

**MULTI PLATFORM QUATERNION BASED
LEG GESTURE RECOGNITION**

LING WEN WEN

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Hons.) Electrical & Electronics Engineering**

**Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2014

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : Ling Wen Wen

ID No. : 0904618

Date : 1st May 2014

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**MULTI PLATFORM QUATERNION BASED LEG GESTURE RECOGNITION**” was prepared by **LING WEN WEN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Electrical & Electronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature :  _____

Supervisor : Prof. Chong Poh Kit _____

Date : 02 May 2014 _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2014, Ling Wen Wen. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Prof. Chong Poh Kit for his invaluable advice, guidance and his enormous patience throughout the development of the research project.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement in progression of my final year project. Their persistent love and courage enable me to overcome any circumstances encountered.

MULTI PLATFORM QUATERNION BASED LEG GESTURE RECOGNITION

ABSTRACT

The tracking of human motion has always been a subject of interest in many disciplines such as sports, health, computer, and virtual applications. With the recent advancement in Micro Electro-Mechanical Systems (MEMS), the size of inertial and magnetic sensors has been reduced significantly. This project is motivated by the combination of public awareness on daily activities that promotes a healthy lifestyle and the increasing number of smart phone users. This project incorporates a system based upon the smart phone, the inertial and magnetic sensors which are used to recognize different leg gesture, track users foot step and distance. An Inertial Measurement Unit (IMU) which consists of a tri-axis accelerometer and gyroscope is attached to a microcontroller and the data is transmitted to an Android mobile device via Bluetooth connection to allow real time tracking functions. A quaternion based algorithm is implemented in the sensor device to monitor the orientation with respect to the sensor's position in the user's body. The results are studied and evaluated to confirm the accuracy of the method used in this project.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF SYMBOLS / ABBREVIATIONS	xvi

CHAPTER

1	INTRODUCTION	1
1.1	Background	1
1.2	Problem Statement and Motivation	2
1.3	Objectives	4
1.4	Outline of Project	4
2	LITERATURE REVIEW	6
2.1	Existing Products/Researches	6
2.1.1	Tri-Axis Inertial/Magnetic Package	6
2.1.2	MTx 9 DOF IMU Sensors	7
2.1.3	Jawbone Up	8
2.1.4	FitBit Flex	9
2.1.5	Nike FuelBand	10
2.1.6	Basis	11
2.2	Conclusion	12

3	METHODOLOGY	14
3.1	Overview	14
3.2	Sensors	17
3.2.1	Accelerometer and Gyroscope	17
3.2.2	Initial Results	19
3.2.3	True Acceleration Accelerometer Model	22
3.2.4	Quaternions	23
4	PROJECT IMPLEMENTATION	25
4.1	Device Setup	25
4.2	Overall Detection Routine and Step Distance Calculation	28
4.2.1	Leg Gesture Detection Routine	28
4.2.2	Step Distance Calculation	29
4.3	Android App Development	30
4.3.1	Application Publication	33
5	RESULTS AND DISCUSSION	34
5.1	Motion Differentiation	34
5.1.1	Static Motion / No Movement	34
5.1.2	Forward Step Detection	35
5.1.3	Side Step Detection	38
5.1.4	Combination Differentiation	42
5.2	Distance Calculation	54
5.2.1	Forward Step Distance	54
5.2.2	Forward Step Distance Threshold and Calculation	57
5.2.3	Side Step Distance	60
5.2.4	Side Step Distance Threshold and Calculation	62
5.3	Empirical Test and Comparison with Commercial Product	66
5.3.1	Forward Step Distance	66
5.3.2	Side Step Right Distance	69
5.3.3	Side Step Left Distance	72
5.3.4	Summary of Results and Comparison	74

6	CONCLUSION AND RECOMMENDATIONS	75
6.1	Conclusion	75
6.2	Future Implementation	75
6.2.1	Recognition of Other Leg Gesture Motion	75
6.2.2	Bluetooth Low Energy	76
6.2.3	Mobile Application Platform	76
	REFERENCES	77

LIST OF TABLES

TABLE	TITLE	PAGE
2.1	Comparison of Existing Products	12
5.1	Forward Step Result of 4 Different Users.	45
5.2	Side Step Result of 4 Different Users.	45
5.3	Result of Threshold Percentage and Error Produced.	58
5.4	Result of Threshold Percentage and Error Produced.	59
5.5	Result of Threshold Percentage and Error Produced on Side Step.	63
5.6	Result of Threshold Percentage and Error Produced for Side Step.	64
5.7	Result Comparison of Sensor Device against FitBit Flex for Forward Step Distance.	66
5.8	Result Comparison of Sensor Device against FitBit Flex for Side Step Right Distance.	69
5.9	Result Comparison of Sensor Device against FitBit Flex for Side Step Right Distance.	72

