

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

# **DESIGN AND IMPLEMENTATION 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

# **Design and Implementation of a Wireless Fall Detection Network prototype using MEMS sensors**

Approved by

---

Prof. Dr.-Ing Kira Kastell, First Supervisor

---

Dipl.-Ing Mohammad Reza Mansooji

## **Disclaimer**

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

©2018 by Ngoc Khang Minh, Ho. All right reserved.

## 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 combines the data from 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

<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>2</b>
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
<b>2 Literature Review</b>	<b>8</b>
2.1 Existing Commercial Devices . . . . .	8
2.2 Existing Products from Other Research Groups . . . . .	9
<b>3 Statement of Problem and Methodology</b>	<b>11</b>
3.1 Problem . . . . .	11
3.2 Methodology and Algorithm . . . . .	11
<b>4 System overview</b>	<b>15</b>
4.1 Hardware Components . . . . .	15
4.1.1 Sensor . . . . .	15
4.1.2 Microcontroller . . . . .	17
4.1.3 Wireless Module . . . . .	18
4.2 Software . . . . .	19

4.2.1	Embeddded Software on Sensor Node . . . . .	19
4.2.2	Data Acquisition Software . . . . .	21
4.2.3	Data Analysis Software . . . . .	22
4.3	System Intergration . . . . .	22
<b>5</b>	<b>Experimental and Procedure</b>	<b>24</b>
5.1	Data Collection . . . . .	24
5.2	Data Analysis . . . . .	24
5.3	Experiment on Real Subjects to Evaluate the Performance of The Network	24
<b>6</b>	<b>Results and Discussion</b>	<b>25</b>
6.1	Results Assessment . . . . .	25
6.2	Comparing the performance with existing works . . . . .	25
<b>7</b>	<b>Conclusion and Future Works</b>	<b>26</b>
7.1	Summary . . . . .	26
7.2	Limitations . . . . .	26
7.3	Future works . . . . .	26
<b>APPENDICES</b>		<b>26</b>
<b>A</b>	<b>C/C++ Code on the Sensor Node</b>	<b>27</b>
<b>B</b>	<b>Connection Handling Library</b>	<b>33</b>
<b>C</b>	<b>Register Map of MPU6050</b>	<b>37</b>
<b>D</b>	<b>Python Code for Data Acquisition Program</b>	<b>40</b>
<b>References</b>		<b>42</b>

# List of Tables

4.1	Full-scale Range Table of Accelerometer . . . . .	16
4.2	Sensitivity Scale Factor Table of Accelerometer . . . . .	16
4.3	Full-scale Range Table of Gyroscope . . . . .	16
4.4	Sensitivity Scale Factor Table of Gyroscope . . . . .	17
4.5	Digital Low Pass Filter Configuration of the MPU-6050 Sensor . . . . .	20

# List of Figures

2.1	Medical Guardian Fall Alert System . . . . .	8
2.2	myHalo Medical Alert System from MobileHelp <sup>TM</sup> . . . . .	9
2.3	TEMPO 3.0 sensor node . . . . .	10
2.4	The final design of device from group research of Binh Nguyen and Jonathan Tomkun . . . . .	10
3.1	Fall detection algorithm . . . . .	14
4.1	MPU-6050 by Invensense [1] . . . . .	15
4.2	ESP8266 D1 Mini Board with an ESP8266EX MCU inside . . . . .	17
4.3	Functional Block Diagram of ESP8266EX MCU . . . . .	18
4.4	Schematic of wireless communication system . . . . .	19
4.5	Graphic User Interface of Data Acquisition Program . . . . .	21
4.6	Data plotting with MATLAB . . . . .	22
4.7	Printed Circuit Board Design . . . . .	23
4.8	Sensor Node . . . . .	23

# **Acronyms**

**ADC** analog-to-digital. 15

**IMU** Inertia Measurement Unit. 15

**IoT** Internet-of-Things. 17, 18

**MCU** Micro-Controller Unit. 17, 18

**NCOA** the National Council of Aging. 4

**SSF** Sensitivity Scale Factor. 15, 16

**U.S.** United State of America. 5, 8

**WHO** World Health Organization. 2, 5

# 1. Introduction

## 1.1 Introduction to Fall

Nowadays, fall is becoming a dangerous issue which mostly causes the injuries and even disability or fatal death on human, especially the elderly. According to Hwang *et.al.* [2], in the United State, 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 [3]. It is also reported that one out every 200 falls results in a hip fracture on people with age among 65 and 69 and increase to one out of 10 for those aged 85 and more [4]. 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 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 [5].

