 2018b]	This battery snap will be used for connecting the device to the 9v battery that will serve as a power source.	P 5.00
Power Switch 	This is a 6A 250V AC/10A 125 Ac Snap boat rocker switch [e Gizmo Mechatronix Central, 2018e]	This switch will be used in connecting the power source to the device.	P 15.00

Continued on next page

Continued from previous page

Components and Parts	**Basic Details**	**Significance of components to the project**	**Price**
300-ohm resistor 	This is a 300-ohm resistor 1/4W. [e Gizmo Mechatronix Central, 2018a]	This resistor will be used for controlling the voltage and current going to the LED	P 0.25
Red LED 	This is a 10-mm red and yellow light emitting diode. [e Gizmo Mechatronix Central, 2018c]	This LED will indicate if the device is already connected to the GPS.	P 6.50/pc
Yellow LED 	This is a 10-mm red and yellow light emitting diode. [e Gizmo Mechatronix Central, 2018c]	This LED will indicate if the device is ready to use.	P 6.50/pc

1.8 Overview of the Thesis

The background of the study, the related studies, the statement of the problem, the general and specific objectives, the significance of the study and the scope and limitations of the study was overviewed in this chapter. On Chapter 2, the compilation of previous studies can be seen. It is a collection of past studies and how those studies helped in assembling and formulating of this study. Chapter 3 shows the theories where this project was based.

In this chapter, the theoretical background on how each part of the fall analyzer was made. Chapter 4 contains the design of the project. Chapter 5 enumerates the methods on how the prototype of fall analyzer has been made. Chapter 6 contains the result and discussion of the experiment. In here the experiments data and the evaluation of those data can be seen. Lastly, Chapter 7 has the conclusion and recommendation of the study.

Chapter 2

LITERATURE REVIEW

Contents

2.1 Existing Work

For aged population, falls are dangerous for they can adversely affect health. [Li et al., 2009b] Fracture, certain possibility to get coma, brain trauma, and paralysis are the most common injury in fall of an elderly. The high impact is the main source of injury at most fall situations. But sometimes the late medical salvage may worsen the situation. That means the faster the salvage comes, the less risk the elderly will face which makes the progress of technology brings more possibilities to help us protect the elderly [Wu et al., 2014]

Several kinds of fall detection methods have been developed or applied in our life. Most of the research on falls where accelerometers are used focus on determining the change in magnitude of acceleration. When the acceleration value exceeds a critical threshold, the fall is detected. These systems successfully detect falls with sensitivities greater than 85% and specificities between 88-94%. However, focusing on large acceleration only can result in many false positives from fall-like activities such as sitting down quickly and running. [Huynh et al., 2013]

Some fall detection algorithms also assume falls happen when the body lies prone on the floor which are less effective when a persons fall posture is not horizontal, e.g. fall happened on stairs. Furthermore, previous studies used complex algorithms like Support Vector Machine (SVM) and Markov models to detect the fall. However, accuracy of these systems has not been proven to be highly effective. They also use excessive amounts of computational resources and cannot respond in real time. In addition, fall activity patterns are particularly difficult to obtain for training systems. Unlike other previous research, this project proposes using the accelerometer sensor to detect the falls for increasing the sensitivities and specificities of a fall detection system. Accelerometers can detect orientation and axis of movements as it is one of the feature of these technologies [Huynh et al., 2013]

Fall detection:

Existing fall detection solutions can be divided into two classes. The first class only analyzes acceleration to detect falls. A second class of solutions utilize both acceleration and body orientation information to detect falls [Li et al., 2009b] The algorithm uses the accelerometer present in cellphones to monitor falls. If a fall is detected, the application automatically notifies predefined contacts (such as parents or emergency services) with the victims GPS coordinates shown on a map. [Kazi et al., 2014]

When the subject falls, the acceleration is rapidly changing, and the angular velocity produces a variety of signals along fall direction. The lower and upper fall thresholds for the acceleration and angular velocity used to identify the fall are derived as follows:

- Lower Fall Treshold (LFT): the negative peaks for the resultant of each recorded activity are referred to as the Lower Peak Values (LPV). The LFT for the acceleration signals are set at the level of the smallest magnitude Lower Fall Peak (LFP) recorded.
- Upper Fall Treshold (UFT): the positive peaks for the recorded signals for each recorded activity are referred to as the signal Upper Peak Value (UPV). The UFT for each of the acceleration and the angular velocity signals were set at the level of the smallest magnitude UPV recorded. The UFT is related to the peak impact force experienced by the body segment during the impact phase of the fall [Huynh et al., 2013]

Like every other past studies, accelerometer and gyroscope have limited abilities to measure a fall. All past algorithms involving these two components, had a good detection rate for actual falls, and had low false negatives except in the case of jumping onto a walking position in a bed or running. [Kazi et al., 2014]

Transmission:

When a fall has certainly occurred, there is still one false positive that remains to be accounted for, that is if a user accidentally falls and can still manage to stand up or simply in cases where the device can call the incident a false alarm. To account for this, a 10-second grace period has been added to the device for the user where the device delays sending a text to a responder or the user can choose to cancel altogether. If the alarm is not cancelled within a pre-specified time, the device will send the persons GPS coordinates with a message to predefined people via GSM. [Kazi et al., 2014]

Data acquisition:

In the hardware part, microcontroller handles the acquisition of accelerometer data, captures UTC time using an additional discriminator connected to the GPS receiver module and merges these data for transmission via Bluetooth module. To communicate with these modules, the microcontroller uses communication protocol for accelerometer module and UART interface for GPS and Bluetooth modules, respectively. Microcontroller program codes were written and compiled with Proton IDE software [Koray et al., 2014]. All data can be logged to memory card for responder/s acquisition.

2.2 Lacking in the Approaches

TABLE 2.1 LACKING APPROACHES

Existing Work	**Lacking Approaches**
Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information [Li et al., 2009b]	This study combines the use of gyroscope and accelerometer to create a device that can detect a fall - which is different to other studies that uses accelerometer that isolates falls from ADL. This study sets two categories that divides ADL, the static and dynamic postures. Motions such as standing, sitting and lying does not limit our study. Our study has a sub feature of determining a dynamic motion such as walking. Using the same modules, the accelerometer and gyroscope, determines whether the user is subjected to fall even when they are in static motion like sitting, standing and lying so as in dynamic motion like walking. This study also uses linear acceleration and angular velocity to determine whether the motion transition is intentional and uses it to determine a sudden fall. Our study however, uses time difference of the previous position and the current position when the user has fallen. Using this simple method, intentional fall can be differentiated with sudden fall and it uses a simple computational algorithm that can help the device to compute data easily compared to this study.

Continued on next page

Continued from previous page

Existing Work	**Lacking Approaches**
A Wireless Body Area Network of Intelligent Motion Sensors for Computer Assisted Physical Rehabilitation [Jovanov et al., 2005]	The focus of this study is to create a wireless sensor that can be used to detect multi-tier telemedicine system that can be implemented by the aid of computer-assisted physical rehabilitation applications and ambulatory monitoring. Our study focuses on the same concept but is specialized on the fall detection of a patient where in order for the device to detect a fall, it does not need the continuous attendance of a computer-based monitor nor the attendance of a person who will manually monitor the capability of device to monitor the overall medical performance of the user. The communication system of this study that gives notification to the responder, and the ability to send the recorded data from the device rely on the use of internet, while our study submits notification to the responder using SMS and the recorded data of the user before, during and after the fall can be access directly through the device. This means the absence of internet does not interfere to the notification ability of the system of our project, especially the design was based from the Philippine settings in which internet connection is not reliable.

Continued on next page

De La Salle University

Continued from previous page

Existing Work	**Lacking Approaches**
Enhancing the Quality of Life Through Wearable Technology [Park and Jayaraman, 2003]	Wearable technology is the focus of this study. It provides a wearable medical device that detects medical issues of user and give information to the attending guardian or doctor. It includes a motherboard that is based on fabric and was created to be a smart wearable shirt. The wearable characteristic of this study was achieved, and, in our study, we tried to create a device that can be carried by the user and can detect fall emergency that can also give details of the pre-fall, during fall and post-fall events of the user to the responder. The user-friendly feature, and the can be carried device characteristic can be considered the wearable feature of the device. Also, the information of the user as they use the device in our study was enhanced by the location detector, fall detector and the ability to send a message to a responder during a fall of the said device.

Continued on next page

Continued from previous page

Existing Work	**Lacking Approaches**
Development of a Wearable-Sensor-Based Fall Detection System [Wu et al., 2014]	This study used the accelerometer to identify, analyze and recognize fall. It comes together with a GPS/GSM module that is used only when the user is outdoor. Through these devices, the location of the user when fallen can be identified and can be used to notify the responder to give an immediate action for the incident. Our study also uses accelerometer. Though through the results of other studies it was proven that the combination of accelerometer and gyroscope increases the efficiency of identifying fall, in this study vector analysis was used in accelerometer to allow to have the module a better range of identifying motions. Using the tri-axial accelerometer, the previous activities of the user such as walking, standing or sitting can also be known before the user has fallen. On the other hand, the use of GPS/GSM module in our study was maximized. The location of the user can also be identified even when he/she is indoor and not just outdoor. An SMS is being sent in case the user has fallen anywhere and anytime to the registered responder from the module itself.

Continued on next page

Continued from previous page

Existing Work	**Lacking Approaches**
Posture and Movement Classification: The Comparison of Tri-Axial Accelerometer Numbers and Anatomical Placement [Fotune et al., 2014]	The placement of tri-axial accelerometers on to the body was studied in this project. Three areas of the body were used to identify the most accurate results that an accelerometer can give. In this study, the waist, thigh and ankle were tested differently. As a result, for a single accelerometer, the thigh part is the best area that can give the better accurate result for the accelerometer. However, for two accelerometers, the combination of waist-thigh gives the most responsive and accurate results among other areas of the body. In our study, waist-thigh area is used to place the accelerometer and two accelerometers were also used. The combination of the built-in accelerometer from the genuino 101 and another accelerometer clearly identifies the static and dynamic motion of the user through analyzing its vector component.
Fall Detection System Using Combination Accelerometer and Gyroscope [Huynh et al., 2013]	This study investigates the methodology to identify falls from normal ADL. In this study, a wireless sensor system, based on accelerometer and gyroscope, is placed at the center of the chest to collect real time fall data. Experiment protocols consisting of four types of fall such as forward, backward and sideward (right and left) falls along with normal gait. [Huynh et al., 2013] Our study also identifies the direction of fall using the combination of two accelerometers, however these modules were not placed on the chest but rather on the waist and thigh as those were the best areas to locate the sensors based from the other studies that concerns the detection of fall and ADLs. In addition, the location on where the user has fallen is being identified in our study and in such case a notification system is activated via GSM that gives notification to the registered responder in the module.

Continued on next page

De La Salle University

Continued from previous page

Existing Work	Lacking Approaches
Fall Detection Using Single Tri-Axial Accelerometer [Kazi et al., 2014]	This study uses a single tri-axial accelerometer that is a mobile phone-based system which implements a fall detection algorithm using a mobile phones built-in accelerometer which can detect falls with the victims GPS coordinates displayed on the map for timely delivery of medical help. [Kazi et al., 2014]. Our study is similar with this -, it can also detect fall using a tri-axial accelerometer. The notification system also identifies the predetermined responders in case of fall together with the location of the user using GPS coordinates. In addition, our study also identifies and differentiate motions such as static and dynamic from fall. Motions such as standing, sitting, lying and walking can also be determined in our study and during the response of the responder in such incident of fall, the previous activity of the user can be identified and be used for further evaluation of the situation.

Continued on next page

Continued from previous page

Existing Work	**Lacking Approaches**
Wireless 3-Axis Accelerometer System for Measuring of Structural Displacement [Koray et al., 2014]	This study presents the design of a 3-axis acceleration measurement system that can Coordinate UTC time-stamping captured from GPS receiver module. Acquired 3-axis acceleration data and UTC are transferred via Bluetooth protocol and developed software which enables monitoring and recording UTC and acceleration data on PC, respectively. [Koray et al., 2014] This study uses UTC to record time and GPS to analyze displacement combined with the help of acceleration to identify displacement of user. These combined project results to an acceptable result but with a huge delay due to the complications of calculations. In our study, time and location can be identified by the help of GPS module while displacements or the body movement of user and the body orientation is being analyzed using the two tri-axial accelerometers. Both sets of modules aims to identify the displacement, , time and body orientation but our study enhances the delay of this study.

Chapter 3

THEORETICAL CONSIDERATIONS

Contents

3.1 Summary

In this chapter, the hardware and software consideration in making the fall analyzer device can be seen. This chapter explains the parts and functions of fall analyzer device.

3.2 Fall Detection

To detect fall, instruments and devices are created to identify if the observed object undergoes the action of fall. In human detection of fall, considerations such as the direction of fall, the sections of the body that can help to consider fall, the types of fall and the way how to analyze fall must be considered.

3.2.1 Motion Detection and Direction of Fall

The orientation of the fall is one of the components on analyzing fall. The objects fall direction whether it is falling or had fallen in the directions such as forward, backward or sideward: right or left, is an important factor to identify fall. With the use of the triaxial-accelerometer, the direction of fall can be identified. The tri-axial accelerometer can acknowledge orientations with respect to the earths gravity.

Specific motions can also be acknowledged by this device. Static motions such as sitting and standing and dynamic motions such as walking and falling are motions that this device can evaluate. Using the same module, the accelerometer can measure motions. Static and dynamic motions can be recognized by the accelerometer thru analyzing the consistency and rapidness of the changes in motions.

3.2.2 Location of the User

The location of the user can be known using the GPS module which is included in this device. The users location is being recorded each time as the device is together with the user. After the fall, the coordinates of the user where s/he has fallen is included on the data to be sent to the defined responder/s so that as the responder/s receives the notification, the user can be easily located. Using the ability of the GPS module to identify the exact coordinate, time and date, added information as such can be helpful to the responder in dealing with the fall.

3.2.3 Alert System

Using the GSM module, alert system via SMS is being implemented. At the moment the user has fallen, and the device has recognized the fall, using the said module information and notification alert will be sent to the defined responder/s. On the other hand, when the user has fallen and chooses not to initiate the alert system, it can be overriden by pressing a specific button that cancels the alert. This is done to avoid false alarm and to give the user a choice whenever the fall does not result to a heavy damage.

3.2.4 Graphical User Interface (GUI)

The GUI makes the personalization setting or initialization of the device - which includes the information of both the user and responder as well as the retrieval of recorded activities or motions before, during and after the fall - easy. With the use of the GUI, the initiating and retrieval became user friendly where the recorded data can be easily found and understood.

3.2.5 Sections of the Body

Another factor to consider in fall detection is the sections of the body. These must be considered to identify which part of the body does the sensors better detect motions such as walking, sitting, standing, lying and most importantly falling. To detect fall, sensors are introduced into specific areas of the body, particularly the hip and thigh. Through these parts of the human body, motions mentioned above can be easily analyzed because of the specific movement that these two create whenever the subject is moving.

Fig. 3.1 Device on the user

3.2.6 Trip, Stumble, and Fall

In many descriptions, fall can be defined as the moment of having tripped, stumbled, then fell. The device which was created to recognize falls detects all of these descriptions if the subject fell flatly on the ground after such events. Generally, those descriptions mentioned above became a sub-definition of fall. Tripping and stumbling will not be recognized if a fall does not occur after those scenarios.

3.2.7 Vector Analysis of Fall

From the sensors data, not only the direction of fall or fall itself will be analyzed. Through vector analysis of the data, other motions such as standing, sitting walking and lying will also be analyzed as a sub-feature of the device. The magnitude of acceleration follows (3.1), where a stands for the acceleration while x, y and z are values detected by the tri-axial accelerometer.

Vector of cosines (3.2 to 3.4), are used to detect walking. The angles are used to determine the difference of the wideness of the swing of the leg during the said motion. As two types of motions have been categorized as static and dynamic motions where they are defined as motions that are in constant pattern and motions that are rapidly changing respectively, averaging algorithm (3.5) is needed to accurately predict and determine static motions in which will be the basis in identifying dynamic motions as well. Lastly, errors must be computed (3.6) to determine the reliability of the measured data.

$$a^2 = x^2 + y^2 + z^2 \tag{3.1}$$

$$cosx\theta = \frac{x}{magnitude} \tag{3.2}$$

$$cosy\theta = \frac{y}{magnitude} \tag{3.3}$$

$$cosz\theta = \frac{z}{magnitude} \tag{3.4}$$

$$StatisticValue - \frac{Mean}{Average} \tag{3.5}$$

$$\left| \frac{Present - Past}{Present} x100 \right| \tag{3.6}$$

3.3 Pre, During and Post Detection of Fall

The device will analyze data before, during and after the fall happens. This will be recorded and will be used to analyze data.

3.3.1 Pre-fall Detection

On the pre-fall evaluation, the device will analyze the movement of the user before the fall happens. These movements can be classified as standing, walking, sitting and lying. These data can be used as a sub-feature of the device and can be used by the responder/s to know the previous activity of the user before the fall happened.

3.3.2 Post Fall Detection

After the fall, the device will send emergency messages to the responder/s that are recorded in the database. The device will send the location and of the user where the subject has fallen and the time they fell.

Chapter 4

DESIGN CONSIDERATIONS

Contents

De La Salle University

4.1 Summary

This chapter discusses the consideration on making the design of fall analyzer. This chapter contains the discussion on the software and hardware that were used in creating the fall analyzer device.

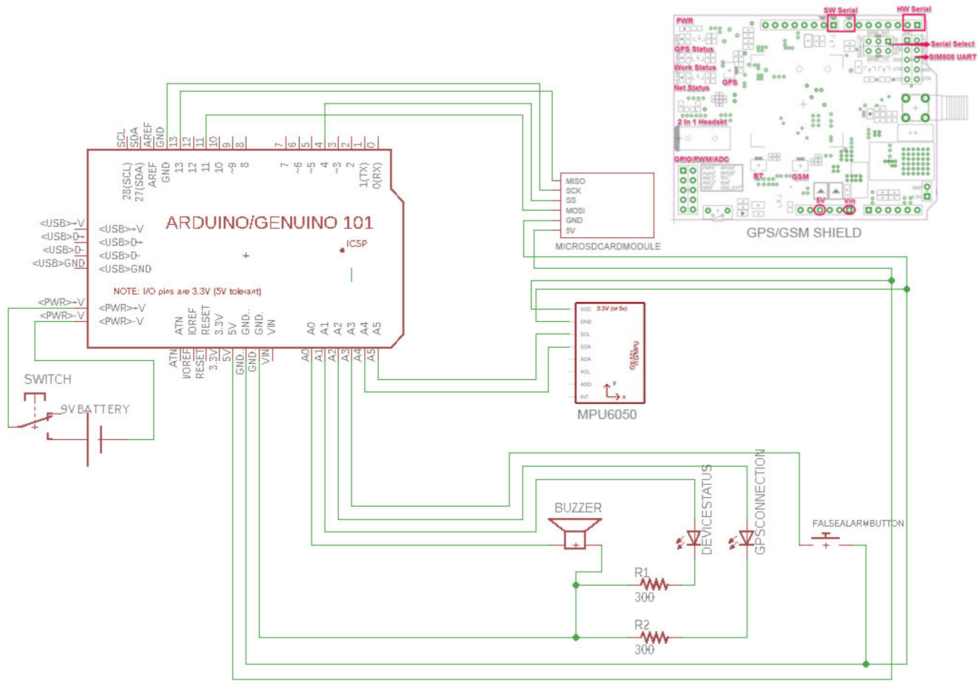


Fig. 4.1 Schematic Diagram of the Device

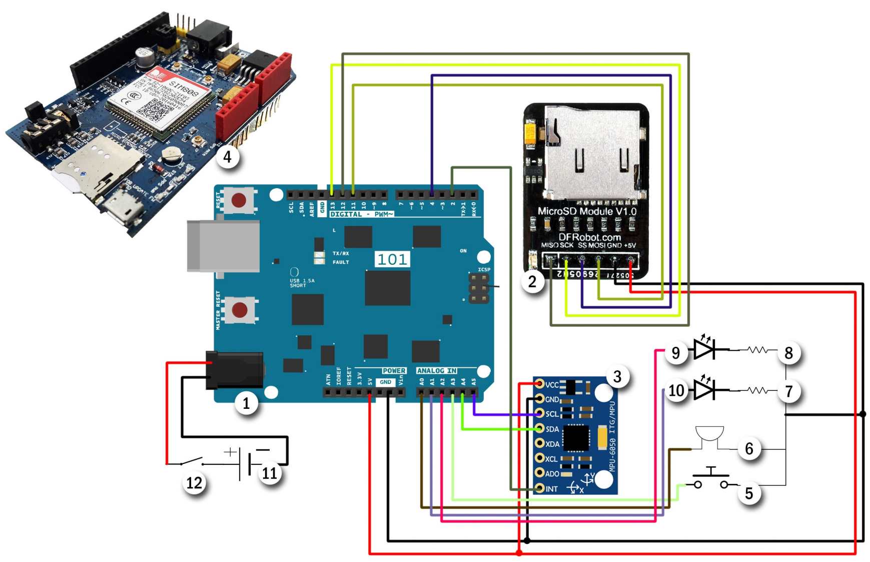


Fig. 4.2 Fall Analyzer Circuit Design

4.2 Development of Fall Detector

In Figure 4.2, the design of fall analyzer device is shown. It is composed of the combined modules such as Arduino 101 board, MicroSD Card Module, MPU6050 Accelerometer, GSM/ GPS Shield, false alarm button, buzzer, and power module. From Figure 4.2, the legends are as follows: (1) Arduino 101 board, (2) MicroSD Module, (3) MPU6050 Accelerometer, (4) GSM/GPS Shield, (5) False Alarm Button, (6) Buzzer, (7) & (8) 300 Ohm resistor, (9) GPS status, (10) Deice status, (11) 9V DC Battery and (12) Power switch.

Figure 4.3 shows the MPU6050 accelerometer and it is connected to the Arduino 101 with the connection as follows: Collector Supply Voltage (VCC) pin uses 5v pin of the Arduino and ground wiring is connected to Ground (GND) pin. The Clock Line (SCL) is connected to A5 while the Data Line (SDA) is to A4 and Interrupt (INT) pin is to D2.

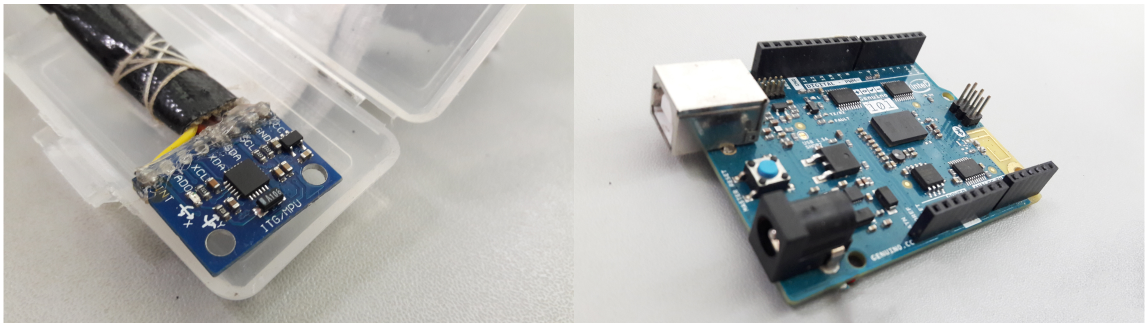


Fig. 4.3 External Accelerometer and Arduino 101

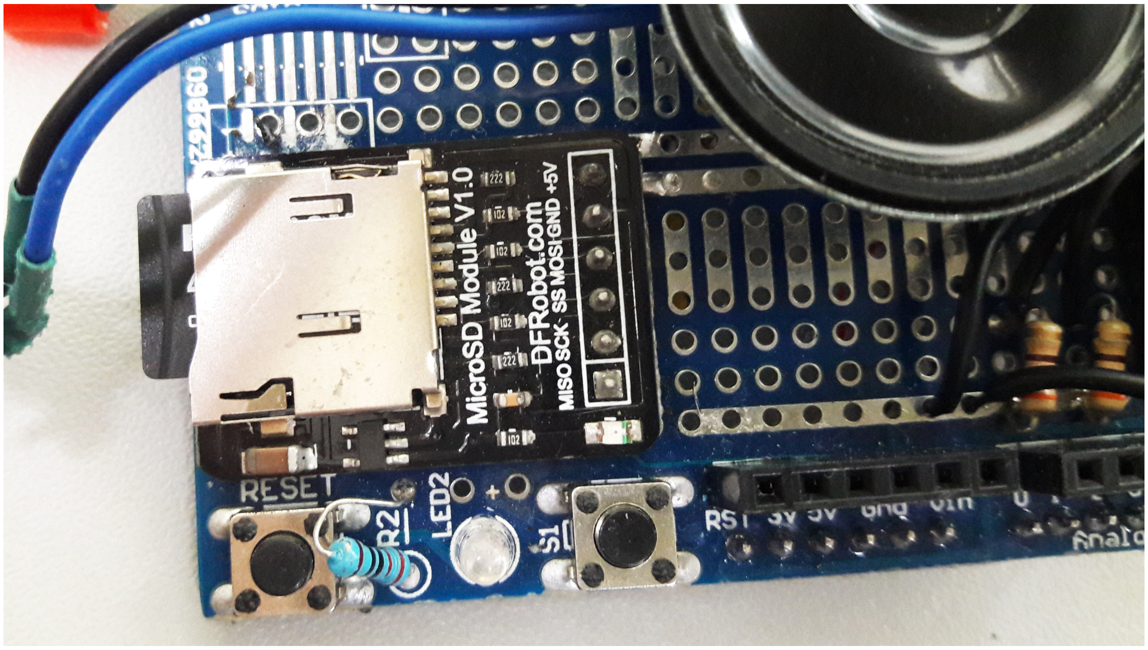


Fig. 4.4 MicroSD Card Module

The MicroSD Module, Figure 4.4, is also connected to the Arduino with the connection as follows: Master In Slave Out (MISO) pin is connected to D12, System Clock (SCK) to D13, Chip Select (SS) to D4, Master Out Slave In (MOSI) pin to D11 with ground to GND and VCC to 5v.

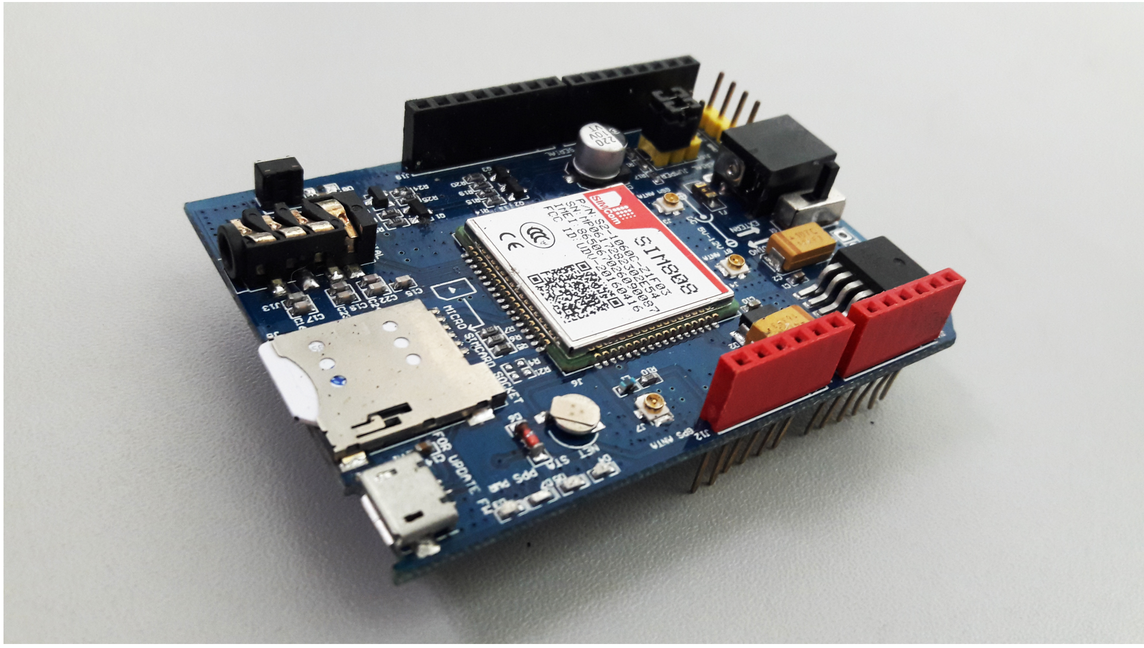


Fig. 4.5 GPS/GSM Shield

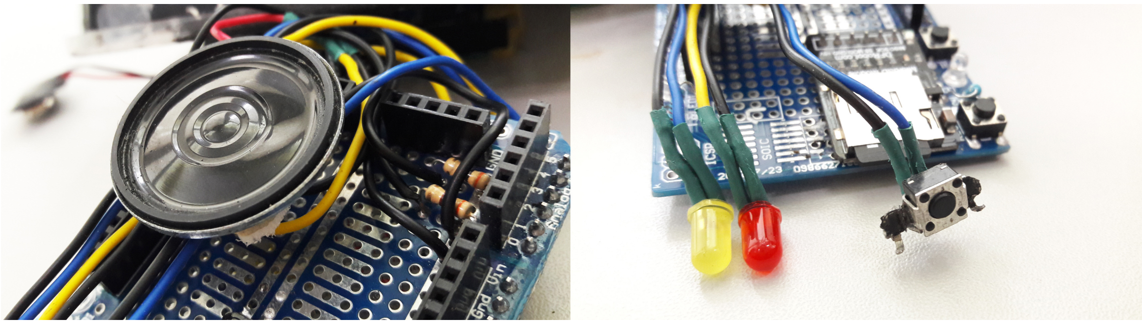


Fig. 4.6 Buzzer, LEDs and Override Button

The GSM Shield in Figure 4.5, is stacked to be connected to Arduino while the Buzzers ground is connected to the Arduinos GND pin and the positive pin of the buzzer to A0 pin of the Arduino. The device status and GPS status indicator using Light Emitting Diode (LED) and false alarm button are connected to the Arduinos A1, A2 and A3 respectively while the ground the three components are connected to GND pin of the Arduino.

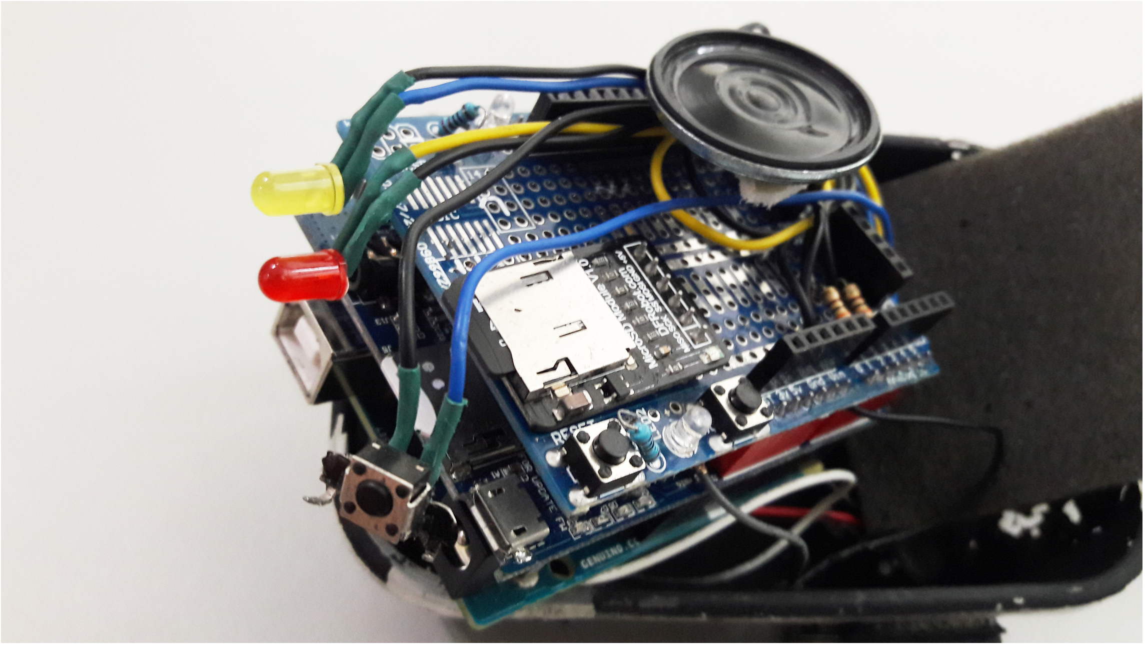


Fig. 4.7 Device Proper

4.2.1 Fall Analyzer and Motion Detection Using Accelerometers

Fall can be analyzed by the help of the built-in accelerometer from the Arduino 101, together with the MPU6050 accelerometer. These accelerometers can measure both static (gravity) and dynamic (motion or vibration) accelerations. [SA, 2018] Accelerometers were used to analyze the acceleration of the user so as the direction of the movement.

Accelerometers are placed on the hip and the thigh of the user. The changes of the values from the result of the movement of the user with the help these accelerometers are computed which was used to identify and recognize whether the user moves statically, dynamically or if the user has fallen. On the other hand, as the user fall, the direction of the fall whether the user fell backwards, sideways or forward was also analyzed by the built-in accelerometer from the Genuino 101.

From Figure 4.8 and 4.9, the flowchart on how the accelerometer works can be seen. The flow chart discusses the flow on how the accelerometer analyze the movements sequentially. The first step is to store the current y-vector to a previous y-vector container then get the new accelerometer inputs. The new inputs will be used to get the magnitude of the accelerations from x, y, z axis. After getting the magnitude, the percentage error between the current and previous values will be calculated. If the error is greater than 1.5%, the magnitude of the current value will become the previous value because there is no consistency, meaning, it is in dynamic motions like walking and falling. Else, it is in static motions like standing, sitting, or lying.

If it is in Dynamic motion, the difference between the current and previous angle of the y-vector will be calculated to determine if the magnitude error is greater than the gravity magnitude. If there is greater force than the gravity acting on the user, the device will check if it is lying position, if yes, the device will set the motion as forward, backward, or sideward fall. If no, the difference angle will be checked if it is in the walking angle range. It will then also check if the acceleration for walking is met. If the two parameters were satisfied, the device will set the motion as walking. If not, the current and previous value will be averaged and will be equal to the new previous value because the error is very minimal that makes it a static motion.

If it is in Static motion, it will check whether the inputs are equal to the standing, sitting, or lying position. If it matches one of the three (3) possible positions, it will set the orientation as one of them, else, it will set the orientation as unknown.

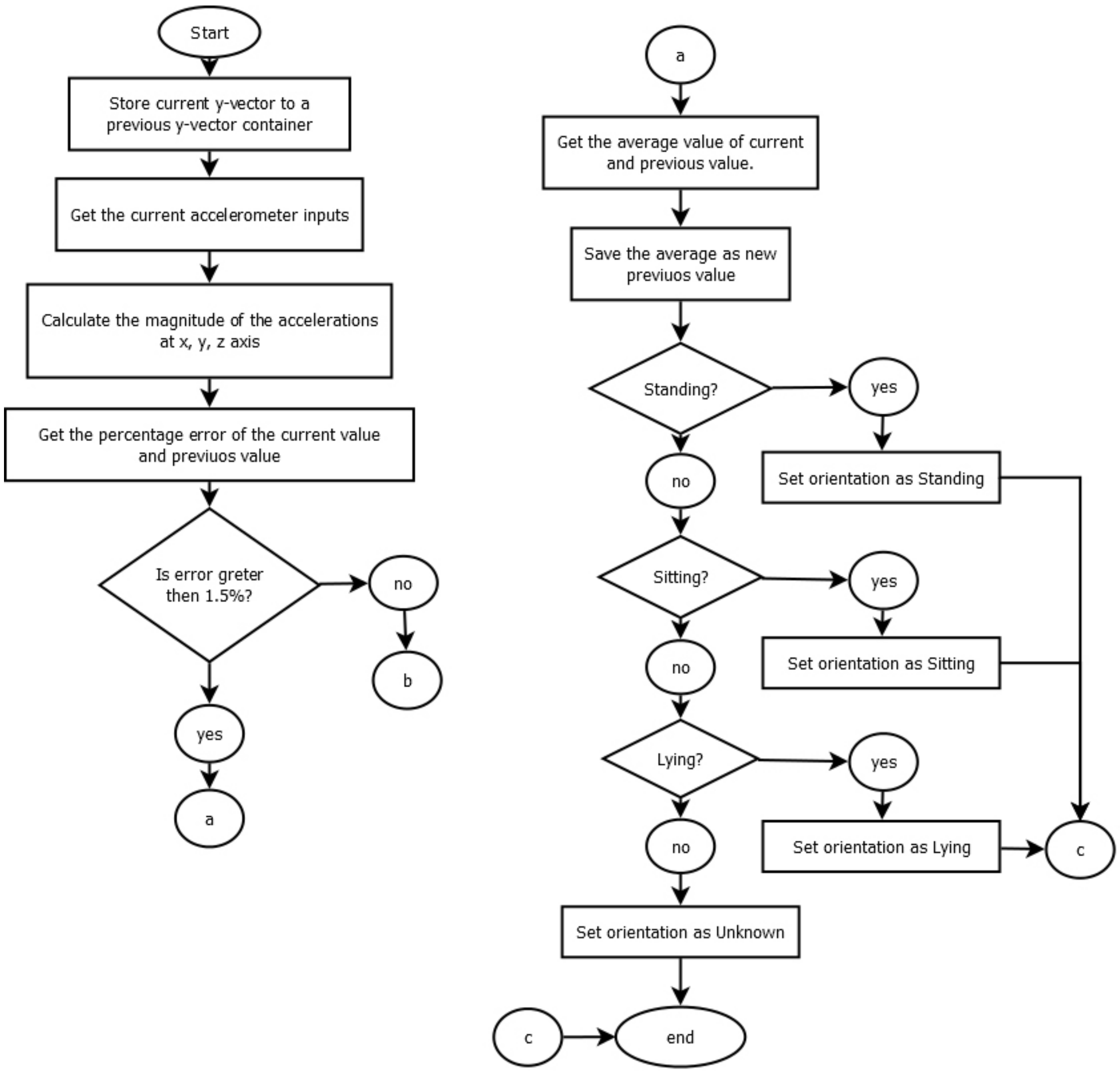


Fig. 4.8 Input Analyzation flowchart (a)

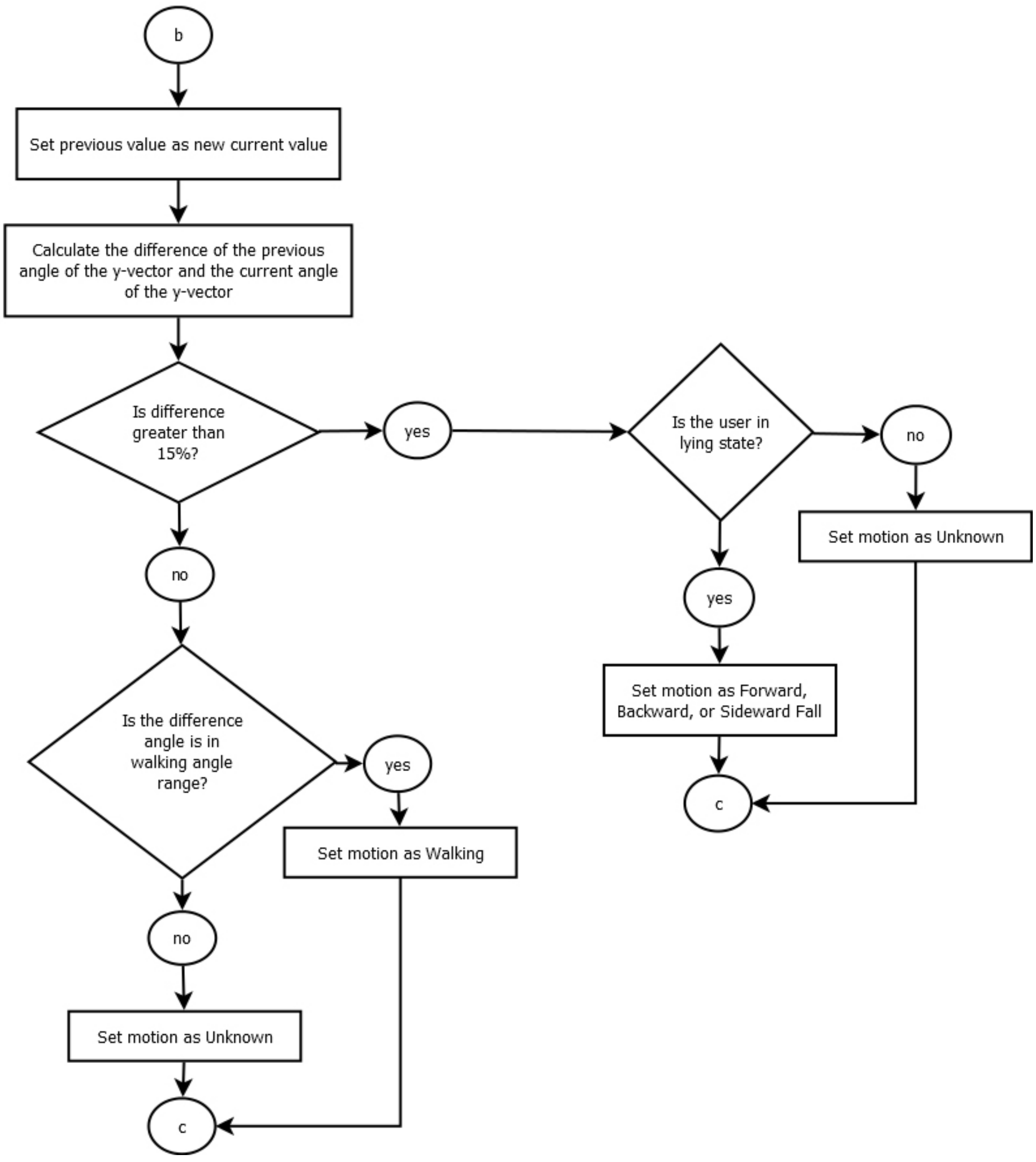


Fig. 4.9 Input Analyzation flowchart (b)

Fig. 4.10 Fall Detection Device

4.3 Languages Used

In creating the fall analyzer device, different languages are used in the codes for better communication and ease of use of the user of the said device. The first language used is Java. Java was used in creating the GUI of the device which was used for a user-friendly interaction of the user and the device. It was also used in creating the initialization page and data view page of the device. The Cascading Style Sheet (CSS) was also used in the GUI for styling the scenes which helps for an ease of interaction of the device to the user. Lastly for creating the logic of the device, C language was used to create an algorithm to the Arduino.

Chapter 5

METHODOLOGY

Contents

5.1 System Process

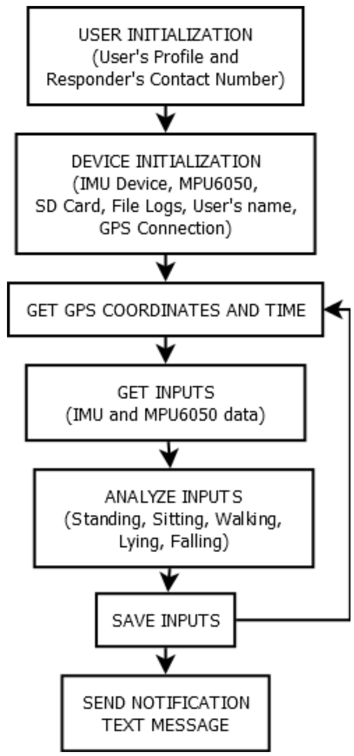


Fig. 5.1 System Block Diagram

As seen in Figure 5.1, the system will have two initialization phases. First is the User Initialization where the user must insert the microSD card into a laptop and access the GUI to set its name, gender, age, address and contact number. The user must also set a username and password that will be used when modifying the activity log, deleting or adding new responders, and updating its profile. The user must input at least one (1) responder for the first initialization phase to be done. The first initialization phase is only done once, only when setting up the system. The second initialization phase will run every time the device is turned on. This is where the device turns on all the required modules connected on it. The second phase also accesses the microSD card to get the information like the username of the user, and file logs. After the initialization, the device will now continuously get the time and GPS coordinates, monitor the motion, and orientation of the user, and save it the data to the microSD card for future reference. When the device detects a sudden fall, it will send notification text message to the responders found in the microSD card if the false alarm button was not pressed after 10 seconds [Kazi et al., 2014]

5.1.1 Device Initialization

The User Initialization phase, which is run in the GUI, has four (4) steps. The user must first set its username and password. The system also asks for a confirmation password before proceeding to the next step. If the confirmation password and the password did not match, the user must re-enter it again. The second step takes the users first and last name, gender, age, address, and contact number. The user must enter a valid age and contact number or else, it cannot proceed to the third step. The third step is where the user sets its responders name and contact number. It must set at least one (1) responder or else, it cannot proceed to the last step. The last step for this initialization is to re-enter the username and password of the user, if it matches, the initialization is done, and the user must now attach the microSD card to the device.

5.1.2 Notification System

Figure 5.3 shows the system flowchart in which the notification system is based. From the flowchart, the device continuously gets the time and users GPS coordinates, get the users orientation and motion, and save it to the microSD card. If the device detects a sudden fall motion, it will start a 10 seconds timer [Kazi et al., 2014] If the user did not press the false

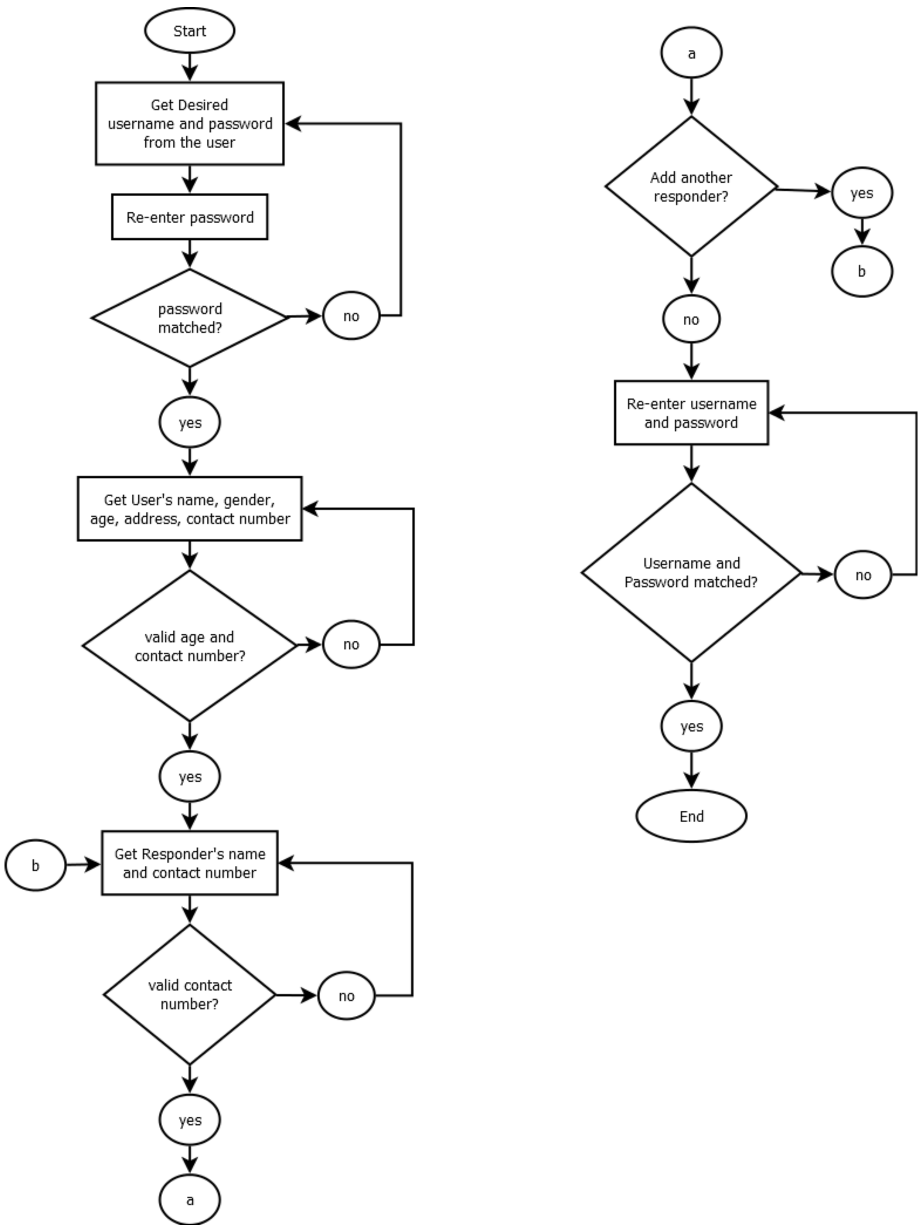


Fig. 5.2 User Initialization

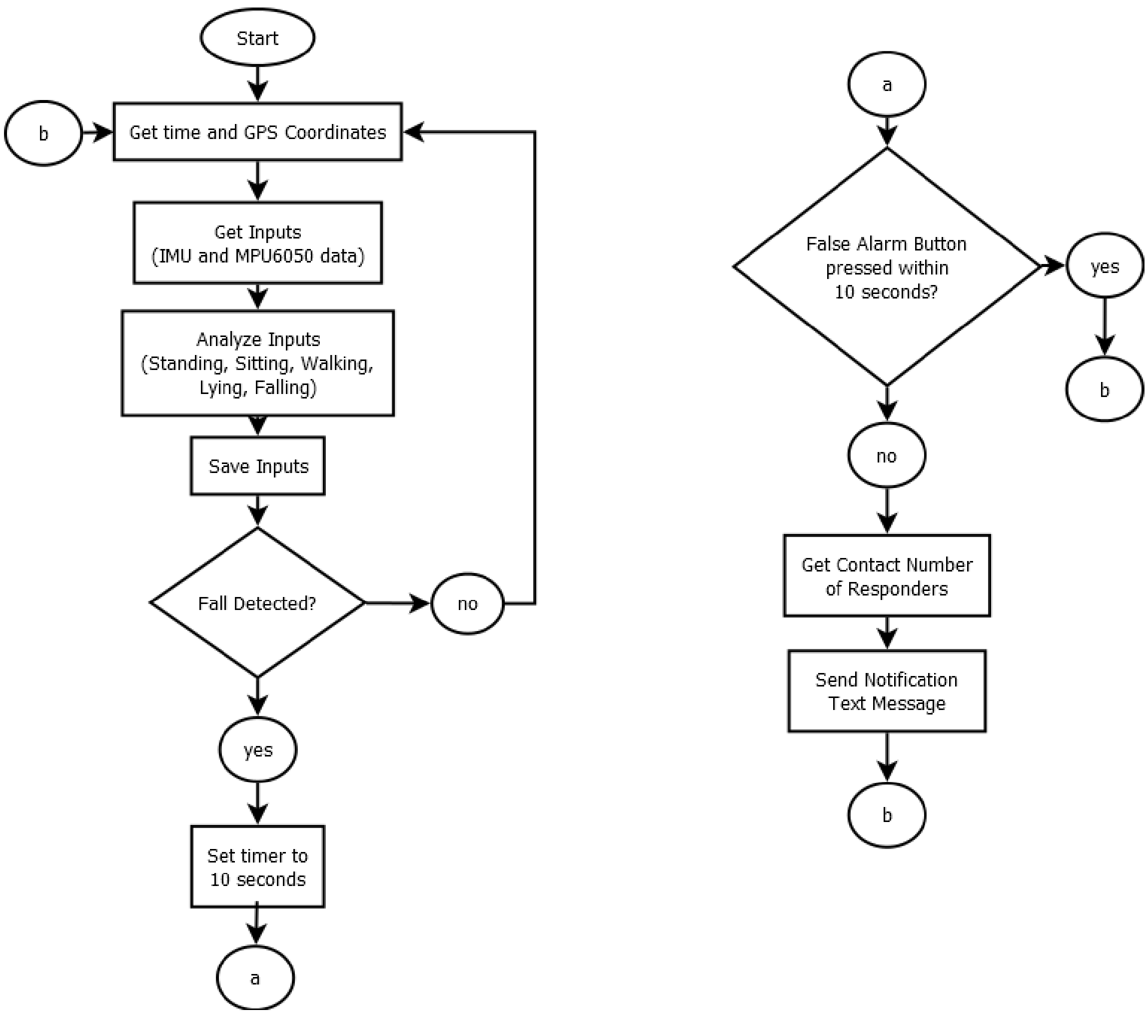


Fig. 5.3 System Flowchart

alarm seconds within 10 seconds, the device will get the contact number of the responders and send them a notification text message that includes the name of the user, time when sudden fall is detected, and its GPS coordinates.

5.2 Implementation

The project follows the modified waterfall model, because each objective is important to proceed with the other objectives. The devices process undergoes six phases and is also goal oriented. This makes each process to have an exact goal for each phase and will not be dependent on a concluding result.

