

# Fall Detection and Posture Monitoring System

A Major Qualifying Project  
Submitted to the Faculty of  
Worcester Polytechnic Institute  
In partial fulfillment of the requirements for the  
Degree in Bachelor of Science  
in  
Electrical and Computer Engineering  
By

-----  
Yifei Jin

-----  
Syed Ayaz Naeem

-----  
Holly Shumway

Date: 3/1/19  
Sponsoring Organization:  
Analog Devices & Allegro Microsystems  
Project Advisors:

----- Professor John McNeill, Advisor

----- Professor Ulkuhan Guler, Co-Advisor

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>Executive Summary</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Brainstorm</b>	<b>1</b>
2.1 Proposed Projects . . . . .	1
2.1.1 Fall Detection System . . . . .	1
2.1.2 Posture Monitoring System . . . . .	2
2.1.3 Skin Impedance Spectrometer . . . . .	3
<b>3 Decision Methods</b>	<b>5</b>
3.1 Evaluation Criteria . . . . .	5
3.2 Decision Matrix . . . . .	7
3.3 Final Decision . . . . .	8
3.4 Use Cases . . . . .	8
<b>4 Background Research &amp; Product Specifications</b>	<b>11</b>
4.1 Target Markets . . . . .	11
4.1.1 Fall Detection and Prevention System . . . . .	11
4.1.2 Posture Monitoring System . . . . .	14
4.2 System Requirements . . . . .	16
4.2.1 Accelerometer . . . . .	16
4.2.2 Gyroscope . . . . .	17
4.2.3 Microcontroller . . . . .	18
4.2.4 Wireless Communication . . . . .	18
4.2.5 Battery . . . . .	19
4.3 Component Research & Specifications . . . . .	19
4.3.1 Accelerometer & Gyroscope . . . . .	19
4.3.2 Microcontroller . . . . .	21
4.3.3 Wireless Communication . . . . .	23
4.3.4 Battery . . . . .	23
<b>5 Financial Analysis</b>	<b>26</b>

5.1	Bill of Materials . . . . .	26
5.2	Initial Investment . . . . .	27
5.3	Break-even Analysis . . . . .	28
<b>6</b>	<b>Hardware Design</b>	<b>29</b>
6.1	Overall Design . . . . .	29
6.2	System Blocks and Specifications . . . . .	30
6.2.1	Power Supply . . . . .	30
6.2.2	Microcontroller . . . . .	31
6.2.3	Sensor . . . . .	31
6.2.4	Bluetooth . . . . .	32
6.2.5	Component Housing . . . . .	32
<b>7</b>	<b>Software Design</b>	<b>36</b>
7.1	Arduino Code . . . . .	36
7.2	Algorithms for Detection . . . . .	39
7.2.1	Fall Detection Algorithm . . . . .	41
7.2.2	Posture Monitoring Algorithm . . . . .	42
7.3	App Design . . . . .	44
<b>8</b>	<b>PCB Construction</b>	<b>47</b>
8.1	Schematic Design . . . . .	47
8.2	Footprint Creation . . . . .	49
8.3	Layout Design . . . . .	49
8.4	Optimization: Routing and Component Placement . . . . .	50
<b>9</b>	<b>System Testing &amp; Results</b>	<b>51</b>
9.1	PCB Testing & Results . . . . .	51
9.2	Angle Monitoring Functionality & Accuracy Testing & Results . . . . .	52
9.3	Fall detection Functionality Testing & Results . . . . .	57
9.4	BLE Testing & Results . . . . .	62
9.5	Battery and PowerBoost Testing & Results . . . . .	67
<b>10</b>	<b>System Improvements</b>	<b>68</b>
10.1	Microprocessor & BLE . . . . .	68
10.1.1	Updated Power Analysis . . . . .	69
10.1.2	Updated Cost Analysis . . . . .	70
10.2	Power Supply . . . . .	71
10.3	App Design . . . . .	72
10.4	PCB Design . . . . .	73

<b>11 Video Production</b>	<b>74</b>
<b>Bibliography</b>	<b>77</b>
<b>Appendices</b>	<b>82</b>
<b>A Decision Matrix</b>	<b>83</b>
<b>B Arduino Code</b>	<b>85</b>
<b>C App Code Blocks</b>	<b>89</b>
<b>D Python Script</b>	<b>96</b>

# List of Figures

1	Project flow chart . . . . .	vi
3.1	Evaluation of optimal sensor placement for fall detection . . . . .	9
3.2	Posture diagram for optimal sensor placement . . . . .	10
4.1	Elderly fall detection target market . . . . .	12
4.2	Hospital patient fall detection target market . . . . .	12
4.3	Active individuals fall detection target market . . . . .	13
4.4	Under the influence fall detection target market . . . . .	13
4.5	Children fall detection target market . . . . .	14
4.6	Children posture target market . . . . .	15
4.7	Student posture target market . . . . .	15
4.8	Office worker's posture target market . . . . .	16
4.9	LSM9DS0 component . . . . .	21
4.10	ATMEGA328-PU component . . . . .	22
4.11	HC-06 module . . . . .	23
4.12	3.7V Polymer Ion battery component . . . . .	24
6.1	System block diagram . . . . .	30
6.2	Power supply block diagram . . . . .	31
6.3	Sensor block diagram . . . . .	32
6.4	Bluetooth block diagram . . . . .	32
6.5	Component housing design . . . . .	33
6.6	Dimensions in mm of component housing design . . . . .	34
6.7	Final component case . . . . .	35
7.1	Bluetooth flag functionality flowchart . . . . .	38
7.2	Algorithm flowchart . . . . .	40
7.3	45 degree fall acceleration analysis . . . . .	41
7.4	Euler angle inertial frame . . . . .	42
7.5	Ideal posture while sitting . . . . .	44
7.6	Main screen for application . . . . .	45
7.7	Settings screen for application . . . . .	46
8.1	Schematic design . . . . .	48
8.2	ATMEGA328-PU footprint . . . . .	49
8.3	LSM9DS0 footprint . . . . .	49

8.4	“ASSEMBLY” view of the PCB . . . . .	50
9.1	Final PCB after soldering and testing . . . . .	51
9.2	Posture angle accuracy results . . . . .	53
9.3	Reference posture to leaning over on desk . . . . .	54
9.4	Reference posture to slouching on desk with elbow . . . . .	55
9.5	Reference posture to slight slouch . . . . .	56
9.6	Reference posture to picking up object . . . . .	57
9.7	No fall jumping test case . . . . .	58
9.8	No fall walking test case . . . . .	59
9.9	Fall while running test case . . . . .	60
9.10	Fall while walking test case . . . . .	61
9.11	User stands up after a fall . . . . .	62
9.12	Bluetooth testing with nRF Connect application . . . . .	63
9.13	Indicator light that turns green or red based off Bluetooth connectivity	63
9.14	Sine wave data graph . . . . .	64
9.15	Flag functionality for Bluetooth module . . . . .	65
9.16	Accelerometer data graph . . . . .	66
9.17	Gyroscope data graph . . . . .	66
9.18	Oscilloscope output for the 5V PowerBoost . . . . .	67
10.1	CC2650MODA launchpad . . . . .	69
10.2	Coin battery as potential system improvement . . . . .	71
10.3	2500mAh Lithium Ion Polymer Battery as potential improvement . .	72
10.4	Android studio logo . . . . .	72
11.1	Steps for video production . . . . .	74
11.2	Storyboard of scenes one through three . . . . .	75
11.3	Script for scene one . . . . .	76
11.4	Editing & compiling video footage in Adobe Premiere Pro . . . . .	77

# List of Tables

2.1	Fall detection device competitors . . . . .	2
2.2	Posture monitoring device competitors . . . . .	3
2.3	Skin impedance spectrometer competitors . . . . .	4
3.1	Decision criteria weights . . . . .	7
3.2	Decision matrix scores . . . . .	8
4.1	Comparison of sensors . . . . .	20
4.2	Sensor characteristics . . . . .	21
4.3	Microcontroller comparisons . . . . .	22
4.4	Bluetooth comparison . . . . .	23
4.5	Battery comparisons . . . . .	24
4.6	Total current usage of components in the system . . . . .	25
5.1	Bill of materials . . . . .	27
5.2	Initial investment . . . . .	28
9.1	Observed charging times based on battery charge . . . . .	67
10.1	Total current usage of components in the system . . . . .	70
10.2	Updated bill of materials . . . . .	71

# Abstract

The healthcare field can greatly benefit from the inclusion of innovative smart devices. Two common health concerns addressed in the industry are injuries due to falls and improper posture. This project aimed at designing a device to prevent the occurrence of these problems by monitoring a user's walking and posture behaviors. The intent was to alert the user or a nearby caregiver if a fall or abnormal posture were detected. The device integrated sensors along with Bluetooth communication to successfully achieve this task.



# Acknowledgements

The completion of this project was made possible by Professor John McNeill, PhD for providing the project with a direction and a focus as well as technical expertise, Professor Ulkuhan Guler, PhD for providing great technical expertise, and William Appleyard for his help in obtaining the resources needed for the construction of our project.

Additionally, we would like to thank the NECAMSID lab as well as Analog Devices and Allegro Microsystems for helping sponsor our project. Furthermore, we would like to thank Michael Coln at Analog Devices for taking the time to meet with us via Skype several times as well as providing us with valuable advice and his technical expertise. Lastly, we would like to thank PhD student Sulin Li for providing us with several soldering techniques that were crucial to the success of our project.

# Executive Summary

To culminate one's education at Worcester Polytechnic Institute (WPI), students are required to complete the Major Qualifying Project (MQP), a capstone, in their respective majors. Though some teams begin their project with a solidified idea, our project was introduced to us with the general description of "analog applications". With no concrete direction on where to begin, we began a process of brainstorming possible ideas for our MQP. Ten possible project ideas were brainstormed and they were: an EEG, a continuation of the skin impedance spectrometer MQP, an ADC design, an application demo for chips, a continuation of the smart shirt MQP, a fall detection system, heart rate monitoring, DUI prevention, a posture monitoring system, and a LIDAR crash prevention system.

After brainstorming ten possible project ideas and conducting preliminary research on each, we needed to critically examine each idea and decide which ones had a significant concentration of analog material and was the most feasible to complete in three terms. To help guide the decision process, we developed ten decision criteria that we felt our project should include. The ten criteria were as follows: relevant and new to industry, solves an interesting and important problem, want it to work and function on time, improves skills and technical growth, interesting, leads to a paper at a conference or a journal, creates a cool 60 second video, leads to a business or patent, leads to future work (MS or MQP), and wins the outstanding MQP award.

With these criteria being used as guidelines, we then used various decision methods to narrow down the potential MQP choices. The first decision method we used was a decision matrix to generate a numbered score for each idea. Using the decision criteria, we created weights, scored each project idea in the criteria categories, and calculated the final scores for each project idea. There were distinct number separations for all ideas except for the fall detection system, posture monitoring system, and skin impedance spectrometer which scored 65, 65, and 47 respectively. As a result, the team decided to combine the fall detection system and posture monitoring system into one device, not only due to its application in the health field, but also for the various applications the device could target.

Using the fall detection and posture monitoring system as the idea for our MQP, we proceeded to develop several functions the device could include. From this second brainstorming period, we developed several applications the device could target. These include the elderly, hospital patients, active individuals, people under the

influence, parents and children, students, and office workers. This system is composed of an inertial measurement unit (IMU) and will alert the user if they have poor posture and will also alert a caretaker if the individual wearing the system has fallen.

Once we determined what the functions of the device would be, the next step was choosing the components that would satisfy each of the device's functions. We used a microprocessor to control the entire system. The microprocessor needed to be compact, inexpensive, lightweight, easy to use, and have enough memory to control all the system requirements. Initially we chose the TI CC2650MODA, a microprocessor that met most of the specifications. However, as we progressed through the system integration, difficulties programming the SPI and Bluetooth communication became prominent. There were little resources online to help with the troubleshooting process. As a result, we switched to the ATMEGA328-PU microcontroller.

For fall detection and posture monitoring, the team wanted a device that had the capability of measuring the position of a human body in space and in real time. Initially we explored accelerometers and gyroscopes separately, but after researching IMU's, decided to focus research there. We compared individual accelerometers and gyroscopes to IMU systems and decided to choose the LSM9DS0, an IMU, sold through Adafruit.

With numerous system functions it was crucial that we found a power source that not only had the ability to support all of the features, but also was lightweight enough to be carried on the waist of a human body. We explored four battery options and decided on the 3.7V Lithium Polymer Ion Battery, as it was relatively small, lightweight, and had a capacity of 500mAh.

To control the system, we decided to use Bluetooth Low Energy (BLE) along with a compatible phone application. The first Bluetooth module we researched was the HC-06 which is compatible with Arduino. The second module we researched was the RedBear BLE Shield which is also compatible with Arduino, however, it comes in the form of a shield. Due to the very similar specifications, we chose the HC-06 module as it was a lower cost and would save space on the future PCB with the system components.

After choosing the necessary components, we proceeded to the construction, testing, and debugging stages of our project. Here we constructed the physical systems for each of the devices functions and wrote the necessary code to control each of these functions. The accelerometer records data in  $\text{m/s}^2$  and alerts the user and caregivers when their motions increases outside the threshold of  $6.9 \text{ m/s}^2$  to  $10.2 \text{ m/s}^2$ . If the

threshold is met, the phone application will display the message “Fall Detected,” sending a text message to the caretaker with the individual’s location. Additionally, the data can be visualized through the application or a Python script, allowing for future data manipulation.

For posture tracking, the IMU continuously saves the values of the x, y, and z coordinates relative to its position in space. The gyroscope records data in degrees per second (dps) and alerts the user and when their posture leaves the threshold of -4 dps to 4 dps. If the threshold is met, the phone application will display the message “Stand Up Straight” to alert the user that they need to adjust their posture. These values can then be displayed on a graph and the data can further be analyzed and manipulated. The overall system works as expected, and we believe that with more refining, the project has the ability to become an on-market product.

Our project comprised of four phases, as outlined in Figure 1. Between August 2018 and October 2018, Phase 1 and Phase 2 were completed, and Phase 3 was initiated. Phase 3 was completed between October 2018 and December 2018. Phase 4 was initiated in January 2019 and will be completed in March 2019.

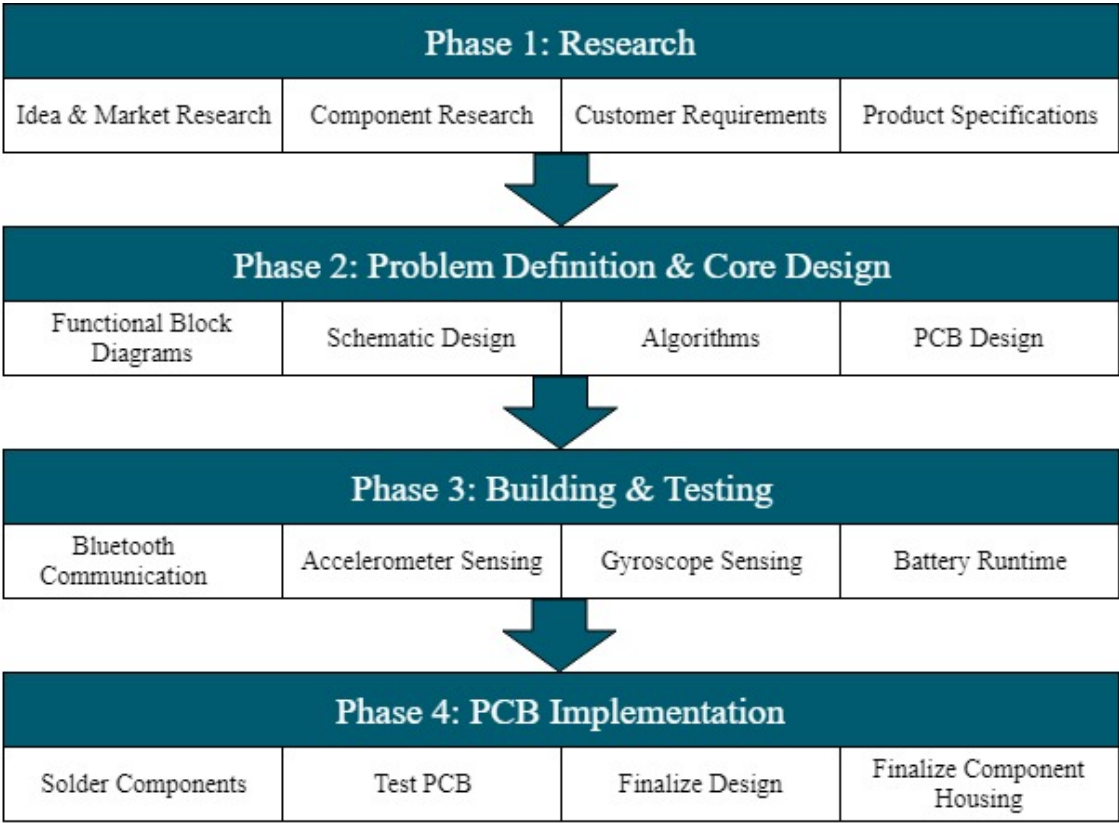


Figure 1: Project flow chart

# Chapter 1

## Introduction

The topic of our Major Qualifying Project (MQP) was analog applications. Within this discipline, we were able to brainstorm our own ideas based off evaluation criteria that we determined. Our team consisted of three individuals whom all had varying backgrounds within our majors of Electrical and Computer Engineering. Throughout the first few weeks of the project, we began to propose various ideas to one another and soon, two stood out.

Two different problems amongst the health industry in the United States are that people are susceptible to falls as well as poor posture. While these are two separate issues, we decided as a team that one device could be the solution to both of these problems. The leading cause of death in the geriatric population, according to the Center for Disease Control, is falling [1]. These tragedies could be prevented through postural corrections that have been shown to improve balance. Equilibrium and balance have shown to directly decrease with degeneration of a person's posture [1]. While the elder population would be a main target for our device, fall detection and imperfect posture are widespread issues that can be applied to a wide variety of individuals.

Per our advisor's suggestion, we began to research various target populations for our device other than the geriatric population. Each year, three million people are treated in emergency departments for fall injuries [2]. Additionally, 90 percent of the United States' population lean forward with their necks, which often gets worse when working on a computer, driving or watching television in bed [3]. To aid in this research, we added several different use cases for our device other than the elderly.

We decided on a fall detection and posture monitoring device that would utilize two sensors to build an overall smart system. The initial goal of our team was to develop a working, wireless system, in which various sensors were able to transmit data to a phone application via Bluetooth. The first sensor would be an accelerometer used to monitor the acceleration of the user and the second would be a gyroscope used to measure the posture of the individual. To accomplish this goal, we designed, implemented, and tested our small device to ensure that it properly detected a fall or a person's poor posture and sent a notification to the phone application via Bluetooth.

# Chapter 2

## Brainstorm

### 2.1 Proposed Projects

After deciding that the project would target the healthcare industry, we began to propose several ideas to each other based off our own interests. As a result, a list of ten project ideas were brainstormed and then discussed. We then narrowed the initial project ideas down to three solid ideas which can be categorized into the following:

- Fall Detection System
- Posture Monitoring System
- Skin Impedance Spectrometer




#### 2.1.1 Fall Detection System

All around the world, falls are a major public health problem. Around 646,000 fatal falls occur each year, making it the second leading cause of unintentional injury and death. Death rates due to falls are the highest among adults over the age of 60 years [4]. In 2015, the total medical costs for falls totaled more than \$50 billion while Medicare and Medicaid covered around 75% of these costs [2]. In addition, the implementation of effective fall prevention strategies would cause a 20% reduction of falls among children ten and under. This would result in a net savings of over \$120 million dollars each year [4].

One of the project ideas was to design a device that monitors whether a person has fallen. If a fall is detected, loved ones, caregivers, or medical personnel, would be notified and if the person did not respond within a certain amount of time, help would be sent to them. This system would use an accelerometer and a microprocessor to interpret the data. The data would then be sent to a mobile application

in order to display the data and send notifications. There are several fall detection monitors in the market, however, they are expensive and some require the use of a landline or the push of a button. Some of these competitors can be seen in Table 2.1. While these devices are similar, they are expensive and require a monthly fee after purchase.

Table 2.1: Fall detection device competitors

GoSafe [5]	GoLiveClip [6]	Sense4Care [7]
		
\$99.95 (device fee) \$44.95 per month	\$299 (device fee) \$14.95 per month for app	\$174
Wearable pendant Fall detection capabilities Autoalert capabilities Requires a landline	Clips onto clothing Fall detection capabilities Pairs with a smartphone Auto-alert capabilities	Clips onto clothing Fall detection capabilities Pairs with a smartphone Auto-alert capabilities

### 2.1.2 Posture Monitoring System




The term posture is used to describe how a person's body is positioned when they are either sitting, standing, or lying down. Good posture means that a person's bones are properly aligned and their muscles, joints, and ligaments can work as nature intended [8]. It also means that their vital organs are in the right position and can function at peak efficiency. Bad posture can cause more problems than most people realize. One in two Americans have a musculoskeletal condition, costing an estimated \$170 billion in annual treatment. Expenditures in lost wages are estimated to average over \$850 billion [8].

A proposed posture monitoring system would comprise of a sensing module that is attached to someone's upper or lower back and could monitor the position of the spine. The sensing module would need to be self-powered, compact, and robust to be competitive in the market. It would also have to be able to transmit real-time data to a smartphone or control hub. Looking at some of the current posture-monitoring



systems in the market, we were able to gauge the different types of competition (See Table 2.2).

Table 2.2: Posture monitoring device competitors

Upright Go [9]	Lumo Lift [10]	Prana [11]
		
\$99.99	\$79.99	\$149.99
Small, comfortable and discreet. Attaches to back with a hypoallergenic adhesive.	Consists of a tracker and a small, square magnetic plate. Tracker houses the internal components. Plate holds the Lift in place on users clothes.	Combines breath and posture tracking. Supports the interconnection between good breathing and good posture.

### 2.1.3 Skin Impedance Spectrometer




The Skin Impedance Spectrometer was a previous MQP project in 2016. The purpose of this project was to design a low power system to measure skin impedance across a frequency range of 5 kHz to 100 kHz. The design included a sensor, an impedance analyzer, and a microcontroller that communicated through Bluetooth [12]. Spectrometers have a wide range of usages such as bio-electrical impedance analysis, diabetes, pain and exertion measurement, as well as skin cancer. It is also a convenient device for family doctors to detect their patients' skin impedance rather than needing to go to a hospital [13].

Our main focus was to work with the the skin impedance meter and make several improvements to the previous system such as increasing the accuracy, improving the communication, and including a smart device user interface. Our first improvement would be to replace the AD5933, a 12-bit impedance converter network analyzer, to either an MCU or FPGA in order to increase the power efficiency. Another improvement would be to use a better amplifier and to include automatic detection.

Another optional advancement would be to include smart communication.

There are several skin response sensors in the market. However, these sensors are expensive and require some professional knowledge about physiology. Table 2.3 shows some of the competitors for the skin impedance spectrometer.

Table 2.3: Skin impedance spectrometer competitors

NeuLog Skin Sensor [14]	Mindfield eSense Skin Response Sensor [15]	3B Scientific Skin Resistance Box [16]
		
\$143	\$99	\$280
Measures the conductivity of skin. Senses sudden noise, smell, touch, pain, and sounds.	Gives exact feedback about momentary stress levels.	Sensor measures the resistance of a person's skin as influenced by external factors.

# Chapter 3

## Decision Methods

### 3.1 Evaluation Criteria

We prepared a list of criteria to determine which of the ideas were most suitable for our project. This allowed us to conduct an analysis of each idea and compare them with each other. Below is each criteria we used for analysis.

