Frankfurt University of Applied Science
Department 2: Computer Science and Engineering
Vietnamese - German University
Department of Electrical Engineering and Infomation Technology

DESIGN AND IMPLENTATION OF A WIRELESS FALL DETECTION NETWORK PROTOTYPE USING MEMS SENSORS

by

Ho Ngoc Khang Minh
Matriculation No.: 1148602

First Supervisor: Prof. Dr.-Ing. Kira Kastell
Second Supervisor: Dipl. -Ing Mohammad Reza Mansooji

Submitted in partial fulfillment of the requirements
for the degree of Bachelor Engineering in study program
Electrical Engineering & Information Technology,
Vietnamese - German University

Frankfurt am Main, Germany 2018

Disclamer

I hereby declare that this thesis is a product of my own work, unless otherwise referenced. I also declare that all opinions, results, conclusions and recommendations are my own and may not represent the policies or opinions of Vietnamese - German University and Frankfurt University of Applied Science.

Ho Ngoc Khang Minh

Abstract

Falling is becoming a serious problem to elderly people with the age of over 65, often causing unpredictable injuries such as hip fractures, head traumas, etc. More seriously, falls may lead to disability or even death on the victim if assistances from caregivers are not received in time. From this situation, there is a need for a wireless communication network which can automatically detect the fall and send an alarm to the caregivers if there is no safe signal from the victim within 10 minutes. There have been several algorithms such as detection of body orientation after a fall, image processing to detect the fall or applying machine learning techniques (Support Vector Machine (SVM), Markov model) in classifying between falling and other activities of daily living (ADL). However, these complex techniques require a huge amount of computations which can result in the system being overloaded or heavily delayed.

Addressing this problem calls for less computation-intensive techniques while retaining the accuracy and robustness of the system. One such approach is the combination of data from both accelerometer and gyroscope. The focus of this thesis is developing a wireless fall detection network which can combine the data from the above sensors to detect falls and distinguish them from fall-like activities. A comparison on the performance of this network with other existing works is also included to evaluate the robustness of the system.

Keywords: *elderly people, fall detection, accelerometer, gyroscope, wireless communication network*

Acknowledgements

I would like to express my sincere appreciation to Prof. Dr.-Ing. Kira Kastell, Vice President of Frankfurt University of Applied Science (FRA-UAS) for supervising, supporting and encouraging me during the time I conduct my senior project and final thesis. Also, I would like to thank Mr. Dipl.-Ing Mansooji - communication lab engineer at FRA-UAS, Dr. Vo Bich Hien - lecturer and Mr. Duong Huynh Bao - former control lab engineer at Vietnamese - German University for guiding me on technical and programming aspect. I also would like to thank my family for their supports during my bachelor degree time. This project will not be completed without their grateful guidances and supports. Thanks to their encouragements, I can improve myself, not only on the fundamental background in my major, but also necessary skills for my future career. I would like to mention all of my seniors and classmates who have helped me through this thesis.

Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 2

1.1 Introduction to Fall . . . 2
1.2 Definition of a Fall . . . 2
1.3 Cause of a Fall . . . 2
1.3.1 Intrinsic risk factors . . . 3
1.3.2 Extrinsic risk factors . . . 4
1.4 Consequence of Falls . . . 4
1.5 Challenges in Detecting Falls . . . 6
1.6 Objectives . . . 7
1.7 Thesis Organization . . . 7

2 Literature Review 9

2.1 Existing Commercial Devices . . . 9
2.2 Existing Products from Other Research Groups . . . 10

3 Statement of Problem and Methodology 12

3.1 Problem . . . 12
3.2 Methodology and Algorithm . . . 12

4 System overview 16

4.1 Hardware Components . . . 16
4.1.1 Sensor . . . 16
4.1.2 Microcontroller . . . 18
4.1.3 Wireless Module . . . 19
4.2 Software . . . 20

4.2.1 Embeddded Software on the Sensor Node 20
4.2.2 Data Acquisition Software 22
4.2.3 Data Analysis Software 23
4.3 System Intergration 23

5 Experimental and Procedure **26**
5.1 Data Collection Method 26
5.2 Data Analysis 26

6 Results and Discussion **31**
6.1 Results Assessment 31
6.2 Comparing the performance with existing works 33

7 Conclusion and Future Works **34**
7.1 Summary 34
7.2 Limitations 34
7.3 Future works 35

APPENDICES **35**

A C/C++ Code on the Sensor Node **36**

B Connection Handling Library **42**

C Register Map of MPU6050 **46**

D Python Code for Data Acquisition Program **49**

References **51**

List of Tables

4.1 Full-scale Range Table of Accelerometer 17
4.2 Sensitivity Scale Factor Table of Accelerometer 17
4.3 Full-scale Range Table of Gyroscope 17
4.4 Sensitivity Scale Factor Table of Gyroscope 18
4.5 Digital Low Pass Filter Configuration of the MPU-6050 Sensor 21

5.1 Detail body information of volunteers 26
5.2 Collected data of simulated falls in different directions 30

6.1 Experimental Results 31

List of Figures

2.1 Medical Guardian Fall Alert System [1] 9
2.2 myHalo Medical Alert System from MobileHelp^{TM} [2] 10
2.3 TEMPO 3.0 sensor node [3] 11
2.4 The final design of device from group research of Binh Nguyen and Jonathan Tomkun [4] 11

3.1 Fall detection algorithm 15

4.1 MPU-6050 by Invensense [5] 16
4.2 ESP8266 D1 Mini Board with an ESP8266EX MCU inside 18
4.3 Functional Block Diagram of ESP8266EX MCU[6] 19
4.4 Schematic of wireless communication system 20
4.5 Graphic User Interface of Data Acquisition Program 22
4.6 Data plotting with MATLAB 23
4.7 Principle of I^2C interface [7] 24
4.8 Wiring schematic of MPU-6050 Inertia Measurement Unit (IMU) [8] 24
4.9 Printed Circuit Board Design 25
4.10 Sensor Node 25

5.1 Fall forward data 27
5.2 Data for walking 27
5.3 Data for running 27
5.4 Data for climbing up stair 28
5.5 Data for walking down stair 28
5.6 Data for sitting down 29
5.7 Data for standing up 29

6.1 Jumping with strong force on the ground 32
6.2 The delay of normalized acceleration causes failure in fall detection 32
6.3 Fall happens, normalized acceleration does not fall below the UFT_{Acc} . . . 33

Acronyms

ADC analog-to-digital converter. 16

ADLs Activity of Daily Living. 26, 29

IMU Inertia Measurement Unit. ix, 16, 24

IoT Internet-of-Things. 18, 19

MCU Micro-Controller Unit. 18, 19, 24

NCOA the National Council of Aging. 4

SSF Sensitivity Scale Factor. 16, 17

U.S. United State of America. 5, 9

WHO World Health Organization. 2, 5

1. Introduction

1.1 Introduction to Fall

Nowadays, falling is becoming a dangerous issue which mostly causes injuries and even disability or fatal death on humans, especially the elderly. According to Hwang *et.at.* [9], in the United States, one-third to one-half of elderly people over 65 years old fall at least one time each year and two-third of them will do so again within the next 6 months. Every 11 seconds, there is an elderly person be treated in the hospital's emergency area due to fall-related injuries and every 19 minutes, one faller dies [10]. It is also reported that one out of every 200 falls results in a hip fracture on people with age among 65 and 69, increasing to one out of 10 for those aged 85 and more [11]. The most profound effect of falling is the loss of functioning associated with the dependency of the elderly for the rest of their life. Besides, a great amount of time and money have been spent on the medical treatments for the falls. Approximately $179 million were used as direct medical costs for treating fatal falls and $19 billion for non-fatal fall injuries within 2000 [12].

1.2 Definition of a Fall

Before starting a research about falls, it is necessary to understand about the meaning of the term *falls*. According to the World Health Organization (WHO), a fall is defined as an event which results in a person coming to rest inadvertently on the ground or floor or other lower level with or without loss of consciousness or injury [13].

1.3 Cause of a Fall

In recent years, a lot of researches have been carried out by different groups to find out the causes of falls. There have been different ways to classify the causes of a fall such as age and sex, drugs, cognitive functions, postural control, etc. However, falling is an unintentional action, many causes usually combine to produce a fall. Therefore, these causes can be divided into two main categories, intrinsic and extrinsic, to ease the complexity of research activities.

1.3.1 Intrinsic risk factors

Intrinsic factors are the factors within the body. It is also the physical aspect of the body that can cause injuries. Intrinsic factors include physical diseases such as cognitive impairment, postural hypotension, cardiovascular problems, etc.

Cognitive Impairment

It is well recognized that cognitive impairment is the most common cause of falls on humans, especially the elderly. Due to the research of Prudheim *et al.* [14] in 1981, fallers have in general been found to have a higher prevalence of cognitive impairment than non-fallers. It is also reported that humans with dementia are approximately 3 times more likely to fall than non-demented [15]. The reason why patients with cognitive impairment are likely to fall is that they have increased reaction time, increased postural sway and increased leaning balance which result in the decrease of muscle strength, worse balance and poorer mobility.

Postural Control

Postural control is a complex motor skill that requires the interaction of multiple body systems, which results in the ability to maintain postural orientation or postural stability [16], [17]. The impairment of postural control is indicated as one of significant causes of the fall during any activity at any age. Impaired control of gait and balance are two main aspects of the postural control that have been considered in many studies about falls recently. Subjectively assessed gait has been reported to be abnormal in many fall activities, and other studies using more objective measurements have found some relations between the impaired gait and balance and risk of falling [18].

Cardiovascular problem

Cardiovascular disorders are responsible for approximately 77% of patients with unexplained or recurrent falls associated with loss of consciousness [19]. The most common cardiac disorders linked to falls are carotid sinus syndrome, postprandial hypotension, orthostatic hypotension, vasovagal syncope, and bradyarrhythmias [20]. There is a fact that fallers with cardiovascular disorders have a greater mortality than those with non-cardiovascular or unknown causes [21].

1.3.2 Extrinsic risk factors

Extrinsic factors are those related to the environment such as lighting, walking surface, loose carpets, and high or narrow steps.

Dim lighting or glare

Dim lighting is one significant cause of fall on humans, especially on the elderly. While walking in the low light condition, the patient's visibility is reduced which prevents them from detecting the obstacles on the walking path, hence causing stumble and fall. Too bright lighting can also cause fall on humans because of creating glares and distorting the way object look.

Bad staircase design

Bad staircase design is also another extrinsic risk that causes the fall on humans. Too high or too low rise of staircases have caused a number of falls because people fail to perceive the abnormal elevation change or incur a misstep on descent. Besides, slippery surface of the treads can cause strip on patients while walking up or down the stairs. One more mistake of staircase design that can lead to the fall is lighting condition on the stair. Poor visibility and inadequate lighting can cause a user to misread the stair edge, resulting in faulty foot placement and falls.

1.4 Consequence of Falls

The consequences of falls are serious to humans at any age in any circumstance, but to the elderly, they have significance beyond that in younger people. There are different consequences related to the falls on humans including physical effects, psychological effects and other consequences such as dependency, hospital admission or economic consequences.

Physical consequences

Physical consequences of falls are always a significant aspect that many scientists have focused on in recent years. According to the the National Council of Aging (NCOA) of the U.S., falls are the leading cause of fatal injuries and the most common cause of non-fatal traumas [10]. There are several physical injuries such as bruise, fracture or head injury

reported to happen with patients who suffer from falls. In such injuries, hip fractures and head traumas are two common problems which are usually analyzed by researchers.