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Block diagram of sensor package.	6
2.2	MTx Sensor.	8
2.3	Jawbone Up	9
3.1	Block Diagram of System.	14
3.2	Arduino Uno R3	16
3.3	Comparison of Wireless Communication Technologies.	16
3.4	Concept of Accelerometer.	17
3.5	Effect of Accelerometer by Gravity.	18
3.6	Gyroscope with Vector R	18
3.7	Connection between Arduino Uno R3 and MPU-6050.	20
3.8	Raw Sensor Values of MPU-6050.	21
3.9	Corrected Raw Sensor Values of MPU-6050	21
3.10	True Acceleration Model.	22
3.11	Relationship of the three imaginary numbers of quaternion.	23
3.12	Before and After turning the box.	24
4.1	Sensors on prototyping board (left) & Arduino Microcontroller (right).	25
4.2	Device in black sport band and portable battery bank.	26

4.3	Device attached to user's leg.	27
4.4	Using Teraterm to log in results.	27
4.5	Flowchart of leg gesture detection routine.	28
4.6	Flowchart of Step Distance Calculation.	29
4.7	Data collection using SENA BTerm.	30
4.8	Eclipse with Android Plugin.	31
4.9	Application request Bluetooth.	4.10
	Main Page	32
4.11	Manually connect device.	4.12
	Data Collection.	32
4.13	Application on Google Play Store.	33
5.1	Static movement result.	35
5.2	Result collected for 7 forward steps.	36
5.3	Result of 7 forward steps with Moving Average Filter.	37
5.4	Result of 7 forward steps with Moving Average Filter.	38
5.5	Results of 8 side steps.	38
5.6	Result of Side Step Count.	39
5.7	Result of 8 Side Steps with False Positive Error.	40
5.8	Result of Side Step Count with False Positive Error Forward Count.	40
5.9	Result of Side Step Data with False Positive Error removed.	41
5.10	Result of Side Step Count with False Positive Error removed.	42
5.11	Results of combined gestures.	43
5.12	Step Count and Side Step Count of combined gestures.	43
5.13	Result of User 1 Forward Steps.	46

5.14	Result of User 1 Step Bit and Side Step Bit.	46
5.15	Result of User 1 Total Forward Step Count.	46
5.16	Result of User 2 Forward Steps.	47
5.17	Result of User 2 Step Bit and Side Step Bit.	47
5.18	Result of User 2 Total Forward Step Count.	47
5.19	Result of User 3 Forward Steps.	48
5.20	Result of User 2 Step Bit and Side Step Bit.	48
5.21	Result of User 3 Total Forward Step Count.	48
5.22	Result of User 4 Forward Steps.	49
5.23	Result of User 4 Step Bit and Side Step Bit.	49
5.24	Result of User 4 Total Forward Step Count.	49
5.25	Result of User 1 Side Steps.	50
5.26	Result of User 1 Step Bit and Side Step Bit.	50
5.27	Result of User 1 Total Side Step Count.	50
5.28	Result of User 2 Side Steps.	51
5.29	Result of User 2 Step Bit and Side Step Bit.	51
5.30	Result of User 2 Total Side Step Count.	51
5.31	Result of User 3 Side Steps.	52
5.32	Result of User 3 Step Bit and Side Step Bit.	52
5.33	Result of User 3 Total Side Step Count.	52
5.34	Result of User 4 Side Steps.	53
5.35	Result of User 4 Step Bit and Side Step Bit.	53
5.36	Result of User 4 Total Side Step Count.	53
5.37	Accelerometer Data of 3 Forward Steps.	55
5.38	Magnitude of Accelerometer X after conversion.	56

5.39	Graph of Magnitude X and Step Bit.	57
5.40	Graph of Error Percentage against Threshold Percentage.	59
5.41	Graph of Error Percentage against Threshold Percentage.	60
5.42	Accelerometer Data of 3 Side Steps.	61
5.43	Magnitude of Accelerometer Z after conversion.	61
5.44	Graph of Magnitude Z and Side Step Bit	62
5.45	Graph of Error Percentage against Threshold Percentage for Side Steps.	63
5.46	Graph of Error Percentage against Threshold Percentage for Side Steps.	64
5.47	Mechanism of User Side Step Left and Right	65
5.48	Forward Step Distance of User 1.	67
5.49	Forward Step Distance of User 2.	67
5.50	Forward Step Distance of User 1 before using Fitbit Flex.	68
5.51	Forward Step Distance of User 1 after using Fitbit Flex.	68
5.52	Forward Step Distance of User 2 before Using Fitbit Flex.	68
5.53	Forward Step Distance of User 2 after Using Fitbit Flex.	69
5.54	Side Step Right Distance of User 1	70
5.55	Side Step Right Distance of User 2.	70
5.56	Side Step Distance of User 1 before using FitBit Flex	71
5.57	Side Step Right Distance of User 1 after using FitBit Flex.	71