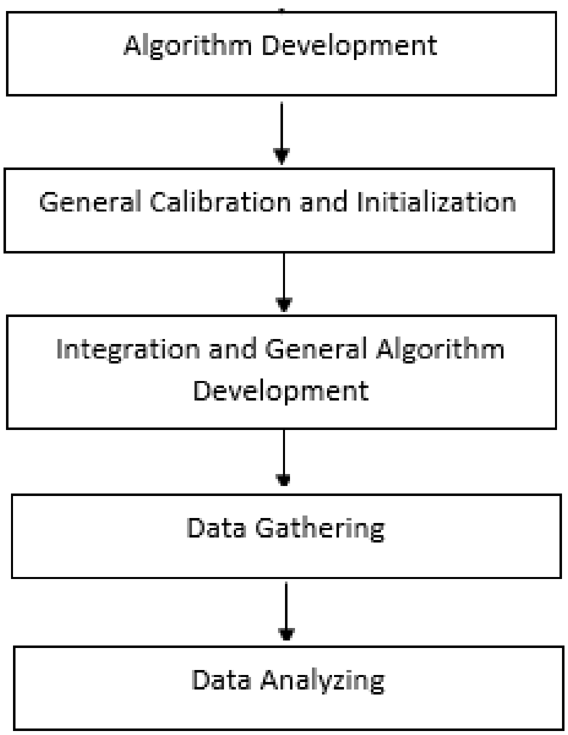


Fig. 5.4 Waterfall Pattern

5.2.1 Preparation, Setup, and Data Acquisition

Phase one Algorithm development:

When the result is already satisfied, the main algorithm will be created to detect the posture and movements of the user.

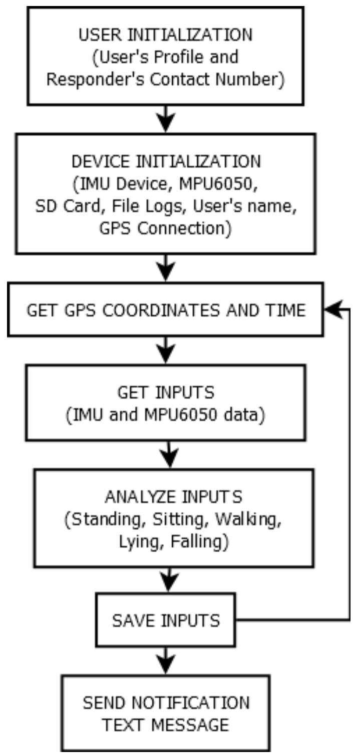


Fig. 5.5 Block Diagram

As seen in figure 5.5, the device should be initialized first. In initialization, the user must connect the microSD Card first to a computer/laptop to set the users name, age, and address. The user must also set the contact details of the responder/s, specifically, their names and contact numbers. After the initialization, the device will then continuously monitor the motion and orientation of the user and analyze it to determine whether the user needs an assistance. The device will consider that the user needs an assistance if it detects that the user had a sudden fall and did not respond after 10 seconds. If the device detects that the user had a sudden fall, the device will automatically send a text message to the set responders.

Phase two Integration and General Algorithm development:

A system will be developed between different modules and sensors. The system will be shown in the development of the project on how the different components will react to each other.

After the initialization process, the device will continuously monitor and save the user's orientation and motion, until the device discovered a sudden fall. After the device detected that the user had a sudden fall, it will begin 10-second countdown. If the user didnt hit the override alarm button within 10 seconds, it will get the users GPS coordinates and will automatically send a text message to set responders stating its current GPS coordinates, and get another input from the users body again.

Phase three General Calibration and initialization:

Accelerometers will be calibrated manually by adjusting the code of the sensors. Filters will be applied to improve the accuracy and remove unwanted noise. GPS technology of the project will also be tested.

SMS technology of the project is also tested by sending a message to a receiver. The project will be tweaking the built-in Real Time Clcok (RTC) of the GPS/GSM shield to put timestamps on each activity of the user. Accelerometer will be used to analyze fall and the direction of fall such as forward, backward and sideward with 80% of accuracy upon detection. An algorithm will be created which will allow the data to gather to be saved to the SD Card.

Phase four Data Gathering:

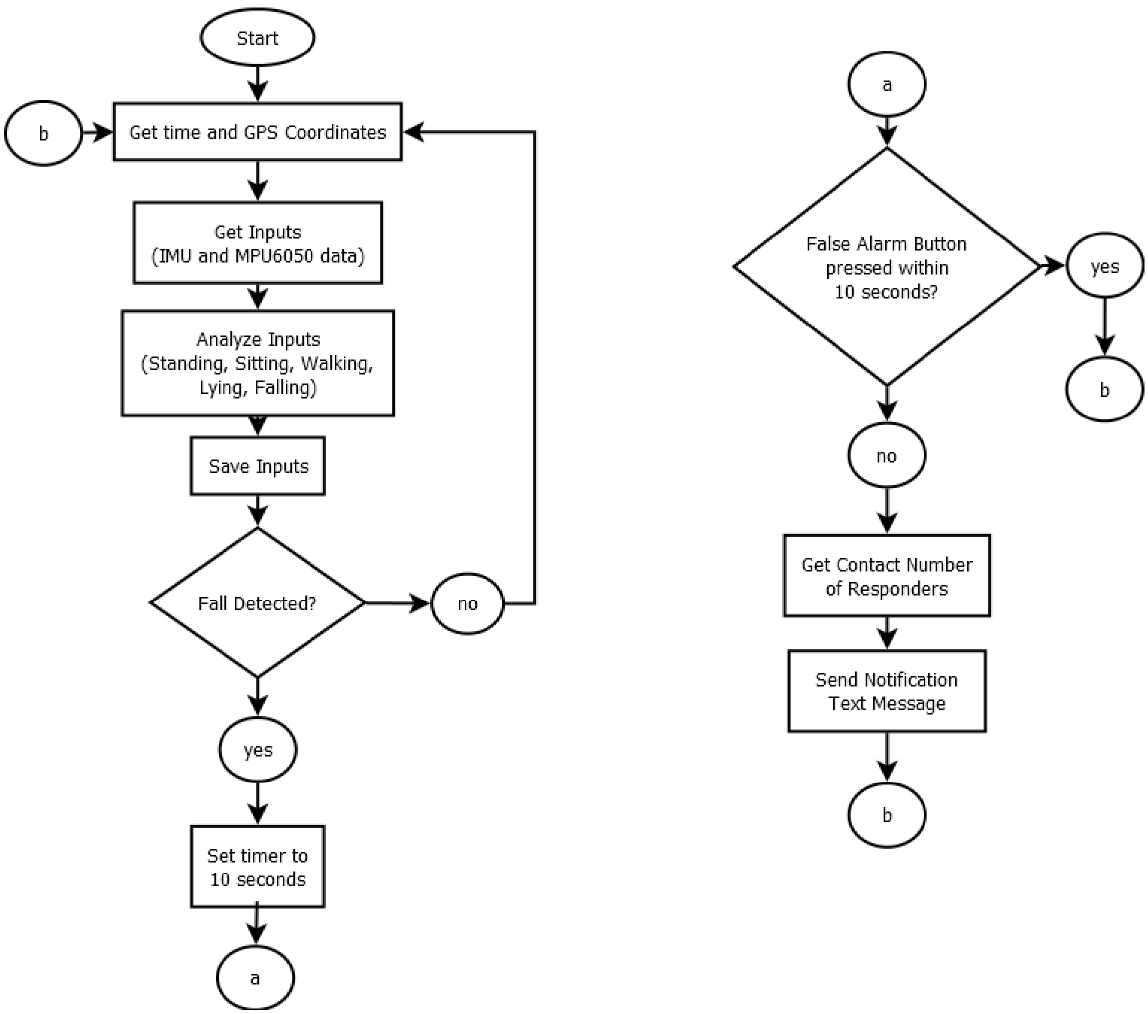


Fig. 5.6 Flow Chart

Thirty people are needed on this phase. Following the statistical standards, 30 people will serve as the minimum number of data samples to gather. These participants are being used to identify the devices working capability as it gathers information about the fall and motion of the subjects which can be further used in many applications in the future especially in the medical field upon innovation of the device. Each person will be given an activity such as sitting, standing, walking, lying and falling. These activities will be recorded using two accelerometers that are attached on the upper body and lower body of the subject. The accelerometers will gather the accelerations, and direction of the body upon falling.

5.3 Evaluation

Phase five Data Gathering:

The collected data can be retrieved by the responder/s. This data includes the activities done by the user before the fall happened. Activities done by the user such as walking, standing, sitting and lying can be seen to better analyze the motion activity of the user before the fall. The data can also show the time of each movement so as the location and direction of the fall. From this data, the users time and location of fall was retrieved by the device and was used in sending a detailed notification system to the registered responder/s onto the device.

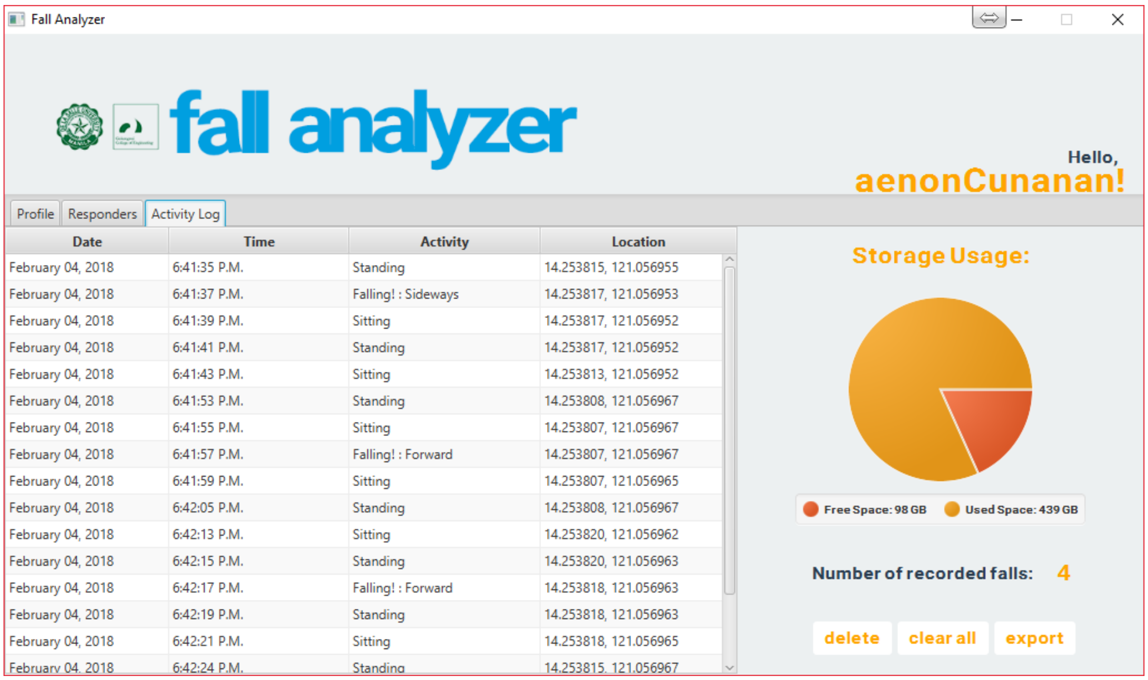


Fig. 5.7 Activity Log

Chapter 6

RESULTS AND DISCUSSIONS

Contents

6.1 Direction of Fall Output

Falling motion is the main concern of this project. With the falling motion, detection of fall and its direction are to be considered to call the device a fall analyzer. From the experiments data, gathered over 30 respondents, participants are asked to do a falling motion with three classifications of fall based on the direction namely: forward fall, side fall and backward fall. Results are shown from the table 6.1

From Table 6.1, the device shows that it can detect fall and the direction of fall. Within 30 participants, the device shows that it can detect forward fall with 96.66% accuracy that resulted from a 29 out of 30 accurate forward fall results. On the other hand, backward fall resulted in 93.33% of backward fall accuracy detection and 86.67% for side fall accuracy detection. Overall fall detection gives a reliability of 92.22% accuracy in detection such motion.

The data shows that the device can clearly recognize forward fall with only 3.33% error. This is because the device recognizes only such direction and motion that does not have any other alike motion that insist a moving forward action. The backward fall follows with 6.67% rate of error. The device sometimes recognizes backward fall as just a lying motion if the user fell in a slow acceleration. As lying position induces a backward motion, the acceleration of the user will be the only boundary that separates the two motions in which if the device recognizes motions that are exactly in between the two, the 6.67% error of backward fall motion comes to reason. Lastly, sideward fall came with 13.33% error which has the highest error among the three falling motions. This is because of the devices sensitivity in recognizing the fall. As the user falls sideward, the device should recognize a sideward fall motion however as the user hits the ground, the algorithm computes the

TABLE 6.1 FALLING MOTIONS DATA

Sample #	Falling Motions		
	Forward	Backward	Sideward
1	Lying	/	/
2	/	/	/
3	/	/	/
4	/	/	/
5	/	/	/
6	/	/	/
7	/	/	/
8	/	/	/
9	/	/	/
10	/	Lying	/
11	/	/	/
12	/	/	/
13	/	Lying	/
14	/	/	/
15	/	/	/
16	/	/	/
17	/	/	/
18	/	/	Backward Fall
19	/	/	Lying
20	/	/	/
21	/	/	Lying
22	/	/	/
23	/	/	/
24	/	/	/
25	/	/	/
26	/	/	Lying
27	/	/	/
28	/	/	/
29	/	/	/
30	/	/	/

last position of the user as they fell basing from the initial position before they fell. In such case, if the user falls sideways, but hits the ground backside, the device recognizes a backward fall motion because of the last position of the user is will be in a slight facing up position. This makes the side fall detection the most sensitive of the three directions due to the position to consider after the user has fallen.

6.2 Motion Detection Data

Detection of other motions other than the falling motion is a sub-feature of this device. These motions to be detected are divided into two categories which are static motions and dynamic motions. Static motions such as sitting, standing and lying can be detected by this device so as the walking motion under the dynamic category. Table 6.2 shows the result of the device's capability on recognizing such motions with experimenting the devices ability to identify the mentioned specific motions with 30 participants as a subject.

From table 6.2, specific motions data such as standing, sitting, lying and walking can be seen. A 100% reliability rate was achieved by the device in recognizing motions such as standing, sitting and walking. Because these motions do not interfere with any other motions that this device can detect, the complete success rate on motion recognition was achieved, however in detecting the lying motion, only 90% accuracy rate was reached. This is because the device sometimes recognizes lying motion into a falling motion or in other cases it stops on the detection of the previous motion before the lying motion, just like the result on the table shows.

TABLE 6.2 MOTION DETECTION DATA

Sample #	Static Motions			Dynamic Motion
	Standing	Sitting	Lying	Walking
1	/	/	/	/
2	/	/	/	/
3	/	/	/	/
4	/	/	Walking	/
5	/	/	/	/
6	/	/	/	/
7	/	/	/	/
8	/	/	/	/
9	/	/	Backward Fall	/
10	/	/	/	/
11	/	/	Standing	/
12	/	/	/	/
13	/	/	/	/
14	/	/	/	/
15	/	/	/	/
16	/	/	/	/
17	/	/	/	/
18	/	/	/	/
19	/	/	/	/
20	/	/	/	/
21	/	/	/	/
22	/	/	/	/
23	/	/	/	/
24	/	/	/	/
25	/	/	/	/
26	/	/	/	/
27	/	/	/	/
28	/	/	/	/
29	/	/	/	/
30	/	/	/	/

6.3 Confusion Matrix

Overall, falling and motion detection data can be seen in table 6.3. The table shows the confusion matrix of all motions including the direction of the falling motion. From the table, it shows that the overall reliability of the device in detecting motions such as walking, standing, sitting, lying, forward fall, backward fall and side fall is 95.23% which makes the error of the device be only at 4.76% only.

TABLE 6.3 CONFUSION MATRIX

						Falling			
		Walking	Standing	Sitting	Lying	Forward	Backward	Side	
	Walking	30	0	0	1	0	0	0	
	Standing	0	30	0	0	0	0	1	
	Sitting	0	0	30	1	0	0	0	
	Lying	0	0	0	27	1	2	4	
Falling	Forward	0	0	0	0	29	0	0	
	Backward	0	0	0	1	0	28	1	
	Side	0	0	0	0	0	0	26	Overall Reliability
Reliability		1	1	1	0.9	0.966667	0.933333	0.866667	0.952380952
	in Percentile	100%	100%	100%	90%	97%	93%	87%	95%

6.4 Notification System

As the user fell, a notification system was activated. Using GSM technology, an SMS was sent into the recorded responder/s. In the message, the name of the user, location where the fall happened, date of fall and time of fall were included into the notification message.

Figure 6.1 shows the notification messages to two different responders. Responder 1 uses sim1 and responder 2 uses sim 2. As can be seen from the image above, the device has sent the users name and details such as location, date and time after he fell. Notice that

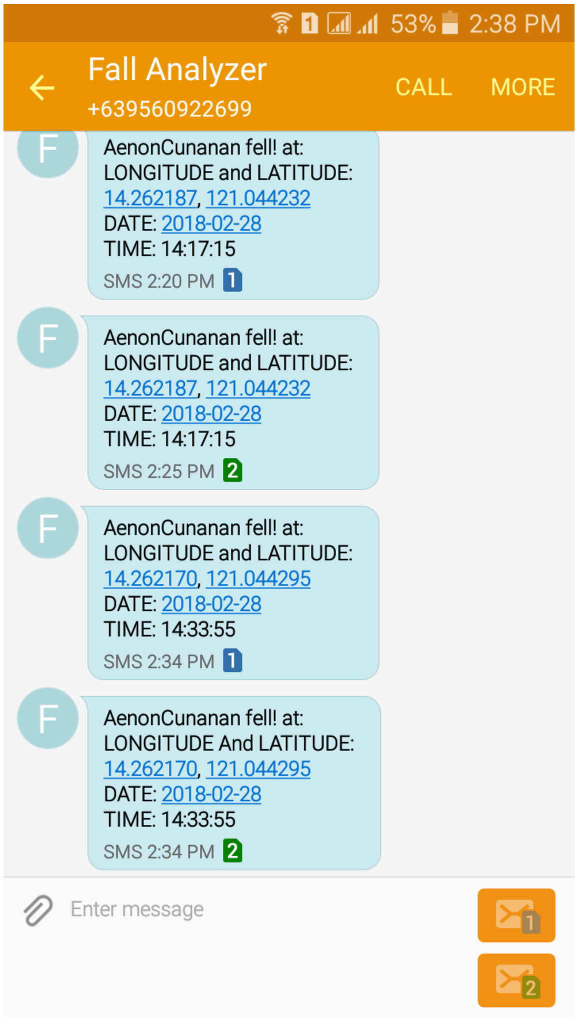


Fig. 6.1 Notification Message

the device sent the message to the two responders approximately 1 minute after the fall. The location of fall was presented into coordinates and on Figure 6.2, the sent coordinates can be found on the google maps to be located at De La Salle University Manila Laguna Campus. However, when the user has fallen and decided to hit the false alarm button in 10 seconds, the message notification system was override and message was not sent.

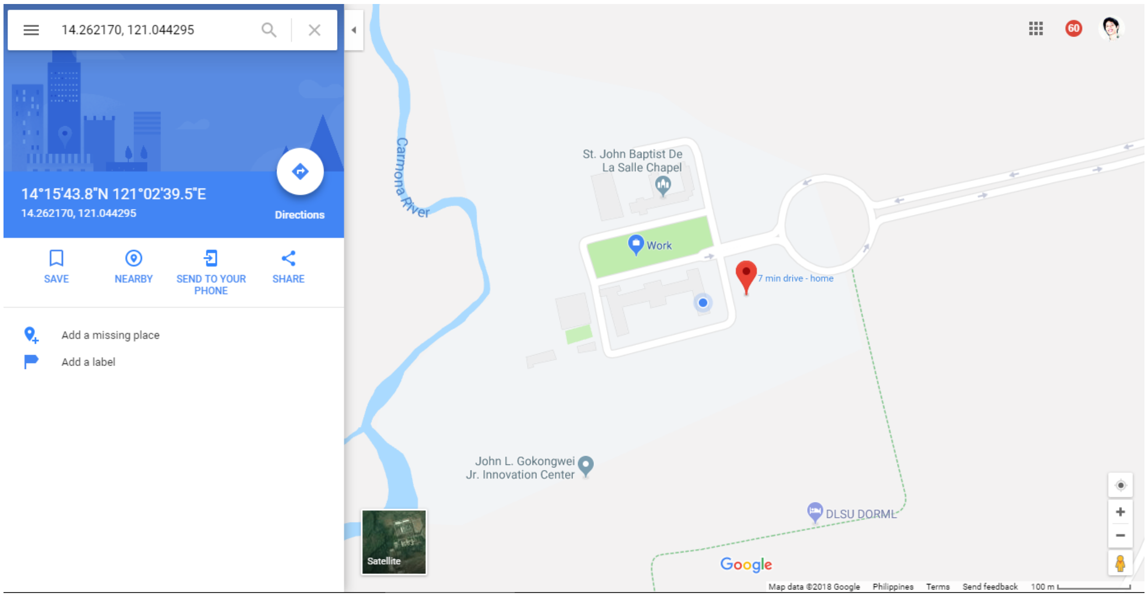


Fig. 6.2   Location of Fall from Google Maps

Chapter 7

CONCLUSIONS, RECOMMENDATIONS, AND FUTURE DIRECTIVES

Contents

7.1 Concluding Remarks

Designing a device that can detect motion of user such as standing, sitting, lying, walking and falling with direction detection of fall whether the fall is in forward, sideward or backward motion was successfully developed with 95% of the devices accuracy of detection. When falling motion was established, the notification system of the device was activated, and message was sent to the recorded responder/s with details such as location of fall, time of fall, name of user who has fallen and the date of fall.

The use of two accelerometers located at the hip and thigh resulted as a good component in making a motion detection device with 92.22% reliability. The development of falling device that can recognize direction of fall with the use of the same two modules resulted with 92.22% of accuracy which supports the conclusion of the detection of the users orientation of fall whether the user has fallen backward, forward or sideward to be achieved. The GPS module helps to get the time and date of the users fall.

Time, date, location, and motions that were detected since the device was activated were also saved on a database through SD Card. Stored data were retrieved through the GUI that interacts with the user as it was created with user-friendly functions and directions.

Motion detection such as standing, sitting, walking and lying that are sub-feature of this device was successfully detected with 97.5% overall accuracy rate. As a general conclusion, the design and development of fall analyzer with specific motion detection which can be personalized with the function of send notification message to recorded responder/s during the fall was achieved with above 80% acceptable accuracy rate.

7.2 Contributions

The synthesis of all the contributions that this thesis has made and developed are as follows:

- Fall detection with the recognition of fall direction whether backward, sideways or forward.
- Detection of motions such as standing, sitting, lying and walking with data logging of such motions which can be accessed by the responder/s thru GUI.
- Notification and alarm system via GSM with GPS location, and time data included on the responders notification message.

7.3 Recommendations

For future innovation of the study, detection of motions along a not plain terrain like stairs and alike is recommended. Power management of the device must also be improved for the further improvement of this study as this project limits itself on the detection. The location of the user when fallen in this project can be identified using the GPS coordinates, so for further study, the use of exact address in locating the user is suggested. Lastly, a training process of the devices system can be added for better detection of motions for every different user.

7.4 Future Prospects

In further accretion of this device, the use of fall and motion detection can be used in medical field. This can serve as a monitoring device to patients like the elderly who might have suffer fall related injuries. Using this technology, accidents related to fall can be prevented and can be taken care of as quickly as possible. Using the real-time notification system with the included time, location and might include the previous activity of patient before fall, a monitoring device such this can be helpful.

Another application for future use of this device is for manpower activity detection. Due to the location register and activity log of the device, it can detect the activity of the user e.g. in the shopping mall, the device can be used by the salesperson and the manager can detect whether his/her employee is working or in his/her assigned work location.

REFERENCES

[Mar, 2011] (2011). A profile of deaths among trauma patients in a university hospital: The philippine experience.

[Cri, 2014] (2014). Fall accidents 5th leading cause of phl kids' deaths.

[Circuitrocks, 2018a] Circuitrocks (2018a). Gsm gps gprs sms sim808 crowtail.

[Circuitrocks, 2018b] Circuitrocks (2018b). Imu 10dof mpu60050 + hmc5883l + bmp085.

[Circuitrocks, 2018c] Circuitrocks (2018c). Sd card socket module.

[e Gizmo Mechatronix Central, 2018a] e Gizmo Mechatronix Central (2018a). 300 ohms resistor.

[e Gizmo Mechatronix Central, 2018b] e Gizmo Mechatronix Central (2018b). Battery clip for 9v.

[e Gizmo Mechatronix Central, 2018c] e Gizmo Mechatronix Central (2018c). Led 10mm.

[e Gizmo Mechatronix Central, 2018d] e Gizmo Mechatronix Central (2018d). Piezo buzzer.

[e Gizmo Mechatronix Central, 2018e] e Gizmo Mechatronix Central (2018e). Rocker switch.

[e Gizmo Mechatronix Central, 2018f] e Gizmo Mechatronix Central (2018f). Sswspst2p.

[Fotune et al., 2014] Fotune, E., Lugade, V. A., and Kaufman, K. R. (2014). Posture and movement classification: The comparison of tri-axial accelerometer numbers and anatomicl placement. pages 1–8.

[Guevarra and Evangelista, 2010] Guevarra, D. G. and Evangelista, T. J. P. (2010). The incidence and identification of risk factors for falls among filipini elederly person in a nursing home facility and at the rehabilitation medicine out-patient department of the philippine general hospital. *ACTA MEDICA PHILIPPINA*.

[Huynh et al., 2013] Huynh, Q. T., Nguyen, U. D., Tran, S. V., Nabili, A., and Tran, B. Q. (2013). Fall detection system using combination accelerometer and gyroscope.

[Igual et al., 2013] Igual, R., Medrano, C., and Plaza, I. (2013). Challenges, issues and trens in fall detection systems.

[Intel, 2018] Intel (2018). Arduino 101.

[Jovanov et al., 2005] Jovanov, E., Milenkovic, A., Otto, C., and Groen, P. C. D. (2005). A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation.

[Kazi et al., 2014] Kazi, S. B., Sikander, S., and Yousafzai, S. (2014). Fall detection using single tri-axial accelerometer.

[Koray et al., 2014] Koray, Gurkan, Gurkan, G., Dindar, A. A., Akpinar, B., and Gulal, E. (2014). Wireless 3-axis acelerometer system for measuring of structural displacement. pages 403–406.

[Li et al., 2009a] Li, Q., Hanson, M. A., Stankovic, J. A., Barth, A. T., and Lach, J. (2009a). Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information.

[Li et al., 2009b] Li, Q., Stankovic, J. A., Hanson, M., Barth, A., and Lach, J. (2009b). Accurate, fast fall detection using gyroscope and accelerometer-derived posture information.

[Park and Jayaraman, 2003] Park, S. and Jayaraman, S. (2003). Enhancing the quality of life through wearable technology. pages 41–48.

[Press, 2017] Press, C. U. (2017). Fall.

[SA, 2018] SA, S. C. (2018). Mems accelerometers.

[Sudarshan et al., 2013] Sudarshan, B. G., Raveendra, H., Prasanna, K. C., and Satyanarayana, B. S. (2013). Design and developmet of fall detector using fall acceleration. pages 57–61.

[Wu et al., 2014] Wu, F., Zhao, Y., Zhao, H., and Zhong, H. (2014). Development of a wearable-sensor-based fall detection system. *International Journal of Telemedecine and Applications*, page 11.

Appendix A
STUDENT RESEARCH ETHICS CLEARANCE

De La Salle University

RESEARCH ETHICS CLEARANCE FORM[1] For Thesis Proposals
Names of Student Researcher(s): SAMPLE ONLY Dela Cruz, Juan Z.
College: Gokongwei College of Engineering
Department: Electronics and Communications Engineering
Course: PhD-ECE
Expected Duration of the Project: from: April 2015 to: April 2017
Ethical considerations None *(The [Ethics Checklists] may be used as guides in determining areas for ethical concern/consideration)*
To the best of my knowledge, the ethical issues listed above have been addressed in the research. Baltasar Dr. Francisco D. Baltasar ________________________________ **Name and Signature of Adviser/Mentor:** **Date:** April 8, 2017
Noted by: R Sison Dr. Rafael W. Sison ________________________________ **Name and Signature of the Department Chairperson:** **Date:** April 8, 2017

[1] The same form can be used for the reports of completed projects. The appropriate heading need only be used.

Appendix B
ANSWERS TO QUESTIONS TO THIS THESIS

Appendix C
REVISIONS TO THE PROPOSAL

TABLE C.1 SUMMARY OF REVISIONS TO THE PROPOSAL

Panelist name	**Comment**	**Summary of how the comment was addressed**	**Locations**
Dr. Francis Roy Navea	• Definition of Fall • Change the title of the thesis • Specify the other motions of the device to be detected	• Fall was defined on the paper as an act of falling, tripping or stumbling to specify that fall on this paper discusses not the free fall motion from a high ground to a different level of grounds. • Title was changed from Development of Wearable Device for Fall Detection Using Accelerometer with Automated Notification via GSM to Design and Development of Fall Analyzer • Motions are specified and set to be walking, standing, sitting and lying	Sec. 1.3 on p. 4, Sec. 1.4 on p. 6
Engr. Noriel Mallari	• Be specific on the objectives and include the reliability of the device • Remove the wearable	• Objectives was specified and above 80% of reliability was set and included in the objectives.	Sec. 1.4 on p. 6
Engr. John Anthony Jose	• Include the direction of fall • Detections of emergency was included on the RRL, be specific if the term emergency has a role in the paper.	• Fall directions was included on the paper. • Emergency features of the paper was removed as the project title focus was changed to be a developmental project.	Sec. 1.4 on p. 6

Appendix D
REVISIONS TO THE FINAL

De La Salle University

Make a table with the following columns for showing the summary of revisions to the proposal based on the comments of the panel of examiners.

1. Panelist name
2. Comment
3. Summary of how the comment has been addressed
4. Locations in the document where the changes have been reflected

TABLE D.1 SUMMARY OF REVISIONS TO THE THESIS

Panelist name	**Comment**	**Summary of how the comment has been addressed**	**Locations**
Engr. Alexander CO Abad	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. **First** itemtext **Second** itemtext **Last** itemtext **First** itemtext **Second** itemtext	Sec. 5.2 on p. 58, Sec. 5.3 on p. 62, Fig. **??** on p. **??**

Continued on next page

De La Salle University

Continued from previous page

Panelist name	**Comment**	**Summary of how the comment has been addressed**	**Locations**
Dr. Francis Roy Navea	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. **First** itemtext **Second** itemtext **Last** itemtext **First** itemtext **Second** itemtext	Sec. 5.2 on p. 58, Sec. 5.3 on p. 62, Fig. **??** on p. **??**

Continued on next page

De La Salle University

Continued from previous page

Panelist name	**Comment**	**Summary of how the comment has been addressed**	**Locations**
Engr. Melvin Cabatuan	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. • First itemtext • Second itemtext • Last itemtext • First itemtext • Second itemtext	Sec. 5.2 on p. 58, Sec. 5.3 on p. 62, Fig. ?? on p. ??

Continued on next page

De La Salle University

Continued from previous page

Panelist name	**Comment**	**Summary of how the comment has been addressed**	**Locations**
Engr. Noriel Mallari	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. 1. First itemtext 2. Second itemtext 3. Last itemtext 4. First itemtext 5. Second itemtext	Sec. 5.2 on p. 58, Sec. 5.3 on p. 62, Fig. **??** on p. **??**

Continued on next page

De La Salle University

Continued from previous page

Panelist name	Comment	Summary of how the comment has been addressed	Locations
	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.	Sec. 5.2 on p. 58, Sec. 5.3 on p. 62, Fig. **??** on p. **??**

Appendix E
USAGE EXAMPLES

Appendix F
PROGRAM

De La Salle University

F1 Arduino Sketch

Arduino_Fall_Analyzer.ino

```
//A: IMPORT THE LIBRARIES AND DEFINITIONS
//A1: DECLARATIONS FOR CURIEIMU
#include "CurieIMU.h"

int axIn, ayIn, azIn;

double vsumcurrentIn = 0;
double vsumcheckIn = 0;
double vsumnewIn = 0;
//A1: END

//A2: DECLARATIONS FOR EXTERNAL ACCELEROMETER
#include "I2Cdev.h"
#include "MPU6050.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

MPU6050 accelgyro;

int16_t axEx, ayEx, azEx, gxEx, gyEx, gzEx;
double byEx, bxEx, bzEx;

double vsumcurrentEx = 0;
double vsumcheckEx = 0;
double vsumnewEx = 0 ;
//A2: END

//A3: DECLARATIONS FOR SDCARD MODULE
#include <SPI.h>
#include "SdFat.h"

#define USE_SDIO 0 // Set USE_SDIO to zero for SPI card access

const uint8_t SD_CHIP_SELECT = SS; // Default SD Chip select is the SPI
    SS pin

#if USE_SDIO // Use faster SdioCardEX
SdFatSdioEX sd;
#else // USE_SDIO
SdFat sd;
#endif  // USE_SDIO

float cardSize; // global for card size
File myFile; //will be used for file creation
SdFile file; //will be used for getting filenames
//A3: END

//A4: DECLARATIONS FOR GPS GSM RTC SHIELD
#include <SoftwareSerial.h>
```

```

#define DEBUG true
#define GPSready A2

SoftwareSerial mySerial(7, 8);
//A4: END

//A5: OTHER DECLARATIONS
#define DeviceReady A1
#define FallMemory A0

#define falseAlarmButton A3

unsigned long int fallStart;

double degreesdiff = 0;

String lastOrientation = "";
String lastKnownTimeLoc = "";
String userName = "";
//A5: END

//A6: DECLARATIONS FOR KEEPING TRACK OF ACTIVITIES
#define maxAct 1000 //Maximum number of activities per text file
int actFileCountR; //activity file counter
int actCounterR;   //activity counter per file
//A6: END

//A7: DECLARATIONS FOR TIMER
#include "CurieTimerOne.h"
const int gpsTimer = 5000000; //5 seconds
const int memoryTimer = 9000000; //9 seconds
//A7: END

//A: END

void parseAndsave(char *buff) {
  char *name = strtok(buff, " =");
  if (name) {
    char *junk = strtok(NULL, " ");
    if (junk) {
      char *valu = strtok(NULL, " ");
      if (valu) {
        int val = atoi(valu);
        if (strcmp(name, "actFileCount") == 0) {
          actFileCountR = val;
        }
        if (strcmp(name, "actCounter") == 0) {
          actCounterR = val;
        }
      }
    }
  }
}

void readFileLog() {
  myFile = sd.open("filelog.txt");
```

```
  if (myFile) {
    char buffer[5];
    byte index = 0;

    while (myFile.available()) {
      //Serial.write(myFile.read());
      char c = myFile.read();
      if (c == '\n' || c == '\r') { //Check for carriage return or line feed
        parseAndsave(buffer);
        index = 0;
        buffer[index] = '\0'; //Keep buffer NULL terminated
      } else {
        buffer[index++] = c;
        buffer[index] = '\0'; //Keep buffer NULL terminated
      }
    }
    myFile.close();
  } else {
    Serial.println("ERROR READING FILE LOG!");
    readFileLog();
  }

  Serial.print("actFileCount: ");
  Serial.println(actFileCountR);
  Serial.print("actCounter: ");
  Serial.println(actCounterR);
}

void logData(String userOrientation) {
  if (!sd.exists("Activities")) {
    sd.mkdir("Activities");
    sd.chdir("Activities");
  } else {
    sd.chdir("Activities");
  }

  String fileName = "activity";
  fileName.concat(actFileCountR);
  fileName.concat(".txt");

  if (actCounterR < maxAct) {
    myFile = sd.open(fileName, FILE_WRITE);

    if (myFile) {
      myFile.println(lastKnownTimeLoc);
      myFile.println(userOrientation);
      myFile.println("-end_of_activity-");
      myFile.close();
      actCounterR++;

      Serial.println(actCounterR);
      //      Serial.println(sendData("AT+CGNSINF", 1000, DEBUG));
      Serial.println(lastKnownTimeLoc);
      Serial.println("User Orientation: " + userOrientation);
      Serial.println("-end_of_activity-");
    } else {
```

De La Salle University

```
      Serial.println("Error Writing Activity File");
    }
  } else {
    actFileCountR++;
    actCounterR = 0;
  }

  if (!sd.chdir()) {
    Serial.println("Error Going back to the root folder");
  }

  updateFileLog();

}

void updateFileLog() {
  String a = "actFileCount = ";
  String b = "actCounter = ";

  a.concat(actFileCountR);
  b.concat(actCounterR);

  removeFile("filelog.txt");
  myFile = sd.open("filelog.txt", FILE_WRITE);
  if (myFile) {
    myFile.println(a);
    myFile.println(b);
    Serial.println(a);
    Serial.println(b);

    myFile.close();
  } else {
    Serial.println("ERROR UPDATING FILE LOG!");
  }
}

void removeFile(String toRemove) {
  myFile = sd.open(toRemove, FILE_WRITE);
  if (myFile) {
    if (!myFile.remove()) {
      Serial.println("ERROR REMOVING OLD FILE LOG!");
    }
    myFile.close();
  } else {
    Serial.println("ERROR OPENING FILE TO BE REMOVED!");
  }
}

void deleteOldFiles() {
  Serial.println("Deleting old files");
  sd.chdir("Activities");
  //sd.ls(LS_R);

  int counter = 0;
  int itemsToDelete = 5;

  char name[20];
```

De La Salle University

```
  while (file.openNext(sd.vwd(), O_READ) && counter < itemsToDelete) {
    //    file.printName();
    file.getName(name, 20); //20bytes

    removeFile(String(name));

    file.close();
    counter++;
  }

  if (!sd.chdir()) {
    Serial.println("ERROR RETURNING BACK TO THE ROOT");
  }
  delay(2000);
}

void initMainBoard() {
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();
  delay(1000);

  // Set the accelerometer range to 2G
  CurieIMU.setAccelerometerRange(2);
  Serial.println("IMU initialization successful!");
}

void initExtAccel() {
  Serial.println("Initializing External Acceleromter...");

  // Join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  accelgyro.initialize();
  delay(1000);

  //Verify Connection with the external accelerometer
  Serial.println("Testing external accelerometer connection...");
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection
      successful" : "MPU6050 connection failed");
  accelgyro.setSleepEnabled(false);

  if (accelgyro.testConnection() == 0) {
    Serial.println("Reinitialization started!");
    delay(500);
    Serial.write(12);
    setup();
  }

  Serial.println("External Accelerometer initialization successful!");
}

void initSDCard() {
  Serial.println("Initializing Sd Card...");
```

```
#if USE_SDIO
  if (!sd.cardBegin()) {
    Serial.println("cardBegin failed");
    Serial.println("Re-initializing Sd Card...");
    initSDCard();
  }
  if (!sd.begin(SD_CHIP_SELECT)) {
    Serial.println("SD Chip Select initialization failed");
    Serial.println("Re-initializing Sd Card...");
    initSDCard();
  }
#else  // USE_SDIO
  // Initialize at the highest speed supported by the board that is
  // not over 50 MHz. Try a lower speed if SPI errors occur.
  if (!sd.cardBegin(SD_CHIP_SELECT, SD_SCK_MHZ(50))) {
    Serial.println("cardBegin failed");
    Serial.println("Re-initializing Sd Card...");
    initSDCard();
  }
  if (!sd.begin(SD_CHIP_SELECT, SD_SCK_MHZ(50))) {
    Serial.println("SD Chip Select initialization failed");
    Serial.println("Re-initializing Sd Card...");
    initSDCard();
  }
#endif  // USE_SDIO

  cardSize = sd.card()->cardSize();

  if (cardSize == 0) {
    Serial.println("cardSize failed");
    Serial.println("Re-initializing Sd Card...");
    initSDCard();
  }

  if (!sd.fsBegin()) {
    Serial.println("\nFile System initialization failed.\n");
    Serial.println("Re-initializing Sd Card...");
    initSDCard();
  }

  Serial.println("Sd Card initialization successful!");
}

void checkGPSConnection() {
  String response = sendData("AT+CGNSINF", 1000, DEBUG);

  //Check if GPS is already connected/fixed
  if (response[25] == ’1’) {
    digitalWrite(GPSready, HIGH);
    lastKnownTimeLoc = response;
    Serial.println("GPS READY!");
  } else {
    digitalWrite(GPSready, !digitalRead(GPSready));
    Serial.println("GPS Connecting...");
  }

  Serial.println(response);
```

```
}

void fallBuzz() {
  tone(FallMemory, 400, 200);
  delay(200);
  noTone(FallMemory);
  tone(FallMemory, 500, 200);
  delay(200);
  noTone(FallMemory);
  tone(FallMemory, 600, 200);
  delay(200);
  noTone(FallMemory);
}

void memoryBuzz() {
  tone(FallMemory, 300, 300);
  delay(300);
  noTone(FallMemory);
}

void initGPSModule() {
  Serial.println("Initializing GPS,GSM,RTC Shield...");
  mySerial.begin(38400);

  pinMode(GPSready, OUTPUT);

  onGPS();

  while (lastKnownTimeLoc == "") {
    checkGPSConnection();
  }

  delay(10000);
}

void onGPS() {
  sendData("AT+CGNSPWR=1", 1000, DEBUG);
  Serial.println("GPS Turned ON!");
}

void offGPS() {
  sendData("AT+CGNSPWR=0", 1000, DEBUG);
  Serial.println("GPS Turned OFF!");
}

String sendData(String command, const int timeout, boolean debug) {
  String response = "";
  mySerial.println(command);

  delay(5);

  if (debug) {
    long int time = millis();
    while ( (time + timeout) > millis()) {
      while (mySerial.available()) {
        response += char(mySerial.read());
      }
```

```
    }
  }

  return response;
}

void checkSpace() {
  float totalSize = 0.000512 * cardSize;
  float freeSize = 0.000512 * sd.vol()->freeClusterCount() * sd.vol()->
      blocksPerCluster();

  float lowLevel = 0.1 * totalSize; //10% of total size
  if (freeSize <= lowLevel) {
    memoryBuzz();
    deleteOldFiles();
    Serial.println("LOW MEMORY SPACE!");
    Serial.print("Remaining Space: ");
    Serial.print(freeSize);
    Serial.println(" MB (MB = 1,000,000 bytes)");
  } else {
    Serial.print("Remaining Space: ");
    Serial.println(freeSize);
  }
}

String setMessage() {
  String message = userName;
  message.concat(" fell! at:  ");
  String date = "";
  String longitude = "";
  String latitude = "";
  int comma = 0;
  int i = 0;

  while (comma <= 5) {
    if (lastKnownTimeLoc[i] == ',' ) {
      comma++;
      i++;
    }

    if (comma == 2) {
      date = date + lastKnownTimeLoc[i];
    }
    if (comma == 3) {
      longitude = longitude + lastKnownTimeLoc[i];
    }
    if (comma == 4) {
      latitude = latitude + lastKnownTimeLoc[i];
    }
    i++;

  }

  String newTime = "";
  String newDate = "";

  for (int j = 0; j < 14; j++) {
```

De La Salle University

```
    if (j < 8) {
      newDate = newDate + date[j];
      if (j == 3 || j == 5 || j == 5) {
        newDate.concat("-");
      }
    }
    if (j > 7) {
      newTime = newTime + date[j];
    }
  }

  String hh = "";
  for (int j = 0; j < 2; j++){
    hh = hh + newTime[j];
  }

  int timeUTC8 = hh.toInt();
  if(timeUTC8 > 12){
    timeUTC8 = timeUTC8 - 12;
  }

  timeUTC8 = hh.toInt() + 8;
  if(timeUTC8 > 24){
    timeUTC8 = timeUTC8 - 24;
  }

  String UTC8 = String(timeUTC8);
  for(int j = 0; j < 6; j++){
    if(j == 1 || j == 4){
      UTC8.concat(":");
    }
    if(j > 1){
      UTC8 = UTC8 + newTime[j];
    }
  }

  message.concat("\n");
  message.concat("LONGITUDE and LATITUDE: ");
  message.concat("\n");
  message.concat(longitude);
  message.concat(", ");
  message.concat(latitude);
  message.concat("\n");
  message.concat("DATE: ");
  message.concat(newDate);
  message.concat("\n");
  message.concat("TIME: ");
  message.concat(UTC8);
  message.concat(" \r");

  Serial.println(message);

  return message;
}

void initUsername() {
  myFile = sd.open("profile.txt");
```

De La Salle University

```
  if (myFile) {
    char buffer[15];
    byte index = 0;
    int lineCount = 0;
    while (myFile.available() && lineCount < 3) {
      char c = myFile.read();
      if (c == '\n' || c == '\r') { //Check for carriage return or line
         feed
        userName.concat(buffer);
        lineCount++;
        index = 0;
        buffer[index] = '\0'; //Keep buffer NULL terminated
      } else {
        buffer[index++] = c;
        buffer[index] = '\0'; //Keep buffer NULL terminated
      }
    }

    myFile.close();
  } else {
    Serial.println("ERROR READING PROFILE!");
  }
}

void SendTextMessage() {
  offGPS(); //turn off GPS to prevent interruption

  Serial.println("=====================");
  Serial.println("===SENDING MESSAGE!===");

  String message = setMessage();

  myFile = sd.open("respondents.txt");
  if (myFile) {
    char buffer[15];
    byte index = 0;

    while (myFile.available()) {
      char c = myFile.read();
      boolean sent = false;
      if (c == '\n' || c == '\r') { //Check for carriage return or line
         feed
        if (buffer[0] == '+') {
          String toContact(buffer);

          Serial.print("CONTACT NUMBER: ");
          Serial.println(toContact);

          while (sent == false) {
            String response = "";
            String receiver = "AT+CMGS=\"";
            receiver.concat(toContact);
            receiver.concat("\"\r");
            Serial.println(receiver);

            mySerial.print("\r");
            mySerial.print("AT+CMGF=1\r");    //Because we want to send
```

De La Salle University

```
              the SMS in text mode
            mySerial.print(receiver);
            delay(1000);
            mySerial.print(message);
            mySerial.write(0x1A);

            long int time = millis();
            while ((time + 1000) > millis()) {
              while (mySerial.available()) {
                response += char(mySerial.read());
              }
            }

            Serial.println("RESPONSE: ");
            Serial.println(response);

            int bracketCount = 0;
            for (int i = 0; i < response.length(); i++) {
              if (response[i] == '>') {
                bracketCount++;
              }
            }
            if (bracketCount >= 2) {
              sent = true;
              Serial.println("===MESSAGE SENT!===");
            } else {
              Serial.println("===MESSAGE WAS NOT SENT!===");
              Serial.println("Resending Message...");
              sent = false;
            }
          }

        }
        index = 0;
        buffer[index] = '\0'; //Keep buffer NULL terminated
      } else {
        buffer[index++] = c;
        buffer[index] = '\0'; //Keep buffer NULL terminated
      }
    }
    myFile.close();
  } else {
    Serial.println("ERROR READING RESPONDENTS FILE!");
  }

  Serial.println("===DONE!===");
  Serial.println("=====================");
  delay(5000);
  onGPS(); //turon on GPS again
}

String getOrientation() {
  byEx = ayEx;

  accelgyro.getMotion6(&axEx, &ayEx, &azEx, &gxEx, &gyEx, &gzEx); //
      mpu6050
  CurieIMU.readAccelerometer(axIn, ayIn, azIn); // curie
```

```

  vsumcurrentEx = sqrt(pow(axEx, 2) + pow(ayEx, 2) + pow(azEx, 2));
  vsumcheckEx = abs((abs(vsumnewEx) - abs(vsumcurrentEx)) / (abs(vsumnewEx))) * 100;

  vsumcurrentIn = sqrt(pow(axIn, 2) + pow(ayIn, 2) + pow(azIn, 2));
  vsumcheckIn = abs((abs(vsumnewIn) - abs(vsumcurrentIn)) / (abs(vsumnewIn))) * 100;

  if (abs(vsumcheckEx) >= 1.5 && abs(vsumcheckIn) >= 1.5) {
    vsumnewEx = vsumcurrentEx;
    vsumnewIn = vsumcurrentIn;
    return dynamicmode();
  } else {
    vsumnewEx = (vsumnewEx + vsumcurrentEx) / 2; //average vector sum
    vsumnewIn = (vsumnewIn + vsumcurrentIn) / 2; //average vector sum
    return staticmode();
  }
}

void setup() {
  Serial.begin(38400);
  delay(1000);
//  while (!Serial) {} //wait for serial port to connect

  //Initialize the devices
  initMainBoard();
  initExtAccel();
  initSDCard();
  readFileLog();
  initUsername();
  initGPSModule();

  pinMode(FallMemory, OUTPUT);
  pinMode(DeviceReady, OUTPUT);
  pinMode(falseAlarmButton, INPUT_PULLUP);

  CurieTimerOne.start(memoryTimer, &checkSpace);

  //Prompt a welcome message
  Serial.println("Device is ready!");
  digitalWrite(DeviceReady, HIGH);
  Serial.println("===============================");
}

boolean falling = false;

void loop() {
  checkGPSConnection();

  falling = false;
  String currentOrientation = getOrientation();

  if (currentOrientation != "UNKNOWN") {
    if (currentOrientation != lastOrientation) {
      logData(currentOrientation);

```

```
      Serial.println("");
      Serial.print("=====");
      Serial.print(currentOrientation);
      Serial.println("=====");
      Serial.println("");

      lastOrientation = currentOrientation;
    }
  }

  //if falling
//  falling = true;          //just for testing
//  fallStart = millis();    //just for testing
  if (falling == true) {
    //check wether to send an alarm in 10 seconds
    boolean flag = true;
    while (flag == true) {
      if (millis() - fallStart <= 10000) {
        if (digitalRead(falseAlarmButton) == LOW) {
          Serial.println("False Alarm!");
          flag = false;
        } else {
          //Serial.println(millis() - fallStart);
          fallBuzz();
        }
      } else {
        SendTextMessage();
        //setMessage();
        flag = false;
      }
    }
  }
}

static String staticmode() {

  String staticstr = "UNKNOWN";

  if (axEx <= -13900 && axIn <= -16000 && axIn >= -17500) {
    staticstr = "Standing Position";
  } else if (ayEx >= 14000 && axIn <= -12500) {
    staticstr = "Sitting Position";
  } else if ((abs(azIn) >=  13500 || abs(ayIn) >=  13500) && (abs(azEx)
     >=  13500 || abs(ayEx) >=  13500)) {
    staticstr = "Lying Position";
  }

  return staticstr;
}

static String dynamicmode() {

  String dynastr = "UNKNOWN";
  falling = false;

  degreesdiff = abs(((180 / 3.14) * (acos(ayEx / vsumnewEx))) - ((180 /
     3.14) * (acos(byEx / vsumnewEx))));
```

```

  //  Serial.print((abs((vsumnew-abs(ayEx))/vsumnew))*9.8);
  //  Serial.println(" m/s^2");
  //

  int fallTreshold = 10;

  if (abs(vsumcheckEx) >= fallTreshold || abs(vsumcheckIn) >=
      fallTreshold ) {
    if (ayIn >= 13500 && ayEx >= 15900 && (abs(vsumcheckEx) >=
        fallTreshold || abs(vsumcheckIn) >= fallTreshold )) {
      fallStart = millis();
      dynastr = "Falling! : Backwards";
      falling = true;
    }
    else if (ayIn <= -14000 && ayEx <= -15400 && (abs(vsumcheckEx) >=
        fallTreshold || abs(vsumcheckIn) >= fallTreshold )) {
      fallStart = millis();
      dynastr = "Falling! : Forward";
      falling = true;
    }
    else if (abs(azIn) >=  15900 && abs(azEx) >= 15400 && (abs(
        vsumcheckEx) >= fallTreshold || abs(vsumcheckIn) >= fallTreshold
         )) {
      fallStart = millis();
      dynastr = "Falling! : Sideways";
      falling = true;
    }

    //  Serial.print((abs(vsumcheck/100))*9.8);
    //  Serial.println(" m/s^2");
  }
  if ((axIn <= -15000 && axIn >= -17500) && degreesdiff <= 55 &&
      degreesdiff >= 10) {
    dynastr = "Walking";
  }
  return dynastr;
}
```

De La Salle University

F2 Code used in GUI

style.css

```
.root{
    -fx-background-color: #ecf0f1;
}

#setupPane{
    -fx-background-image: url("Main.png");
    -fx-background-size: 1030;
    -fx-background-repeat: stretch;
    -fx-background-position: center center;
}

#explorePane{
    -fx-background-image: url("Explore.png");
    -fx-background-size: 1030;
    -fx-background-repeat: stretch;
    -fx-background-position: center center;
}

#header1Label{
    -fx-font-family: Asimov;
    -fx-text-fill: orange;
    -fx-font-size: 30;
    -fx-font-position: center center;
}

#header2Label{
    -fx-font-family: Asimov;
    -fx-text-fill: #2c3e50;
    -fx-font-size: 15;
    -fx-font-position: center center;
    -fx-text-alignment: justify;
}

#header3Label{
    -fx-font-family: Asimov;
    -fx-text-fill: orange;
    -fx-font-size: 20;
    -fx-font-position: center center;
}

#fieldsLabelLabel{
    -fx-font-family: Asimov;
    -fx-text-fill: #2c3e50;
    -fx-font-size: 13;
    -fx-font-position: center center;
    -fx-text-alignment: justify;
}

#feedbackLabel{
    -fx-font-family: Asimov;
    -fx-text-fill: #2c3e50;
```

```
    -fx-font-size: 10;
    -fx-font-position: center center;
}

#errorLabel{
    -fx-font-family: Asimov;
    -fx-text-fill: red;
    -fx-font-size: 10;
    -fx-font-position: center center;
}

#start{
        -fx-font-family: Asimov;
        -fx-text-fill: orange;
        -fx-font-size: 20;
        -fx-font-position: center center;
        -fx-background-color: white;
}

#respondButtons{
        -fx-font-family: Asimov;
        -fx-text-fill: orange;
        -fx-font-size: 15;
        -fx-font-position: center center;
        -fx-background-color: white;
}

#selected{
        -fx-font-family: Asimov;
        -fx-text-fill: white;
        -fx-font-size: 20;
        -fx-font-position: center center;
        -fx-background-color: #009fe0;
        -fx-background-radius: 5em;
        -fx-max-height: 45px;
        -fx-max-width: 45px;
        -fx-min-height: 45px;
        -fx-min-width: 45px;
}

#idle{
        -fx-font-family: Asimov;
        -fx-text-fill: orange;
        -fx-font-size: 20;
        -fx-font-position: center center;
        -fx-background-color: white;
        -fx-background-radius: 5em;
        -fx-max-height: 45px;
        -fx-max-width: 45px;
        -fx-min-height: 45px;
        -fx-min-width: 45px;
}
```