## 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 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 [6].

## 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 common cause of fall on human, especially the elderly. Due to the research of Prudheim *et al.* in 1981, fallers have in general been found to have a higher prevalence of cognitive impairment than non-fallers [7]. It is also reported that human with dementia are approximately 3 times more likely to fall than non-demented [8]. 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 on 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 [9][10]. The impairment of postural control is indicated as one of significant causes of the fall during any activities 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. 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 [11].

#### ***Cardiovascular problem***

Cardiovascular disorders are responsible for approximately 77% of patients with unexplained or recurrent falls and falls associated with unexplained loss of consciousness [12]. There is a fact that fallers with cardiovascular disorders have a greater mortality than those with non-cardiovascular or unknown causes [13].

### **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 human, 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 human because of creating glares and distorting the way object look.

#### ***Bad staircase design***

Bad staircase design is also another extrinsic risk that cause the fall on human. Too high or too low rise of staircases has 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 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 human 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 human 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 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 [3]. 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 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 woman than men and increase in frequency with increasing age [11] [14]. 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 [14].

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 [11]. In California from 1996 to 1999, 71% of fall-related head injuries occurred in adults aged 65 and more [15]. The head trauma from 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 patient. According to data from the WHO, the fall mortality 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 [16]. 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. For 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 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 prevent patients from living alone and start depending on the long-term nursing care. This consequence directly leads to increased dependency, which require more time and financial resources from both government and family.

Falls are also the main reason for fallers to be admitted to hospital as an emergency case. Due to the information provided by 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 [17].

Finally, falls are also a economic burden of every countries 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 [3]. These costs are spent for hospitalization, medical, pharmaceutical, nursing home and other costs which related to the medical treatments for fall patients.

## **1.5 Challenges in Detecting Falls**

As mentioned in 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-like activities such as sitting, jumping or running, which also result in high acceleration. Therefore, it is required to collect a great amount of data from different subjects; and then, there should be some analysis 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 body, since it is an accidental, dangerous action which can happen randomly at any time. Therefore, it is a overwhelming task for researcher to built a dataset for their system.

## 1.6 Objectives

From above challenges, it is necessary to study and design a wireless sensor network which can automatically detect the fall of patient and send alarm to the caregiver instantly to reduce their time of arrival, and hence reduce mortality rate. The sensor node is portable and attached on the chest of user. 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 Wi-Fi module called ESP8266 from ESPRESSIF®company and send data directly to a monitoring software on the computer. Upon detecting a fall, sensor node will wait for the safe signal from patient within 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 help 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 fall and 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 group is included. Chapter 3 describes the problem of fall detection and 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 aspects. Chapter 5 and 6 describe the data collection and analysis procedure as well as result assessment of the experiments on real human body. Finally, chapter 7 gives the conclusion for this thesis, limitation of current solution and recommendation 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 current society where there are more and more elderly people fall every day. These products are used widely by many people around the world and have positive effects on health-care systems of these 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 have no disturbance on user's daily activity. Medical Guardian® provides a home system including one base unit connecting to the monitoring station through landline and one auto-alert pendant on 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 actual need help, do not press the button. Then, the system will connect to operators at the monitoring center, they will ask if patient need help or not. If the patient cannot speak because of unconsciousness, the operators will automatically send emergency services to 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

Another device on the market is myHalo<sup>TM</sup> from MobileHelp<sup>TM</sup>. This device has more advantages compared to Medical Guardian one because of full body monitoring functions. It can detect a fall, monitoring heart rate, skin temperature, sleep/wake pattern, etc. and send alert signal to the authorized contacts. This product supports the medical alert outside the home powered by nationwide cellular network and do not require landline phones. It also provides precise location of patients when they go outside. 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<sup>TM</sup>

## 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 have used TEMPO (Technology-Enabled Medical Precision Observation) 3.0 sensor node for their project [18]. This sensor node contains one tri-axial accelerometer and one tri-axial gyroscope and 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 computational requirements and

real-time response. The authors stated that their method has difficulties in differentiating jumping into bed and falling against wall with a seated posture.