5.58	Side Step Right Distance of User 2 before using FitBit Flex	71
5.59	Side Step Right Distance of User 2 after using FitBit Flex.	72
5.60	Side Step Left Distance of User 1	73
5.61	Side Step Left Distance of User 2	73

LIST OF SYMBOLS / ABBREVIATIONS

AHRS	Altitude and Heading Reference System
MEMS	Micro Electro-Mechanical System
IMU	Inertial Measurement Unit
TWI	Two Wire Interface
I2C	Inter-Integrated Circuit
SDA	Serial Data Line
SCL	Serial Clock Line
DOF	Degrees of Freedom

CHAPTER 1

INTRODUCTION

1.1 Background

Human motion studies, also known as kinesiology have always been a very fascinating subject of study. The origin of the subject is derived from exercise physiology and dates back to 384 B.C. as found in Aristotle's book 'De Motu Animalium' – On the Movement of Animals. The book explains the physiological differences of imagining an action to be performed and actually doing it (Roternberg, 2006). Claudius Galenus (126-199 A.D.), who is known as the founder of experimental physiology, is the first human to use experiments to probe the functions of body. In between 776 B.C. to 393 A.D., ancient Greek physician controlled training methods and diet plans of Olympic competitors with the work of Galen, and many of the principles are also still used today. In the modern era, representatives of the field include Edward Hitchcock Jr. (1828-1911), an Amherst College professor who had co-authored 1860 texts on exercise physiology (Wikipedia, 2013).

Human motion tracking is generally applied in multiple disciplinary fields which require the capturing of human body posture and movement. In scientific field, scientist seeks recognising mechanisms that may be used to translate muscle contraction into useful function and thus enabling the creation of robots with human-like nature, and are able to perform human-like activities. Researchers have considerable interests in understanding the relationship of the proper posture and motion in gait analysis. In sports science, athletes seek to use human motion analysis as a tool to improve athletic performances. In the virtual world, graphic designers

create human models and animate movie characters through the study of human motion. This method of human motion analysis is known as rotoscoping which originates from Disney Studios (Rotenberg, 2006).

The technologies used to track human motion are generally divided into two approaches; namely the visual and non-visual tracking systems. Visual tracking systems are divided into two categories which is visual marker based and marker-free visual based tracking system. Visual marker based tracking system uses cameras to track human movement whilst marker-free visual based tracking system utilizes optical sensors to track human motion. These conventional methods are generally limited with the problems of sensors adrift, occlusion, and the 3D model unable to be rendered. As for non-visual based tracking systems, sensors are generally worn on a subject's body and the sensors are commonly categorised as mechanical, inertial, acoustic and magnetic based. The most common example of non-visual based tracking system is the accelerometer sensors which measures by using piezoelectric or variable capacitive. The non-visual system is bounded by drift errors and noisy measuring environment, but these problems can be corrected with internal offset and pre-selective filters (Zhou and Hu, 2008).

1.2 Problem Statement and Motivation

Inertial sensors have been used in Attitude & Heading Reference System (AHRS) application such as navigation of aircraft, ships and vehicles. With the recent advancement of technology in Micro-electro-mechanical systems (MEMS), the size, weight and cost of a wearable sensor has been largely reduced. Today, the smallest accelerometer available on the market is only $1.2 \text{ mm} \times 1.7 \text{ mm} \times 1.0 \text{ mm}$ in size invented by MEMSIC (MEMSIC, 2011). The availability of small non-visual tracking system opens up the possibility of real-time human motion tracking for a longer period of time with the use of inertial and magnetic sensors that is low in current consumption. Thus, our aim of sensors selection in this project will be

generally focused on wearable inertial and magnetic sensors (Altun, Bsrshan and Tuncel, 2010).

In the earlier research works on human motion sensing, data collected is generally processed through the use of personal computers. This gives the disadvantage that data collected are bound to be in a certain range near the computer and cannot be tracked for a long period due to the weight and size of personal computers and the transmission method used. However, with the evolution of smartphones in the recent years, processing powers of smartphones have grown exponentially and is sufficiently equipped for computing data collected. Comparatively, the general smartphone held in the human hand today already has more processing power than the computer used by NASA for their first space launch project (Robertson, 2009).

Human motion tracking has been studied thoroughly long ago and this study provides a basic foundation of understanding how human motion is related to our daily lives. With the turn of 21st century, the public grows more educated and has acquired the awareness of how technology can improve and propagate healthy lifestyle. The increase in medical expenditure is among the leading factors that raise public concern and the general consensus is that prevention is better than cure. For example, rehabilitation for a stroke patient requires physical correction program and it necessitates the patient to visit the hospital for a period of time. Alternatively, technologies used for home based rehabilitation and physical therapy are available but only through visual tracking system where the major disadvantage lies in the detection zone being confined to a limited area in the building and the inconvenience in mobility (Altun, Bsrshan and Tuncel, 2010). Hence, these problems have inspired and motivated us to create a device that can track and differentiate human motion on-the-go for a long period of time and simultaneously provides health information.