Hip fractures cause the greatest health problems and greatest number of death. It is reported that a quarter million hip fractures occur each year among people older than 50 years old in the U.S. but more common in women than men and increase in frequency with increasing age [18], [22]. Most patients with the hip fractures after falls are hospitalized, but about half of them cannot return home or live independently after that. In 1986, it costs for more than $3 billions for direct medical treatment of hip fractures [22].

Head injuries also usually happen to patients after suffering a fall. It is reported that people with the age of over 65 years old make up for 10-15% of admission to the hospital because of head injuries while three-quarters of them are caused by fall [18]. In California from 1996 to 1999, 71% of fall-related head injuries occurred in adults aged 65 and more [23]. The head trauma from a previous fall can increase the risk of repeated fall and head injuries in the near future.

Finally, the most serious impact of fall is fatal injuries causing death to a patient. According to data from the WHO, the fall morality rate of people with the age of over 65 in the United State of America (U.S.) (in 2003) was 36.8 per 100 000 citizens [24]. The main reason of these accidental death is that many patients had lied on the ground for several hours before receiving the help from caregivers.

Psychological consequences

Fall can result in long lasting psychological impact, called fear of falling again, more than just short-term injuries. Those people who have fallen one time before seem to have a feeling of vulnerability which limits them from their normal activities. They may be worried about falling and hurting themselves again, thus stop going out on their own or reduce their level of physical activities. Unfortunately, in attempting to reduce their risk of falling, they may increase the risk of falling because of the attenuation of physical functions and mobility.

Other consequences

Some research indicated that there are other consequences such as dependency, hospital admission or economic consequence. As already mentioned, the fear of falling again prevents patients from living alone and starting depending on the long-term nursing care. This consequence directly leads to increased dependency, which requires more time and financial resources from both government and family.

Falls are also the main reason for people to be admitted to hospital as an emergency case. Due to the information provided by the Accident Department at a sample of hospitals in England and Wales, in 1998, approximately 200 000 people over 60 are treated at this hospital per year because of a fall at home [25].

Finally, falls are also an economic burden of every country around the world, especially for low and lower middle countries. According to the U.S. Center for Disease Control and Prevention, in 2015, there was $50 billion being used for fall injuries and expected to increase as the population ages and may reach $67.7 billion by 2020 [10]. These costs are spent for hospitalization, medical, pharmaceutical, nursing home and other costs which are related to the medical treatments for fall patients.

1.5 Challenges in Detecting Falls

As mentioned in the previous section, most of patient's death is caused by the late assistance from the caregivers after fall accidents. Timely detection can minimize the negative impacts of falls on patients. However, depending on age and physical condition, different people have different gaits, different balance ability that result in several types of fall. These falls can happen in any direction at any speed. Therefore, it is still a challenge to design a system which can detect successfully all the falls of patients.

Another challenge in detecting the fall is that it is difficult for a system to distinguish a real fall from other fall-liked activities such as sitting, jumping or running, which also result in high acceleration. Therefore, it is required to collect data from different subjects; and then, there should be some analyses to find out the most common pattern of every fall. From that, researchers can design an algorithm to detect the falls based on this pattern. However, it is not easy to collect data for the system from the daily life on real human

bodies, since it is an accidental, dangerous action which can happen randomly at any time. Therefore, it is a overwhelming task for researchers to built a dataset for their system.

1.6 Objectives

From the above challenges, it is necessary to study and design a wireless sensor network which can automatically detect the fall of a patient and send an alarm to the caregivers instantly to reduce their time of arrival, and hence reduce the mortality rate. The sensor node is portable and attached to the chest of the user. According to research of Gjoreski *et al.* [26], chest and waist are the most common places to put the sensor in fall detection system. Beside that, Huynh *et al.* [27] also determined chest as the optimal location for the sensor node. In addition, wearing such small device on the chest would cause no discomfort on user in their daily activities. This sensor node contains one accelerometer and one gyroscope to provide information about the body's acceleration and angular velocity during the fall event. This sensor node can connect to the host computer via Wi-Fi with the help of a Wi-Fi module called ESP8266 from ESPRESSIF®company and send data directly to a monitoring software on the computer. Upon detecting a fall, the sensor node will wait for the safe signal from the patient within the next 10 minutes. If there is no response, an emergency email will be sent immediately to the caregiver. There is also a super bright red led on the sensor node to indicate the fall that helps the caregiver to easily find out the patient.

1.7 Thesis Organization

In chapter 1, basic knowledge about falls as well as the causes and consequences of them are introduced. It also provides the information about current challenges in detecting the fall and the motivation for this thesis. In chapter 2, a review of existing commercial fall detection devices on the market and researches related to this topic from other research groups is included. Chapter 3 describes the problem of fall detection and a method to solve this problem. Also included in this chapter is the description of the algorithm used in this thesis. Chapter 4 describes the system with both hardware and software perspectives. Chapter 5 and 6 describe the data collection and analysis procedure as well as the result

assessment of the experiments on a real human body. Finally, chapter 7 gives the conclusion for this thesis, the limitation of the current solution and recommendations for future works.

2. Literature Review

Nowadays, there are a lot of companies focusing on developing portable fall detection devices to suffice the need of the current society where there are more and more elderly people falling every day. These products are used widely by many people around the world and have positive effects on health-care systems of the corresponding countries. Beside that, numerous universities and research groups carried out studies on the topic of fall detection from various perspectives and have great contributions to the evolution of health monitoring services.

2.1 Existing Commercial Devices

One popular fall detection device, which is highly evaluated by customers, is the Medical Fall Alert in the *Home Guardian* option from Medical Guardian. It is designed in form of a lightweight wearable neck pendant. This pendant is waterproof and small enough that it has no disturbance on the user's daily activity. Medical Guadian provides a home system including one base unit connecting to the monitoring station through landline and one auto-alert pendant on the user's body. Whenever detecting a fall, the wearable pendant will send a signal to the base unit and the base station will announce *"Fall Detected, Press or Hold Button to Cancel"*. If the patient actually needs help, he(she) does not press the button. Then, the system will connect to operators at the monitoring center, they will ask if the patient needs help or not. If the patient cannot speak because of unconsciousness, the operators will automatically send emergency services to the patient's home. The monthly fee for this service is around $44.95 per month and it is used inside the U.S..

Figure 2.1: Medical Guardian Fall Alert System [1]

Another device on the market is myHalo^{TM} from MobileHelp^{TM}. This device has more advantages compared to Medical Guadian one because of full body monitoring functions. It can detect a fall, mornitoring heart rate, skin temperature, sleep/wake pattern, etc. and send an alert signal to the authorized contacts. This product supports the medical alert outside the home powered by a nationwide cellular network and does not require landline phones. It also provides precise location of patients when they go outside with integrated mobile GPS system. This enable emergency call center to pinpoint patients' location and direct medical support to them. This service costs from $41.95 per month and supports 24/7. However, because this system offers so many features, it seems to less focus on the fall detection.

Figure 2.2: myHalo Medical Alert System from MobileHelp^{TM} [2]

2.2 Existing Products from Other Research Groups

Beside many commercial fall detection devices being available on the market, there are still many products from different research groups surrounding the fall detection topic. A group of Qiang Li and colleagues has used TEMPO (Technology-Enabled Medical Precision Observation) 3.0 sensor node for their project [3]. This sensor node contains one tri-axial accelerometer and one tri-axial gyroscope and is controlled by an TI MSP430F1611 microcontroller. This group proposed an algorithm which combines the data from both accelerometer and gyroscope to reduce both false positive and false negative detection and

improving the fall detection accuracy. This system is low in computational requirements and able to give a real-time response. The authors stated that their method has difficulties in differentiating jumping into bed and falling against a wall with a seated posture.

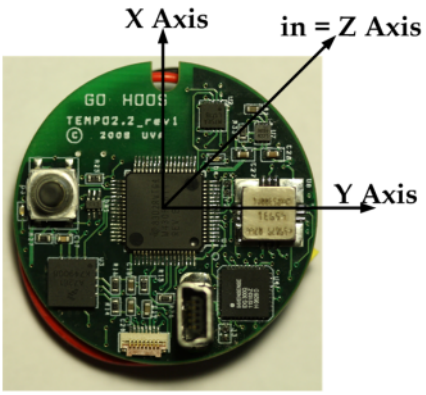


Figure 2.3: TEMPO 3.0 sensor node [3]

Binh Nguyen and Jonathan Tomkun designed and created a fall detection system for the elderly as a wearable monitoring device which can distinguish between fall and non-fall events [4]. This device can link wirelessly with a pre-programmed laptop computer or Bluetooth-compatible mobile phone. Upon detecting a fall, the device communicates wirelessly with the laptop or cellphone to call 911 or issued emergency contacts. This device can also detect abnormal tilt and warns the user to correct their posture to minimize the risk of falling. In addition to visual LED to alert the fall, there are also audio and tacticle alert options for people with hearing or seeing disabilities. Regarding the performance of the system, some actions have not been distinguished by one of four proposed algorithms to be falls or non-falls.

Figure 2.4: The final design of device from group research of Binh Nguyen and Jonathan Tomkun [4]

3. Statement of Problem and Methodology

3.1 Problem

Nowadays, there have been various research groups studying on the area of fall detection on humans. The majority of them focus on designing a new algorithm to successfully distinguish between fall and fall-like activities. From the first time, most research groups used algorithms mainly based on the data from only one accelerometer such as [28],[29],[30]. However, focusing only on the data from the accelerometer can result in many false positives as other activities such as sitting, running and jumping may cause large peak accelerations. After that, there are other algorithms which rely on the detection of the body orientation after the fall such as [31],[32]. The main drawback of these strategies is that the fall can be confused by activities with similar postures such as sleeping, reclining, etc. It is also less effective when a person's fall posture is not horizontal. Then, there are some groups who have nominated such complex algorithms that used Support Vector Machine (SVM)[33] and Markov model [34] to detect the fall. These techniques require a huge amount of computational resources which cause a delay which effects the robustness and accuracy of the system.

3.2 Methodology and Algorithm

From the disadvantages of prior works as mentioned above, this project proposes using an algorithm which combines data from one accelerometer and one gyroscope to detect the fall. The data from the accelerometer provides valuable information regarding body inertial change due to impact, while the gyroscope's data provides information about the body's rotational velocity during a fall event. This method helps improving the accuracy of the falling detection system without the need of complex computations.

This algorithm is nominated by Quoc T. Huynh *et al.* [35], which used data from both accelerometer and gyroscope sensors with predefined critical thresholds, to detect a

fall with maximum sensitivity and specificity. Two parameters used to analyze in this algorithm are normalized acceleration and normalized angular velocity.

Normalized acceleration is the total sum acceleration vector, named $\boldsymbol{Acc_{norm}}$, containing both static and dynamic acceleration components in different directions. This parameter is calculated with the formula below

$$\boldsymbol{Acc_{norm}} = \sqrt{(A_x)^2 + (A_y)^2 + (A_z)^2} \tag{3.1}$$

where A_x, A_y and A_z represent the acceleration of three directions *x, y, z*.