#### **Relevant and New to Industry**

One of the most important criteria we considered when evaluating ideas was that they had to be new and relevant to the industry. We wanted to find topics within an industry that were creating a lot of ‘buzz’ because they were new and innovative ideas. While a particular topic may be relevant to an industry, it may also be too saturated. We did not want to work on a project that had been completed several times before in the industry, since it would not have any room for improvement.

#### **Solves an Interesting and Important Problem**

One criteria that we considered most important was that the project would solve an interesting and important problem. We wanted our project to be meaningful and beneficial to society. While the project had to be interesting to each member, it also had to solve an important problem as well as aid various individuals.

#### **Works and Functions on Time**

While we wanted to complete an advanced project, it was important that we kept in mind our three academic term time frame. We defined a working and functioning project as one that could be demonstrable to an audience as well as exceeded the standards of all of the set criteria for the project. It was also important to understand the time frame since there are several components in the design including the prototyping, testing, and writing process of the project.

#### **Improves Skills and Technical Growth**

It was important to us that we work on a project where we have the flexibility to be exposed to technical aspects that we had not yet encountered. Therefore, it was

important for the project to be technically diverse so that each member had the opportunity to develop a variety of skills.

### **Interesting for Team**

The project idea had to be interesting for each member as it would ensure that we were motivated throughout the year. If we were motivated and driven towards the end goal, the project would have a higher probability of being successful.

### **Leads to Paper at Conference or in a Journal**

One of our goals was to publish a paper at a conference or in a journal. While this may not have been our most important goal, we believed that this would be a major achievement and honor.

### **A Cool 60 Second Video**

We believed that a cool, 60 second video would provide a brief overview of the entire project. It would be able to describe our project clearly to a non-professional audience as well as attract them and make them interested in learning more about the design.

### **Business / Patent / Start up**

Another one of our goals was to either start a business and/or apply for a patent. While this may conflict with the other goal of publishing a paper at a conference or journal, we believed it should be considered if the business evaluation of the project is positive. In addition, one member of the team was pursuing a minor in business, so this could also be an opportunity to use their skills in pursuing a patent.

### **Leads to Future Work (MS, MQP)**

We believed another important goal was that the project would lead to future work, such as for another MQP or a Masters degree. We thought this was important as it would contribute to the WPI community and allow future students the foundation to add their own ideas by completing further research on our project.

### **Outstanding MQP Award**

Lastly, one of our final goals for the MQP was to try and win the Outstanding MQP Award. This award would be an honor for our team as it would indicate that the idea made a significant contribution to the community. The award would also commend our three-term effort and represent all of our hard work.

## 3.2 Decision Matrix

To judge the ten ideas we brainstormed, we used a decision matrix. The full decision matrix can be found in Appendix A. A decision matrix is a logical approach to narrowing down project ideas. It is beneficial as it weights the choices purely based on numbers, excluding personal opinions [17]. However, this method is also disadvantageous as it may be unsuccessful at providing a clear choice since options may be close in value due to the weighted criteria.

The decision matrix used the weighted decision criteria to rank the ten brainstormed project options. We ranked each criterion as either a one, two, or three; where a one was least important, two was medium importance, and three was most important. Table 3.1 shows the decision criteria weights.

Table 3.1: Decision criteria weights

<b>Most important (x3)</b>
Relevant & New to Industry
Solves an Interesting & Important Problem
Want it to Work Properly on Time
Improves Skills & Technical Growth
Interesting for Team Members
<b>Medium Importance (x2)</b>
Paper at a Conference or in a Journal
A Cool 60 Second Video
<b>Least important (x1)</b>
Business / Patent / Start up
Leads to Future Work (MS, MQP)
Outstanding MQP Award

To calculate the scores, the weights of the criteria and the rankings given to each project idea were multiplied and the total was summed. The higher score indicated that the project better fit the criteria described.

Table 3.2 shows the final scores for the top three project ideas. The top three ideas were the fall detection system, the posture monitoring system, and the skin impedance spectrometer.

Table 3.2: Decision matrix scores

Project Idea	Score
Fall Detection System	65
Posture Monitoring System	65
Skin Impedance Spectrometer	47

### 3.3 Final Decision

Since both the fall detection system and the posture monitoring system tied as a result of the decision matrix and had a significantly higher score than any of the other designs, we decided to combine the two ideas and make a smart system. Both of these ideas were relevant to the industry and solved an important problem. Not many devices in the market currently connect to smartphones, and none combine the technology of an accelerometer and gyroscope to detect both falls and imperfect posture. We believed that we could make the device work and function within the time constraints and that it could be easily demonstrated working in a video. While this topic was interesting to our team, it was also believed that the construction of the device would improve the skills and technical growth of each member.

### 3.4 Use Cases

After determining that our project would focus on building a fall detection and posture monitoring device, it was important that we determined the best position of the device to ensure that both functionalities would be accurate. We decided that the ideal positioning for the device would be placed on the waist of the user. To determine this, we researched the optimal placement of our sensors on the human body.

The most challenging aspect of fall detection is the distinction between falls and Activities of Daily Living (ADLs) such as sitting, standing, and walking since falls typically occur while performing daily activities. According to research conducted in 2017, the optimal placement of the sensor for fall detection would be either the waist or the thigh (Figure 3.1) [18].

Classifier	Sensors	Accuracy (%)	Sensitivity (%)	Specificity (%)
(a) J48	Head	96.48	91.06	95.76
	Chest	97.53	97.70	97.31
	Waist	97.96	97.99	97.94
	Wrist	93.71	94.78	92.37
	Thigh	<b>98.24</b>	98.71	97.67
	Ankle	97.45	97.49	97.40
(b) IBk	Head	93.70	92.84	94.77
	Chest	97.45	97.28	97.67
	Waist	<b>98.61</b>	98.85	98.30
	Wrist	89.74	84.13	96.77
	Thigh	96.42	94.20	99.19
	Ankle	95.58	93.34	98.39
(c) RC	Head	97.17	98.57	95.41
	Chest	98.61	99.07	98.03
	Waist	<b>98.89</b>	99.28	98.39
	Wrist	94.63	96.35	92.47
	Thigh	98.77	99.00	98.48
	Ankle	98.77	98.85	98.66
(d) RF	Head	96.77	99.36	93.51
	Chest	98.61	99.28	97.76
	Waist	<b>99.28</b>	99.64	98.84
	Wrist	95.62	98.28	92.29
	Thigh	99.20	99.43	98.93
	Ankle	98.77	99.07	98.39
(e) SMO	Head	97.29	97.92	96.49
	Chest	98.89	99.28	98.39
	Waist	99.36	99.50	99.19
	Wrist	96.78	97.71	95.61
	Thigh	<b>99.48</b>	99.21	99.82
	Ankle	98.57	98.85	98.21

Figure 3.1: Evaluation of optimal sensor placement for fall detection

This study performed feature extraction and classification based on the data acquired from a single sensor unit placed on a specific body part. The study investigated several sensor locations including the head, chest, waist, wrist, thigh, and ankle. Additionally, it used several classification algorithms to determine that the waist and the thigh were the optimal locations [18].

Posture is similar to fall detection as it is important to identify the distinction between falls and ADLs. To identify the optimal sensor placement, we first needed to identify what is considered poor posture. We discovered that it is defined when an individual tilts their head forward or when their anterior pelvic tilts (Figure 3.2).

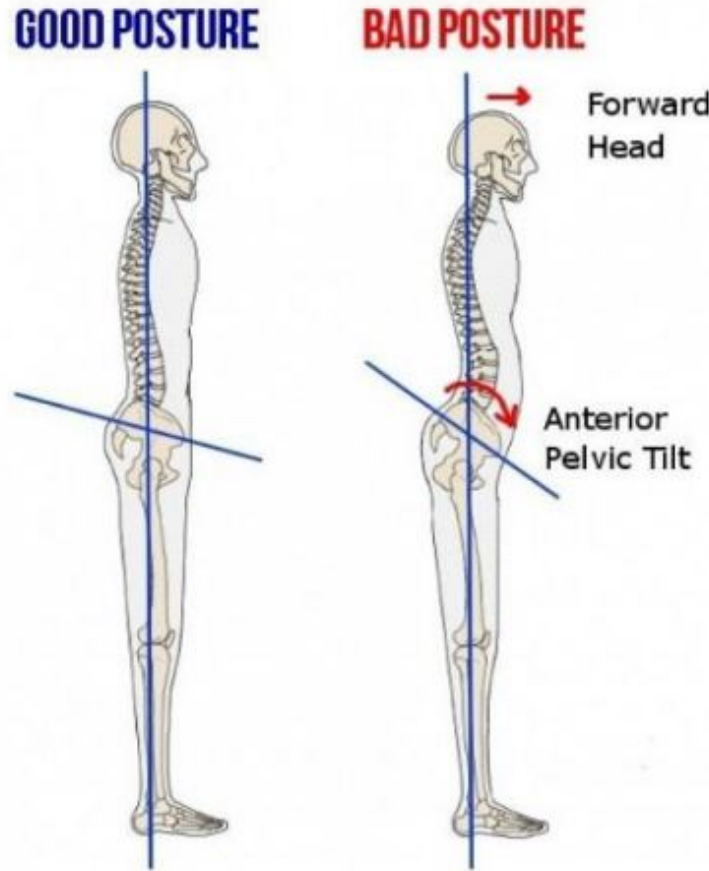


Figure 3.2: Posture diagram for optimal sensor placement

Since the most feasible sensor placement for fall detection was the waist, we decided that the device would be positioned on the belt of the user since it would accurately detect a fall as well as detect whether the user's anterior pelvic or head was titled.



# Chapter 4

## Background Research & Product Specifications

### 4.1 Target Markets

Using the fall detection and posture monitoring system as the basis for the MQP project, we proceeded to target several markets where the device could possibly succeed. From this second brainstorming period, we developed eight target markets where both the fall detection or posture monitoring technology could be applied.

#### 4.1.1 Fall Detection and Prevention System

In order to attract the proper market, it was important to determine who would benefit from wearing a fall detection device. While falls are most commonly associated to the elderly, we did not want the device to only focus on a single group of people. Below are the different categories of people that we believed would benefit from the device.

##### **Elderly**

One of the main target markets for the device was to target it towards the elderly, including those in a nursing home and those who live alone (Figure 4.1). This would provide urgent care to people if needed and reduce the medical costs associated with falls. If falls are not dealt with in time, they can cause functional impairment and a significant decrease in a person's mobility, independence, and life quality [4]. In addition, some falls may be preventable and it is important to recognize when a person may be about to fall and prevent the situation from occurring.



Figure 4.1: Elderly fall detection target market

### Hospital Patients

Another market would be to target hospital patients of all ages whom are under the risk of falling, so that if they got up from their room and happened to fall, hospital staff would be notified immediately (Figure 4.2). Patients who fall while in a hospital add health care costs for both the patient and the facility. Also, they will have to stay longer and use more hospital resources, taking up space and time. Hospital patient are at a higher risk of falling due to a variety of factors including muscle weakness, medications, and dizziness [20]. It is important to both detect a fall as well as to prevent them by recognizing when a person may be more susceptible to a fall.



Figure 4.2: Hospital patient fall detection target market

### Active Individuals

Active individuals would be another main target for the fall detection device. These individuals could include rock climbers, hikers, and bikers who all participate in risky activities (Figure 4.3). These individuals have a high risk for falling and these

falls may be life-threatening due to their environments. A fall detection system would allow them to participate in the activities that they love and give them peace of mind that if an accident were to happen, someone would be notified to provide them with help.



Figure 4.3: Active individuals fall detection target market

### **People Under the Influence**

Another group of people would be people under the influence of drugs or alcohol as the fall detection system could sense that their walking patterns are not normal and indicate that they may be susceptible to a fall. A person who is under the influence is more likely to fall due to their visual impairment and loss of balance (Figure 4.4). Therefore, it would be beneficial for them to wear one of these devices in case they were to accidentally fall and injure themselves.



Figure 4.4: Under the influence fall detection target market

### **Parents and Children**

Finally, last group of people that we could target with the device would be children and their parents (Figure 4.5). Many parents would have peace of mind knowing

that their children could go out and play and that if they were to fall and injure themselves, they would be notified immediately.



Figure 4.5: Children fall detection target market

### 4.1.2 Posture Monitoring System

It was also important to determine the different market opportunities for a posture monitoring system. While some of these markets overlap with the fall detection system, it was still important to point out the different types of people that would benefit most from monitoring their posture. Below are the different groups of people:

#### **Children**

The spine is being developed at a young age, and most posture problems start in the early years of a person's life [21]. Therefore an important target audience is children, especially children who spend time sitting in a specific position such as playing video games or reading (Figure 4.6). If any abnormalities in posture are detected in the early years of a person, it is possible to reduce costs incurred in treatment later in life.



Figure 4.6: Children posture target market

### Students

Students spend a large amount of time sitting in a certain position at a desk or in a meeting; therefore, they are at a high risk of having posture problems. Therefore another important target audience was students. Additionally, students constantly wear heavy backpacks which causes problems with their spine (Figure 4.7). Constant posture monitoring and correction can result in reducing treatment costs.



Figure 4.7: Student posture target market

### Office Workers

Office workers are similar to students as they spend extended periods of time sitting at a desk or in a meeting, therefore, they are at a high risk of having posture problems

(Figure 4.8). A great number of professionals are too busy trying to get ahead in their career that they don't have time to concentrate on discrete problems such as posture. Therefore, a device that can take care of this problem without a visit to the doctor may prove to be a good idea.



Figure 4.8: Office worker's posture target market

### Data Collection

Another aspect of this application can be to collect real-time data on a specific person's posture and analyze the data to see if there are any patterns. This can help the team identify if there are some specific habits a person has that can cause him or her to have posture problems. This can also be useful for medical researchers that are looking into analyzing posture problems in detail.

## 4.2 System Requirements

When deciding on which components will best suit our system, we first needed to decide the most important requirements for our product. We decided to create specific product requirements for each component of our product.

### 4.2.1 Accelerometer

Below are the system requirements for the accelerometer component in our device:

- Sensitivity of at least 1 mg/LSB
- Range of at least 4g
- Low cost ( $< \$4$ )
- 8 mA worst case current usage
- Small size
- $-25^{\circ}\text{C}$  to  $40^{\circ}\text{C}$  temperature range
- I2C or SPI communication

An important requirement for the accelerometer of our system is that it must have high sensitivity so that it can precisely detect a fall and measure when a person is more susceptible to a fall. It also needs to have a suitable range so that it will be able to correctly detect a fall. The accelerometer we choose must have a low cost so it is as cost efficient as possible. In addition, we also want our accelerometer to have a low current usage so that the user would not have to recharge the battery often. Another requirement is that the accelerometer is on the smaller size so that it does not make our product too large since it will be worn by the user. Finally, the accelerometer needs to have a suitable temperature range as well as SPI or I2C communication.

### 4.2.2 Gyroscope

Below are the system requirements for the gyroscope component in our device:

- Sensitivity of at least 10 mdps/LSB
- Low cost ( $< \$4$ )
- 8 mA worst case current usage
- Small size
- $-25^{\circ}\text{C}$  to  $40^{\circ}\text{C}$  temperature range
- I2C or SPI communication

An important requirement for the accelerometer of our system is that it must have high sensitivity so that it can precisely detect a change in a person's posture. It also needs to have a high output range. The gyroscope we choose must have a low cost so that the device is as cost efficient as possible. Additionally, we also want our gyroscope to have a low current usage so that the user would not have to recharge

the battery often. Another requirement is that the gyroscope is on the smaller size so that it does not make our product too large since it will be worn by the user. Finally, the gyroscope needs to have a suitable temperature range as well as SPI or I2C communication.

### 4.2.3 Microcontroller

Below are the system requirements for the microcontroller component in our device:

- Low cost ( $< \$10$ )
- 12 mA worst case current usage
- Small size
- At least 12 GPIO pins
- $-25^{\circ}\text{C}$  to  $40^{\circ}\text{C}$  temperature range
- I2C or SPI communication

An important requirement is that the microcontroller we choose must have a low cost so that our product is as cost efficient as possible. In addition, we also want our microcontroller to have a low current usage so that the user would not have to recharge the battery often. It would be feasible that the user could go at least a day without needing to recharge the battery. The microcontroller will use a lot of the power so we want the current usage to be as low as possible. Another requirement is that the microcontroller must be on the smaller size so that it does not make our product too large since it will be worn by the user. Additionally, our microcontroller needs to have a suitable temperature range and can be integrated with Bluetooth. It also needs to be able to interface with SPI or I2C to accommodate both the accelerometer and gyroscope and have a sufficient number of GPIO pins.

### 4.2.4 Wireless Communication

Below are the system requirements for the microcontroller component in our device:

- Range of at least 2 meters
- Low cost ( $< \$4$ )
- Integrates with phone



An important requirement for the wireless communication module is that it must be able to interface with a phone or application. An important part of our system is that the data can be sent to an app where it can be processed and viewed by the user. Another requirement is that the module must have a range of at least 2 meters. While this is a close range, we believe that the user will always have their phone close to their body.

### 4.2.5 Battery

Below are the system requirements for the battery component in our device:

- Rechargeable
- Weight less than 15 grams
- Low cost ( $< \$10$ )
- Nominal voltage of at least 3.3V
- Small size
- Capacity of at least 500 mA/h

An important requirement for the battery in our system is that it must be rechargeable so that the user will not have to replace the batteries often. Another important requirement is that the battery is low cost and low weight to help benefit the user. Additionally, the battery has to have a nominal voltage of at least 3.3V. Finally, depending on the current usage of the other components, the battery needs to have a high enough capacity so that the battery will at least last a day.

## 4.3 Component Research & Specifications

When deciding which components will best suit our system, we used the system requirements we determined in Section 4.2 to help us decide between our options. Below is the analysis of our component decisions based of the requirements we already determined.

### 4.3.1 Accelerometer & Gyroscope

The fall detection and posture monitoring system incorporates a sensing device to track the both the positioning and motion of the user for applications both in fall

detection as well as posture correction for people in everyday life.

To provide the user with relevant information on their posture and motion, we researched the accuracy of an accelerometer, gyroscope, as well as an inertial measurement unit (IMU). An IMU incorporates an accelerometer, gyroscope, and magnetometer in one chip [22]. This is beneficial to the system as we will utilize the accelerometer and gyroscope and it will save space on the PCB and could potentially be a lower cost. We compared the price and accuracy of the three different sensors. The results of the three different component options are shown below in Table 4.1. After comparing the cost, current usage, and additional features of each sensor, we ultimately selected the LSM9DS0 IMU.

Table 4.1: Comparison of sensors

		<b>ADXL362 (Acc) [23]</b>	<b>L3GD20H (Gyro) [24]</b>	<b>LSM9DS0 (IMU) [25]</b>
Cost		\$3.97	\$3.42	\$6.33
Current Usage	Measurement Mode	3 $\mu$ A	6.1mA	6.35 mA
	Standby Mode	10 nA	2 mA	2 mA
	Power Down Mode	—	5 $\mu$ A	6 $\mu$ A
Voltage Supply		1.8V to 3.3V	2.4V to 3.6V	-0.3V to 4.8V
Sensitivity	—	1 mg per LSB	—	—
	Output Range: 245 dps	—	8.75 mdps/LSB	
	Output Range: 2000 dps	—	70 mdps/LSB	
	Linear acceleration FS = $\pm 2$ g	—	—	0.061 mg/LSB
	Linear acceleration FS = $\pm 16$ g	—	—	0.732 mg/LSB
Size of Chip		12 mm <sup>3</sup>	9 mm <sup>3</sup>	16 mm <sup>3</sup>
Temperature Range		-50 °C to 150 °C	-40 °C to 85 °C	

This system uses the LSM9DS0 to track the users movement as well as their posture (See Figure 4.9). The sensor combines an accelerometer, a gyroscope, and magnetometer and transmits data to the microcontroller through a digital I2C output.

Additionally, this sensor satisfies our requirements of a least a 1 mg/LSB accelerometer sensitivity and 10 mdps/LSB gyroscope sensitivity. The accelerometer output range has a wide range of  $\pm 2$  g to  $\pm 16$  g which satisfies our requirement of at least  $\pm 4$  g. The worst case current usage also satisfies our 8 mA requirement and the combined cost of both the accelerometer and gyroscope is under eight dollars. Although all of the sensor options satisfied our requirements, the LSM9DS0 combines both the accelerometer and gyroscope which will be a lower cost, lower current usage, and save us space on our PCB. Additionally, the LSM9DS0 has a higher sensitivity range for both the accelerometer and gyroscope measurements.



Figure 4.9: LSM9DS0 component

Although the sensor is capable of handling different sensitivities, the ranges that we chose can be seen in Table 4.2.

Table 4.2: Sensor characteristics

	Accelerometer	Gyroscope
Range	$\pm 2$ g	$\pm 245$ dps

### 4.3.2 Microcontroller

To process the data received from the sensor, we investigated a variety of microcontrollers. The microcontroller needed to have the ability to control every peripheral used by the fall detection and posture monitoring system. The data processed by the microcontroller will be sent to the user wirelessly through Bluetooth. While some microcontrollers have built in Bluetooth modules, others can be interfaced with an external Bluetooth modules through either SPI or I2C. Additionally, we considered the microcontroller's ease of use, the amount of instructional resources available, and the cost. Table 4.3 compares these characteristics against two different microcontrollers.

Table 4.3: Microcontroller comparisons

	<b>CC2650MODA (Bluetooth Integrated) [26]</b>	<b>ATMEGA328-PU (No Bluetooth Integrated) [27]</b>
Cost	\$9.90	\$3.25
Current Usage	9.4 mA (Active mode) 1 $\mu$ A (Standby mode) 100 nA (Shutdown mode)	19 mA (Active mode)
Size of Chip	16.9 mm x 11 mm	35.544 mm x 7.62 mm
Interface	I2C, UART, SPI	
Temperature Range	-40 to 85 C degree	
Integrated with BLE?	Yes	No
GPIO Pins	15	Digital: 14 Analog: 8
Embedded with RTOS?	Yes	No
Supply Voltages	1.8V to 3.8V	6V to 20V

Initially, we selected the CC2650MODA for the device due to the integrated Bluetooth capabilities and low cost. However, after encountering multiple problems when trying to set up the SPI communication between the MCU and sensor, we chose to switch to the ATMEGA328-PU and found an external Bluetooth module (See Figure 4.10). While the ATMEGA328 does not satisfy our 12 mA worst case current usage and is a much larger size than the CC2650MODA, it does satisfy our 12 GPIO pin minimum requirement as well as our low cost. Even though these requirements were not met, it was important to consider our three-term time frame as well as the skillset of the team members. We decided as a team to use the ATMEGA328 as the device's microcontroller as it was easier to use and had many more instructional resources available.

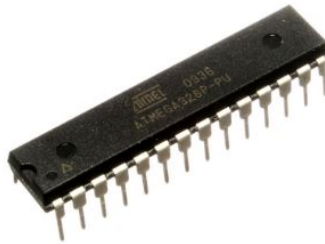


Figure 4.10: ATMEGA328-PU component

### 4.3.3 Wireless Communication

The ATMEGA328-PU itself does not have built in Bluetooth capabilities, so we decided to use an external Bluetooth module to send data to the phone app. We explored two different Bluetooth modules, the RedBear BLE Shield v2.1 and the HC-06 module sold through Adafruit. Both were very similar, however, we chose the module over the shield due to the additional space the module made available on the PCB and the module was easier to integrate with the phone application. Table 4.4 shows the comparison between the two different Bluetooth modules.