1.3 Objectives

The main objective of this project is to build a wearable sensor system that features the measurement in number of footstep, estimation of the speed and distance travelled and provides a user interface by communication through a wireless transmission module to an android mobile device. The wearable sensor system must be able to differentiate basic human motion such as walking, running, jumping, sitting and climbing stairs by using appropriate algorithms.

In the interest of fulfilling the objectives, the following actions are taken:

- To study different types of inertial and magnetic system. In particular, MEMS sensors that is wearable and small in size and to compare the advantages and disadvantages with existing technologies.
- To study essential programming techniques such as C/C++ and Assembly Language to process sensors data with a suitable micro-controller.
- To study different types of established wireless communication standards such as IEEE 802.11b (Bluetooth) and IEEE 802.11g (Wi-Fi) that allows data transmission of the wearable system to android devices in a close distanced range and offline manner.
- To study and implement algorithms and methods that have the ability to pre-selectively determine data and make intelligent decisions to compensated for errors made by different sensors.

1.4 Outline of Project

This thesis will be organized as follow: In Chapter 2, a literature review will be done to investigate all the pros and cons of the methods, design preferences, algorithms and electronic components used in this project. Methodology is discussed in depth in Chapter 3 to explain the functionality of the system and some initial result will be analysed. Chapter 4 describes all the necessary procedures taken to setup the device

and to conduct studies and experiment. Lastly, Chapter 5 discusses the result obtained and addresses issues arise during the development of the prototype phase.

CHAPTER 2

LITERATURE REVIEW

2.1 Existing Products/Researches

2.1.1 Tri-Axis Inertial/Magnetic Package

Typical tri-axis inertial and magnetic sensors are used to sense human motions. The results collected in the sensors are then processed by a pre-selective filter in order to obtain a more accurate result from the noisy environment. The typical diagram of tri-axis inertial and magnetic sensors package follows by Figure 2.1.

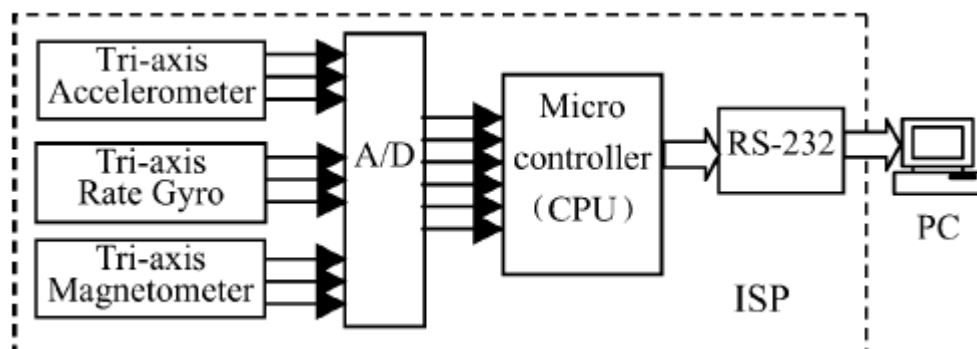


Figure 2.1: Block diagram of sensor package.

The paper here presented by Zhu and Zhou (2004) suggest the methodology of human motion sensing by using sensor packages interfacing with a microcontroller and the data is collected through a RS-232 cable to the computer.

The inertial and magnetic sensors that are used include two two-axis accelerometers, a single and a two axis magnetometers and also three single axis gyroscope. The orientation of the sensor package is evaluated with the Kalman algorithm in order to obtain the direction of the sensor package tilt and also for signal recognition in human motion.

The paper here provides us the basic idea about how the actual sensors will be used while we compare our objectives. Magnetometers in this system will be unnecessary for the project because mobile phones already have in built magnetometers. The usage of the mobile phone magnetometer is sufficient as we only need it for directional and navigational purposes. However, actual sensors such as accelerometer and gyroscope are still necessary although mobile phone also have them in built because the data collected will be more accurate. The RS-232 transmission method will be eliminated as well because the prefixed length of cable limits the mobility of the system.

2.1.2 MTx 9 DOF IMU Sensors

MTx Inertial Measurement Unit (IMU) is manufactured by Xsens Technologies and is one of the inertial and magnetic sensors combination chipset that is currently available in the market for motion sensing and navigation purposes. It is a digital measurement unit that measures acceleration, rotation and earth-magnetic field. It contains tri-axis accelerometer, gyroscope and magnetometers which is also known as 9 Degrees of Freedom (DOF). MT9 which is a family of the MTx sensors has high angular resolution up to 0.05° and 3° RMS dynamic accuracy. MTx sensors can be interconnected with each other by using a 1m cable to the central processing unit Xbus Master (Zhou and Hu, 2008).