Normalized angular velocity is the total sum angular velocity vector, named $\boldsymbol{\omega_{norm}}$, containing the rotational velocity components in different directions. This parameter is calculated with the formula below

$$\boldsymbol{\omega_{norm}} = \sqrt{(\omega_x)^2 + (\omega_y)^2 + (\omega_z)^2} \tag{3.2}$$

where ω_x, ω_y and ω_z represent the angular velocity of three directions *x, y, z*.

When it stays in stationary condition, the body of the user gives the acceleration magnitude of a constant at +1g and angular velocity at $0^o/s$. When the user falls, there is a rapid change in the body's acceleration while the angular velocity produces a variety of signals in the fall direction. To detect the fall, the acceleration and angular velocity are compared with critical thresholds. These thresholds are defined as followed by Quoc T. Huynh *et al.*

(a) *Lower fall threshold(LFT)*: local minima for the Acc resultant of each recorded activity are referred to as the signal lower peak values (LPVs). The LFT_{Acc} for the acceleration signals is set at the level of the smallest magnitude lower fall peak (LFP) recorded.

(b) *Upper Fall Threshold(UFT)*: local maxima for the Acc and ω resultant of each recorded activity are referred to as the signal upper peak values (UPV_s). The UFT for each of the acceleration (UFT_{Acc}) and the angular velocity (UFT_{Gyro}) signals are set at the level of the lowest upper fall peak (UFP) recorded for the $\boldsymbol{Acc}$ and $\boldsymbol{\omega}$, respectively. The UFT_{Acc} is related to the peak impact force experienced by the body segment during the impact phase of the fall.

When a fall happens, the normalized acceleration first falls below the LFT_{Acc} threshold which indicates the start of a fall event. In the next 0.5 seconds, usually called fall windows, data from both accelerometer and gyroscope are compared to UFT_{Acc} and UFT_{Gyro}. The fall windows was obtained from the literature [36][37]. If the magnitude of both parameters exceed the two upper thresholds, a fall is detected. However, if only one of these conditions is not satisfied, there is no fall detection. Quoc T. Huynh et al. [35] have proven, with practical experiments on real subjects, that the three conditions above just happen simultaneously only with a fall but not other activities like standing, running, jumping of sitting up. Figure 3.1 summarizes the main steps of this algorithm.

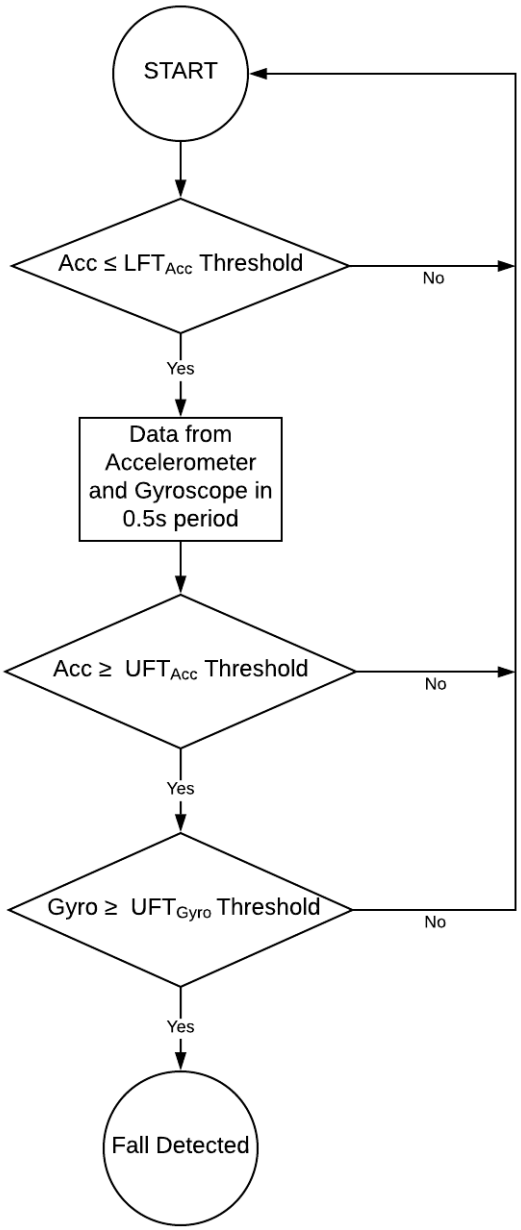


Figure 3.1: Fall detection algorithm

4. System overview

4.1 Hardware Components

4.1.1 Sensor

Since the algorithm requires information of both acceleration and angular velocity, an IMU named MPU6050 from Invensense®, which contains one accelerometer and one gyroscope, has been chosen for its convenience. This sensor can also accepts inputs from an external 3-axis compass via I^2C sensor bus to provide a complete 9-axis MotionFusion^{TM} output.

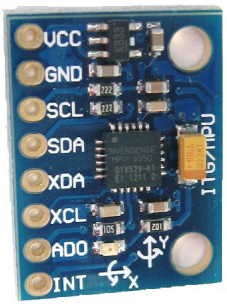


Figure 4.1: MPU-6050 by Invensense [5]

The accelerometer inside the MPU6050 board can measure the acceleration in four different programmable full-scale ranges ($\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$). An integrated 16-bit analog-to-digital converter (ADC) supports simultaneous sampling of the accelerometer while requiring no external multiplexer. This accelerometer normally operates at a very small current of 500 µA with low power consumption. According to the data-sheet given by Invensense®, the raw output of the accelerometer is displayed in form of LSB/g. Divided with an appropriate Sensitivity Scale Factor (SSF), we can obtain the acceleration in g - standard gravity unit, which is used in this project. Each full scale has its own SFF. Table 4.1 shows different full-scales of the accelerometer and table 4.2 shows the SSF for each scale.

Scale	TYP	UNIT
AFS_SEL=0	±2	g
AFS_SEL=1	±4	g
AFS_SEL=2	±8	g
AFS_SEL=3	±16	g

Table 4.1: Full-scale Range Table of Accelerometer

Scale	Sensitivity Scale Factor	UNIT
AFS_SEL=0	16,384	LSB/g
AFS_SEL=1	8,192	LSB/g
AFS_SEL=2	4,096	LSB/g
AFS_SEL=3	2,048	LSB/g

Table 4.2: Sensitivity Scale Factor Table of Accelerometer

With AFS_SEL being bits of the accelerometer configuration register of the sensor. These bits are used to select the full scale range of the accelerometer.

The gyroscope integrated on the same board can measure the angular velocity at 4 programmable full-scale ranges ($\pm 250^o/s$, $\pm 500^o/s$, $\pm 1000^o/s$ and $\pm 2000^o/s$). This sensor has an external sync signal which supports image, video capturing and GPS synchronization. Its normal operating current is 500 µA which enables the reduction on power consumption. The raw output of the gyroscope is displayed in form of LSB/($^o/s$). Divided with an appropriate Sensitivity Scale Factor (SSF), we can obtain the angular velocity with unit of ($^o/s$). Each full scale has its own SFF. Table 4.3 shows different full-scale of the gyroscope and table 4.4 shows the SSF for each scale.

Scale	TYP	UNIT
FS_SEL=0	±250	$^o/s$
FS_SEL=1	±500	$^o/s$
FS_SEL=2	±1000	$^o/s$
FS_SEL=3	±2000	$^o/s$

Table 4.3: Full-scale Range Table of Gyroscope

Scale	Sensitivity Scale Factor	UNIT
FS_SEL=0	131	LSB/($^o/s$)
FS_SEL=1	65.5	LSB/($^o/s$)
FS_SEL=2	32.8	LSB/($^o/s$)
FS_SEL=3	16.4	LSB/($^o/s$)

Table 4.4: Sensitivity Scale Factor Table of Gyroscope

With FS_SEL being bits of the gyroscope configuration register on the sensor. These bits are used to select the full scale range of gyroscope.

After getting acceleration and angular velocity in three different directions from two sensors, normalized acceleration and normalized angular velocity are calculated with formulas (3.1) and (3.2) and compared with thresholds to detect the fall.

4.1.2 Microcontroller

To perform all data processing and communications, the ESP8266EX Micro-Controller Unit (MCU) integrated with L106 32-bit RISC processor from Tensilica is chosen. This MCU archives extra low power consumption and reaches a maximum clock speed up to 160 MHz, which is specially designed for mobile and Internet-of-Things (IoT) applications. The ESP8266EX MCU supports most of popular interfaces such as GPIO, UART, I^2C, ADC, PWM etc. to connect with external peripheral devices. Figure 4.3 shows the functional block diagram of ESP8266EX.

Figure 4.2: ESP8266 D1 Mini Board with an ESP8266EX MCU inside

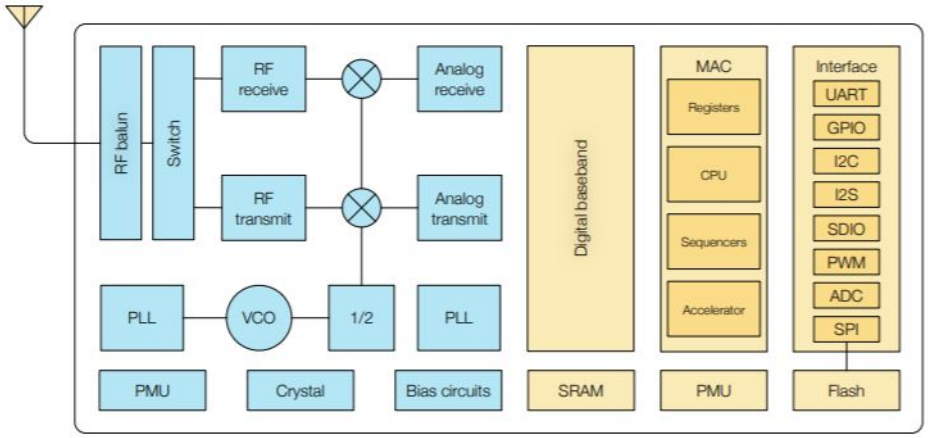


Figure 4.3: Functional Block Diagram of ESP8266EX MCU[6]

The ESP8266EX MCU integrates memory units including one 50kB SRAM and one 16MB external SPI flash that are sufficient for most IoT applications. The SRAM size is less than 50kB when the MCU is working under Station Mode or connecting to a router. All user programs are stored in the external SPI flash and non-volatile after switching off the power.

4.1.3 Wireless Module

ESP8266EX MCU integrates a Wi-Fi module in which is implemented TCP/IP protocol with the full 802.11 b/g/n WLAN MAC standard and Wi-Fi Direct specification. The ESP8266EX is designed with advance power management technologies. The low-power architecture operates in three modes: active mode, sleep mode and deep-sleep mode. ESP8266 consumes only20 µA in deep-sleep mode and the standby power consumption below 1.0mW. These two features make the ESP8266 Wi-Fi module fully compatible with mobile wireless applications that require low power dissipation.

In this project, a standard TCP/IP protocol has been chosen because of its highly reliable connection and its ability to control data congestion. Additionally, this system also requires that the received data on the monitoring computer must be in the same order as the original one on the sensor module. Therefore, TCP/IP is chosen because of its capability of ordering the received data and error recovery. Whenever there is an error on the connection, all the erroneous packets are retransmitted to the destination and rearranged in the correct order.

A brief description of the wireless communication system: there is a TCP server running on the monitoring computer and listening to the requests from the clients. The ESP8266 module (programmed in *client mode*) continuously sends sensors' data to the server via a Wi-Fi connection. All data is captured by a Python program and plotted on a visual graph. The received data is also stored in a database to serve later data analysis and algorithm optimization.