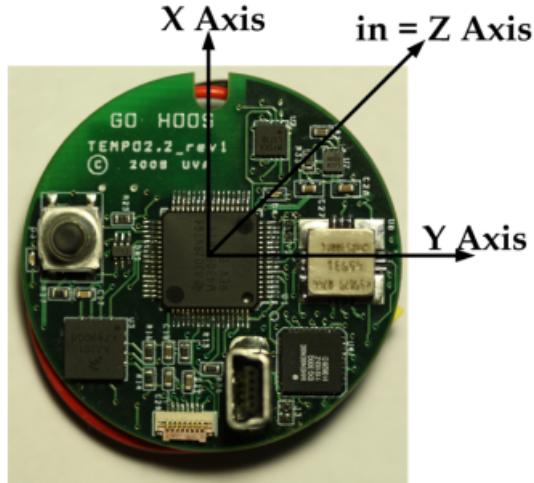


Figure 2.3: TEMPO 3.0 sensor node

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 [19]. 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 tactile alert options for people with hearing or seeing disabilities. Regarding the performance of system, some actions have not been distinguished by one of four proposed algorithm to be falls or non-falls.



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

# **3. Statement of Problem and Methodology**

## **3.1 Problem**

Nowadays, there have been various research groups studying on the area of fall detection on human. The majority of them focus on designing a new algorithm to successfully distinguish between fall and fall-like activities. From the first time, most of research groups used algorithms mainly based on the data from only one accelerometer such as [20][21][22]. However, focusing only on the data from the accelerometer can result in many false positives as other activities such as sitting, running and jumping which may also cause large peak accelerations. After that, there are other algorithms rely on the detection of body orientation after the fall such as [23][24]. 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 have nominated such complex algorithms that used Support Vector Machine (SVM)[25] and Markov model [26] to detect the fall. These techniques require huge amount of computational resources which cause the delay on system 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 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 falling detection system without the need of complex computations.

This algorithm is nominated by Quoc T. Huynh *et al.* [27], 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  $\mathbf{Acc}_{norm}$  containing both static and dynamic acceleration components in different directions. This parameter is calculated with the formula below

$$\mathbf{Acc}_{norm} = \sqrt{(A_x)^2 + (A_y)^2 + (A_z)^2} \quad (3.1)$$

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 direction. This parameter is calculated with the formula below

$$\boldsymbol{\omega}_{norm} = \sqrt{(\omega_x)^2 + (\omega_y)^2 + (\omega_z)^2} \quad (3.2)$$

where  $\omega_x$ ,  $\omega_y$  and  $\omega_z$  represent the angular velocity of three directions  $x$ ,  $y$ ,  $z$ .

When stay in stationary condition, the body of user gives the acceleration magnitude of a constant at  $+1g$  and angular velocity at  $0^\circ/s$ . When user falls, there is a rapid change in body's acceleration while 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 follows 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  $\boldsymbol{\omega}$  resultant of each recorded activity are referred to as the signal upper peak values ( $UPVs$ ). 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  $\mathbf{Acc}$  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 [28][29]. 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. [27] 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 or sitting up. Figure 3.1 summarizes the main steps of this algorithm.