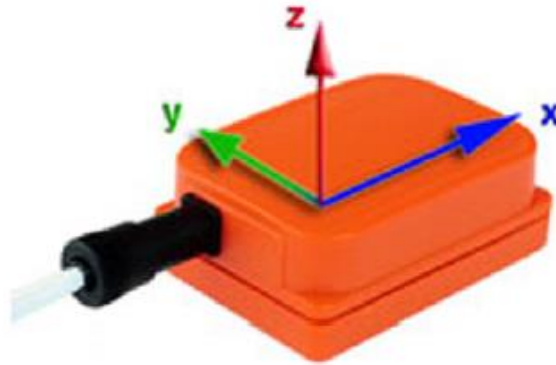


Figure 2.2: MTx Sensor.

The availability of such commercially available sensors chipset allows easier interfacing with sensors. Altun, Barshan and Tuncel (2010) introduced a method of classifying human activities with MTx sensors. The sensors are placed on 5 different parts of the body which is; the right arm, left arm, right leg, left leg and torso and is interconnected to an Xbus master. Several classification methods are used to differentiate different human activities and the most accurate classification method can go up to 98.8%. The result shows that MTx sensors are excellent in human motion sensing. However the cost of MTx sensors is very expensive and does not comply with our objective of creating a sensors system that is light, small and inexpensive.

2.1.3 Jawbone Up

One of the popular motion tracking gadgets in the market is Jawbone UP. Jawbone is a bendable and waterproof gadget that tracks user's health, promotes good sleeping pattern and exercise daily. Jawbone UP contains two lithium ion batteries, a tri-axis accelerometer, vibrating motor and a TRS plug to sync data to smartphone devices. Jawbone UP's most attractive feature is enabling users to track sleeping patterns, by using sensors to register subtle movements. The sensor in Jawbone UP is similar to

an actimetry sensor which registers user's movement when they rest and sleep. Besides a well-developed sleep mode tracking function, Jawbone UP also includes the vibrating motor to alert user from time to time to get up and exercise, or as an alarm to wake user from sleep.



Figure 2.3: Jawbone Up

The disadvantage of Jawbone UP is where it is unable to provide wireless syncing function. Each operation of the product requires the user to take of the device and connect to a smartphone through the TRS plug. Besides, Jawbone does not provide any visible display for users and only possesses one button interface for users to manually keep track of the three of the basic functions which is entering/leaving sleep mode, power nap mode and stopwatch mode (Ha, 2012).

2.1.4 FitBit Flex

Fitbit Flex, much alike the previous product Jawbone UP, is also a popular gadget that tracks users movement daily. Fitbit Flex features wireless syncing function which allows the device to upload data through a built-in wireless Bluetooth chip and

also a real time data tracking. Fitbit Flex includes a visible interface on the device itself to allow users to control the functions and also to display several data. The visible interface appears as a user tap on the device several times. Fitbit Flex also has a 5 LED panel that displays user's daily progress through the LED indicator. Similarly like the technology used in Jawbone UP, Fitbit Flex also includes a tri-axis accelerometer and also a vibrating motor to track user's movement and alert user. Fitbit Flex also includes sleep pattern monitoring to indicate users sleeping condition (Goode, 2013).

2.1.5 Nike FuelBand

Another product that shows analogous traits is the Nike FuelBand which was developed by the sports giant Nike. The Nike FuelBand as compared to other popular products such as Fitbit and Jawbone shows more hardware inferiority as Nike FuelBand does not include additional features such as sleeping pattern optimized alarm. Nike's marketing strategy on this product was not based on the hardware feature but a different calculation algorithm which is called Nike Fuel. The basic goal is to create users motivation in exercising. By accomplishing certain exercises, users gain Nike Fuel points and can be updated to either their smartphone app or users internet profile.

The FuelBand is actually made of thermoplastic rubber and has 20 LED that shows red, yellow and green according to goals set by users. Another 100 white LED is used to show time, Nike Fuel Points, calories and steps taken which can be revealed by a button attached on the band. The only motion detecting sensor in the band is a tri-axis accelerometer plus two lithium polymer batteries that can be recharged through a USB port. The wireless transmission method that connects the band to a smartphone is through a Bluetooth chip.

Although Nike FuelBand enjoys popularity among sports enthusiasts, several disadvantages are presented. Like the two other previous products, the only sensor

found in Nike FuelBand is the accelerometer and accelerometer has the characteristic of drifting error. Drifting error causes sensor to be inaccurate and the only way to solve this problem is by reset sensor data, inner compensation method or pre-selective filtering. Notably, Nike in partnership with the Arizona State University monitors test participants oxygen consumption on several basic activities in order to correlate with motion which the method is denoted as “oxygen kinetics”. This compensation method is inaccurate due to the oxygen consumption results are taken in the lab and it might vary from individual users (Fankhauser, 2013). Lastly, wrist worn tracking gadgets are bound to create static motion errors which might need a different algorithm to detect static motion such as standing and sitting while the arm is still moving.