Table 4.4: Bluetooth comparison

	RedBear BLE Shield v2.1 [28]	HC-06 [29]
Price	\$19.90	\$3.99
Dimensions	53.44mm × 62.17mm × 11.17mm	27mm × 13mm × 2mm
Voltage	5V	3.6V to 6V

The final product uses the Arduino ATMEGA328-PU to process the data and the HC-06 module to transmit the data wirelessly and the data can be visualized from an Android App (See Figure 4.11). The Bluetooth module meets the requirement of a two meter range and is inexpensive compared to the RedBear BLE Shield v2.1.



Figure 4.11: HC-06 module

### 4.3.4 Battery

An important part of the fall detection and posture monitoring system was the power source. We needed a power supply which would not only be able to power the whole system, but also keep it running for a reasonable amount of time. It was also important that the battery was lightweight so that the user is able to carry it on their back for a long period of time. Finally, the battery needed to be at a low cost to keep the overall of price of our system low. Table 4.5 displays the comparison between the different batteries that we researched.

Table 4.5: Battery comparisons

	<b>3.7V Lithium Polymer Ion [30]</b>	<b>RJD2450 [31]</b>	<b>Panasonic NiMh AAA [32]</b>	<b>ML2430 [33]</b>
Current Capacity	500mAh	200mAh	800mAh	100mAh
Cost	\$7.95	\$13.18	\$2.81	\$3.82
Weight	10.5g	6.5g	2.9g	4.1g
Nominal Voltage	3.7V	3.7V	1.2V (for 1)	3V

Based on the above specifications we selected the 3.7V Lithium Polymer Ion battery to power the system due to its nominal voltage, light weight, and low cost (See Figure 4.12). The cost is under the \$10 requirement as well as the weight is below 15 grams. The battery is rechargeable and its nominal voltage is greater than 3.3V.



Figure 4.12: 3.7V Polymer Ion battery component

After selecting our battery and system components, it was important that we conducted a power analysis to determine the runtime of the overall system. To start, we calculated the total current draw from all of the components in our system (See Table 4.6)

Table 4.6: Total current usage of components in the system

Component	Current Usage (Worst Case)
Sensor	6.35 mA
Microcontroller	19mA
Bluetooth Module	8mA
<b>Total</b>	<b>33.35mA</b>

Next we divided the current capacity of the battery by the total current usage of the system as seen in Equation 4.1.

$$Runtime = \frac{Battery\ Capacity}{Total\ Current\ Usage} \times \frac{1}{PowerBoost\ 500\ Efficiency} \quad (4.1)$$

The results indicated that with the worst case current usage, the battery will last up to roughly 14 hours which is less than a day. However, we agreed that the average person is only awake for 14 to 18 hours a day. Therefore, a runtime of 14.39 hours will provide the user with a one days use. To calculate the overall time to charge the battery once it has completely died, we will divide the battery capacity by the charge rate current in Equation 4.2. The standard charge rate current of a micro-usb port is 500mA.

$$Charge\ Time = \frac{Battery\ Capacity}{Charge\ Rate\ Current} \quad (4.2)$$

The maximum time to charge the battery was calculated to be 1 hour and with a 20% efficiency lost, the total charge time would be around 1.2 hours. This is a very short charge time and would not inconvenience the user if they needed to charge the device during the day.

# Chapter 5

## Financial Analysis

A key, non-technical aspect of our project was the financial analysis. The Electrical and Computer Engineering Department at WPI requires students to complete an engineering design course. One of the outcomes of this course is to be able to demonstrate a working knowledge of financial, scheduling, and administrative elements of the design process. In this course, students create and adhere to project schedules, as well as estimate possible Return-of-Investment (ROI) if their product were to be manufactured [34]. We decided to utilize the material learned in this course and apply it to our fall detection and posture monitoring system. This chapter will look at the bill of materials for bulk production, initial investment required costs, as well as the ROI analysis.

### 5.1 Bill of Materials

A bill of materials (BOM) is a comprehensive inventory of the raw materials, assemblies, subassemblies, parts and components, as well as the quantities needed to manufacture a product [35]. To summarize, it is the complete list of items that are required to build a product. The bill of materials for the production of a single device is shown in Table 5.1, where costs are assumed for bulk production (>4000pcs) [36].



Table 5.1: Bill of materials

Component	Quantity	Total Price (\$) (4000pcs)
ATMEGA328 MCU	1	0.98
BSS138 Logic Shifter	2	0.12
LSM9DS0 Sensor	1	6.50
PowerBoost	1	6.50
BLE Mod HC06	1	2.86
Rechargeable Battery	1	4.00
LM1117 Regulator	1	0.45
Misc - R,C,LED	-	1.50
PCB	1	0.40
Plastic Case	1	0.60
<b>Total</b>		<b>\$23.91</b>

## 5.2 Initial Investment

An initial investment would be made if we wanted to take the product into market. The numbers seen in Table 5.2. were extracted from a lecture by Professor W. Michalson in ECE 2799.

Manufacturing costs are typically divided into three categories. The first category is direct materials which is the cost of the materials that become part of the finished product [37]. For example, a direct material for our product would be the FR-4 used to make the PCB and soldering lead. The second category is direct labor which is the cost of individual's wages who are physically involved in converting raw materials into the finished product [37]. For example, the wages of the people operating the PCB Assembly machine would be considered direct labor for our system. The third category is the factory overhead or manufacturing overhead which is all of the other costs incurred in the manufacturing activity which cannot be directly traced to physical units in an economically feasible way [37]. An example would be the wages of the people that verify assembly/component soldering as well as the depreciation on the factory equipment. Shipping costs are the costs incurred when the finished product is shipped from the manufacturer to the company's warehouse and then to the customer [37]. The following are the calculations for the required initial investment:

Table 5.2: Initial investment

Bill of Materials (BOM)	= \$23.91
Manufacturer (+25%)	= \$29.89
Shipping to/from Manufacturer (+25%)	= \$37.36
Warehouse (+5%) = Landed Cost	= \$39.23
Slight Profit Target Price	= \$60
3*Total Initial Investment Projection	= (# of units x Landed Cost) + Additional Fees
	= (5000 x 39.23) + 5000
	= \$201,150

### 5.3 Break-even Analysis

Break-even analysis is a technique widely used by production management and management accountants. It is based on categorizing production costs between those which are "variable" (costs that change when the production output changes) and those that are "fixed" (costs not directly related to the volume of production).

Total variable and fixed costs are compared to sales revenue to determine the break-even point, which is the level of sales volume at which the business makes neither a profit nor a loss (Equation 5.1). A fixed cost is an expense that does not change with an increase or decrease in the number of goods or services produced or sold. Fixed costs are expenses that have to be paid by a company, independent of any business activity [38]. It is one of the two components of the total cost of running a business, the other being variable costs.

$$N = \frac{\text{Fixed Cost}}{\text{Unit Price} - \text{Unit Cost}} \quad (5.1)$$

where N is the number of units needed to be sold to break even

For our project, a few of the fixed costs would include paying the engineers, using lab equipment, and renting space. Our conservative pricing and cost analysis has led to the result that around 20,000 pieces would need to be sold before our business could be profitable.

# Chapter 6

## Hardware Design

This chapter details the hardware design of the Fall Detection and Posture Monitoring System and elaborates on each system block.

### 6.1 Overall Design

The overall block diagram of the Fall Detection and Posture Monitoring System is depicted in Figure 6.1. Each module and its specifications are elaborated upon below.

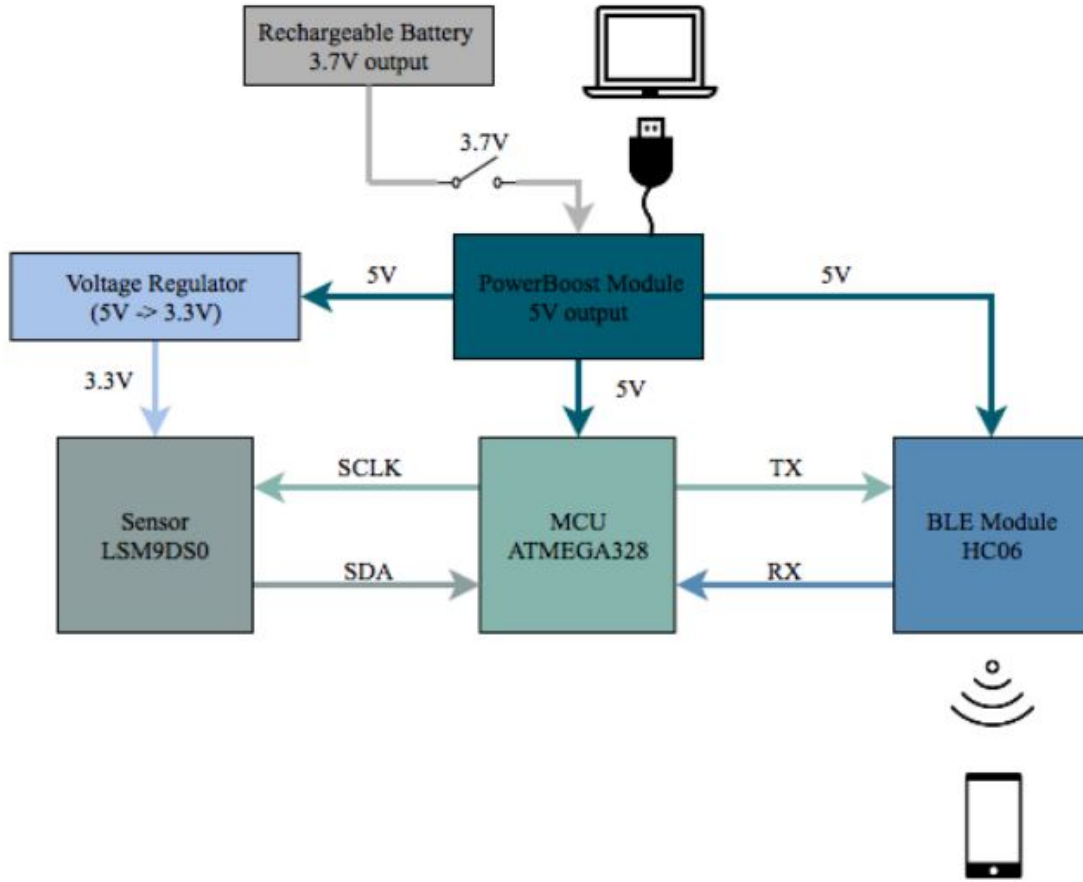


Figure 6.1: System block diagram

## 6.2 System Blocks and Specifications

### 6.2.1 Power Supply

The 5V PowerBoost Charger and the 3.7V Lithium Ion Polymer Battery provided by Adafruit is used to both power and charge the fall detection and posture monitoring system. The benefits of using these components are that it includes a built in boost circuit to provide 5V with the 3.7V power supply so that we do not need an extra voltage regulator in our system. Also, it allows our system to be charged via micro-usb rather than needing an external battery charger. In the design, the team utilizes the micro-usb port to charge the 3.7V Lithium Ion Polymer Battery which then provides the necessary 5V to the ATMEGA328-PU microcontroller (Figure 6.2). A

voltage regulator is also used to scale the 5V in order to provide the necessary 3.3V to the sensor.

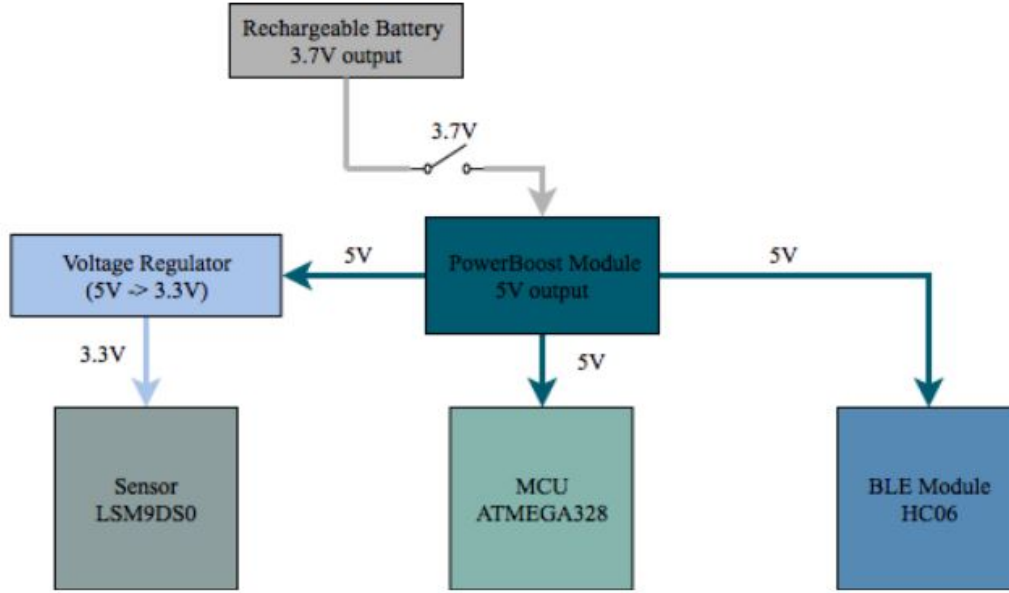


Figure 6.2: Power supply block diagram

### 6.2.2 Microcontroller

We chose the ATMEGA328-PU microcontroller to control the system as it offers a lot of user support and a friendly coding environment. It has 14 GPIO pins, SPI/I2C serial interfaces, a power consumption of 15mA, and an analog supply voltage of 6V. The MCU is powered by the necessary 5V which also supplies power to the Bluetooth module. The system uses a total of 5 digital inputs. Pins PC6 is connected to our switch while pins PD2 and PD3 are connected to the Bluetooth pins Tx and Rx respectively. Finally, pins PC4 and PC5 are connected to SDA and SCL on the sensor respectively.

### 6.2.3 Sensor

We chose the LSM9DS0 as our sensor since it had a 3-axis accelerometer, 3-axis gyroscope, and a 3-axis magnetometer all in one. The sensor will be placed on the waist and calibrated using that point as the zero. The person's orientation

is measured by the sensor and if deviations of more than the specific threshold is detected in the x, y, or z direction, an alert will be sent to the user's smartphone. A voltage regulator will scale the 5V from the power boost and supply the sensor the necessary 3.3V to function. Data is sent to the microcontroller through Inter-Integrated Circuit (I2C) protocol. The data line connects to pin SDA and the clock line connects to pin SCL (Figure 6.3).

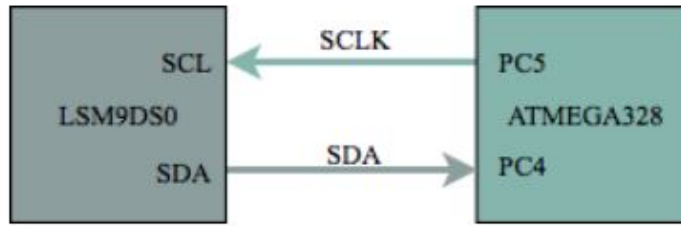


Figure 6.3: Sensor block diagram

### 6.2.4 Bluetooth

We chose the HC-06 Bluetooth module due to its low price and compatibility with Arduino. In addition, it was compatible with MIT App Inventor 2 which is used for the application development. The HC-06 requires 5V as an input voltage which it receives from the 5V from the microcontroller. Also, the Rx and Tx lines are connected to their respective pins on the microcontroller (Figure 6.4).

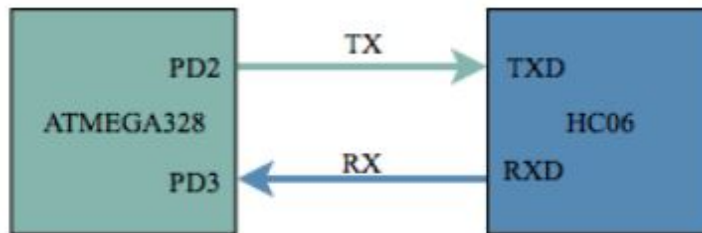


Figure 6.4: Bluetooth block diagram

### 6.2.5 Component Housing

In order to accurately detect whether a person has fallen or that their posture is abnormal, we decided that the components would need to sit on the belt of a person's

back. We designed a small box that has cutouts for the micro-usb charger, a switch to power the system, and an LED indicator light to determine whether or not the system is powered on (Figure 6.5). The box will have a lid that screws into the rest of the box. Additionally, belt loops were attached using Velcro so that a person is able to remove it from their belt when needed.

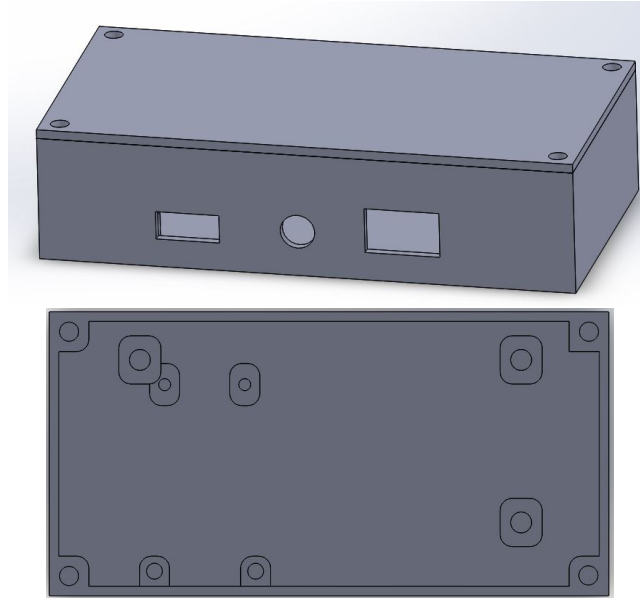


Figure 6.5: Component housing design

The box is 93.2mm x 47.2mm x 23.6mm while the thickness of the box is about a 1/16 of an inch thick. (Figure 6.6). Extra space was added inside the box to ensure that the PCB would fit as well as allow for the wires of the battery to have plenty of space.

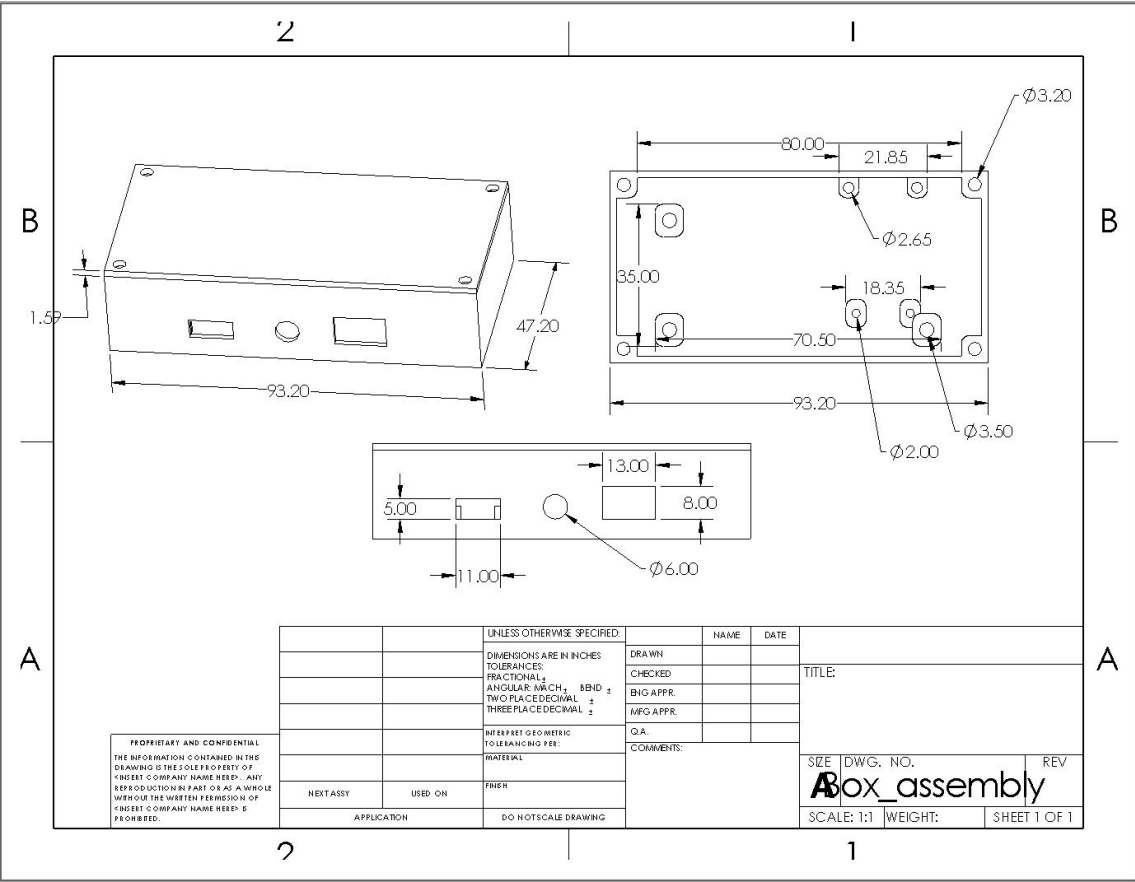






Figure 6.7: Final component case

# Chapter 7

## Software Design

This section will briefly outline the strategies and methods used to code both the Arduino and the Android app. The language used by the Arduino is C++, and the environment coded in is Arduino's own IDE [39]. Arduino code is structured so that there are two main functions that must be modified. The 'setup' function is used to enact initializations that should only happen once at the beginning of the code, and the 'loop' function is used to run processes repeatedly while the Arduino is functioning. The Android app, in contrast, uses MIT App Inventor 2 which is a free, cloud based service that allows people to make their own mobile application. This tool utilizes a block based programming language for users to build apps and run them on an Android mobile device [40].

### 7.1 Arduino Code

The following sections outline the three main parts of the Arduino code for our Fall Detection and Posture Monitoring System. These main sections are the setup of the sensor, I2C communication, and Bluetooth. The full code can be viewed in Appendix B.

#### **Sensor Setup**

The code written to acquire data from the sensor is the simplest part of the Arduino code. Certain variables are declared at the beginning of the code that have values to the Tx and Rx pins being used. These variables are then used with functions from built-in Arduino libraries corresponding to the LSM9DS0 sensor. For the accelerometer sensor, we look at the three axes of the accelerometer data to see if there has been a change and the data is outside of the current threshold. If this is the case, then a byte integer flag is set to 6. For the gyroscope sensor, we convert the data into a three-axes Euler angle. This data is compared to the normalized posture created by the user and if there is an irregular posture is detected, then a different byte integer flag is set to 7. Both the accelerometer and gyroscope code are repeated a single time in the 'loop' function, therefore updating their values every iteration of the loop.

### **I2C Communication**

The LSM9DS0 comes equipped with its own Arduino library that easily implements the communication necessary to sample the accelerometer, gyroscope, and magnetometer. The library allows an object to be declared, which holds all the functions and variables needed to run the accelerometer. After this object is created and the settings initialized in the ‘setup’ function, the accelerometer and gyroscope are looked at using a library specific function in the ‘loop’ code. Following this function, the x, y, and z accelerations and angles are obtained from the object that was declared in the beginning. The code then analyzes the values to determine if the person is walking or positioned in an irregular manner. If this is found to be true, a flag is set to the value of 1, indicating that a problem has occurred.

### **Bluetooth Setup**

The HC-06 module is able to run with its own library, which integrates with the Arduino. To implement this, a serial monitor is initialized and named in the ‘setup’ function. This then allows the Arduino to read values from, or write values to the characteristics built into the serial monitor. In the ‘loop’ function, functions were used to write the values of the various flags, described as seen in Figure 7.1, which then allows them to be sent to the central device (the app).