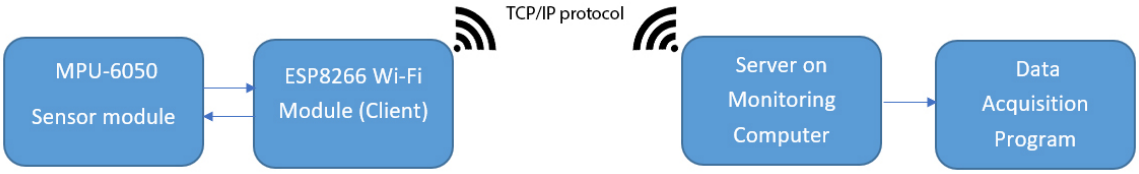


Figure 4.4: Schematic of wireless communication system

4.2 Software

4.2.1 Embeddded Software on the Sensor Node

On the sensor node, the micro-controller board is programmed to read the data from sensors, calculate normalized parameters and send data to the host computer. First, the microcontroller communicates with sensors, using I^2C protocol with the help of MPU6050 library written by Jeff Rowberg [38], to set clock source and full scale range for two sensors. It is also important to set the sensor board to wake-up mode in this step to make sure it is working. After successfully connecting, the frequency of these two sensors are set to 1kHz by configuring the digital low pass filter with the *setDLPFMode* function from the library. Normally, these sensors operate at the frequency of 8kHz, which means there are 8000 output samples within one second. That is too much compared to the requirement of a fall detection system and usually causes congestion and lost data during the transmission from the sensor node to the host computer. To avoid this problem, it is necessary to slow down the internal sampling rate of the two sensors. See the table 4.5 for the digital low pass filter configuration.

	Accelerometer		Gyroscope		
DLPF_CFG	Bandwidth	Delay	Bandwidth	Delay	Sample Rate
0	260Hz	0ms	256Hz	0.98ms	8kHz
1	184Hz	2.0ms	188Hz	1.9ms	1kHz
2	94Hz	3.0ms	98Hz	2.8ms	1kHz
3	44Hz	4.9ms	42Hz	4.8ms	1kHz
4	21Hz	8.5ms	20Hz	8.3ms	1kHz
5	10Hz	13.8ms	10Hz	13.4ms	1kHz
6	5Hz	19.0ms	5Hz	18.6ms	1kHz

Table 4.5: Digital Low Pass Filter Configuration of the MPU-6050 Sensor

Next, like other sensors, both gyroscope and accelerometer always have non-zero errors even when leveling and need to be calibrated with a set of offset values before being used. These offsets are calculated by another program included in the MPU6050 library and set into appropriate sensor's registers with the provided function from the library. See the appendix C for the register map of the MPU6050 sensor.

After initializing the sensors, the sensor node establishes a wireless connection with the local network and host computer. While the host computer is not connected, the sensor node continuously calls for a connection and the LED indicator on the micro-controller board keeps brighting until receiving a successful connection. Once the connection is established, the sensor node turns off the indicator LED for a while and then starts blinking while transferring data. All the connection handling functions are included in the *Connection Handling* library written by myself. See the Appendix B.

Finally, the micro-controller starts reading data from the two sensors with the function *getMotion6*. The output data is used to calculate the normalized acceleration and angular velocity. Whenever the normalized acceleration falls below the LFT_{Acc} threshold, the first flag is set true to indicate the start of a fall event and a 500ms timer starts counting. For the next 500ms, if normalized acceleration and angular velocity exceed the UFT_{Acc} and UFT_{Gyro} respectively, the second and third flags are also set true. When the timer is executed, all flags are checked. If three flags are all true, a fall is detected. The sensor node will send an emergency email to the caregivers while it starts blinking the super bright red led simultaneously. That red led helps the caregivers to find the patient when they reach the patients' location. If one condition is not true, the system would reset all flags and start a new detection loop. All the reading data are also converted into strings, concatenated

in comma-separated form and sent to the host computer for monitoring. Each message is ended with a new line character to separate them on the PC side. See Appendix A for the main code on the sensor node.

4.2.2 Data Acquisition Software

Data acquisition software is written in Python programming language with Pycharm environment because of its convenience and full support. This software starts an always-run server on the host computer and waits for the requests from the client. When receiving a message from the client, the software de-concatenates this message into two arrays which contain acceleration and angular velocity values separately. Data from two arrays are then converted back to float form and visualized on the graph. These values are also written to text files in comma-separated (.CSV) form and stored for future analysis with MATLAB. Figure 4.5 shows the graphic user interface of this software. See the Appendix D for the code.

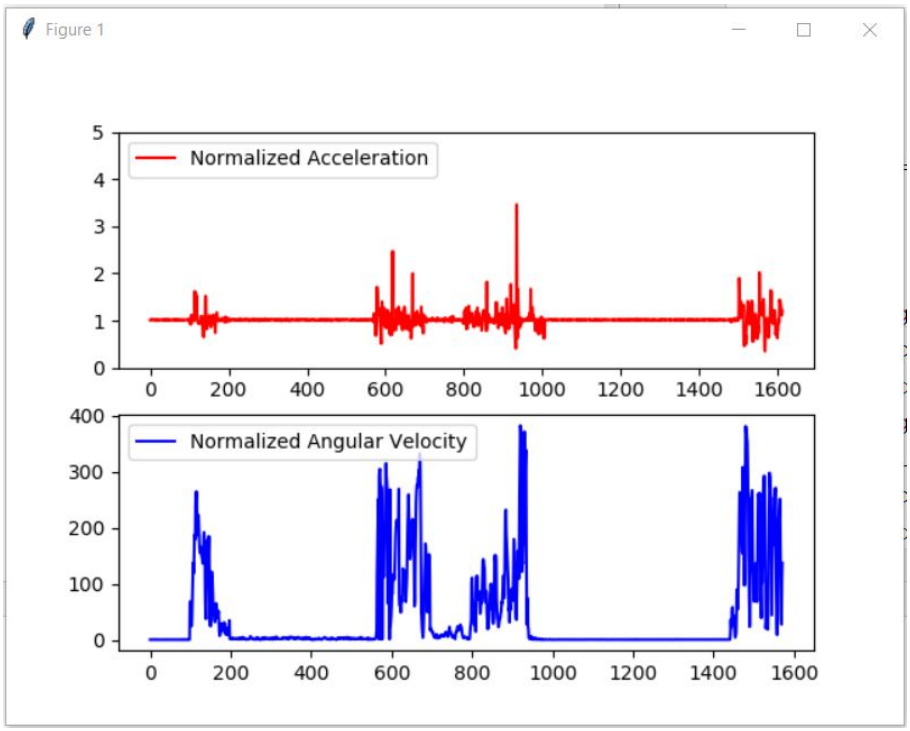


Figure 4.5: Graphic User Interface of Data Acquisition Program

4.2.3 Data Analysis Software

MATLAB is used to analyze the collected data from the experiments. Data stored in the CSV file is first loaded to MATLAB arrays with MATLAB function *csvread*. Data after being loaded to arrays is visualized with the plotting function supported by MATLAB. While working with the data, we can use the data cursor tool provided by MATLAB to figure out the valued of upper and lower peaks (Figure 4.6).

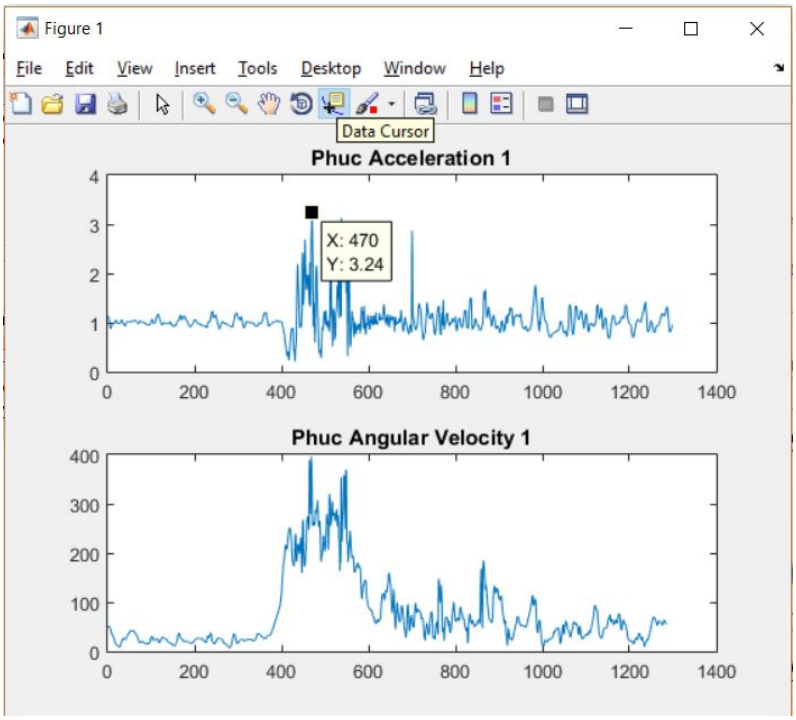


Figure 4.6: Data plotting with MATLAB

4.3 System Intergration

The communication between micro-controller and sensor board is established with I^2C interface. This interface uses only two wires named SCL(serial clock) and SDA(serial data) and both of them need to be pulled up with resistors to +Vdd. I^2C interface is popular to use because of its ability to support low-speed device such as micro-controller, EEPROM, I/O interface or A/D and D/A converters.

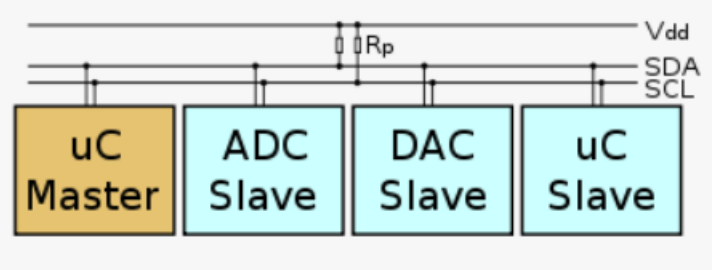


Figure 4.7: Principle of I^{2}C interface [7]

I^{2}C transfers every 8 bits (byte) and uses 7-bit address to communicate with a certain device. Bit 0 is set to 0 or 1 to define the signal reading from or writing to the device. Every device has its own unique address and multiple devices (slaves) can connect to a single MCU (master) on only one I^{2}C bus. With this feature, both the accelerometer and gyroscope can connect to the ESP8266EX MCU with only one bus. Figure 3.3 shows the connection between MPU6050 IMU and ESP8266 board.

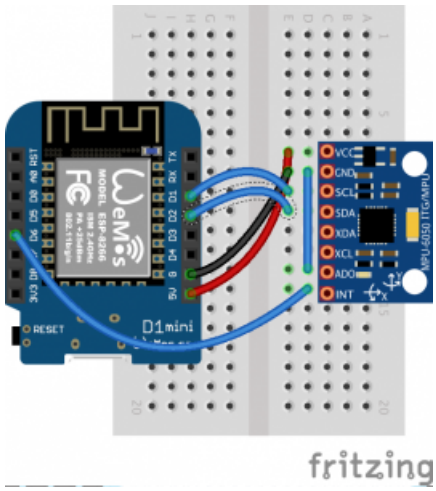


Figure 4.8: Wiring schematic of MPU-6050 IMU [8]

To suffice the compact and portable requirement of the sensor node, sensor board and micro-controller are connected together on the Printed Circuit Board (PCB) with the dimension of 60x40mm (Figure 4.9). All the sensor node is powered by a power-shield especially designed for the ESP8266 D1 Mini Board. This shield supports the Lithium-Ion Polymer battery with JST-PH connector and allows the user to re-charge the battery with a USB cable. It is convenient for the user to charge the device everyday with the micro-USB cable which can be found everywhere. Figure 4.8 is the final design of the sensor node.