2.1.6 Basis

Basis, another similar product on the wearable sensor market, is based on the identical concept of monitoring user’s health conditions such as sleeping, activities, calories spent and intake and also heart rate. The Basis band includes 4 different sensors which all have different purposes. The ECG sensor is used to sense blood flow of user throughout the day to monitor activity intensities. Alike the existing products, Basis also includes a tri-axis accelerometer for monitoring users step count, body movement in day time and sleep quality in the night time. A perspiration monitor sensor is also included to monitor user’s workout intensity by monitoring the sweat level on user’s wrist. Lastly, the Basis includes a thermistor-like sensor to monitor user’s skin temperature in correlation to the environmental temperature. The Basis band also includes a Bluetooth module and also a USB connector to allow users sync data to either a computer or a smartphone that runs on IOS or Android operating system (Basis, 2012).

2.2 Conclusion

Table 2.1: Comparison of Existing Products

Product	Sensors	Sensors Position	Data Transmission Method	Sleep Tracking
Tri-axis inertial/magnetic package	3 axis accelerometer, magnetometer, gyroscope	Wrist	RS-232	No
MTx	3 axis accelerometer, magnetometer, gyroscope	Multiple Positions	USB	No
Jawbone Up	3 axis accelerometer	Wrist	Bluetooth, TRS plug	Yes
Fitbit Flex	3 axis accelerometer	Wrist	Bluetooth	Yes
Nike FuelBand	3 axis accelerometer	Wrist	Bluetooth	No
Basis	3 axis accelerometer, ECG sensor, thermistor, perspiration sensor	Wrist	Bluetooth, USB	Yes

From all the products reviewed, the conclusion derived is to use one tri-axis accelerometer and gyroscope to monitor user's activity. Activities that could be monitored through both sensors are step counts, speed, micro-movements during sleep and also simple motions performed by user. The aspect where our project differs and distinguishes itself from the existing products lies in the complementation

of the gyroscope's measurement. Accelerometers measures the inertial force caused by movements. However, such forces could also be affected by gravitational force (even in steady state) as well as being sensitive to vibrations and noises. The gyroscope complements the accelerometer as it measures angle of rotation, which in terms is less prone to mechanical noises and small vibrations. However, the gyroscope suffers from sensor drifts which essentially is an error value measured when it returns to a zero-rate value (Gadget Gangster, 2010). In order to resolve the drawback characteristics of the sensors, we will apply an algorithm to solve the problem, precisely by using quaternion rotation as the solution to monitor user's leg gesture.

In order to perform heavy load calculation and raw sensor values sampling at the same time, a micro-controller will be used to perform complex calculation to off-load the calculations needed to be performed on the sensor chip. A Bluetooth device will be connected to the micro-controller so we could transmit the data and result to an Android mobile device for further data interpretation and for users to view data in real time.

The sensors will be placed at the shank position on the human body (precisely defined as the human body segment in between the knee and ankle). The main reason of doing so is because the existing products position the sensor on the wrist and is bound to static motion error such as hands movement. The shank position is also bound to static motion error such as the action of the leg shaking. But this could be resolved by adding a shake detection function and also a zero movement function to detect while sitting, which we highlight, simply could not be done by placing sensors on the wrist (as human hands are constantly moving).

CHAPTER 3

METHODOLOGY

3.1 Overview

The overview of how the project will be implemented will follow as in Figure 3.1.

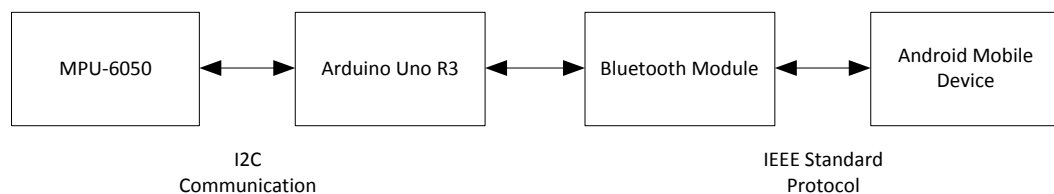


Figure 3.1: Block Diagram of System.

Firstly the sensor module chosen is MPU-6050. MPU-6050 is an Inertial Measurement Unit (IMU) with six degrees of freedom (6DOF). Inertial Measurement Units is differentiated by the degrees of axis that they are able to measure. The sensor module features a tri-axis accelerometer and a tri-axis gyroscope thus the 6 degrees of freedom. The MPU-6050 has a 16-bit analog to digital converter (ADCs) for digitizing the output of each axis of the sensors. A on chip 1024 bytes FIFO buffer in the IMU helps to store sensor data temporary and reduces the system power consumption. It also has a Digital Motion Processor (DMP) that is able to perform simple calculation for the sensor fusion data to offload the responsibility of the master processor. The MPU-6050 has different tracking range for both fast and slow

motion that is user-programmable in the full scale range of ± 250 , ± 500 , ± 1000 and $\pm 2000^\circ/\text{sec}$ for the gyroscope and a full scale range of $\pm 2\text{g}$, $\pm 4\text{g}$, $\pm 8\text{g}$ and $\pm 16\text{g}$ for the accelerometer. The MPU-6050 is chosen because of the low cost and more functionality that is able to perform and it is compatible with the microcontroller used in this project (Invensense, 2012).