Main.java

```
package dlsu;
```

```

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("SelectCard
            .fxml"));
        root.getStylesheets().addAll(this.getClass().getResource("style.
            css").toExternalForm());
        primaryStage.setTitle("Fall Analyzer | Select Card");
        primaryStage.setResizable(false);
        primaryStage.setScene(new Scene(root, 1024, 575.5));
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

changeScene.java

```
package dlsu;

import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import javafx.event.ActionEvent;

import java.io.IOException;

/**
 * Created by aenon on 11/11/2017.
 */

public class changeScene {

    public void setScene(String fxmlFile, String cssFile, ActionEvent
        actionEvent, String title) throws IOException {
        Parent parent = FXMLLoader.load(getClass().getResource(fxmlFile)
            );
        parent.getStylesheets().addAll(this.getClass().getResource(
            cssFile).toExternalForm());
        Scene scene = new Scene(parent);
        Stage stage = (Stage) ((Node) actionEvent.getSource()).getScene
            ().getWindow();
        stage.setScene(scene);
```

```
        stage.setTitle(title);
        stage.show();
    }
}
```

exploreController.java

```
package dlsu;

import com.sun.org.apache.xpath.internal.SourceTree;
import dlsu.Utils.checkContactNumber;
import dlsu.Utils.logInDialog;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.Initializable;
import javafx.scene.Group;
import javafx.scene.chart.PieChart;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import javax.sound.midi.Soundbank;
import java.io.*;
import java.net.URL;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class exploreController implements Initializable{

    public Label usernameHello;
    public TextField usernameLogin;
    public PasswordField passwordLogin;
    public PasswordField confirmLogin;
    public Label feedbackLabelLogin;
    public Button editLogin;
    public Button saveLogin;

    public TextField firstNameProfile;
    public TextField LastNameProfile;
    public ChoiceBox genderProfile;
    public TextField ageProfile;
    public TextField addressProfile;
    public TextField contactNumberProfile;
    public Label feedbackLabel;
    public Button editProfile;
    public Button saveProfile;

    public Group editResponders;
    public TableView respondersTable;
```

```
    public TableColumn firstName;
    public TableColumn lastName;
    public TableColumn contactNumber;

    public TextField firstNameResponders;
    public TextField LastNameResponders;
    public TextField contactNumberResponders;
    public Button saveResponder;
    public Label allFieldsError;
    public Label successful;
    public Button deleteResponder;
    public Button editResponder;
    public Button addResponder;

    public TableView activityTable;
    public TableColumn dateCell;
    public TableColumn timeCell;
    public TableColumn activityCell;
    public TableColumn locationCell;
    public Label feedbackData;
    public Button deleteData;
    public Button deleteAllData;
    public Button exportData;
    public Label numberOfFalls;
    public PieChart storageSpace;

    public int fallTimes = 0;
    public Tab profileTab;
    public Tab respondersTab;
    public Tab activityTab;

    dlsu.Utils.logInDialog logInDialog = new logInDialog();

    public final ObservableList<responders> data = FXCollections.
        observableArrayList();
    public final ObservableList<activities> log = FXCollections.
        observableArrayList();

    public static class activities{
        private SimpleStringProperty date;
        private SimpleStringProperty time;
        private SimpleStringProperty activity;
        private SimpleStringProperty location;

        private activities(String dateLog, String timeLog, String
            activityLog, String locationLog){
            this.date = new SimpleStringProperty(dateLog);
            this.time = new SimpleStringProperty(timeLog);
            this.activity = new SimpleStringProperty(activityLog);
            this.location = new SimpleStringProperty(locationLog);
        }

        public String getDate() {
            return date.get();
        }

        public void setDate(String dateLog) {
```

De La Salle University

```
            date.set(dateLog);
        }

        public String getTime() {
            return time.get();
        }

        public void setTime(String timeLog) {
            time.set(timeLog);
        }

        public String getActivity() {
            return activity.get();
        }

        public void setActivity(String activityLog) {
            activity.set(activityLog);
        }

        public String getLocation() {
            return location.get();
        }

        public void setLocation(String locationLog) {
            location.set(locationLog);
        }
    }

    public static class responders{
        private SimpleStringProperty firstName;
        private SimpleStringProperty lastName;
        private SimpleStringProperty contactNumber;

        private responders(String first, String last, String contact){
            this.firstName = new SimpleStringProperty(first);
            this.lastName = new SimpleStringProperty(last);
            this.contactNumber = new SimpleStringProperty(contact);
        }

        public String getFirstName(){
            return firstName.get();
        }

        public void setFirstName(String first){
            firstName.set(first);
        }

        public String getLastName(){
            return lastName.get();
        }

        public void setLastName(String last){
            lastName.set(last);
        }

        public String getContactNumber(){
            return contactNumber.get();
```

```
        }

        public void setContactNumber(String contact){
            contactNumber.set(contact);
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        initProfileTab();
        initRepondersTab();
        initActivityTab();
    }

    private void initActivityTab() {
        File dir = new File(selectCardController.driveLetter + "
            Activities");
        if (dir.isDirectory()){
            for(File file : dir.listFiles()){
                if (file.isFile()){
                    BufferedReader br = null;
                    try {
                        br = new BufferedReader(new FileReader(
                            selectCardController.driveLetter + "
                            Activities\\" + file.getName()));
                        String line = "";
                        String date = "";
                        String time = "";
                        String activity = "";
                        String location = "";
                        String toExtract = "";

                        while((line = br.readLine()) != null){
                            if (line.equals("-end_of_activity-")){
                                //do nothing
                            } else {
                                //extract rtc and coordinates here.
                                String rtc = "";
                                String latitude = "";
                                String longitude = "";

                                br.readLine();
                                toExtract = br.readLine();

                                int comma = 0;
                                int g = 0;

                                while(comma < 5){
                                    if (toExtract.charAt(g) == ','){
                                        comma++;
                                        g++;
                                    }
                                    if (comma == 2){
                                        rtc = rtc + toExtract.charAt(g);
                                    }
                                    if (comma == 3){
                                        longitude = longitude +
```

```
                                toExtract.charAt(g);
                        }
                        if (comma == 4){
                            latitude = latitude + toExtract.
                                charAt(g);
                        }

                        g++;
                    }

                    String year = "";
                    String month = "";
                    String day = "";

                    String hour = "";
                    String min = "";
                    String sec = "";

                    //extract date and time here.
                    for (int i = 0; i < 14; i++){
                        if (i < 8){
                            if (i < 4){
                                year = year + rtc.charAt(i);
                            }
                            if (i > 3 && i < 6){
                                month = month + rtc.charAt(i
                                    );
                            }
                            if (i > 5 && i < 8){
                                day = day + rtc.charAt(i);
                            }
                        } else{
                            if (i == 8 || i == 9){
                                hour = hour + rtc.charAt(i);
                            } else if (i == 10 || i == 11){
                                min = min + rtc.charAt(i);
                            } else if (i == 12 || i == 13){
                                sec = sec + rtc.charAt(i);
                            }
                        }
                    }

                    //convert numbered months to words:
                    date = getDate(year, month, day);

                    //convert 24hour format to 12hour format
                        (AM, PM, NN, MN)here:
                    time = getTime(hour, min, sec);

                    //combine coordinates here:
                    location = longitude + ", " + latitude;

                    br.readLine();
                    br.readLine();
                    br.readLine();
                    activity = br.readLine();

```

```java
                                if(activity.contains("Falling!")){
                                    fallTimes++;
                                    numberOfFalls.setText(String.valueOf(fallTimes));
                                }
                                log.add(new activities(date, time, activity, location));
                                dateCell.setCellValueFactory(new PropertyValueFactory<activities, String>("date"));
                                timeCell.setCellValueFactory(new PropertyValueFactory<activities, String>("time"));
                                activityCell.setCellValueFactory(new PropertyValueFactory<activities, String>("activity"));
                                locationCell.setCellValueFactory(new PropertyValueFactory<activities, String>("location"));
                                activityTable.getItems().setAll(this.log);
                            }
                        }

                    } catch (IOException e) {
                        System.out.println(e);
                    } finally {
                        try {
                            br.close();
                        } catch (IOException e) {
                            System.out.println(e);
                        }
                    }
                } else {
                    System.out.println("NOT A FILE!");
                }
            }
        } else {
            System.out.println("NOT A DIRECTORY!");
        }

        setStorageChart();
    }

    private void initRepondersTab(){
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader(selectCardController.driveLetter + "respondents.txt"));

            Boolean end = false;
            String line = "";
            String first = "";
            String last = "";
            String contact = "";
            while (end == false) {
                line = br.readLine();
```

```
                if (line.equals("-end-")) {
                    end = true;
                } else if (line.equals("breakLine")) {
                    //just do nothing
                } else {
                    first = line;
                    last = br.readLine();
                    contact = br.readLine();
                    data.add(new responders(first, last, contact));
                    firstName.setCellValueFactory(new PropertyValueFactory<responders, String>("firstName"));
                    lastName.setCellValueFactory(new PropertyValueFactory<responders, String>("lastName"));
                    contactNumber.setCellValueFactory(new PropertyValueFactory<responders, String>("contactNumber"));
                    respondersTable.getItems().setAll(this.data);
                }
            }
        } catch (IOException e) {
        } finally {
            try {
                br.close();
            } catch (IOException e) {
            }
        }
    }

    private void initProfileTab() {
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader(selectCardController.driveLetter + "login.txt"));
            String temp = br.readLine();
            usernameHello.setText(temp + "!");
            usernameLogin.setText(temp);
            temp = br.readLine();
            passwordLogin.setText(temp);
            confirmLogin.setText(temp);
        } catch (IOException e) {
        } finally {
            try {
                br.close();
            } catch (IOException e) {
            }
        }

        genderProfile.getItems().add("male");
        genderProfile.getItems().add("female");
        try {
            br = new BufferedReader(new FileReader(selectCardController.driveLetter + "profile.txt"));
            firstNameProfile.setText(br.readLine());
            LastNameProfile.setText(br.readLine());
            genderProfile.setValue(br.readLine());
```

```
            ageProfile.setText(br.readLine());
            addressProfile.setText(br.readLine());
            contactNumberProfile.setText(br.readLine());
        } catch (IOException e) {
        } finally {
            try {
                br.close();
            } catch (IOException e) {
            }
        }

    }

    private void setStorageChart() {
        File space = new File(selectCardController.driveLetter);
        long freespace = space.getFreeSpace() / (1024*1024*1024); //
            outputs in GB. (1024*1024) if MB
        long totalspace = space.getTotalSpace() / (1024*1024*1024); //
            outputs  in*GB (1024*1024) if MB
        long usedspace = (space.getTotalSpace() - space.getFreeSpace())
            / (1024*1024*1024); //outputs  in*GB (1024*1024) if MB

        ObservableList<PieChart.Data> pieDate =
                FXCollections.observableArrayList(
                        new PieChart.Data("Free Space: " + freespace + "
                             GB", freespace),
                        new PieChart.Data("Used Space: " + usedspace + "
                             GB", usedspace)
                );
        storageSpace.setData(pieDate);
        storageSpace.setLabelLineLength(10);
        storageSpace.setLabelsVisible(false);

        storageSpace.getData().stream().forEach(data -> {
            Tooltip tooltip = new Tooltip();
            tooltip.setText(String.format("%.1f%%", 100*data.getPieValue
                ()/totalspace));
            Tooltip.install(data.getNode(), tooltip);
            data.pieValueProperty().addListener((observable, oldValue,
                newValue) ->
                    tooltip.setText(newValue + "%"));
        });
    }

    private String getDate(String year, String month, String day){
        switch (month){
            case "01":
                month = "January";
                break;
            case "02":
                month = "February";
                break;
            case "03":
                month = "March";
                break;
            case "04":
                month = "April";
```

De La Salle University

```
                break;
            case "05":
                month = "May";
                break;
            case "06":
                month = "June";
                break;
            case "07":
                month = "July";
                break;
            case "08":
                month = "August";
                break;
            case "09":
                month = "September";
                break;
            case "10":
                month = "October";
                break;
            case "11":
                month = "November";
                break;
            case "12":
                month = "December";
                break;
        }

        String date = month + " " + day + ", " + year;
        return date;
    }

    private String getTime(String hour, String min, String sec){
        String time = "";
        int hh = Integer.parseInt(hour);

        System.out.println("hh raw: " + hh);

        if (hh > 12){
            hh = hh - 12;
        }

        System.out.println("hh 12hr: " + hh);

        hh = hh + 8; //convert to UTC+8:00

        System.out.println("hh utc: " + hh);

        if (hh > 12){
            hh = hh - 12;

            if (hh == 12){
                time = time + hh + ":" + min + ":" + sec + " M.N.";
            } else{
                time = time + hh + ":" + min + ":" + sec + " P.M.";
            }

        } else if (hh < 13){
```

De La Salle University

```
            if (hh == 12) {
                time = time + hh + ":" + min + ":" + sec + " N.N.";
            } else if (hh == 0) {
                time = time + "12" + ":" + min + ":" + sec + " A.M.";
            } else {
                time = time + hh + ":" + min + ":" + sec + " A.M.";
            }
        }

        System.out.println("TIME: " + time);

        return time;
    }

    public void onEditLogin(ActionEvent actionEvent) {
        if(logInDialog.logIn()){
            usernameLogin.setEditable(true);
            passwordLogin.setEditable(true);
            confirmLogin.setEditable(true);
            saveLogin.setDisable(false);
            feedbackLabelLogin.setText("");
        }
    }

    public void onSaveLogin(ActionEvent actionEvent) {
        if(usernameLogin.getText().isEmpty() || passwordLogin.getText().
            isEmpty() || confirmLogin.getText().isEmpty()){
            feedbackLabelLogin.setText("All fields are required!");
        } else if (!passwordLogin.getText().equals(confirmLogin.getText
            ())){
            feedbackLabelLogin.setText("Passwords do not match!");
        } else {
            usernameHello.setText(usernameLogin.getText() + "!");
            BufferedWriter writer = null;
            try {
                writer = new BufferedWriter(new OutputStreamWriter(new
                    FileOutputStream(selectCardController.driveLetter +
                    "login.txt"), "utf-8"));
                writer.write(usernameLogin.getText());
                writer.newLine();
                writer.write(passwordLogin.getText());
                writer.close();

                feedbackLabelLogin.setText("Login Credentials was
                    successfully modified!");
                usernameLogin.setEditable(false);
                passwordLogin.setEditable(false);
                confirmLogin.setEditable(false);
                saveLogin.setDisable(true);
                setStorageChart();
            } catch (Exception e) {
            }
        }

    }

    public void onEditProfile(ActionEvent actionEvent) {
```

```
        if(logInDialog.logIn()){

            firstNameProfile.setEditable(true);
            LastNameProfile.setEditable(true);
            genderProfile.setDisable(false);
            ageProfile.setEditable(true);
            addressProfile.setEditable(true);
            contactNumberProfile.setEditable(true);
            saveProfile.setDisable(false);
            feedbackLabel.setText("");
        }
    }

    public void onSaveProfile(ActionEvent actionEvent) {
        if(firstNameProfile.getText().isEmpty() || LastNameProfile.getText().isEmpty() || ageProfile.getText().isEmpty() || addressProfile.getText().isEmpty() || contactNumberProfile.getText().isEmpty()){
            feedbackLabel.setText("All fields are required!");
        } else {
            if (checkContactNumber.validContactNumber(contactNumberProfile.getText())) {
                try {
                    Integer.parseInt(ageProfile.getText());

                    BufferedWriter writer = null;
                    try {
                        writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(selectCardController.driveLetter + "profile.txt"), "utf-8"));
                        writer.write(firstNameProfile.getText());
                        writer.newLine();
                        writer.write(LastNameProfile.getText());
                        writer.newLine();
                        writer.write(genderProfile.getValue().toString());
                        writer.newLine();
                        writer.write(ageProfile.getText());
                        writer.newLine();
                        writer.write(addressProfile.getText());
                        writer.newLine();
                        writer.write(contactNumberProfile.getText());
                        writer.close();

                        feedbackLabel.setText("Profile was successfully modified!");
                        firstNameProfile.setEditable(false);
                        LastNameProfile.setEditable(false);
                        genderProfile.setDisable(true);
                        ageProfile.setEditable(false);
                        addressProfile.setEditable(false);
                        contactNumberProfile.setEditable(false);
                        saveProfile.setDisable(true);
                        setStorageChart();
                    } catch (Exception e) {
                    }
```

De La Salle University

```
                } catch (Exception e) {
                    feedbackLabel.setText("Your age is invalid!");
                }
            } else{
                feedbackLabel.setText("Your contact number is invalid!")
                    ;
            }
        }
    }

    public void saveResponder(){
        BufferedWriter writer = null;
        try{
            writer = new BufferedWriter(new OutputStreamWriter(new
                FileOutputStream(selectCardController.driveLetter + "
                respondents.txt"), "utf-8"));

            int size = data.size();
            for (int i = 0; i < size; i++){
                writer.write(data.get(i).getFirstName());
                writer.newLine();
                writer.write(data.get(i).getLastName());
                writer.newLine();
                writer.write(data.get(i).getContactNumber());
                writer.newLine();
                writer.write("breakLine");
                writer.newLine();
            }
            writer.write("-end-");
            writer.close();
            setStorageChart();
        } catch(Exception e){
        }
    }

    public void onSaveResponder(ActionEvent actionEvent) {
        if (firstNameResponders.getText().isEmpty() ||
            LastNameResponders.getText().isEmpty() ||
            contactNumberResponders.getText().isEmpty()){
            allFieldsError.setText("All fields are required!");
            successful.setText("");
        } else {
            if (checkContactNumber.validContactNumber(
                contactNumberResponders.getText())){
                allFieldsError.setText("");

                data.add(new responders(firstNameResponders.getText(),
                    LastNameResponders.getText(),
                    contactNumberResponders.getText()));
                firstName.setCellValueFactory(new PropertyValueFactory<
                    responders, String>("firstName"));
                lastName.setCellValueFactory(new PropertyValueFactory<
                    responders, String>("lastName"));
                contactNumber.setCellValueFactory(new
                    PropertyValueFactory<responders, String>("
                    contactNumber"));
                respondersTable.getItems().setAll(this.data);
```

```java

                    firstNameResponders.setText("");
                    LastNameResponders.setText("");
                    contactNumberResponders.setText("");

                    successful.setText("Responders were successfully modified!");
                    editResponders.setDisable(true);
                    saveResponder();
                } else {
                    allFieldsError.setText("Contact number is invalid!");
                    successful.setText("");
                }

            }
        }

        public void onDeleteResponder(ActionEvent actionEvent) {
            int selectedIndex = respondersTable.getSelectionModel().getSelectedIndex();
            if (selectedIndex >= 0){
                if (logInDialog.logIn()){
                    respondersTable.getItems().remove(selectedIndex);
                    data.remove(selectedIndex);
                    allFieldsError.setText("");
                    successful.setText("Responders were successfully modified!");
                    saveResponder();
                }
            }
        }

        public void onEditResponder(ActionEvent actionEvent) {
            if(logInDialog.logIn()){
                int selectedIndex = respondersTable.getSelectionModel().getSelectedIndex();
                if (selectedIndex >= 0){
                    respondersTable.getItems().remove(selectedIndex);
                    firstNameResponders.setText(data.get(selectedIndex).getFirstName());
                    LastNameResponders.setText(data.get(selectedIndex).getLastName());
                    contactNumberResponders.setText(data.get(selectedIndex).getContactNumber());
                    data.remove(selectedIndex);
                }

                allFieldsError.setText("");
                successful.setText("");
                editResponders.setDisable(false);
            }
        }

        public void onAddResponder(ActionEvent actionEvent) {
            if (logInDialog.logIn()){
                contactNumberResponders.setText("+639");
                allFieldsError.setText("");
```

```
            successful.setText("");
            editResponders.setDisable(false);
        }
    }

    public void saveLog(){
        BufferedWriter writer = null;
        try{
            writer = new BufferedWriter(new OutputStreamWriter(new
                FileOutputStream(selectCardController.driveLetter + "
                activity.txt"), "utf-8"));

            int size = log.size();
            for (int i = 0; i < size; i++){
                writer.write(log.get(i).getDate());
                writer.newLine();
                writer.write(log.get(i).getTime());
                writer.newLine();
                writer.write(log.get(i).getActivity());
                writer.newLine();
                writer.write(log.get(i).getLocation());
                writer.newLine();
                writer.write("-end_of_activity-");
                writer.newLine();
            }
            writer.close();
            setStorageChart();
        } catch(Exception e){
        }
    }

    public void onDeleteData(ActionEvent actionEvent){
        int selectedIndex = activityTable.getSelectionModel().
            getSelectedIndex();
        if (selectedIndex >= 0){
            if (logInDialog.logIn()){
                if(activityCell.getCellObservableValue(selectedIndex).
                    getValue().toString().contains("Falling!")){
                    fallTimes--;
                    numberOfFalls.setText(String.valueOf(fallTimes));
                }
                activityTable.getItems().remove(selectedIndex);
                log.remove(selectedIndex);
                feedbackData.setText("Data log was successfully modified
                    !");
                saveLog();
            }
        }

    }

    public void onDeleteAllData(ActionEvent actionEvent) {
        if (logInDialog.logIn()){
            fallTimes = 0;
            numberOfFalls.setText(String.valueOf(fallTimes));
            activityTable.getItems().clear();
            log.clear();
```

```
            feedbackData.setText("Data log was successfully modified!");
            saveLog();
        }
    }

    public void onExportData(ActionEvent actionEvent) throws IOException
         {
        HSSFWorkbook workbook = new HSSFWorkbook();
        HSSFSheet sheet = workbook.createSheet("Fall Analyzer Data Log")
            ;

        HSSFRow row = null;
        HSSFCell cell = null;

        row = sheet.createRow(0);
        cell = row.createCell(0);
        cell.setCellValue("DATE");
        cell = row.createCell(1);
        cell.setCellValue("TIME");
        cell = row.createCell(2);
        cell.setCellValue("ACTIVITY");
        cell = row.createCell(3);
        cell.setCellValue("LOCATION");

        row = sheet.createRow(1);
        cell = row.createCell(0);
        cell.setCellValue("try");

        row = sheet.createRow(2);
        cell = row.createCell(1);
        cell.setCellValue("tr2y");

        for (int rowIterate = 0; rowIterate < activityTable.getItems().
            size(); rowIterate++){
            row = sheet.createRow(rowIterate+1);
            for (int columnIterate = 0; columnIterate < activityTable.
                getColumns().size(); columnIterate++){
                cell = row.createCell(columnIterate);
                switch (columnIterate){
                    case 0: //DATE
                        cell.setCellValue(dateCell.
                            getCellObservableValue(rowIterate).getValue
                            ().toString());
                        break;
                    case 1: //TIME
                        cell.setCellValue(timeCell.
                            getCellObservableValue(rowIterate).getValue
                            ().toString());
                        break;
                    case 2: //ACTIVITY
                        cell.setCellValue(activityCell.
                            getCellObservableValue(rowIterate).getValue
                            ().toString());
                        break;
                    case 3: //LOCATION
                        cell.setCellValue(locationCell.
                            getCellObservableValue(rowIterate).getValue
```

```
                            ().toString());
                        break;
                }
            }
        }

        sheet.autoSizeColumn(0);
        sheet.autoSizeColumn(1);
        sheet.autoSizeColumn(2);
        sheet.autoSizeColumn(3);

        DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd_HHmmss");
        Date date = new Date();
        String fileName = "Fall Analyzer Data Log " + dateFormat.format(
            date) + ".xls";

        workbook.write(new FileOutputStream(fileName));
        workbook.close();

        feedbackData.setText("Data log was successfully exported!");
    }
}
```

Explore.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.chart.*?>
<?import java.lang.*?>
<?import javafx.scene.image.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="explorePane" prefHeight="575.5" prefWidth="1024" stylesheets="
    @style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="dlsu.exploreController">
   <children>
      <Group layoutX="802.0" layoutY="91.0">
         <children>
            <Label id="header2Label" layoutX="154.0" layoutY="10.0"
                stylesheets="@style.css" text="Hello," />
            <TilePane alignment="CENTER_RIGHT" layoutX="-227.0" layoutY
                ="21.0" prefHeight="19.0" prefWidth="434.0">
               <children>
                  <Label id="header1Label" fx:id="usernameHello"
                      stylesheets="@style.css" text="Username!"
                      textAlignment="RIGHT" />
               </children>
            </TilePane>
         </children>
      </Group>
      <TabPane layoutY="144.0" prefHeight="438.0" prefWidth="1025.0"
          tabClosingPolicy="UNAVAILABLE">
        <tabs>
```

```
            <Tab fx:id="profileTab" text="Profile">
              <content>
                <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight
                    ="424.0" prefWidth="1026.0">
                  <children>
                          <Group layoutX="499.0" layoutY="-8.0">
                             <children>
                                <TilePane layoutX="97.0" layoutY="260.0"
                                    prefHeight="75.0" prefWidth="161.0">
                                   <children>
                                      <Line endX="100.0" startX="-100.0"
                                          stroke="#009fe0" />
                                      <Label id="errorLabel" fx:id="
                                          feedbackLabelLogin" stylesheets
                                          ="@style.css" textAlignment="
                                          CENTER" />
                                      <HBox alignment="CENTER" prefHeight
                                          ="25.0" prefWidth="78.0" spacing
                                          ="5.0">
                                         <children>
                                            <Button fx:id="editLogin"
                                                mnemonicParsing="false"
                                                onAction="#onEditLogin"
                                                text="edit" />
                                            <Button fx:id="saveLogin"
                                                disable="true"
                                                mnemonicParsing="false"
                                                onAction="#onSaveLogin"
                                                text="save" />
                                         </children>
                                      </HBox>
                                   </children>
                                </TilePane>
                            <GridPane hgap="10.0" layoutX="66.0" layoutY
                                ="160.0" prefHeight="90.0" prefWidth
                                ="263.0">
                                <columnConstraints>
                                    <ColumnConstraints hgrow="SOMETIMES"
                                        maxWidth="115.0" minWidth="10.0"
                                        prefWidth="91.0" />
                                    <ColumnConstraints hgrow="ALWAYS"
                                        maxWidth="162.0" minWidth="10.0"
                                        prefWidth="162.0" />
                                </columnConstraints>
                                <rowConstraints>
                                    <RowConstraints minHeight="10.0"
                                        prefHeight="30.0" vgrow="SOMETIMES
                                        " />
                                    <RowConstraints minHeight="10.0"
                                        prefHeight="30.0" vgrow="SOMETIMES
                                        " />
                                    <RowConstraints minHeight="10.0"
                                        prefHeight="30.0" vgrow="SOMETIMES
                                        " />
                                </rowConstraints>
                                <children>
                                    <Label id="fieldsLabelLabel"
```

```
                                stylesheets="@style.css" text="
                                Username:" />
                            <Label id="fieldsLabelLabel"
                                stylesheets="@style.css" text="
                                Password:" GridPane.rowIndex="1"
                                />
                            <Label id="fieldsLabelLabel"
                                stylesheets="@style.css" text="
                                Confirm:" GridPane.rowIndex="2" />
                            <TextField fx:id="usernameLogin"
                                editable="false" prefHeight="25.0"
                                 prefWidth="162.0" promptText="
                                username" GridPane.columnIndex="1"
                                 />
                            <PasswordField fx:id="passwordLogin"
                                editable="false" promptText="
                                password" GridPane.columnIndex="1"
                                 GridPane.rowIndex="1" />
                            <PasswordField fx:id="confirmLogin"
                                editable="false" prefHeight="25.0"
                                 prefWidth="117.0" promptText="
                                confirm password" GridPane.
                                columnIndex="1" GridPane.rowIndex
                                ="2" />
                        </children>
                    </GridPane>
                    <Label id="header3Label" layoutX="66.0"
                        layoutY="126.0" stylesheets="@style.css"
                        text="Login credentials:" textAlignment="
                        JUSTIFY" />
                </children>
            </Group>
        <Group layoutX="-235.0" layoutY="-168.0">
            <children>
                <Label id="header3Label" layoutX="379.0"
                    layoutY="248.0" stylesheets="@style.css"
                    text="Profile:" textAlignment="JUSTIFY"
                    />
                <GridPane hgap="10.0" layoutX="380.0" layoutY
                    ="275.0" prefHeight="90.0" prefWidth
                    ="278.0">
                    <columnConstraints>
                        <ColumnConstraints hgrow="SOMETIMES"
                            maxWidth="128.66665649414062"
                            minWidth="10.0" prefWidth
                            ="119.3333435058938" />
                        <ColumnConstraints hgrow="ALWAYS"
                            maxWidth="162.0" minWidth="10.0"
                            prefWidth="133.66665649414062" />
                    </columnConstraints>
                    <rowConstraints>
                        <RowConstraints minHeight="10.0"
                            prefHeight="30.0" vgrow="
                            SOMETIMES" />
                        <RowConstraints minHeight="10.0"
                            prefHeight="30.0" vgrow="
                            SOMETIMES" />
```

```
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            </rowConstraints>
            <children>
                <Label id="fieldsLabelLabel" stylesheets="@style.css" text="First Name:" />
                <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Last Name:" GridPane.rowIndex="1" />
                <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Gender:" GridPane.rowIndex="2" />
                <TextField fx:id="firstNameProfile" editable="false" prefHeight="25.0" prefWidth="162.0" promptText="first name" GridPane.columnIndex="1" />
                <TextField fx:id="LastNameProfile" editable="false" prefHeight="25.0" prefWidth="142.0" promptText="last name" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                <ChoiceBox fx:id="genderProfile" disable="true" prefHeight="25.0" prefWidth="82.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />
            </children>
        </GridPane>
        <GridPane hgap="10.0" layoutX="379.0" layoutY="368.0" prefHeight="90.0" prefWidth="279.0">
            <children>
                <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Age:" />
                <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Address:" GridPane.rowIndex="1" />
                <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Contact Number:" GridPane.rowIndex="2" />
                <TextField fx:id="ageProfile" editable="false" prefHeight="25.0" prefWidth="162.0" promptText="age" GridPane.columnIndex="1" />
                <TextField fx:id="addressProfile" editable="false" prefHeight="25.0" prefWidth="146.0" promptText="address" GridPane.
```

```
                                    columnIndex="1" GridPane.rowIndex
                                    ="1" />
                                <TextField fx:id="
                                    contactNumberProfile" editable="
                                    false" prefHeight="25.0"
                                    prefWidth="117.0" promptText="
                                    contact number" text="+639"
                                    GridPane.columnIndex="1" GridPane
                                    .rowIndex="2" />
                            </children>
                            <columnConstraints>
                                <ColumnConstraints hgrow="SOMETIMES"
                                    maxWidth="123.0" minWidth="10.0"
                                    prefWidth="120.0" />
                                <ColumnConstraints hgrow="ALWAYS"
                                    maxWidth="162.0" minWidth="10.0"
                                    prefWidth="133.0" />
                            </columnConstraints>
                            <rowConstraints>
                                <RowConstraints minHeight="10.0"
                                    prefHeight="30.0" vgrow="
                                    SOMETIMES" />
                                <RowConstraints minHeight="10.0"
                                    prefHeight="30.0" vgrow="
                                    SOMETIMES" />
                                <RowConstraints minHeight="10.0"
                                    prefHeight="30.0" vgrow="
                                    SOMETIMES" />
                            </rowConstraints>
                        </GridPane>
                           <TilePane layoutX="419.0" layoutY="467.0"
                               prefHeight="57.0" prefWidth="201.0">
                              <children>
                                 <Line endX="100.0" startX="-100.0"
                                     stroke="#009fe0" />
                              <Label id="errorLabel" fx:id="
                                  feedbackLabel" stylesheets="@style.
                                  css" textAlignment="CENTER" />
                                 <HBox alignment="CENTER" prefHeight
                                     ="25.0" prefWidth="81.0" spacing
                                     ="5.0">
                                    <children>
                                       <Button fx:id="editProfile"
                                           mnemonicParsing="false"
                                           onAction="#onEditProfile"
                                           text="edit" />
                                       <Button fx:id="saveProfile"
                                           disable="true"
                                           mnemonicParsing="false"
                                           onAction="#onSaveProfile"
                                           text="save" />
                                    </children>
                                 </HBox>
                              </children>
                           </TilePane>
                    </children>
                 </Group>
```

De La Salle University

```
                    </children>
                  </AnchorPane>
                </content>
              </Tab>
              <Tab fx:id="respondersTab" text="Responders">
                <content>
                  <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight
                      ="180.0" prefWidth="200.0">
                        <children>
                           <TableView fx:id="respondersTable" layoutX
                               ="-1.0" layoutY="-3.0" prefHeight="365.0"
                               prefWidth="527.0">
                             <columns>
                               <TableColumn fx:id="firstName" prefWidth
                                   ="163.3333282470703" text="First Name"
                                   />
                                 <TableColumn fx:id="lastName" prefWidth
                                     ="173.0" text="Last Name" />
                                 <TableColumn fx:id="contactNumber"
                                     prefWidth="189.0" text="Contact Number
                                     " />
                             </columns>
                           </TableView>
                            <Group fx:id="editResponders" disable="true"
                                layoutX="236.0" layoutY="-139.0">
                                <children>
                                    <Label id="header2Label" layoutX
                                        ="428.0" layoutY="247.0"
                                        stylesheets="@style.css" text="Let'
                                        s modify your responders!"
                                        textAlignment="JUSTIFY" />
                                    <GridPane hgap="10.0" layoutX="380.0"
                                        layoutY="275.0" prefHeight="90.0"
                                        prefWidth="328.0">
                                        <columnConstraints>
                                            <ColumnConstraints hgrow="
                                                SOMETIMES" maxWidth
                                                ="137.6666259765625"
                                                minWidth="10.0" prefWidth
                                                ="103.33334350585938" />
                                            <ColumnConstraints hgrow="
                                                ALWAYS" maxWidth
                                                ="196.66665649414062"
                                                minWidth="10.0" prefWidth
                                                ="196.66665649414062" />
                                        </columnConstraints>
                                        <rowConstraints>
                                            <RowConstraints minHeight
                                                ="10.0" prefHeight="30.0"
                                                vgrow="SOMETIMES" />
                                            <RowConstraints minHeight
                                                ="10.0" prefHeight="30.0"
                                                vgrow="SOMETIMES" />
                                            <RowConstraints minHeight
                                                ="10.0" prefHeight="30.0"
                                                vgrow="SOMETIMES" />
                                        </rowConstraints>
```

```
                                    <children>
                                        <Label id="fieldsLabelLabel"
                                            stylesheets="@style.css"
                                            text="First Name:" />
                                        <Label id="fieldsLabelLabel"
                                            stylesheets="@style.css"
                                            text="Last Name:" GridPane.
                                            rowIndex="1" />
                                        <TextField fx:id="
                                            firstNameResponders"
                                            prefHeight="25.0" prefWidth
                                            ="162.0" promptText="first
                                            name" GridPane.columnIndex
                                            ="1" />
                                        <TextField fx:id="
                                            LastNameResponders"
                                            prefHeight="25.0" prefWidth
                                            ="182.0" promptText="last
                                            name" GridPane.columnIndex
                                            ="1" GridPane.rowIndex="1"
                                            />
                                        <Label id="fieldsLabelLabel"
                                            stylesheets="@style.css"
                                            text="Contact Number:"
                                            GridPane.rowIndex="2" />
                                        <TextField fx:id="
                                            contactNumberResponders"
                                            prefHeight="25.0" prefWidth
                                            ="117.0" promptText="
                                            contact number" text="+639"
                                             GridPane.columnIndex="1"
                                            GridPane.rowIndex="2" />
                                    </children>
                                </GridPane>
                                <Button fx:id="saveResponder" layoutX
                                    ="526.0" layoutY="375.0"
                                    mnemonicParsing="false" onAction="#
                                    onSaveResponder" text="save" />
                             <Label id="errorLabel" fx:id="
                                 allFieldsError" layoutX="491.0"
                                 layoutY="415.0" stylesheets="@style.
                                 css" textAlignment="CENTER" />
                             <Label id="errorLabel" fx:id="successful"
                                 layoutX="439.0" layoutY="415.0"
                                 stylesheets="@style.css" textAlignment
                                 ="CENTER" />
                         </children>
                     </Group>
                    <HBox layoutX="374.0" layoutY="371.0" prefHeight
                        ="29.0" prefWidth="137.0" spacing="5.0">
                        <children>
                            <Button fx:id="deleteResponder"
                                mnemonicParsing="false" onAction="#
                                onDeleteResponder" text="delete" />
                            <Button fx:id="editResponder"
                                mnemonicParsing="false" onAction="#
                                onEditResponder" text="edit" />
```

```
                              <Button fx:id="addResponder"
                                 mnemonicParsing="false" onAction="#
                                 onAddResponder" text="add" />
                           </children>
                        </HBox>
                     </children>
                  </AnchorPane>
            </content>
          </Tab>
            <Tab fx:id="activityTab" text="Activity Log">
               <content>
                  <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight
                     ="180.0" prefWidth="200.0">
                     <children>
                        <TableView fx:id="activityTable" prefHeight
                           ="402.0" prefWidth="659.0">
                           <columns>
                              <TableColumn fx:id="dateCell" prefWidth
                                 ="146.33333206176758" text="Date" />
                              <TableColumn fx:id="timeCell" prefWidth
                                 ="161.6666717529297" text="Time" />
                              <TableColumn fx:id="activityCell"
                                 prefWidth="171.33331298828125" text="
                                 Activity" />
                              <TableColumn fx:id="locationCell"
                                 prefWidth="176.66668701171875" text="
                                 Location" />
                           </columns>
                        </TableView>
                        <TilePane alignment="CENTER" layoutX="671.0"
                           layoutY="325.0" prefHeight="59.0" prefWidth
                           ="346.0">
                           <children>
                              <Label id="feedbackLabel" fx:id="
                                 feedbackData" stylesheets="@style.css"
                                  textAlignment="CENTER" />
                              <HBox alignment="TOP_CENTER" prefHeight
                                 ="26.0" prefWidth="241.0" spacing
                                 ="5.0">
                                 <children>
                                    <Button id="respondButtons" fx:id="
                                       deleteData" mnemonicParsing="
                                       false" onAction="#onDeleteData"
                                       prefHeight="30.0" prefWidth
                                       ="71.0" stylesheets="@style.css"
                                        text="delete">
                                        <tooltip>
                                            <Tooltip text="Delete high-
                                               lighted data log" />
                                        </tooltip>
                                    </Button>
                                     <Button id="respondButtons" fx:id="
                                        deleteAllData" mnemonicParsing
                                        ="false" onAction="#
                                        onDeleteAllData" stylesheets="
                                        @style.css" text="clear all">
                                         <tooltip>
```

```
                                        <Tooltip text="Clear the entire data log" />
                                    </tooltip>
                                </Button>
                                <Button id="respondButtons" fx:id="exportData" mnemonicParsing="false" onAction="#onExportData" stylesheets="@style.css" text="export">
                                    <tooltip>
                                        <Tooltip text="Export data log to an Excel File" />
                                    </tooltip>
                                </Button>
                            </children>
                        </HBox>
                    </children>
                </TilePane>
                <GridPane layoutX="726.0" layoutY="296.0" prefHeight="30.0" prefWidth="246.0">
                  <columnConstraints>
                    <ColumnConstraints hgrow="SOMETIMES" maxWidth="230.0" minWidth="10.0" prefWidth="206.0" />
                    <ColumnConstraints hgrow="SOMETIMES" maxWidth="95.0" minWidth="10.0" prefWidth="13.0" />
                  </columnConstraints>
                  <rowConstraints>
                    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                  </rowConstraints>
                   <children>
                      <Label id="header2Label" text="Number of recorded falls:" />
                      <Label id="header3Label" fx:id="numberOfFalls" text="0" GridPane.columnIndex="1" />
                   </children>
                </GridPane>
                <PieChart id="feedbackLabel" fx:id="storageSpace" layoutX="666.0" layoutY="48.0" prefHeight="225.0" prefWidth="351.0" stylesheets="@style.css" />
                <Label id="header3Label" layoutX="762.0" layoutY="14.0" stylesheets="@style.css" text="Storage Usage:" />
             </children></AnchorPane>
          </content>
       </Tab>
    </tabs>
   </TabPane>
  </children>
</Pane>
```

De La Salle University

selectCardController.java

```
package dlsu;


import dlsu.Utils.getExternalDevices;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Cursor;
import javafx.scene.control.*;

import java.io.*;
import java.net.URL;
import java.util.*;

public class selectCardController implements Initializable{
    dlsu.Utils.getExternalDevices getExternalDevices = new
        getExternalDevices();

    @FXML
    private Button onNext;

    @FXML
    private Label feedbackLabel;

    @FXML
    private ComboBox selectCard;

    public static String driveLetter;


    @Override
    public void initialize(URL location, ResourceBundle resources) {
        getExternalDevices.getDevices();
        System.out.println("Storage Devices: " + getExternalDevices.
            storages);
        for (int i = 0; i < getExternalDevices.storages.size(); i++){
            selectCard.getItems().add(getExternalDevices.storages.get(i)
                );
        }
        onNext.setOnMouseEntered(event -> onNext.setCursor(Cursor.HAND))
            ;
        onNext.setOnMousePressed(event -> onNext.setCursor(Cursor.
            CLOSED_HAND));
        onNext.setOnMouseReleased(event -> onNext.setCursor(Cursor.HAND)
            );
        onNext.setOnMouseExited(event -> onNext.setCursor(Cursor.DEFAULT
            ));

        selectCard.setOnMouseEntered(event -> selectCard.setCursor(
            Cursor.HAND));
        selectCard.setOnMousePressed(event -> selectCard.setCursor(
            Cursor.CLOSED_HAND));
        selectCard.setOnMouseReleased(event -> selectCard.setCursor(
            Cursor.HAND));
```

```java
        selectCard.setOnMouseExited(event -> selectCard.setCursor(Cursor.DEFAULT));
    }

    @FXML
    private void onNext(ActionEvent actionEvent) throws IOException{
        List<String> messages = new ArrayList<String>();

        messages.add("Please select your card first!");
        messages.add("You haven't selected your card yet.");
        messages.add("Ooopsie! Please select your card please.");
        messages.add("Select your card first.");

        Collections.shuffle(messages);

        try {
            String drive = selectCard.getValue().toString();
            driveLetter = "";
            for (int i = 0; i < drive.length(); i++) {
                if (Objects.equals(drive.charAt(i), '(')) {
                    i++;
                    driveLetter = driveLetter + drive.charAt(i) + ":" + "\\";
                }
            }
            System.out.println("Selected Drive: " + drive);
            System.out.println("Selected Drive Letter: " + driveLetter);

            File loginFile = new File(driveLetter + "login.txt");
            File profileFile = new File(driveLetter + "profile.txt");
            File respondentsFile = new File(driveLetter + "respondents.txt");
            File activityFile = new File(driveLetter + "Activities");
            File fileLog = new File(driveLetter + "filelog.txt");
            if (loginFile.exists() && profileFile.exists() && respondentsFile.exists() && activityFile.exists() && fileLog.exists()) {
                feedbackLabel.setText("");
                changeScene changeScene = new changeScene();
                changeScene.setScene("Explore.fxml", "style.css", actionEvent, "Fall Analyzer");
            } else if (!loginFile.exists() && !profileFile.exists() && !respondentsFile.exists() && !activityFile.exists() && !fileLog.exists()){
                feedbackLabel.setText("");
                changeScene changeScene = new changeScene();
                changeScene.setScene("SetupHome.fxml", "style.css", actionEvent, "Fall Analyzer | Home Setup");
            }else {
                System.out.println("Some of the important files are missing!\nCard must be initialized again.");

                feedbackLabel.setText("Some of the important files are missing!\nCard must be initialized again.");

                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("ERROR!");
```

```
                    String s = "Some of the important files are missing!\
                        nCard must be initialized again.";
                    alert.setContentText(s);

                    Optional<ButtonType> result = alert.showAndWait();

                    changeScene changeScene = new changeScene();
                    changeScene.setScene("SetupHome.fxml", "style.css",
                        actionEvent, "Fall Analyzer | Home Setup");
                }

            } catch (Exception e){
                feedbackLabel.setText(messages.get(0));
                System.out.println(e);
            }

    }

    public void onSelect(ActionEvent actionEvent) {
        List<String> messages = new ArrayList<String>();

        messages.add("Great! Click next to proceed");
        messages.add("Ok! Now, click next to proceed");
        messages.add("Alright! To proceed, click next");
        messages.add("Let's proceed by clicking the next button");

        Collections.shuffle(messages);

        feedbackLabel.setText(messages.get(0));
    }

}
```

SelectCard.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="
    @style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="dlsu.selectCardController">
   <children>
      <TilePane alignment="CENTER" layoutX="352.0" layoutY="317.0"
          prefHeight="152.0" prefWidth="321.0">
         <children>
            <Label id="header1Label" stylesheets="@style.css" text="
                Hello!" />
            <Label id="header2Label" stylesheets="@style.css" text="
                Please choose your card:" />
            <GridPane hgap="5.0">
              <columnConstraints>
```

```
                <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
                    prefWidth="150.0" />
                <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0"
                    prefWidth="50.0" />
              </columnConstraints>
              <rowConstraints>
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow
                    ="SOMETIMES" />
              </rowConstraints>
               <children>
                  <ComboBox fx:id="selectCard" onAction="#onSelect"
                      prefHeight="25.0" prefWidth="154.0" promptText="
                      select your card">
                     <tooltip>
                        <Tooltip text="Select the microSD card for your
                            device." />
                     </tooltip></ComboBox>
                  <Button fx:id="onNext" mnemonicParsing="false"
                      onAction="#onNext" prefHeight="25.0" prefWidth
                      ="41.0" text="Next" GridPane.columnIndex="1" />
               </children>
               <TilePane.margin>
                  <Insets />
               </TilePane.margin>
            </GridPane>
            <Label id="feedbackLabel" fx:id="feedbackLabel" stylesheets
                ="@style.css" />
         </children>
      </TilePane>
         </children>
</Pane>
```

Setup1.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="
    @style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="dlsu.setUp1Controller">
    <children>
      <Group layoutX="1.0" layoutY="42.0">
         <children>
              <Label id="header2Label" layoutX="380.0" layoutY="254.0"
                  stylesheets="@style.css" text="Let's setup your login
                  credentials!" textAlignment="JUSTIFY" />
            <GridPane hgap="10.0" layoutX="380.0" layoutY="288.0"
                prefHeight="90.0" prefWidth="263.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="115.0"
                    minWidth="10.0" prefWidth="91.0" />
                <ColumnConstraints hgrow="ALWAYS" maxWidth="162.0"
                    minWidth="10.0" prefWidth="162.0" />
```

```
            </columnConstraints>
            <rowConstraints>
              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow
                  ="SOMETIMES" />
              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow
                  ="SOMETIMES" />
              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow
                  ="SOMETIMES" />
            </rowConstraints>
             <children>
                <Label id="fieldsLabelLabel" stylesheets="@style.css"
                    text="Username:" />
                <Label id="fieldsLabelLabel" stylesheets="@style.css"
                    text="Password:" GridPane.rowIndex="1" />
                 <Label id="fieldsLabelLabel" stylesheets="@style.css"
                      text="Confirm:" GridPane.rowIndex="2" />
                <TextField fx:id="username" prefHeight="25.0"
                    prefWidth="162.0" promptText="username" GridPane.
                    columnIndex="1" />
                <PasswordField fx:id="password" promptText="password"
                    GridPane.columnIndex="1" GridPane.rowIndex="1" />
                <PasswordField fx:id="confirm" prefHeight="25.0"
                    prefWidth="117.0" promptText="confirm password"
                    GridPane.columnIndex="1" GridPane.rowIndex="2" />
             </children>
          </GridPane>
            <Label id="errorLabel" fx:id="feedbackLabel" layoutX
                ="497.0" layoutY="387.0" stylesheets="@style.css"
                textAlignment="CENTER" />
       </children>
    </Group>
      <Group layoutX="-33.0">
          <children>
              <HBox layoutX="376.0" layoutY="484.0" prefHeight="45.0"
                  prefWidth="310.0" spacing="10.0">
                  <children>
                      <Button id="selected" fx:id="first"
                          mnemonicParsing="false" onAction="#onFirst"
                          prefHeight="38.0" prefWidth="81.0"
                          stylesheets="@style.css" text="1" />
                      <Button id="idle" fx:id="second" mnemonicParsing
                          ="false" onAction="#onSecond" prefHeight
                          ="38.0" prefWidth="81.0" stylesheets="@style
                          .css" text="2" />
                      <Button id="idle" fx:id="third" mnemonicParsing
                          ="false" onAction="#onThird" prefHeight
                          ="38.0" prefWidth="81.0" stylesheets="@style
                          .css" text="3" />
                      <Button id="idle" fx:id="fourth" mnemonicParsing
                          ="false" onAction="#onFourth" prefHeight
                          ="38.0" prefWidth="81.0" stylesheets="@style
                          .css" text="4" />
                      <Button id="start" fx:id="finish"
                          mnemonicParsing="false" onAction="#onFinish"
                           prefHeight="61.0" prefWidth="88.0"
                          stylesheets="@style.css" text="finish" />
                  </children>
```

De La Salle University

```
                </HBox>
            </children>
        </Group>
    </children>
</Pane>
```

Setup2.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="@style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="dlsu.setUp2Controller">
    <children>
      <Group layoutX="-146.0" layoutY="50.0">
         <children>
              <Label id="header2Label" layoutX="566.0" layoutY="229.0" stylesheets="@style.css" text="Let's setup your profile!" textAlignment="JUSTIFY" />
            <GridPane hgap="10.0" layoutX="380.0" layoutY="275.0" prefHeight="90.0" prefWidth="263.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="115.0" minWidth="10.0" prefWidth="91.0" />
                <ColumnConstraints hgrow="ALWAYS" maxWidth="162.0" minWidth="10.0" prefWidth="162.0" />
              </columnConstraints>
              <rowConstraints>
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
              </rowConstraints>
               <children>
                  <Label id="fieldsLabelLabel" stylesheets="@style.css" text="First Name:" />
                  <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Last Name:" GridPane.rowIndex="1" />
                   <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Gender:" GridPane.rowIndex="2" />
                  <TextField fx:id="firstName" prefHeight="25.0" prefWidth="162.0" promptText="first name" GridPane.columnIndex="1" />
                  <TextField fx:id="LastName" promptText="last name" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                  <ChoiceBox fx:id="gender" prefHeight="25.0" prefWidth="82.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />
               </children>
```

```
            </GridPane>
              <Label id="errorLabel" fx:id="feedbackLabel" layoutX
                  ="496.0" layoutY="380.0" stylesheets="@style.css"
                  textAlignment="CENTER" />
            <GridPane hgap="10.0" layoutX="674.0" layoutY="317.0"
                prefHeight="90.0" prefWidth="263.0">
               <children>
                  <Label id="fieldsLabelLabel" stylesheets="@style.css"
                      text="Age:" />
                  <Label id="fieldsLabelLabel" stylesheets="@style.css"
                      text="Address:" GridPane.rowIndex="1" />
                  <Label id="fieldsLabelLabel" stylesheets="@style.css"
                      text="Contact Number:" GridPane.rowIndex="2" />
                  <TextField fx:id="age" prefHeight="25.0" prefWidth
                      ="162.0" promptText="age" GridPane.columnIndex="1"
                       />
                  <TextField fx:id="address" promptText="address"
                      GridPane.columnIndex="1" GridPane.rowIndex="1" />
                  <TextField fx:id="contactNumber" prefHeight="25.0"
                      prefWidth="97.0" promptText="contact number" text
                      ="+639" GridPane.columnIndex="1" GridPane.rowIndex
                      ="2" />
               </children>
               <columnConstraints>
                  <ColumnConstraints hgrow="SOMETIMES" maxWidth="123.0"
                      minWidth="10.0" prefWidth="120.0" />
                  <ColumnConstraints hgrow="ALWAYS" maxWidth="162.0"
                      minWidth="10.0" prefWidth="133.0" />
               </columnConstraints>
               <rowConstraints>
                  <RowConstraints minHeight="10.0" prefHeight="30.0"
                      vgrow="SOMETIMES" />
                  <RowConstraints minHeight="10.0" prefHeight="30.0"
                      vgrow="SOMETIMES" />
                  <RowConstraints minHeight="10.0" prefHeight="30.0"
                      vgrow="SOMETIMES" />
               </rowConstraints>
            </GridPane>
         </children>
      </Group>
        <Group layoutX="-33.0">
            <children>
                <HBox layoutX="376.0" layoutY="484.0" prefHeight="45.0"
                    prefWidth="310.0" spacing="10.0">
                    <children>
                        <Button id="idle" fx:id="first" mnemonicParsing
                            ="false" onAction="#onFirst" prefHeight
                            ="38.0" prefWidth="81.0" stylesheets="@style
                            .css" text="1" />
                        <Button id="selected" fx:id="second"
                            mnemonicParsing="false" onAction="#onSecond"
                             prefHeight="38.0" prefWidth="81.0"
                            stylesheets="@style.css" text="2" />
                        <Button id="idle" fx:id="third" mnemonicParsing
                            ="false" onAction="#onThird" prefHeight
                            ="38.0" prefWidth="81.0" stylesheets="@style
                            .css" text="3" />
```