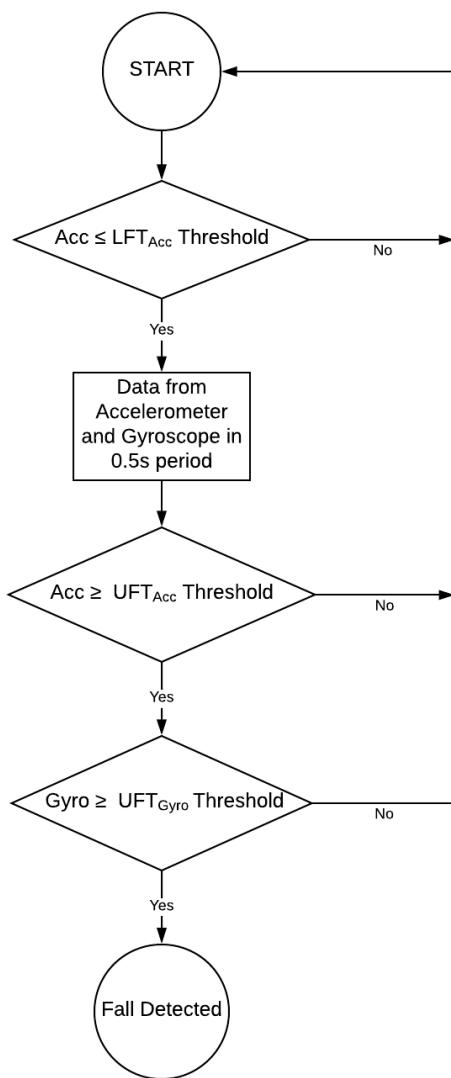


Figure 3.1: Fall detection algorithm

# 4. System overview

## 4.1 Hardware Components

### 4.1.1 Sensor

Since the algorithm requires both information about acceleration and angular velocity, an Inertia Measurement Unit (IMU) named MPU6050 from Invensense®, which contains one accelerometer and one gyroscope, has been chosen for its convenience. This sensor can also accept inputs from an external 3-axis compass via I<sup>2</sup>C sensor bus to provide a complete 9-axis MotionFusion<sup>TM</sup> output.



Figure 4.1: MPU-6050 by Invensense [1]

The accelerometer inside 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 (ADC) supports simultaneous sampling of the accelerometer while requiring no external multiplexer. This accelerometer normally operates at a very small current of  $500\ \mu A$  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 2.1 shows the different full-scales of the accelerometer and table 2.2 shows the SSF for each scale.

Scale	TYP	UNIT
AFS_SEL=0	$\pm 2$	g
AFS_SEL=1	$\pm 4$	g
AFS_SEL=2	$\pm 8$	g
AFS_SEL=3	$\pm 16$	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 ( $\pm 250^\circ/s$ ,  $\pm 500^\circ/s$ ,  $\pm 1000^\circ/s$  and  $\pm 2000^\circ/s$ ). This sensor has an external sync signal which supports image, video capturing and GPS synchronization. Its normal operating current is  $500\ \mu A$  which enables the reduction on power consumption. The raw output of the gyroscope is displayed in form of  $LSB/(\circ/s)$ . Divided with an appropriate Sensitivity Scale Factor (SSF), we can obtain the angular velocity with unit of  $(\circ/s)$ . Each full scale has its own SFF. Table 2.3 shows different full-scale of the gyroscope and table 2.4 shows the SSF for each scale.

Scale	TYP	UNIT
FS_SEL=0	$\pm 250$	$^\circ/s$
FS_SEL=1	$\pm 500$	$^\circ/s$
FS_SEL=2	$\pm 1000$	$^\circ/s$
FS_SEL=3	$\pm 2000$	$^\circ/s$

Table 4.3: Full-scale Range Table of Gyroscope

Scale	Sensitivity Scale Factor	UNIT
FS_SEL=0	131	LSB/(°/s)
FS_SEL=1	65.5	LSB/(°/s)
FS_SEL=2	32.8	LSB/(°/s)
FS_SEL=3	16.4	LSB/(°/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<sup>2</sup>C, ADC, PWM etc. to connect with external peripheral devices. The figure 4.3 shows the functional block diagram of ESP8266EX.