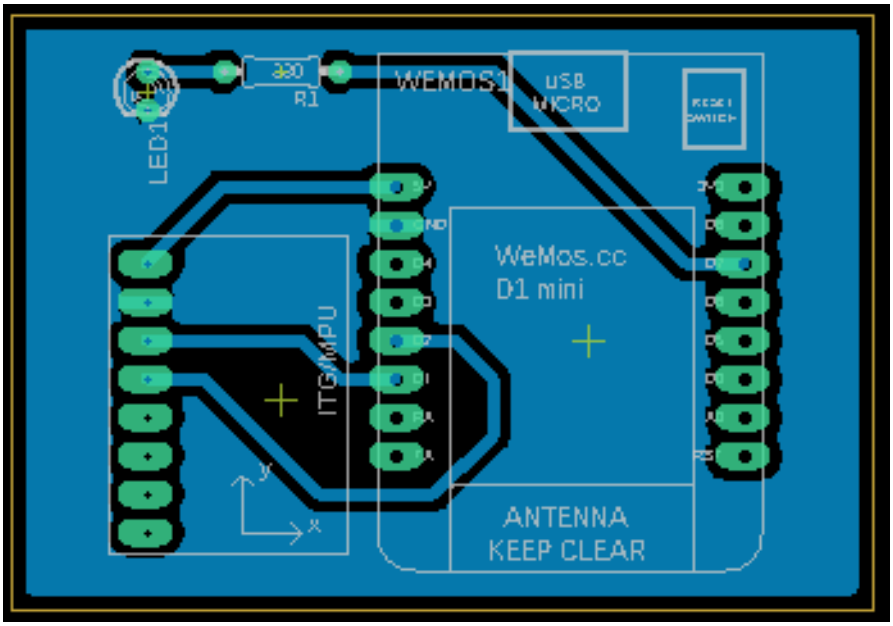


Figure 4.9: Printed Circuit Board Design

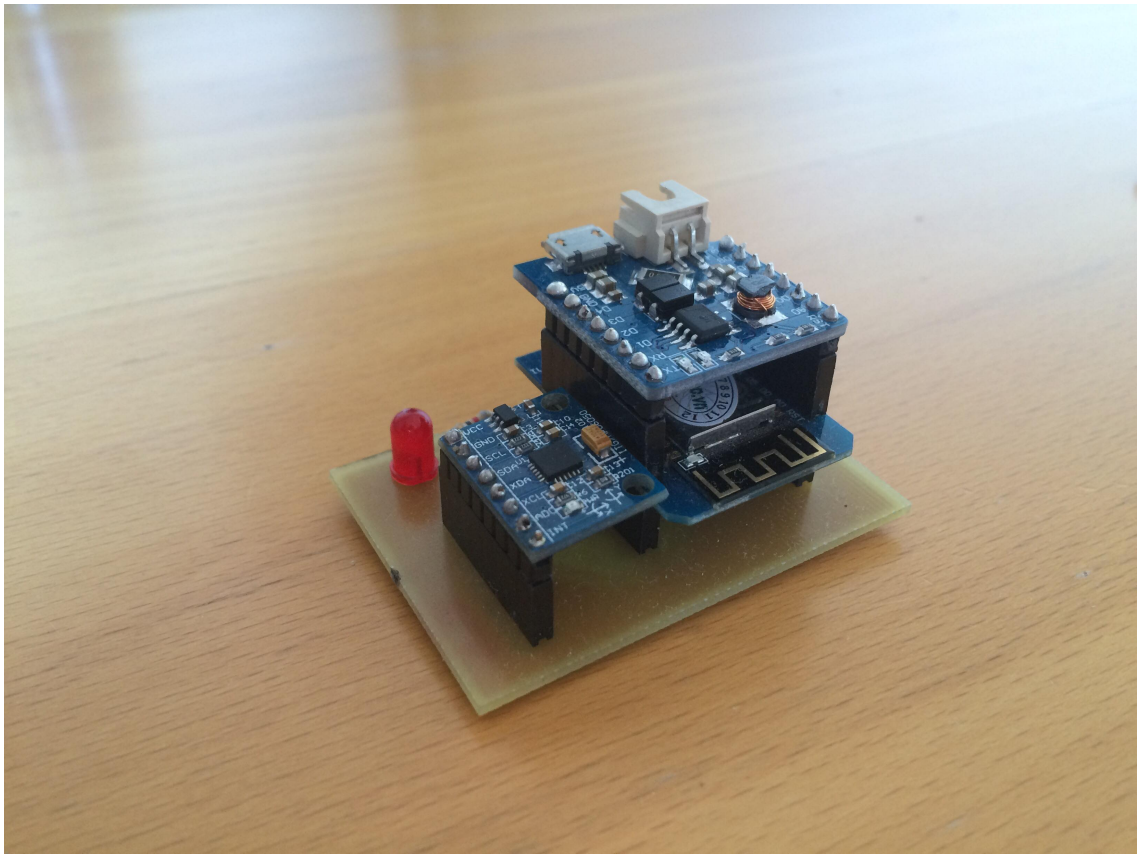


Figure 4.10: Sensor Node

5. Experimental and Procedure

5.1 Data Collection Method

To collect the data of falls and fall-liked activities for the development and optimization of the fall detection algorithm, experiments were performed on 5 young healthy subjects (male volunteers, age of 22 years old, weight from 55 to 90kg and height from 167 to 179 cm) to simulate the Activity of Daily Living (ADLs) and falls. These subjects are numbered from S1 to S5 with detail information provided below.

Subject	Age(years)	Height	Weight(Kg)	Gender
S1	22	167	55	M
S2	22	179	71	M
S3	22	170	90	M
S4	22	178	65	M
S5	22	175	75	M

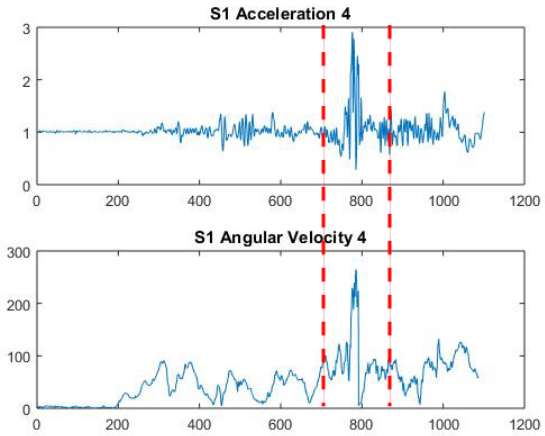
Table 5.1: Detail body information of volunteers

All the subjects were asked to perform different movements such as standing, sitting-down/standing-up, walking, running, lying on the bed, going up/down the stair, jumping and different fall tests including forward, backward, right/left side way. The sensor node was attached to the center of chest, where was determined to be the optimal place for sensor placement. To ensure the safety of the volunteers, all the experiments were conducted on a 36-cm high cushion at the dormitory.

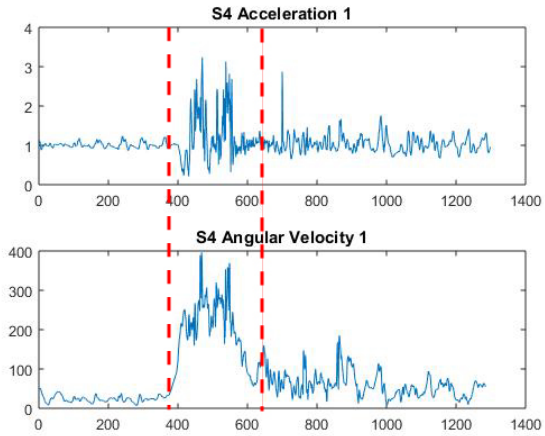
5.2 Data Analysis

Figure 5.1 is the visualized data of two typical fall events collected on volunteer S1 and S4. From the two datasets in the figure, it is clear that when a fall happen, the normalized acceleration first falls to a certain point below the normal acceleration to indicate the start of a fall. Within the next 0.5s, the normalized acceleration from two experiments both rise up to the upper peak at around 2.8 - 3.2 g and then return to the normal level. During the falls, the normalized angular velocity of both experiments also reach the peaks at around

264.9 - 396$^o/s$. This pattern can be met in most of other fall experiments with other volunteers (see appendix E) and agree with the pattern that was stated by the algorithm in section 3.2. It is a confirmation for the theory founded by Huynh *et al.* [35].



(a) S1 falls forward



(b) S4 falls right sideway

Figure 5.1: Fall forward data

In comparison with falls, the normalized acceleration data of walking (figure 5.2) and running (figure 5.3) are more fluctuating and repeating. However, the peak values of the acceleration in these two activities are just around 1.5 - 2 g and more less than those of falls. This characteristic helps the fall detection device to differentiate the falls with walking or running - one activity that is usually supposed to be confused by most of fall detection devices because of the high resulted acceleration. With the angular velocity, the peak values of running seem to be greater than walking but still less than falls.

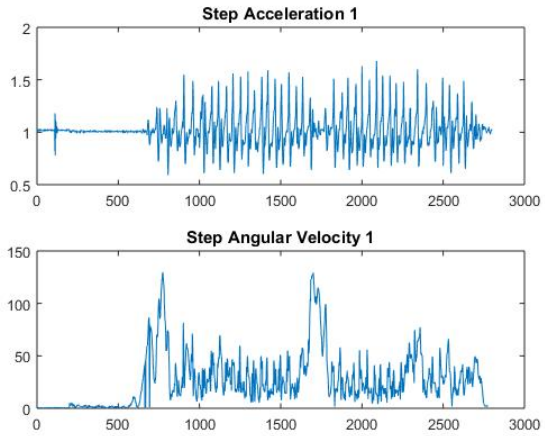


Figure 5.2: Data for walking

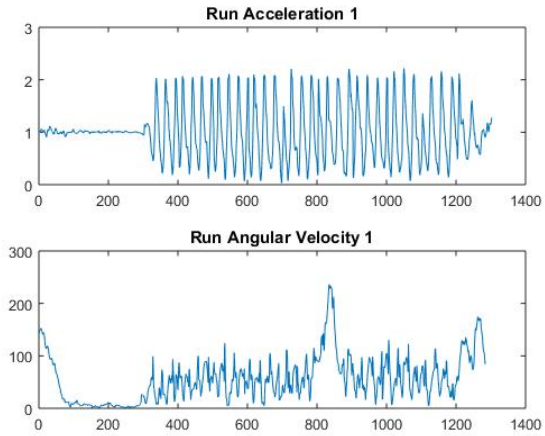


Figure 5.3: Data for running

Similarly, for the climbing up and down the stair, the normalized acceleration is just around 1.3 - 1.5 g and far more less than those acceleration values resulted by falls. The normalized angular velocity of these two activities are only around less than 50 with some abnormal peaks up to $100^o/s$. This common characteristic of these two activities enables the fall detection device to completely distinguish them with falls and cause no false positive on system.