```
                        <Button id="idle" fx:id="fourth" mnemonicParsing="false" onAction="#onFourth" prefHeight="38.0" prefWidth="81.0" stylesheets="@style.css" text="4" />
                        <Button id="start" fx:id="finish" mnemonicParsing="false" onAction="#onFinish" prefHeight="61.0" prefWidth="88.0" stylesheets="@style.css" text="finish" />
                    </children>
                </HBox>
            </children>
        </Group>
    </children>
</Pane>
```

setUp2Controller.java

```
package dlsu;

import dlsu.Utils.checkContactNumber;
import javafx.event.ActionEvent;
import javafx.fxml.Initializable;
import javafx.scene.Cursor;
import javafx.scene.control.*;

import java.io.*;
import java.net.URL;
import java.util.ResourceBundle;

/**
 * Created by aenon on 11/11/2017.
 */
public class setUp2Controller implements Initializable {
    public Button first;
    public Button second;
    public Button third;
    public Button fourth;
    public ChoiceBox gender;
    public TextField firstName;
    public TextField LastName;
    public TextField age;
    public TextField address;
    public TextField contactNumber;
    public Label feedbackLabel;
    public Button finish;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        File file = new File(selectCardController.driveLetter + "profile.txt");
        if(file.exists()){
            BufferedReader br = null;
            try {
                br = new BufferedReader(new FileReader(selectCardController.driveLetter + "profile.txt"));

```

```
                firstName.setText(br.readLine());
                LastName.setText(br.readLine());
                gender.setValue(br.readLine());
                age.setText(br.readLine());
                address.setText(br.readLine());
                contactNumber.setText(br.readLine());

            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        gender.getItems().add("male");
        gender.getItems().add("female");

        File respondentsFile = new File(selectCardController.driveLetter
             + "respondents.txt");
        if (!respondentsFile.exists()){
            fourth.setDisable(true);
        }

        finish.setDisable(true);

        first.setOnMouseEntered(event -> first.setCursor(Cursor.HAND));
        first.setOnMousePressed(event -> first.setCursor(Cursor.
            CLOSED_HAND));
        first.setOnMouseReleased(event -> first.setCursor(Cursor.HAND));
        first.setOnMouseExited(event -> first.setCursor(Cursor.DEFAULT))
            ;

        second.setOnMouseEntered(event -> second.setCursor(Cursor.HAND))
            ;
        second.setOnMousePressed(event -> second.setCursor(Cursor.
            CLOSED_HAND));
        second.setOnMouseReleased(event -> second.setCursor(Cursor.HAND)
            );
        second.setOnMouseExited(event -> second.setCursor(Cursor.DEFAULT
            ));

        third.setOnMouseEntered(event -> third.setCursor(Cursor.HAND));
        third.setOnMousePressed(event -> third.setCursor(Cursor.
            CLOSED_HAND));
        third.setOnMouseReleased(event -> third.setCursor(Cursor.HAND));
        third.setOnMouseExited(event -> third.setCursor(Cursor.DEFAULT))
            ;

        fourth.setOnMouseEntered(event -> fourth.setCursor(Cursor.HAND))
            ;
        fourth.setOnMousePressed(event -> fourth.setCursor(Cursor.
```

```
CLOSED_HAND));
        fourth.setOnMouseReleased(event -> fourth.setCursor(Cursor.HAND));
        fourth.setOnMouseExited(event -> fourth.setCursor(Cursor.DEFAULT));

        finish.setOnMouseEntered(event -> finish.setCursor(Cursor.HAND));
        finish.setOnMousePressed(event -> finish.setCursor(Cursor.CLOSED_HAND));
        finish.setOnMouseReleased(event -> finish.setCursor(Cursor.HAND));
        finish.setOnMouseExited(event -> finish.setCursor(Cursor.DEFAULT));
    }

    public void onFirst(ActionEvent actionEvent) throws IOException {
            try{
                Integer.parseInt(age.getText());

                BufferedWriter writer = null;
                try{
                    writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(selectCardController.driveLetter + "profile.txt"), "utf-8"));
                    writer.write(firstName.getText());
                    writer.newLine();
                    writer.write(LastName.getText());
                    writer.newLine();
                    writer.write(gender.getValue().toString());
                    writer.newLine();
                    writer.write(age.getText());
                    writer.newLine();
                    writer.write(address.getText());
                    writer.newLine();
                    writer.write(contactNumber.getText());
                    writer.close();
                } catch(Exception e){
                }
            }catch(Exception e){
                feedbackLabel.setText("Your age is invalid!");
            }finally {
                feedbackLabel.setText("");
                changeScene changeScene = new changeScene();
                changeScene.setScene("Setup1.fxml", "style.css", actionEvent, "Fall Analyzer | Login Credentials");
            }
    }

    public void onSecond(ActionEvent actionEvent) {
    }

    public void onThird(ActionEvent actionEvent) throws IOException {
        if(checkContactNumber.validContactNumber(contactNumber.getText())) {
            if (firstName.getText().isEmpty() || LastName.getText().isEmpty() || age.getText().isEmpty() || address.getText
```

```
                ().isEmpty() || contactNumber.getText().isEmpty()) {
                feedbackLabel.setText("All fields are required!");
            } else {
                try {
                    Integer.parseInt(age.getText());

                    BufferedWriter writer = null;
                    try {
                        writer = new BufferedWriter(new
                            OutputStreamWriter(new FileOutputStream(
                            selectCardController.driveLetter + "profile.
                            txt"), "utf-8"));
                        writer.write(firstName.getText());
                        writer.newLine();
                        writer.write(LastName.getText());
                        writer.newLine();
                        writer.write(gender.getValue().toString());
                        writer.newLine();
                        writer.write(age.getText());
                        writer.newLine();
                        writer.write(address.getText());
                        writer.newLine();
                        writer.write(contactNumber.getText());
                        writer.close();
                    } catch (Exception e) {
                    }
                } catch (Exception e) {
                    feedbackLabel.setText("Your age is invalid!");
                } finally {
                    feedbackLabel.setText("");
                    changeScene changeScene = new changeScene();
                    changeScene.setScene("Setup3.fxml", "style.css",
                        actionEvent, "Fall Analyzer | Responders Setup")
                        ;
                }
            }
        } else {
            feedbackLabel.setText("Your contact number is invalid!");
        }
    }

    public void onFourth(ActionEvent actionEvent) throws IOException {
        if(checkContactNumber.validContactNumber(contactNumber.getText()
            )) {
            try{
                Integer.parseInt(age.getText());

                BufferedWriter writer = null;
                try{
                    writer = new BufferedWriter(new OutputStreamWriter(
                        new FileOutputStream(selectCardController.
                        driveLetter + "profile.txt"), "utf-8"));
                    writer.write(firstName.getText());
                    writer.newLine();
                    writer.write(LastName.getText());
                    writer.newLine();
                    writer.write(gender.getValue().toString());
```

De La Salle University

```
                    writer.newLine();
                    writer.write(age.getText());
                    writer.newLine();
                    writer.write(address.getText());
                    writer.newLine();
                    writer.write(contactNumber.getText());
                    writer.close();
                } catch(Exception e){
                }
            }catch(Exception e){
                feedbackLabel.setText("Your age is invalid!");
            }finally {
                feedbackLabel.setText("");
                changeScene changeScene = new changeScene();
                changeScene.setScene("Setup4.fxml", "style.css",
                    actionEvent, "Fall Analyzer | Finish Setup");
            }
        } else {
            feedbackLabel.setText("Your contact number is invalid!");
        }
    }

    public void onFinish(ActionEvent actionEvent) {

    }
}
```

Setup3.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="
    @style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="dlsu.setUp3Controller">
    <children>
        <Group layoutX="-18.0" layoutY="-6.0">
            <children>
                <Group layoutX="-79.0" layoutY="16.0">
                    <children>
                        <ListView fx:id="list" layoutX="708.0" layoutY="310.0"
                             prefHeight="119.0" prefWidth="200.0" />
                        <Label id="header2Label" layoutX="706.0" layoutY
                            ="275.0" stylesheets="@style.css" text="Here are
                            your responders:" textAlignment="JUSTIFY" />
                        <Button layoutX="817.0" layoutY="445.0"
                            mnemonicParsing="false" onAction="#onEdit" text="
                            edit" />
                        <Button layoutX="858.0" layoutY="445.0"
                            mnemonicParsing="false" onAction="#onDelete" text
                            ="delete" />
                    </children>
```

```
            </Group>
            <Group layoutX="-146.0" layoutY="50.0">
               <children>
                    <Label id="header2Label" layoutX="441.0" layoutY="243.0" stylesheets="@style.css" text="Let's add your responders!" textAlignment="JUSTIFY" />
                  <GridPane hgap="10.0" layoutX="380.0" layoutY="275.0" prefHeight="90.0" prefWidth="328.0">
                    <columnConstraints>
                      <ColumnConstraints hgrow="SOMETIMES" maxWidth="137.6666259765625" minWidth="10.0" prefWidth="103.33334350585938" />
                      <ColumnConstraints hgrow="ALWAYS" maxWidth="196.66665649414062" minWidth="10.0" prefWidth="196.66665649414062" />
                    </columnConstraints>
                    <rowConstraints>
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                    </rowConstraints>
                     <children>
                        <Label id="fieldsLabelLabel" stylesheets="@style.css" text="First Name:" />
                        <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Last Name:" GridPane.rowIndex="1" />
                        <TextField fx:id="firstName" prefHeight="25.0" prefWidth="162.0" promptText="first name" GridPane.columnIndex="1" />
                        <TextField fx:id="LastName" prefHeight="25.0" prefWidth="182.0" promptText="last name" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                        <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Contact Number:" GridPane.rowIndex="2" />
                        <TextField fx:id="contactNumber" prefHeight="25.0" prefWidth="117.0" promptText="contact number" text="+639" GridPane.columnIndex="1" GridPane.rowIndex="2" />
                     </children>
                  </GridPane>
                  <Button fx:id="add" layoutX="526.0" layoutY="375.0" mnemonicParsing="false" onAction="#onAdd" text="add" />
               </children>
            </Group>
              <Label id="errorLabel" fx:id="feedbackLabel" layoutX="382.0" layoutY="466.0" stylesheets="@style.css" textAlignment="CENTER" />
         </children>
      </Group>
        <Group layoutX="-33.0">
```

```
                <children>
                    <HBox layoutX="376.0" layoutY="484.0" prefHeight="45.0"
                        prefWidth="310.0" spacing="10.0">
                        <children>
                            <Button id="idle" fx:id="first" mnemonicParsing
                                ="false" onAction="#onFirst" prefHeight
                                ="38.0" prefWidth="81.0" stylesheets="@style
                                .css" text="1" />
                            <Button id="idle" fx:id="second" mnemonicParsing
                                ="false" onAction="#onSecond" prefHeight
                                ="38.0" prefWidth="81.0" stylesheets="@style
                                .css" text="2" />
                            <Button id="selected" fx:id="third"
                                mnemonicParsing="false" onAction="#onThird"
                                prefHeight="38.0" prefWidth="81.0"
                                stylesheets="@style.css" text="3" />
                            <Button id="idle" fx:id="fourth" mnemonicParsing
                                ="false" onAction="#onFourth" prefHeight
                                ="38.0" prefWidth="81.0" stylesheets="@style
                                .css" text="4" />
                            <Button id="start" fx:id="finish"
                                mnemonicParsing="false" onAction="#onFinish"
                                 prefHeight="61.0" prefWidth="88.0"
                                stylesheets="@style.css" text="finish" />
                        </children>
                    </HBox>
                </children>
            </Group>
        </children>
</Pane>
```

setUp3Controller.java

```
package dlsu;

import dlsu.Utils.checkContactNumber;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.Initializable;
import javafx.scene.Cursor;
import javafx.scene.control.*;

import java.io.*;
import java.net.URL;
import java.util.ResourceBundle;

/**
 * Created by aenon on 11/11/2017.
 */
public class setUp3Controller implements Initializable {
    public Button first;
    public Button second;
    public Button third;
    public Button fourth;
```

De La Salle University

```
    public TextField firstName;
    public TextField LastName;
    public TextField contactNumber;
    public Label feedbackLabel;
    public ListView list;

    public final ObservableList <responders> data = FXCollections.
        observableArrayList();
    public Button finish;

    public static class responders{
        private SimpleStringProperty firstName;
        private SimpleStringProperty lastName;
        private SimpleStringProperty contactNumber;

        private responders(String first, String last, String contact){
            this.firstName = new SimpleStringProperty(first);
            this.lastName = new SimpleStringProperty(last);
            this.contactNumber = new SimpleStringProperty(contact);
        }

        public String getFirstName(){
            return firstName.get();
        }

        public void setFirstName(String first){
            firstName.set(first);
        }

        public String getLastName(){
            return lastName.get();
        }

        public void setLastName(String last){
            lastName.set(last);
        }

        public String getContactNumber(){
            return contactNumber.get();
        }

        public void setContactNumber(String contact){
            contactNumber.set(contact);
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        File respondentsFile = new File(selectCardController.driveLetter
             + "respondents.txt");
        if (respondentsFile.exists()){
            BufferedReader br = null;
            try {
                br = new BufferedReader(new FileReader(
                    selectCardController.driveLetter + "respondents.txt
                    "));

```

```
                Boolean end = false;
                String line;
                String first = "";
                String last = "";
                String contact = "";
                while(end == false){
                    line = br.readLine();
                    if (line.equals("-end-")){
                        end = true;
                    } else if (line.equals("breakLine")){
                        //just do nothing
                    } else{
                        first = line;
                        last = br.readLine();
                        contact = br.readLine();
                        data.add(new responders(first, last, contact));
                        list.getItems().add(first + " " + last);
                    }
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        finish.setDisable(true);

        first.setOnMouseEntered(event -> first.setCursor(Cursor.HAND));
        first.setOnMousePressed(event -> first.setCursor(Cursor.
            CLOSED_HAND));
        first.setOnMouseReleased(event -> first.setCursor(Cursor.HAND));
        first.setOnMouseExited(event -> first.setCursor(Cursor.DEFAULT))
            ;

        second.setOnMouseEntered(event -> second.setCursor(Cursor.HAND))
            ;
        second.setOnMousePressed(event -> second.setCursor(Cursor.
            CLOSED_HAND));
        second.setOnMouseReleased(event -> second.setCursor(Cursor.HAND)
            );
        second.setOnMouseExited(event -> second.setCursor(Cursor.DEFAULT
            ));

        third.setOnMouseEntered(event -> third.setCursor(Cursor.HAND));
        third.setOnMousePressed(event -> third.setCursor(Cursor.
            CLOSED_HAND));
        third.setOnMouseReleased(event -> third.setCursor(Cursor.HAND));
        third.setOnMouseExited(event -> third.setCursor(Cursor.DEFAULT))
            ;

```

De La Salle University

```
        fourth.setOnMouseEntered(event -> fourth.setCursor(Cursor.HAND));
        fourth.setOnMousePressed(event -> fourth.setCursor(Cursor.CLOSED_HAND));
        fourth.setOnMouseReleased(event -> fourth.setCursor(Cursor.HAND));
        fourth.setOnMouseExited(event -> fourth.setCursor(Cursor.DEFAULT));

        finish.setOnMouseEntered(event -> finish.setCursor(Cursor.HAND));
        finish.setOnMousePressed(event -> finish.setCursor(Cursor.CLOSED_HAND));
        finish.setOnMouseReleased(event -> finish.setCursor(Cursor.HAND));
        finish.setOnMouseExited(event -> finish.setCursor(Cursor.DEFAULT));
    }

    public void onAdd(ActionEvent actionEvent) {
        if (firstName.getText().isEmpty() || LastName.getText().isEmpty() || contactNumber.getText().isEmpty()){
            feedbackLabel.setText("All fields are required!");
        }else{
            if (checkContactNumber.validContactNumber(contactNumber.getText())){
                list.getItems().add(firstName.getText() + " " + LastName.getText());
                data.add(new responders(firstName.getText(), LastName.getText(), contactNumber.getText()));
                firstName.setText("");
                LastName.setText("");
                contactNumber.setText("+639");
                feedbackLabel.setText("");
            }else {
                feedbackLabel.setText("Your contact number is invalid!");
                contactNumber.setText("+639");
            }
        }
    }

    public void onDelete(ActionEvent actionEvent) {
        try{
            int index   = list.getSelectionModel().getSelectedIndex();
            list.getItems().remove(index);
            data.remove(index);
        }catch (Exception e){
        }
    }

    public void onEdit(ActionEvent actionEvent) {
        try{
            int index   = list.getSelectionModel().getSelectedIndex();
            firstName.setText(String.valueOf(data.get(index).getFirstName()));
            LastName.setText(String.valueOf(data.get(index).getLastName
```

```
                ()));
            contactNumber.setText(String.valueOf(data.get(index).
                getContactNumber()));
            data.remove(index);
            list.getItems().remove(index);
        }catch (Exception e){
        }
    }

    public void onFirst(ActionEvent actionEvent) throws IOException {
        if (data.isEmpty()){
            feedbackLabel.setText("You must set at least one (1)
                responder!");
        }else {
            BufferedWriter writer = null;
            try {
                writer = new BufferedWriter(new OutputStreamWriter(new
                    FileOutputStream(selectCardController.driveLetter +
                    "respondents.txt"), "utf-8"));

                int size = data.size();
                for (int i = 0; i < size; i++) {
                    writer.write(data.get(i).getFirstName());
                    writer.newLine();
                    writer.write(data.get(i).getLastName());
                    writer.newLine();
                    writer.write(data.get(i).getContactNumber());
                    writer.newLine();
                    writer.write("breakLine");
                    writer.newLine();
                }
                writer.write("-end-");
                writer.close();
            } catch (Exception e) {
            }
            feedbackLabel.setText("");
            changeScene changeScene = new changeScene();
            changeScene.setScene("Setup1.fxml", "style.css", actionEvent
                , "Fall Analyzer | Login Credentials");
        }
    }

    public void onSecond(ActionEvent actionEvent) throws IOException {
        if (data.isEmpty()){
            feedbackLabel.setText("You must set at least one (1)
                responder!");
        }else {
            BufferedWriter writer = null;
            try {
                writer = new BufferedWriter(new OutputStreamWriter(new
                    FileOutputStream(selectCardController.driveLetter +
                    "respondents.txt"), "utf-8"));

                int size = data.size();
                for (int i = 0; i < size; i++) {
                    writer.write(data.get(i).getFirstName());
                    writer.newLine();
```

```
                    writer.write(data.get(i).getLastName());
                    writer.newLine();
                    writer.write(data.get(i).getContactNumber());
                    writer.newLine();
                    writer.write("breakLine");
                    writer.newLine();
                }
                writer.write("-end-");
                writer.close();
            } catch (Exception e) {
            }
            feedbackLabel.setText("");
            changeScene changeScene = new changeScene();
            changeScene.setScene("Setup2.fxml", "style.css", actionEvent
                , "Fall Analyzer | Profile Setup");
        }
    }

    public void onThird(ActionEvent actionEvent) {
    }

    public void onFourth(ActionEvent actionEvent) throws IOException {
        if (data.isEmpty()){
            feedbackLabel.setText("You must set at least one (1)
                responder!");
        }else{
            BufferedWriter writer = null;
            try{
                writer = new BufferedWriter(new OutputStreamWriter(new
                    FileOutputStream(selectCardController.driveLetter +
                    "respondents.txt"), "utf-8"));

                int size = data.size();
                for (int i = 0; i < size; i++){
                    writer.write(data.get(i).getFirstName());
                    writer.newLine();
                    writer.write(data.get(i).getLastName());
                    writer.newLine();
                    writer.write(data.get(i).getContactNumber());
                    writer.newLine();
                    writer.write("breakLine");
                    writer.newLine();
                }
                writer.write("-end-");
                writer.close();
            } catch(Exception e){
            }
            feedbackLabel.setText("");
            changeScene changeScene = new changeScene();
            changeScene.setScene("Setup4.fxml", "style.css", actionEvent
                , "Fall Analyzer | Finish Setup");
        }

    }

    public void onFinish(ActionEvent actionEvent) {
    }
```

```
}
```

Setup4.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="@style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="dlsu.setUp4Controller">
    <children>
      <Group layoutX="1.0" layoutY="40.0">
         <children>
              <Label id="header2Label" layoutX="409.0" layoutY="259.0" stylesheets="@style.css" text="To finish setup, enter your" textAlignment="CENTER" />
            <GridPane hgap="10.0" layoutX="380.0" layoutY="315.0" prefHeight="68.0" prefWidth="263.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="115.0" minWidth="10.0" prefWidth="91.0" />
                <ColumnConstraints hgrow="ALWAYS" maxWidth="162.0" minWidth="10.0" prefWidth="162.0" />
              </columnConstraints>
              <rowConstraints>
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
              </rowConstraints>
               <children>
                  <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Username:" />
                  <Label id="fieldsLabelLabel" stylesheets="@style.css" text="Password:" GridPane.rowIndex="1" />
                  <TextField fx:id="username" prefHeight="25.0" prefWidth="162.0" promptText="username" GridPane.columnIndex="1" />
                  <PasswordField fx:id="password" promptText="password" GridPane.columnIndex="1" GridPane.rowIndex="1" />
               </children>
            </GridPane>
              <Label id="errorLabel" fx:id="feedbackLabel" layoutX="497.0" layoutY="393.0" stylesheets="@style.css" textAlignment="CENTER" />
            <Label id="header2Label" layoutX="389.0" layoutY="278.0" stylesheets="@style.css" text="username and password below:" textAlignment="CENTER" />
         </children>
      </Group>
      <Group layoutX="-33.0">
         <children>
```

```
                <HBox layoutX="376.0" layoutY="484.0" prefHeight="45.0"
                    prefWidth="310.0" spacing="10.0">
                    <children>
                        <Button id="idle" fx:id="first" mnemonicParsing="
                            false" onAction="#onFirst" prefHeight="38.0"
                            prefWidth="81.0" stylesheets="@style.css" text
                            ="1" />
                        <Button id="idle" fx:id="second" mnemonicParsing="
                            false" onAction="#onSecond" prefHeight="38.0"
                            prefWidth="81.0" stylesheets="@style.css" text="2"
                             />
                        <Button id="idle" fx:id="third" mnemonicParsing="false
                            " onAction="#onThird" prefHeight="38.0" prefWidth
                            ="81.0" stylesheets="@style.css" text="3" />
                        <Button id="selected" fx:id="fourth" mnemonicParsing="
                            false" onAction="#onFourth" prefHeight="38.0"
                            prefWidth="81.0" stylesheets="@style.css" text="4"
                             />
                        <Button id="start" fx:id="finish" mnemonicParsing="
                            false" onAction="#onFinish" prefHeight="61.0"
                            prefWidth="88.0" stylesheets="@style.css" text="
                            finish" />
                    </children>
                </HBox>
            </children>
        </Group>
    </children>
</Pane>
```

setUp4Controller.java

```
package dlsu;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Cursor;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;

import java.io.*;
import java.net.URL;
import java.util.ResourceBundle;

/**
 * Created by aenon on 11/11/2017.
 */
public class setUp4Controller implements Initializable {
    public Button first;
    public Button second;
    public Button third;
    public Button fourth;
    public Button finish;
    @FXML
```

```
    private TextField username;
    @FXML
    private PasswordField password;
    @FXML
    private Label feedbackLabel;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        first.setOnMouseEntered(event -> first.setCursor(Cursor.HAND));
        first.setOnMousePressed(event -> first.setCursor(Cursor.
            CLOSED_HAND));
        first.setOnMouseReleased(event -> first.setCursor(Cursor.HAND));
        first.setOnMouseExited(event -> first.setCursor(Cursor.DEFAULT))
            ;

        second.setOnMouseEntered(event -> second.setCursor(Cursor.HAND))
            ;
        second.setOnMousePressed(event -> second.setCursor(Cursor.
            CLOSED_HAND));
        second.setOnMouseReleased(event -> second.setCursor(Cursor.HAND)
            );
        second.setOnMouseExited(event -> second.setCursor(Cursor.DEFAULT
            ));

        third.setOnMouseEntered(event -> third.setCursor(Cursor.HAND));
        third.setOnMousePressed(event -> third.setCursor(Cursor.
            CLOSED_HAND));
        third.setOnMouseReleased(event -> third.setCursor(Cursor.HAND));
        third.setOnMouseExited(event -> third.setCursor(Cursor.DEFAULT))
            ;

        fourth.setOnMouseEntered(event -> fourth.setCursor(Cursor.HAND))
            ;
        fourth.setOnMousePressed(event -> fourth.setCursor(Cursor.
            CLOSED_HAND));
        fourth.setOnMouseReleased(event -> fourth.setCursor(Cursor.HAND)
            );
        fourth.setOnMouseExited(event -> fourth.setCursor(Cursor.DEFAULT
            ));

        finish.setOnMouseEntered(event -> finish.setCursor(Cursor.HAND))
            ;
        finish.setOnMousePressed(event -> finish.setCursor(Cursor.
            CLOSED_HAND));
        finish.setOnMouseReleased(event -> finish.setCursor(Cursor.HAND)
            );
        finish.setOnMouseExited(event -> finish.setCursor(Cursor.DEFAULT
            ));
    }

    public void onFirst(ActionEvent actionEvent) throws IOException {
        String user = "";
        String pass = "";

        File loginFile = new File(selectCardController.driveLetter + "
            login.txt");
        if (loginFile.exists()){
```

```
                BufferedReader br = null;
                try {
                    br = new BufferedReader(new FileReader(
                        selectCardController.driveLetter + "login.txt"));
                    user = br.readLine();
                    pass = br.readLine();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    try {
                        br.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                feedbackLabel.setText("");
                changeScene changeScene = new changeScene();
                changeScene.setScene("Setup1.fxml", "style.css", actionEvent
                    , "Fall Analyzer | Login Credentials");

            }
        }

        public void onSecond(ActionEvent actionEvent) throws IOException {
            String user = "";
            String pass = "";

            File loginFile = new File(selectCardController.driveLetter + "
                login.txt");
            if (loginFile.exists()){
                BufferedReader br = null;
                try {
                    br = new BufferedReader(new FileReader(
                        selectCardController.driveLetter + "login.txt"));
                    user = br.readLine();
                    pass = br.readLine();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    try {
                        br.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                feedbackLabel.setText("");
                changeScene changeScene = new changeScene();
                changeScene.setScene("Setup2.fxml", "style.css", actionEvent
                    , "Fall Analyzer | Profile Setup");

            }
```

De La Salle University

```
    }

    public void onThird(ActionEvent actionEvent) throws IOException {
        String user = "";
        String pass = "";

        File loginFile = new File(selectCardController.driveLetter + "
            login.txt");
        if (loginFile.exists()){
            BufferedReader br = null;
            try {
                br = new BufferedReader(new FileReader(
                    selectCardController.driveLetter + "login.txt"));
                user = br.readLine();
                pass = br.readLine();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            feedbackLabel.setText("");
            changeScene changeScene = new changeScene();
            changeScene.setScene("Setup3.fxml", "style.css", actionEvent
                , "Fall Analyzer | Responders Setup");

        }
    }

    public void onFourth(ActionEvent actionEvent) throws IOException {
    }

    public void onFinish(ActionEvent actionEvent) throws IOException {
        String user = "";
        String pass = "";

        File loginFile = new File(selectCardController.driveLetter + "
            login.txt");
        if (loginFile.exists()){
            BufferedReader br = null;
            try {
                br = new BufferedReader(new FileReader(
                    selectCardController.driveLetter + "login.txt"));
                user = br.readLine();
                pass = br.readLine();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
```

```
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        if (username.getText().isEmpty() || password.getText().
            isEmpty()){
            feedbackLabel.setText("All fields are required!");
        }else if(!username.getText().equals(user)){
            feedbackLabel.setText("Invalid username!");
        }else if(!password.getText().equals(pass)){
            feedbackLabel.setText("Invalid password!");
        }else{
            feedbackLabel.setText("");
            changeScene changeScene = new changeScene();
            changeScene.setScene("SetupFinish.fxml", "style.css",
                actionEvent, "Fall Analyzer | Remove card");
        }
    }
  }
}
```

SetupFinish.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="
    @style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="dlsu.setUpFinishController">
    <children>
        <Label id="header1Label" layoutX="379.0" layoutY="288.0"
            stylesheets="@style.css" text="Congratulations!"
            textAlignment="CENTER" />
      <TilePane alignment="CENTER" layoutX="360.0" layoutY="347.0"
          prefHeight="67.0" prefWidth="183.0" vgap="5.0">
          <children>
                <Label id="header2Label" stylesheets="@style.css" text="
                    You have successfully setup your card!" textAlignment
                    ="CENTER" />
                <Label id="header2Label" stylesheets="@style.css" text="
                    Please safely remove the card," textAlignment="CENTER"
                     />
              <Label id="header2Label" layoutX="114.0" layoutY="71.0"
                  stylesheets="@style.css" text="and insert it to your
                  device." textAlignment="CENTER" />
          </children>
      </TilePane>
        <Button id="start" fx:id="onFinish" layoutX="468.0" layoutY
            ="437.0" mnemonicParsing="false" onAction="#onFinish"
            prefHeight="38.0" prefWidth="88.0" stylesheets="@style.css"
            text="finish" />
    </children>
```

```
</Pane>
```

setUpFinishController.java

```
package dlsu;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Cursor;
import javafx.scene.control.Button;
import javafx.stage.Stage;

import java.io.*;
import java.net.URL;
import java.util.ResourceBundle;

/**
 * Created by aenon on 11/11/2017.
 */
public class setUpFinishController implements Initializable {

    @FXML
    private Button onFinish;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        onFinish.setOnMouseEntered(event -> onFinish.setCursor(Cursor.
            HAND));
        onFinish.setOnMousePressed(event -> onFinish.setCursor(Cursor.
            CLOSED_HAND));
        onFinish.setOnMouseReleased(event -> onFinish.setCursor(Cursor.
            HAND));
        onFinish.setOnMouseExited(event -> onFinish.setCursor(Cursor.
            DEFAULT));
    }

    @FXML
    private void onFinish(ActionEvent actionEvent) throws IOException {
        BufferedWriter writer = null;
        try {
            writer = new BufferedWriter(new OutputStreamWriter(new
                FileOutputStream(selectCardController.driveLetter + "
                filelog.txt"), "utf-8"));
            writer.write("actFileCount = 0");
            writer.newLine();
            writer.write("actCounter = 0");
            writer.newLine();
            writer.close();
        } catch (Exception e){
        }

        File dir = new File(selectCardController.driveLetter + "
            Activities");
        dir.mkdir();

```

```
        Stage stage = (Stage) onFinish.getScene().getWindow();
        stage.close();
    }

}
```

SetupHome.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane id="setupPane" prefHeight="575.5" prefWidth="1024" stylesheets="
    @style.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://
    javafx.com/fxml/1" fx:controller="dlsu.setUpHomeController">
    <children>
        <Label id="header1Label" layoutX="467.0" layoutY="288.0"
            stylesheets="@style.css" text="Hello!" />
        <Label id="header2Label" layoutX="382.0" layoutY="337.0"
            stylesheets="@style.css" text="Thank you for using fall
            analyzer! " textAlignment="JUSTIFY" />
        <Label id="header2Label" layoutX="383.0" layoutY="357.0"
            stylesheets="@style.css" text="To begin setup, please click
            start." textAlignment="JUSTIFY" />
        <Button id="start" fx:id="onNext" layoutX="473.0" layoutY
            ="411.0" mnemonicParsing="false" onAction="#onNext"
            prefHeight="38.0" prefWidth="81.0" stylesheets="@style.css"
            text="start" />
    </children>
</Pane>
```

setUpHomeController.java

```
package dlsu;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Cursor;
import javafx.scene.control.Button;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

/**
 * Created by aenon on 11/11/2017.
 */
public class setUpHomeController implements Initializable {

    @FXML
    private Button onNext;

    @Override
```

De La Salle University

```java
    public void initialize(URL location, ResourceBundle resources) {
        onNext.setOnMouseEntered(event -> onNext.setCursor(Cursor.HAND))
            ;
        onNext.setOnMousePressed(event -> onNext.setCursor(Cursor.
            CLOSED_HAND));
        onNext.setOnMouseReleased(event -> onNext.setCursor(Cursor.HAND)
            );
        onNext.setOnMouseExited(event -> onNext.setCursor(Cursor.DEFAULT
            ));
    }

    @FXML
    private void onNext(ActionEvent actionEvent) throws IOException {
        changeScene changeScene = new changeScene();
        changeScene.setScene("Setup1.fxml", "style.css", actionEvent, "
            Fall Analyzer | Login Credentials");
    }

}
```

checkContactNumber.java

```java
package dlsu.Utils;

/**
 * Created by aenon on 25/11/2017.
 */
public class checkContactNumber {
    public static Boolean validContactNumber(String contactNumber){
        if (contactNumber.length() > 13){
            return false;
        } else{
            String num = contactNumber.substring(1);

            try{
                Long.parseLong(num);
                if (num.length() != 12){
                    return false;
                } else {
                    return true;
                }
            } catch (NumberFormatException e){
                return false;
            }
        }
    }
}
```

getExternalDevices.java

```java
package dlsu.Utils;

import javax.swing.filechooser.FileSystemView;
import java.io.File;
import java.util.ArrayList;
```

```
import java.util.List;

/**
 * Created by aenon on 09/11/2017.
 */
public class getExternalDevices {

    public final List<String> storages = new ArrayList<String>();
    public final List<String> storageLetters = new ArrayList<String>();

    public void getDevices(){

        FileSystemView fileSystemView = FileSystemView.getFileSystemView
            ();

        File[] f = File.listRoots();
        for (int i = 0; i < f.length; i++)
        {
            if (!fileSystemView.getSystemDisplayName(f[i]).equals("")){
                storages.add(fileSystemView.getSystemDisplayName(f[i]));
                    //Disk name and Disk Letter
                storageLetters.add(String.valueOf(f[i])); //Disk Letter
            }
        }
    }
}
```

logInDialog.java

```
package dlsu.Utils;

import dlsu.selectCardController;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.util.Pair;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Optional;

/**
 * Created by aenon on 14/11/2017.
 */
public class logInDialog {
    public Boolean logIn(){
        String loginusername = "";
        String loginpassword = "";
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader(selectCardController.
                driveLetter + "login.txt"));
            loginusername = br.readLine();
```

```
            loginpassword = br.readLine();
        } catch (IOException e) {
        } finally {
            try {
                br.close();
            } catch (IOException e) {
            }
        }

        // Create the custom dialog.
        Dialog<Pair<String, String>> dialog = new Dialog<>();
        dialog.setTitle("Login to continue!");
        dialog.setHeaderText("Please enter username and password to
            continue.");

//// Set the icon (must be included in the project).
//            dialog.setGraphic(new ImageView(this.getClass().
    getResource("Main.png").toString()));

// Set the button types.
        ButtonType loginButtonType = new ButtonType("Login", ButtonBar.
            ButtonData.OK_DONE);
        dialog.getDialogPane().getButtonTypes().addAll(loginButtonType,
            ButtonType.CANCEL);

// Create the username and password labels and fields.
        GridPane grid = new GridPane();
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(20, 150, 10, 10));

        TextField username = new TextField();
        username.setPromptText("Username");
        PasswordField password = new PasswordField();
        password.setPromptText("Password");

        grid.add(new Label("Username:"), 0, 0);
        grid.add(username, 1, 0);
        grid.add(new Label("Password:"), 0, 1);
        grid.add(password, 1, 1);

// Enable/Disable login button depending on whether a username was
    entered.
        Node loginButton = dialog.getDialogPane().lookupButton(
            loginButtonType);
        loginButton.setDisable(true);

// Do some validation (using the Java 8 lambda syntax).
        username.textProperty().addListener((observable, oldValue,
            newValue) -> {
            loginButton.setDisable(newValue.trim().isEmpty());
        });

        dialog.getDialogPane().setContent(grid);

// Request focus on the username field by default.
        Platform.runLater(() -> username.requestFocus());
```

```

// Convert the result to a username-password-pair when the login button
    is clicked.
        dialog.setResultConverter(dialogButton -> {
            if (dialogButton == loginButtonType) {
                return new Pair<>(username.getText(), password.getText()
                    );
            }
            return null;
        });

        Optional<Pair<String, String>> result = dialog.showAndWait();

        final String[] user = {""};
        final String[] pass = {""};
        result.ifPresent(usernamePassword -> {
            user[0] = usernamePassword.getKey();
            pass[0] = usernamePassword.getValue();
            //System.out.println("Username=" + usernamePassword.getKey()
                + ", Password=" + usernamePassword.getValue());
        });