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

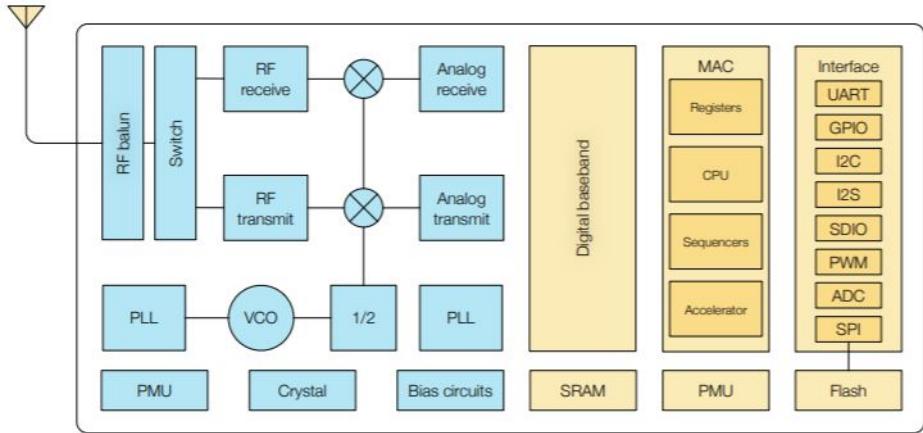


Figure 4.3: Functional Block Diagram of ESP8266EX MCU

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 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 only  $20\text{ }\mu\text{A}$  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 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 the requests from 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.

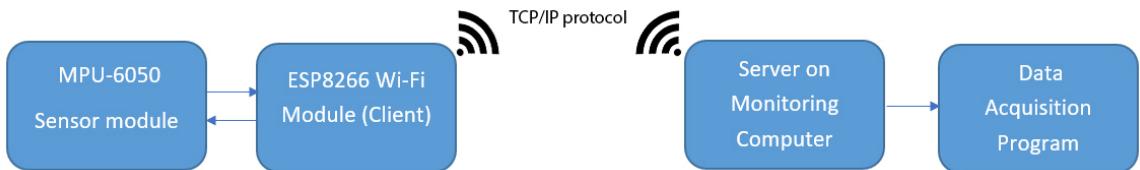


Figure 4.4: Schematic of wireless communication system

## 4.2 Software

### 4.2.1 Embedded Software on 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<sup>2</sup>C protocol with the help of MPU6050 library provided by Invensense®, 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 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 low down the internal sampling rate of two sensors. See the table 4.5 for the digital low pass filter configuration.

DLPF_CFG	Accelerometer		Gyroscope		Sample Rate
	Bandwidth	Delay	Bandwidth	Delay	
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 offset are calculated by another program included in the MPU6050 library and set into appropriate sensor's registers with 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 keep brighting until receiving a successful connection. Once the connection is established, the sensor node turns off the indicator led for a while and then start 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 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 start blinking the super bright red led simultaneously. That red led help the caregivers to find out 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 the 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 a always-run server on the host computer and waits for requests from 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 file 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.