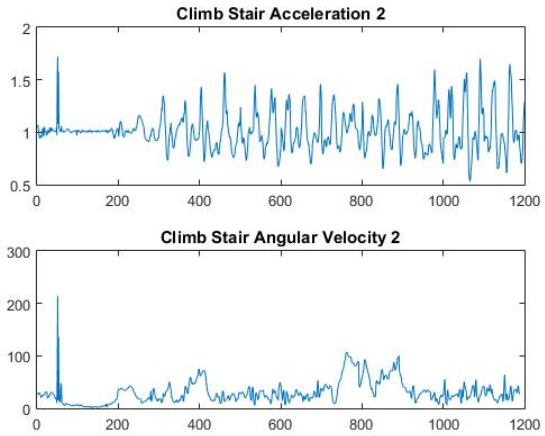


Figure 5.4: Data for climbing up stair

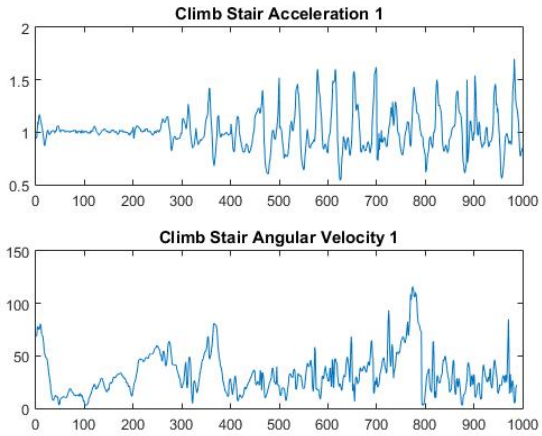


Figure 5.5: Data for walking down stair

Regarding the sitting-down, the pattern of this activity is the most similar to falls. Looking in the acceleration, it first falls down to a certain lower point and then immediately go up to the highest peak. However, the peak of this activity is just around 1.7g - smaller compared to the peak of falls in previous experiments. In contrast, the acceleration of standing up is first go up to the highest peak and then fall to a lower point. This pattern is completely different compared to the one of falls . The normalized angular velocity values of both sitting-down and standing-up activities are clearly less than those of falls. Therefore, sitting-down and standing-up are distinguishable from a typical fall.

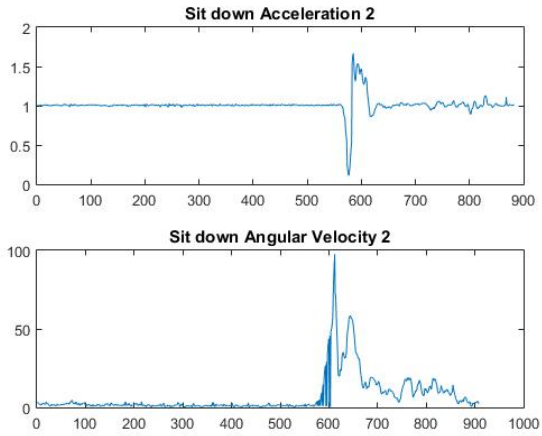


Figure 5.6: Data for sitting down

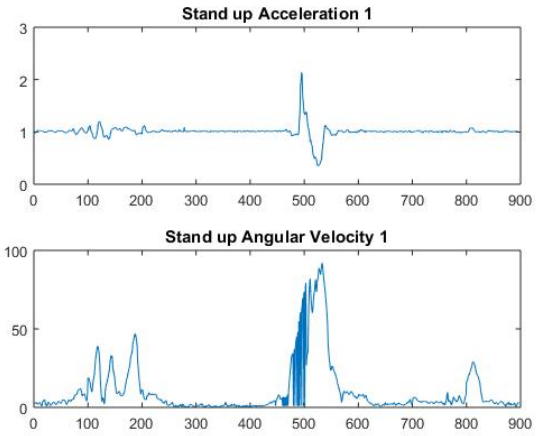


Figure 5.7: Data for standing up

From the difference between falls and those common ADLs mentioned above, it is necessary to find out the optimal threshold values for the fall detection algorithm, to improve the accuracy of the system. The collected data of falls on five volunteers are represented in the table 5.2. This table demonstrates the comparison between the peak values resulted by different subjects when they performed fall simulations. This table helps me to determine three thresholds by using observation method.

First, it is important to find the optimal Lower Fall Threshold for the acceleration (LFT_{Acc}), because it is the first condition to start detecting a fall. Improper selection of the LFT_{Acc} may lead to the miss of the fall event and cause failure of fall detection. From the given data in table, most of the lower peaks are below 0.35 g. To ensure the system can detect most of the falls, I decided to choose the Lower Fall Threshold for the acceleration at 0.35g.

For the upper peaks of the acceleration in all kind of falls, the value is in the range of 2.31 - 3.46 g. This data helped me to choose the Upper Fall Threshold for the acceleration (UFT_{Acc}) at about 2.75, to make sure the acceleration during 90% of all falls will exceed the upper threshold and satisfy the second condition of the fall detection algorithm.

The final step is choosing the upper fall threshold for the angular velocity as a confirmatory threshold for the fall detection algorithm. From the data in table 5.2, it is suitable to choose the UFT_{Gyro} at around $254^o/s$ to ensure all the falls will be detected.

Subject	experiment	Lower Peak Acc (g)	Upper Peak Acc (g)	Upper Peak Gyro ($^o/s$)
S1	1	0.23	3.24	282.9
	2	0.26	2.87	278.9
	3	0.23	2.84	262.1
	4	0.53	2.91	264.9
	5	0.24	2.99	327.5
S2	1	0.22	2.77	350.9
	2	0.28	3.23	321.5
	3	0.08	2.83	328.1
	4	0.13	2.74	314.3
	5	0.16	3.46	319.1
S3	1	0.33	3.22	260.7
	2	0.39	3.10	271.7
	3	0.14	3.07	278.7
	4	0.03	2.31	327.1
	5	0.05	3.19	294.0
S4	1	0.21	3.24	396.4
	2	0.09	2.93	345.4
	3	0.05	3.36	307.6
	4	0.03	3.11	356.9
	5	0.31	3.35	348.1
S5	1	0.14	3.46	433.3
	2	0.07	3.35	385.3
	3	0.08	2.96	362.9
	4	0.12	3.21	343.8
	5	0.31	3.28	278.1

Table 5.2: Collected data of simulated falls in different directions

6. Results and Discussion

From the threshold values which were determined in previous chapter, a set of tests was established to evaluate the accuracy and robustness of the system. For the consistency, all the tests were performed on only one tester because of the repetition in the gait and style of fall/non-fall activities. Volunteer S3 was chosen for these experiments. Due to the dangerous of the falls on human, I decided to limit the number of tests to 20 for both fall and non-fall activities. Volunteer S3 was asked to perform 10 true falls on the cushion and 10 non-fall activities. All the test results were recorded.

6.1 Results Assessment

The experimental results can be seen in the table 6.1. From the table, we can see that the device can correctly recognize 8/10 of non-fall activities and 6/10 of real falls. The accuracy of system is 70% with 14/20 true detections. This result is expected and reasonable.

True Positive	False Positive	True Negative	False Negative	Total
6	2	**8**	4	**14/20**

Table 6.1: Experimental Results

with

(i)True Positive: fall occurs, device detects it
(ii)False Positive: device detect falls, fall did not occur
(iii)True Negative: normal movement, device does not declare a fall
(iv)False Negative: fall occurs, device does not detect it

For the non-fall experiments, the failure in detection happened only when volunteer S3 performed the jumping with strong force on the ground (Figure 6.1). However, the important target user of this system is the elderly with the age of 65+ who really need assistance when living alone. Jumping is just an assumed activity for the testing, not usually happened with this group of people in daily life. Therefore, this failure does not effect much on the accuracy of the system.

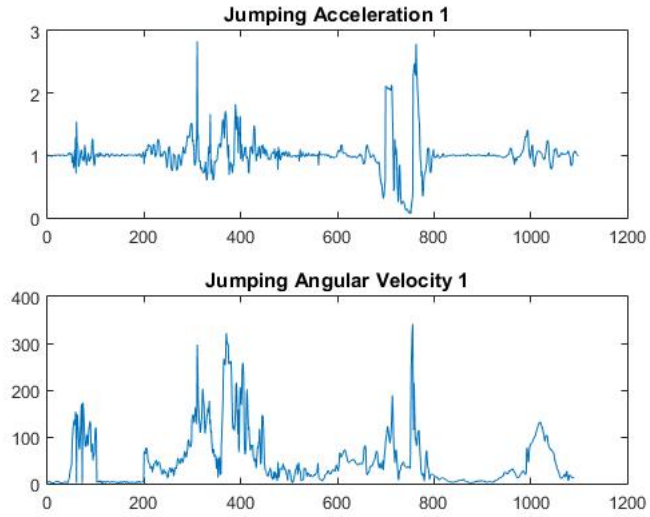


Figure 6.1: Jumping with strong force on the ground

In contrast, for the real fall experiments, there are 4/10 times that device cannot detect the falls. By examining the recorded data with visual graph, there are different reasons that cause these false negatives. The first reason is that when performing a fall simulation, the fear of vulnerability prevents the volunteer from falling as fast as a real fall. This thing causes the delay in the resulted acceleration and make the device not detect the fall. In detail, within 0.5s after the normalized acceleration falling below the LFT_{Acc}, it must reach the UFT_{Acc}. However, because of the delay, the peak happened after the fall window and make the second condition of the algorithm is not satisfied. See the figure 6.2 for this situation.

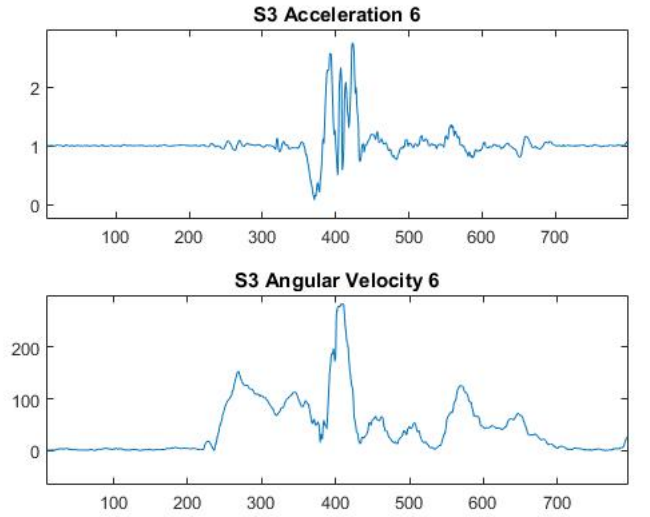


Figure 6.2: The delay of normalized acceleration causes failure in fall detection

The second reason is that when a fall happen, the normalized acceleration does not fall below the UFT_{Acc} and the first condition of the algorithm is not satisfied. This situation causes false negatives and affects the accuracy of the system. See the figure 6.3.