        if (user[0].equals(loginusername) && pass[0].equals(
            loginpassword)){
            return true;
        } else{
            return false;
        }
    }
}
```

Appendix G
DATASHEETS

G1 MPU6050 Datasheet

	InvenSense Inc. 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A. Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104 Website: www.invensense.com	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

MPU-6000 and MPU-6050 Product Specification Revision 3.1

De La Salle University

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

CONTENTS

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

1 Revision History

Revision Date	Revision	Description
11/24/2010	1.0	Initial Release
05/19/2011	2.0	For Rev C parts. Clarified wording in sections (3.2, 5.1, 5.2, 6.1-6.4, 6.6, 6.9, 7, 7.1-7.6, 7.11, 7.12, 7.14, 8, 8.2-8.4, 10.3, 10.4, 11, 12.2)
07/28/2011	2.1	Edited supply current numbers for different modes (section 6.4)
08/05/2011	2.2	Unit of measure for accelerometer sensitivity changed from LSB/*mg* to LSB/*g*
10/12/2011	2.3	Updated accelerometer self test specifications in Table 6.2. Updated package dimensions (section 11.2). Updated PCB design guidelines (section 11.3)
10/18/2011	3.0	For Rev D parts. Updated accelerometer specifications in Table 6.2. Updated accelerometer specification note (sections 8.2, 8.3, & 8.4). Updated qualification test plan (section 12.2).
10/24/2011	3.1	Edits for clarity Changed operating voltage range to 2.375V-3.46V Added accelerometer Intelligence Function increment value of 1m*g*/LSB (Section 6.2) Updated absolute maximum rating for acceleration (any axis, unpowered) from 0.3ms to 0.2ms (Section 6.9) Modified absolute maximum rating for Latch-up to Level A and ±100mA (Section 6.9, 12.2)

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

2 Purpose and Scope

This product specification provides advanced information regarding the electrical specification and design related information for the MPU-6000™ and MPU-6050™ Motion Processing Unit™, collectively called the MPU-60X0™ or MPU™.

Electrical characteristics are based upon design analysis and simulation results only. Specifications are subject to change without notice. Final specifications will be updated based upon characterization of production silicon. For references to register map and descriptions of individual registers, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

3 Product Overview

3.1 MPU-60X0 Overview

The MPU-60X0 Motion Processing Unit is the world's first motion processing solution with integrated 9-Axis sensor fusion using its field-proven and proprietary MotionFusion™ engine for handset and tablet applications, game controllers, motion pointer remote controls, and other consumer devices. The MPU-60X0 has an embedded 3-axis MEMS gyroscope, a 3-axis MEMS accelerometer, and a Digital Motion Processor™ (DMP™) hardware accelerator engine with an auxiliary I²C port that interfaces to 3rd party digital sensors such as magnetometers. When connected to a 3-axis magnetometer, the MPU-60X0 delivers a complete 9-axis MotionFusion output to its primary I²C or SPI port (SPI is available on MPU-6000 only). The MPU-60X0 combines acceleration and rotational motion plus heading information into a single data stream for the application. This MotionProcessing™ technology integration provides a smaller footprint and has inherent cost advantages compared to discrete gyroscope plus accelerometer solutions. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I²C port. The MPU-60X0 is a 2nd generation motion processor and is footprint compatible with the MPU-30X0 family.

The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2*g*, ±4*g*, ±8*g*, and ±16*g*.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-60X0 uniquely supports a variety of advanced motion-based applications entirely on-chip. The MPU-60X0 thus enables low-power MotionProcessing in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-60X0 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using either I²C at 400kHz or SPI at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-60X0 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000*g* shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, and the on-chip temperature sensor.

For power supply flexibility, the MPU-60X0 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I²C interface. The VLOGIC voltage may be 1.8V±5% or VDD.

The MPU-6000 and MPU-6050 are identical, except that the MPU-6050 supports the I²C serial interface only, and has a separate VLOGIC reference pin. The MPU-6000 supports both I²C and SPI interfaces and has a single supply pin, VDD, which is both the device's logic reference supply and the analog supply for the part. The table below outlines these differences:

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

Primary Differences between MPU-6000 and MPU-6050

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I^2C, SPI	I^2C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

4 Applications

- *BlurFree*™ technology (for Video/Still Image Stabilization)
- *AirSign*™ technology (for Security/Authentication)
- *TouchAnywhere*™ technology (for “no touch” UI Application Control/Navigation)
- *MotionCommand*™ technology (for Gesture Short-cuts)
- Motion-enabled game and application framework
- InstantGesture™ iG™ gesture recognition
- Location based services, points of interest, and dead reckoning
- Handset and portable gaming
- Motion-based game controllers
- 3D remote controls for Internet connected DTVs and set top boxes, 3D mice
- Wearable sensors for health, fitness and sports
- Toys

5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5µA
- Factory calibrated sensitivity scale factor

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- Free-fall interrupt
- High-G interrupt
- Zero Motion/Motion interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I^2C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant
- 400kHz Fast Mode I^2C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)

- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)
- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

5.4 MotionProcessing

- Internal Digital Motion Processing™ (DMP™) engine supports 3D MotionProcessing and gesture recognition algorithms
- The MPU-60X0 collects gyroscope and accelerometer data while synchronizing data sampling at a user defined rate. The total dataset obtained by the MPU-60X0 includes 3-Axis gyroscope data, 3-Axis accelerometer data, and temperature data. The MPU's calculated output to the system processor can also include heading data from a digital 3-axis third party magnetometer.
- The FIFO buffers the complete data set, reducing timing requirements on the system processor by allowing the processor burst read the FIFO data. After burst reading the FIFO data, the system processor can save power by entering a low-power sleep mode while the MPU collects more data.
- Programmable interrupt supports features such as gesture recognition, panning, zooming, scrolling, zero-motion detection, tap detection, and shake detection
- Digitally-programmable low-pass filters
- Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count.

5.5 Clocking

- On-chip timing generator ±1% frequency variation over full temperature range
- Optional external clock inputs of 32.768kHz or 19.2MHz

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		º/s	
	FS_SEL=1		±500		º/s	
	FS_SEL=2		±1000		º/s	
	FS_SEL=3		±2000		º/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(º/s)	
	FS_SEL=1		65.5		LSB/(º/s)	
	FS_SEL=2		32.8		LSB/(º/s)	
	FS_SEL=3		16.4		LSB/(º/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		º/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		º/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		º/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		º/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		º/s	
Linear Acceleration Sensitivity	Static		0.1		º/s/g	
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		º/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		º/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		º/s/√Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE	Programmable Range	5		256	Hz	
OUTPUT DATA RATE	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling	to ±1º/s of Final		30		ms	

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance[1]	X and Y axes		±50		m*g*	
	Z axis		±80		m*g*	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		m*g*	
SELF TEST RESPONSE			0.5		*g*	
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		μ*g*/√Hz	
LOW PASS FILTER RESPONSE	Programmable Range	5		260	Hz	
OUTPUT DATA RATE	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			1		m*g*/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning

6.3 Electrical and Other Common Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
TEMPERATURE SENSOR						
Range			-40 to +85		°C	
Sensitivity	Untrimmed		340		LSB/ºC	
Temperature Offset	35°C		-521		LSB	
Linearity	Best fit straight line (-40°C to +85°C)		±1		°C	
VDD POWER SUPPLY						
Operating Voltages		2.375		3.46	V	
Normal Operating Current	Gyroscope + Accelerometer + DMP		3.9		mA	
	Gyroscope + Accelerometer (DMP disabled)		3.8		mA	
	Gyroscope + DMP (Accelerometer disabled)		3.7		mA	
	Gyroscope only (DMP & Accelerometer disabled)		3.6		mA	
	Accelerometer only (DMP & Gyroscope disabled)		500		µA	
Accelerometer Low Power Mode Current	1.25 Hz update rate		10		µA	
	5 Hz update rate		20		µA	
	20 Hz update rate		60		µA	
	40 Hz update rate		110		µA	
Full-Chip Idle Mode Supply Current			5		µA	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			100	ms	
VLOGIC REFERENCE VOLTAGE	MPU-6050 only					
Voltage Range	VLOGIC must be ≤VDD at all times	1.71		VDD	V	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			3	ms	
Normal Operating Current			100		µA	
START-UP TIME FOR REGISTER READ/WRITE			20	100	ms	
TEMPERATURE RANGE						
Specified Temperature Range	Performance parameters are not applicable beyond Specified Temperature Range	-40		+85	°C	

De La Salle University

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

6.4 Electrical Specifications, Continued

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE						
SPI Operating Frequency, All Registers Read/Write	MPU-6000 only, Low Speed Characterization		100 ±10%		kHz	
	MPU-6000 only, High Speed Characterization		1 ±10%		MHz	
SPI Operating Frequency, Sensor and Interrupt Registers Read Only	MPU-6000 only		20 ±10%		MHz	
I^2C Operating Frequency	All registers, Fast-mode			400	kHz	
	All registers, Standard-mode			100	kHz	
I^2C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			
DIGITAL INPUTS (SDI/SDA, AD0, SCLK/SCL, FSYNC, /CS, CLKIN)						
V_{IH}, High Level Input Voltage	MPU-6000	0.7*VDD			V	
	MPU-6050	0.7*VLOGIC			V	
V_{IL}, Low Level Input Voltage	MPU-6000			0.3*VDD	V	
	MPU-6050			0.3*VLOGIC	V	
C_I, Input Capacitance			< 5		pF	
DIGITAL OUTPUT (SDO, INT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000	0.9*VDD			V	
	R_{LOAD}=1MΩ; MPU-6050	0.9*VLOGIC			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000			0.1*VDD	V	
	R_{LOAD}=1MΩ; MPU-6050			0.1*VLOGIC	V	
$V_{OL.INT1}$, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink Current			0.1	V	
Output Leakage Current	OPEN=1		100		nA	
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	
DIGITAL OUTPUT (CLKOUT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ	0.9*VDD			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ			0.1*VDD	V	

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

6.5 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, TA = 25°C

Parameters	Conditions	Typical	Units	Notes
Primary I^2C I/O (SCL, SDA)				
V_{IL}, LOW-Level Input Voltage	MPU-6000	-0.5 to 0.3*VDD	V	
V_{IH}, HIGH-Level Input Voltage	MPU-6000	0.7*VDD to VDD + 0.5V	V	
V_{hys}, Hysteresis	MPU-6000	0.1*VDD	V	
VIL, LOW Level Input Voltage	MPU-6050	-0.5V to 0.3*VLOGIC	V	
VIH, HIGH-Level Input Voltage	MPU-6050	0.7*VLOGIC to VLOGIC + 0.5V	V	
Vhys, Hysteresis	MPU-6050	0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	3mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	3	mA	
	V_{OL} = 0.6V	5	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	
Auxiliary I^2C I/O (AUX_CL, AUX_DA)	**MPU-6050: *AUX_VDDIO*=0**			
V_{IL}, LOW-Level Input Voltage		-0.5V to 0.3*VLOGIC	V	
V_{IH}, HIGH-Level Input Voltage		0.7*VLOGIC to VLOGIC + 0.5V	V	
V_{hys}, Hysteresis		0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	VLOGIC > 2V; 1mA sink current	0 to 0.4	V	
V_{OL3}, LOW-Level Output Voltage	VLOGIC < 2V; 1mA sink current	0 to 0.2*VLOGIC	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	1	mA	
	V_{OL} = 0.6V	1	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	
Auxiliary I^2C I/O (AUX_CL, AUX_DA)	**MPU-6050: *AUX_VDDIO*=1; MPU-6000**			
V_{IL}, LOW-Level Input Voltage		-0.5 to 0.3*VDD	V	
V_{IH}, HIGH-Level Input Voltage		0.7*VDD to VDD+0.5V	V	
V_{hys}, Hysteresis		0.1*VDD	V	
V_{OL1}, LOW-Level Output Voltage	1mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	1	mA	
	V_{OL} = 0.6V	1	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus cap. in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	

6.6 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, TA = 25°C

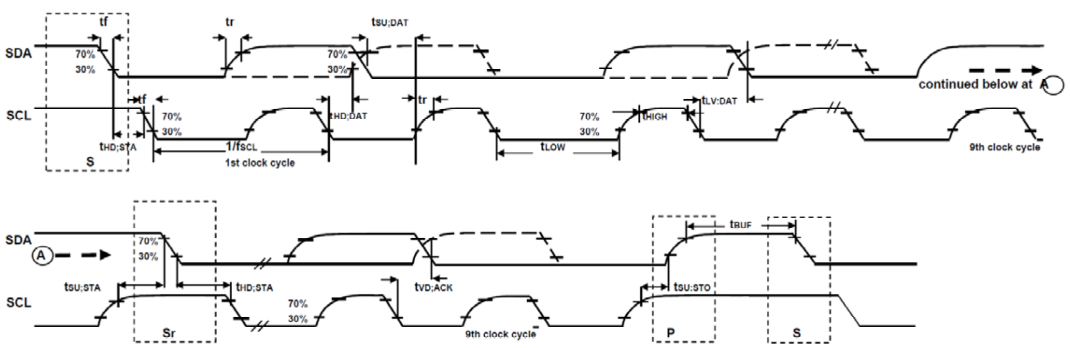
Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLK_SEL=0,1,2,3**					
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Reference Clock Output	CLKOUTEN = 1		1.024		MHz	
Clock Frequency Initial Tolerance	CLK_SEL=0, 25°C	-5		+5	%	
	CLK_SEL=1,2,3; 25°C	-1		+1	%	
Frequency Variation over Temperature	CLK_SEL=0		-15 to +10		%	
	CLK_SEL=1,2,3		±1		%	
PLL Settling Time	CLK_SEL=1,2,3		1	10	ms	
EXTERNAL 32.768kHz CLOCK	**CLK_SEL=4**					
External Clock Frequency			32.768		kHz	
External Clock Allowable Jitter	Cycle-to-cycle rms		1 to 2		μs	
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8.192		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1.024		kHz	
Accelerometer Sample Rate			1.024		kHz	
Reference Clock Output	CLKOUTEN = 1		1.0486		MHz	
PLL Settling Time			1	10	ms	
EXTERNAL 19.2MHz CLOCK	**CLK_SEL=5**					
External Clock Frequency			19.2		MHz	
Gyroscope Sample Rate	Full programmable range	3.9		8000	Hz	
Gyroscope Sample Rate, Fast Mode	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow Mode	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Reference Clock Output	CLKOUTEN = 1		1.024		MHz	
PLL Settling Time			1	10	ms	

6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I²C Bus



I²C Bus Timing Diagram

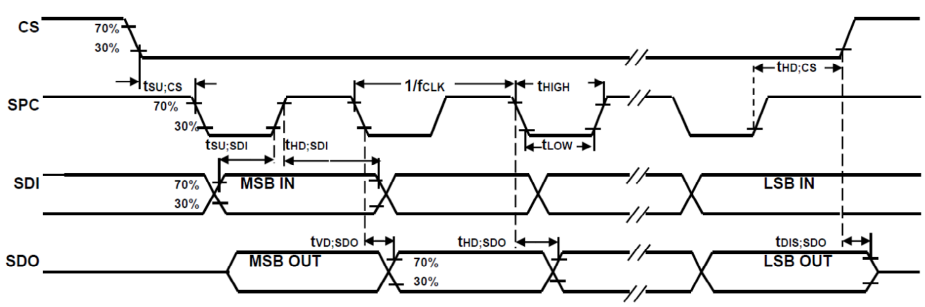
De La Salle University

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

6.8 SPI Timing Characterization (MPU-6000 only)

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD,T_A = -40°C to +85°C, unless otherwise noted.

Parameters	Conditions	Min	Typical	Max	Units	Notes
SPI TIMING						
f_{SCLK}, SCLK Clock Frequency				1	MHz	
t_{LOW}, SCLK Low Period		400			ns	
t_{HIGH}, SCLK High Period		400			ns	
$t_{SU.CS}$, CS Setup Time		8			ns	
$t_{HD.CS}$, CS Hold Time		500			ns	
$t_{SU.SDI}$, SDI Setup Time		11			ns	
$t_{HD.SDI}$, SDI Hold Time		7			ns	
$t_{VD.SDO}$, SDO Valid Time	C_{load} = 20pF			100	ns	
$t_{HD.SDO}$, SDO Hold Time	C_{load} = 20pF	4			ns	
$t_{DIS.SDO}$, SDO Output Disable Time				10	ns	



SPI Bus Timing Diagram

6.9 Absolute Maximum Ratings

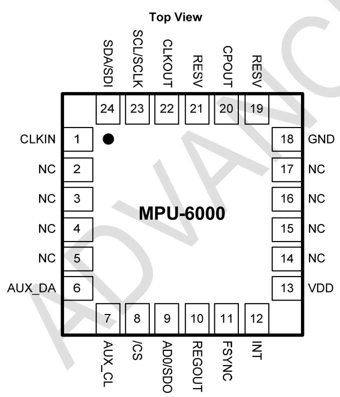
Stress above those listed as “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level (MPU-6050)	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA)	-0.5V to VDD + 0.5V
CPOUT (2.5V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000g for 0.2ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	2kV (HBM); 200V (MM)
Latch-up	JEDEC Class II (2),125°C Level A, ±100mA

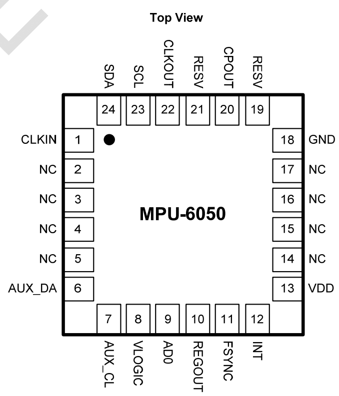
7 Applications Information

7.1 Pin Out and Signal Description

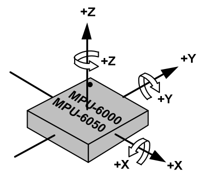
Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I^2C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I^2C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (0=SPI mode)
8		Y	VLOGIC	Digital I/O supply voltage
9	Y		AD0 / SDO	I^2C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I^2C Slave Address LSB (AD0)
10	Y	Y	REGOUT	Regulator filter capacitor connection
11	Y	Y	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	Y	Y	INT	Interrupt digital output (totem pole or open-drain)
13	Y	Y	VDD	Power supply voltage and Digital I/O supply voltage
18	Y	Y	GND	Power supply ground
19, 21	Y	Y	RESV	Reserved. Do not connect.
20	Y	Y	CPOUT	Charge pump capacitor connection
22	Y	Y	CLKOUT	System clock output
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock (SCL)
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data (SDA)
2, 3, 4, 5, 14, 15, 16, 17	Y	Y	NC	Not internally connected. May be used for PCB trace routing.



QFN Package
24-pin, 4mm x 4mm x 0.9mm



QFN Package
24-pin, 4mm x 4mm x 0.9mm

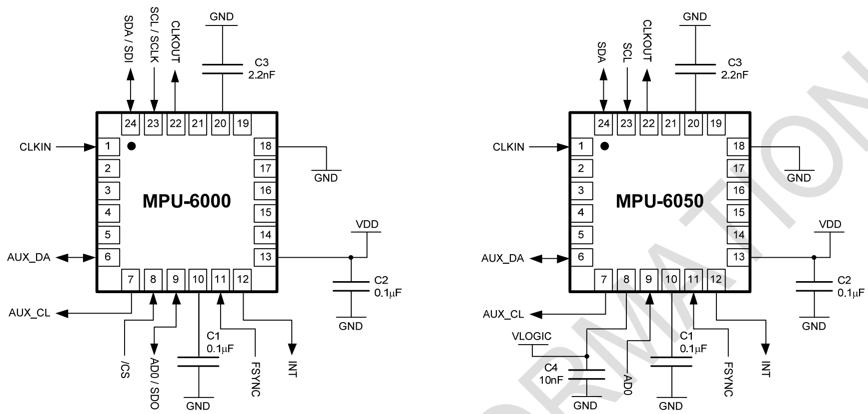


Orientation of Axes of Sensitivity and Polarity of Rotation

De La Salle University

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

7.2 Typical Operating Circuit



Typical Operating Circuits

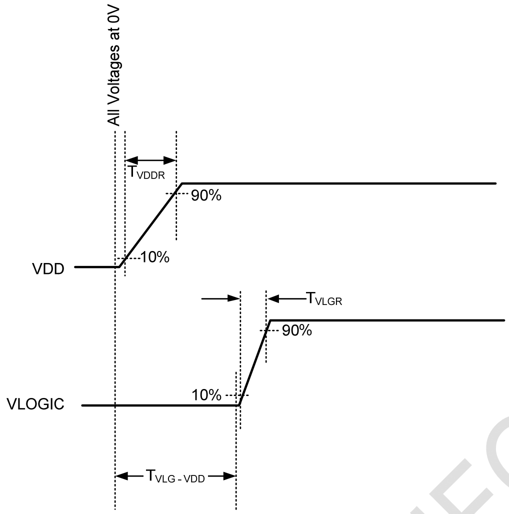
7.3 Bill of Materials for External Components

Component	Label	Specification	Quantity
Regulator Filter Capacitor (Pin 10)	C1	Ceramic, X7R, 0.1µF ±10%, 2V	1
VDD Bypass Capacitor (Pin 13)	C2	Ceramic, X7R, 0.1µF ±10%, 4V	1
Charge Pump Capacitor (Pin 20)	C3	Ceramic, X7R, 2.2nF ±10%, 50V	1
VLOGIC Bypass Capacitor (Pin 8)	C4*	Ceramic, X7R, 10nF ±10%, 4V	1

* MPU-6050 Only.

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

7.4 Recommended Power-on Procedure

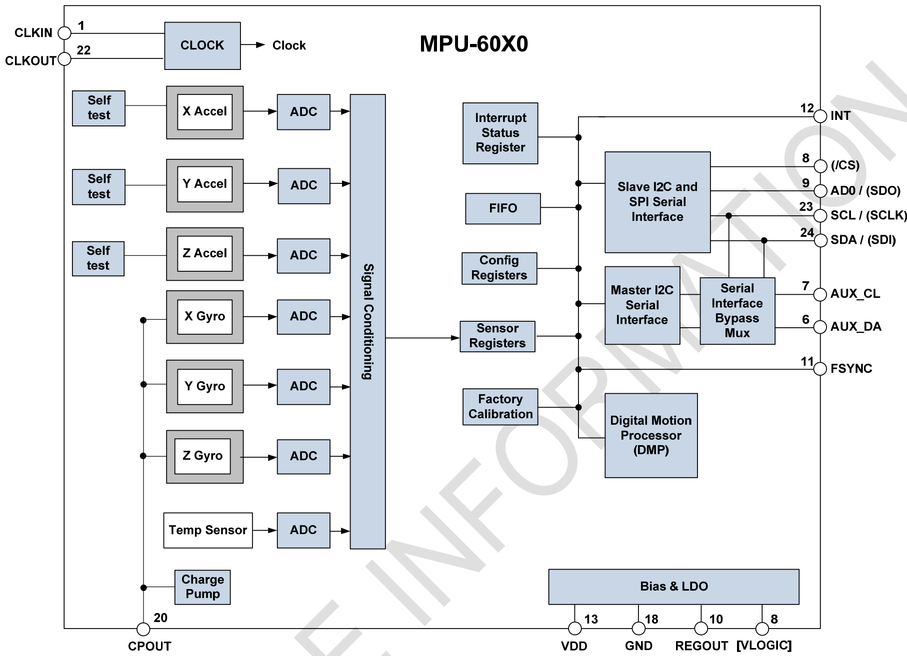


Power-Up Sequencing

1. VLOGIC amplitude must always be ≤VDD amplitude
2. T_{VDDR} is VDD rise time: Time for VDD to rise from 10% to 90% of its final value
3. T_{VDDR} is ≤100ms
4. T_{VLGR} is VLOGIC rise time: Time for VLOGIC to rise from 10% to 90% of its final value
5. T_{VLGR} is ≤3ms
6. $T_{VLG-VDD}$ is the delay from the start of VDD ramp to the start of VLOGIC rise
7. $T_{VLG-VDD}$ is ≥0
8. VDD and VLOGIC must be monotonic ramps

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

7.5 Block Diagram



Note: Pin names in round brackets ( ) apply only to MPU-6000
Pin names in square brackets [ ] apply only to MPU-6050

7.6 Overview

The MPU-60X0 is comprised of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning
- Digital Motion Processor (DMP) engine
- Primary I^2C and SPI (MPU-6000 only) serial communications interfaces
- Auxiliary I^2C serial interface for 3rd party magnetometer & other sensors
- Clocking
- Sensor Data Registers
- FIFO
- Interrupts
- Digital-Output Temperature Sensor
- Accelerometer Self-test
- Bias and LDO
- Charge Pump

7.7 Three-Axis MEMS Gyroscope with 16-bit ADCs and Signal Conditioning

The MPU-60X0 consists of three independent vibratory MEMS rate gyroscopes, which detect rotation about the X-, Y-, and Z- Axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis. The full-scale range of the gyro sensors may be digitally programmed to ±250, ±500, ±1000, or ±2000 degrees per second (dps). The ADC sample rate is programmable from 8,000 samples per second, down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

7.8 Three-Axis MEMS Accelerometer with 16-bit ADCs and Signal Conditioning

The MPU-60X0's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-60X0's architecture reduces the accelerometers' susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0*g* on the X- and Y-axes and +1*g* on the Z-axis. The accelerometers' scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2*g*, ±4*g*, ±8*g*, or ±16*g*.

7.9 Digital Motion Processor

The embedded Digital Motion Processor (DMP) is located within the MPU-60X0 and offloads computation of motion processing algorithms from the host processor. The DMP acquires data from accelerometers, gyroscopes, and additional 3rd party sensors such as magnetometers, and processes the data. The resulting data can be read from the DMP's registers, or can be buffered in a FIFO. The DMP has access to one of the MPU's external pins, which can be used for generating interrupts.

The purpose of the DMP is to offload both timing requirements and processing power from the host processor. Typically, motion processing algorithms should be run at a high rate, often around 200Hz, in order to provide accurate results with low latency. This is required even if the application updates at a much lower rate; for example, a low power user interface may update as slowly as 5Hz, but the motion processing should still run at 200Hz. The DMP can be used as a tool in order to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in the application.

7.10 Primary I^{2}C and SPI Serial Communications Interfaces

The MPU-60X0 communicates to a system processor using either a SPI (MPU-6000 only) or an I^{2}C serial interface. The MPU-60X0 always acts as a slave when communicating to the system processor. The LSB of the of the I^{2}C slave address is set by pin 9 (AD0).

The logic levels for communications between the MPU-60X0 and its master are as follows:

- MPU-6000: The logic level for communications with the master is set by the voltage on VDD
- MPU-6050: The logic level for communications with the master is set by the voltage on VLOGIC

For further information regarding the logic levels of the MPU-6050, please refer to Section 10.

7.11 Auxiliary I²C Serial Interface

The MPU-60X0 has an auxiliary I²C bus for communicating to an off-chip 3-Axis digital output magnetometer or other sensors. This bus has two operating modes:

- I²C Master Mode: The MPU-60X0 acts as a master to any external sensors connected to the auxiliary I²C bus
- Pass-Through Mode: The MPU-60X0 directly connects the primary and auxiliary I²C buses together, allowing the system processor to directly communicate with any external sensors.

Auxiliary I²C Bus Modes of Operation:

- I²C Master Mode: Allows the MPU-60X0 to directly access the data registers of external digital sensors, such as a magnetometer. In this mode, the MPU-60X0 directly obtains data from auxiliary sensors, allowing the on-chip DMP to generate sensor fusion data without intervention from the system applications processor.

  For example, In I²C Master mode, the MPU-60X0 can be configured to perform burst reads, returning the following data from a magnetometer:

  - X magnetometer data (2 bytes)
  - Y magnetometer data (2 bytes)
  - Z magnetometer data (2 bytes)

  The I²C Master can be configured to read up to 24 bytes from up to 4 auxiliary sensors. A fifth sensor can be configured to work single byte read/write mode.

- Pass-Through Mode: Allows an external system processor to act as master and directly communicate to the external sensors connected to the auxiliary I²C bus pins (AUX_DA and AUX_CL). In this mode, the auxiliary I²C bus control logic (3rd party sensor interface block) of the MPU-60X0 is disabled, and the auxiliary I²C pins AUX_DA and AUX_CL (Pins 6 and 7) are connected to the main I²C bus (Pins 23 and 24) through analog switches.

  Pass-Through Mode is useful for configuring the external sensors, or for keeping the MPU-60X0 in a low-power mode when only the external sensors are used.

  In Pass-Through Mode the system processor can still access MPU-60X0 data through the I²C interface.

Auxiliary I²C Bus IO Logic Levels

- MPU-6000: The logic level of the auxiliary I²C bus is VDD
- MPU-6050: The logic level of the auxiliary I²C bus can be programmed to be either VDD or VLOGIC

For further information regarding the MPU-6050's logic levels, please refer to Section 10.2.

7.12 Self-Test

Self-test allows for the testing of the mechanical and electrical portions of the accelerometers. The self-test for each measurement axis can be activated by controlling the bits of the ACCEL_CONFIG control register. When self-test is activated, the electronics cause the sensors to be actuated and produce an output signal.

The sensor reading is determined by the sum of the sensor output and the self-test response. The self-test response for each accelerometer axis is defined in the specification table (Section 6) to be nominally 0.5*g*.

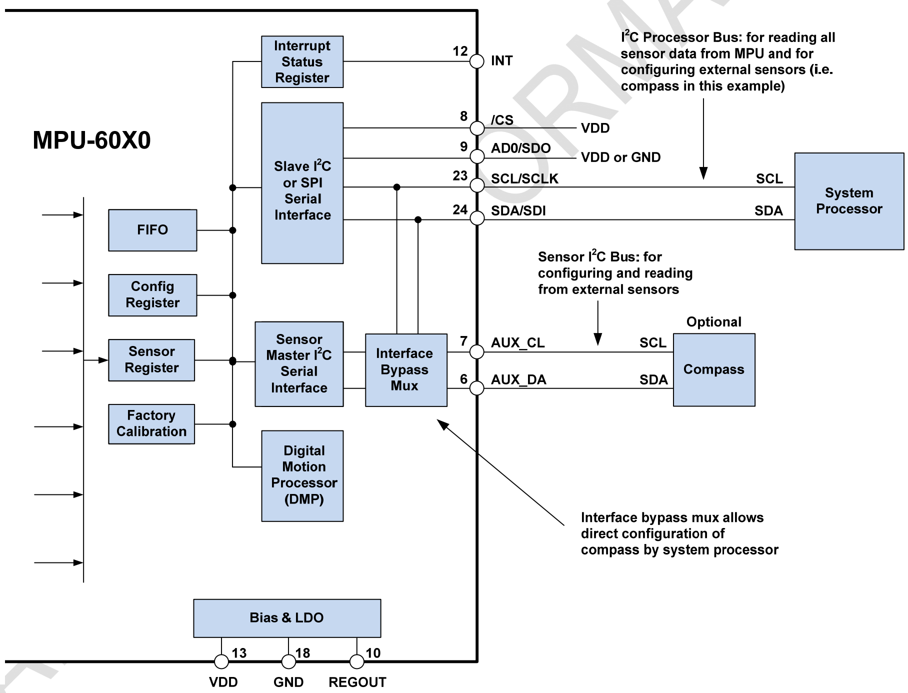
For further information regarding the Accel control register, please refer to the MPU-60X0 Register Map and Register Descriptions document.

7.13 MPU-60X0 Solution for 9-axis Sensor Fusion Using I²C Interface

In the figure below, the system processor is an I²C master to the MPU-60X0. In addition, the MPU-60X0 is an I²C master to the optional external compass sensor. The MPU-60X0 has limited capabilities as an I²C Master, and depends on the system processor to manage the initial configuration of any auxiliary sensors. The MPU-60X0 has an interface bypass multiplexer, which connects the system processor I²C bus pins 23 and 24 (SDA and SCL) directly to the auxiliary sensor I²C bus pins 6 and 7 (AUX_DA and AUX_CL).

Once the auxiliary sensors have been configured by the system processor, the interface bypass multiplexer should be disabled so that the MPU-60X0 auxiliary I²C master can take control of the sensor I²C bus and gather data from the auxiliary sensors.

For further information regarding I²C master control, please refer to Section 10.



7.14 MPU-6000 Using SPI Interface

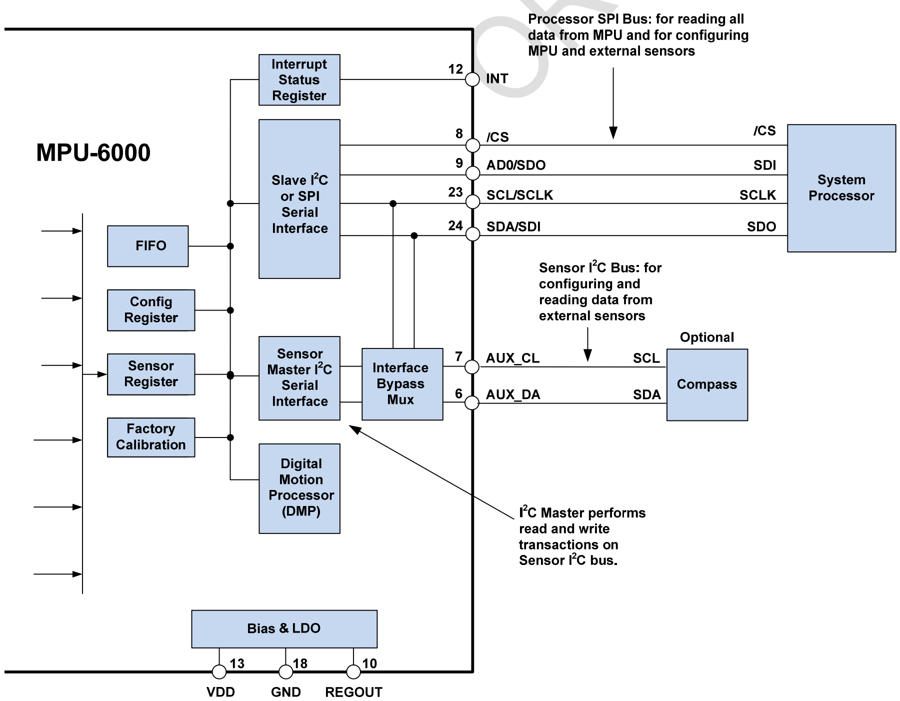
In the figure below, the system processor is an SPI master to the MPU-6000. Pins 8, 9, 23, and 24 are used to support the /CS, SDO, SCLK, and SDI signals for SPI communications. Because these SPI pins are shared with the I^2C slave pins (9, 23 and 24), the system processor cannot access the auxiliary I^2C bus through the interface bypass multiplexer, which connects the processor I^2C interface pins to the sensor I^2C interface pins.

Since the MPU-6000 has limited capabilities as an I^2C Master, and depends on the system processor to manage the initial configuration of any auxiliary sensors, another method must be used for programming the sensors on the auxiliary sensor I^2C bus pins 6 and 7 (AUX_DA and AUX_CL).

When using SPI communications between the MPU-6000 and the system processor, configuration of devices on the auxiliary I^2C sensor bus can be achieved by using I^2C Slaves 0-4 to perform read and write transactions on any device and register on the auxiliary I^2C bus. The I^2C Slave 4 interface can be used to perform only single byte read and write transactions.

Once the external sensors have been configured, the MPU-6000 can perform single or multi-byte reads using the sensor I^2C bus. The read results from the Slave 0-3 controllers can be written to the FIFO buffer as well as to the external sensor registers.

For further information regarding the control of the MPU-60X0's auxiliary I^2C interface, please refer to the MPU-60X0 Register Map and Register Descriptions document.



7.15 Internal Clock Generation

The MPU-60X0 has a flexible clocking scheme, allowing a variety of internal or external clock sources to be used for the internal synchronous circuitry. This synchronous circuitry includes the signal conditioning and ADCs, the DMP, and various control circuits and registers. An on-chip PLL provides flexibility in the allowable inputs for generating this clock.

Allowable internal sources for generating the internal clock are:

- An internal relaxation oscillator
- Any of the X, Y, or Z gyros (MEMS oscillators with a variation of ±1% over temperature)

Allowable external clocking sources are:

- 32.768kHz square wave
- 19.2MHz square wave

Selection of the source for generating the internal synchronous clock depends on the availability of external sources and the requirements for power consumption and clock accuracy. These requirements will most likely vary by mode of operation. For example, in one mode, where the biggest concern is power consumption, the user may wish to operate the Digital Motion Processor of the MPU-60X0 to process accelerometer data, while keeping the gyros off. In this case, the internal relaxation oscillator is a good clock choice. However, in another mode, where the gyros are active, selecting the gyros as the clock source provides for a more accurate clock source.

Clock accuracy is important, since timing errors directly affect the distance and angle calculations performed by the Digital Motion Processor (and by extension, by any processor).

There are also start-up conditions to consider. When the MPU-60X0 first starts up, the device uses its internal clock until programmed to operate from another source. This allows the user, for example, to wait for the MEMS oscillators to stabilize before they are selected as the clock source.

7.16 Sensor Data Registers

The sensor data registers contain the latest gyro, accelerometer, auxiliary sensor, and temperature measurement data. They are read-only registers, and are accessed via the serial interface. Data from these registers may be read anytime. However, the interrupt function may be used to determine when new data is available.

For a table of interrupt sources please refer to Section 8.

7.17 FIFO

The MPU-60X0 contains a 1024-byte FIFO register that is accessible via the Serial Interface. The FIFO configuration register determines which data is written into the FIFO. Possible choices include gyro data, accelerometer data, temperature readings, auxiliary sensor readings, and FSYNC input. A FIFO counter keeps track of how many bytes of valid data are contained in the FIFO. The FIFO register supports burst reads. The interrupt function may be used to determine when new data is available.

For further information regarding the FIFO, please refer to the MPU-60X0 Register Map and Register Descriptions document.

7.18 Interrupts

Interrupt functionality is configured via the Interrupt Configuration register. Items that are configurable include the INT pin configuration, the interrupt latching and clearing method, and triggers for the interrupt. Items that can trigger an interrupt are (1) Clock generator locked to new reference oscillator (used when switching clock sources); (2) new data is available to be read (from the FIFO and Data registers); (3) accelerometer event interrupts; and (4) the MPU-60X0 did not receive an acknowledge from an auxiliary sensor on the secondary I^2C bus. The interrupt status can be read from the Interrupt Status register.

For further information regarding interrupts, please refer to the MPU-60X0 Register Map and Register Descriptions document.

For information regarding the MPU-60X0's accelerometer event interrupts, please refer to Section 8.

7.19 Digital-Output Temperature Sensor

An on-chip temperature sensor and ADC are used to measure the MPU-60X0 die temperature. The readings from the ADC can be read from the FIFO or the Sensor Data registers.

7.20 Bias and LDO

The bias and LDO section generates the internal supply and the reference voltages and currents required by the MPU-60X0. Its two inputs are an unregulated VDD of 2.1V to 3.6V and a VLOGIC logic reference supply voltage of 1.71V to VDD (MPU-6050 only). The LDO output is bypassed by a capacitor at REGOUT. For further details on the capacitor, please refer to the Bill of Materials for External Components (Section 7.3).

7.21 Charge Pump

An on-board charge pump generates the high voltage required for the MEMS oscillators. Its output is bypassed by a capacitor at CPOUT. For further details on the capacitor, please refer to the Bill of Materials for External Components (Section 7.3).

8 Programmable Interrupts

The MPU-60X0 has a programmable interrupt system which can generate an interrupt signal on the INT pin. Status flags indicate the source of an interrupt. Interrupt sources may be enabled and disabled individually.

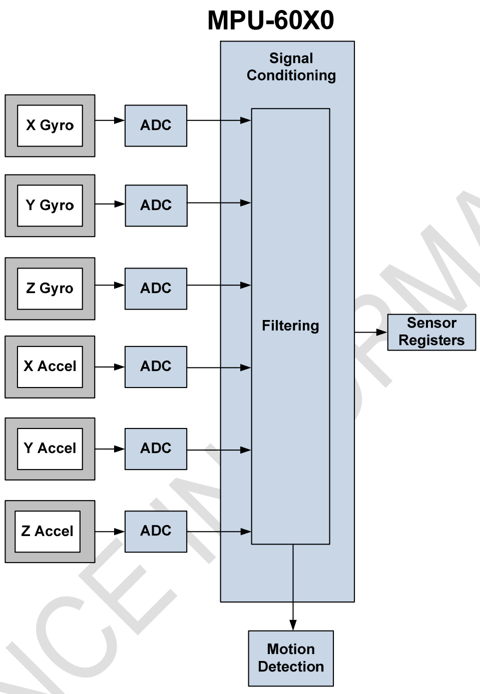
Table of Interrupt Sources

Interrupt Name	Module
Free Fall Detection	Free Fall
Motion Detection	Motion
Zero Motion Detection	Zero Motion
FIFO Overflow	FIFO
Data Ready	Sensor Registers
I^2C Master errors: Lost Arbitration, NACKs	I^2C Master
I^2C Slave 4	I^2C Master

For information regarding the interrupt enable/disable registers and flag registers, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document. Some interrupt sources are explained below.

8.1 Free Fall, Motion, and Zero Motion Signal Paths

The diagram below shows the signal path for the gyroscope and accelerometer sensors. Note that each digital low pass filter (DLPF) is configured identically, as is each sample rate divider and digital high pass filter (DHPF).

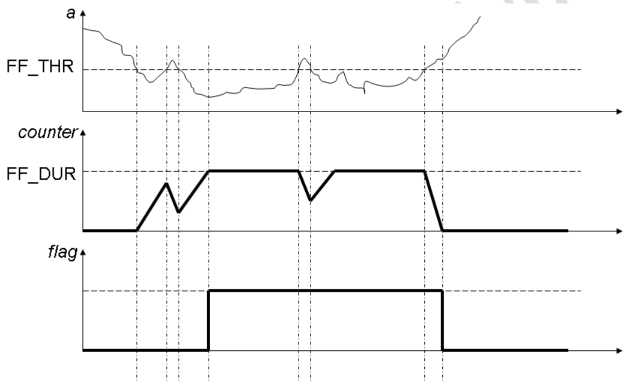


8.2 Free Fall Interrupt

Free fall is detected by checking if the accelerometer measurements from all 3 axes have an absolute value below a user-programmable threshold (acceleration threshold). For each sample where this condition is true (a qualifying sample), a counter is incremented. For each sample where this condition is false (a non-qualifying sample), the counter is decremented. Once the counter reaches a user-programmable threshold (the counter threshold), the Free Fall interrupt is triggered and a flag is set. The flag is cleared once the counter has decremented to zero. The counter does not increment above the counter threshold or decrement below zero.

The user is given several configuration parameters to fine tune Free Fall detection. Both, the acceleration threshold and counter threshold are user configurable. The FF_THR register allows the user to set a threshold in 1 *m*g increments. The FF_DUR register allows the user to set duration in 1 ms increments.

The decrement rate for non-qualifying samples is also configurable. The MOT_DETECT_CTRL register allows the user to specify whether a non-qualifying sample makes the counter reset to zero, or decrement in steps of 1, 2, or 4.



The figure above shows a simplified example with just one axis. An example acceleration input signal (simplified to only show one axis), qualifying sample counter, and Free Fall flag are shown.

8.3 Motion Interrupt

The MPU-60X0 provides Motion detection capability with similar functionality to Free Fall detection. Accelerometer measurements are passed through a configurable digital high pass filter (DHPF) in order to eliminate bias due to gravity. A qualifying motion sample is one where the high passed sample from any axis has an absolute value exceeding a user-programmable threshold. A counter increments for each qualifying sample, and decrements for each non-qualifying sample. Once the counter reaches a user-programmable counter threshold, a motion interrupt is triggered. The axis and polarity which caused the interrupt to be triggered is flagged in the MOT_DETECT_STATUS register.

Like Free Fall detection, Motion detection has a configurable acceleration threshold MOT_THR specified in 1 *m*g increments. The counter threshold MOT_DUR is specified in 1 ms increments. The decrement rate has the same options as Free Fall detection, and is specified in the MOT_DETECT_CTRL register.

8.4 Zero Motion Interrupt

The Zero Motion detection capability uses the digital high pass filter (DHPF) and a similar threshold scheme to that of Free Fall detection. Each axis of the high passed accelerometer measurement must have an absolute value less than a threshold specified in the ZRMOT_THR register, which can be increased in 1 m*g* increments. Each time a motion sample meets this condition, a counter increments. When this counter reaches a threshold specified in ZRMOT_DUR, an interrupt is generated.

Unlike Free Fall or Motion detection, Zero Motion detection triggers an interrupt both when Zero Motion is first detected and when Zero Motion is no longer detected. While Free Fall and Motion are indicated with a flag which clears after being read, reading the state of the Zero Motion detected from the MOT_DETECT_STATUS register does not clear its status.

De La Salle University

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

9 Digital Interface

9.1 I²C and SPI (MPU-6000 only) Serial Interfaces

The internal registers and memory of the MPU-6000/MPU-6050 can be accessed using either I²C at 400 kHz or SPI at 1MHz (MPU-6000 only). SPI operates in four-wire mode.

Serial Interface

Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
8	Y		/CS	SPI chip select (0=SPI enable)
8		Y	VLOGIC	Digital I/O supply voltage. VLOGIC must be ≤ VDD at all times.
9	Y		AD0 / SDO	I²C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I²C Slave Address LSB
23	Y		SCL / SCLK	I²C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I²C serial clock
24	Y		SDA / SDI	I²C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I²C serial data

Note:

To prevent switching into I²C mode when using SPI (MPU-6000), the I²C interface should be disabled by setting the *I2C_IF_DIS* configuration bit. Setting this bit should be performed immediately after waiting for the time specified by the "Start-Up Time for Register Read/Write" in Section 6.3.

For further information regarding the *I2C_IF_DIS* bit, please refer to the MPU-60X0 Register Map and Register Descriptions document.

9.2 I²C Interface

I²C is a two-wire interface comprised of the signals serial data (SDA) and serial clock (SCL). In general, the lines are open-drain and bi-directional. In a generalized I²C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master.

The MPU-60X0 always operates as a slave device when communicating to the system processor, which thus acts as the master. SDA and SCL lines typically need pull-up resistors to VDD. The maximum bus speed is 400 kHz.

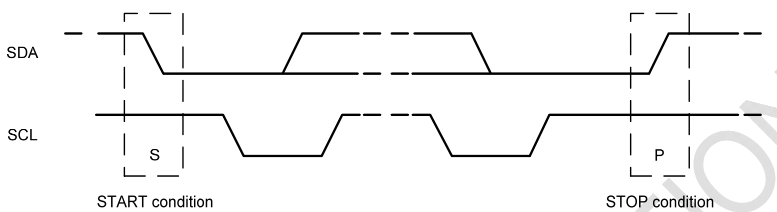
The slave address of the MPU-60X0 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two MPU-60X0s to be connected to the same I²C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin AD0 is logic low) and the address of the other should be b1101001 (pin AD0 is logic high).

9.3 I²C Communications Protocol

START (S) and STOP (P) Conditions

Communication on the I²C bus starts when the master puts the START condition (S) on the bus, which is defined as a HIGH-to-LOW transition of the SDA line while SCL line is HIGH (see figure below). The bus is considered to be busy until the master puts a STOP condition (P) on the bus, which is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH (see figure below).

Additionally, the bus remains busy if a repeated START (Sr) is generated instead of a STOP condition.

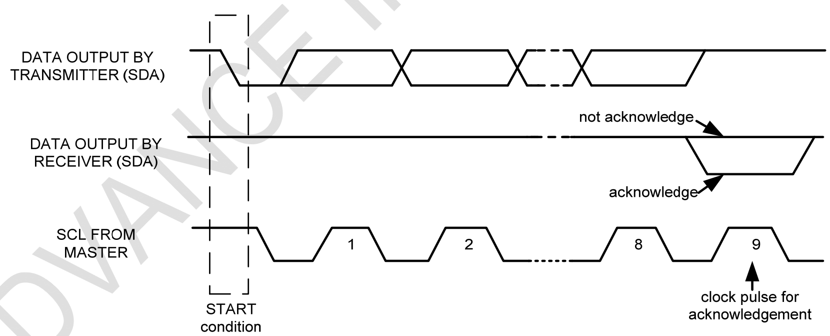


START and STOP Conditions

Data Format / Acknowledge

I²C data bytes are defined to be 8-bits long. There is no restriction to the number of bytes transmitted per data transfer. Each byte transferred must be followed by an acknowledge (ACK) signal. The clock for the acknowledge signal is generated by the master, while the receiver generates the actual acknowledge signal by pulling down SDA and holding it low during the HIGH portion of the acknowledge clock pulse.

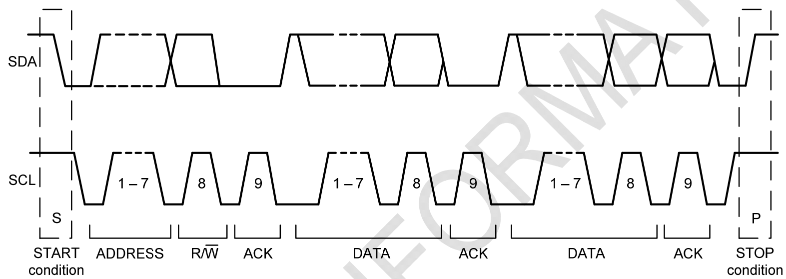
If a slave is busy and cannot transmit or receive another byte of data until some other task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready, and releases the clock line (refer to the following figure).



Acknowledge on the I²C Bus

Communications

After beginning communications with the START condition (S), the master sends a 7-bit slave address followed by an 8th bit, the read/write bit. The read/write bit indicates whether the master is receiving data from or is writing to the slave device. Then, the master releases the SDA line and waits for the acknowledge signal (ACK) from the slave device. Each byte transferred must be followed by an acknowledge bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW for the high period of the SCL line. Data transmission is always terminated by the master with a STOP condition (P), thus freeing the communications line. However, the master can generate a repeated START condition (Sr), and address another slave without first generating a STOP condition (P). A LOW to HIGH transition on the SDA line while SCL is HIGH defines the stop condition. All SDA changes should take place when SCL is low, with the exception of start and stop conditions.



Complete I²C Data Transfer

To write the internal MPU-60X0 registers, the master transmits the start condition (S), followed by the I²C address and the write bit (0). At the 9th clock cycle (when the clock is high), the MPU-60X0 acknowledges the transfer. Then the master puts the register address (RA) on the bus. After the MPU-60X0 acknowledges the reception of the register address, the master puts the register data onto the bus. This is followed by the ACK signal, and data transfer may be concluded by the stop condition (P). To write multiple bytes after the last ACK signal, the master can continue outputting data rather than transmitting a stop signal. In this case, the MPU-60X0 automatically increments the register address and loads the data to the appropriate register. The following figures show single and two-byte write sequences.

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

To read the internal MPU-60X0 registers, the master sends a start condition, followed by the I²C address and a write bit, and then the register address that is going to be read. Upon receiving the ACK signal from the MPU-60X0, the master transmits a start signal followed by the slave address and read bit. As a result, the MPU-60X0 sends an ACK signal and the data. The communication ends with a not acknowledge (NACK) signal and a stop bit from master. The NACK condition is defined such that the SDA line remains high at the 9th clock cycle. The following figures show single and two-byte read sequences.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

9.4 I²C Terms

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

9.5 SPI Interface (MPU-6000 only)

SPI is a 4-wire synchronous serial interface that uses two control lines and two data lines. The MPU-6000 always operates as a Slave device during standard Master-Slave SPI operation.

With respect to the Master, the Serial Clock output (SCLK), the Serial Data Output (SDO) and the Serial Data Input (SDI) are shared among the Slave devices. Each SPI slave device requires its own Chip Select (/CS) line from the master.

/CS goes low (active) at the start of transmission and goes back high (inactive) at the end. Only one /CS line is active at a time, ensuring that only one slave is selected at any given time. The /CS lines of the non-selected slave devices are held high, causing their SDO lines to remain in a high-impedance (high-z) state so that they do not interfere with any active devices.

SPI Operational Features

1. Data is delivered MSB first and LSB last
2. Data is latched on the rising edge of SCLK
3. Data should be transitioned on the falling edge of SCLK
4. The maximum frequency of SCLK is 1MHz
5. SPI read and write operations are completed in 16 or more clock cycles (two or more bytes). The first byte contains the SPI Address, and the following byte(s) contain(s) the SPI data. The first bit of the first byte contains the Read/Write bit and indicates the Read (1) or Write (0) operation. The following 7 bits contain the Register Address. In cases of multiple-byte Read/Writes, data is two or more bytes:

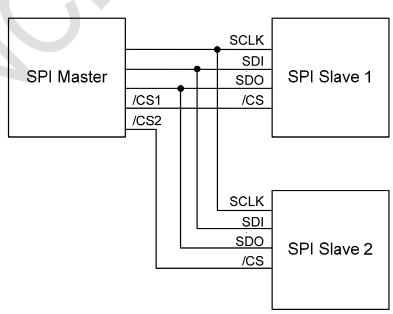
SPI Address format

MSB							**LSB**
R/W	A6	A5	A4	A3	A2	A1	A0

SPI Data format

MSB							**LSB**
D7	D6	D5	D4	D3	D2	D1	D0

6. Supports Single or Burst Read/Writes.



Typical SPI Master / Slave Configuration

10 Serial Interface Considerations (MPU-6050)

10.1 MPU-6050 Supported Interfaces

The MPU-6050 supports I^2C communications on both its primary (microprocessor) serial interface and its auxiliary interface.

10.2 Logic Levels

The MPU-6050's I/O logic levels are set to be either VDD or VLOGIC, as shown in the table below.

I/O Logic Levels vs. *AUX_VDDIO*

AUX_VDDIO	MICROPROCESSOR LOGIC LEVELS (Pins: SDA, SCL, AD0, CLKIN, INT)	AUXILLARY LOGIC LEVELS (Pins: AUX_DA, AUX_CL)
0	VLOGIC	VLOGIC
1	VLOGIC	VDD

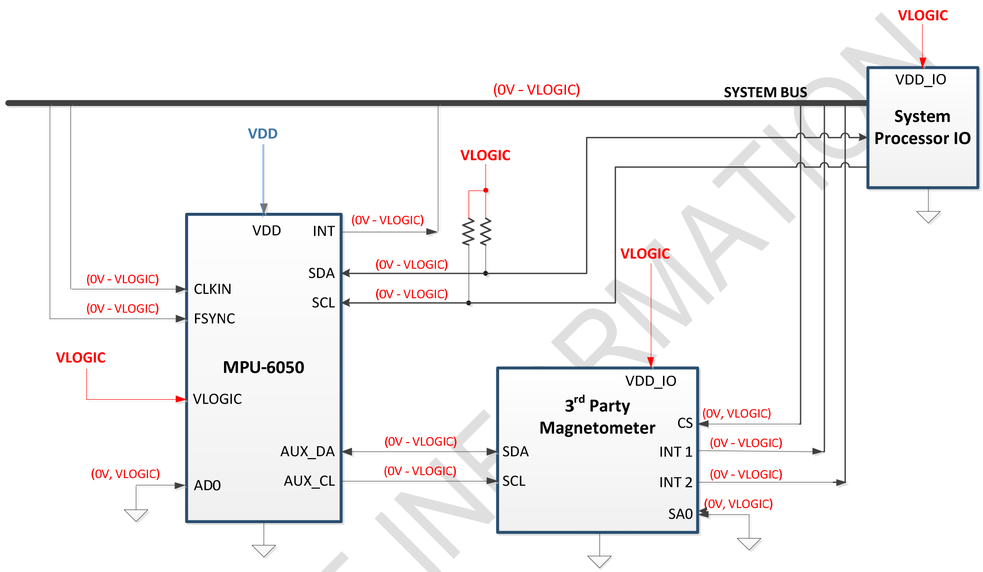
Note: The power-on-reset value for *AUX_VDDIO* is 0.

VLOGIC may be set to be equal to VDD or to another voltage. However, VLOGIC must be ≤ VDD at all times. When *AUX_VDDIO* is set to 0 (its power-on-reset value), VLOGIC is the power supply voltage for both the microprocessor system bus and the auxiliary I^2C bus, as shown in the figure of Section 10.3. When *AUX_VDDIO* is set to 1, VLOGIC is the power supply voltage for the microprocessor system bus and VDD is the supply for the auxiliary I^2C bus, as shown in the figure of Section 10.4.