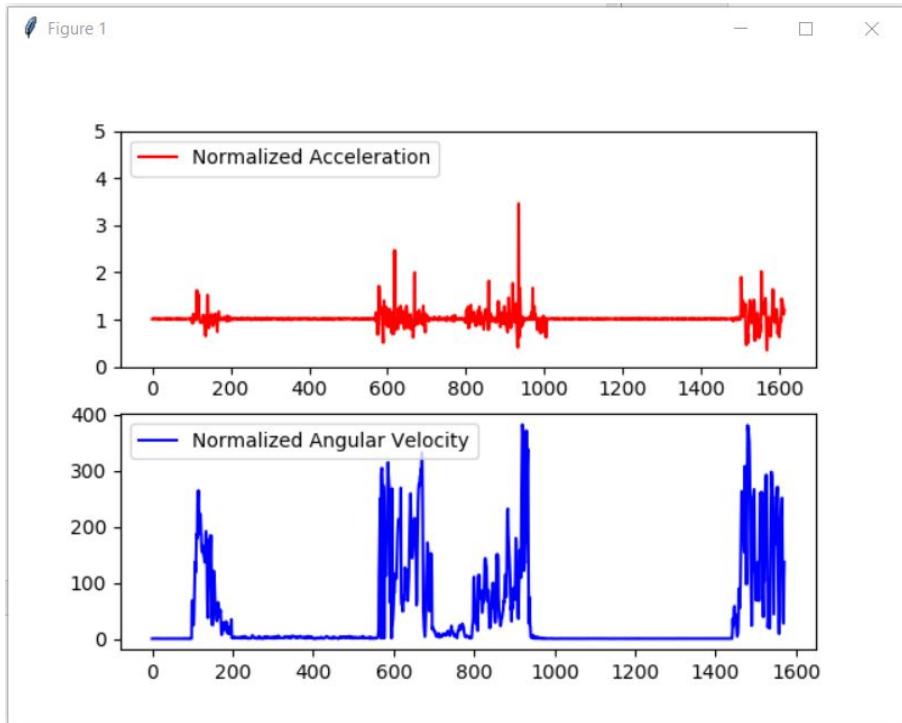


Figure 4.5: Graphic User Interface of Data Acquisition Program

### 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 plotting function supported by MATLAB ®. While working with data, we can use data cursor tool provided by MATLAB to figure out the value of upper and lower peaks (Figure 4.6).

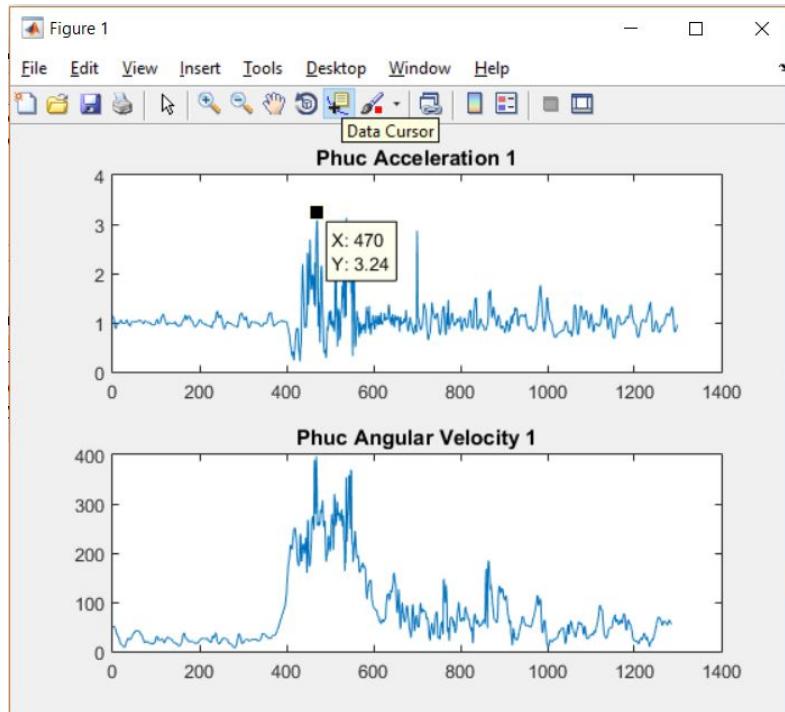


Figure 4.6: Data plotting with MATLAB

## 4.3 System Intergration

Sensor and micro-controller are connected together on the Printed Circuit Board (PCB) which is designed with EAGLE® software from the Autodesk® company. 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 user to re-charge the battery with USB cable. It is convenient for user to charge the device everyday with micro-USB cable which can be found everywhere. Figure 4.8 is the final design of the sensor node.