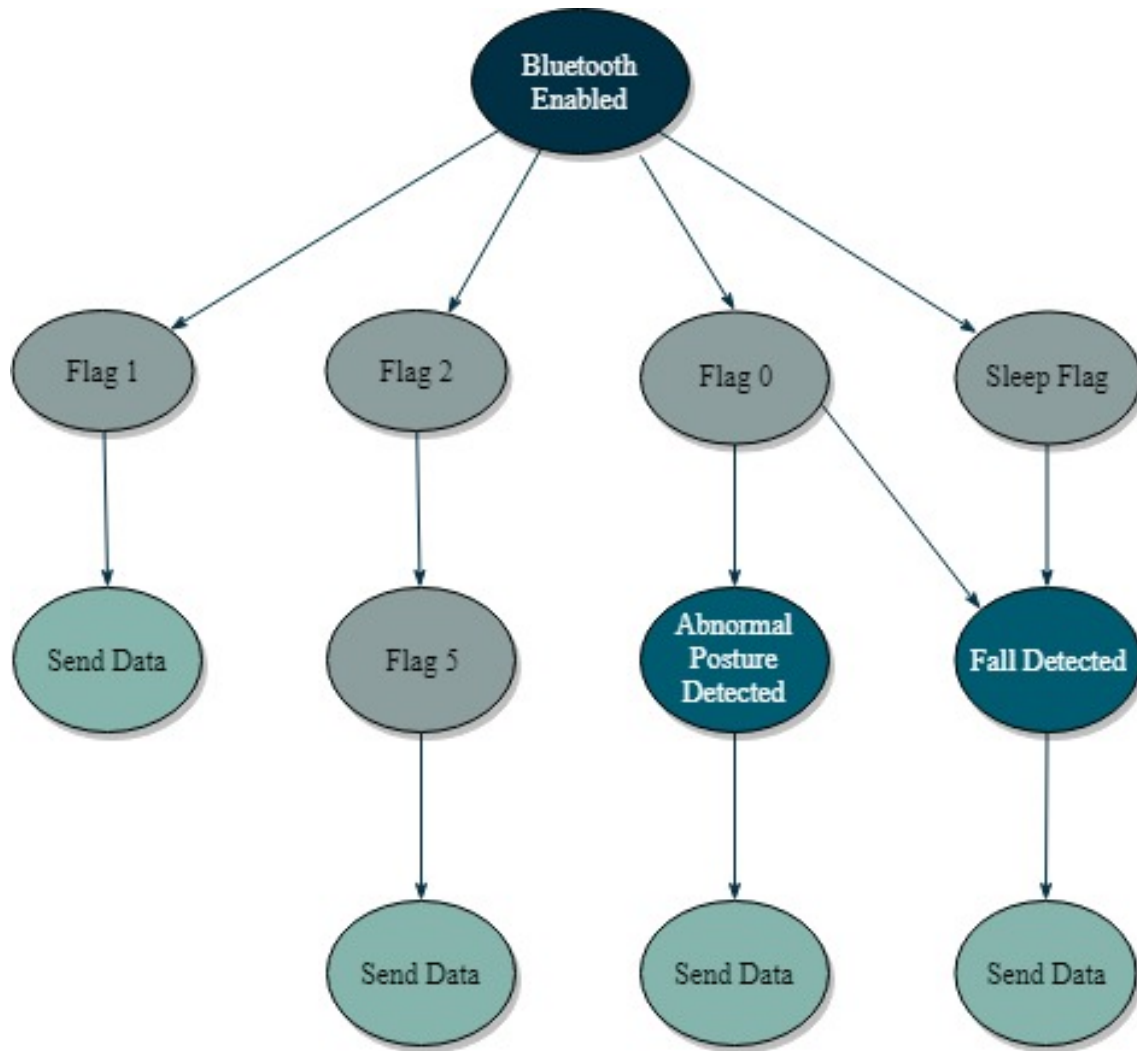
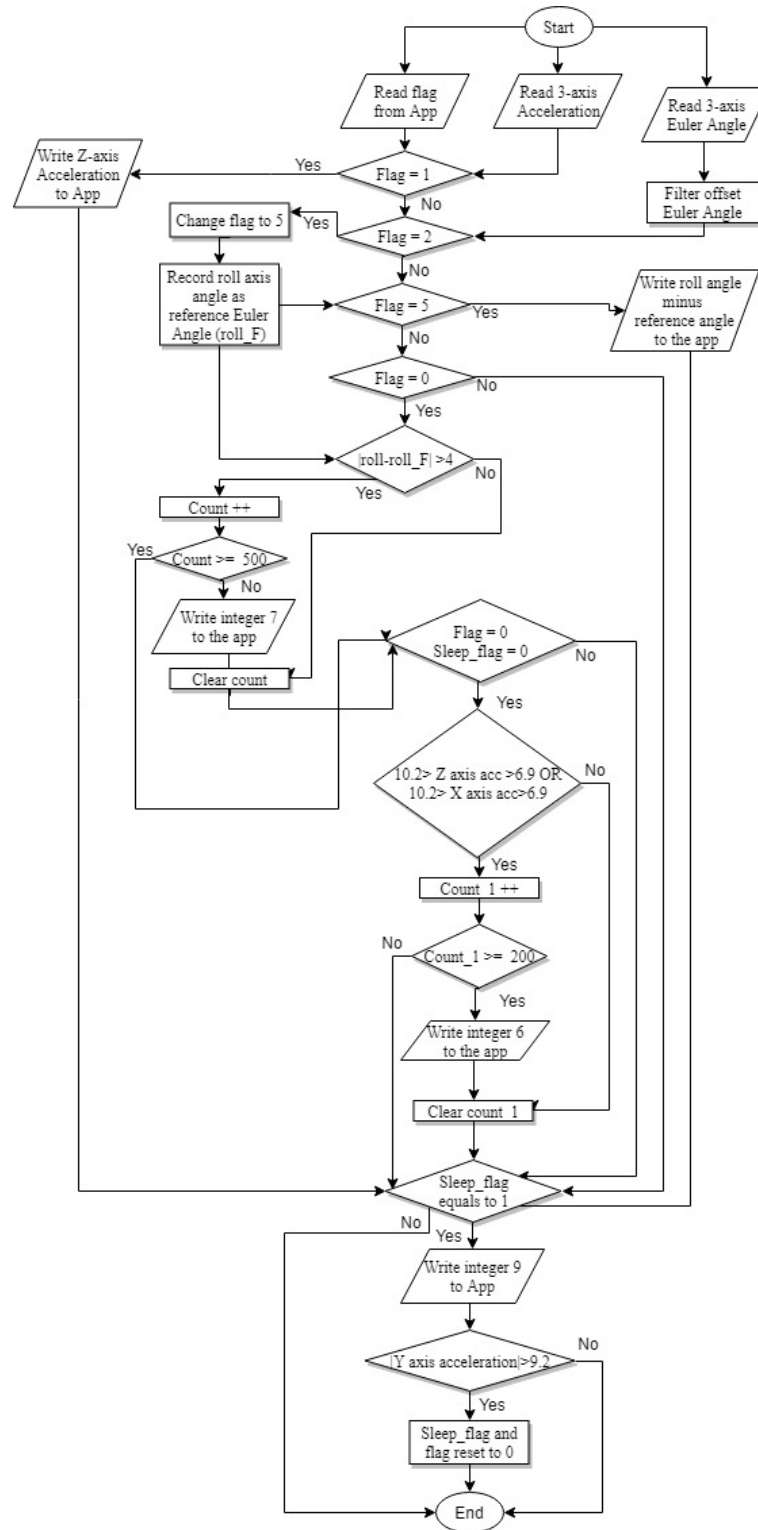


Figure 7.1: Bluetooth flag functionality flowchart

Each flag is a single byte of data because the function used to write the data requires only a byte of data as a parameter. These flags are sent when an abnormality was detected in the user's walking or posture. In addition, the data from the accelerometer and gyroscope is sent to the App for graphing purposes. This functionality is included once at the end of the 'loop' function, after each flag has had a chance to update its value based on its corresponding sensor.

## 7.2 Algorithms for Detection

All of the detection algorithms are on the Arduino code side. The MCU will read a flag from the app, analyze the current condition, and then judge whether to send an alert and/or data to the app. The app will then use the alert and data to send a posture message, a fall alert message, or graph the accelerometer and Euler angle data. The full software flowchart can be seen in Figure 7.2.



### 7.2.1 Fall Detection Algorithm

When we first developed our algorithm for fall detection, we only looked at the Y-axis data, which is the gravity axis when a person is standing. Whenever the magnitude of the Y-axis acceleration became zero, we assumed that the user had fallen down. However, this “naive algorithm” did not work when a person fell down on stairs and their body had an angle to the ground. In this situation, the magnitude of the Y-axis acceleration would not be zero.

The next version of the algorithm focused on the X-axis and Z-axis. We did several tests to see the X-axis and Z-axis acceleration changes under different motion conditions, including jumping, running, walking, falling when the Y-axis is equal to zero, and falling when the Y-axis is not equal to zero. (Detailed test results are included in Chapter 9.) When the magnitude of the X-axis or Z-axis is greater than  $6.9 \text{ m/s}^2$  and lower than  $10.2 \text{ m/s}^2$ , the user falls down. The reason of the lower boundary is that we assume that 45 degrees to the ground is the largest angle a person can fall. Under that falling condition, the Y-axis as well as the X- or Z-axes will split the gravity acceleration. Since the gravity on the earth is around  $9.8 \text{ m/s}^2$ , we then calculated that the Y-axis and X or Z-axes which are equal to the gravity/sqrt(2)  $\approx 6.9 \text{ m/s}^2$  (Figure 7.3).

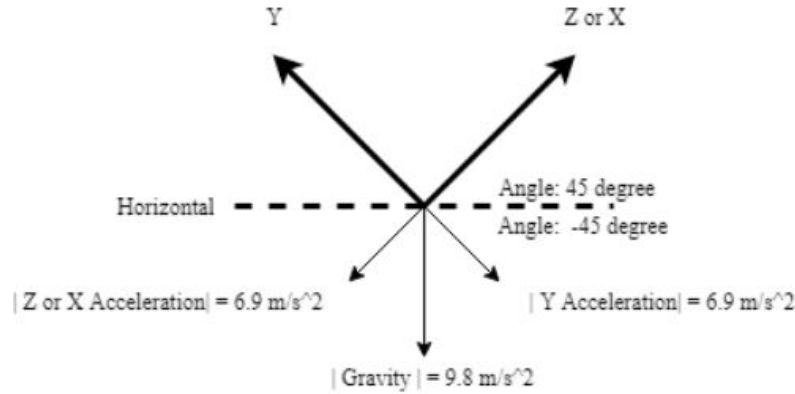


Figure 7.3: 45 degree fall acceleration analysis

Another feature of any falling condition is that all of the acceleration axes will stay the same until the user stands up. To accommodate this functionality, we added a debounce algorithm for our the fall detection algorithm. When the X-axis or Z-axis acceleration remains in the threshold range for more than two second, the MCU will send a fall detection flag to the app.

Since we only wanted to send one fall down flag for every fall detection, we added a sleep mode in the algorithm. Once the person falls down, the MCU will send the fall detection flag one time and then the system will go into a sleep mode. It will remain in this mode until the Y-axis acceleration is higher than  $9.2\text{m/s}^2$ , indicating a person has stood up. This will trigger the system to wake up and continue monitoring the X-axis and the Z-axis.

### 7.2.2 Posture Monitoring Algorithm

The system uses the LSM9DS0 driver which is a open source provided by Adafruit [25]. The driver reads in raw data of the accelerometer, gyroscope, and magnetometer from the sensor. After we received the raw data, we used several formulas to calculate a three-axis Euler angle in real-time. The Euler angle is three angles used to describe the orientation of a rigid body with respect to a fixed coordinate system (Figure 7.4) [41]. We decided to use the Euler angle as the sensor will never operate near pitch angles of  $\pm 90$  degrees for our applications.

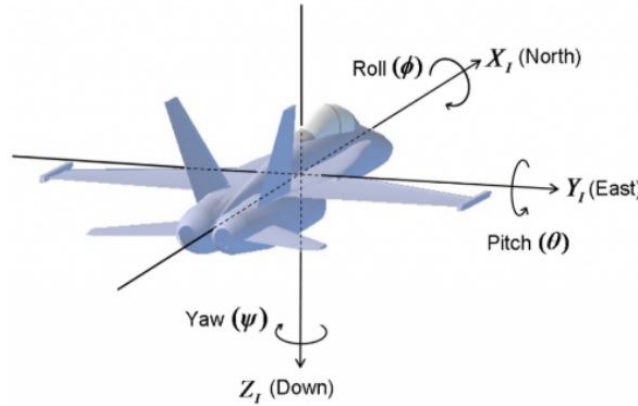


Figure 7.4: Euler angle inertial frame

Equation 7.1 shows the roll axis which is the rotation around the X-axis. A positive roll angle is defined to be a clockwise rotation about the positive X-axis.

$$roll = atan2 \frac{y}{x} \quad (7.1)$$

where  $y$  = acceleration in y-direction,  
 $z$  = acceleration in z-direction



Equation 7.2 shows the pitch axis which is the rotation around the Y-axis. A positive roll angle is defined to be a clockwise rotation about the Y-axis.

$$pitch = atan \frac{-x}{y \times \sin(roll) + z \times \cos(roll)} \quad (7.2)$$

where x = acceleration in x-direction,  
y = acceleration in y-direction,  
z = acceleration in z-direction

Finally, Equation 7.3 shows the heading axis which is the rotation around the Z-axis. A positive heading angle is defined to be a clockwise rotation about the positive Z-axis.

$$heading = atan2 \frac{(z \times \sin(roll)) - (y \times \cos(roll))}{(x \times \cos(pitch)) + (y \times \sin(pitch) \times \sin(roll)) + (z \times \sin(pitch) \times \cos(roll))} \quad (7.3)$$

where x = magnitude of acceleration in x-direction,  
y = magnitude of acceleration in y-direction,  
z = magnitude of acceleration in z-direction

Our device will focus on the major axis (roll axis) Euler angle which is suitable for posture monitoring, because the motions of human lumbar are forward and backward.

After we calculated the three-axes Euler angles, we used Python3.6 to monitor the user's changes in posture in real time. The roll axis is the majority motion axis of span. We implemented the functionality that users can press a button on the app to set their reference posture. When the roll axis angle is greater or lower than a certain degree, the MCU will send a posture alert flag to the phone application. Based on our different posture tests, we assume +/-4 degrees to be the threshold (See Section 9.2). The +/-4 degree threshold is based off the reference posture identified by the user. Ideally, the user sets the posture in the ideal position as seen in Figure 7.5.



Figure 7.5: Ideal posture while sitting

We then needed to consider the test case for when the user may pick up something on the ground. As a result, we decided to add a debounce algorithm and assumed a five seconds debounce. If the roll angle stays within the threshold range for more than five second, the MCU will send a bad posture flag to the app.

### 7.3 App Design

In order to build the app, we utilized MIT App Inventor 2 to design the Android application for our project. The purpose of the app is to alert a caregiver that the user has fallen or is having difficulties walking. In addition, the app will also monitor a person's posture and alert the user if their posture is abnormal. All of the code blocks used to design the app can be viewed in full in Appendix C.

The functionality of the app is simple to provide users with an easy interface. When the app is first turned on, it brings you to the main screen which has several functionalities (Figure 7.6).

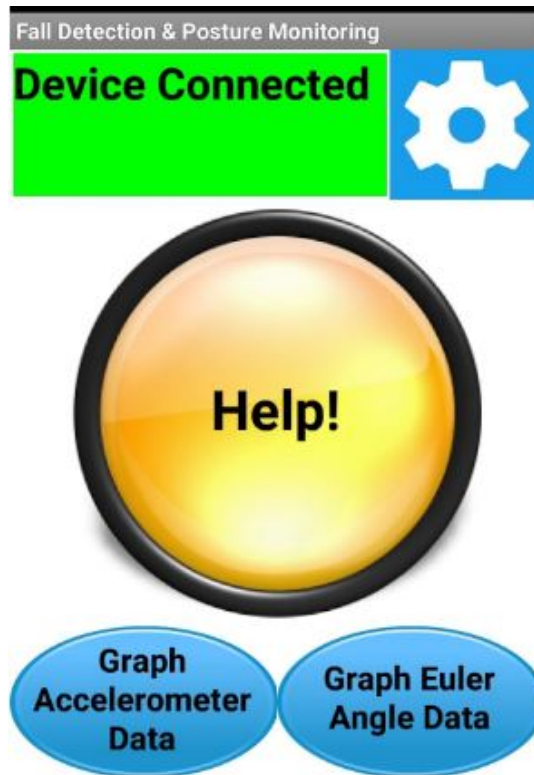


Figure 7.6: Main screen for application

The settings button allows the user to scan for nearby Bluetooth devices and connect to them as well as set the phone number the user wants as their primary contact (Figure 7.7). If the app connects to the Bluetooth device, the user is then able to graph sensor data as well as be notified whether or not a fall has occurred or abnormal posture is detected.

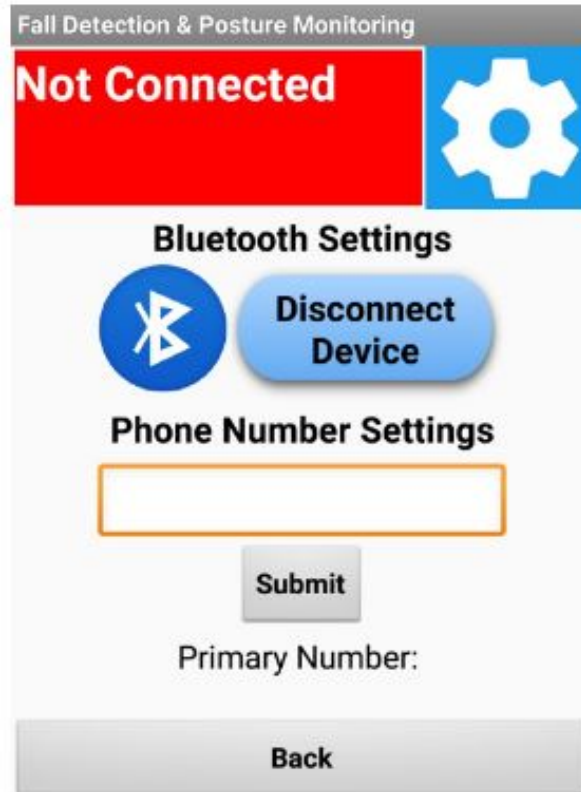


Figure 7.7: Settings screen for application

The help button allows the user to send a text message to their primary contact to indicate that they have either fallen or may be about to fall. The button sends an automatic text message that can be customized to alert the contact that help may be needed. This text message also will indicate the location of the person in case it is necessary. When a fall or poor posture has occurred, an alert will appear on the phone screen and either vibrate for two seconds to indicate that a fall has been detected or say "Stand Up Straight" if the user needs to fix their posture.

Finally, the last three buttons graph the various sensor data received by the MCU communicated through the Bluetooth module. When the Graph Euler Angle Data button is pressed, it sends a flag to the microcontroller to indicate that the user is in its ideal posture. For the graphs of the accelerometer and Euler angle data, we added a threshold value to indicate what the normal values are. For the Euler angle graph, we determined this threshold to be  $\pm 4$  degrees. We calculated this value by running a test on a team members posture (See Section 9.2). When the posture worsens, the data went outside the threshold of  $\pm 4$  degrees.

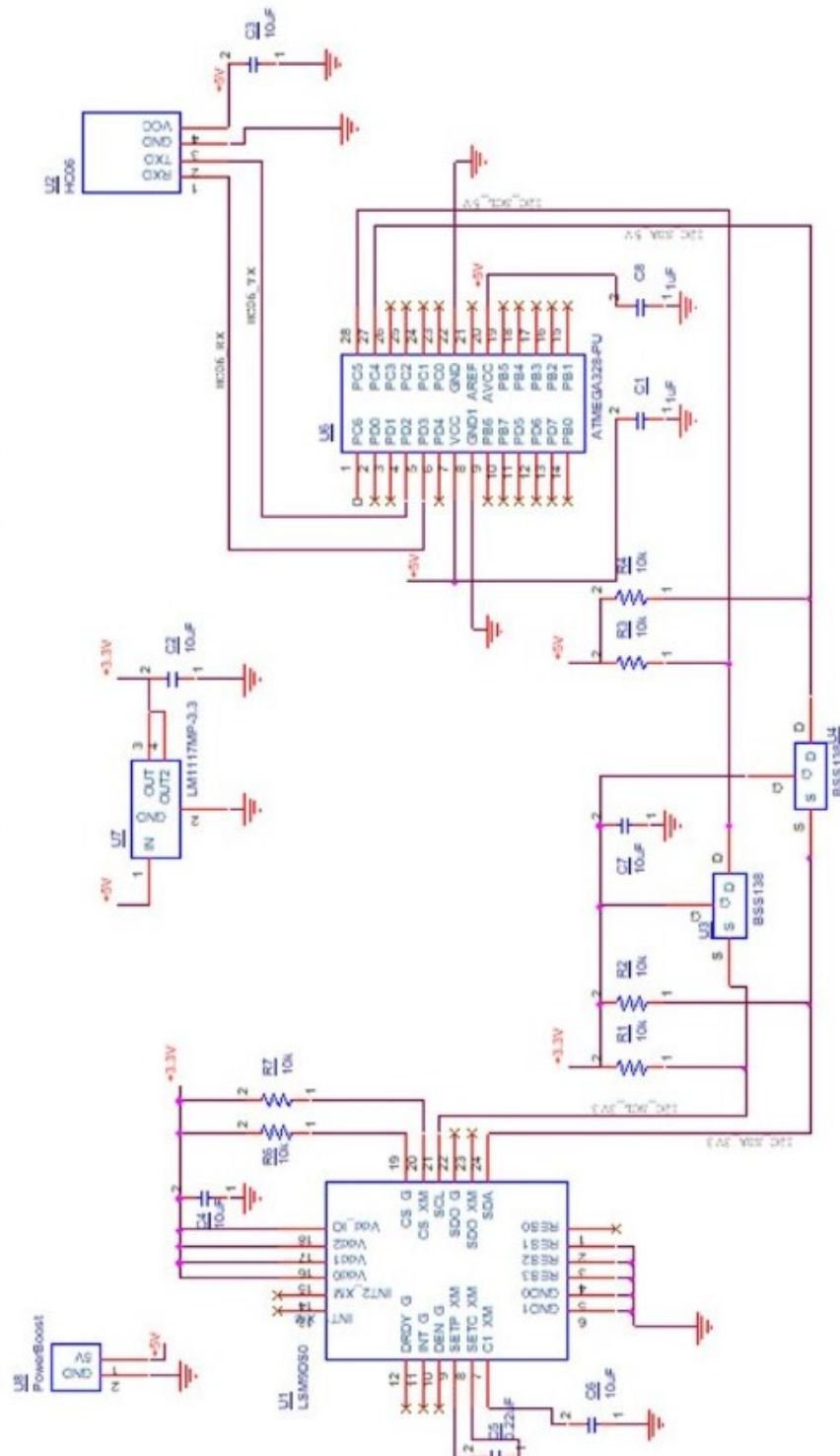
# Chapter 8

## PCB Construction

### 8.1 Schematic Design

Once the hardware design and functional block diagram were finalized, the next step was to generate an accurate schematic in Cadence Design Entry HDL. This was done by creating each custom part and putting them together in a custom library. It was important to have an accurate representation of pins on each part to correspond to the footprint of the part to ensure correct placement on the PCB. Our schematic design can be seen in Figure 8.1.