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

10.3 Logic Levels Diagram for AUX_*VDDIO* = 0

The figure below depicts a sample circuit with a third party magnetometer attached to the auxiliary I^2C bus. It shows logic levels and voltage connections for *AUX_VDDIO* = 0. Note: Actual configuration will depend on the auxiliary sensors used.



I/O Levels and Connections for *AUX_VDDIO* = 0

Notes:

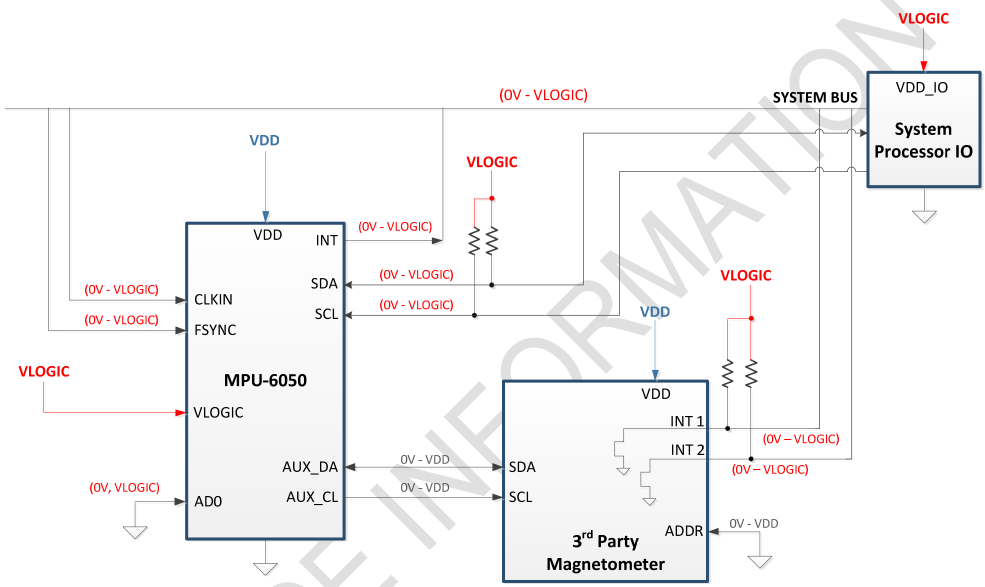
1. AUX_VDDIO determines the IO voltage levels of AUX_DA and AUX_CL (0 = set output levels relative to VLOGIC)
2. CLKOUT is referenced to VDD.
3. All other MPU-6050 logic IOs are referenced to VLOGIC.

De La Salle University

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

10.4 Logic Levels Diagram for *AUX_VDDIO* = 1

The figure below depicts a sample circuit with a 3rd party magnetometer attached to the auxiliary I^2C bus. It shows logic levels and voltage connections for *AUX_VDDIO* = 1. This configuration is useful when the auxiliary sensor has only one supply for logic and power. Note: Actual configuration will depend on the auxiliary sensors used.



Voltage/ Configuration	Configuration 1	Configuration 2
VLOGIC	1.8V±5%	3.0V±5%
VDD	2.5V±5%	3.0V±5%
AUX_VDDIO	1	1

I/O Levels and Connections for Two Example Power Configurations (*AUX_VDDIO* = 1)

Notes:

1. *AUX_VDDIO* determines the IO voltage levels of AUX_DA and AUX_CL. *AUX_VDDIO* = 1 sets output levels relative to VDD.
2. 3rd-party auxiliary device logic levels are referenced to VDD. Setting INT1 and INT2 to open drain configuration provides voltage compatibility when VDD ≠ VLOGIC. When VDD = VLOGIC, INT1 and INT2 may be set to push-pull outputs, and external pull-up resistors are not needed.
3. CLKOUT is referenced to VDD.
4. All other MPU-6050 logic IOs are referenced to VLOGIC.

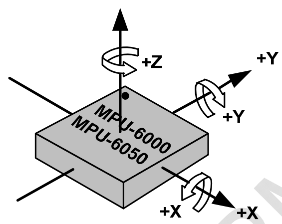
InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

11 Assembly

This section provides general guidelines for assembling InvenSense Micro Electro-Mechanical Systems (MEMS) gyros packaged in Quad Flat No leads package (QFN) surface mount integrated circuits.

11.1 Orientation of Axes

The diagram below shows the orientation of the axes of sensitivity and the polarity of rotation. Note the pin 1 identifier (•) in the figure.

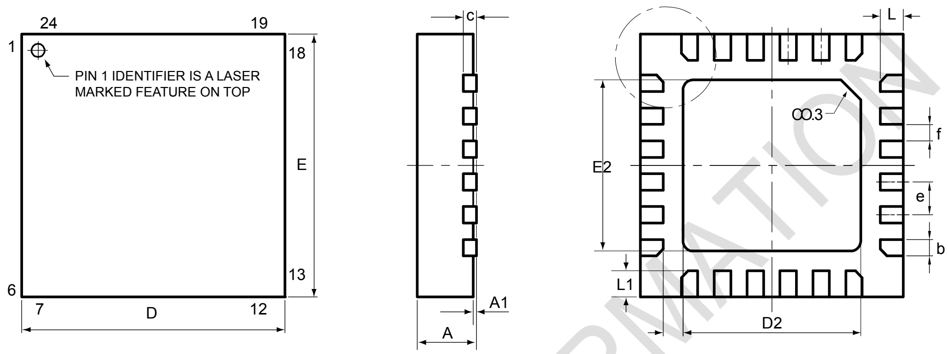


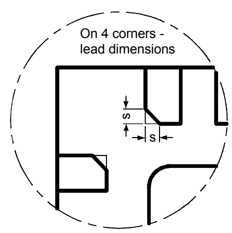
Orientation of Axes of Sensitivity and Polarity of Rotation

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

11.2 Package Dimensions

24 Lead QFN (4x4x0.9) mm NiPdAu Lead-frame finish

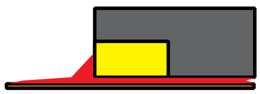




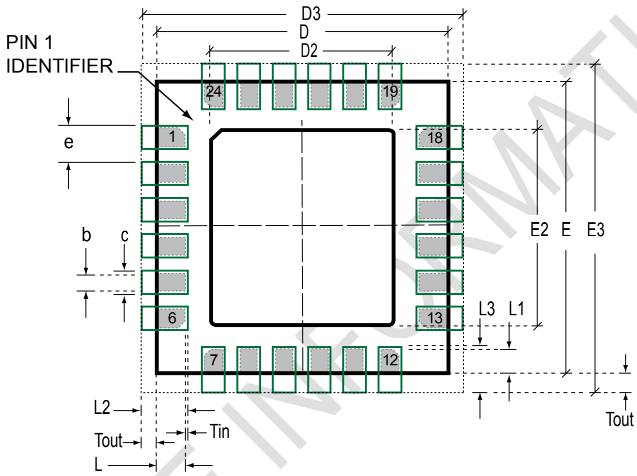
SYMBOLS	DIMENSIONS IN MILLIMETERS		
	MIN	NOM	MAX
A	0.85	0.90	0.95
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
c	---	0.20 REF	---
D	3.90	4.00	4.10
D2	2.65	2.70	2.75
E	3.90	4.00	4.10
E2	2.55	2.60	2.65
e	---	0.50	---
f (e-b)	---	0.25	---
K	0.25	0.30	0.35
L	0.30	0.35	0.40
L1	0.35	0.40	0.45
s	0.05	---	0.15

11.3 PCB Design Guidelines

The Pad Diagram using a JEDEC type extension with solder rising on the outer edge is shown below. The Pad Dimensions Table shows pad sizing (mean dimensions) recommended for the MPU-60X0 product.



JEDEC type extension with solder rising on outer edge



PCB Layout Diagram

SYMBOLS	DIMENSIONS IN MILLIMETERS	NOM
	Nominal Package I/O Pad Dimensions	
e	Pad Pitch	0.50
b	Pad Width	0.25
L	Pad Length	0.35
L1	Pad Length	0.40
D	Package Width	4.00
E	Package Length	4.00
D2	Exposed Pad Width	2.70
E2	Exposed Pad Length	2.60
	I/O Land Design Dimensions (Guidelines)	
D3	I/O Pad Extent Width	4.80
E3	I/O Pad Extent Length	4.80
c	Land Width	0.35
Tout	Outward Extension	0.40
Tin	Inward Extension	0.05
L2	Land Length	0.80
L3	Land Length	0.85

PCB Dimensions Table (for PCB Lay-out Diagram)

	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

11.4 Assembly Precautions

11.4.1 Gyroscope Surface Mount Guidelines

InvenSense MEMS Gyros sense rate of rotation. In addition, gyroscopes sense mechanical stress coming from the printed circuit board (PCB). This PCB stress can be minimized by adhering to certain design rules:

When using MEMS gyroscope components in plastic packages, PCB mounting and assembly can cause package stress. This package stress in turn can affect the output offset and its value over a wide range of temperatures. This stress is caused by the mismatch between the Coefficient of Linear Thermal Expansion (CTE) of the package material and the PCB. Care must be taken to avoid package stress due to mounting.

Traces connected to pads should be as symmetric as possible. Maximizing symmetry and balance for pad connection will help component self alignment and will lead to better control of solder paste reduction after reflow.

Any material used in the surface mount assembly process of the MEMS gyroscope should be free of restricted RoHS elements or compounds. Pb-free solders should be used for assembly.

11.4.2 Exposed Die Pad Precautions

The MPU-60X0 has very low active and standby current consumption. The exposed die pad is not required for heat sinking, and should not be soldered to the PCB. Failure to adhere to this rule can induce performance changes due to package thermo-mechanical stress. There is no electrical connection between the pad and the CMOS.

11.4.3 Trace Routing

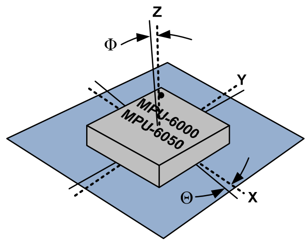
Routing traces or vias under the gyro package such that they run under the exposed die pad is prohibited. Routed active signals may harmonically couple with the gyro MEMS devices, compromising gyro response. These devices are designed with the drive frequencies as follows: X = 33±3Khz, Y = 30±3Khz, and Z=27±3Khz. To avoid harmonic coupling don't route active signals in non-shielded signal planes directly below, or above the gyro package. Note: For best performance, design a ground plane under the e-pad to reduce PCB signal noise from the board on which the gyro device is mounted. If the gyro device is stacked under an adjacent PCB board, design a ground plane directly above the gyro device to shield active signals from the adjacent PCB board.

11.4.4 Component Placement

Do not place large insertion components such as keyboard or similar buttons, connectors, or shielding boxes at a distance of less than 6 mm from the MEMS gyro. Maintain generally accepted industry design practices for component placement near the MPU-60X0 to prevent noise coupling and thermo-mechanical stress.

11.4.5 PCB Mounting and Cross-Axis Sensitivity

Orientation errors of the gyroscope and accelerometer mounted to the printed circuit board can cause cross-axis sensitivity in which one gyro or accel responds to rotation or acceleration about another axis, respectively. For example, the X-axis gyroscope may respond to rotation about the Y or Z axes. The orientation mounting errors are illustrated in the figure below.



Package Gyro & Accel Axes (– – –) Relative to PCB Axes (——) with Orientation Errors (Θ and Φ)

The table below shows the cross-axis sensitivity as a percentage of the gyroscope or accelerometer's sensitivity for a given orientation error, respectively.

Cross-Axis Sensitivity vs. Orientation Error

Orientation Error (θ or Φ)	Cross-Axis Sensitivity (sinθ or sinΦ)
0º	0%
0.5º	0.87%
1º	1.75%

The specifications for cross-axis sensitivity in Section 6.1 and Section 6.2 include the effect of the die orientation error with respect to the package.

11.4.6 MEMS Handling Instructions

MEMS (Micro Electro-Mechanical Systems) are a time-proven, robust technology used in hundreds of millions of consumer, automotive and industrial products. MEMS devices consist of microscopic moving mechanical structures. They differ from conventional IC products, even though they can be found in similar packages. Therefore, MEMS devices require different handling precautions than conventional ICs prior to mounting onto printed circuit boards (PCBs).

The MPU-60X0 has been qualified to a shock tolerance of 10,000*g*. InvenSense packages its gyroscopes as it deems proper for protection against normal handling and shipping. It recommends the following handling precautions to prevent potential damage.

- Do not drop individually packaged gyroscopes, or trays of gyroscopes onto hard surfaces. Components placed in trays could be subject to *g*-forces in excess of 10,000*g* if dropped.
- Printed circuit boards that incorporate mounted gyroscopes should not be separated by manually snapping apart. This could also create *g*-forces in excess of 10,000*g.*

11.4.7 ESD Considerations

Establish and use ESD-safe handling precautions when unpacking and handling ESD-sensitive devices.

- Store ESD sensitive devices in ESD safe containers until ready for use. The Tape-and-Reel moisture-sealed bag is an ESD approved barrier. The best practice is to keep the units in the original moisture sealed bags until ready for assembly.

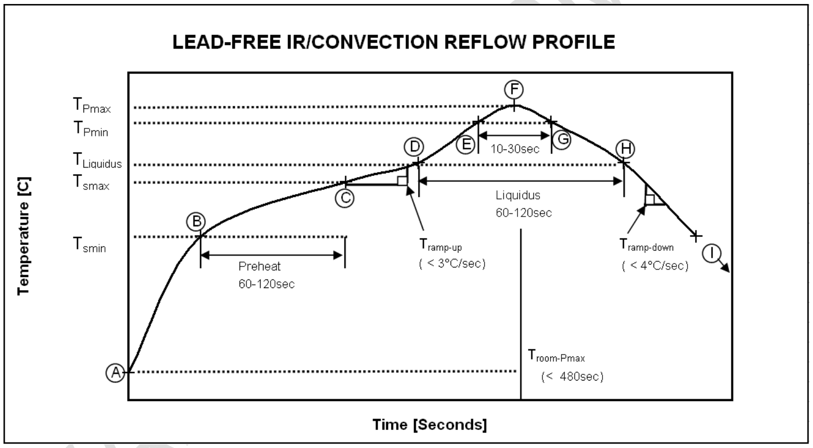
Restrict all device handling to ESD protected work areas that measure less than 200V static charge. Ensure that all workstations and personnel are properly grounded to prevent ESD.

11.4.8 Reflow Specification

Qualification Reflow: The MPU-60X0 was qualified in accordance with IPC/JEDEC J-STD-020D.01. This standard classifies proper packaging, storage and handling in order to avoid subsequent thermal and mechanical damage during the solder reflow attachment phase of assembly. The classification specifies a sequence consisting of a bake cycle, a moisture soak cycle in a temperature humidity oven, followed by three solder reflow cycles and functional testing for qualification. All temperatures refer to the topside of the QFN package, as measured on the package body surface. The peak solder reflow classification temperature requirement is (260 +5/-0°C) for lead-free soldering of components measuring less than 1.6 mm in thickness.

Production Reflow: Check the recommendations of your solder manufacturer. For optimum results, production solder reflow processes should reduce exposure to high temperatures, and use lower ramp-up and ramp-down rates than those used in the component qualification profile shown for reference below.

Production reflow should never exceed the maximum constraints listed in the table and shown in the figure below that were used for the qualification profile, as these represent the maximum tolerable ratings for the device.



Approved IR/Convection Solder Reflow Curve

Temperature Set Points for IR / Convection Reflow Corresponding to Figure Above

Step	Setting	CONSTRAINTS		
		Temp (°C)	Time (sec)	Rate (°C/sec)
A	T_{room}	25		
B	T_{Smin}	150		
C	T_{Smax}	200	$60 < t_{BC} < 120$	
D	$T_{Liquidus}$	217		$r_{(TLiquidus-TPmax)} < 3$
E	T_{Pmin} [255°C, 260°C]	255		$r_{(TLiquidus-TPmax)} < 3$
F	T_{Pmax} [260°C, 265°C]	260	$t_{AF} < 480$	$r_{(TLiquidus-TPmax)} < 3$
G	T_{Pmin} [255°C, 260°C]	255	$10 < t_{EG} < 30$	$r_{(TPmax-TLiquidus)} < 4$
H	$T_{Liquidus}$	217	$60 < t_{DH} < 120$	
I	T_{room}	25		

Notes:

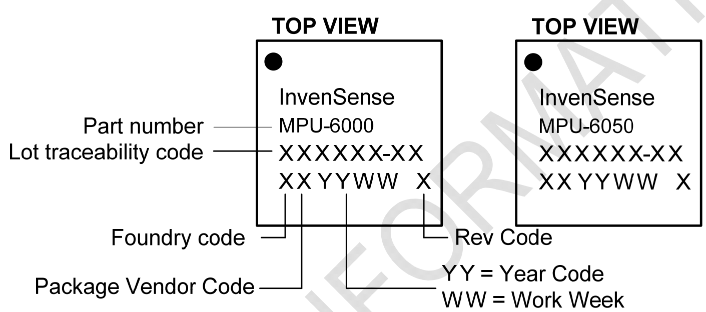
- For users T_{Pmax} must not exceed the Classification temperature (260°C).
- For suppliers T_{Pmax} must equal or exceed the classification temperature.

11.5 Storage Specifications

The storage specification of the MPU-60X0 conforms to IPC/JEDEC J-STD-020D.01 Moisture Sensitivity Level (MSL) 3.

Calculated shelf-life in moisture-sealed bag	12 months -- Storage conditions: <40°C and <90% RH
After opening moisture-sealed bag	168 hours -- Storage conditions: ambient ≤30°C at 60%RH

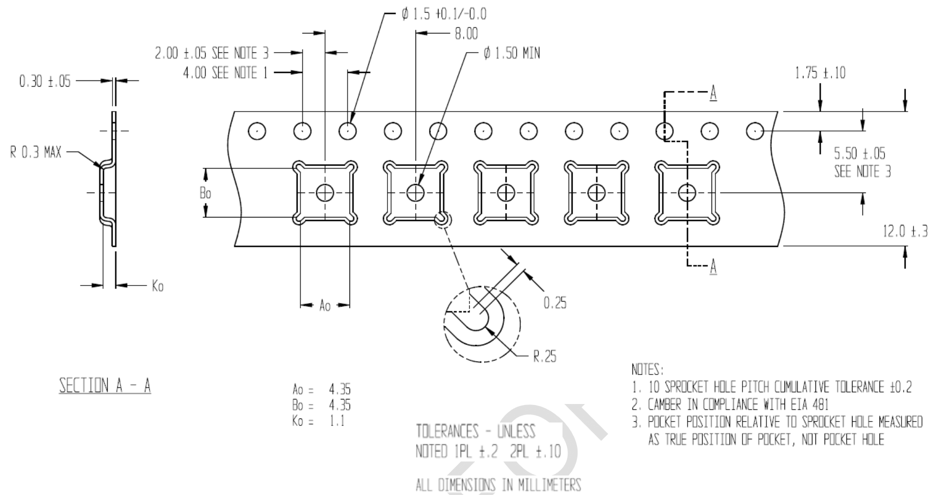
11.6 Package Marking Specification



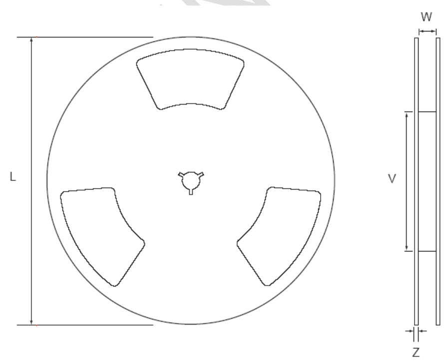
Package Marking Specification

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

11.7 Tape & Reel Specification



Tape Dimensions

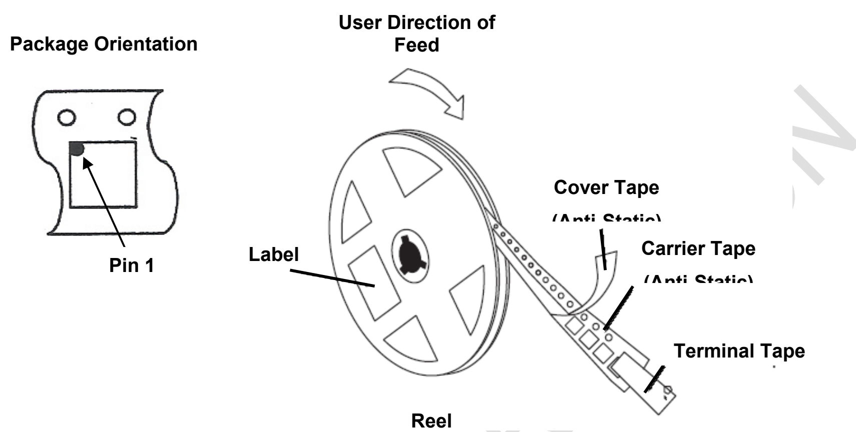


Reel Outline Drawing

Reel Dimensions and Package Size

PACKAGE SIZE	REEL (mm)			
	L	V	W	Z
4x4	330	100	13.2	2.2

InvenSense	**MPU-6000/MPU-6050 Product Specification**	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

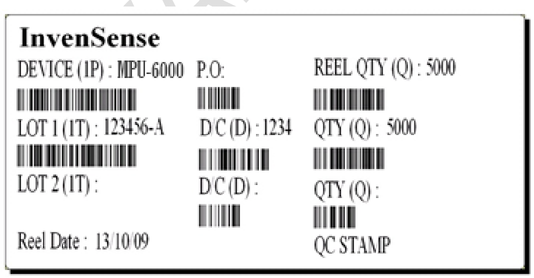


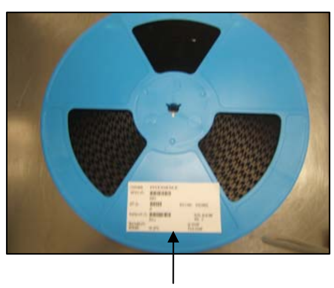
Tape and Reel Specification

Reel Specifications

Quantity Per Reel	**5,000**
Reels per Box	**1**
Boxes Per Carton (max)	**3**
Pcs/Carton (max)	**15,000**

11.8 Label

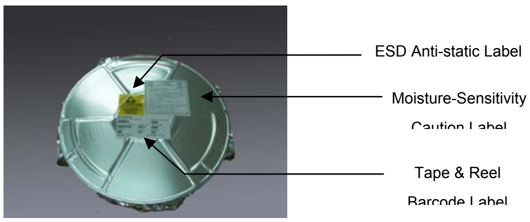




Location of Label

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

11.9 Packaging



Moisture Barrier Bag With Labels

Moisture-Sensitive Caution Label

Reel in Box

Box with Tape & Reel Label

12 Reliability

12.1 Qualification Test Policy

InvenSense's products complete a Qualification Test Plan before being released to production. The Qualification Test Plan for the MPU-60X0 followed the JEDEC 47G.01 Standard, "Stress-Test-Driven Qualification of Integrated Circuits," with the individual tests described below.

12.2 Qualification Test Plan

Accelerated Life Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
High Temperature Operating Life (HTOL/LFR)	JEDEC JESD22-A108C, Dynamic, 3.63V biased, Tj>125°C [read-points 168, 500, 1000 hours]	3	77	(0/1)
Highly Accelerated Stress Test, Unbiased(1) (HAST)	JEDEC JESD22-A118 Condition A, 130°C, 85%RH, 33.3 psia., unbiased, [read-point 96 hours]	3	77	(0/1)
High Temperature Storage Life (HTS)	JEDEC JESD22-A103C, Cond. A, 125°C, Non-Biased Bake [read-points 168, 500, 1000 hours]	3	77	(0/1)

Device Component Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
ESD-HBM	JEDEC JESD22-A114F, (2KV)	1	3	(0/1)
ESD-MM	JEDEC JESD22-A115-A, (200V)	1	3	(0/1)
Latch Up	JEDEC JESD78B Class II (2), 125°C; Level A ±100mA	1	6	(0/1)
Mechanical Shock	JEDEC JESD22-B104C, Mil-Std-883H, method 2002.5, Cond. E, 10,000*g's*, 0.2ms, ±X, Y, Z – 6 directions, 5 times/direction	3	5	(0/1)
Vibration	JEDEC JESD22-B103B, Variable Frequency (random), Cond. B, 5-500Hz, X, Y, Z – 4 times/direction	3	5	(0/1)
Temperature Cycling (TC) (1)	JEDEC JESD22-A104D Condition N, [-40°C to +85°C], Soak Mode 2 [5'], 100 cycles	3	77	(0/1)

Board Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
Board Mechanical Shock	JEDEC JESD22-B104C, Mil-Std-883H, method 2002.5, Cond. E, 10000*g's*, 0.2ms, ±X, Y, Z – 6 directions, 5 times/direction	1	5	(0/1)

(1) Tests are preceded by MSL3 Preconditioning in accordance with JEDEC JESD22-A113F

13 Environmental Compliance

The MPU-60X0 is RoHS and Green compliant and is in full environmental compliance as evidenced in report HS-MPU-6000A, Materials Declaration Data Sheet.

Environmental Declaration Disclaimer:

InvenSense believes this environmental information to be correct but cannot guarantee accuracy or completeness. Conformity documents for the above component constitutes are on file. InvenSense subcontracts manufacturing and the information contained herein is based on data received from vendors and suppliers, which has not been validated by InvenSense.

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.1 Release Date: 10/24/2011

G2 Intel Curie Datasheet

Intel® Curie™ Module

Datasheet

March 2017

Document rev. 1.3

The Intel® Curie™ module is a hardware product offering design flexibility in a small form factor. This complete, low-power solution comes with compute, motion sensor, Bluetooth® low energy, battery-charging, and pattern matching capabilities for optimized analysis of sensor data—enabling quick and easy identification of actions and motions.

Intel® Quark™ SE Microcontroller C1000 Processor Core

- x86 ISA-compatible CPU
- 32 MHz clock, 32-bit address bus
- 8 kB 2-way L1 instruction cache
- 1.33 DMIPs/MHz, total of 42.56 DMIPs max.

Sensor Subsystem

- ARC* EM4 DSP core with floating point unit
- 8 kB L1 instruction cache, 8 kB data CCM
- Tightly coupled IO to interface sensors/actuators
- 1.4 DMIPS/MHz

Pattern Recognition Accelerator

- Built-in Neuron nodes
- K-Nearest Neighbors (k-NN) and Radial Basis Functions (RBF)

Sensor Subsystem Interfaces

- Two I^{2}C master with standard and fast modes
- Two SPI master up to 16 MHz clock with 4 chip selects
- 19 channel 12-bit ADC
- 16 GPIOs
- Two timers

Power Management

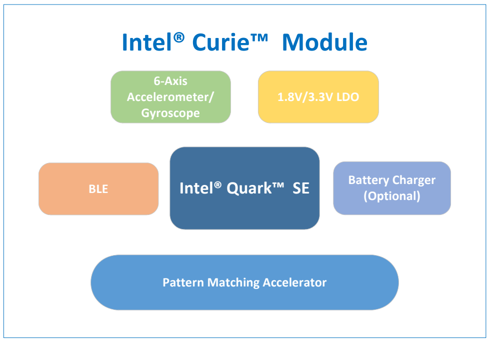
- SoC states: Active, Sleep and Off
- Sensor subsystem: Sensing active, sensing wait and sensing standby
- Platform power DC-DC 1.8 V, 3.3 V

Memory

- 384 kB Flash +8 kB OTP Flash
- 80 kB SRAM

Thermals

- -25 to 70°C operating range



Security

- Secure boot and update
- Isolated SRAM regions
- Flash (NVM) read/write access

Host Interfaces

- USB 1.1 FS device
- Two I^{2}C master/slave with standard, fast & fast mode plus
- Two SPI master up to 16MHz clock with 4 chip selects
- One SPI slave
- Two UARTs, 300 kBaud to 2 MBaud
- Four timers
- Four PWM
- I^{2}S with sample size from 12 to 32-bit
- 32 GPIOs
- 19 comparators

Clock

- 32 kHz and 32 MHz crystal oscillators
- 32 MHz oscillator
- 32 kHz RTC and AON counters/timers
- control

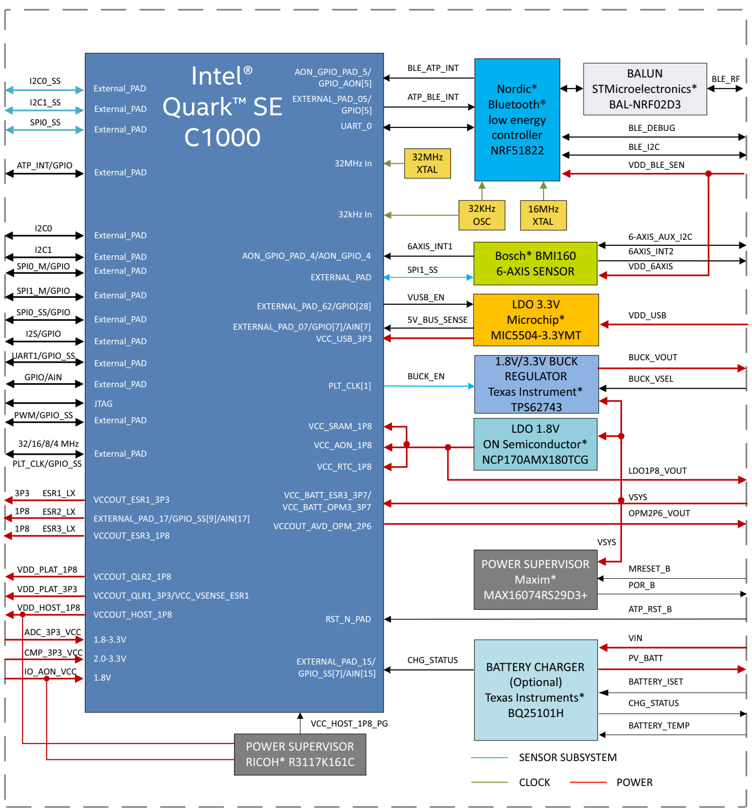
Hardware Reference Designs

Intel has multiple reference designs available through select partners and ODMs. One reference design for the maker community, for example, is the Arduino 101* (branded Genuino 101* in some countries) board which can be used for quick prototyping of concepts. It has the same pin configuration as the Arduino UNO, making it compatible with the majority of components in the Arduino community.

Intel Software Platform

Intel offers several software solutions for the Intel® Curie™ module. These include an open source solution designed to assist developers by increasing the speed and ease of development for a wide variety of products. The Intel® Curie™ Open Developer Kit (ODK) provides access to software, firmware, and services needed for a variety of use cases. The software platform affords more flexibility to the developers to build with multiple environments and the capability to integrate into different IDEs.

Intel® Curie™ Module Block Diagram



For more information, please visit: http://www.intel.com/curie

Intel® Curie™ Module
Datasheet
2

March 2017
Document rev. 1.3

De La Salle University

Content

Tables

Figures

Revision History

Revision	Description	Date
1.0	Initial release.	August 2016
1.1	Reorganized the content into the existing chapters and renamed some chapters. Updated the list of reference documents. Added sections about Manufacturing Information, Ordering information, Package marking, and ESD considerations. Updated Table 1-9 to remove external pull-up/pull-down recommendation for JTAG signals.	October 2016
1.2	Added reference to *Intel® Curie™ Power Sequence Considerations* application note in the Power and timing considerations section of the Specifications chapter.	November 2016
1.21	Minor fixes.	December 2016
1.3	Restructured and expanded the content of chapters "Ball Map and Pin Definitions", "Specifications", and "Detailed Description". Also updated the content of the front cover.	March 2017

§

About This Datasheet

This section introduces the Intel® Curie™ module. It covers:

- Intel® Curie™ module
- Intended audience
- Resources, references and terminology

Intel® Curie™ module

This datasheet documents the Intel® Curie™ module, an advanced device built around the Intel® Quark™ SE microcontroller C1000, integrating compute, sense, awareness, connectivity and a programmable input/output controller within a common package.[1] See Chapter 3, "Detailed Description", for more descriptive details.

Intended audience

This datasheet is intended to provide detailed technical information about the Intel® Curie™ module to the vendors, system integrators, and other engineers and technicians who need this level of information.

Resources, references and terminology

See Chapter 4, "Reference and Resources" for information on Software support, Related documents, and Intel® Curie™ module-related community resources as well as Component reference and a list of Terminology.

1. Note that this module is not a FCC-certified module. Emission testing has been performed and designs based on the Intel® Curie™ module that have FCC certification are available, but all new designs based on the module require regulatory approval prior to public availability.

1 Ball Map and Pin Definitions

1.1 Module physical bump map

Table 1-1 shows a map of the module pin out map as a top view into the part.

Table 1-1. Module ball / pin map

	1	2	3	4	21	22	23	24
A	NO BALL	ATP_GND1	I2S_RXD	SPI0_M_CS1	6AXIS_SDA	6AXIS_SCL	6AXIS_INT2	ATP_GND1
B	I2S_RWS	I2S_RSCK	SPI0_M_CS0	ATP_RST_B	SPI1_M_CS2	SPI1_M_MISO	BLE_SDA	BT_GPIO
C	I2S_TWS	I2S_TSCK	SPI0_M_CS2	SPI0_M_SCK	SPI1_M_SCK	SPI1_M_CS3	ATP_GND1	GPIO/AIN_14
D	I2S_TXD	I2C1_SDA	SPI0_M_MOSI	SPI0_M_MISO	SPI1_M_CS1	SPI1_M_CS0	MRESET_B	BLE_SW_CLK
E	I2C1_SCL	I2C1_SS_SDA	PLT_CLK_0	ATP_SPI_S_SCK	AON_IO_VCC	SPI1_M_MOSI	POR_B	BLE_SWDIO
F	I2C1_SS_SCL	PWM3_OUT	ATP_SPI_S_CS	ATP_SPI_S_MOSI	ATP_INT3	ATP_INT0	BLE_SCL	BLE_RF
G	PWM2_OUT	PWM1_OUT	ATP_SPI_S_MISO	SPI0_SS_CS3	ATP_GND1	COMP_AREF	BLE_DEC2	ATP_GND1
H	PWM0_OUT	GPIO/AIN_12	SPI0_SS_CS1	SPI0_SS_CS0	GPIO/AIN_11	CMP_3P3_VCC	ADC_3P3_VCC	VDD_BLE_SEN
J	AVD_OPM_2P6	GPIO/AIN_10	SPI0_SS_MISO	SPI0_SS_SCK	UART1_TX	UART1_CTS	USB_DM	USB_DP
K	BUCK_VOUT	BUCK_VSEL	SPI0_SS_CS2	VDD_USB	UART1_RTS	UART1_RX	ATP_TRST_B	VIN[1]
L	LDO1P8_VOUT	GPIO/AIN_13	ATP_INT2	VSYS	ATP_ADC_AGND	BATT_ISET	ATP_TCK	ATP_TMS
M	PV_BATT	VDD_PLAT_1P8	SPI0_SS_MOSI	ESR2_LX	ATP_TDI	CHG_STATUS	ATP_TDO	I2C0_SCL
N	ESR1_LX	VDD_PLAT_3P3	ATP_INT1	ESR2_VBATT	VIN[2]	BATT_TEMP	I2C0_SS_SDA	I2C0_SS_SCL
P	ATP_GND1	ESR1_VBATT	VDD_HOST_1P8	ESR3_LX	SW_FG_VBATT	ATP_GND1	I2C0_SDA	ATP_GND1

1.2 Module-to-SoC mapping table

The Intel® Curie™ module is based on the Intel® Quark™ SE microcontroller C1000. Some of the microcontroller signals are used internally within the module to operate the integrated Bosch* BMI160 six-axis sensor, the battery charger and the Nordic* nRF51822 Bluetooth® low energy controller, while other SoC signals are routed directly to the module interface.

Table 1-2 provides the list of high-level reference mapping signals between the Intel Quark microcontroller core and the Intel Curie module. It also includes, where available, alternate functions that interface pins can be configured for.

Consult the Intel® Quark™ SE microcontroller C1000 datasheet for additional information on specific microcontroller functions and registers.

Table 1-2. Intel® Curie™ module to Intel® Quark™ SE microcontroller C1000 signal mapping

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Primary Function	Alt Function1	Alt Function2	Intel® Quark™ SE Microcontroller C1000/Component Ball Name	Intel Quark microcontroller Ball Number
A1	No Ball	-	-	-	-	-
A2	ATP_GND1[3]	ATP_GND1[3]	VSS3	ATP_GND1	VSS	F3
A3	I2S_RXD	I2S_RXD	GPIO[15]	I2S_RXD	EXTERNAL_PAD_49	B8
A4	SPI0_M_CS1	SPI0_M_CS_B[1]	GPIO[25]	SPI0_M_CS_ B[1]	EXTERNAL_PAD_59	A10
A21	6AXIS_SDA	6AXIS_SDA	ASDX (6AXIS / 2)	-	6AXIS I^2C to external magnetometer to get 9AXIS	-
A22	6AXIS_SCL	6AXIS_SCL	ASCX (6AXIS / 3)	-		-
A23	6AXIS_INT2	6AXIS_INT2	INT2 (6AXIS / 9)	-	6AXIS Interrupt to external magnetometer to get 9AXIS	-
A24	ATP_GND1[9]	ATP_GND1[9]	VSS9	ATP_GND1	VSS	M1
B1	I2S_RWS	I2S_RWS	GPIO[17]	I2S_RWS	EXTERNAL_PAD_51	B9
B2	I2S_RSCK	I2S_RSCK	GPIO[16]	I2S_RSCK	EXTERNAL_PAD_50	A8
B3	SPI0_M_CS0	SPI0_M_CS_B[0]	GPIO[24]	SPI0_M_CS_ B[0]	EXTERNAL_PAD_58	E8
B4	ATP_RST_B	RST_B	RST_B		RST_N_PAD	F11
B21	SPI1_M_CS2	SPI1_M_CS_B[2]	GPIO[13]	SPI1_M_CS_ B[2]	EXTERNAL_PAD_47	B07
B22	SPI1_M_ MISO	SPI1_M_MISO	GPIO[9]	SPI1_M_MISO	EXTERNAL_PAD_43	D6
B23	BLE_SDA	BLE P0_31 / E8	-	-	BLE I^2C to external device optional	-
B24	BT_GPIO	BLE P0_18 / H1	-	-	BLE GPIO to external device optional	-
C1	I2S_TWS	I2S_TWS	GPIO[19]	I2S_TWS	EXTERNAL_PAD_53	C9
C2	I2S_TSCK	I2S_TSCK	GPIO[18]	I2S_TSCK	EXTERNAL_PAD_52	A9
C3	SPI0_M_CS2	SPI0_M_CS_B[2]	GPIO[26]	SPI0_M_CS_ B[2]	EXTERNAL_PAD_60	B10
C4	SPI0_M_SCK	SPI0_M_SCK	GPIO[21]	SPI0_M_SCK	EXTERNAL_PAD_55	D8

Table 1-2. Intel® Curie™ module to Intel® Quark™ SE microcontroller C1000 signal mapping (continued)

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Primary Function	Alt Function1	Alt Function2	Intel® Quark™ SE Microcontroller C1000/Component Ball Name	Intel Quark microcontroller Ball Number
C21	SPI1_M_SCK	SPI1_M_SCK	GPIO[8]	SPI1_M_SCK	EXTERNAL_PAD_42	C6
C22	SPI1_M_CS3	SPI1_M_CS_B[3]	GPIO[14]	-	EXTERNAL_PAD_48	A7
C23	ATP_GND1[5]	ATP_GND1[5]	VSS5	ATP_GND1	VSS	L12
C24	GPIO/AIN_14	GPIO_SS[6]	GPIO_SS[6]	AIN[14]	EXTERNAL_PAD_14	F01
D1	I2S_TXD	I2S_TXD	GPIO[20]	I2S_TXD	EXTERNAL_PAD_54	D9
D2	I2C1_SDA	I2C1_SDA	I2C1_SDA	-	EXTERNAL_PAD_23	D2
D3	SPI0_M_ MOSI	SPI0_M_MOSI	GPIO[23]	SPI0_M_MOSI	EXTERNAL_PAD_57	E9
D4	SPI0_M_ MISO	SPI0_M_MISO	GPIO[22]	SPI0_M_MISO	EXTERNAL_PAD_56	E7
D21	SPI1_M_CS1	SPI1_M_CS_B[1]	GPIO[12]	SPI1_M_CS_ B[1]	EXTERNAL_PAD_46	C7
D22	SPI1_M_CS0	SPI1_M_CS_B[0]	GPIO[11]	SPI1_M_CS_ B[0]	EXTERNAL_PAD_45	D7
D23	MRESET_B	MR (MAX16074 / B2)	-	-	-	-
D24	BLE_SW_CLK	SWDCLK (BLE / H2)	GPIO[27]	SPI0_M_CS_ B[3]	EXTERNAL_PAD_61	C10
E1	I2C1_SCL	I2C1_SCL	-	-	EXTERNAL_PAD_22	D1
E2	I2C1_SS_ SDA	I2C1_SS_SDA	-	-	EXTERNAL_PAD_26	B3
E3	PLT_CLK_0	PLT_CLK[0]	GPIO_SS[14]	PLT_CLK[0]	EXTERNAL_PAD_67	D12
E4	ATP_SPI_S_ SCK	SPI_S_SCK	GPIO[2]	AIN[2]/ SPI_S_SCK	EXTERNAL_PAD_02	H5
E21	AON_IO_VCC	AON_IO_VCC	VCC_IO_AON1	VCC_IO_AON1	VCC_IO_AON1	A11
			VCC_IO_AON2	VCC_IO_AON2	VCC_IO_AON2	G6
E22	SPI1_M_ MOSI	SPI1_M_MOSI	GPIO[10]	SPI1_M_MOSI	EXTERNAL_PAD_44	E6
E23	POR_B	RESETN (MAX16074 / B1)	-	-	Power supervisory Power On Reset output	-
E24	BLE_SWDIO	SWDIO (BLE/J2)	GPIO[6]	AIN[6]	EXTERNAL_PAD_06	H4
F1	I2C1_SS_SCL	I2C1_SS_SCL	I2C1_SS_SCL	-	EXTERNAL_PAD_27	A3
F2	PWM3_OUT	PWM[3]	GPIO_SS[13]	PWM[3]	EXTERNAL_PAD_66	B11
F3	ATP_SPI_S_ CS	SPI_S_CS_B	GPIO[0]	AIN[0] / SPI_S_CS_B	EXTERNAL_PAD_00	F2
F4	ATP_SPI_S_ MOSI	SPI_S_MOSI	GPIO[3]	AIN[3] / SPI_S_MOSI	EXTERNAL_PAD_03	J6
F21	ATP_INT3	ATP_INT3	GPIO_AON[3]	-	AON_GPIO_PAD_3	F9
F22	ATP_INT0	ATP_INT0	GPIO_AON[0]	-	AON_GPIO_PAD_0	G9
F23	BLE_SCL	BLE P0_30 / D8		-	BLE I^2C to external device optional	-
F24	BLE_RF	BALUN SE / A1	-	-	Antenna Micro Strip (50 ohms)	-

De La Salle University

intel

Table 1-2. Intel® Curie™ module to Intel® Quark™ SE microcontroller C1000 signal mapping (continued)

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Primary Function	Alt Function1	Alt Function2	Intel® Quark™ SE Microcontroller C1000/Component Ball Name	Intel Quark microcontroller Ball Number
G1	PWM2_OUT	PWM[2]	GPIO_SS[12]	PWM[2]	EXTERNAL_PAD_65	C11
G2	PWM1_OUT	PWM[1]	GPIO_SS[11]	PWM[1]	EXTERNAL_PAD_64	D11
G3	ATP_SPI_S_MISO	SPI_S_MISO	GPIO[1]	AIN[1]	EXTERNAL_PAD_01	G4
G4	SPI0_SS_CS3	SPI0_SS_CS_B[3]	GPIO[30]	SPI0_SS_CS_B[3]	EXTERNAL_PAD_34	A4
G21	ATP_GND1[7]	ATP_GND1[7]	VSS7	ATP_GND1	VSS	M11
G22	COMP_AREF	COMP_AREF	COMP_AREF	-	COMP_AREF (or AREF_PAD)	F5
G23	BLE_DEC2	DEC2 (BLE / F1)	-	-	-	-
G24	ATP_GND1[6]	ATP_GND1[6]	VSS6	ATP_GND1	VSS	M12
H1	PWM0_OUT	PWM[0]	GPIO_SS[10]	PWM[0]	EXTERNAL_PAD_63	E10
H2	GPIO/AIN_12	GPIO_SS[4]	GPIO_SS[4]	AIN[12]	EXTERNAL_PAD_12	J04
H3	SPI0_SS_CS1	SPI0_SS_CS_B[1]	SPI0_SS_CS_B[1]	-	EXTERNAL_PAD_32	C4
H4	SPI0_SS_CS0	SPI0_SS_CS_B[0]	SPI0_SS_CS_B[0]	-	EXTERNAL_PAD_31	D4
H21	GPIO/AIN_11	GPIO_SS[3]	GPIO_SS[3]	AIN[11]	EXTERNAL_PAD_11	G1
H22	CMP_3P3_VCC	CMP_3P3_VCC	VCC_CMP_3P3[2]	-	VCC_CMP_3P3[2]	J3
			VCC_CMP_3P3[1]	-	VCC_CMP_3P3[1]	M2
H23	ADC_3P3_VCC	-	-	-	-	-
H24	VDD_BLE_SEN	-	-	-	-	-
J1	OPM2P6_VOUT	OPM2P6_VOUT	VCC_AVD_OPM_2P6	-	VCCOUT_AVD_OPM_2P6	K11
		AVD_OPM_2P6	VCCOUT_AVD_OPM_2P6	-		
		VCC_AVD_OPM_2P6	VCC_AVD_OPM_2P6	-	VCC_AVD_OPM_2P6	K12
		AVD_OPM_2P6		-		
J2	GPIO/AIN_10	GPIO_SS[2]	-	AIN[10]	EXTERNAL_PAD_10	K5
J3	SPI0_SS_MISO	SPI0_SS_MISO	SPI0_SS_MISO	-	EXTERNAL_PAD_28	C3
J4	SPI0_SS_SCK	SPI0_SS_SCK	SPI0_SS_SCK	-	EXTERNAL_PAD_30	D3
J21	UART1_TX	UART1_TX	GPIO_SS[8]	AIN[16]/UART1_TXD	EXTERNAL_PAD_16	L4
J22	UART1_CTS	UART1_CTS	GPIO_SS[0]	AIN[8]/UART1_CTS_B	EXTERNAL_PAD_08	L5
J23	USB_DM	USB_DM	USB_DN	-	USB_PADN	H2
J24	USB_DP	USB_DP	USB_DP	-	USB_PADP	H1

De La Salle University

Table 1-2. Intel® Curie™ module to Intel® Quark™ SE microcontroller C1000 signal mapping (continued)

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Primary Function	Alt Function1	Alt Function2	Intel® Quark™ SE Microcontroller C1000/Component Ball Name	Intel Quark microcontroller Ball Number
K1	BUCK_VOUT	BUCK_VOUT	-	-	-	-
K2	BUCK_VSEL	VSEL1,3 (TPS62743)	-	-	-	-
K3	SPI0_SS_CS2	SPI0_SS_CS2]	SPI0_SS_CS_B[2]	GPIO[29]	EXTERNAL_PAD_03	B4
K4	VDD_USB	VDD_USB	-	-	EXTERNAL_PAD_07	G3
K21	UART1_RTS	UART1_RTS	GPIO_SS[1]	AIN[9]/ UART1_RTS_B	EXTERNAL_PAD_09	M5
K22	UART1_RX	UART1_RX	GPIO_SS[9]	AIN[17]/ UART1_RXD	EXTERNAL_PAD_17	M4
K23	ATP_TRST_B	ATP_TRST_B	TRST_B	-	TRST_PAD	G5
K24	VIN[1]	VIN (BQ25101 / A2)	-	-	-	-
L1	LDO1P8_VOUT	LDO1P8_VOUT	VCC_AON_1P8[2]	-	VCC_AON_1P8	A2
			VCC_SRAM_1P8	-	VCC_SRAM_1P8	C8
			VCC_RTC_1P8	-	VCC_RTC_1P8	G10
			VCC_AON_1P8[1]	-	VCC_AON_1P8	J7
L2	GPIO/AIN_13	GPIO/AIN_13]	GPIO_SS[5]	AIN[13]	EXTERNAL_PAD_13	G2
L3	ATP_INT2	ATP_INT2	GPIO_AON[2]	-	AON_GPIO_PAD_2	E12
L4	VSYS	VSYS	VCC_BATT_OPM_3P7	-	VCC_BATT_OPM_3P7	L10
			VCC_BATT_ESR3_3P7	-	VCC_BATT_ESR3_3P7	M9
L21	ATP_ADC_AGND	ATP_ADC_AGND	VSS_ADC_AGND	-	VSS_ADC_AGND	L3
L22	BATT_ISET	BATT_ISET	-	-	-	-
L23	ATP_TCK	ATP_TCK	TCK	-	TCK_PAD	G7
L24	ATP_TMS	ATP_TMS	TMS	-	TMS_PAD	F6
M1	PV_BATT	OUT (BQ25101 / A1)	-	-	-	-
M2	VDD_PLAT_1P8	VDD_PLAT_1P8	VCCOUT_QLR2_1P8	-	VCCOUT_QLR2_1P8	J11
M3	SPI0_SS_MOSI	SPI0_SS_MOSI	SPI0_SS_MOSI	-	EXTERNAL_PAD_29	E3
M4	ESR2_LX	ESR2_LX	VCCOUT_ESR2_1P8	-	EXTERNAL_PAD_17	J12
M21	ATP_TDI	ATP_TDI	TDI	-	TDI_PAD	F4
M22	CHG_STATUS	CHGN (BQ25101 / C1)	GPIO_SS[7]	AIN[15]	EXTERNAL_PAD_15	J5
M23	ATP_TDO	ATP_TDO	TDO	-	TDO_PAD	F8
M24	I2C0_SCL	I2C0_SCL	I2C0_SCL	-	EXTERNAL_PAD_20	C1

De La Salle University

intel

Table 1-2. Intel® Curie™ module to Intel® Quark™ SE microcontroller C1000 signal mapping (continued)

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Primary Function	Alt Function1	Alt Function2	Intel® Quark™ SE Microcontroller C1000/Component Ball Name	Intel Quark microcontroller Ball Number
N1	ESR1_LX	ESR1_LX	VCCOUT_ESR1_3P3	-	VCCOUT_ESR1_3P3	J8
N2	VDD_PLAT_3P3	VDD_PLAT_3P3	VCC_VSENSE_ESR1	-	VCC_VSENSE_ESR1	H9
			VCCOUT_QLR1_3P3	-	VCCOUT_QLR1_3P3	J9
N3	ATP_INT1	ATP_INT1	GPIO_AON[1]	-	AON_GPIO_PAD_1	E11
N4	ESR2_VBATT	ESR2_VBATT	VCC_BATT_ESR2_3P7	-	VCC_BATT_ESR2_3P7	L11
N21	VIN[2]	VIN (BQ25101 / A2)	-	-	-	-
N22	BATT_TEMP	TSN (BQ25101 / B1)	-	-	-	-
N23	I2C0_SS_SDA	I2C0_SS_SDA	I2C0_SS_SDA	-	EXTERNAL_PAD_24	E1
N24	I2C0_SS_SCL	I2C0_SS_SCL	I2C0_SS_SCL	-	EXTERNAL_PAD_25	E2
P1	ATP_GND1[1]	ATP_GND1[1]	VSS1	ATP_GND1	VSS	A1
P2	ESR1_VBATT	ESR1_VBATT	VCC_BATT_ESR1_3P7	-	VCC_BATT_ESR1_3P7	M10
P3	VDD_HOST_1P8	VDD_HOST_1P8	VCC_HOST_1P8[2]	-	VCC_HOST_1P8[2]	A5
			VCC_HOST_1P8[1]	-	VCC_HOST_1P8[1]	H6
			VCC_PLL_1P8	-	VCC_PLL_1P8	K3
			VCCOUT_HOST_1P8	-	VCCOUT_HOST_1P8	K10
P4	ESR3_LX	ESR3_LX	VCCOUT_ESR3_1P8	-	VCCOUT_ESR3_1P8	K9
P21	SW_FG_VBATT	SW_FG_VBATT / AIN[4])	-	-	EXTERNAL_PAD_04	K6
P22	ATP_GND1[4]	-	-	-	-	-
P23	I2C0_SDA	I2C0_SDA	I2C0_SDA	-	EXTERNAL_PAD_21	C2
P24	ATP_GND1[10]	ATP_GND1[10]	VSS9	ATP_GND1	VSS	M01
-	Internal to 6AXIS	6AXIS_MISO	SDO (6AXIS / 1)	SPI1_SS_MISO	-	B5
-		6AXIS_MOSI	SDX (6AXIS / 14)	SPI1_SS_MOSI	EXTERNAL_PAD_36	C5
-		6AXIS_SCLK	SCX (6AXIS / 13)	SPI1_SS_SCK	EXTERNAL_PAD_37	D5
-		6AXIS_CS	CSB (6AXIS / 12)	SPI1_SS_CS_B[0]	EXTERNAL_PAD_38	E5
-		6AXIS_INT1	INT1 (6AXIS / 4)	GPIO_AON[4]	AON_GPIO_PAD_4	F10

Table 1-2. Intel® Curie™ module to Intel® Quark™ SE microcontroller C1000 signal mapping (continued)

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Primary Function	Alt Function1	Alt Function2	Intel® Quark™ SE Microcontroller C1000/Component Ball Name	Intel Quark microcontroller Ball Number
-	Internal to BLE	UART0_CTS	P0_12 (BLE / J5)	SPI1_SS_CS_B[2]	UART0_CTS_B	A6
-		UART0_TXD	P0_09 (BLE / J7)	UART0_TXD	GPIO[31]	B2
-		UART0_RTS	P0_10 (BLE / H6)	SPI1_SS_CS_B[3]	UART0_RTS_B	B6
-		UART0_RXD	P0_11 (BLE / J6)	UART0_RXD	AIN[18]	K4
-		ATP_BLE_INT	GPIO[5]	AIN[5]	EXTERNAL_PAD_05	L6
-	ATP_GND1	ATP_GND1[2]	VSS2	ATP_GND1	VSS	B1
-		ATP_GND1	VSS8			M6
-	GND	-	-	-	VSS_IO_AON1	B12
-		-	-	-	VSS_IO_AON2	F7
-		-	-	-	VSS_GNDSENSE_OPM	K7
-		-	-	-	VSS_PLL	L2
-		-	-	-	VSS_RTC	H11
-		-	-	-	VSS_USB	J1
-		-	-	-	VSS_AVS_ESR1	L7
-		-	-	-	VSS_GNDSENSE_ESR1	L8
-		-	-	-	VSS_AVS_ESR2	H7
-		-	-	-	VSS_GNDSENSE_ESR2	H8
-		-	-	-	VSS_GNDSENSE_ESR3	M7
-		-	-	-	VSS_AVS_ESR3	M8
-		-	-	-	VSS_AVSS_CMP1	H3
-		-	-	-	VSS_AVSS_CMP2	L1
-		-	PLT_REG_EN	-	PLT_REG_EN	H10
-	VSYS	VCC (MAX16074 / A2)	-	-	Power supervisory Power On Reset output	-

1.3 Pin definitions

This section presents the Intel® Curie™ module pins and their definitions grouped by function.

Note: The ball names differ from the Intel® Curie™ module to the Intel® Quark™ SE microcontroller C1000 processor core integrated in the module. Table 1-2 above shows the mapping between the two.

1.3.1 Battery and power management pins

Table 1-3 lists the pins on the module that provide battery and power management functionality.

The Intel® Curie™ module has a built-in battery charger, which is optional to use. The user can bypass this internal charger. Refer to the *Battery Charging and Management* section of the *Intel® Curie™ Module Design Guide* for more information.

Table 1-3. Battery and power management pins

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Function
J1	OPM2P6_VOUT	2.6 V reference voltage output. Can be used to power CMP_3P3_VCC. Otherwise leave disconnected.
K4	VDD_USB	DC power for USB interface (optional). This is supplied to the module by the USB cable or external 5V supply. It is also connected to SoC AIN7 internally via voltage divider to be able to detect the voltage presence by software (22k pull-down / 36k pull-up to VDD_USB). Then USB voltage converter can be enabled d by VUSB_EN (GPIO28) to provide 3.3V to the SoC for USB controller.
K24, N21	VIN[1], VIN[2]	Battery charger input voltage, These two pins are connected together internally in module to provide more current. Both pins externally need to be connected to the same voltage source.
L1	LDO1P8_VOUT	AON LDO power output. If used outside Intel® Curie™ module, maximum of 50mA can be drawn externally. It is also connected internally to the SoC VCC_AON_1P8[1], VCC_AON_1P8[2], VCC_SRAM_1P8.
L4	VSYS	Main DC input power. Provides input voltage to BUCK_VOUT (TPS62743) converter, VCC_BATT_OPM_3P7 (SoC), VCC_BATT_ESR3_3P7 (SoC), VCC_AON_PWR (NCP170AMX180TCG).
L22	BATT_ISET	Use a Pull-Down resistor value 0.54-13.5kohm to set charging current. Do not leave floating. Refer to (BQ2510H) datasheet for ISET.
M1	PV_BATT	Battery charger output (4.35 V maximum ±50 mV) to battery (positive) to charge it. External application circuit can be added for protection or / and fuel gauge circuit.
M22	CHG_STATUS	Open drain (15 mA maximum) pulls low when battery is being charged.
N21	See K24 above	
N22	BATT_TEMP	Connect to battery thermistor. See TI BQ25101H* Datasheet for details. If battery does not have internal thermistor to measure temperature, then external thermistor can be used touching the battery to measure temperature for safety and meeting charging requirement of the battery manufacture for reliability.
P21	SW_FG_VBATT	Analog input for software fuel gauge (bat voltage measurement). Connected to SoC AIN4 and can be configured via software to use ADC to measure voltage or current.

1.3.2 Platform buck converter pins

Table 1-4 lists the pins that provide platform control for the buck converter integrated in the module.

Table 1-4. Platform buck converter pins

Ball No.	Ball Name	Function
K1	BUCK_VOUT	Buck converter output of 1.8 V / 3.3 V. It can be connected and used for Awake ON (AON) IO supply voltage. Connect a 0.1 uF decoupling capacitor. The input voltage requirement is minimum of 3.7 V and maximum of 4.4 V. Internal signal BUCK_EN (GPIO_SS15) signal is used for software to disable or enable this converter. If software does not configure the BUCK_EN signal and leave it floating then AON_IO_VCC will enable it via a 10M pull-up resistor. It is highly recommended to enable and disable this via software. In some noisy application this pull-up may not be enough to keep this converter enabled all the time.
K2	BUCK_VSEL	0 (Ground) sets BUCK_VOUT to 1.8 V 1 (VSYS) sets BUCK_VOUT to 3.3 V

1.3.3 Additional buck converter pins

Table 1-5 lists the other pins that provide access to the buck converter integrated in the module.

Table 1-5. Additional buck converter pins

Ball No.	Ball Name	Function
M2	VDD_PLAT_1P8	Platform 1.8 V output[1]
M4	ESR2_LX	External inductor and capacitor connection (for Platform 1V8)[1]
N1	ESR1_LX	External inductor and capacitor connection (for Platform 3V3)[1]
N2	VDD_PLAT_3P3	Platform 3.3 V output[1]
N4	ESR2_VBATT	DC input for switching regulator 2[1]
P2	ESR1_VBATT	DC input for switching regulator 1[1]
P3	VDD_HOST_1P8	1.8 V input to host SoC
P4	ESR3_LX	External inductor and capacitor connection (for Host 1V8)

Notes:

1. Refer to the *Intel® Curie™ Module Design Guide* for guidance on the use of these pins. for further information regarding these pins, refer to the Power Architecture section of the *Intel® Quark™ SE Microcontroller C1000 Datasheet* (in the *Power Management* chapter).

1.3.4 Reference voltage pins

Table 1-6 lists the reference voltage pins for the module.

Table 1-6. Reference voltages on module

Ball No.	Ball Name	Function
E21	AON_IO_VCC	Alway-on GPIO supply voltage
G22	COMP_AREF	Comparator reference voltage external input. Software selectable external 0-3.63 V or internal 1.09 V reference voltage
H22	CMP_3P3_VCC	Comparator supply voltage. See Table 2-6 for voltage specifications.
H23	ADC_3P3_VCC	ADC supply and reference voltage

1.3.5 Module ground pins

Table 1-7 lists the ground pins for the module.

Table 1-7. Ground pins

Ball No.	Ball Name	Function
A2	ATP_GND1[3]	Module ground
A24	ATP_GND1[9]	Module ground
C23	ATP_GND1[5]	Module ground
G21	ATP_GND1[7]	Module ground
G24	ATP_GND1[6]	Module ground
L21	ATP_ADC_AGND	Analog ground. Can be connected directly to analog ground at a single point.
P1	ATP_GND1[1]	Module ground
P22	ATP_GND1[4]	Module ground
P24	ATP_GND1[10]	Module ground

1.3.6 Reset pins

Table 1-8 lists the reset pins for the module.

Table 1-8. Reset pins

Ball No.	Ball Name	Function
B4	ATP_RST_B	SoC hardware reset. Active low. See Section 2.2.6 for information on reset wiring.
D23	MRESET_B	Manual reset. Connect POR_B to reset signal ATP_RST_B and pull low trigger a hardware reset. Refer Power supervisor chip (MAX16074) datasheet for more information.
E23	POR_B	Power on reset from power supervisor chip. Active low / open drain requires a pull-up on this signal. Refer Power supervisor chip (MAX16074) datasheet for more information.

1.3.7 Debugging pins

Table 1-9 lists debugging pins for the module.

When the JTAG emulator is used to connect to the debugging pins on the module, the emulator options need to be configured appropriately to ensure there are no conflicts with these debugging pins.

Table 1-9. Debugging pins

Ball No.	Ball Name	Function
D24	BLE_SW_CLK	Two wire debug interface. For JTAG programming using J-Link. This is also connected to a GPIO[27] signal.
E24	BLE_SWDIO	Two wire debug interface. For JTAG programming using J-Link. This is also connected to a GPIO[6] signal.
K23	ATP_TRST_B	JTAG emulator debugger / programmer TRST signal.
L23	ATP_TCK	JTAG emulator debugger / programmer TCK signal.
L24	ATP_TMS	JTAG emulator debugger / programmer TMS signal.
M21	ATP_TDI	JTAG emulator debugger / programmer TDI signal.
M23	ATP_TDO	JTAG emulator debugger / programmer TDO signal.

1.3.8 Wake-capable interrupt pins

Table 1-10 lists the pins that are capable of waking the Intel® Quark™ SE microcontroller from a sleep state.

These signals can be programmed to be GPIO input or output. These are always powered.

Table 1-10. Intel® Quark™ SE microcontroller C1000 always-on wake capable interrupt pins

Ball No.	Ball Name	Function	Drive (Low / High)
F21	ATP_INT3	AON GPIO_AON3 / Always On Wake capable digital IO / Interrupt 3, can be configured for one of the two cores.	4mA / 8mA
F22	ATP_INT0	AON GPIO_AON0 / Always On Wake capable digital IO / Interrupt 0, can be configured for one of the two cores.	4mA / 8mA
L3	ATP_INT2	AON GPIO_AON2 / Always On Wake capable digital IO / Interrupt 2, can be configured for one of the two cores.	4mA / 8mA
N3	ATP_INT1	AON GPIO_AON1 / Always On Wake capable digital IO / Interrupt 1, can be configured for one of the two cores.	4mA / 8mA

1.3.9 Clock out pin

Table 1-11 documents the clock out pin on the module.

The software can configure the PLT_CK_0 pin to bring out the SoC core clock (32/16/8/4). It can be used for debugging or for synchronizing the application circuitry. To reduce the power consumption, and if the application does not need it, you should not enable this pin.

Table 1-11. Clock out pin

Ball No.	Ball Name	Primary Function	Alt Function1	Drive (Low / High)
E3	PLT_CLK_0	32/16/8/4MHz Clock output from module	GPIO_SS[14]	4/8mA

1.3.10 GPIO pin mapping

The following subsections document the general purpose input/output (GPIO) pins for the module. These generic pins can be programmed either as input or as output pins at run time. The GPIO pins can be configured for primary or alternate function. Table 1-14 provides the mapping for the multiple functions each pin can be used for.

Table 1-15 documents the internal GPIO signals within the module.

1.3.10.1 GPIO/analog input pins

Table 1-12 documents the GPIO pins capable of receiving analog inputs.
Please note that the internal pull-up resistors are disabled at reset.

Table 1-12. GPIO/analog input pins

Ball No.	Ball Name	Primary Function	Alt Function1	Drive (Low / High)	Internal Pull-up / Pull-down
C24	GPIO/AIN_14	GPIO_SS[6] (Should be used only for sensor devices)	AIN[14]	Selectable as 4/8	47 kohm
J2	GPIO/AIN_10	GPIO_SS[2] (Should be used only for sensor devices)	AIN[10]		
H21	GPIO/AIN_11	GPIO_SS[3] (Should be used only for sensor devices)	AIN[11]		
H2	GPIO/AIN_12	GPIO_SS[4] (Should be used only for sensor devices)	AIN[12]		
L2	GPIO/AIN_13	GPIO_SS[5] (Should be used only for sensor devices)	AIN[13]		
M22	CHG_STATUS	GPIO_SS[7] (Should be used only for sensor devices)[1]	AIN[15]		

Notes:
1. For CHG_STATUS, please note:
   - CHG_STATUS can be read via software at GPIO_SS[7] / AIN[15]
   - Charge status: Open drain (LOW) means charging and open means complete first charging cycle is complete
   - GPIO_SS[7] / AIN[15] is connected to the external pin (M22) for the device hardware to read the state
   - Internal or external pull-up resistor should be used for this pin
   - If the battery charger is not used then AIN[15] can be used as an external analog input or GPIO_SS[7]. Disable the battery charger by connecting BATT_TEMP to ground.
   - CHG_STATUS or GPIO_SS[7] / AIN[15] is also connected to Nordic* nRF51822 port P0_00. Make sure you keep this pin floating and do not define it as output.

1.3.10.2 GPIO / AON / INT mapping module to SoC

Table 1-13 shows the GPIO pins that are directly connected from the Intel® Curie™ module to the Intel® Quark™ SE microcontroller.

Please note that only AON pins and timers can wake the module from sleep. See Section 1.3.8, "Wake-capable interrupt pins".

Table 1-13. GPIO / AON / INT

Intel® Curie™ Module Ball No.	Intel® Curie™ Module Ball Name	Intel® Quark™ SE Microcontroller C1000 Signal	Intel® Quark™ SE Microcontroller C1000 Pin
C24	AIN[14]	GPIO_SS[6]	F1
F21	ATP_INT3	GPIO_AON3	F9
F22	ATP_INT0	GPIO_AON0	G9
J2	AIN[10]	GPIO_SS[2]	K5
H21	AIN[11]	GPIO_SS[3]	G1
H2	AIN[12]	GPIO_SS[4]	J4
L2	AIN[13]	GPIO_SS[5]	G2
L3	ATP_INT2	GPIO_AON2	E12
N3	ATP_INT1	GPIO_AON1	E11

1.3.10.3 GPIO multifunction mapping

GPIO pins can be configured for primary or alternate functions. Table 1-14 provides the mapping for the multiple functions each pin can be used for.

Table 1-14. GPIO / multifunction lines

Intel® Curie™ Module Ball No.	Primary Function	Alternate Function 1	Alternate Function 2
A3	I2S_RXD	GPIO[15]	-
A4	SPI0_M_CS1	GPIO[25]	-
B1	I2S_RWS	GPIO[17]	-
B2	I2S_RSCK	GPIO[16]	-
B3	SPI0_M_CS0	GPIO[24]	-
B21	SPI1_M_CS2	GPIO[13]	-
B22	SPI1_M_MISO	GPIO[9]	-
C1	I2S_TWS	GPIO[19]	-
C2	I2S_TSCK	GPIO[18]	-
C3	SPI0_M_CS2	GPIO[26]	-
C4	SPI0_M_SCK	GPIO[21]	-
C21	SPI1_M_SCK	GPIO[8]	-
C22	SPI1_M_CS3	GPIO[14]	-
D1	I2S_TXD	GPIO[20]	-
D3	SPI0_M_MOSI	GPIO[23]	-
D4	SPI0_M_MISO	GPIO[22]	-

Table 1-14. GPIO/multifunction lines (continued)

Intel® Curie™ Module Ball No.	Primary Function	Alternate Function 1	Alternate Function 2
D21	SPI1_M_CS1	GPIO[12]	-
D22	SPI1_M_CS0	GPIO[11]	-
E3	PLT_CLK[0]	GPIO_SS[14]	-
E4	SPI_S_CLK	GPIO[2]	AIN[2]
E22	SPI1_M_MOSI	GPIO[10]	-
F2	PWM3_out	GPIO_SS[13]	-