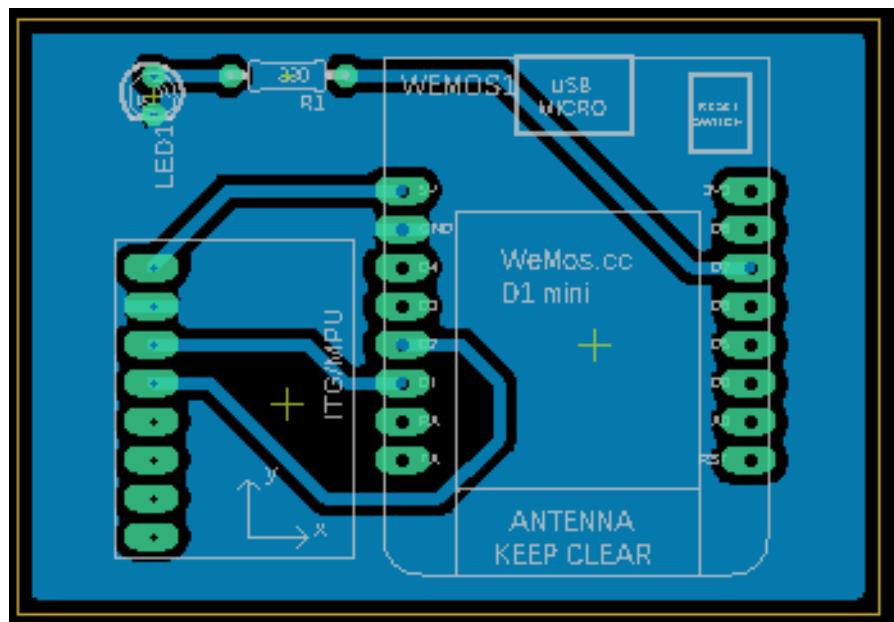


Figure 4.7: Printed Circuit Board Design

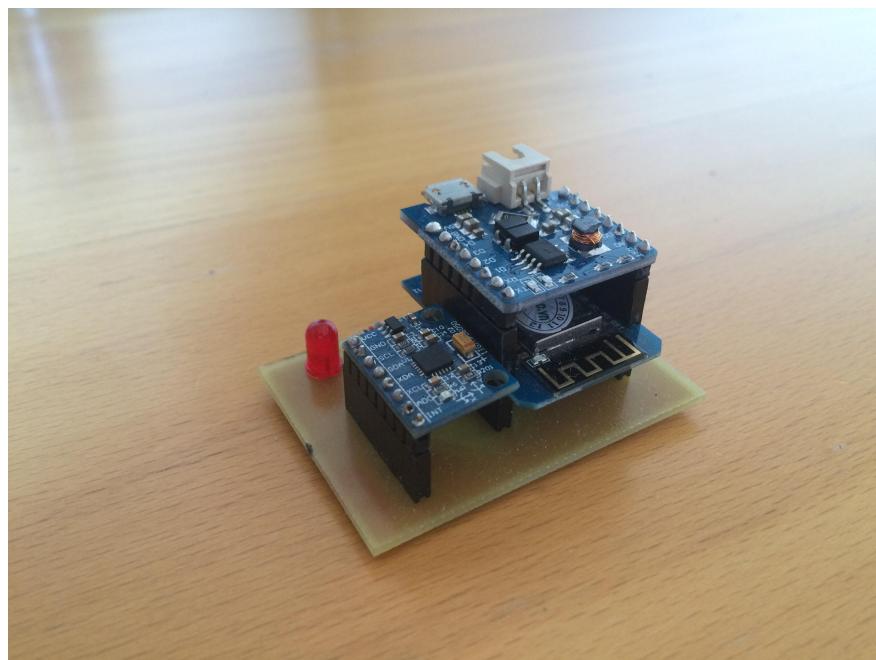


Figure 4.8: Sensor Node

## **5. Experimental and Procedure**

### **5.1 Data Collection**

### **5.2 Data Analysis**

### **5.3 Experiment on Real Subjects to Evaluate the Performance of The Network**

## **6. Results and Discussion**

### **6.1 Results Assessment**

### **6.2 Comparing the performance with existing works**

## **7. Conclusion and Future Works**

### **7.1 Summary**

### **7.2 Limitations**

### **7.3 Future works**

# 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 ADO 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
}

```

```

#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

	<b>MPU-6000/MPU-6050 Register Map and Descriptions</b>	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_BYT_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_BYT_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_BYT_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_BYT_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											

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

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]