Fall Detection and Posture Monitoring System



All bypass cap rated 10uF

1156	ANALOG APPLICATION MGP	
File	Document	Rev
A	SCH	A
Ver	1	01
Tuesday, January 15, 2019 15:52		

Figure 8.1: Schematic design

## 8.2 Footprint Creation

While the schematic was being finalized, we worked on creating the footprints for the parts to go on the PCB. The footprints had to be carefully created with respect to the dimensions, drill holes, and pad stacks from the parts' datasheets. A few examples are as seen in Figure 8.2 and Figure 8.3:

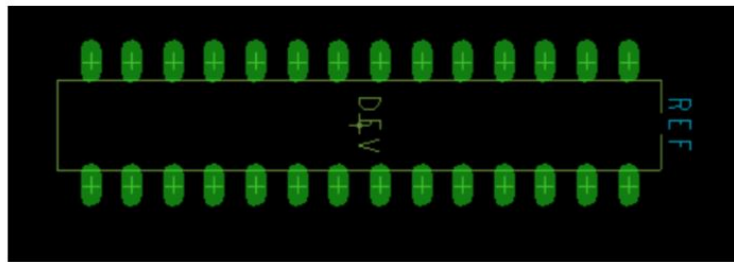


Figure 8.2: ATMEGA328-PU footprint

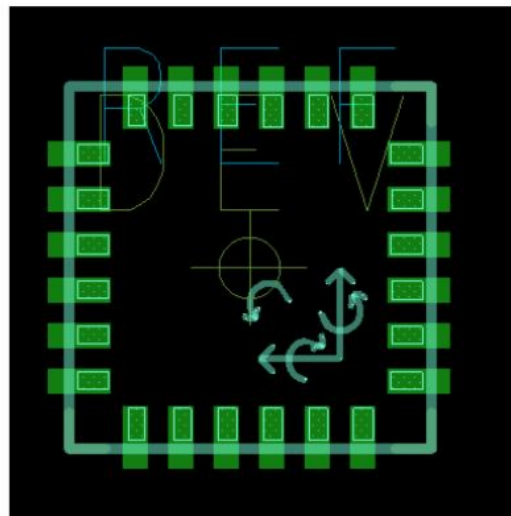


Figure 8.3: LSM9DS0 footprint

## 8.3 Layout Design

After the schematic and footprints had been created, we were successfully able to generate a netlist in Design Entry CIS which translated into a rats nest layout on

our PCB. Our PCB dimensions are 35 mm by 70 mm, optimized for component placement and routing. The board has 4 layers. Two of them are conductor layers: top and bottom while the other two layers are the power plane and ground plane. The board also includes four mount holes on each corner to accommodate the mounting applications. The power plane is divided into a 3.3V power plane and 5V power plane. The ‘Assembly’ view of the PCB can be seen in Figure 8.4. The +5V plane is marked in blue and the +3.3V plane is marked in purple.

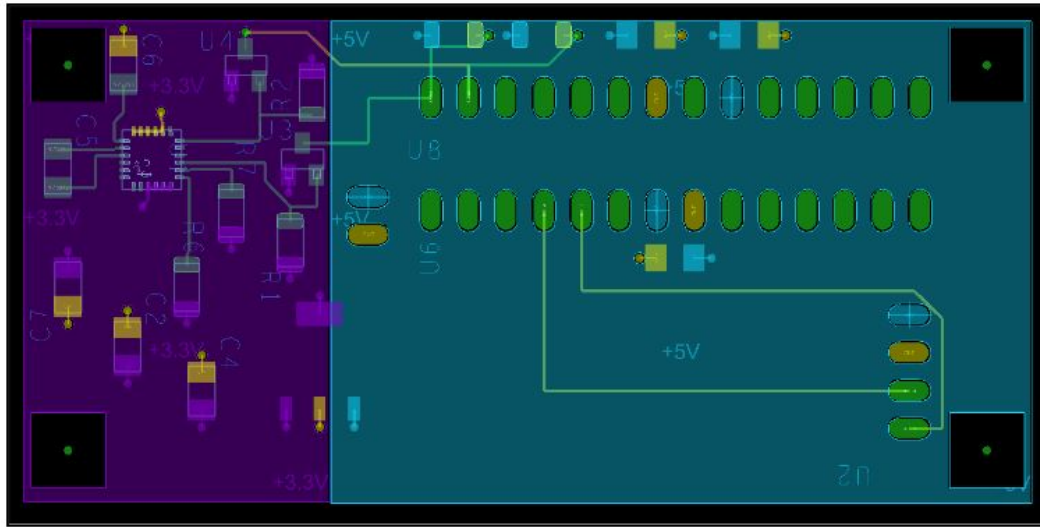


Figure 8.4: “ASSEMBLY” view of the PCB

## 8.4 Optimization: Routing and Component Placement

Components have been placed on either side of the board depending on their operating voltage which is either 3.3V or 5V. Bypass capacitors were placed as physically close to the respective component to make sure the plane conductance did not hinder the performance. We tried to keep the trace length as small as possible to reduce transmission line effects. We also tried to route traces at 45° to reduce eradicate inductive spikes at 90° traces.



# Chapter 9

## System Testing & Results

### 9.1 PCB Testing & Results

Before soldering all of the components on the PCB, we used a multimeter to test each solder pastes' connections and determine whether there were any short or open circuit problems on the PCB. After we soldered a component, we also checked whether the solderability of the component under a microscope. After we finished soldering each component on the PCB, we also used a microscope to recheck every connection. Finally, we charged the PCB with a battery and uploaded the code onto the MCU. Once all of the tests have been verified, we knew that the PCB was working as what we had expected from the breadboard results. As a result, we concluded that all of the functionalities worked for our PCB (Figure 9.1).



Figure 9.1: Final PCB after soldering and testing

## 9.2 Angle Monitoring Functionality & Accuracy Testing & Results

To test the angle measures, we first needed to check that operation worked correctly in the roll, pitch, and yaw directions. We needed to ensure that we could read data for each different axis. To do this, we created a clock from the MCU and set up a timer circuit to control the sensor on the PCB. We tried different motions in different directions to ensure that we could see a change in the angle in the microcontroller environment. Another component we needed to test was the communication between the microcontroller via I2C. We ensured that this was working properly by checking that the data read by the sensor could be seen in the microcontroller environment. To test this, we connected the sensor to the microcontroller evaluation board and set up the UART communication between the microcontroller evaluation board and Matlab. We then printed the sensor angle results in Matlab and verified that they were correct. For continuous testing, we used python3.6 to read the sensor angle output through UART and plot them on screen in real time. The Python scripts used to generate the test plots can be seen in Appendix D.

To test the accuracy of angle, We used a soldering frame to stable the prototype and tested the pitch axis angle from 0 to 45 degrees incrementing by 5 degrees. We then collected 10,000 data points through UART for each angle by python3.6, and analyzed the data and plotted box plots in Excel to analyze the accuracy of angle (Figure 9.2).

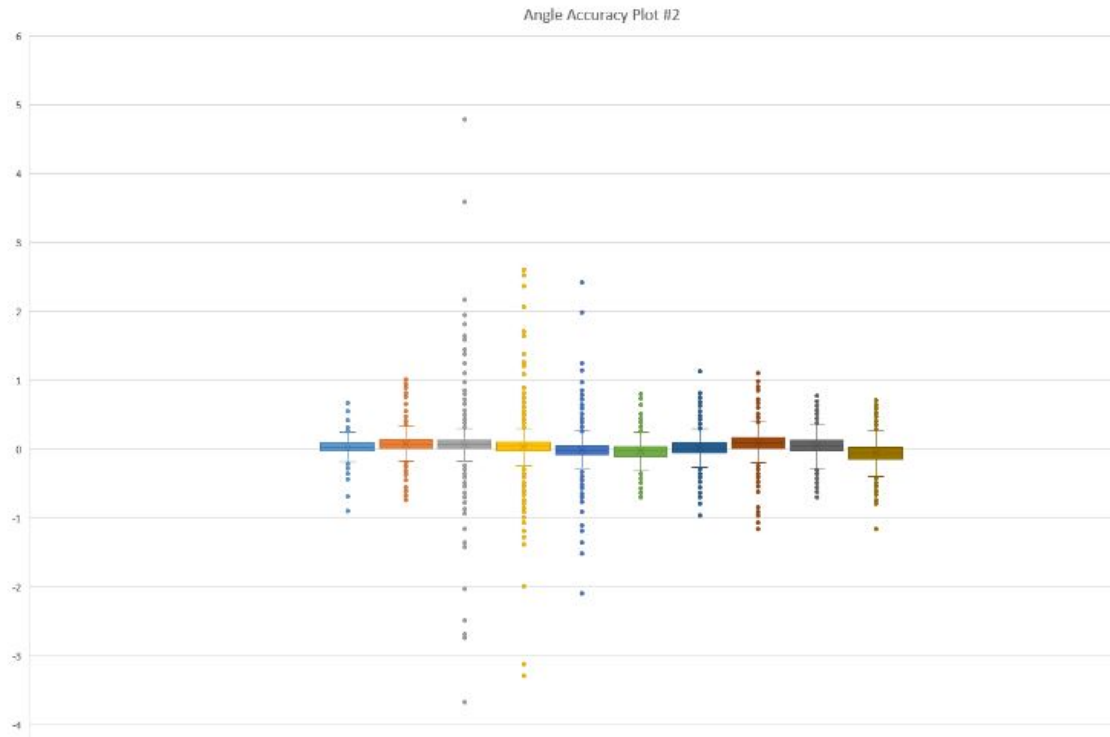


Figure 9.2: Posture angle accuracy results

For the spine angle monitoring functionality tests, we tested the prototype under several different test cases. The team collected 10 seconds of data and plotted the major axis (roll axis) on python3.6. The first test case was to monitor the user from their reference posture to leaning over on a desk while sitting in a chair (Figure 9.3).

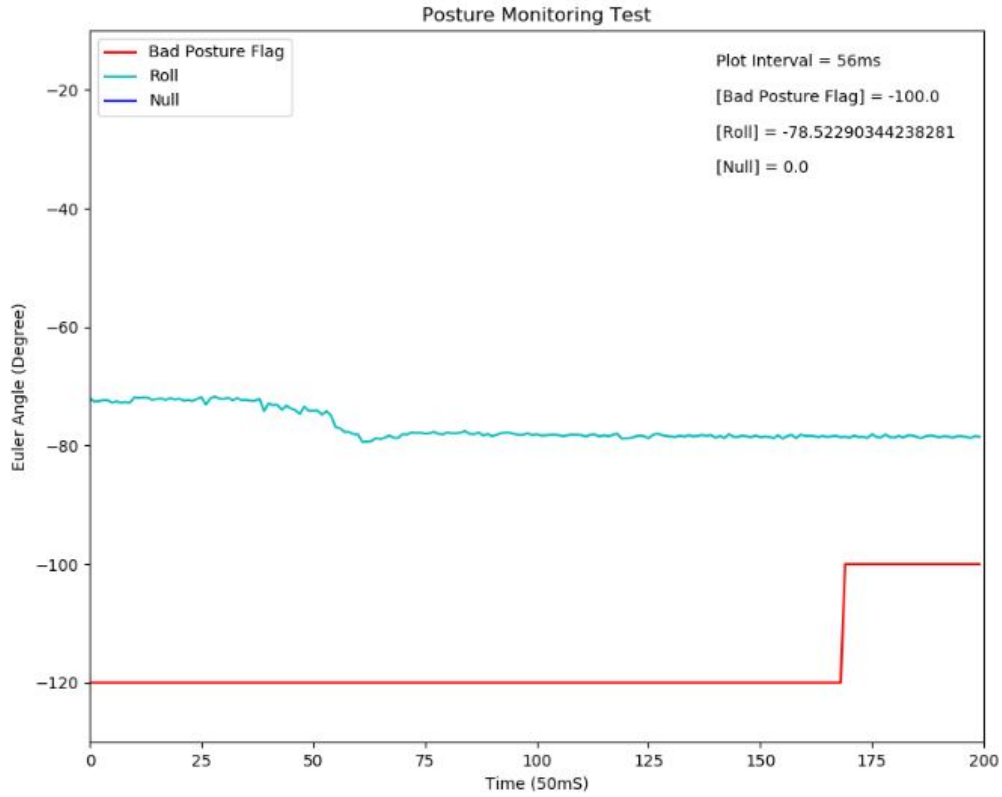


Figure 9.3: Reference posture to leaning over on desk

After five seconds of the user leaning over on their desk, the bad posture flag (shown in red) changes to one indicating that bad posture has been detected. The second test case that we monitored was the user changing their posture from the reference point to slouching on a desk with their elbow (Figure 9.4).

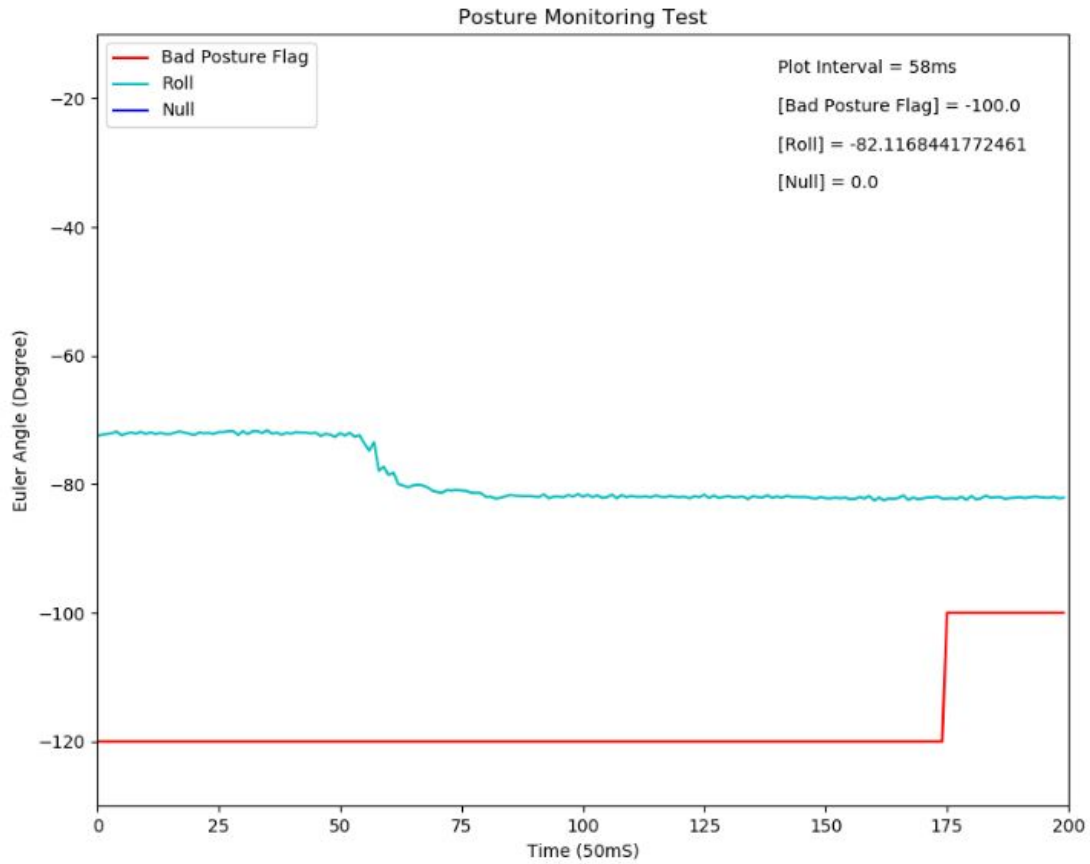


Figure 9.4: Reference posture to slouching on desk with elbow

After five seconds of the user slouching on the desk with their elbow, the bad posture flag (shown in red) changes to one indicating that bad posture has been detected. The third test case we conducted was to have the user sit in a chair and monitor their position change from their reference posture to a slight slouch (Figure 9.5).

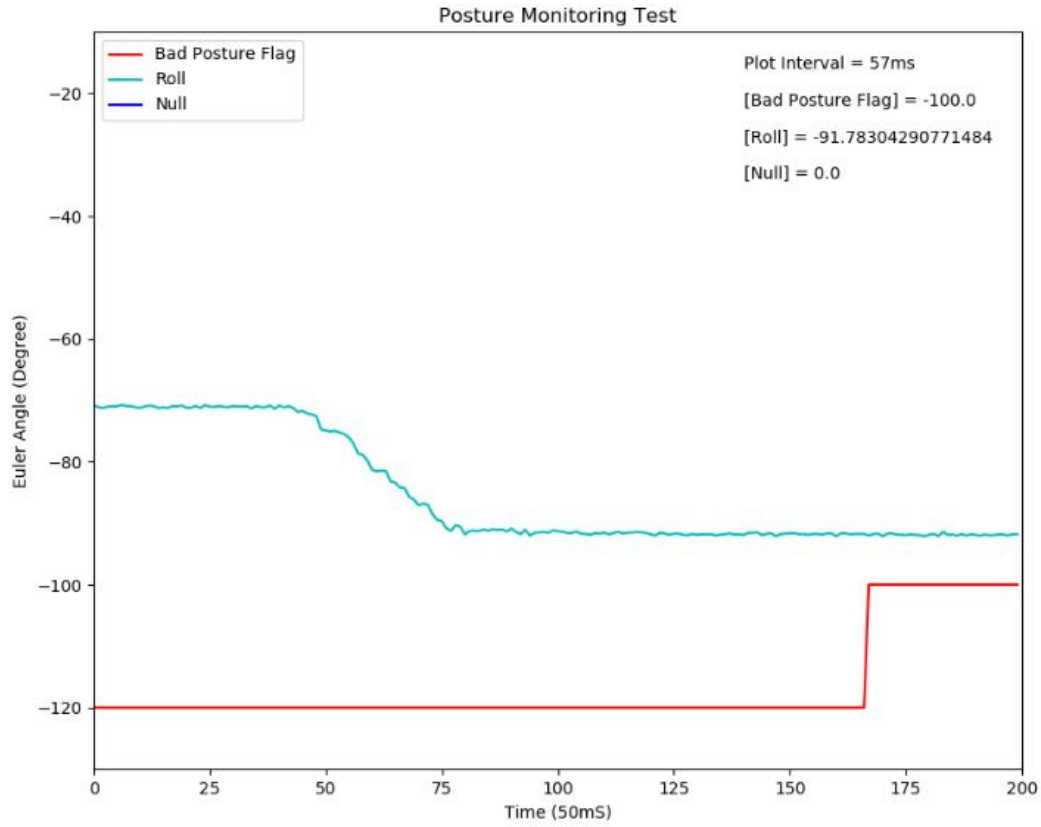


Figure 9.5: Reference posture to slight slouch

After five seconds of the user slightly slouching, the bad posture flag (shown in red) changes to one indicating that bad posture has been detected. The final test case for posture monitoring we looked at was when the user slouches when picking up an object (Figure 9.6).

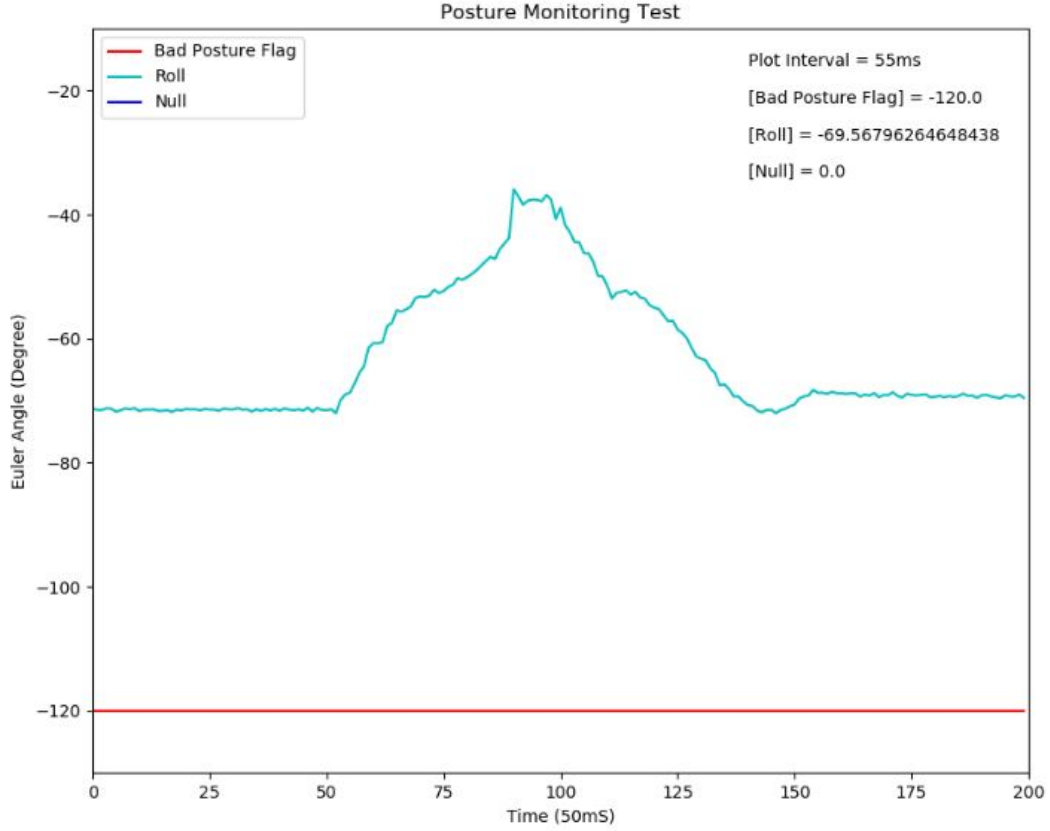


Figure 9.6: Reference posture to picking up object

Since the posture changes were not more than five seconds, our poor posture flag was never triggered. From these test plots, we determined that a  $\pm 4$  degrees angle threshold makes sense for our algorithm. With the alert system and Bluetooth, the app then sends the “Stand Up Straight” voice message on time when the user has poor posture for more than five seconds.

### 9.3 Fall detection Functionality Testing & Results

To test the fall detection functionality, we needed to check whether the system will send warning messages for different human motions. It was important to ensure that our system would only send an alert fall detection and not when the user is running, walking, or jumping. Based on our algorithm, when X-axis or Z-axis’s acceleration is greater than  $6.9 \text{ m/s}^2$  and lower than  $10.2 \text{ m/s}^2$  for more than two seconds, the

user has fallen down. The first test we conducted was to monitor the X- and Z-axes accelerations when the user is jumping (Figure 9.7).