In communication with the microcontroller the only communication method available is the TWI/ I2C communication which is mentioned in the datasheet. Two Wire Interface (TWI) or Inter-Integrated Circuit (I2C) communications which was originally invented by Philips as a communication method for two or more integrated chips. The communication only uses two bidirectional lines for the communication between devices which are commonly known as the Serial Data Line (SDA) and Serial Clock Line (SCL). One of the two devices which is the master device, (in this case the Arduino Uno R3) which sends queries to the slave device (which is MPU-6050) to obtain data. Serial Clock Line (SCL) is a clock line bus that synchronizes the event of read/write from the master to the slave device. Serial Data Line (SDA) is the data bus used to transfer the data but read or write can only be performed one at a time and thus an acknowledgement signal is required from both sides to synchronize the event (Wikipedia the free encyclopedia, 2013b).

The main processing of all the sensor data happens in the micro-controller which will be the main component in this project. The objective of using the micro-controller is it serves as a medium for the raw sensor values to be processed into useful information and translated into an IEEE standard protocol for Bluetooth before sending to the user's Android device. The Arduino Uno R3 features an 8-bit ATmega328P microcontroller which has a 16MHz of frequency oscillator. The reason Arduino Uno R3 is selected to be use in this project is because Arduino is popular for project prototyping. Arduino does not require a conventional program loader like PIC microcontrollers (and thereby saving the cost of acquiring one) and the program can be loaded into the chip using USB cable. The program is developed in C++ programming languages and Arduino projects are open-sourced so that the coding of integrated chips can be easily acquired. The diagram below shows the physical appearances of Arduino Uno R3.



Figure 3.2: Arduino Uno R3

The Arduino is connected to a Bluetooth module which can be communicated with the Android mobile device. As shown in the figure below, the comparison of each communication technology is used to be considered in this project. Bluetooth Low Energy as announced by Google in 2013 is compatible in Android mobile devices (Google, 2013). Bluetooth Low Energy is selected as it meets the requirement of the project of low power consumption. Zigbee has similar characteristic however, though is not supported in Android devices. Bluetooth Low Energy can be used from a range within 50m and a transfer rate of 1Mbit/s. The peak current consumption is less than 15mA which is much lower than the other technologies. Bluetooth Low Energy has the same spectrum range as the classic Bluetooth technology which is 2.4GHz-2.4835GHz, and thus it can be compatible to Android device users who do not have Bluetooth Low Energy functions (Wikipedia the free encyclopedia, 2013c).

	Voice	Data	Audio	Video	State
Bluetooth ACL / HS	x	Y	Y	x	x
Bluetooth SCO/eSCO	Y	x	x	x	x
Bluetooth low energy	x	x	x	x	Y
Wi-Fi (VoIP)	Y	Y	Y	Y	x
Wi-Fi Direct	Y	Y	Y	x	x
ZigBee	x	x	x	x	Y
ANT	x	x	x	x	Y

State = low bandwidth, low latency data

Low Power

Figure 3.3: Comparison of Wireless Communication Technologies.

3.2 Sensors

3.2.1 Accelerometer and Gyroscope

In this section, we will explain the working principles of the accelerometer and gyroscope and how to extract the raw sensors values.

Accelerometer functions as a type of sensor that measures the acceleration of the sensor itself. The sensor measures acceleration not in a way that is congruent with the definition in physics (which is rate of change of velocity with respect to time). Accelerometer measures the acceleration experienced by its weight resting at the frame of the reference. The imaginary concept of the working principle of accelerometer can be seen as an object resting on the point of origin of 3 axes as shown in the figure below. The box would represent the full scale range defined by gravity which is $\pm 1g$ ($1g=9.87m/s^2$) on each axis. This would be the ideal case that this imaginary box is in outer space which all measures are zero. Thus by the effect of gravity, accelerometer will measure an acceleration of $1g$ straight upwards given the presence of gravity on earth. Figure 3.2.2 illustrates the effect on accelerometer with gravity in presence, with notice that the accelerometer measures in the opposite direction with respect to the gravity.