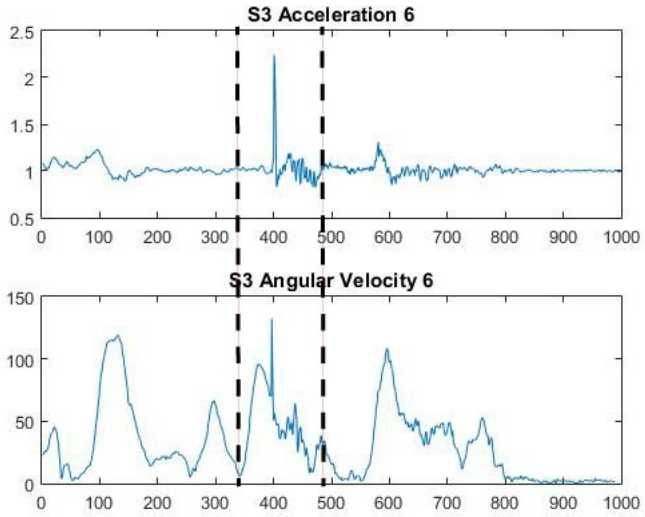


Figure 6.3: Fall happens, normalized acceleration does not fall below the UFT_{Acc}

6.2 Comparing the performance with existing works

In section 2.2, I have mentioned about two fall detection systems designed by Li *et al.* [3] and Binh *et al.* [4]. In comparison between the system in this thesis with the first system, both of them use one accelerometer and one gyroscope simultaneously to detect a fall. However, the

7. Conclusion and Future Works

7.1 Summary

Fall detection is one of the most important areas in the health-care system for the elderly. The objective of this thesis is to design and implement a wireless fall detection network prototype that can automatically detect the fall of user and send an emergency alert email to the caregivers. The system uses data from both accelerometer and gyroscope with the help of an optimized algorithm founded by Huynh *et al.* [35] to detect a fall. Looking at the testing results, the system can detect correctly up to 14/20 experiments (70%) for both fall and non-fall activities. This result is acceptable for a bachelor thesis but not for a commercial device. Therefore, to realize this system, there must be further developments to optimize the accuracy and robustness of it.

7.2 Limitations

Despite the system working well, one such limitation of this thesis is that the thresholds finding and testing experiments were performed on young healthy subjects, for the safety reasons, on a thick cushion. The collected data of falls maybe different from a real fall for both acceleration and angular velocity. However, I expect that the real fall on user would experience higher peak accelerations and potentially higher angular velocity. Therefore, I believe that the thresholds founded in this thesis can be applied to a real-world device.

Another limitation of this thesis is that when detecting a fall, an email will be sent to the caregiver for informing them about the accident. However, an email is not suited for the instant assistance requirement because the caregiver may not check for a new mail continuously. Therefore, in the near future, there will be a SIM module implemented on the system to solve this problem. Whenever detecting a fall, beside sending an email, the micro-controller also sends an SMS message to a certain issued number with the help of a SIM module. Nowadays, most of the SIM modules on the market support UART protocol for the communication with the micro-controller. Hence, when adding the SIM module, it only needs to write a small function in the main program to send a pre-defined message to the caregiver.

7.3 Future works

The system proposed in this thesis is using a simple threshold method to detect a fall. Even the result is acceptable, there are still some failures in the fall detection. The main reason for these failures is that all the thresholds are fix and not adaptive. When applied on a real user, the difference on the gait or physical condition of user's body may cause some changes in acceleration and angular velocity during a fall. Therefore, in the near future, there should be some further studies on developing an algorithm using adaptive thresholds method to recognize the falls to improve the accuracy of the system.

Beside that, within this thesis, the sensor node is designed as a small device which is attached to the chest of user. When using it, users must wear it as a shirt button or a neck pendant that may cause some uncomfortable even not disturb much on their daily life. These users also need to recharge the device everyday if they want to use it. To solve this problem, in the near future, there should be an integration of the fall detection system onto a smartphone or a smartwatch device. With this method, the fall detection system can make use of the sensors inside smart devices without the need of another external sensor. Furthermore, this integration also helps solving the power problem. Users only need to charge their smart device as what they do everyday to use the fall detection system.

Finally, because of the limitations of a bachelor thesis, the data of falls were collected only on volunteers who are my friends with the similarity in age and body's condition. The collected data may not suitable for different users, especially the elderly who really need this system. Therefore, in further developments, I would try to carry out experiments to collect data on people with different physical conditions and range of age to improve the reliability of the system.

A. C/C++ Code on the Sensor Node

```
#include <Event.h>
#include <Timer.h>
#include <AsyncPrinter.h>
#include <async_config.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncTCPbuffer.h>
#include "MPU6050.h"
#include "Wire.h"
#include "WiFiClient.h"
#include "ESP8266WiFiMulti.h"
#include "ESP.h"
#include "Time.h"
#include <CSensorSender.h>


ESP8266WiFiMulti WiFiMulti ;
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
#define OUTPUT_READABLE_ACCELGYRO
//#define SERIAL_DEBUG
#define BUZZER 13
#define CONN_INTERVAL   (1000)
#define NUMB_ELEMS_PACKET (100)
#define SAMPL_INTV      (CONN_INTERVAL/NUMB_ELEMS_PACKET)
#define dataRate 9
#define lowAcc 0.26
#define highAcc 2.75
#define highGyro 254.5

Timer t;
const uint16_t port = 8000;
```

```
String host = "192.168.137.1"; // ip or dns

CSensorSender mySender(host, port, NUMB_ELEMS_PACKET, 2);

// Use WiFiClient class to create TCP connections
AsyncClient client;
MPU6050 accelgyro;

//MPU6050 accelgyro(0x69); // <-- use for AD0 high
int16_t ax, ay, az;
int16_t gx, gy, gz;
float t_time = 0,send_time;
float rotX, rotY, rotZ,normAcc;
float gyroX, gyroY, gyroZ, normGyro;
String packet = "",packetSend = "";
int8_t i=0;
bool blinkState = false;
bool firstCond = false, secCond = false, thirdCond = false;

void Print_IMU()
{
      #ifdef SERIAL_DEBUG
      Serial.println("Accelerometer:");
      Serial.print(ax);
      Serial.print("\t");
      Serial.print(ay);
      Serial.print("\t");
      Serial.println(az);
      Serial.println("Gyroscope: ");
      Serial.print(gx);
      Serial.print("\t");
      Serial.print(gy);
      Serial.print("\t");
      Serial.println(gz);
```

```
      #endif
      Serial.print(normAcc);
      Serial.print(" ");
      Serial.println(normGyro);
}
void alert(){
      if (firstCond == true && secCond == true && thirdCond == true)
      {
         digitalWrite(BUZZER, HIGH);
      }
      firstCond = false;
      secCond = false;
      thirdCond = false;
}
void init_IMU() {
      // join I2C bus (I2Cdev library doesn't do this automatically)
      Wire.begin();
      Serial.begin(115200);
      // initialize device
      Serial.println("Initializing I2C devices...");
      accelgyro.initialize();
      // verify connection
      Serial.println("Testing device connections...");
      Serial.println(accelgyro.testConnection() ? "MPU6050 connection
          successful" : "MPU6050 connection failed");
      Serial.print("\n");
      accelgyro.setDLPFMode(1); //set the Frequency to 1kHz
      accelgyro.setRate(dataRate); //Set data Rate: Rate = (Freq)/(dataRate+1)
      Serial.print("DLPF Mode: ");
      Serial.println(accelgyro.getDLPFMode());
      Serial.print("Data rate of sensor: ");
      Serial.println(accelgyro.getRate());
      // use the code below to change accel/gyro offset values
      Serial.println("Updating internal sensor offsets...");
```

```
      accelgyro.setXGyroOffset(55);
      accelgyro.setYGyroOffset(-28);
      accelgyro.setZGyroOffset(-2);
      accelgyro.setXAccelOffset(-2581);
      accelgyro.setYAccelOffset(-3937);
      accelgyro.setZAccelOffset(1199);
}
void Init_Wifi()
{
      // Initialize Wifi_connection
      WiFiMulti.addAP("MinhKhang", "qwertyuiop");
      Serial.println();
      Serial.println();
      Serial.print("Wait for WiFi... ");
      while(WiFiMulti.run() != WL_CONNECTED) {
      Serial.print(".");
      delay(500);
      ESP.wdtFeed();
      }
      Serial.println("");
      Serial.println("WiFi connected");
      Serial.println("IP address: ");
      Serial.println(WiFi.localIP());
      digitalWrite(2, LOW);
      delay(500);
}

void setup(){
      //initialize pin mode
      pinMode(BUZZER, OUTPUT);
      Serial.begin(115200);
      Wire.begin();
      init_IMU();
      Init_Wifi();
```

```
}
void loop() {
      sensorData_t;
      ESP.wdtFeed();
      if ((millis()-t_time)>SAMPL_INTV)
      {
         //get data from sensor
         accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
         rotX = ax / 16384.0;
         rotY = ay / 16384.0;
         rotZ = az /16384.0;
         normAcc = sqrt(rotX*rotX + rotY*rotY + rotZ*rotZ);
         gyroX = gx /131.0;
         gyroY = gy /131.0;
         gyroZ = gz /131.0;
         normGyro = sqrt(gyroX*gyroX + gyroY*gyroY + gyroZ*gyroZ);
         //Start the comparator
         if (normAcc < lowAcc)
         {
            firstCond = true;
            t.after(500,alert);
         }
         if (firstCond == true && normAcc > highAcc)
         {
            secCond = true;
         }
         if (firstCond == true && normGyro > highGyro)
         {
            thirdCond =true;
         }
         t_time=millis();
         currentSensorData.normAccel = normAcc;
         currentSensorData.normGyro = normGyro;
         mySender.queueSensorData(currentSensorData);
```

```
        }
        t.update();
    }
```