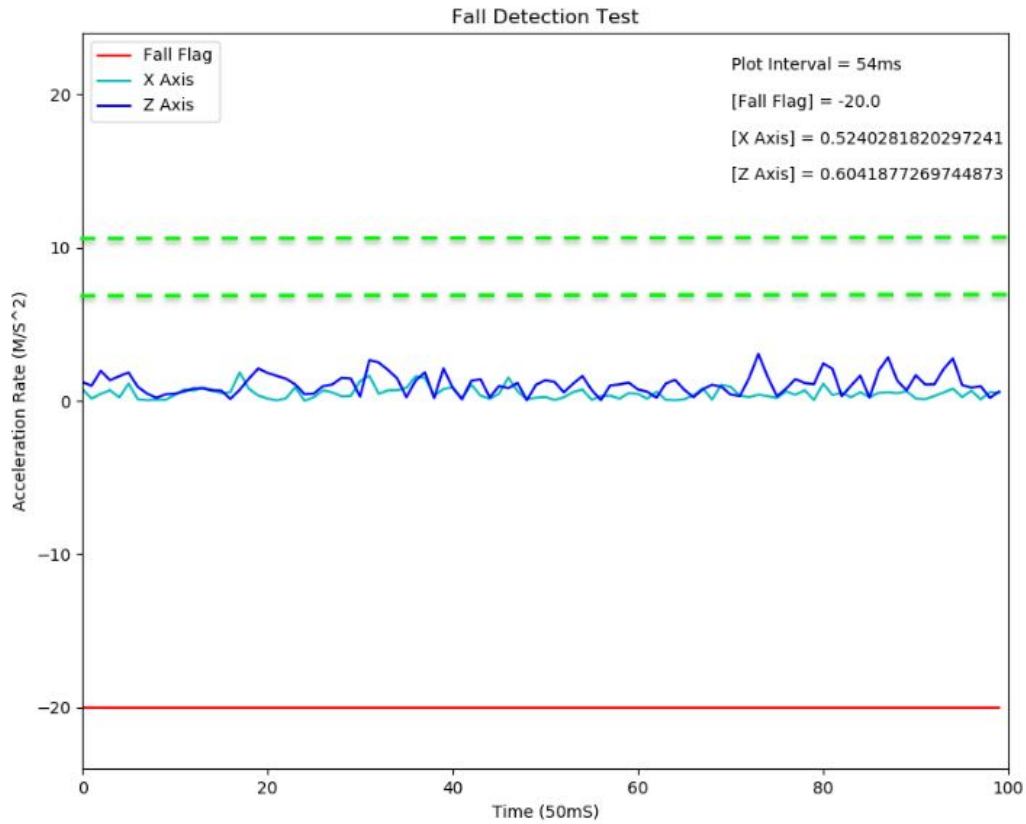


Figure 9.7: No fall jumping test case

As seen in the graph above, the X-axis and Y-axis accelerations move slightly, however, they are not within our threshold and therefore no fall flag is triggered. The next test case we monitored was the motion of the user walking (Figure 9.8).



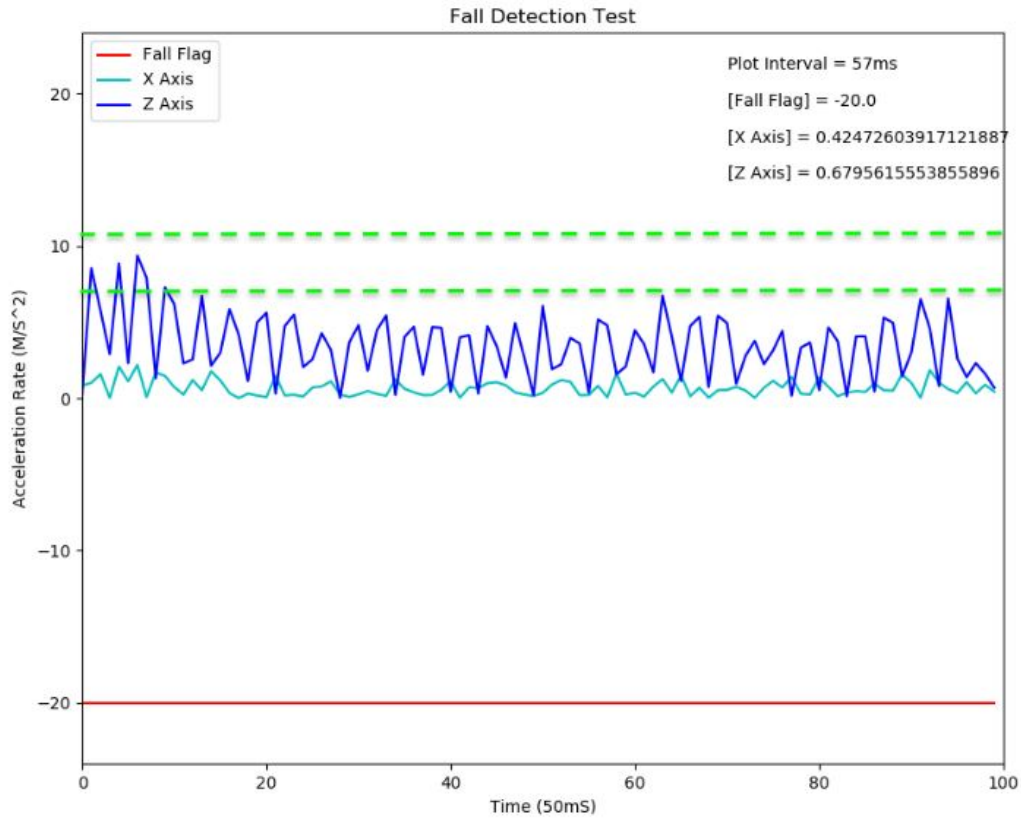


Figure 9.8: No fall walking test case

As seen in the graph above, the X-axis and Y-axis accelerations sometimes are within our threshold. However, they do not remain in our threshold for more than two seconds, therefore, no fall flag is triggered. The next test case we monitored was the motion of the user falling while running (Figure 9.9).

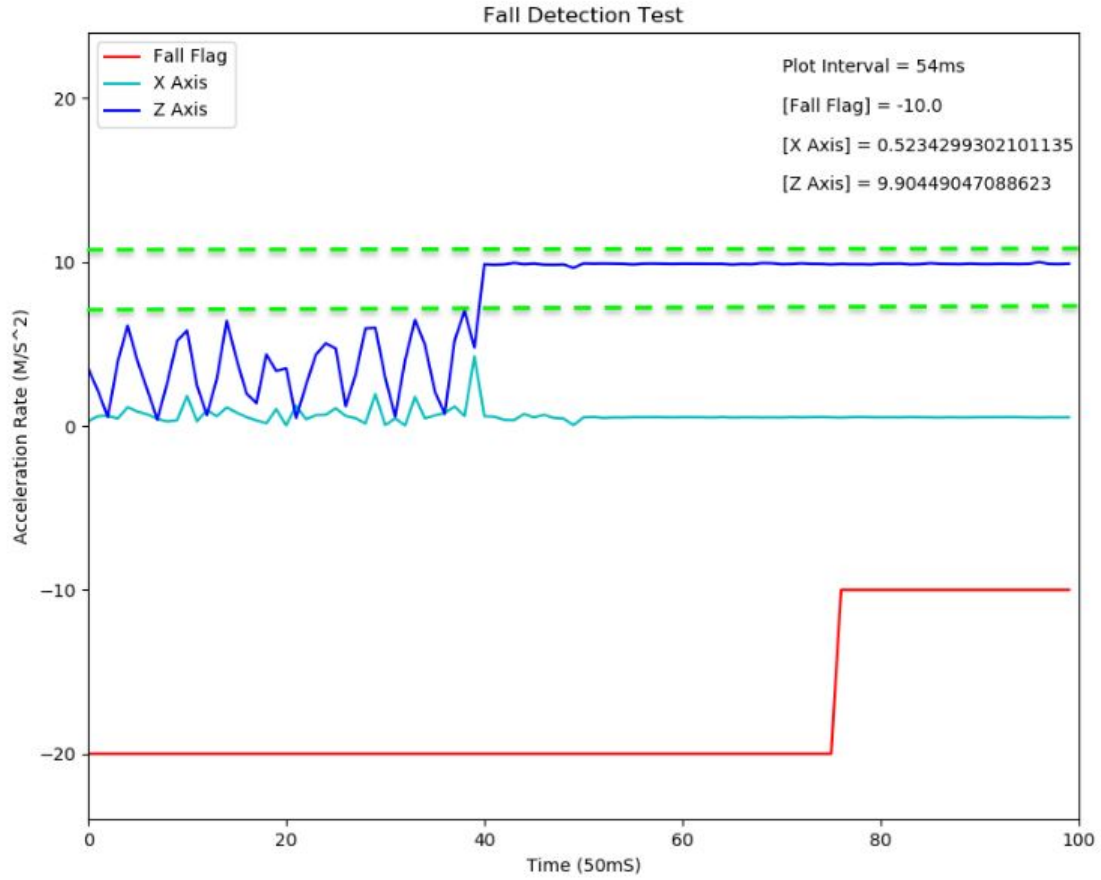


Figure 9.9: Fall while running test case

As seen in the graph above, the X-axis and Y-axis accelerations stay within our threshold for more than two seconds. This indicates that the user has fallen and therefore the fall flag is triggered and set to one. The next test case we monitored was the motion of the user falling while walking (Figure 9.10).

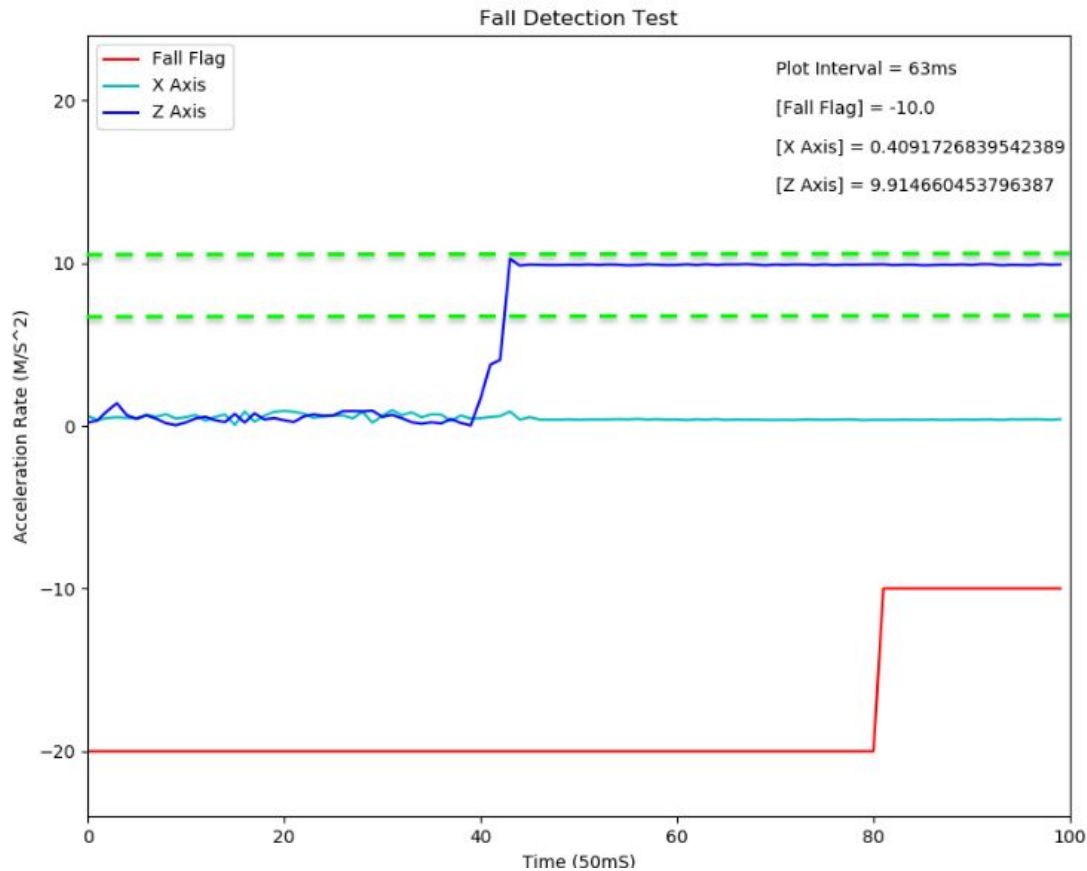


Figure 9.10: Fall while walking test case

As seen in the graph above, the X-axis and Y-axis accelerations stay within our threshold for more than two seconds. This indicates that the user has fallen and therefore the fall flag is triggered and set to one. The final test case we monitored was the motion of the user standing up after they have fallen (Figure 9.11).

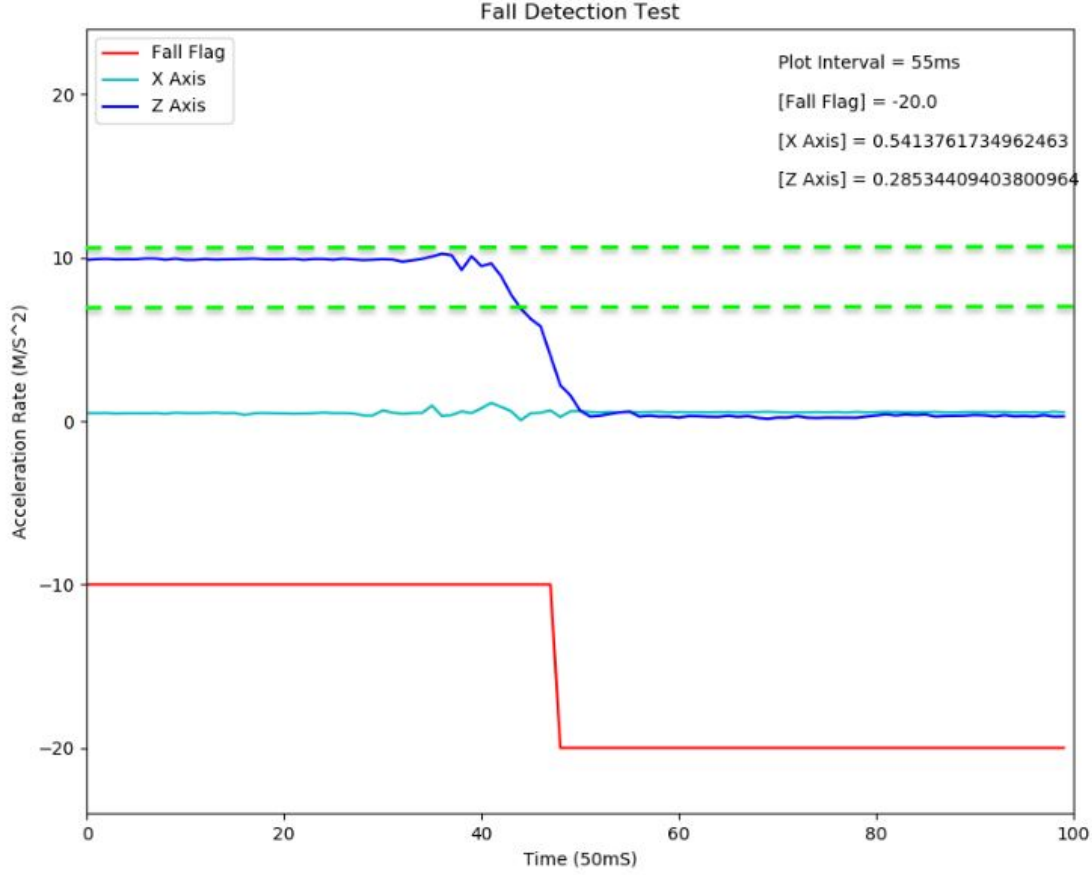


Figure 9.11: User stands up after a fall

As seen in the graph above, the X-axis and Y-axis accelerations leave the threshold for more than two seconds. This indicates that the user has stood up after falling, therefore, the fall flag is reset to zero.

## 9.4 BLE Testing & Results

To demonstrate wireless capabilities, we added the HC-06 to our design. This Bluetooth module adopts the Bluetooth 2.0+EDR standard and operates at 2.4GHz. This standard allows for a 0.5 seconds transmit time which significantly reduces the workload of the Bluetooth chip. It also allows the Bluetooth module to be in sleep mode longer. In addition, this module has an easy interface to the ATMEGA328-PU microcontroller.

To test this module, we first connected it to a third party application called nRF Connect to ensure that the Bluetooth module could properly transmit data from the app to the Arduino. We tested this by sending a certain integer from the application and then tested to make sure that we could read that number in the Arduino terminal. We also ensured that we could send a number from the Arduino through the Bluetooth module and that the app would be able to receive it (Figure 9.12).

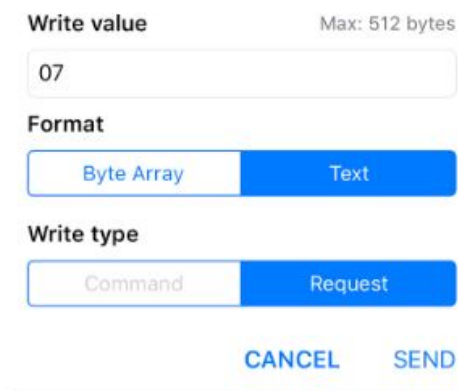


Figure 9.12: Bluetooth testing with nRF Connect application

Next, we used the application that we designed using MIT App Inventor 2 to test the Bluetooth integration. First, we ensured that the app indicator light turned green once it paired with the HC-06 module. Otherwise, the indicator light would stay red and an error message would display (See Figure 9.13).

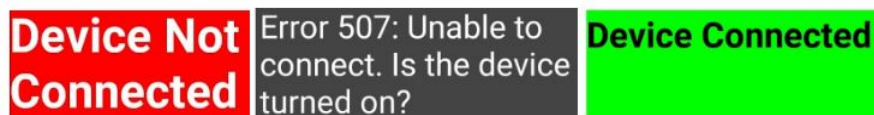


Figure 9.13: Indicator light that turns green or red based off Bluetooth connectivity

In order to ensure that the BLE module can properly transmit our sensor data, we generated simple sine wave data in the Arduino code and sent it via Bluetooth to the app (See Figure 9.14). We confirmed that no data was being missed since the graph generated was a perfect sine wave.

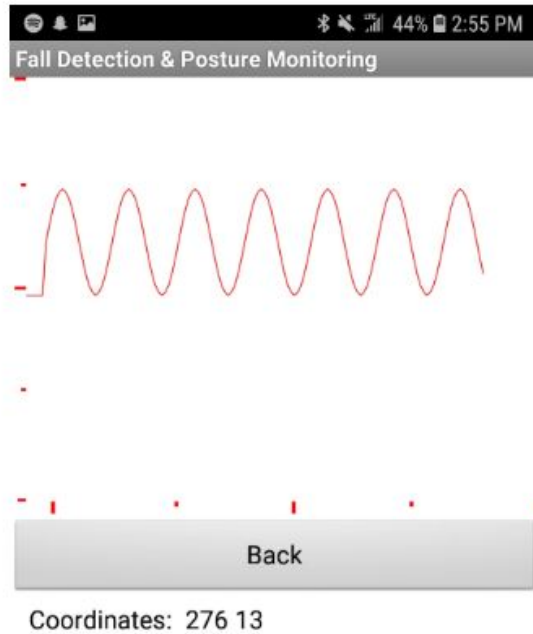


Figure 9.14: Sine wave data graph

We also tested that the MCU could receive the data sent by the Bluetooth module from the application. We did this by sending a flag to the MCU each time a button was pressed to tell the MCU which sensor data to send back to the application (See Figure 9.15).

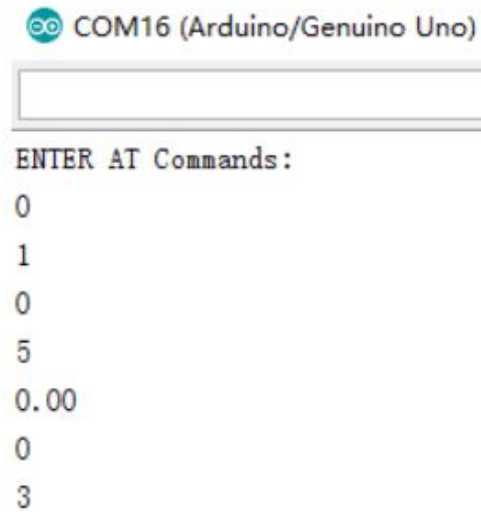


Figure 9.15: Flag functionality for Bluetooth module

Finally, we ensured that the sensor data for both the accelerometer and gyroscope were being sent correctly to the application. We added functionality to the app so that it would be able to graph both data sets in real time by the press of a button. Figure 9.16 and Figure 9.17 show the accelerometer data and gyroscope data graphs respectively.



Figure 9.16: Accelerometer data graph



Figure 9.17: Gyroscope data graph



## 9.5 Battery and PowerBoost Testing & Results

Since we are using a PowerBoost that internally boosts the 3.7 battery to 5V, we needed to ensure that the output was in fact around 5V. We did this by using an oscilloscope to test the output DC voltage level and confirmed that it was around 5.4V (Figure 9.18). From the powerboost specifications, the output range is from 5V to 5.2V [42]. The output is a bit higher than the standard output, however, it is still sufficient as the power supply for the MCU and voltage regulator.

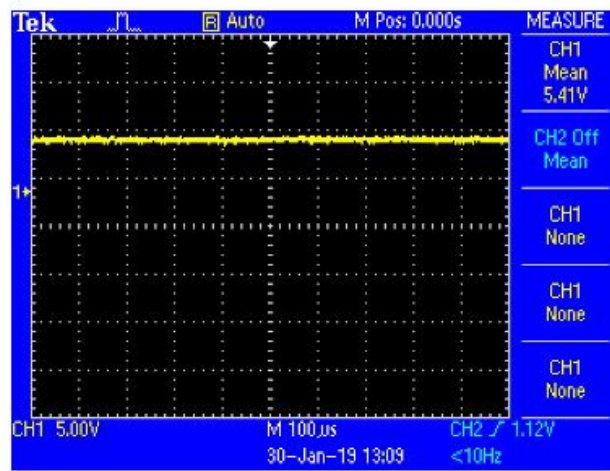


Figure 9.18: Oscilloscope output for the 5V PowerBoost

Next, we needed to test the battery in order to determine the actual runtime time of our system. We tested this by running the product continuously for 24 hours to see the duration of battery. After conducting this test, we determined that the runtime of the battery is approximately 16 hours. Finally, we needed to test how long it took to charge the battery. To do this, we needed to test several cases for when the battery is at 0%, 25%, 50%, and 75%. We will observe the LED of the battery until it lights up and record the time using a stopwatch. The results of this test can be observed in Table 9.1.

Table 9.1: Observed charging times based on battery charge

Battery Charge	Time Taken to Charge (Minutes)
0%	110
25%	84
50%	60
75%	31

# Chapter 10

## System Improvements

Since the Fall Detection and Posture Monitoring system is in the prototype stage, there are several improvements that could be made. These improvements include updating the microprocessor and BLE module, using an alternative power supply, using a different interface to build the app, and finally improving the PCB design. These improvements are elaborated upon in the following sections.

### 10.1 Microprocessor & BLE

To improve the microprocessor in our design, we would replace the Arduino AT-MEGA328 processor. The TI CC2650MODA processor described in Section 4.2.2 is a favorable replacement, as it has integrated BLE functionality, as well as a low power mode (Figure 10.1). This microprocessor would replace the HC-06 as the Bluetooth module. Using the CC2650MODA would minimize the size of our product as the ATMEGA328-PU is 35.544mm x 7.62mm in size while the CC2650MODA is half the size. Although the CC2650MODA is a higher cost than the ATMEGA328, we would not need the extra expense of a Bluetooth module.



Figure 10.1: CC2650MODA launchpad

Initially, our team chose to use the CC2650MODA as the microprocessor for our system. Unfortunately, the CC2650 is one of the newer processors in Texas Instruments' RF communication line, therefore, there was limited useful and relevant guides, documentation, and resources for us to utilize. In addition, we found the MCU SPI communication difficult to program and found little support online. As a result, we needed to keep our three-term project deadline into consideration and discontinued our work with this chip for the remainder of our project. We decided as a team that it required a greater expertise in embedded software development and demanded much more time to implement than we had available.

### 10.1.1 Updated Power Analysis

After deciding the CC2650MODA would be a better fit for our system's MCU, it was important that we conducted another power analysis to determine if the runtime of the overall system had improved. To start, we calculated the total current draw from all of the components in our system (See Table 10.1)

Table 10.1: Total current usage of components in the system

Component	Current Usage (Worst Case)
Sensor	6.35mA
<b>Microcontroller</b>	<b>9.4mA</b>
Bluetooth Module	8mA
<b>Total</b>	<b>23.75mA</b>

The overall worst case current usage decreased by almost 10mA by using the CC2650MODA MCU rather than the ATMEGA328. Next we divided the current capacity of the battery by the total current usage of the system as seen in Equation 1. The results indicate that with the worst case current usage, the battery will last up to roughly 23 hours. This increases the battery lifetime by roughly 9 hours, worst case.