F3	SPI_S_CS_B	GPIO[0]	AIN[0]
F4	SPI_S_MOSI_B	GPIO[3]	AIN[3]
G1	PWM2_out	GPIO_SS[12]	-
G2	PWM1_out	GPIO_SS[11]	-
G3	SPI_S_MISO	GPIO[1]	AIN[1]
G4	SPI0_SS_CS3	GPIO[30]	-
H1	PWM0_out	GPIO_SS[10]	-
J21	UART1_TX	GPIO_SS[8]	AIN[16]
J22	UART1_CTS_B	GPIO_SS[0]	AIN[8]
K3	SPI0_SS_CS2	GPIO[29]	-
K21	UART1_RTS_B	GPIO_SS[1]	AIN[9]
K22	UART1_RX	GPIO_SS[9]	AIN[17]

1.3.10.4 Internal GPIO mapping

Table 1-15 documents the internal GPIO signals within the Intel® Curie™ module.

Some application software can be designed using SW_FG_VBAT to interrupt the Intel® Quark™ SE microcontroller to control the application external battery charging / protection circuit in addition to the battery charger internal resources. The Intel® Quark™ SE microcontroller can monitor the voltages and send signals to the application circuit to turn on and off the supply voltage (VIN) to the charger circuit.

BLE_SW_CLK signal has a 22 kohm internal pull-down resistor to keep it from floating.

BLE_SWDIO signal has a 22 kohm internal pull-up resistor to keep it from floating.

BLE_SW_CLK and BLE_SWDIO are used with the J-Link emulator to program or debug the Bluetooth® low energy controller. It is also connected to the Intel® Quark™ SE microcontroller for the software to implement the debugging function and programming capability if required by the application.

Table 1-15. Internal GPIO signals

Intel® Curie™ Module Signal (Primary Function)	Intel® Curie™ Module Signal (Secondary Function)	Intel® Quark™ SE Microcontroller C1000 / Component Signal	Intel® Quark™ SE Microcontroller C1000 Pin
5V_BUS_SENSE	GPIO[7]	AIN[7]	G3 VDD_USB (via resistor)
ATP_BLE_INT	GPIO[5]	BLE H4	AIN[5]
BLE_SW_CLK (Intel Curie module PIN D24)	BLE_SW_CLK	GPIO[27]	C10
BLE_SWDIO (Intel Curie module PIN E24)	BLE_SWDIO (BLE PIN J2)	GPIO[6]/AIN[6]	H4
BUCK_EN	GPIO_SS[15]	BUCK CONVERTER Enable PIN B1	PLT_CLK[1]
CHG_STATUS (Intel Curie module Pin M22)	SPIO_SS[7]	AIN[15]	J5
SW_FG_VBATT (Intel Curie module PIN P21)	SW_FG_VBAT	GPIO[4]/AIN[4]	K6
UART0_TXD	GPIO[31]	BLE J6	-
VUSB_EN	GPIO[28]	LDO VUSB PIN3 EN	D10
UART0_RXD	AIN[18]	BLE J7	-

1.3.11 6-Axis sensing device pins

Table 1-17 lists the pins exposed externally from the 6-axis sensing device.

The I^2C interface on the 6-axis sensing device can be connected to a Bosch* BMM150 external magnetometer to make it a 9-axis sensing device.

Refer to the Bosch* BMI160 datasheet for more details.

Table 1-16. Six-axis interface pins

Ball No.	Ball Name	Function
A21	6AXIS_SDA	I^2C data interface to external sensor. Requires external pull-up.
A22	6AXIS_SCL	I^2C clock output to external sensor. Requires external pull-up.
A23	6AXIS_INT2	Auxiliary output pin. Interrupt to external sensor
N/A	6AXIS_INT1	Internal interrupt signal from the 6-axis accelerometer to the Intel Quark processor core

1.3.12 Bluetooth® low energy controller pins

The Intel® Curie™ module includes a Bluetooth® low energy controller that can be controlled externally using the pins in Table 1-17. below.

The debug pins for the integrated Bluetooth® low energy controller are documented in Table 1-9.

Table 1-17. Bluetooth® low energy controller interface pins

Ball No.	Ball Name	Function	Drive (Low / High)
B23	BLE_SDA	I^2C data interface Bluetooth® low energy controller; Can be connected to external master. Requires external pull-up.	Refer to the Nordic* nRf51822 datasheet.
B24	BT_GPIO	GPIO from Bluetooth® low energy (BLE) controller	
F23	BLE_SCL	I^2C clock interface for the Bluetooth® low energy controller chip. Requires external pull-up. Note that this is not supported in software.	
F24	BLE_RF	Bluetooth® antenna connection	
G23	BLE_DEC2	For 3.3 V IO leave unconnected. For 1.8V IO connections to VDD_BLE_SEN. Refer to nrf51822 datasheet for more information	
H24	VDD_BLE_SEN	Bluetooth® low energy controller power supply with internal 0.1 uF capacitor. Refer to the AVDD and VDD power rail in the nRf51822 datasheet for more information.	

1.3.13 I^2C interface pins

I^2C is a multi-master, multi-slave, single-ended serial bus. It is typically used to interface low speed components to the processor. There are four I^2C interfaces available in Intel® Curie™ module: two I^2C interfaces on Intel® Quark™ SE microcontroller processor core and two I^2C interfaces on the ARC* processor core (sensor subsystem). Table 1-18 lists the pins for these interfaces.

Table 1-18. I^2C interface pins

Ball No.	Ball Name	Function	Drive (Low / High)	Internal Pull-Up[1] / Pull-Down
M24	I2C0_SCL	I2C0 clock	4/8 mA	47 kohm
P23	I2C0_SDA	I2C0 data		internal pull-up configuration can be set in device Firmware for slower speed without external resistor
D2	I2C1_SDA	I2C1 data	2/4 mA	
E1	I2C1_SCL	I2C1 clock (maximum of 1MHz)		
N23	I2C0_SS_SDA	I2C0 sensor subsystem data	2/4 mA	
N24	I2C0_SS_SCL	I2C0 sensor subsystem clock (maximum of 400 kHz)		External pull-up resistor needed to operate at higher speed
E2	I2C1_SS_SDA	I2C1 sensor subsystem data		
F1	I2C1_SS_SCL	I2C1 sensor subsystem clock (maximum of 400 kHz)		Refer to I^2C specification

Notes:
1. Internal pull-up resistors are disabled at reset. These I/O's all have typical 47kohm optional internal pull-up that can be enabled by the software.

1.3.14 I^2S interface pins

Table 1-19 lists the pins for the two I^2S bus interfaces on the module. This electrical serial bus interface is used to connect digital audio devices.

The I^2S interfaces on the module have a fixed output sample size of 32 bits. The input sample size ranges from 12–32bits. The data is sent in raw PCM format and has to be processed by the device connected to the port.

Table 1-19. I^2S interface pins

Ball No.	Ball Name	Function	Alternate Function	Drive (Low / High)	Internal Pull-Up[1] / Pull-Down
A3	I2S_RXD	Receive RX data	GPIO[15]	2/4 mA	47 kohm
B2	I2S_RSCK	Receive clock input	GPIO[16]		Internal pull-up configuration can be set in device Firmware for slower speed without external resistor
B1	I2S_RWS	Receive word select	GPIO[17]		
C1	I2S_TWS	Transmit word select	GPIO[19]	4/8 mA	
C2	I2S_TSCK	Transmit clock output	GPIO[18]		
D1	I2S_TXD	Transmit TX data	GPIO[20]		External pull-up resistor needed to operate at maximum speed

Notes:

1. Internal pull-up resistors are disabled at reset. These I/Os all have typical 47 kohm optional internal pull-up that can be enabled by the software.

1.3.15 Pulse width modulator (PWM) pins

Pulse width modulation is a technique used to encode a message into a pulsing signal. The main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. The Intel® Curie™ module has four counters capable of operating in PWM mode. Table 1-20 lists the pins providing the output from these counters.

Table 1-20. PWM output pins

Ball No.	Ball Name	Alternate Function	Function	Drive (Low / High)	Internal Pull-Up[1] / Pull-Down
F2	PWM3_OUT	PWM [3]	GPIO_SS[13]	4/8 mA	47 kohm
G1	PWM2_OUT	PWM [2]	GPIO_SS[12]		
G2	PWM1_OUT	PWM [1]	GPIO_SS[11]		
H1	PWM0_OUT	PWM [0]	GPIO_SS[10]		

Notes:

1. Internal pull-up resistors are disabled at reset. These I/O's all have typical 47 kohm optional internal pull-up that can be enabled by the software.

1.3.16 SPI master pin out

SPI is a synchronous serial communication interface used for communication in embedded platforms. Intel® Curie™ module has three SPI ports available to work in master mode: two SPIs from Intel® Quark™ SE microcontroller processor core and one from the ARC* processor core (sensor subsystem). Table 1-21 lists the pins for these ports, grouped according to the three ports respectively.

Table 1-21. SPI master pins

Ball No.	Ball Name	Source	Primary Function	Alt Function	Drive (Low / High)	Internal Pull-Up[1] / Pull-Down
A4	SPI0_M_CS1	SoC SPI0	chip select 1	GPIO[25]	2/4 mA	47 kohm
B3	SPI0_M_CS0		chip select 0	GPIO[24]		
C3	SPI0_M_CS2		chip select 2	GPIO[26]		
C4	SPI0_M_SCK		clock	GPIO[21]		
D3	SPI0_M_MOSI		data out	GPIO[23]		
D4	SPI0_M_MISO		data in	GPIO[22]		
B21	SPI1_M_CS2	SoC SPI1	chip select 2	GPIO[13]	2/4 mA	
B22	SPI1_M_MISO		data in	GPIO[9]	4/8 mA	
C21	SPI1_M_SCK		clock	GPIO[8]		
C22	SPI1_M_CS3		chip select 3	GPIO[14]	2/4 mA	
D21	SPI1_M_CS1		chip select 1	GPIO[12]		
D22	SPI1_M_CS0		chip select 0	GPIO[11]		
E22	SPI1_M_MOSI		data out	GPIO[10]		
H3	SPI0_SS_CS1	Sensor system	chip select 1	-	4/8 mA	
H4	SPI0_SS_CS0		chip select 0	-		
G4	SPI0_SS_CS3		chip select 3	GPIO[30]	2/4 mA	
J3	SPI0_SS_MISO		data in	-	4/8 mA	
J4	SPI0_SS_SCK		clock	-		
K3	SPI0_SS_CS2		chip select 2	GPIO[29]	2/4 mA	
M3	SPI0_SS_MOSI		data out	-	4/8 mA	

Notes:

1. Internal pull-up resistors are disabled at reset. These I/Os all have typical 47 kohm optional internal pull-up that can be enabled by the software.

1.3.17 SPI slave pin out

The module also has one SPI port available to work in slave mode. Table 1-22 lists the pins for that port.

Table 1-22. SPI slave pins

Ball No.	Ball Name	Primary Function	Alt Function 1	Alt Function 2	Drive (Low / High)	Internal Pull-Up[1] / Pull-Down
E4	ATP_SPI_S_SCK	clock	GPIO[2]	AIN[2]	4/8 mA	47 kohm
F3	ATP_SPI_S_CS	chip select	GPIO[0]	AIN[0]		
F4	ATP_SPI_S_MOSI	data in	GPIO[3]	AIN[3]		
G3	ATP_SPI_S_MISO	data out	GPIO[1]	AIN[1]		

Notes:
1. Internal pull-up resistors are disabled at reset. These I/Os all have typical 47 kohm optional internal pull-up that can be enabled by the software.

1.3.18 UART interface pins

The module has two UART interfaces for asynchronous serial communication. The data format and transmission speeds on this interface are configurable. Only one UART interface is available for the user, the other UART interface is dedicated to the Bluetooth® low energy controller inside the module. Table 1-23 lists the pins for the user-facing interface.

Table 1-23. UART Interface pins

Ball No.	Ball Name	Primary Function	Alt Function 1	Alt Function 2	Drive (Low / High)	Internal Pull-Up[1] / Pull-Down
J21	UART1_TX	Transmit Data	GPIO_SS[8]	AIN[16]	4/8 mA	47 kohm pull-up
J22	UART1_CTS	flow control	GPIO_SS[0]	AIN[8]		
K21	UART1_RTS	flow control	GPIO_SS[1]	AIN[9]		
K22	UART1_RX	Receive Data	GPIO_SS[9]	AIN[17]		

Notes:
1. Internal pull-up resistors are disabled at reset. These I/Os all have typical 47 kohm optional internal pull-up that can be enabled by the software.

1.3.19 USB interface pins

USB is an industry standard used to connect peripheral devices or any pair of devices together through the USB port in order to communicate and/or supply power. the Intel® Curie™ module has a single USB 1.1 device port. Table 1-24 lists the pins for the USB port.

Table 1-24. USB Interface pins

Ball No.	Ball Name	Primary Function
J1	GROUND also known as VSS_USB and USB_VSS	Ground connection with external USB source
J23	USB_DM	USB Data Minus (-) bidirectional signal
J24	USB_DP	USB Data Positive (+) bidirectional signal
K4	VDD_USB	5 V DC used to sense from a power host and used to power the USB transceiver.

1.3.20 ADC pins

The module offers an analog-to-digital converter. Table 1-25 lists the Intel® Curie™ module external pins connected to Intel® Quark™ SE microcontroller ADC interface pins and Table 1-26 provides a list of Intel® Quark™ SE microcontroller ADC interface multifunction pins connected to Intel® Curie™ module external pins.

Table 1-25. Intel® Curie™ module external pins connected to Intel® Quark™ SE microcontroller ADC interface pins

Intel® Curie™ Module Ball No.	Analog Input Signal Name	Intel® Quark™ SE Microcontroller C1000 Ball No.
C24	AIN_14	F1
H2	AIN_12	G1
H21	AIN_11	G2
J2	AIN_10	J4
L2	AIN_13	K5

Table 1-26. Intel® Quark™ SE microcontroller ADC interface multifunction pins connected to Intel® Curie™ module external pins

Intel® Curie™ Module Ball No. (Intel® Quark™ SE Microcontroller C1000 Ball No.)	Optional Functions
E4	GPIO[2]/AIN[2]/SPI_S_SCK
E24 (Intel® Quark™ SE microcontroller pin H4)	GPIO[6]/AIN[6]/Intel® Quark™ SE microcontroller_SWDIO
F3	GPIO[0]/AIN[0]/SPI_S_CS_B
F4	GPIO[3]/AIN[3]/SPI_S_MOSI
G3	SPI_S_MISO/GPIO[1]/AIN[1]
J22 (Intel® Quark™ SE microcontroller pin L5)	GPIO_SS[0]/AIN[8]/UART1_CTS_B/
K4 (VDD_USB, Intel® Quark™ SE microcontroller pin G3)	GPIO[7]/5V_BUS_SENSE/AIN[7]
K21 (Intel® Quark™ SE microcontroller pin M5)	GPIO_SS[1]/AIN[9]/UART1_RTS_B
M22 (Intel® Quark™ SE microcontroller pin J5)	CHG_STATUS/GPIO_SS[7]/AIN[15]
P21 (Intel® Quark™ SE microcontroller pin K6)	GPIO[4]/SW_FG_VBATT/AIN[4]

2 Specifications

2.1 Electrical characteristics

This section covers the absolute maximum ratings, operating maximum and minimum voltages, and the DC operating specifications.

2.1.1 Absolute maximum and minimum voltages

The absolute maximum and minimum specifications are used to specify conditions allowable outside of the functional limits of the Intel® Curie™ module, but with possibly reduced life expectancy once returned to functional limits. At conditions exceeding absolute specifications, neither functionality nor long term reliability can be expected. Parts may not function at all once returned to functional limits.

Table 2-1 shows the absolute ranges for the Intel® Quark™ SE microcontroller C1000 integrated in the module.

Caution: Although the module contains protective circuitry to resist damage from electrostatic discharge (ESD), always take precautions to avoid high static voltages or electric fields. See Section 2.1.5 for ESD data.

Table 2-1. Absolute min-max specifications for the Intel® Quark™ SE microcontroller C1000

Intel® Curie™ Module Signal	Function	Intel® Quark™ SE Microcontroller C1000 Mapped Signal	Range
VDD_USB	USB input voltage	VUSB	–0.3 V to 3.63 V
VSYS	Supply voltage	VCC_BATT_OPM_3P7	–0.3 V to 4.4 V
AIN	Analog input voltage	VCC_ADC	–0.3 V to 3.63 V

2.1.2 Operating maximum and minimum voltages

Table 2-2 provides the operating ranges for the Intel® Curie™ module.

Table 2-2. Operating Min - Max specifications for the Intel® Curie™ module

Ball	Function	Power I/O	Min	Typ	Max	Unit
VDD_PLAT_1P8	3.3 V for platform devices	-	1.62	1.8	1.98	V
VDD_PLAT_3P3	1.8 V for platform devices	-	2.97	3.30	3.63	V
VDD_HOST_1P8	1.8 V for host blocks	-	1.62	1.8	1.98	V
VDD_USB	USB power	-	3.5	5.0	5.25	V
VIN	Charging DC input	-	4.45	5.0	6.45	V
VSYS[1]	Main DC input power	-	2.1	-	4.4	V

Notes:
1. VSYS should be greater than 3.3 V if the module VDD_PLAT_3P3 regulator is used.

2.1.3 DC operating specifications

Table 2-3 through Table 2-8 include the DC operating specifications for Intel® Curie™ module.

2.1.3.1 DC specifications for I/O

Table 2-3 provides the voltage ranges for AON_IO_VCC at 3.3 V, while Table 2-4 provides the voltage ranges for AON_IO_VCC at 1.8 V.

Table 2-3. AON_IO_VCC=3.3 VDC

Symbol	Parameter	Min	Typ	Max	Unit
VIL	Input low voltage	-	-	0.8	V
VIH	Input high voltage	2	-		V
VOL	Output low voltage	-	-	0.4	V
VOH	Output high voltage	2.4	-	-	V
IOL	2 mA @ VOL	2.4	3.8	5.3	mA
	4 mA @ VOL	4.7	7.6	10.6	mA
	8 mA @ VOL	9.4	15.3	21.2	mA
IOH	2 mA @ VOH	3.4	7.0	11.6	mA
	4 mA @ VOH	6.9	14.0	23.2	mA
	8 mA @ VOH	13.8	27.9	46.4	mA
RPU	Pull-up resistor	34K	49K	74K	ohm
VT	Threshold point	1.33	1.4	1.47	V
VT+	L-> H threshold point	1.53	1.6	1.66	V
VT-	H-> L threshold point	1.13	1.2	1.27	V

Table 2-4. AON_IO_VCC=1.8V

Symbol	Parameter	Min	Typ	Max	Unit
VIL	Input low voltage	-	-	0.63	V
VIH	Input high voltage	1.17	-	-	V
VOL	Output low voltage	-	-	0.45	V
VOH	Output high voltage	1.35	-	-	V
IOL	2 mA @ VOL	1.0	2.0	3.6	mA
	4 mA @ VOL	1.9	4.0	7.2	mA
	8 mA @ VOL	3.9	8.1	14.4	mA
IOH	2 mA @ VOH	0.8	2.0	4.1	mA
	4 mA @ VOH	1.6	4.0	8.1	mA
	8 mA @ VOH	3.2	8.0	16.2	mA
RPU	Pull-up resistor	34K	49K	74K	ohm
VT	Threshold point	0.82	0.89	0.93	V
VT+	L-> H threshold point	0.99	1.07	1.12	V
VT-	H-> L threshold point	0.62	0.69	0.77	V

2.1.3.2 ADC - DC I/O specifications

Table 2-5 provides the voltage ranges for the analog-to-digital converter.

Note: The parasitic capacitance of the Intel® Quark™ SE microcontroller and the package are TBD. We recommend adding 10 pF to the calculations for input capacitor charging.

Table 2-5. ADC - DC I/O specifications

Symbol	Parameter	Min	Typ	Max	Unit
AIN	Full-scale input range	0	-	≤ADC_3P3_VCC ≤AON_IO_VCC	V
AGNDREF	Negative reference voltage	0	0	0.1	V
ADC_cap	Input sampling capacitance	-	5	-	pF

2.1.3.3 Comparator voltage specification

Table 2-6 provides the voltage ranges for the comparator.

Table 2-6. Comparator voltage range

Symbol	Parameter	Min	Typ	Max	Unit
COMP_AREF	External reference voltage	0.0	-	≤ CMP_3P3_VCC	V
AIN	Full scale input voltage	0.0	-	≤ CMP_3P3_VCC ≤ AON_IO_VCC	V
CMP_3P3_VCC	Comparator power	2.0	-	3.6	V

2.1.3.4 USB I/O - DC specifications

Table 2-7 provides the DC specifications for the USB I/O.

Note: Refer to the Intel® Quark™ SE microcontroller datasheet for the complete electrical specifications.

Table 2-7. USB I/O - DC specifications

Symbol	Parameter	Min	Typ	Max	Unit
VIL	Input low voltage	-	-	0.8	V
VIH	Input voltage high	2.0	-	-	V
VOL	Output low voltage	-	-	0.3	V
VOH	Output high voltage	2.8	-	-	V
VCRS	Differential output signal cross-point	1.3	-	2.0	V
VCM	Common mode range	0.8	-	2.5	V
RPU	External pull-up resistor	1.425	-	1.575	kohm
Vtrm	Termination voltage	3.0	-	3.6	V

De La Salle University

2.1.3.5 USB I/O - AC specifications

Table 2-8 provides the AC specifications for the USB I/O.

Table 2-8. USB IO AC specifications

Symbol	Parameter	Min	Typ	Max	Unit
TFR	Rise time 50 pF	4	-	20	ns
TFF	Fall time 50 pF	4	-	20	ns
TFRFF	Rise/fall matching	90	-	111.11	%

2.1.4 Operating temperature range

Operating temperature range: -25°C to +70°C.

Charging: 0°C to 60°C when using the internal charger.

2.1.5 ESD considerations

Table 2-9 shows the ESD considerations for the Intel® Curie™ module.

Note: A reduction in either the HBM or CDM goal does not require additional design changes beyond the actual ESD guidance.

Table 2-9. Intel® Curie™ module component ESD data

Symbol	Description	Stress Conditions
HBM	ESD - Human Body Model - JS-001-2014	±1000 V
CDM	ESD - Charge Device Model - JESD22-C101	±500 V

2.2 Power and timing considerations

This section covers all the aspects of power consideration for the Intel® Curie™ module.

2.2.1 Power architecture

The power architecture is a power generation solution that provides both the internal – host and always on (AON) – and external (platform) supply rails. Some of these supply voltages can be internally or externally turned off to reduce the power consumption for the application.

The Intel® Curie™ module internal power supply resources are limited and can only be used in some applications. Applications which require higher power can use the application power converter circuits to provide their power needs. If any of these voltage converters are not used, the application firmware and hardware need to disable the unused voltage converters to reduce the leakage and improve the power consumption.

Also look at the third-party switching supplies and make sure their minimum output load is met otherwise their output voltage regulation may not be stable.

2.2.2 Primary power

The Intel® Curie™ module is designed to operate under battery-operated systems using 3.3 V or 1.8 V interfaces, or may be powered by external regulators providing 1.8 V or 3.3 V. The module has regulators available to power internal circuits or for use by platform applications. They are documented in this section and in Section 2.2.3, "Optional platform power".

An integrated power supervisor holds the system in reset when the input voltage drops below 2.9 V. Intel recommends ensuring that the OPM2P6_VOUT node (2.6 V reference voltage output; Ball Number J1) is discharged to ground (zero volt) before a power-up cycle. An external power supervisor may be required. Refer to the *Intel® Curie™ Power Sequence Considerations* application note for important information when designing your product.

Intel® Curie™ module has an 1.8 V power regulator (labeled ESR3) which must be used to power internal circuits.

The power regulators require an inductor and a bulk capacitor as a part of the switching regulator function. Table 2-10 provides the values of the inductors and the capacitors for the maximum power delivery for ESR3, as well as the optional regulators documented in Section 2.2.3, "Optional platform power". Table 2-11 shows the maximum current range.

Other core voltages include:

- VDD_BLE_SEN, used to supply voltage to the internal Bluetooth® low energy controller and the 6-axis sensing device.
- AON_IO_VCC, used to set the supply voltage to the IO level for the peripheral interfaces. The choice is 1.8 V or 3.3 V level.

2.2.3 Optional platform power

Intel® Curie™ module also has 1.8 V and 3.3 V regulators available for use by platform applications. These internal LDO and converters are optional to use and the application can provide their own converter or power source to meet their requirements.

- ESR1 provides the platform 3.3 V, which can be used by external platform devices.
- ESR2 provides the platform 1.8 V, which can be used by external platform devices.

The power regulators require an inductor and a bulk capacitor as a part of the switching regulator function. Table 2-10 provides the value of the inductor and the capacitor for the maximum power delivery for all the power regulators. Table 2-11 shows the maximum current range.

On-module regulators supply the integrated components while external devices need their own power system.

Note: Exceeding the maximum input specifications or using non-compliant USB chargers damages Intel® Curie™ module input regulators.

De La Salle University

Table 2-10. ESR requirements

Parameter	Description	Tolerance Min	Typical	Tolerance Max	Unit
3.3V Platform rail requirements					
C_ESR1	Decoupling capacitor	-20%	8.2	+20%	µF
L_ESR1	Inductor	-30%	10	+30%	µH
1.8V Platform rail requirements					
C_ESR2	Decoupling capacitor	-20%	4.7	+20%	µF
L_ESR2	Inductor	-30%	22	+30%	µH
1.8V Host rail requirements					
C_ESR3	Decoupling capacitor	-20%	4.7	+20%	µF
L_ESR3 -	Inductor	-30%	22	+30%	µH

Table 2-11. Maximum current range

Power Rails from Intel® Curie™ Module	Description	Maximum Current (mA)
VDD_PLAT_3P3	ESR1 - 3.3 V for platform devices (from Intel® Quark™ SE microcontroller SoC)	150
VDD_PLAT_1P8	ESR2 - 1.8 V for platform device (from Intel® Quark™ SE microcontroller SoC)	100
VDD_HOST_1P8	ESR3 - 1.8 V for Host Blocks (from Intel® Quark™ SE microcontroller SoC). Internal switching regulator 1.8v supply that can be used to power the cores. Additional inductor and capacitor is required.	100
BUCK_VOUT*	1.8 V/3.3 V for platform devices. (from internal buck)	300

2.2.4 Device power states

2.2.4.1 Off state

In the Off state, all the voltage rails are disabled and no clocks are running.

The Off state is entered if the main power is removed or if the power is at an insufficient level to power the device.

When the power is applied, the device exits the Off state and transitions to Active.

2.2.4.2 Sleep state

Sleep is a suspend state in which:

- The core voltage rail and core clock are turned off, RTC clock is running.
- The always-on voltage rail remains powered in this state.
- The 3P3 and 1P8 voltage rails can be individually configured to power down for energy savings.
- A wake event via an enabled comparator, AON GPIO, AON timers or RTC event is required to exit the Sleep state.
- The contents of SRAM can optionally be retained during the Sleep state
- Peripherals within the module can remain active when the SoC is in a sleep state.

- GPIOs can be configured to go in retention mode to hold their state while the core is in sleep mode.
- Refer to the diagram "SoC power states" in the Intel® Quark™ SE microcontroller Datasheet under power management.

2.2.4.3 Active state

- The Active state is the normal operating state of the SoC:
  - The core clock and RTC clock are running.
  - The Core (1P8) and AON voltage rails are enabled.
  - Applications can select if the 3P3 voltage rail is energized.
- Within the Active state:
  - The host processor can transition into and out of various C-states.
  - The sensor subsystem can transition into and out of various sensing states.
  - The SoC peripherals can be disabled or clock-gated for additional power savings.

2.2.5 Intel® Quark™ SE microcontroller C1000 power states

The Intel Quark microcontroller and the ARC* processor cores can run at a lower internal clock, if acceptable to application / design intent, for additional power savings that can be further enhanced with proper use of sleep states and wake events.

Refer to the Intel® Quark™ SE Microcontroller C1000 Datasheet for detailed information.

2.2.6 Boot and reset sequences

2.2.6.1 Power-up - Off to Active

When VCC_BATT_V3P7 is applied by means of an external battery or other power source, the Intel® Curie™ module can power up with any internal or external software-enabled power converter that is used in the application design.

LDO 1.8V NCP170 is enabled by the VCC_AON_1P8 rail when it reaches the logic high threshold.

TPS62743 buck regulator (1.8V / 3.3V) is enabled by the AON_IO_VCC rail when it reaches the logic high threshold if the software disables BUCK_EN by tri-stating GPIO_SS[15]. Or the software can set GPIO_SS[15] = 1 to enable or GPIO_SS[15] = 0 to disable the bulk regulator.

MIC5504-3.3YMT LDO (3.3v) is enabled by the VUSB_EN signal that is 61% of the 5 V USB supply voltage when the USB cable is plugged. The software can then be configured to get an interrupt and the software can set GPIO[28] = 1 to enable the LDO for the USB interface. The software can disable the LDO by setting GPIO[28] = 0.

2.2.6.2 Power-up sequence timings and thresholds

The power sequence and timings given in this section are representative of typical values measured.

All rails except VSYS and AON_IO_VCC are outputs.

VIN is the system power supply. AON_IO_VCC has to be supplied externally. It is recommended to use VCC_AON_PWR to power the AON_IO_VCC. If it is fed by any other source, please make sure that the timings are met.

Note: Refer to the *Intel® Curie™ Power Sequence Considerations* application note for important information when designing your product.

Refer to *Intel® Quark™ SE Microcontroller C1000 Datasheet* for power-up sequence timing parameters and power architecture section.

The Intel® Curie™ module includes a power supervisor shown in Figure 2-1. This holds off the boot up sequence until VCC_HOST_1P8 voltage is within specification.

Figure 2-2 below presents the power state change diagram, while Figure 2-3 illustrates the power rail timing sequence and Table 2-12 details the power-up sequence parameters.

Figure 2-1. Power supervisor

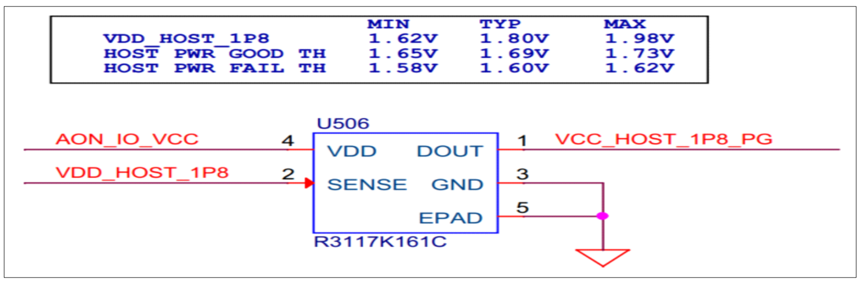


Figure 2-2. Power state change diagram

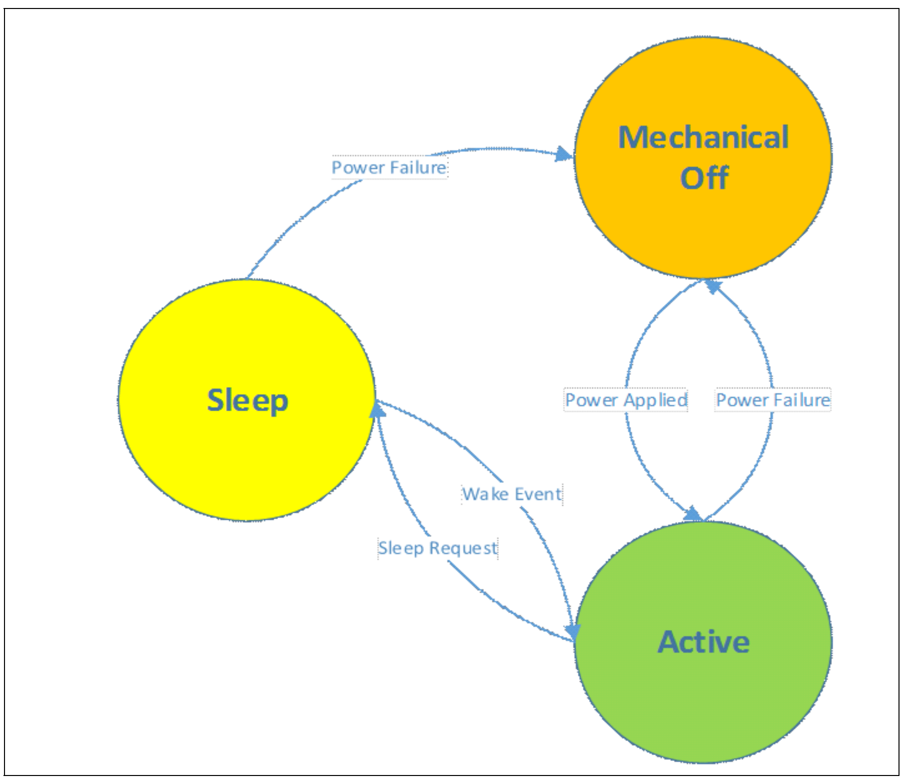


Figure 2-3. Power rail timing sequence

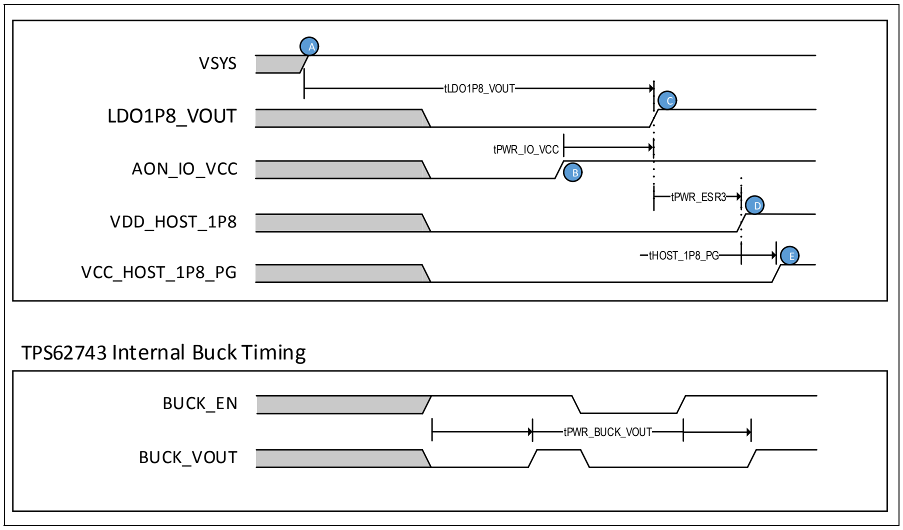


Table 2-12. Power-up sequence parameters

Parameter	Description	Min	Typ	Max	Unit
tLDO1P8_VOUT	Time interval between the VSYS and the availability of LDO1P8_VOUT	-	6.2	-	ms
tPRW_ESR3	Time interval for eSR3 starts regulate the output voltage when VCCOUT_AON_1P8 reaches threshold (1.55V).	-	925	-	µs
tPWR_IO_VCC	Time interval for LDO properly regulate output voltage when Enable asserted.	0	-	-	ms
tHOST_1P8_PG	Time interval for buck regulator properly regulate output voltage when Enable asserted.	-	100	-	µs
tPWR_BUCK_VOUT	Time interval for buck regulator (TPS62743) properly regulate output voltage when Enable (BUCK_EN) asserted.	-	10	25	ms

Waveforms from lab measurements are provided in Figure 2-4 and Figure 2-5 below as reference:

Figure 2-4. Timing relationship between VIN, OPM2P6_VOUTand LDO1P8_VOUT

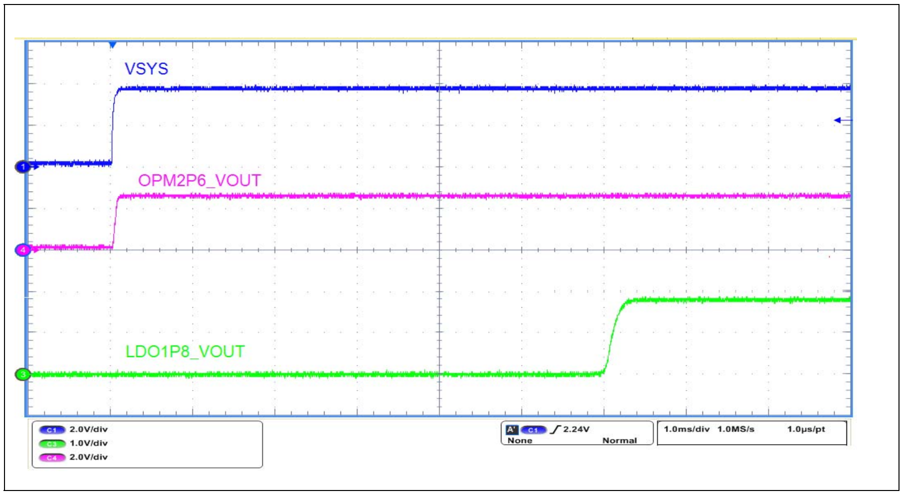
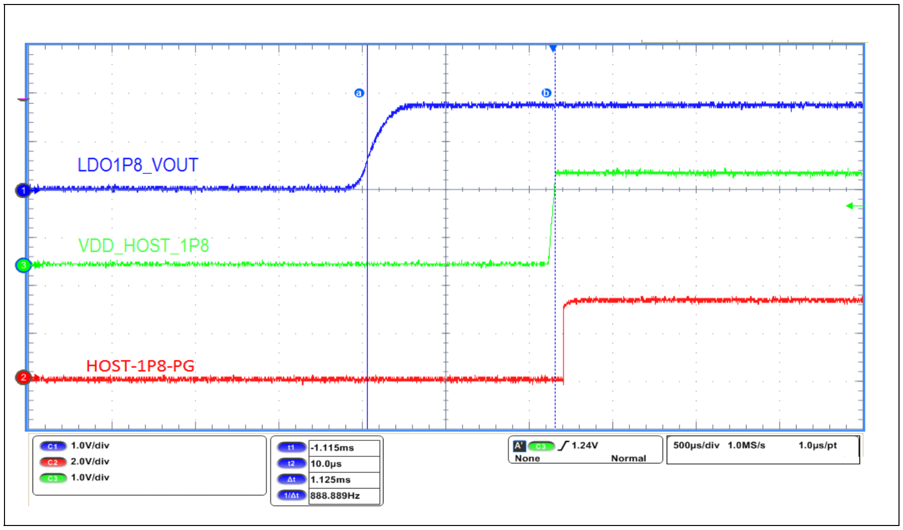


Figure 2-5. Timing relationship between LDO1P8_VOUT, VDD_HOST_1P8 and VDD_HOST_1P8-PG



2.2.7 Platform power distribution

Table 2-13 documents the maximum current output for each power rail.

Table 2-13. Power rail maximum current output

Power Rail	Sources	Maximum Current
1.9VDC to 4.4V RAIL		
VCC_BATT_OPM_3P7	Battery	300ua
VCC_BATT_ESR1_3P7	Battery	250mA
VCC_BATT_ESR2_3P7	Battery	125mA
VCC_BATT_ESR3_3P7	Battery	125mA
1.8VDC RAIL		
VCC_AON_1P8	Internal LDO in Intel® Curie™ module	150mA
VCC_HOST_1P8	ESR3	50mA (Preliminary)
VCC_SRAM_1P8	Internal LDO within Intel® Curie™ module	4mA
VCC_PLL_1P8	ESR3	2.2mA
VCC_RTC_1P8	Internal LDO within Intel® Curie™ module	125nA
3.3V RAIL		
ADC_3P3_VCC	Dedicated ADC LDO	1mA
VCC_CMP_3P3	ESR1/OPM_2P6/Dedicated Comp LDO	60uA
VCC_USB_3P3	Dedicated LDO for USB	1mA
VDD_BLE_SEN	BUCK_OUT	26.75mA

2.2.8 Current draw (typical)

The data provided in this section was measured on a design-development platform and does not include components outside the Intel® Curie™ module. Results shown are averages that are highly influenced by software configuration choices.

2.2.8.1 Module current consumption

Table 2-14 through Table 2-18 provide data on Intel® Curie™ module current consumption.

Table 2-14. Intel® Curie™ module power consumption: idle - without motion sensing

State	Value	Unit	Description
Intel® Curie™ module Idle - No motion sensing	29.62	uA	-
Peripherals breakdown			No motion sensing and No Bluetooth® low energy activity SoC Idle (Deep Sleep + WDT wakeup every 5s) Nordic* Bluetooth® low energy Idle BMI160 Accelerometer and Gyrometer suspend
Intel® Quark™ SE microcontroller Deep Sleep (RTC ON/SRAM/IO retention)	18.7	uA	
Intel® Quark™ SE microcontroller watchdog wake every 33.5s	1		
Bluetooth® low energy Idle - Low power mode	2.55		
Bosch* BMI160 (Accel / Gyro in suspend mode)	1.87		
Intel® Curie™ module hardware floor current	5.5		

Table 2-15. Intel® Curie™ module power consumption: idle - motion sensing without movement

State	Value	Unit	Description
Intel® Curie™ module Idle - Motion sensing without any move	54.13	uA	-
Peripherals breakdown			No motion sensing and No Bluetooth® low energy activity SoC Idle (Deep Sleep + WDT wakeup every 5s) Nordic* Bluetooth® low energy Idle BMI160 Accelerometer low power & Gyrometer suspend
Intel® Quark™ SE microcontroller Deep Sleep (RTC ON/SRAM/IO retention/External IRQ Wakeup	18.7	uA	
Intel® Quark™ SE microcontroller watchdog wake every 33.5s	1		
Bluetooth® low energy Idle - Low power mode	2.36		
Bosch* BMI160 (Accel ODR 100Hz@AVG1 / Gyro in suspend mode)	26.57		
Intel® Curie™ module hardware floor current	5.5		

Table 2-16. Intel® Curie™ module power consumption: Bluetooth® low energy fast advertising at 100ms - without motion sensing

State	Value	Unit	Description
Intel® Curie™ module Bluetooth® low energy Fast Advertising @100ms - No motion sensing	153.67	uA	-
Peripherals breakdown			No motion sensing - BMI160 Accelerometer & Gyrometer suspend SoC Idle (Deep Sleep + WDT wakeup every 5s) Nordic* Bluetooth® low energy advertising every 100ms; in Bluetooth® low energy Low power mode rest of the time
Intel® Quark™ SE microcontroller Deep Sleep (RTC ON/SRAM/IO retention/External IRQ Wakeup)	18.7	uA	
Intel® Quark™ SE microcontroller watchdog wake every 33.5s	1		
Bluetooth® low energy Fast Adv 100ms + Low power mode	126.6		
Bosch* BMI160 (Accel / Gyro in suspend mode)	1.87		
Intel® Curie™ module hardware floor current	5.5		

Table 2-17. Intel® Curie™ module power consumption: Bluetooth® low energy connection at 150 ms - without motion sensing

State	Value	Unit	Description
Intel® Curie™ module Bluetooth® low energy Connection @150ms - No motion sensing	57.77	uA	-
Peripherals breakdown			No motion sensing - BMI160 Accelerometer & Gyrometer suspend
Intel® Quark™ SE microcontroller Deep Sleep (RTC ON/SRAM/IO retention/External IRQ Wakeup)	18.7	uA	SoC Idle (Deep Sleep + WDT wakeup every 5s) Nordic* Bluetooth® low energy advertising every 100ms; in Bluetooth® low energy Low power mode rest of the time
Intel® Quark™ SE microcontroller watchdog wake every 33.5s	1		
Bluetooth® low energy Fast Adv 100ms + Low power mode	30.7		
Bosch* BMI160 (Accel / Gyro in suspend mode)	1.87		
Intel® Curie™ module hardware floor current	5.5		

Table 2-18. Dhrystone 2.1 results: without motion sensing

State	Value	Unit	Description
Intel® Curie™ module Bluetooth® low energy Connection @150ms - No motion sensing	36.01	mA	-
Peripherals breakdown			No motion sensing - BMI160 Accelerometer & Gyrometer suspend
Intel® Quark™ SE microcontroller LTM C0 Dhrystone 2.1 ARC SS0 Dhrystone 2.1 with all peripherals clock gated ON	36	mA	SoC Idle (Deep Sleep + WDT wakeup every 5s) Nordic* Bluetooth® low energy advertising every 100ms; in Bluetooth® low energy Low power mode rest of the time
Bluetooth® low energy Idle - Low power mode	2.36	uA	
Bosch* BMI160 (Accel / Gyro in suspend mode)	1.87		
Intel® Curie™ module hardware floor current	5.5		

2.3 Clocking

The Intel® Curie™ module contains several clock sources which are described in this section. Accuracy is specified over the operating temperature range for the Intel® Curie™ module.

Refer to the chapter on *Clocking* in the *Intel® Quark™ SE Microcontroller C1000 Datasheet*, for further details on the SoC clocks mentioned in Table 2-19.

Table 2-19. Clock sources

Clock	Use	Accuracy
32kHz[1]	Always on timer	± 5ppm
SoC Silicon Oscillator	Enabled at boot time	± 20,000ppm
SoC XTAL Oscillator 4/8/16/32MHz[2]	Can be enabled by software	± 30ppm
16MHz Bluetooth® low energy clock	Active during BLE transmit/receive	± 30ppm

Notes:
1. This is a MEMS based temperature compensated clock that feeds both the AP SoC as well as the Nordic BLE controller. It is always running when power is applied. Primary use is for real time clock and software timer.
2. Must be used to meet UART timing

2.4 BLE (Bluetooth® low energy)

The Bluetooth® low energy controller in the Intel® Curie™ module provides BLE connectivity with the following performance metrics:

- -93 dBm sensitivity
- 250 kbps, 1 Mbps, and 2 Mbps supported data rates
- -20 to +4 dBm TX power output in 4 dB steps
- -30 dBm in whisper mode
- RSSI reporting with 1 dB resolution
- BT certified 4.1 LE stack

3 Detailed Description

The Intel® Curie™ module is an advanced device built around the Intel® Quark™ SE microcontroller C1000 processor core. It integrates compute, sense, and awareness functionality, as well as connectivity and a programmable input/output controller within a common package.[1]

The Intel® Curie™ module is a tiny hardware product offering design flexibility. The complete, low-power solution comes with compute, motion sensors, Bluetooth® low energy, battery-charging, and pattern matching capabilities for optimized analysis of sensor data, enabling quick and easy identification of actions and motions. The module is packaged into a very small form factor and runs a new software platform created specifically for the Intel® Curie™ module.

Powered by the Intel® Quark™ SE Microcontroller C1000, the Intel® Curie™ module is extremely power efficient and ideal for "always-on" applications such as health and wellness, social notification and sports activities. The Intel Quark microcontroller integrates a pattern classification engine that allows it to identify different motions and activities quickly and accurately.

The chapter includes the following sections:

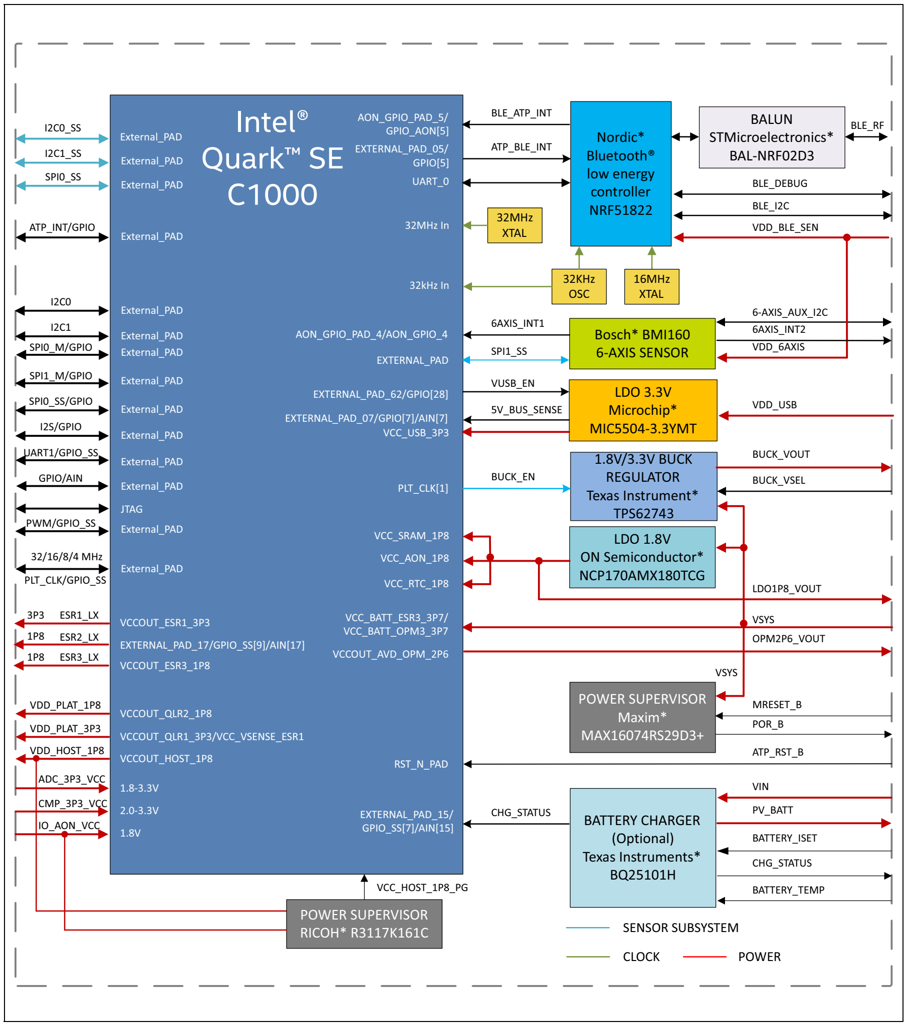
- Block diagram
- Features
- Bluetooth® low energy controller
- Sensor device
- Memory

1. Note that this module is not a FCC-certified module. Emission testing has been performed and designs based on the Intel® Curie™ module that have FCC certification are available, but all new designs based on the module require regulatory approval prior to public availability.

3.1 Block diagram

Figure 3-1 depicts the main functional blocks and discrete devices within the Intel® Curie™ module.

Figure 3-1. Intel® Curie™ module block diagram



3.2 Features

This section lists the features of the components and interfaces included in Intel® Curie™ module.

3.2.1 Processor cores

The Intel® Curie™ module integrates an Intel® Quark™ SE microcontroller C1000 processor core and an ARC* EM4 DSP core-based sensor subsystem running on:

- 32-bit processor with 32-bit data bus
- 32 MHz clock frequency
- 32-bit address bus

3.2.1.1 Intel® Quark™ SE microcontroller C1000 processor core

- Pentium x86 ISA-compatible without x87 floating point unit
- 8 kB L1 instruction cache
- Low-latency data tightly-coupled memory (TCM) interface to on-die SRAM
- Integrated local APIC and I/O APIC

The *Intel® Quark™ SE Microcontroller C1000 Datasheet* provides additional details on the core processor.

3.2.1.2 ARC* EM4 DSP core-based sensor subsystem

The Intel® Curie™ module contains an ARC* EM4 DSP-based sensor subsystem and interrupt controller with the following features:

- 8 kB L1 instruction cache and 8 kB of closely coupled memory for data
- Four counters that can be used in PWM or timer mode. The timer mode supports 32-bit operation at 32 MHz granularity. These timers can be configured and used by both cores.
- One configuration watchdog timer with support to trigger an interrupt and/or a system reset upon timeout. This timer can be configured and used by both cores.

3.2.2 Memory subsystem

- 384 kB of on-die flash
- 80 kB of on-die SRAM

3.2.3 Six-axis accelerometer / gyroscope

The Intel® Curie™ module includes a Bosch* BMI160 six-axis sensing device connected to the SPI1_SS interface port; only the ARC* processor core can communicate with this logical block.

The key features of the Bosch BMI160 integrated device include:

- SPI interface from the ARC processor core to the six-axis sensor to configure and read sensor data.

- GPIO_AON[4] is used to receive interrupt from the six-axis sensor when data is available or when an error condition occurs. The ARC core can be configured to receive this interrupt to process the information.
- Hardware synchronization of inertial sensor data
- Available I^2C from the Bosch* BMI160 6-axis sensing device to interface with compatible, external geomagnetic / magnetometer devices.
- 16-bit digital, tri-axial accelerometer
- 16-bit digital, tri-axial gyroscope
- Ability to average sampled data for more accuracy and improve the ARC core performance by reducing its processing time.
- Sleep or standby mode to support low-power applications
- Separate power supply for the sensor block, allowing the application to turn it off and on as needed to reduce the power consumption.

Note: Refer to Bosch* BMI160 datasheet for more detailed information.

3.2.4 Bluetooth® low energy integration

The Intel® Curie™ module includes a Nordic* nRF51822 that is interfaced to the Intel® Quark™ SE microcontroller C1000 core via UART0.

All the low-level Bluetooth® low energy controller operations are handled by the Bluetooth low energy stack in the device. This makes the application implementation easier by treating the device as a modem and not worry about affecting all the Bluetooth® low energy low level activity because of the application code.

Features of this integrated device include:

- A 32-bit processor with AES hardware encryption
- 2.4 GHz transceiver
- Firmware update via DFU boot-loader, JTAG interface and Over-the-Air (OTA) methods
- Nordic S130 SoftDevice* software that supports Bluetooth 4.1 services and the Gazell™ protocol stack at the same time.
- Low-power sleep mode when not transmitting nor receiving messages. The stack can be configured in this mode without the application real time involvement saving processor real time cycles.
- The Bluetooth® low energy controller can receive or transmit messages while in sleep state, waking the Intel® Quark™ SE microcontroller when done and is also able to interrupt the core via GPIO_AON[5].
- The Bluetooth® low energy blocks are powered separately, allowing the application to turn it off and on as needed to reduce the power consumption. Ensure that there is no leakage or conflict between the blocks when one section is powered down with the others being powered up.

3.2.5 Pattern matching engine

The pattern matching engine (PME) is capable of learning and recognizing patterns in arbitrary sets of data. It is a parallel data recognition engine with the following features:

- 128 parallel arithmetic units (Processing Element or PE) with 8-bit features per PE
- Two pattern matching algorithms:
  - K Nearest Neighbors (KNN)
  - Radial Basis Function (RBF)
- Two distance evaluation norms with 16-bit resolution:
  - L1 norm (Manhattan Distance)
  - Lsup (Supremum) norm (Chebyshev Distance)
- Constant recognition time
- Vector data: up to 128 bytes
- Classification states:
  - ID - identified, only one category matches
  - UNC - uncertain, more than one category matches
  - UNK- unknown, no match
- Support for up to 32,768 categories
- Support for up to 127 contexts
- Supervised learning
- Save and restore network knowledge
- Three main operations supported:
  - Recognize a vector
  - Save the knowledge base from the network
  - Load the knowledge base to the network

3.2.6 USB device

- Single USB 1.1 device port
- Supports full speed (12 Mbps) operation.
- UART mode profile support
- Core detection of USB connected state via a comparator on GPIO[7]/AIN[7] (USB supply VDD_USB is connected to this interrupt).
  - AIN[7] is the interrupt line that will notify the Intel® Quark™ SE microcontroller that a USB voltage is present and thus enable the LDO via firmware. This feature only works with 5V USB sources, it will not detect a 3.3V USB source connection.
- The regulated USB 3.3V LDO supply is software controllable by setting (VUSB_EN) GPIO[28]=0 to disable or =1 to enable.

3.2.7 I^{2}C

- Four I^{2}C master interfaces: two on the Intel® Quark™ SE microcontroller processor core and two on the sensor subsystem (ARC* processor core)
- Three I^{2}C speeds supported:
  - Standard mode (100 kbps)
  - Fast mode (400 kbps)
  - Fast mode Plus (1 Mbps) supported for the two I^{2}C on the Intel® Quark™ SE microcontroller processor core only
  - These can also be configured as slave with a maximum data rate of 400 kbps for the ARC-only I^{2}C
- Support for both 7-bit and 10-bit addressing modes
- Support for 8-entry transmit and 8-entry receive FIFO
- Support for hardware DMA that allows data transfer without CPU involvement
- Support for FIFO threshold setting to generate an interrupt for applications to retrieve received data or when multiple bytes have completed transmission

Note: Refer to the block diagram for I^{2}C ports that are accessible by the main processor core and the ARC processor core.

3.2.8 I^{2}S

- Two I^{2}S interfaces – one transmit interface and one receive interface. Each interface supports two channels for stereo left and right channels.
- Sample size from 12- to 32-bits
- Support for Left Justified, Right Justified and DSP modes
- Each interface can operate in Master or Slave mode.
- FIFO mode supporting four words of selected data size transmit and receive for each channel.
- Support for hardware DMA that allows data transfer without CPU involvement.
- Audio sample rates up to 48 kHz
- Support for FIFO threshold setting to generate an interrupt for applications to retrieve received data or when multiple bytes have completed atransmission.

3.2.9 UART

- One of two 16550-compliant UART interfaces available to user (UART1); UART0 dedicated to the Bluetooth® low energy controller
- Baud rates from 300 baud to 2 Mbaud
- Hardware and software flow control
- FIFO mode support (16 bytes Tx and Rx FIFOs)
- Hardware DMA with configurable FIFO thresholds
- Hardware, software and no flow control

3.2.10 SPI

The Intel® Curie™ module exposes three master SPI ports:

- Two from the Intel® Quark™ SE microcontroller processor and ARC* processor cores, and one from the sensor subsystem (only accessible through the ARC core)
- One of the ARC SPI is used internally to communicate with the 6-axis sensing device (SPI1_SS)
- Master SPI clock frequencies from 488 hz up to 16 MHz
- One SPI slave interface with support for SPI clock frequencies up to 3.2 MHz (accessible from either the ARC processor core or from the Intel Quark microcontroller core).
- 1-4 chip select lines per each master SPI interface and one for slave interface
- Five slave select pins per master interface
- Support for 4-bit up to 16-bit frame size
- Eight word (4-16 bits) entry for TX and RX FIFOs
- Hardware DMA with configurable FIFO thresholds
- Configurable clock phase and polarity

3.2.11 DMA controller

The DMA engine can be configured and can start performing data transfer without using the CPU in real time to perform the transfer, thus improving performance and reducing the power consumption.

- Eight unidirectional channels
- Support for 16 hardware handshake interfaces
- Dedicated hardware handshaking interfaces with peripherals plus software handshaking support
- Single and multi-block transfers