B. Connection Handling Library

```
#include "CSensorSender.h"

#define DEBUG_PRINT 1

static inline void debugPrint(String inputString)
{
   #if DEBUG_PRINT
   Serial.println(inputString);
   #endif
}

void clientConnectedHandler(void* arg, AsyncClient* aClient)
{
   debugPrint("Connected");
   String sentString = ((CSensorSender*)arg)->getSentString();
   aClient->write(sentString.c_str());
   ((CSensorSender*)arg)->setIndicatorLed(LOW);
}

void clientDisconnectedHandler(void* arg, AsyncClient* aClient)
{
   debugPrint("Disconnected");
   senderState_t senderState = ((CSensorSender*)arg)->getSenderState();
   if ((senderState == SENDING) || (senderState == SENDING_ERROR)) {
      ((CSensorSender*)arg)->setSenderState(READY);
   }
   ((CSensorSender*)arg)->setIndicatorLed(HIGH);
}

void clientErrorHandler(void* arg, AsyncClient* aClient, unsigned char error)
{
   debugPrint("Error");
```

```
   senderState_t senderState = ((CSensorSender*)arg)->getSenderState();
   if (senderState == SENDING) {
   ((CSensorSender*)arg)->setSenderState(SENDING_ERROR);
   }
   ((CSensorSender*)arg)->setIndicatorLed(HIGH);
}

void CSensorSender::populateSentString()
{
   mSentString = "";
   for (int i = 0; i < mNoOfSamples; i++) {
   mSentString += String(mSensorData[i].normAccel) + "," +
       String(mSensorData[i].normGyro);
      if (i < mNoOfSamples - 1) {
         mSentString += ",";
      }
      else {
         mSentString += "\n";
      }
   }
}

// Constructor
CSensorSender::CSensorSender(String IPString, int16_t port, uint16_t
    noOfSamplesPerPacket, int ledIndicatorPin)
{
   // Load the arguments onto the attributes
   mIPString = IPString;
   mPort = port;
   mNoOfSamplesPerPacket = noOfSamplesPerPacket;
   // Allocate the exact number of elements of sensorData_t
   // this is going to be the array to store queued data
   mSensorData = new sensorData_t[mNoOfSamplesPerPacket];
   mSentString = "";
```

```
    mNoOfSamples = 0;
    // Initialize the state to READY
    mSenderState = READY;
    mLedIndicatorPin = ledIndicatorPin;
    pinMode(ledIndicatorPin, OUTPUT);
    digitalWrite(ledIndicatorPin, HIGH);
    // Initialize the ASyncClient
    mClient.onConnect(clientConnectedHandler, this);
    mClient.onError(clientErrorHandler, this);
    mClient.onDisconnect(clientDisconnectedHandler, this);
}

senderErrorCode_t CSensorSender::queueSensorData(sensorData_t&
     sensorDataInput)
{
    senderErrorCode_t returnSenderErrorCode = SENDER_ERROR_NOT_READY;
    // If there is still space left in the buffer
    if (mNoOfSamples < mNoOfSamplesPerPacket) {
    mSensorData[mNoOfSamples] = sensorDataInput;
    mNoOfSamples++;
    returnSenderErrorCode = SENDER_ERROR_OK;
}
else if (mSenderState == READY) {
    // Update the state
    mSenderState = SENDING;
    // Since this is a normal behavior
    returnSenderErrorCode = SENDER_ERROR_OK;
    populateSentString();
    mNoOfSamples = 0;
    debugPrint(mSentString);
    //Do the sending here
    mClient.connect(mIPString.c_str(), mPort);
}
else {
```

```
    returnSenderErrorCode = SENDER_ERROR_NOT_READY;
}
return returnSenderErrorCode;
}

// Destructor, clean up the allocated memory here
CSensorSender::~CSensorSender()
{
    // Deallocated allocated memory
    delete[] mSensorData;
}

String CSensorSender::getSentString()
{
    return mSentString;
}

senderState_t CSensorSender::getSenderState()
{
    return mSenderState;
}

void CSensorSender::setSenderState(senderState_t senderState)
{
    mSenderState = senderState;
}

void CSensorSender::setIndicatorLed(uint8_t ledState)
{
    digitalWrite(mLedIndicatorPin, ledState);
}
```

C. Register Map of MPU6050

InvenSense	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013

3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2 _EN	I2C_SLV2 _BYTE_SW	I2C_SLV2 _REG_DIS	I2C_SLV2 _GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3 _RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3 _EN	I2C_SLV3 _BYTE_SW	I2C_SLV3 _REG_DIS	I2C_SLV3 _GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4 _RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4 _EN	I2C_SLV4 _INT_EN	I2C_SLV4 _REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_ THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC _INT_EN	I2C _BYPASS _EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN
3A	58	INT_STATUS	R	-	-	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

Note: Register Names ending in _H and _L contain the high and low bytes, respectively, of an internal register value.

In the detailed register tables that follow, register names are in capital letters, while register values are in capital letters and italicized. For example, the ACCEL_XOUT_H register (Register 59) contains the 8 most significant bits, *ACCEL_XOUT*[15:8], of the 16-bit X-Axis accelerometer measurement, *ACCEL_XOUT*.

The reset value is 0x00 for all registers other than the registers below.

- Register 107: 0x40.
- Register 117: 0x68.

D. Python Code for Data Acquisition Program

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from drawnow import *
import socketserver

accel = []
gyro = []
plt.ion() #tell the matplotlib that we want to draw live data
def makeFig():
plt.subplot(2,1,1)
plt.plot(accel, 'r', label= 'Normalized Acceleration')
plt.legend(loc='upper left')
plt.ylim(0,5)
plt.subplot(2,1,2)
plt.plot(gyro, 'b', label = 'Normalized Angular Velocity')
plt.legend(loc='upper left')
plt.savefig('testplot.png')

class myTCPServer(socketserver.StreamRequestHandler):
def handle(self):
data = self.rfile.readline()
dataArray = data.decode().split(',')
normAcc = float(dataArray[0])
normGyro = float(dataArray[1])
accel.append(normAcc)
gyro.append(normGyro)
if len(accel) > 100:
accel.pop(0)
```

```
if len(gyro) > 100:
gyro.pop(0)
drawnow(makeFig)
#create TCP server
serv = socketserver.TCPServer(("",8000),myTCPServer)
serv.serve_forever()
```

References

[1] "How medical alert systems can help alzheimers & dementia patients." [Online]. Available: http://medicalalertsystemshq.com/guides/how-medical-alert-systems-can-help-alzheimers-dementia-patients.html

[2] [Online]. Available: https://www.mobilehelp.com/medical-alert-systems/mobilehelp-solo/

[3] Q. Li, J. A. Stankovic, M. Hanson, A. Barth, and J. Lach, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Wearable and Implantable Body Sensor Networks*, Berkeley, CA, USA, Jun. 2009.

[4] J. Tomkun and B. Nguyen, "Design of a Fall Detection and Prevention System for the Elderly," Bachelor Thesis, McMaster University Hamilton, Ontario, Canada, 2010.

[5] "MPU-6050 Accelerometer + Gyro." [Online]. Available: https://playground.arduino.cc/Main/MPU-6050

[6] *ESP8266EX Datasheet*, Espressif. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[7] "I2C Bus, Interface and Protocol." [Online]. Available: http://i2c.info/

[8] "Testing mpu6050 before embarking on project." [Online]. Available: https://wifienabler.com/tag/mpu6050/

[9] J. Y. Hwang, J. M. Kang, and H. C. Kim, "Development of novel algorithm and real-time monitoring ambulatory system using bluetooth module for fall detection in the elderly," *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEMBS 04)*, vol. 1, pp. 2204–2207, September 2004.

[10] Nation Council of Aging, "Falls prevention facts." [Online]. Available: https://www.ncoa.org/news/resources-for-reporters/get-the-facts/falls-prevention-facts/

[11] "10 shocking statistics about elderly falls," Aug. 201. [Online]. Available: https://www.shellpoint.org/blog/2012/08/13/10-shocking-statistics-about-elderly-falls/

[12] J. A. Stevens, P. S. Corso, E. A. Finkelstein, and T. R. Miller, "The costs of fatal and non-fatal falls among older adults," Oct. 2006.

[13] World Health Organiztion, "Falls." [Online]. Available: http://www.who.int/en/news-room/fact-sheets/detail/falls

[14] D. Prudham and J. G. Evans, "Factors associated with falls in the elderly: a community study." in *Age Ageing*, 1981.

[15] J. C. Morris, E. H. Rubin, E. J. Morris, and S. A. Mandel, "Senile dementia of the alzheimer's type: An important risk factor for serious falls," *Journal of Gerontology*, vol. 42, no. 4, Jul.

[16] A. J. Samuel, J. Solomon, and D. Mohan, "A critical review on the normal postural control," *Physiotherapy and occupational Journal*, vol. 8, no. 2, pp. 71–75, 2015.

[17] P. K. Pardasaney, M. D. Slavin, R. C. Wagenaar, N. K. Latham, P. Ni, and A. M. Jette, "Conceptual limitations of balance measures for community-dwelling older adults," *Physical Therapy*, vol. 93, pp. 1351–1368, 2013.

[18] J. H. Downtow, *Falls in the Elderly*. Edward Arnold, 1993, ch. 1, p. 5.

[19] A. J. Davies and R. A. Kenny, "Falls presenting to the accident and emergency department: types of presentation and risk factor profile," *Age Aging*, vol. 25, p. 3626, 1996.

[20] H. Cronnin and R. A. Kenny, "Cardiac cause of falls and their treatment," *Clinics in Geriatric Medicine*, vol. 26, no. 4, pp. 539–567.

[21] J. J. Carrero, D. J. de Jager, M. Verduijn, P. Ravani, J. D. Meester, J. G. Heaf, P. Finne, A. J. Hoitsma, J. Pascual, F. Jarraya, A. V. Reisaete, F. Collart, F. W. Dekker, and K. J. Jager, "Cardiovascular and noncardiovascular mortality among men and women starting dialysis," *Clinical Journal of the American Society of Nephrology.*

[22] MedicineNet.com, "Falls and fractures in seniors," 2004. [Online]. Available: https://www.medicinenet.com/script/main/art.asp?articlekey=7774

[23] MMWR Morb Mortal Wkly Rep, "Nonfatal fall-related traumatic brain injury among older adultscalifornia, 19961999," *Morbidity and Mortality Weekly Report*, 2003.

[24] J. A. Stevens, G. Ryan, and M. Kresnow, "Fatalities and Injuries from Falls Among Older Adults United States, from 1993 to 2003 and from 2001 to 2005," *Journal of the American Medical Association*, pp. 32–33, 2007.

[25] Consumer Safety Unit, "Home and leisure accident research," 1988.

[26] H. Gjoreski, M. Lutrek, and M. Gams, "Accelerometer placement for posture recognition and fall detection," in *Seventh International Conference on Intelligent Environments*, 2011.

[27] Q. T. Huynh, U. D. Nguyen, S. V. Tran, and B. Q. Tran, "Optimum location for sensors in fall detection," *Proceedings of the International Conference on Green and Human Information Technology*, 2013.

[28] W. Qu, F. Lin, A. Wang, and W. Xu, "Evaluation of a low-complexity fall detection algorithm on wearable sensor towards falls and fall-alike activities," *Gait Posture*, Aug. 2008.

[29] B. Alan, O. J.V., and L. Gearid, "Evaluation of a threshold-based tri-axial accelerometer," vol. 26, pp. 194–199, 01 2006.

[30] D. M. Karantonis and N. H. Lovell, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*, vol. 10, no. 1, Jan. 2006.

[31] L. H. Nair and Ragimol, "Ahrs based body orientation estimation for real time fall detection," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, March 2017, pp. 1–4.

[32] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy, "Wearable sensors for reliable fall detection," in *Proceedings of the 2005 IEEE.* Shanghai, China: Engineering in Medicine and Biology 27th Annual Conference, Sep. 2005.

[33] T. Zhang, J. Wang, L. Xu, and P. Liu, "Fall detection by wearable sensor and one-class SVM algorithm," *Intelligent Computing in Signal Processing and Pattern Recognition: International Conference on Intelligent Computing, ICIC 2006 Kunming, China*, vol. 345, August 1619, 2006.

[34] P. Jayachandran, T. F. Abdelzaher, J. A. Stankovic, and R. K. Ganti, "Satire: a sofware architecture for smart attire," *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys 06)*, pp. 110–123, June 2016.

[35] Q. T. Huynh, U. D. Nguyen, L. B. Irazabal, N. Ghassemian, and B. Q. Tran, "Optimization of an accelerometer and gyroscope-based fall detection algorithm," *Journal of Sensor*, no. 452078, 2015.

[36] A. Ibrahim and M. I. Younis, "Simple fall criteria for mems sensors: Data analysis and sensor concept," *Sensor*, vol. 14, no. 7, p. 1214912173, 2014.

[37] M. Popescu, B. Hotrabhavananda, M. Moore, and M. Skubic, "V ampir- an automatic fall detection system u sing a vertical pir sensor array," in *Proceedings of the 6th International Conference on Pervasive Computing Technologies for Healthcare*, San Diago, California, USA, May 2012, pp. 163–166.

[38] J. Rowberg, "Mpu6050 library." [Online]. Available: https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050