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

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CT_RL	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_[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] “MPU-6050 Accelerometer + Gyro.” [Online]. Available: <https://playground.arduino.cc/Main/MPU-6050>
- [2] 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.
- [3] Nation Council of Aging, “Falls prevention facts.” [Online]. Available: <https://www.ncoa.org/news/resources-for-reporters/get-the-facts/falls-prevention-facts/>
- [4] “10 shocking statistics about elderly falls,” Aug. 201. [Online]. Available: <https://www.shellpoint.org/blog/2012/08/13/10-shocking-statistics-about-elderly-falls/>
- [5] E. A. F. T. R. M. J. A. Stevens, P. S. Corso, “The costs of fatal and non-fatal falls among older adults,” Oct. 2006.
- [6] World Health Organiztion, “Falls.” [Online]. Available: <http://www.who.int/en/news-room/fact-sheets/detail/falls>
- [7] E. J. Prudham D, “Factors associated with falls in the elderly: a community study.” in *Age Ageing*, 1981.
- [8] J. C. M. E. H. R. E. J. M. 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.
- [9] D. M. Asir John Samuel, John Solomon, “A critical review on the normal postural control,” *Physiotherapy and occupational Journal*, vol. 8, no. 2, pp. 71–75, 2015.

- [10] R. C. W. N. K. L. P. N. A. M. J. Poonam K. Pardasaney, Mary D. Slavin, “Conceptual limitations of balance measures for community-dwelling older adults,” *Physical Therapy*, vol. 93, pp. 1351–1368, 2013.
- [11] J. H. Downtow, *Falls in the Elderly*. Edward Arnold, 1993, ch. 1, p. 5.
- [12] K. R. Davies AJ, “Falls presenting to the accident and emergency department: types of presentation and risk factor profile,” *Age Aging*, vol. 25, p. 3626, 1996.
- [13] M. V. P. R. J. D. M. J. G. H. P. F. A. J. H. J. P. F. J. A. V. R. F. C. F. W. D. K. J. J. Juan J. Carrero, Dinanda J. de Jager, “Cardiovascular and noncardiovascular mortality among men and women starting dialysis,” *Clinical Journal of the American Society of Nephrology*.
- [14] MedicineNet.com, “Falls and fractures in seniors,” 2004. [Online]. Available: <https://www.medicinenet.com/script/main/art.asp?articlekey=7774>
- [15] M. M. M. W. Rep, “Nonfatal fall-related traumatic brain injury among older adults California, 1996–1999,” *Morbidity and Mortality Weekly Report*, 2003.
- [16] 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.
- [17] Consumer Safety Unit, “Home and leisure accident research,” 1988.
- [18] 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.
- [19] J. Tomkun and B. Nguyen, “Design of a Fall Detection and Prevention System for the Elderly,” Bachelor Thesis, McMaster University Hamilton, Ontario, Canada, 2010.
- [20] A. W. W. X. Weihao Qu, Feng Lin, “Evaluation of a low-complexity fall detection algorithm on wearable sensor towards falls and fall-alike activities,” *Gait Posture*, Aug. 2008.
- [21] A. Bourke, J. O’Brien, and G. Laighin, “Evaluation of a threshold-based tri-axial accelerometer,” vol. 26, pp. 194–199, 01 2006.

- [22] N. H. L. Dean M. Karantonis, "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.
- [23] 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.
- [24] D. C. J. L. R. B. Jay Chen, Karric Kwong, "Wearable sensors for reliable fall detection," in *Proceedings of the 2005 IEEE*. Shanghai, China: Engineering in Medicine and Biology 27th Annual Conference, Sep. 2005.
- [25] 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.
- [26] P. Jayachandran, T. F. Abdelzaher, J. A. Stankovic, and R. K. Ganti, "Satire: a software architecture for smart attire," *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys 06)*, pp. 110–123, June 2016.
- [27] 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.
- [28] M. I. Y. Alwathiqbella Ibrahim, "Simple fall criteria for mems sensors: Data analysis and sensor concept," *Sensor*, vol. 14, no. 7, p. 1214912173, 2014.
- [29] M. M. M. S. Mihail Popescu, Benjapon Hotrabhavananda, "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.