- Supported transfer modes[1]:
  - Memory to Memory
  - Peripheral to Memory
  - Memory to Peripheral
  - Peripheral to Peripheral

1. Please note that peripherals for the sensor subsystem are not supported by the DMA controller.

3.2.12 GPIO subsystem

Two sets of GPIO signals are available; the ARC* processor core can access its private group of GPIO lines and another set is accessible by both the ARC* processor core and the Intel® Quark™ SE microcontroller C1000 processor core.

There are 54 GPIOs coming from the Intel® Quark™ SE microcontroller C1000 processor core (see the section 1.10 in the Intel® Quark™ SE microcontroller C1000 datasheet). Out of the 54 GPIOs, 12 GPIOs are used internally for Intel® Curie™ Module, and 42 are available out of Intel® Curie™ Module.

3.2.12.1 GPIO controller features

- All GPIOs are interrupt-capable supporting level-sensitive and edge-triggered modes.
- De-bounce logic for interrupt source
- Four external and two internal awake-ON interrupts and wake-capable GPIOs
- Up to four digital inputs and up to four analog inputs that can also can be used as digital I/O; configuration dependent.
- Four pulse width modulated outputs that can be configured as digital I/O lines.
- Separate data register bit and data direction control bit for each GPIO
- GPIO registers retain their state during sleep and wake events.
- Software selectable drive strength via internal pull-up; unused lines must be configured as inputs with internal pull-up or as GPIO with output set to low, or set as input with external pull-up.
- Interrupt mode supported for all GPIOs, with the following configuration:
  - Active High Level
  - Active Low Level
  - Rising Edge
  - Falling Edge
  - Both Edge

3.2.12.2 GPIO of Intel® Curie™ module

The Intel® Curie™ module includes 54 GPIOs in total:

- Four external AON interrupts
- Two internal wake-capable GPIO lines
- Four lines used for analog input for the ADC mux
- 32 GPIO/multifunction lines; these can be configured as GPIO if not consumed by other I/O signals. Refer to the Intel® Quark™ SE microcontroller datasheet for more information.
- 12 GPIO lines used internally within the module

The module internal GPIO signals are 12 in total: three are dedicated to the ARC* processor core, nine are shared and can be configure for either core. These are used internally for control or monitor/interrupt functions.

GPIO mapping tables are provided in section Section 1.3.10

3.2.13 Timers and pulse width modulator (PWM)

- Four counters capable of operating in PWM mode or in timer mode
- Configurable PWM high and low time with granularity of a single 32 MHz clock period per output
- Timer mode support for 32-bit timer operating at 32 MHz
- Two timers for the ARC* processor core and two timers for the Intel® Quark™ SE microcontroller processor core that can also be accessed by the ARC core.
- ARC* processor core timers can be configured as watchdog timers.

Note: PWM keep their state when going to sleep mode, allowing the application to set the wake configuration.

3.2.14 Analog comparators

- 19 analog comparators
  - Six high-performance comparators
  - 13 low-power comparators
- Configurable polarity
- Interrupt and wake event capable
- Can be used with any of the AIN external pins or can be internally connected between the SoC and the resources on the Intel® Curie™ module.
- Wake events support by the low-power comparator interrupt
- Comparator for Interrupt/wake event generation based on programmed match value
- Each comparator can be powered down to achieve even lower power.

3.2.15 Watchdog timer (WDT)

- Configurable watchdog timer able to issue an interrupt and/or a system reset upon timeout.
- Selectable timeout value settable between ~2ms and ~60s (at 32 MHz)
- Interrupt generation on first timeout
- If the interrupt has not been cleared by the second timeout, the WDT requests a microcontroller warm reset to recover for an application program error or unexpected condition.

3.2.16 Real-time clock (RTC)

- 32-bit counter running from 1 Hz up to 32.768 kHz
- Supports interrupt and wake event generation upon matching a programmed value
- Only requires a running 32.768 kHz clock to generate interrupt and wake events
- Supports an additional 32-bit always-on counter
- Supports a 32-bit always-on timer with interrupt and wake capability

3.2.17 Analog-to-digital converter (ADC) unit

- Five analog inputs AIN_10 – AIN_14, 11 other possible analog inputs using alternate functions
- The ADC controller is only accessible from the ARC* processor core and the DMA controller. All configuration and read access is via ARC software.
- Successive-approximation engine with selectable resolution (6, 8, 10 or 12 bit)
- 2.24 MSPS conversion rate – See the *Intel® Quark™ SE Microcontroller C1000 Datasheet* for more details.
- Internal voltage regulator and digital calibration algorithm to improve accuracy
- Only single-ended input options supported
- The digital offset calibration block aids the measurement and correction of the offset voltage for the ADC. This needs to be done after selection of the input pin and the temperature or supply voltage changes. Many applications do not require this type of accuracy and in that case initial calibration should be acceptable and re-calibration may not be required.
- ADC resolution set via ADC_RES[1:0] register; lower bit resolution can be sampled faster than the 12-bit maximum.
  - 11 = 12 bit
  - 10 = 10 bit
  - 01 = 8 bit
  - 00 = 6 bit.
    Note that a lower number of bits can be sampled quicker in less clock cycle than the 12-bit maximum number of bits.
- The ADC supply voltage must match the AIN supply voltage and COMP_AREF to meet the voltage requirement.

Note: The ADC block is sensitive to any noise from the digital IO, RF, Ground and VCC that may affect the accuracy of sampling. We recommend averaging samples, adding filtering to the input and increasing the number of clocks for the samples.

3.2.18 Interrupt

The interrupt is described in the Intel® Quark™ SE microcontroller datasheet.

The interrupt controller to either Intel® Quark™ SE microcontroller processor core or to the sensor subsystem is provided in the system control subsystem, and has the following features:

- Configurable interrupt priority
- Level or pulse sensitivity
- Fast interrupt support for the sensor subsystem second register bank for context switching without saving and restoring registers
- Interrupt vector mapping showing the vector number and associated peripheral, core and co-processor
- Interrupt wake event supported only by the AON signals identified by AON (Awake On / Always On) name

3.2.19 Power management

3.2.19.1 Power supply

The Intel® Curie™ module operates under battery-operated systems using 3.3 V or 1.8 V interfaces, or may be powered by external regulators that provide 1.8 V or 3.3 V. The regulators inside Intel® Curie™ module are used to power the internal circuits or the platform applications: ESR3 is the 1.8 V power regulator used to power the internal circuits, ESR 1 is the 3.3 V power regulator used by external platform devices, and ESR2 is the 1.8 V power regulator used by external platform devices.
An integrated power supervisor holds the system in reset when the input voltage drops below 2.9 V. An inductor and a bulk capacitor are used with the power regulators. VDD_BLE_SEN is the core voltage used to supply the internal Bluetooth® low energy controller. AON_IO_VCC controls the supply voltage to the IO level (1.8 V or 3.3 V) for the peripheral interfaces.
Refer to Section 2.2, "Power and timing considerations" for more details on power management.

3.2.19.2 Power states

Section 2.2.4, "Device power states" describes the Active, Sleep and Off states of the Intel® Curie™ module.

Section 2.2.6, "Boot and reset sequences" describes the power-up sequence that changes the Intel® Curie™ module mode from Off to Active.

Within the Intel® Quark™ SE microcontroller, the Intel Quark SE microcontroller processor core has three main power states: Active, Sleep and Off. The power consumption can be less than 1 uA in Sleep mode.The sensor subsystem also has three power states: sensing active, sensing wait and sensing standby. The power consumption can be as low as 250 uA in low-power sensing standby.
Refer to the Intel® Quark™ SE microcontroller C1000 datasheet for details on the SoC power management.

Section 3.3.1, "Power" provides an overview of the power management for the Bluetooth® low energy controller.

Section 3.4.2, "Power" provides information on the power management of the 6-axis sensing device.

The 3.3 V LDO voltage regulator comes with an active-high enable pin that allows the regulator to be disabled. Forcing the EN low disables the regulator and sends it into an off current state drawing virtually zero current. When disabled, the 3.3 V LDO voltage regulator switches an internal 25 ohm load on the regulator output to discharge the external capacitor. Refer to Microchip* MIC5504-3.3YMT datasheet for more information.

The 1.8 V LDO voltage regulator uses the EN pin to enable and/or disable the device and activate or deactivate the active discharge function. If the EN pin voltage is pulled below 0.4 V, the 1.8 V LDO regulator is guaranteed to be disabled, the Active Discharge Feature is activated and the output voltage is pulled to GND through an internal circuitry with effective resistance of about 100 ohm. iI the EN pin voltage is higher than 1.2 V, the 1.8 V LDO voltage regulator is guaranteed to be enabled. The internal active discharge circuitry is switched off and the desired output voltage is available at the output pin. Refer to ON Semiconductor* NCP170 AMX180TCG datasheet for more information.

Texas Instrument * TPS62743 bulk step-down converter operates in Power Save mode at light loads, and in Pulse Width Modulation (PWM) mode for medium and high load conditions. During PWM mode, the bulk converter operates in continuous conducting mode. The switching frequency is typically 1.2 MHz with a controlled frequency variation depending on the input voltage and load current. If the load current decreases, the converter enters the Power Save mode to maintain high efficiency down to very light loads. The transition from PWM to Power Save mode is seamless with minimum output voltage ripple. Refer to Texas Instrument * TPS62743 datasheet for more information.

3.2.20 Clock management

The system clock has the features listed hereafter. Specific properties are listed in Table 3-1. The accuracy of the clock is maintained within the operating temperature range.

- Dynamic frequency scaling
  - The clocks can be reduced for all blocks—including the Intel® Quark™ SE microcontroller C1000 core, the ARC* processor core, the pattern matching engine, the AHB bus, and the peripherals—to reduce power consumption.
- Dynamic clock gating
  - For example the SPI master clock can be gated off when not sending data to a peripheral and gated back on when the SPI is in use again.
- Autonomous state-based clock gating and autonomous peripheral clock gating
- The 32 kHz clock is a MEMS-based temperature-compensated clock that feeds both the Intel® Quark™ SE microcontroller and the Nordic* Bluetooth® low energy controller. It is always running when power is supplied. Its primary use is for the real time clock and software timer.
- The 32 MHz XTAL oscillator must be used to meet the UART timing

Table 3-1. Intel® Curie™ module system clocks

Clock	Use	Accuracy
32 kHz	Always on timer. 32 kHz oscillator for RTC	±5ppm
SoC Silicon oscillator 4/8/16/32 MHz	Enabled at boot time	±20,000ppm
SoC XTAL oscillator - 32 MHz	Required for USB operation and timing. Can be enabled by software	±30ppm
16 MHz Bluetooth® low energy controller clock	Active during Bluetooth® low energy transmit/ receive	±30ppm

3.2.21 Test and debug

- Test and debug for Intel® Quark™ SE microcontroller
  - Five-pin IEEE 1149.1 JTAG interface
  - Boundary scan support
  - Intel® Quark™ SE microcontroller processor minutia debugger
- ARC* metaware debugger
- Serial Boot Loader
- Test and debug on separate hardware and out of Intel® Curie™ Module
  - Bluetooth® low energy debug and programming via J-Link / SWD emulator

3.3 Bluetooth® low energy controller

The Intel® Curie™ module incorporates a Nordic* nRF51822, which is built around a 32-bit central processing unit with 256 kB of embedded flash and 16 kB of RAM for improved application performance. The embedded 2.4 GHz transceiver supports Bluetooth® low energy as well as Nordic Gazell 2.4 GHz protocol stack. The Bluetooth® low energy controller interfaces with the Intel® Quark™ processor core within Intel® Quark™ SE microcontroller C1000 via UART0.

The Quark processor core software sends commands and receives status and messages from the Nordic* S130 Bluetooth® low energy protocol stack that runs in the Bluetooth® low energy controller and acts like a modem to simplify the software design.

Refer to Nordic* nRF51822 Bluetooth® low energy controller product specification for more details about the device.

3.3.1 Power

The Bluetooth® low energy controller supports three power supply alternatives:

- Internal LDO setup
- DC/DC converter setup
- Low-voltage mode setup

The internal LDO setup can be used with DC/DC converter bypassed, and the system power is generated directly from the supply voltage VDD. The internal LDO can also be used in combination with the DC/DC converter setup. The low voltage mode setup is for devices that are used in low-voltage mode where a steady 1.8 V supply is available externally.

The Bluetooth® low energy controller is powered by the 1.8 V discrete BULK regulator inside the Intel® Curie™ module. There are no hardware handshaking signals that control the power state of the Bluetooth® low energy controller.

In Intel® Curie module, VDD_BLE_SEN is the core voltage supply for the Bluetooth® low energy controller. BLE_DEC2 is the power supply decoupling signal used for 1.8 V IO connections to VDD_BLE_SEN.

Within the Bluetooth® low energy controller the power management is highly flexible with functional blocks such as the CPU, the radio transceiver and the peripherals for which the power state controls are separate, in addition to the global ON and OFF modes.

- In system OFF mode, the Bluetooth® low energy controller is in the deepest power saving mode. The system core functionality is powered down and all ongoing tasks are terminated. Only the Pin wakeup functionality can be setup to be responsive. One or more blocks of RAM can be retained and the device state can be changed to system ON through Reset, the GPIO DETECT signal, or the LPCOMP ANADETECT signal.
- In system ON mode the system is fully operational. The CPU and selected peripherals can be brought into a state where they are functional and more or less responsive depending whether the Low power or the constant latency sub-power mode is selected. All the functional blocks are independently in RUN or IDLE mode depending on the needed functionality.

In Low Power mode the automatic power management system is optimized to save power by keeping as much as possible of the system powered down.
In Constant Latency mode the system is optimized for keeping the CPU latency and the Programmable Peripheral Interconnect (PPI) task response constant and at a minimum, by forcing a set of base resources ON while in sleep mode. This results in higher power consumption for the system.

The Nordic* S130 Bluetooth® low energy protocol stack provides configuration parameters to allow the device to go to sleep when idle. Alternately the controller can be powered by external sources for direct activation to create application specific states.

3.3.2 Clock

Intel® Curie™ module provides the 16 MHz (run) and 32 kHz (standby) clocks to the Bluetooth® low energy controller;

- 16 MHz XTAL clock
- 32 kHz clock oscillator shared between Bluetooth® low energy and the Intel® Quark™ SE microcontroller

In Intel® Curie module, BLE_CL is the I^2C clock interface pin used for the Bluetooth® low energy controller.

3.3.3 Interfaces

The following blocks of the Nordic* nRF51822 interface with other devices within the Intel® Curie™ Module.

- Universal Asynchronous Receiver/Transmitter 0 (UART0) is the interface with Intel® Quark™ processor core
- Clock management (CLOCK) is the interface with the 16 MHz XTAL and the 32 kHz OSC clocks
- The 2.4 GHz radio (RADIO) is connected to the Balun transformer (STMicroelectronics* BAL-nRFD203) via the ANT1 and ANT2 interface pins
- The Two-Wire Interface (TWI) that communicates with a bi-directional wired-AND bus with two lines (SDA and SCL)
- The two-pin Serial Wired Debug (SWD) interface is part of a serial Debug Access port

3.3.4 2.4 GHz radio

The 2.4 GHz RF transceiver operates in the ISM frequency band at 2.400 to 2.4835 GHz. The transceiver receives and transmits data directly from the system memory for efficient packet data management.

The 2.4 GHz multi-protocol radio has the following characteristics:

- +4 dBm to -20 dBm output power in 4 dBm steps; default TX power of 0 dBm
- -30 dBm output power in whisper mode
- Adjustable data rates and power levels at the application level, dependent upon driver and API

3.3.5 Signals

Table 2 lists all the signals used with the Bluetooth® low energy controller, and their function. Also see Section 1.3.12, "Bluetooth® low energy controller pins".

Table 2. Bluetooth® low energy controller signals

Ball Name	Description
ATP_BLE_INT	Input interrupt signal from Intel® Quark™ SE microcontroller to the Bluetooth® low energy controller
BLE_ATP_INT	Output interrupt signal from the Bluetooth® low energy controller to Intel® Quark™ SE microcontroller
BLE_SW_CLK	Intel Curie module pin D24. This signal has a 22 kohm internal pull-down resistor to keep it from floating. Signal used with the J-Link emulator to program or debug the Bluetooth® low energy controller.
BLE_SWDIO	This signal has a 22 kohm internal pull-down resistor to keep it from floating. Signal used with the J-Link emulator to program or debug the Bluetooth® low energy controller.
BLE_SDA	I^2C interface data output signal from the Bluetooth® low energy controller to an external master. Requires an external pull-up
BT_GPIO	GPIO output signal from the Bluetooth® low energy controller
BLE_SCL	I^2C interface output clock signal from the Bluetooth® low energy controller to an external device (optional)
BLE_RF	Input/output signal used for the connection of the Bluetooth® low energy controller to the antenna via the BALUN (balanced/unbalanced) transformer
BLE_DEC2	Power supply decoupling. Leave unconnected for 3.3 V IO. Used for 1.8 V IO connection to VDD_BLE_SEN
VDD_BLE_SENS	Bluetooth® low energy controller power supply with internal 0.1 uF capacitor.
UART0_CTS	UART0 interface line input signal from Intel Quark processor core to Bluetooth® low energy controller, used for flow control
UART0_TXD	UART0 interface line transmitter output signal from the Bluetooth® low energycontroller to Intel Quark processor core
UART0_RTS	UART0 interface line output signal from the Bluetooth® low energycontroller to Intel Quark processor core, used for flow control
UART0_RXD	UART0 interface line receiver input signal from Intel Quark processor core to the Bluetooth® low energy controller

3.3.6 Software stack support

The Nordic* S130 Bluetooth® low energy protocol stack provides concurrent multi-link Central, Peripheral, Broadcaster, and Observer roles. The S130 series SoftDevice is compliant with Bluetooth® 4.1 and the SoftDevice enables Bluetooth network topologies.

3.3.7 Programming and debug

Multiple methods are available to load a software image into the Bluetooth® low energy section:

- If USB is implemented for the application, it can be used to load an image using the DFU utility.
- The JTAG programmer (Flyswatter2* or J-Link) supported by the Intel® Quark™ SE microcontroller can be used to load the image. Refer to the Nordic website for other ways to load an image to the Bluetooth low energy block.
- Software solutions can be used to receive the image from UART and use Over-the-Air to program it.

3.4 Sensor device

The Bosh* BMI160 is a 16-bit inertial measurement unit designed for low-power, low-noise and high precision 6-axis and 9-axis applications. The inertial measurement unit combines the data captured from the tri-axis low-g accelerometer and the 3-axis gyroscope. The 16-bit tri-axial accelerometer detects the linear motion and gravitational forces. The 16-bit tri-axial gyroscope measures the rate of rotation in space. The 6-axis sensor device is capable of handling external sensor data, for example geomagnetic or barometric pressure sensors.

Refer to Bosch* BMI160 datasheet and software library support.

3.4.1 Feature summary

- Digital resolution
  - Accelerometer (A): 16 bit
  - Gyroscope (G): 16 bit
- Measurement ranges (programmable)
  - (A): ± 2 g, ± 4 g, ± 8 g, ± 16 g
  - (G): ± 125°/s, ± 250°/s, ± 500°/s, ± 1000°/s, ± 2000°/s
- Sensitivity (calibrated)
  - (A): ±2g: 16384 LSB/g
  - ±4g: 8192 LSB/g
  - ±8g: 4096 LSB/g
  - ±16g: 2048 LSB/g
  - (G): ±125°/s: 262.4 LSB/°/s
  - ±250°/s: 131.2 LSB/°/s
  - ±500°/s: 65.6 LSB/°/s
  - ±1000°/s: 32.8 LSB/°/s
  - ±2000°/s: 16.4 LSB/°/s
- Zero-g offset (typ., over life-time)
  - (A): ±40 mg
  - (G): ± 10°/s

- Noise density (typ.)
  - (A): 180 µg/vHz
  - (G): 0.008°/s/vHz
- Bandwidths (programmable)
  - 1600 Hz … 25/32 Hz
- Temperature range
  - -40 … +85°C
- Current consumption
  - full operation = 950 µA
  - low-power mode = 3 µA
- FIFO data buffer
  - 1024 byte
- Shock resistance
  - 10,000 g x 200 µs

3.4.2 Power

Within Intel® Curie™ module, the VDD_6AXIS signal brings the power supply to Bosh* BMI160 device. The 6-axis sensing device is powered by the 1.8 V discrete BULK regulator inside Intel® Curie™ module. There are no hardware handshaking signals that control the power state of the sensing device. The power management configuration is done by software through the SPI interface with the ARC* processor core.

The built-in power management unit (PMU) of the 6-axis sensing device can be configured for example to further lower the power consumption by automatically sending the gyroscope into fast start-up mode and waking it up again by internally using the any-motion interrupt of the accelerometer.

By default the accelerometer and the gyroscope are in suspend mode after powering up the device. The device powers up in less than 10 ms.

The accelerometer has the following power modes:

- Normal mode: full operation
- Low-power mode: duty cycling between suspend and normal mode. FIFO data readout supported to a limited extent.
- Suspend mode: no data sampling, all data is retained and delays between subsequent I^2C operations are allowed. FIFO data readout is not supported.

The gyroscope has the following power modes:

- Normal mode: full operation
- Suspend mode: no data sampling, all data is retained and delays between subsequent I^2C operations are allowed. FIFO data readout is not supported.
- Fast start-up mode: start-up delay time to normal mode takes 10 ms or less

Suspend and fast start-up modes are sleep modes. Switching between normal and low power mode does not impact the output data from the sensor. This allows the system to switch from low power mode to normal mode to read out the sensor data in the FIFO with a data rate limited by the serial interface.

The power mode setting can be configured independently from the output data rate. The main difference between normal and low power modes is the power consumption. The highest current consumption occurs when both the accelerometer and the gyroscope are in normal mode, and reaches 925 uA. When both the accelerometer and the gyroscope are in suspend mode, the power consumption is of 3 uA.

When an external magnetometer is connected via the secondary I^2C interface in Intel® Curie™ module, the power management unit allows advanced power management, and supports the three power modes for the magnetometer: suspend, normal and low power.

3.4.3 Sensor timing

The register SENSORTIME is a free-running counter which increments with a resolution of 39 us. All sensor events such as the updates of data registers are synchronous to the SENSORTIME register. The time stamps in the SENSORTIME register are available independently of the power mode of the device.

3.4.4 Data synchronization

The sensor time is synchronized with the update of the data register. Synchronization is a digital statement.

The sensor data from the accelerometer and the gyroscope are strictly synchronized on hardware level. That is, they run on exactly the same sampling rate. Bosch* BMI160 supports various levels of data synchronization. See the Bosch* BMI160 datasheet for the list of supported data synchronization.

3.4.5 Data processing

The data from the sensor is always sampled with a data rate of 6400 Hz for the gyroscope and 1600 Hz for the accelerometer. The data processing implements a low pass filter configured in the register ACC_CONF for the accelerometer, and GYR_CONF for the gyroscope. Further down sampling for the interrupt engines and the FIFO is possible and configured in the register FIFO_DOWNS. This down sampling discards data frames.

3.4.6 FIFO

The FIFO block integrated in the Bosch* BMI260 sensing device is used to support low power applications and to prevent data loss in non-real time systems. The FIFO space for the accelerometer and the gyroscope is dynamically allocated.

Data is stored in the FIFO in data frame units. The frame rate for the FIFO is defined by the maximum output date rate of the sensors enabled via the register FIFO_CONFIG. If the setting for pre-filtered data is selected in the register FIFO_DOWNS, the data rate for the gyroscope is 6400 Hz and it is 1600 Hz for the accelerometer. The frame rate can be further reduced by selecting the down sampling setting for each or both sensors in FIFO_DOWNS register. Down sampling decreases the sensor data rate. During down sampling no data processing nor data filtering are performed.

Refer to the Nordic* nRF51822 datasheet to see details on the FIFO frame configurations.

3.4.7 Interfaces

In Intel® Curie™ module, the Bosch* BMI260 sensing device primary SPI interface is used to connect to the ARC* processor core. On the ARC* processor core, the SPI interface is SPI1_SS.

Also in Intel® Curie™ module, the Bosch* BMI260 sensing device secondary I^2C interface can be used to connect to a magnetometer and achieve some 9-axis (9AXIS) sensing capability. When connected to a geometric sensor, the Bosh* BMI160 triggers autonomous read-out of the sensor data from the magnetometer without the need for intervention by the ARC* processor core. In this configuration the 6-axis sensing device controls the data acquisition of the external sensor. The synchronized data of all the sensors can be stored in the register data and additionally in the built-in FIFO.The integrated 1024 byte built-in FIFO prevents data loss in non-real time systems.

3.4.8 Interrupt management

The 6-axis sensing device features an on-chip interrupt engine enabling low-power motion-based gesture recognition and context awareness such as any-or no-motion detection, tap or double tap sensing, orientation detection, free-fall or shock events.

There are two output interrupt pins, to which 13 different interrupt signals can be mapped independently via user programmable parameters.

The available interrupts supported by the accelerometer in normal mode are:

- Any-motion (slope) detection
- Significant motion
- Step detection
- Tap sensing (detection of single or double tapping events)
- Orientation detection
- Flat detection
- Low-g/high for the detection of very small or very high accelerations
- No/slow motion

There are two additional common interrupts for the accelerometer and the gyroscope:

- Data ready (“new data”) for synchronizing the sensor data read-out with the ARC* processor core
- FIFO full/FIFO watermark, to handle the FIFO fill level and overflow

The interrupts are available in normal and low-power modes, but not in suspend mode.

The 6-axis sensing device can interrupt the external magnetometer to coordinate the communication to it. It can also interrupt the ARC* processor core when it needs priority attention.

In Intel® Curie™ module, GPIO_AON[4] in Intel® Quark™ SE microcontroller receives the interrupt signal 6AXIS_INT1 from the 6-axis sensing device when data is available.

3.4.9 Signals

Table 2 lists all the signals used with the Bosch* BMI2606-axis sensor, and their function. Also see Section 1.3.11, "6-Axis sensing device pins".

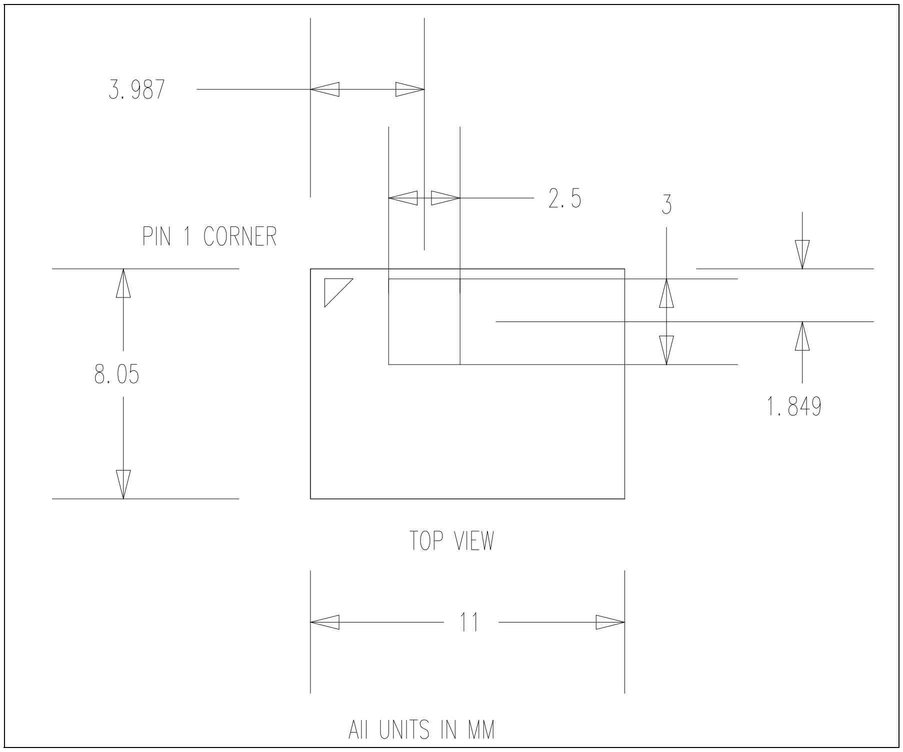
Table 3. Bosch* BMI260 6-axis Sensing Device Signals

Ball Name	Description
6AXIS_MISO	6-axis sensor SPI interface master input slave output signal
6AXIS_MOSI	6-axis sensor SPI interface master output slave input signal
6AXIS_SCLK	6-axis sensor SPI interface serial clock signal (output from the master)
6AXIS_CS	6-axis sensor SPI interface chip select signal
6AXIS_INT1	Internal interrupt output signal from the 6-axis sensing device to the Intel ARC* processor core
6AXIS_SDA	6-axis sensor I^2C interface serial data line output signal to the an external magnetometer to get a 9-axis sensing device. Requires external pull-up
6AXIS_SCL	6-axis sensor I^2C interface serial clock output signal to the an external magnetometer to get a 9-axis sensing. Requires external pull-up.
6AXIS_INT2	6-axis interrupt output signal to an external magnetometer to get 9-axis sensing

3.4.10 Sensor position on Intel® Curie™ module

Figure 3-2 shows the approximate location of the sensor device within Intel® Curie™ module.

Figure 3-2. Intel® Curie™ module X-Y dimensions and position of the six-axis sensor



3.5 Memory

The Intel® Quark™ SE microcontroller C1000 supports two address space mappings:

- Physical address space mappings
- Sensor subsystem auxiliary address space mappings

3.5.1 Memory map

3.5.1.1 Physical address space mappings

There are 4 GB (32 bits) of physical address space that can be used as:

- Memory mapped I/O (MMIO – I/O fabric)
- Physical memory (system Flash/system SRAM/external SRAM)
- System Flash 0: 192 kB (including system ROM)
- System ROM: 8 kB: Write protected section of the Flash that is used as One Time Programmable (OTP) memory for information memory and also for user information like ID or serial number
- System Flash 1: 192 kB
- Internal System SRAM: 80 kB
- 8 kB of Data Closely Coupled Memory (DCCM) in the sensor subsystem

Both the Intel Quark microcontroller core and the ARC* core can access the full physical address space. The ARC* processor core maps the peripherals directly attached to the sensor subsystem to an auxiliary address space that the ARC* processor core has exclusive access to.

All Intel® Curie™ module peripherals, except the ones on the ARC* processor core, map their registers and memory to physical address space. Other devices within the Intel® Curie™ module can only access regions of the physical address space presented to a given device via the multi-layer SoC fabric.

Refer to Intel® Quark™ SE microcontroller datasheet for the memory mapping information.

3.5.1.2 Sensor subsystem auxiliary memory map

The ARC* processor core has access to two physically separate memory spaces.

- The first memory space is the main memory space and is shared with the host processor and SoC peripherals.
- The other memory space is an auxiliary memory space that the ARC* processor core uses to access the peripherals that are directly connected to the sensor subsystem.

Only the ARC* processor core can access the auxiliary memory space.

4 Reference and Resources

4.1 Software support

The Intel® Curie™ Open Developer Kit (ODK) includes the software, tools and documentation for developers to build boards based on the Intel Curie module and turn them into products. Further information and resources for the ODK can be found at:

Intel® Curie™ Open Developer Kit (ODK)

4.2 Related documents

The following documents provide information related to the Intel® Curie™ module and the Intel® Quark™ SE microcontroller C1000:

- Intel® Curie™ Module Design Guide
- Module Power Sequencing for Intel® Curie™ Module
- Intel® Curie™ Modules support page
- Intel® Curie™ Module developer page
- Intel® Quark™ SE Microcontroller C1000 Datasheet
- Intel® Curie™ Module Specification Update

4.3 Intel® Curie™ module-related community resources

For frequently asked questions about the Intel® Curie™ module, please visit Intel® Curie™ Module Support page.

If you need further information or help, or to find and share solutions with Intel® Curie™ module users across the world, please visit Intel® Curie™ Forum in Intel's Support Communities.

4.4 Component reference

The Intel® Curie™ module incorporates the following components. Please see the respective manufacturer's website for related documentation and other resources:

- SoC: Intel® Quark™ SE microcontroller C1000
- Bluetooth® low energy: Nordic* nRF51822--CEAAE0/PAN V3.0 (Stack S130)
- Balun transformer: STMicroelectronics* BAL-NRF02D3
- 6-axis sensing device: Bosch* BMI160
- Battery charger: Texas Instruments* BQ25101H
- Power supervisor: Maxim* MAX16074RS29D3+T
- Power supervisor: RICOH* R3117K161C
- LDO 1.8 V: OnSemiconductor* NCP170AMX180TCG

- LDO 3.3 V (USB): Microchip* MIC5504-3.3YMT
- Buck regulator 1.8 V/3.3 V: Texas Instruments* TPS62743

Note: Refer to the respective third-party, and Intel® Quark™ SE microcontroller documents as required.

4.5 Terminology

Table 4-1 lists some terms that are used in this document.

Table 4-1. Terminology

Term	Definition
AC	Alternating current
ADC	Analog-to-digital converter
AIN	Analog input
AHB	AMBA high-performance bus is a single clock-edge protocol
AON	Always-on (Wake Event)
ARC*	Argonaut RISC core
Intel® Quark™ SE microcontroller	Intel® Quark™ SE microcontroller C1000 (WLCSP package used in Intel® Curie™ module)
BALUN	Balanced-unbalanced
BLE	Bluetooth® low energy (formerly Bluetooth® Smart)
CPU	Central processing unit
CCM	Counter with CBC-MAC
DC	Direct current
DCCM	Data closely coupled memory
DFU	Device firmware update
DMA	Direct memory access
DMIPS	Dhrystone million instructions per second
DSP	Digital signal processor
ESD	Electrostatic discharge
FIFO	First in first out
GPIO	General-purpose input/output
I^2C	Inter-integrated circuit (bus)
I^2S	Inter-IC sound (bus)
IDE	Integrated development environment
IO, I/O	Input/output
ISA	Instruction set architecture
JTAG	Joint test action group (debugging interface)
LDO	Low-dropout regulator
MSL	Moisture sensitivity level
nCTF	Non critical to function
NVM	Non-volatile memory
ODM	Original design manufacturer
OS	Operating system

Table 4-1. Terminology (continued)

Term	Definition
OTA	Over the air
OTP	One-time programmable
PCB	Printed circuit board
PCM	Pulse code modulation
PTU	Power transmit unit
PWM	Pulse width modulation
RAM	Random access memory
RTC	Real-time clock
SMT	Surface mount technology
SoC	System on a Chip. Used for the Intel® Quark™ SE microcontroller C1000 in some sections of the document
SPI	Serial peripheral interface (bus)
SRAM	Static random access memory
SW	Software
TBD	To be determined
TCM	Tightly coupled memory
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus
WDT	Watchdog timer
XTAL	Crystal

5 Packaging and Ordering

5.1 Package information

This section presents information on Intel® Curie™ module package geometry and package marking.

5.1.1 Packing geometry

Table 5-1 shows details for the Intel® Curie™ module package.

Table 5-1. Module package details

Module Attribute	Value and Tolerance
Package size	11.00 x 8.05 mm ±0.05 mm
Package height	1.95 mm ±0.09 mm
Solder ball material	SAC 1205
Surface finish	CuOSP
Ball count	111
Ball pitch	0.57 x 0.45 mm
Solder result opening	0.240 mm
Ball diameter (pre-attach)	0.308 mm / 12.13 mil
Die thickness	0.240 mm
Substrate thickness	0.70 mm
Maximum Z-height (pre/post SMT)	1.86 mm to 1.95 mm ± 0.09 mm
NCFT corner balls	N/A
Overmold (mold cap showing device identification and marking)	Yes - 1 mm

5.1.2 Package marking

Figure 5-1 shows the marking for Intel® Curie™ module.

Figure 5-1. Package marking

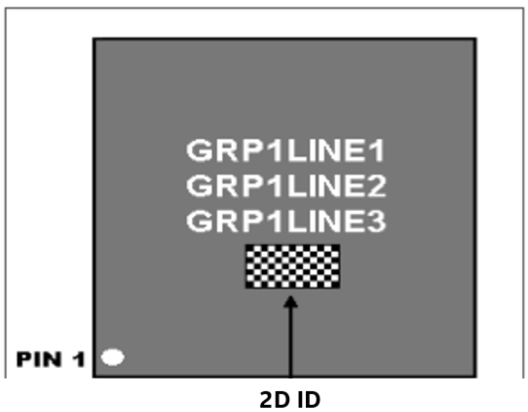


Table 5-2. Package marking

Line Number	Overview	Description
Line 1	-	Intel symbol
Line 2	{FPO}	FPO field number
Line 3	SR2NW {e1}	Product identification (engineering sample of HVM) and Pb-free compliance indicator {e1}

5.2 Ordering information

Table 5-3 provides information on Intel part numbers. Contact your regional sales representative for pricing information.

Table 5-3. Ordering information for Intel® Curie™ module

Part Number (MM#)	Description	Packing
948119 - OBSOLETE[1]	Intel® Curie™ module integrated in Arduino* 101 (branded Genuino* 101 in some countries)	N.A.
949350	Intel® Curie™ module with battery charger, in 11.00 mm x 8.05 mm package	Tray[2]

Notes:
1. These parts cannot be ordered.
2. No being shipped in tape and reel at this time.

5.3 Storage information

Table 5-4 specifies the package storage requirements.

Please note that:

- $T_{ABSOLUTE\ STORAGE}$ applies to the un-assembled component only and does not apply to the shipping media, moisture barrier bags or desiccant. Refers to a component device that is not assembled in a board or socket that is not to be electrically connected to a voltage reference or I/O signals.
- The specified temperatures are based on data collected. Exceptions for surface mount re-flow are specified by applicable JEDEC J-STD-020 and MAS documents. The JEDEC, J-STD-020 moisture level rating and associated handling practices apply to all moisture sensitive devices removed from the moisture barrier bag.
- Post board attach storage temperature limits are not specified. Consult your board manufacturer for storage specifications.

Table 5-4. Package storage specifications

Parameter	Description	Min	Max
$T_{ABSOLUTE\ STORAGE}$	The non-operating device storage temperature. Damage (latent or otherwise) may occur when subjected to this temperature for any length of time.	-25°C	125°C
$T_{SUSTAINED\ STORAGE}$	The ambient storage temperature limit (in shipping media) for a sustained period of time.	-5°C	40°C
$RH_{SUSTAINED\ STORAGE}$	The maximum device storage relative humidity for a sustained period of time.	60% @ 24°C	-
$TIME_{SUSTAINED\ STORAGE}$	A prolonged or extended period of time: associated with customer shelf life in applicable Intel® boxes and bags	0 months	6 months

6 Manufacturing Information

6.1 Bootloader information

The Intel® Curie™ module comes with a manufacturing bootloader by default and customers can flash the bootloader of their choice through JTAG.

6.2 PCB pad design guidelines

To achieve the highest solder joint strength and reliability, we recommend the following:

- PCB pad size: 10 mil, MD pad
- No nCTF

Figure 6-1 illustrates the pad layout.

.

Figure 6-1. PCB pad layout

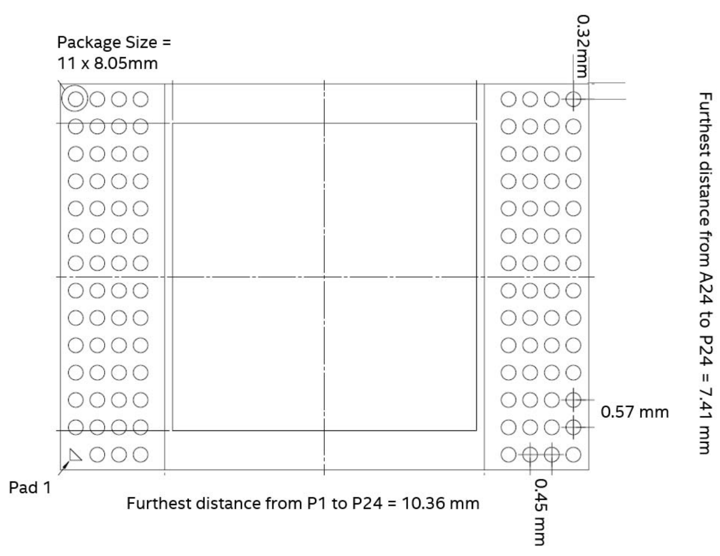
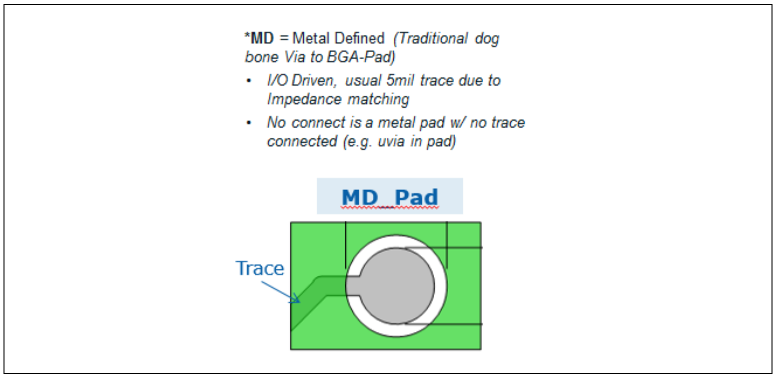


Figure 6-2 provides details about the metal defined pad.

Figure 6-2. Metal-defined pad



6.3 Manufacturing guidelines

The following two subsections provide some general guidelines on the reflow and rework for systems associated with Intel® Curie™ module.

6.3.1 SMT reflow parameters

Table 6-1 lists the SMT reflow parameters.

Disclaimer: These results are provided for informational purposes only. Any sourcing decision is solely at the discretion of the Customer. Intel neither warrants, nor makes any representation whatsoever as to any supplier's products, including its availability, suitability, and reliability for the application for which this information is provided. Other pastes not listed above may perform better or worse based on customer SMT / board characteristics. The solder paste formulation above have passed Intel's SMT stress test and are recommended for customer evaluation with the Intel® Curie™ module, to achieve acceptable SMT Yields.

Table 6-1. SMT reflow parameters

Parameter	Description / Values
Intel evaluated solder paste	SAC305 Type 4
Solder joint peak temperature	240°C ±5°C, should remain below 245°C
Maximum body and substrate temperature	Never exceed 245°C
Time above =<217°C (TAL)	60-90 sec
Soak	Paste dependent. Consult paste manufacturer.
Rising ramp rate	< 3°C per second
Falling ramp rate	Maximum 3°C per second. Minimum 1°C per second from peak to 205°C
Reflow ambient[1]	N_2 ambient (air is acceptable)
Reflow profile	Intel® Curie™ module is validated for 3 reflows
Pallet Support for Board Warpage	Recommended
SMT stencil information	Stencil thickness 3 mil aperture 10 mil round
MSL level	**MSL Level 6**. Must be reflowed within the time limit specified on the label. Units should not be left on the floor as parts can absorb moisture and fail. If parts are not being used they should be put back into MBB and the bags should be vacuum packed or sealed with desiccant and HIC card. Floor exposure should be tracked.
Other critical information	Please SMT the parts within 8 hours of opening the bag (typical 1 shift). If the cumulative time out of the bag (as measured from the 1st open bag time) exceeds 8 hours but is below 48 hours, the parts should be baked at 125°C for 24 hours. Please note that only 1 bake is allowed. Please SMT the parts within 8 hours of bake. Partial lots after bake can be put back in MBB with desiccant to stop (but not reset) MET clock but cumulative moisture exposure should not exceed 8 hours after bake. If moisture exposure is beyond 48 hours then the units are irrecoverable.

Notes:
1. Intel internal SMT development result indicated N2 is highly recommended to achieve the optimized SMT result. Intel internal SMT development used O2<1000 ppm concentration.

6.3.2 Board rework

The primary factors for board rework are the following:

- Flux formulation, solder paste formulation and volume
- A capable thermal reflow profile
- Proper PCB pad solder preparation/wicking (clean-up of the residual solder from the PCB pads)

The application design/board size, material and thickness will change the SMT profile. This section provides some guideline for building a board size.

Caution: Always remove the batteries before reworking the board.

Table 6-2 lists the rework recommendations for customer considerations.

Notes: Intel internal SMT development result indicated N2 is highly recommended to achieve the optimized SMT result. Intel internal SMT development used O2<1000 ppm concentration.

Disclaimer: These results are provided for informational purposes only. Any sourcing decision is solely at the discretion of the Customer. Intel neither warrants, nor makes any representation whatsoever as to any supplier's products, including its availability, suitability, and reliability for the application for which this information is provided. Other pastes not listed above may perform better or worse based on customer SMT / board characteristics. The solder paste formulation above have passed Intel's SMT stress test and are recommended for customer evaluation with the Intel® Curie™ module, to achieve acceptable SMT Yields.

Please share SMT yield summary (yield Pareto if possible) with Intel representative for revenue builds so we can track performance against our goals. In case of SMT failures, please secure the boards for timely failure analysis by Intel team to understand the defect mode.

Table 6-2. Rework reflow parameter recommendations

Parameter	FBGA and Chipset
Solder paste formulation	SAC305 type A, same stencil used in SMT
Flux formulation	TBD
Solder paste volume	Over-print
Rework pallets	Case dependent. Follow customer practice.
Gap between nozzle and PCB surface (optimize air flow)	762 um (30mils)
Placement force	0 grams (paste application) 140 grams (flux application)
Rework ambient	Air (Nitrogen was not evaluated)
Solder joint peak reflow temperature	230 to 245°C
Time above ≥ 217°C	60 to 90 seconds
Maximum body temperature	Never exceed component temperature ≥ 245°C
Component delta T (ΔT)	≤ 10°C
Soak temperature and time	Paste dependent - Consult the paste manufacturer.
Rising ramp rate below 150°C (+)	0.5 to 2.5°C/sec
Rising ramp rate between 205°C and 215°C (+)	0.35 to 0.75°C/sec
Falling ramp rate (-)	0.50 to 2.0°C/sec

6.4 General handling recommendations

The general handling recommendations are illustrated in Figure 6-3.

Figure 6-3. General handling recommendations