### 10.1.2 Updated Cost Analysis

It was also important that we conducted another cost analysis to determine if the cost of the overall system improved. To start, we removed the total cost of the AT-MEGA328 as well as the cost of the HC-06 and added the cost of the CC2650MODA (Table 10.2).

Table 10.2: Updated bill of materials

Component	Quantity	Total Price (\$) (4000pcs)
CC2650MODA MCU	1	3.23
BSS138 Logic Shifter	2	0.12
LSM9DS0 Sensor	1	6.50
PowerBoost	1	6.50
Rechargeable Battery	1	4.00
LM1117 Regulator	1	0.45
Misc - R,C,LED	-	1.50
PCB	1	0.40
Plastic Case	1	0.60
Total Cost		\$23.30

As a result, the overall cost of the system for 4000 pieces is roughly \$23.30. While this is not much cheaper than our \$23.91 design, it will save us significant space on our PCB to have the Bluetooth module integrated with the MCU.

## 10.2 Power Supply

For further improvement, a Coin Battery could replace our Lithium Ion Battery for its lower cost and its smaller size (Figure 10.2). Our current runtime is around a day and a new battery with a week runtime would be ideal.



Figure 10.2: Coin battery as potential system improvement

Another potential improvement would be to use the 2500mAh Lithium Ion Polymer Battery as it would increase our current system runtime to around five days (Figure 10.3). However, it is heavier and larger than our current battery. Its weight approximately 56 g while our current battery is 10.5g. Additionally the size is 51mm X 65mm X 8mm while the size of our current battery is 29mm X 36mm X 4.75mm.



Figure 10.3: 2500mAh Lithium Ion Polymer Battery as potential improvement

### 10.3 App Design

Since none of our team member's had any experience designing a phone application, we decided to use MIT App Inventor 2 as it is an easy to use software in which user's assemble program blocks that specify how the components should behave. However, after using the program we found that MIT App Inventor had limited functionality for the complexity of our design. It was important that our phone application graphed the real-time data so the user could see their results. MIT App Inventor had limited graphing abilities which made it difficult to graph the three axes needed for fall detection. As a result, we considered to improve the phone application by using Android Studio which is the official integrated development environment for Google's Android operating system (Figure 10.4) [43]. It is more professional on Android application design and we can code in Java to include better graph performance and user interface.



Figure 10.4: Android studio logo

## 10.4 PCB Design

For further PCB improvement, the 90 degree angle circuit issues should be solved, as it is not good for current to flow through. These components with this kind of issue on a PCB should be placed properly. In our current design, the PowerBoost and BLE components are individual modules that plug into the PCB. A improvement would be to move the circuit and components of the PowerBoost and BLE to the PCB design, which will lower the cost and decrease the size of our product.

# Chapter 11

## Video Production

A predefined objective of our MQP was to make a cool video through which we could showcase our project at open houses and at presentations. To produce this video, we completed the steps shown in Figure 11.1.



Figure 11.1: Steps for video production

Before we began designing the video, we completed brief research on the video production process. This helped inform us of the best practices for putting together the MQP video. Story boarding, a key process in video production, is the use of a visual ‘script’ to outline the different scenes of a video production. According to Paez and Jew, story boarding begins as an outline of the content and accompanying video shots that must be filmed to make the video [44]. This is usually done by creating a basic outline with sketches of each scene or shots with the information that will be presented at that point in the video. Once the storyboard has been finalized and actors, props, and the script have been approved, filming can finally take place. During this time, any audio soundtracks should also be recorded. Once all of the content is collected, editing and finalizing the footage is the final step before making the video public [44].

In order to generate content for our video, we collected the functional specifica-



tions of our device that included dimensions, run time, and charging method. We also drew up a few use cases of our device that we would want to showcase in the video. After outlining the content of the video, we began story boarding with the goal to integrate humor into the video to make it more appealing and expand its reach on social media. Before filming, we also identified the logistics of filming, such as finding actors and narrators, and determined filming locations and all necessary shots. Figure 11.2 shows a snippet of our storyboard.

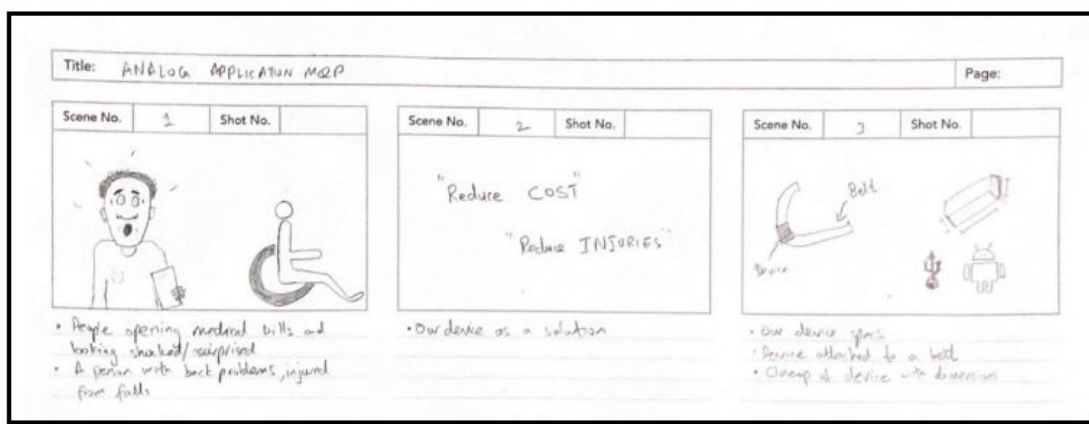


Figure 11.2: Storyboard of scenes one through three

Once our storyboard was finalized, we began filming. Filming took a couple hours, during which we worked to obtain all the shots we needed for the video. We used the voice of one of our MQP partners, Holly Shumway, to narrate the video. Figure 11.3 shows the script for our opening scene.



Figure 11.3: Script for scene one

Editing was the final step in the video production, and we utilized Adobe Premiere Pro, as shown in Figure 11.4, to put the video together. After reviewing the footage to ensure all scenes were accounted for, we utilized the storyboard and script to put the footage in the right order. The video was edited scene by scene, in accordance with the storyboard. This allowed us to work on scenes in any order, instead of working on the video chronologically. The body was worked on and completed before the introduction, as the introduction utilized snippets of the body footage. We also edited individual pieces of footage to ensure a smooth video and cut down length as necessary. Matching up video footage and the recorded voice narrations led to cutting down more footage, or re-recording in order to match the footage better. Once individual scenes were completed, we moved them into one main sequence where we added transitions to allow for a better flow when viewing the video.

## Fall Detection and Posture Monitoring System

---

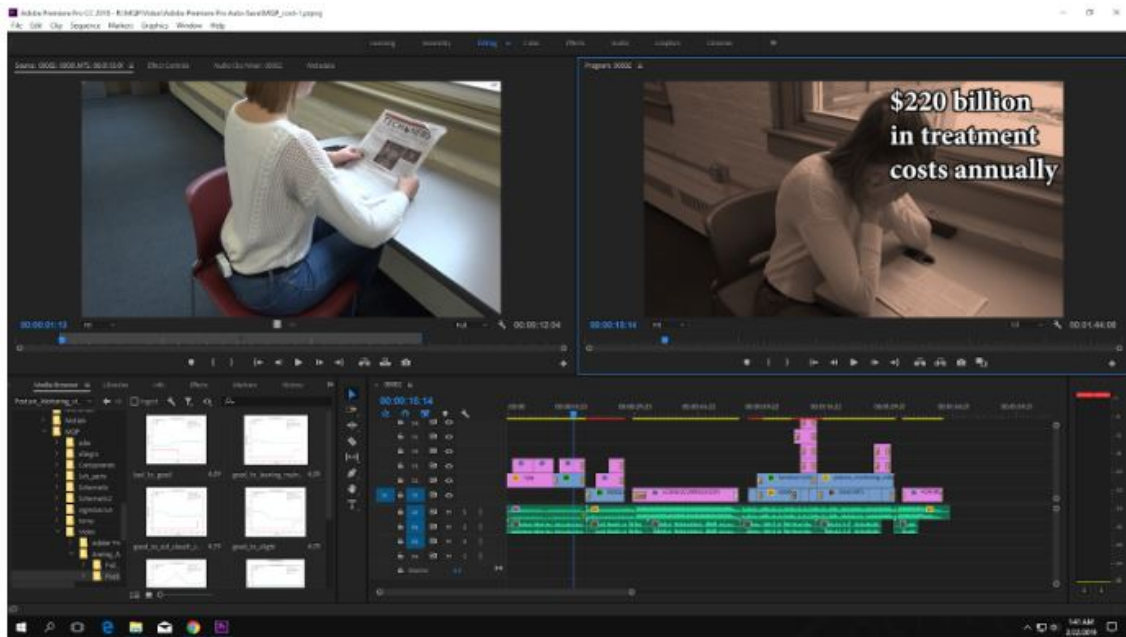


Figure 11.4: Editing & compiling video footage in Adobe Premiere Pro

# Bibliography

- [1] American Posture Institute. Bad posture: major cause for falls in the elderly. <https://americanpostureinstitute.com/bad-posture-major-cause-for-falls-in-the-elderly/>, 2015.
- [2] Centers for Disease Control and Prevention. Home and recreational safety. <https://www.cdc.gov/homeandrecreationalafety/falls/adultfalls.html>, 2017.
- [3] Dudley Brown. Experts say posture matters. <https://www.goupstate.com/news/20100511/experts-say-posture-matters-the-good--and-the-bad>, 2010.
- [4] World Health Organization. Falls. <https://www.who.int/news-room/fact-sheets/detail/falls>, 2018.
- [5] Philips Lifeline. Gosafe. <https://www.lifeline.philips.com/medical-alert-systems/gosafe.html>, 2019.
- [6] Goliveclip Global. Golivephone app. <https://www.goliveclip.eu/solutions/golivephone-app/>, 2019.
- [7] SENSE4CARE. Angel4 fall detection. 2019.
- [8] Marko Maslakovic. Wearables that monitor your posture. <https://gadgetsandwearables.com/2018/10/08/posture-corrector/>, 2018.
- [9] Upright. Upright go. <https://www.uprightpose.com/>, 2018.
- [10] Lumo. Lumo lift. <https://support.lumobodytech.com/hc/en-us/categories/201535363-Lumo-Lift>, 2018.
- [11] Prana Tech. Prana. <http://prana.co/>, 2018.
- [12] Muhammad Ehtisham Abid, Sara Kim, Tyler Newman, and Sebastian Rojas. *Low Power Skin Impedance Spectrometer*. Worcester Polytechnic Institute, 2016.
- [13] Russell Abbink and Craig Gardner. Getting under the skin. <http://spie.org/newsroom/getting-under-the-skin?SS0=1>, 2003.
- [14] NeuLog. Gsr logger sensor nul-217. <https://neulog.com/gsr/>, 2017.

- [15] Mindfield Body Systems. Mindfield esense skin response. <https://www.mindfield.de/en/Biofeedback/Products/Mindfield\%C2\%AE-eSense-Skin-Response.html>, 2018.
- [16] 3B Scientific. Skin resistance box. [https://www.a3bs.com/skin-resistance-box-1000576-u11393-3b-scientific,p\\_1109\\_14230.html](https://www.a3bs.com/skin-resistance-box-1000576-u11393-3b-scientific,p_1109_14230.html), 2019.
- [17] Carlyann Edwards. What is a decision matrix? definition and examples. <https://www.businessnewsdaily.com/6146-decision-matrix.html>, 2018.
- [18] Periklis Ntanasis, Evangelia Pippa, Ahmet Ozdemir, Billur Barshan, and Vasileios Megalooikonomou. Investigation of sensor placement for accurate fall detection. [https://www.researchgate.net/publication/318146579\\_Investigation\\_of\\_Sensor\\_Placement\\_for\\_Accurate\\_Fall\\_Detection](https://www.researchgate.net/publication/318146579_Investigation_of_Sensor_Placement_for_Accurate_Fall_Detection), 2017.
- [19] B Blank. Blank. <http://my.url.com/>, 2018.
- [20] M Vilas-Boas, P Silva, SR Cunha, and MV Correia. *Monitoring of bedridden patients: development of a fall detection too*. IEEE, 2013.
- [21] Victorian, Paediatric, Orthopaedic, and Network. Good posture and spine care in children. [https://www.rch.org.au/uploadedFiles/Main/Content/rheumatology/Good\\_posture\\_and\\_spine\\_care\\_in\\_children.pdf](https://www.rch.org.au/uploadedFiles/Main/Content/rheumatology/Good_posture_and_spine_care_in_children.pdf), 2011.
- [22] Analog Devices. Inertial measurement units (imu). <https://www.analog.com/en/products/sensors-mems/inertial-measurement-units.html>, 2018.
- [23] Analog Devices. Adxl362. <https://www.analog.com/en/products/adxl362.html#product-overview>, 2018.
- [24] Adafruit. L3gd20h triple-axis gyro breakout board. <https://www.adafruit.com/product/1032>, 2018.
- [25] Adafruit. Adafruit 9-dof accel/mag/gyro+temp breakout board - lsm9ds0. <https://www.adafruit.com/product/2021>, 2018.
- [26] Texas Instruments. Cc2650moda. <http://www.ti.com/product/CC2650MODA>, 2018.
- [27] Spark Fun. Atmega328. <https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf>, 2018.

- [28] Seed Studio. Getting started with ble shield 2. <http://www.seeedstudio.com/document/pdf/Getting\%20Started\%20with\%20BLE\%20Shield\%202.pdf>, 2018.
- [29] Ltd. Guangzhou HC Information Technology Co. Hc-06 datasheet. <https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf>, 2018.
- [30] Adafruit. Li-ion & lipoly batteries. <https://learn.adafruit.com/li-ion-and-lipoly-batteries/overview>, 2018.
- [31] Digikey. Rjd rechargeable li-ion batteries. [https://media.digikey.com/pdf/Data\%20Sheets/Illinois\%20Cap/RJD\\_Series\\_Rev2016.pdf](https://media.digikey.com/pdf/Data\%20Sheets/Illinois\%20Cap/RJD_Series_Rev2016.pdf), 2018.
- [32] Panasonic. Rechargeable batteries aaa 2-pack. [http://www.panasonicbatteryproducts.com/rechargeable\\_batteries/rechargeable-batteries-aaa-2-pack-copy/](http://www.panasonicbatteryproducts.com/rechargeable_batteries/rechargeable-batteries-aaa-2-pack-copy/), 2017.
- [33] Digikey. Cell type ml2430 specifications. <https://media.digikey.com/pdf/Data\%20Sheets/FDK/ML2430.PDF>, 2018.
- [34] WPI. Ece2799 course description. [https://users.wpi.edu/~sjbitar/ece2799/overview/ece2799\\_course\\_description.pdf](https://users.wpi.edu/~sjbitar/ece2799/overview/ece2799_course_description.pdf), 2018.
- [35] TechTarget. Bill of materials (bom). <https://searcherp.techtarget.com/definition/bill-of-materials-BoM>, 2018.
- [36] DigiKey Electronics. Digikey. [https://www.digikey.com/?&utm\\_adgroup=DigiKey&slid=&gclid=Cj0KCQiAwc7jBRD8ARIsAKSUBHJIWLZ29kC6opfGve2Y\\_7QJYcUUeAH1RhUdQoFOvvYVqdq0qXCc0e8aAu1nEALw\\_wcB](https://www.digikey.com/?&utm_adgroup=DigiKey&slid=&gclid=Cj0KCQiAwc7jBRD8ARIsAKSUBHJIWLZ29kC6opfGve2Y_7QJYcUUeAH1RhUdQoFOvvYVqdq0qXCc0e8aAu1nEALw_wcB), 2018.
- [37] Accounting Coach. What are manufacturing costs? <https://www.accountingcoach.com/blog/what-are-manufacturing-costs>, 2018.
- [38] Investopedia. Fixed cost. <https://www.investopedia.com/terms/f/fixedcost.asp>, 2018.
- [39] Arduino. Frequently asked questions. <https://www.arduino.cc/en/main/FAQ>, 2018.
- [40] MIT App Inventor. Hour of code with mit app inventor. <http://appinventor.mit.edu/explore/hour-of-code.html>, 2018.
- [41] ChRobotics. Understanding euler angles. <http://www.chrobotics.com/library/understanding-euler-angles>, 2018.

- [42] Adafruit. Adafruit powerboost 500 and charger. <https://learn.adafruit.com/adafruit-powerboost-500-plus-charger/pinouts>, 2018.
- [43] Developers. Android studio. <https://developer.android.com/studio>, 2018.
- [44] A Jew and S Paez. Storyboarding. in professional storyboarding. focal press. [https://proquest-safaribooksonlinecom.ezproxy.wpi.edu/book/illustration-andgraphics/9780240817705/1ot-overview/ch01\\_sec3\\_xhtml](https://proquest-safaribooksonlinecom.ezproxy.wpi.edu/book/illustration-andgraphics/9780240817705/1ot-overview/ch01_sec3_xhtml), 2017.

# Appendices



# Appendix A

## Decision Matrix

## Fall Detection and Posture Monitoring System

	EEG	Skin Impedence Spectrometer	ADC Design	Application Demo for Chips	Smart Shirt	Fall Detection System	Heart Rate Monitoring	DUI Prevention	Posture Monitoring System	LIDAR Crash Prevention
<b>Most important</b>										
Relevant / New to Industry	1	2	1	2	2	3	3	1	2	3
Solved an interesting/important problem	1	2	1	1	2	3	3	1	2	3
Want it to work / Not just function on time	3	3	3	3	2	3	3	2	2	3
Improving skills technical growth	2	2	1	1	2	3	3	2	2	3
Interesting for team	1	2	1	1	2	3	3	1	2	3
<b>Medium Importance</b>										
Paper at conf/Journal	1	2	1	1	1	2	2	1	1	2
A cool 60 seconds video	2	2	1	1	2	3	3	2	3	3
<b>Least important</b>										
Business / Patent / Start up	1	2	1	1	1	2	2	1	1	2
Leads to future work (MS, MQP)	1	2	1	1	1	2	2	1	2	2
Outstanding MQP Award	1	2	1	1	1	2	2	1	2	2
Total	33	47	28	31	39	65	65	40	45	65
										39

# Appendix B

## Arduino Code

```
#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM9DS0.h>
#include <Adafruit_Simple_AHRS.h>
#define LSM9DS0_XM_CS 10
#define LSM9DS0_GYRO_CS 9
#define LSM9DS0_SCLK 13
#define LSM9DS0_MISO 12
#define LSM9DS0_MOSI 11
// Use I2C, ID
Adafruit_LSM9DS0 lsm = Adafruit_LSM9DS0(1000);
#1000
Adafruit_Simple_AHRS ahrs(&lsm.getAccel(), &lsm.getMag());
int sleep_flag=0;
SoftwareSerial hc06(2,3);
int count=0;
int count_1=0;

void setup(){
  //Initialize Serial Monitor
  #ifndef ESP8266
    // will pause Zero, Leonardo, etc until serial console
    // opens
    while (!Serial);
  #endif
  if(!lsm.begin())
  {
    /* There was a problem detecting the LSM9DS0
    ... check your connections */
    Serial.print(F("Ooops, no LSM9DS0 detected
```

```

    ... Check your wiring or I2C ADDR!"));
    while(1);
}
Serial.begin(9600);
Serial.println("ENTER AT Commands:");
//Initialize Bluetooth Serial Port
hc06.begin(9600);
}
void loop(){
    //Write data from HC06 to Serial Monitor
    sensors_event_t accel, mag, gyro, temp;
    sensors_vec_t orientation;
    lsm.getEvent(&accel, &mag, &gyro, &temp);
    ahrs.getOrientation(&orientation);
    int flag;
    double a_x,a_y,a_z;
    double g_x,g_y,g_z;
    double M_A;
    a_x=(accel.acceleration.x)*(accel.acceleration.x);
    a_y=(accel.acceleration.y)*(accel.acceleration.y);
    a_z=(accel.acceleration.z)*(accel.acceleration.z);
    M_A=sqrt(a_x+a_y+a_z);
    double roll,pitch,heading;
    double roll_F, roll_K;
    roll = orientation.roll+0.4;
    pitch = orientation.pitch-3.7;
    heading = orientation.heading;

    if (hc06.available()){
        flag = (hc06.read());
        if(flag==2){
            roll_F = roll;
            flag = 5;
        }
        Serial.println(flag);
    }
    if (flag == 1){
        hc06.write(byte((M_A*10)-100));
        delay(220);
    }
}

```

```
if(flag==5){
    roll_K = roll - roll_F;
    hc06.write(byte(roll_K));
    delay(220);
}
if(flag==0){
    if(((sqrt((roll - roll_F)*(roll - roll_F)) > 4))){

        count = count + 1;
        Serial.println(count);
        //change the number if jump is not detecting!
        if (count >= 500){
            delay(200);
            hc06.write(7);
            delay(600);
            count = 0;
        }
    }
    else{
        count = 0;
    }
}

if((flag==0) && (sleep_flag==0)){
    if((sqrt(a_z)> 6.9 && sqrt(a_z)<10.2)||
    (sqrt(a_x)> 6.9 && sqrt(a_x)<10.2)){
        count_1 = count_1 + 1;

        if(count_1>=200){
            delay(200);
            hc06.write(6);
            Serial.println("Person Fell");
            sleep_flag = 1;
            delay(1000);
        }
    }
    else{
        count_1 = 0;
    }
}
```

```
if(sleep_flag == 1){  
    delay(200);  
    hc06.write(9);  
    delay(1000);  
    Serial.println("Fallen person");  
    if(sqrt(a_y)>9.2){  
        Serial.println("Stand up");  
        sleep_flag=0;  
        flag=0;  
    }  
}  
}
```

# Appendix C

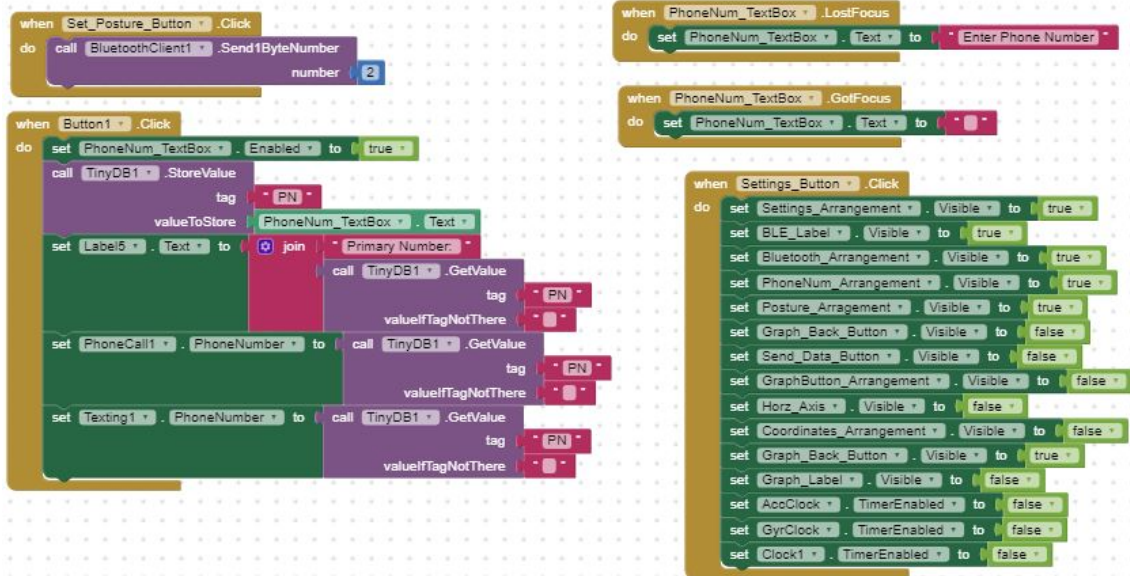
## App Code Blocks

Screen Initialization:

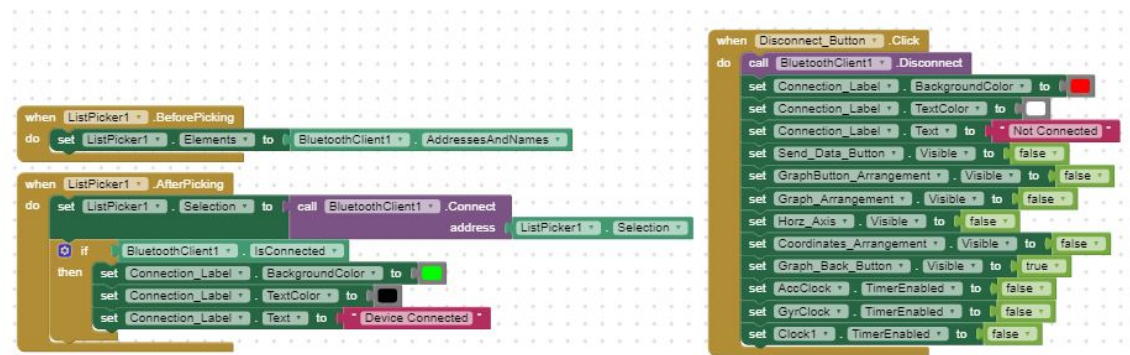


Settings Screen:

## Fall Detection and Posture Monitoring System



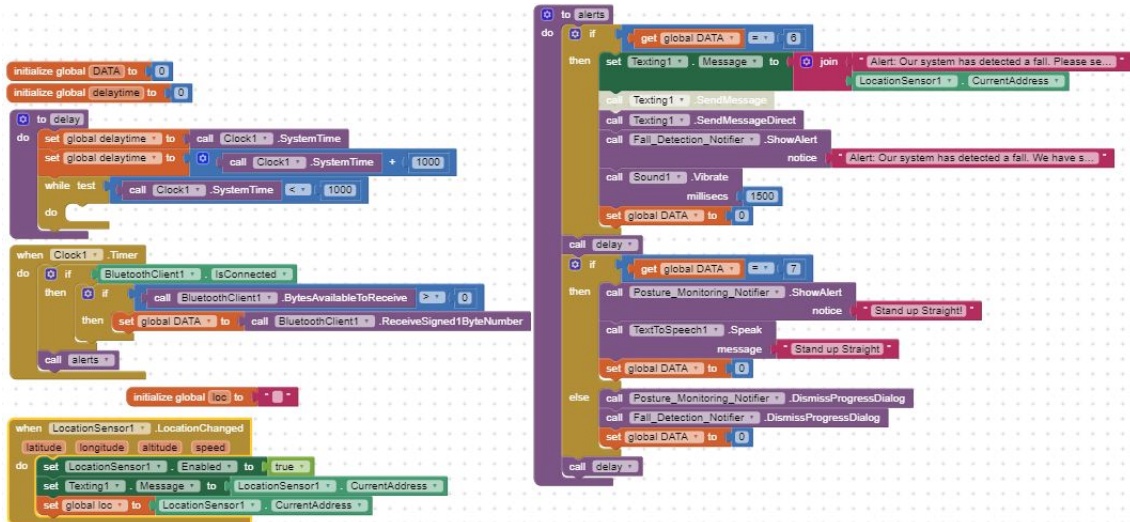
## Bluetooth Setup:



## Alerts:



## Fall Detection and Posture Monitoring System

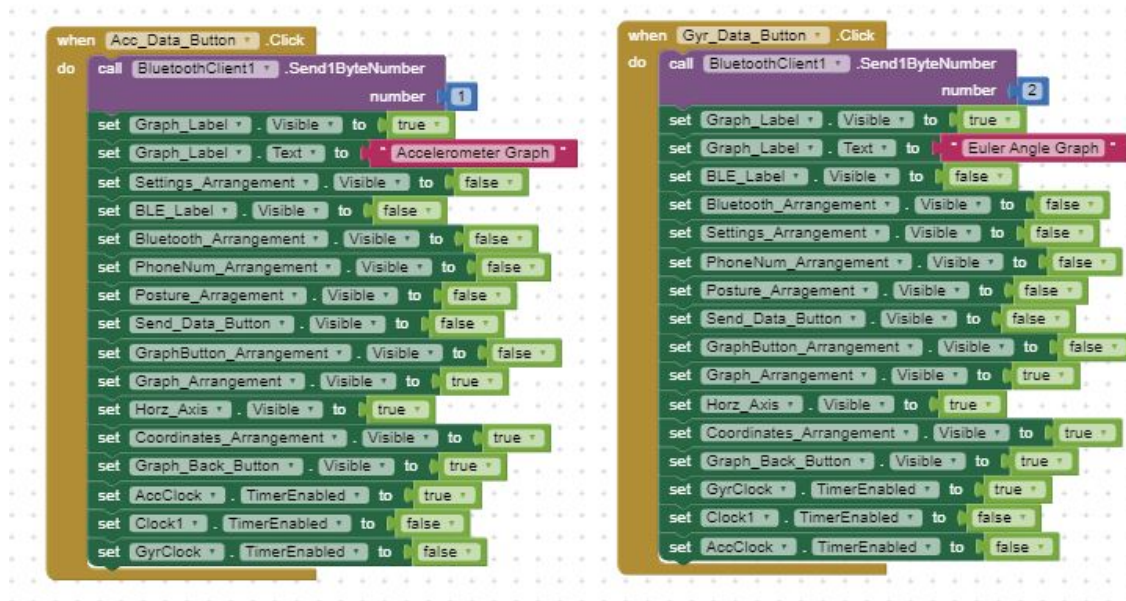


Back Button:

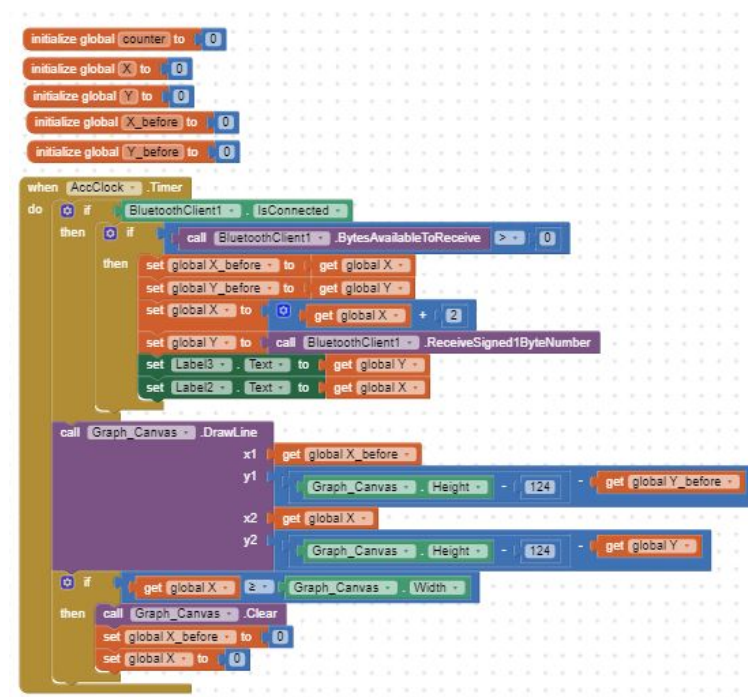


Graph Button:

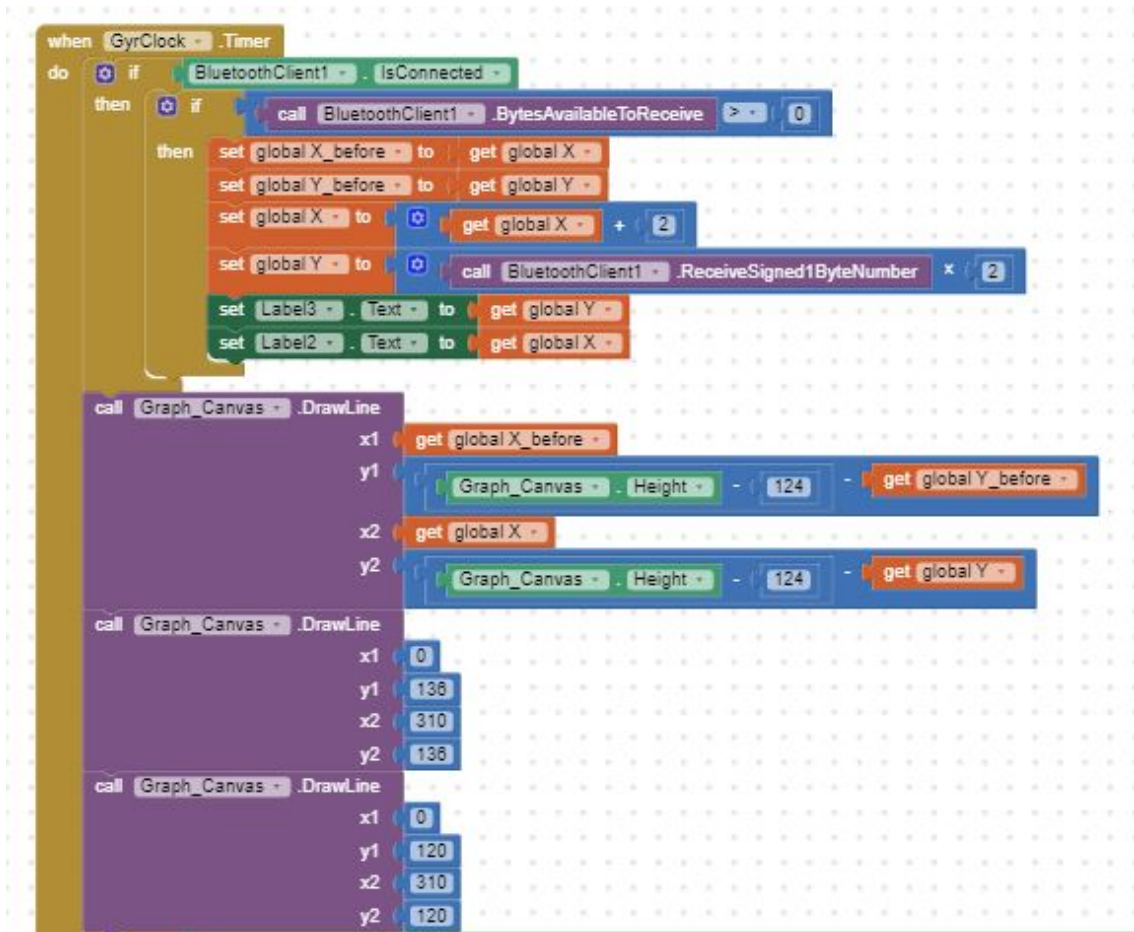
## Fall Detection and Posture Monitoring System

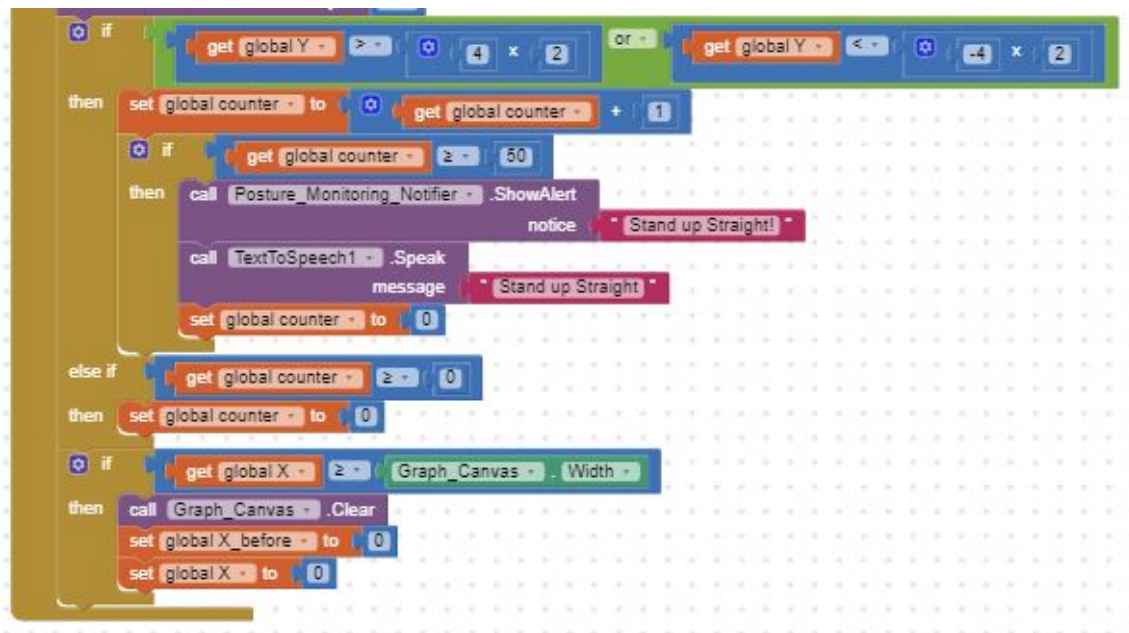


### Accelerometer Data Graph:



### Euler Angle Data Graph:







# Appendix D

## Python Script

Fall Detection Python Script:

```
from threading import Thread
import serial
import time
import collections
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import struct
import copy
#import pandas as pd

class serialPlot:
    def __init__(self, serialPort='/dev/ttyUSB0', serialBaud=38400,
        plotLength=100, dataNumBytes=2, numPlots=1):
        self.port = serialPort
        self.baud = serialBaud
        self.plotMaxLength = plotLength
        self.dataNumBytes = dataNumBytes
        self.numPlots = numPlots
        self.rawData = bytearray(numPlots * dataNumBytes)
        self.dataType = None
        if dataNumBytes == 2:
            self.dataType = 'h' # 2 byte integer
        elif dataNumBytes == 4:
            self.dataType = 'f' # 4 byte float
        self.data = []
        for i in range(numPlots): # give an array for each type of
            data and store them in a list
            self.data.append(collections.deque([0] * plotLength,
```

```

        maxlen=plotLength))
self.isRun = True
self.isReceiving = False
self.thread = None
self.plotTimer = 0
self.previousTimer = 0
# self.csvData = []

print('Trying to connect to: ' + str(serialPort) + ' at ' +
      str(serialBaud) + ' BAUD.')
try:
    self.serialConnection = serial.Serial(serialPort,
                                           serialBaud, timeout=4)
    print('Connected to ' + str(serialPort) + ' at ' + str(
          serialBaud) + ' BAUD.')
except:
    print("Failed to connect with " + str(serialPort) + ' at ' +
          str(serialBaud) + ' BAUD.')

def readSerialStart(self):
    if self.thread == None:
        self.thread = Thread(target=self.backgroundThread)
        self.thread.start()
        # Block till we start receiving values
        while self.isReceiving != True:
            time.sleep(0.1)

def getSerialData(self, frame, lines, lineValueText, lineLabel,
                  timeText):
    currentTimer = time.clock()
    self.plotTimer = int((currentTimer - self.previousTimer) *
        1000) # the first reading will be erroneous
    self.previousTimer = currentTimer
    timeText.set_text('Plot Interval= ' + str(self.plotTimer) + '
        ms')
    privateData = copy.deepcopy(self.rawData[:]) # so that the 3
        values in our plots will be synchronized to the same
        sample time
    for i in range(self.numPlots):
        data = privateData[(i*self.dataNumBytes):(self.

```

```

        dataNumBytes + i*self.dataNumBytes)]
    value, = struct.unpack(self.dataType, data)
    self.data[i].append(value) # we get the latest data point
    and append it to our array
    lines[i].set_data(range(self.plotMaxLength, self.data[i]))
    lineValueText[i].set_text('[' + lineLabel[i] + ']= ' +
        str(value))
# self.csvData.append([self.data[0][-1], self.data[1][-1],
    self.data[2][-1]])

def backgroundThread(self): # retrieve data
    time.sleep(1.0) # give some buffer time for retrieving data
    self.serialConnection.reset_input_buffer()
    while (self.isRun):
        self.serialConnection.readinto(self.rawData)
        self.isReceiving = True
        #print(self.rawData)

def close(self):
    self.isRun = False
    self.thread.join()
    self.serialConnection.close()
    print('Disconnected...')
    # df = pd.DataFrame(self.csvData)
    # df.to_csv('/home/rikisenia/Desktop/data.csv')

def main():
    portName = 'COM15'
    #portName = '/dev/ttyUSB0'
    baudRate = 38400
    maxPlotLength = 100 # number of points in x-axis of real time
    plot
    dataNumBytes = 4 # number of bytes of 1 data point
    numPlots = 3 # number of plots in 1 graph
    s = serialPlot(portName, baudRate, maxPlotLength, dataNumBytes,
        numPlots) # initializes all required variables
    s.readSerialStart() # starts background thread

# plotting starts below

```



```
pltInterval = 50 # Period at which the plot animation updates [
    ms]
xmin = 0
xmax = maxPlotLength
ymin = -(20)
ymax = 20
fig = plt.figure(figsize=(10, 8))
ax = plt.axes(xlim=(xmin, xmax), ylim=(float(ymin - (ymax - ymin)
    / 10), float(ymax + (ymax - ymin) / 10)))
ax.set_title('Fall_Detection_Test')
ax.set_xlabel("Time_(50mS)")
ax.set_ylabel("Acceleration_Rate_(M/S^2)")

lineLabel = ['Fall_Flag', 'X_Axis', 'Z_Axis']
style = ['r-', 'c-', 'b-'] # linestyles for the different plots
timeText = ax.text(0.70, 0.95, '', transform=ax.transAxes)
lines = []
lineValueText = []
for i in range(numPlots):
    lines.append(ax.plot([], [], style[i], label=lineLabel[i])[0])
    lineValueText.append(ax.text(0.70, 0.90-i*0.05, '', transform=
        ax.transAxes))
anim = animation.FuncAnimation(fig, s.getSerialData, fargs=(lines
    , lineValueText, lineLabel, timeText), interval=pltInterval) #
    fargs has to be a tuple

plt.legend(loc="upper_left")
plt.show()

s.close()

if __name__ == '__main__':
    main()
```

Posture Monitoring Python Script:

```
from threading import Thread
import serial
import time
import collections
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import struct
import copy
#import pandas as pd

class serialPlot:
    def __init__(self, serialPort='/dev/ttyUSB0', serialBaud=38400,
        plotLength=100, dataNumBytes=2, numPlots=1):
        self.port = serialPort
        self.baud = serialBaud
        self.plotMaxLength = plotLength
        self.dataNumBytes = dataNumBytes
        self.numPlots = numPlots
        self.rawData = bytearray(numPlots * dataNumBytes)
        self.dataType = None
        if dataNumBytes == 2:
            self.dataType = 'h' # 2 byte integer
        elif dataNumBytes == 4:
            self.dataType = 'f' # 4 byte float
        self.data = []
        for i in range(numPlots): # give an array for each type of
            data and store them in a list
            self.data.append(collections.deque([0] * plotLength,
                maxlen=plotLength))
        self.isRun = True
        self.isReceiving = False
        self.thread = None
        self.plotTimer = 0
        self.previousTimer = 0
        # self.csvData = []

    print('Trying to connect to: ' + str(serialPort) + ' at ' +
```

```

        str(serialBaud) + ' BAUD.')
    try:
        self.serialConnection = serial.Serial(serialPort,
            serialBaud, timeout=4)
        print('Connected to ' + str(serialPort) + ' at ' + str(
            serialBaud) + ' BAUD.')
    except:
        print("Failed to connect with " + str(serialPort) + ' at '
            + str(serialBaud) + ' BAUD.')

def readSerialStart(self):
    if self.thread == None:
        self.thread = Thread(target=self.backgroundThread)
        self.thread.start()
        # Block till we start receiving values
        while self.isReceiving != True:
            time.sleep(0.1)

def getSerialData(self, frame, lines, lineValueText, lineLabel,
    timeText):
    currentTimer = time.clock()
    self.plotTimer = int((currentTimer - self.previousTimer) *
        1000) # the first reading will be erroneous
    self.previousTimer = currentTimer
    timeText.set_text('Plot Interval = ' + str(self.plotTimer) + '
        ms')
    privateData = copy.deepcopy(self.rawData[:]) # so that the 3
        values in our plots will be synchronized to the same
        sample time
    for i in range(self.numPlots):
        data = privateData[(i*self.dataNumBytes):(self.
            dataNumBytes + i*self.dataNumBytes)]
        value, = struct.unpack(self.dataType, data)
        self.data[i].append(value) # we get the latest data point
            and append it to our array
        lines[i].set_data(range(self.plotMaxLength), self.data[i])
        lineValueText[i].set_text('[' + lineLabel[i] + '] = ' +
            str(value))
    # self.csvData.append([self.data[0][-1], self.data[1][-1],
        self.data[2][-1]])

```

```
def backgroundThread(self): # retrieve data
    time.sleep(1.0) # give some buffer time for retrieving data
    self.serialConnection.reset_input_buffer()
    while (self.isRun):
        self.serialConnection.readinto(self.rawData)
        self.isReceiving = True
        #print(self.rawData)

def close(self):
    self.isRun = False
    self.thread.join()
    self.serialConnection.close()
    print('Disconnected...')
    # df = pd.DataFrame(self.csvData)
    # df.to_csv('/home/rikisenia/Desktop/data.csv')

def main():
    portName = 'COM15'
    #portName = '/dev/ttyUSB0'
    baudRate = 38400
    maxPlotLength = 200 # number of points in x-axis of real time
    plot
    dataNumBytes = 4 # number of bytes of 1 data point
    numPlots = 3 # number of plots in 1 graph
    s = serialPlot(portName, baudRate, maxPlotLength, dataNumBytes,
        numPlots) # initializes all required variables
    s.readSerialStart() # starts background thread

    # plotting starts below
    pltInterval = 50 # Period at which the plot animation updates [ms
    ]
    xmin = 0
    xmax = maxPlotLength
    ymin = -(120)
    ymax = -20
    fig = plt.figure(figsize=(10, 8))
    ax = plt.axes(xlim=(xmin, xmax), ylim=(float(ymin - (ymax - ymin)
        / 10), float(ymax + (ymax - ymin) / 10)))
```

```
ax.set_title('Posture Monitoring Test')
ax.set_xlabel("Time (50mS)")
ax.set_ylabel("Euler Angle (Degree)")

lineLabel = ['Bad Posture Flag', 'Roll', 'Null']
style = ['r-', 'c-', 'b-'] # linestyles for the different plots
timeText = ax.text(0.70, 0.95, '', transform=ax.transAxes)
lines = []
lineValueText = []
for i in range(numPlots):
    lines.append(ax.plot([], [], style[i], label=lineLabel[i])[0])
    lineValueText.append(ax.text(0.70, 0.90-i*0.05, '', transform=
        ax.transAxes))
anim = animation.FuncAnimation(fig, s.getSerialData, fargs=(lines
    , lineValueText, lineLabel, timeText), interval=pltInterval) #
    fargs has to be a tuple

plt.legend(loc="upper left")
plt.show()

s.close()

if __name__ == '__main__':
    main()
```