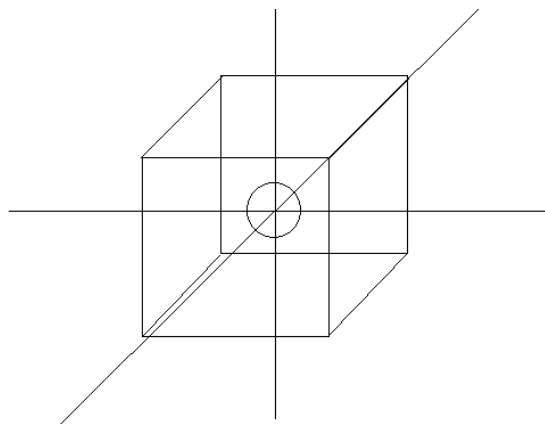


Figure 3.4: Concept of Accelerometer.

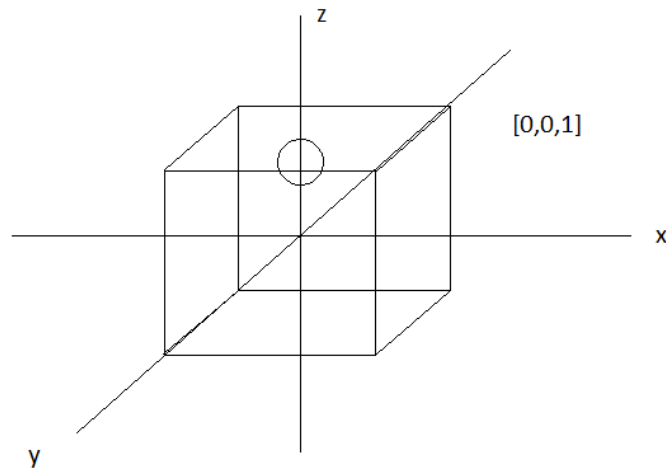


Figure 3.5: Effect of Accelerometer by Gravity.

Gyroscopes have similar traits as the accelerometer, except that it measures the rotation with respect to the object's last position. Thus in simple gyroscope measure rate of rotation in degrees per second ($^{\circ}/\text{sec}$).

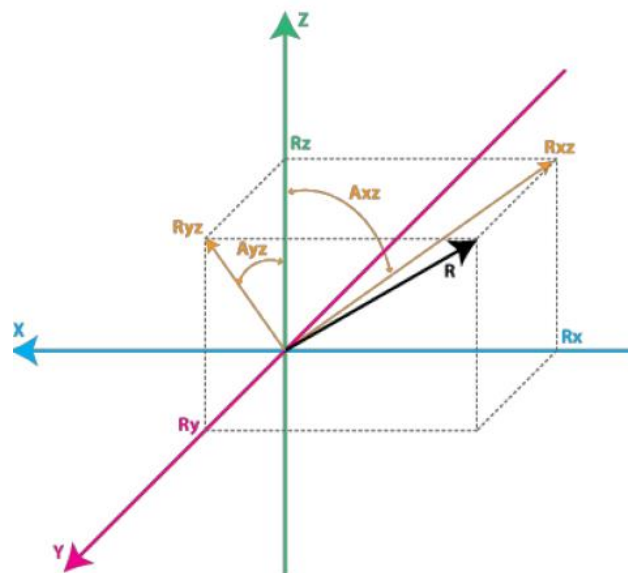


Figure 3.6: Gyroscope with Vector R

As shown in Figure 3.6, we can see that:

$$\begin{aligned}
 R_{xz}^2 &= R_x^2 + R_z^2 \\
 R_{yz}^2 &= R_y^2 + R_z^2 \\
 R^2 &= R_{xz}^2 + R_y^2, \text{ or} \\
 R^2 &= R_x^2 + R_{yz}^2
 \end{aligned}$$

A_{xz} is the angle between R_{xz} and Z axis and A_{yz} is the angle between R_{yz} and Z axis.

Thus the equation that defines a gyroscope is,

$$\frac{\partial A_{xz}}{\partial t} = \frac{A_{xz1} - A_{xz0}}{t_1 - t_0} (\circ / s)$$

Assuming that we measure a rotation angle around axis Y at t_0 and later at t_1 we would have the equation above.

3.2.2 Initial Results

The circuit connection between MPU-6050 and Arduino Uno R3 is connected as shown in the figure below. The operating range of the MPU-6050 is 2.375-3.4V is as stated in the datasheet. Thus we will directly use the ready 3.3V voltage source that the Arduino is able to provide. The Ground (GND) and AD0 are connected to both the ground pin on Arduino. The AD0 pin functions to defining the default I2C address and thus we need to ground it for the default evaluation purposes. SDA and SCL on the sensor module are then connected to the dedicated I2C communication on the Arduino which is the A4 and A5 analogue pin. As for obtaining the raw sensor values, the MPU-6050 outputs a digital signal from the ADC for each of the sensors. The sensors values are represented by 16-bit 2's complement format. Take example for a $\pm 2g$ full scale range accelerometer the bits that are able to represent the scale

are $2^{16}-1$ which is a total of 65535 bits. $-2g$ is represented as -32767 bit and $2g$ is 32767 (Invensens, 2012).

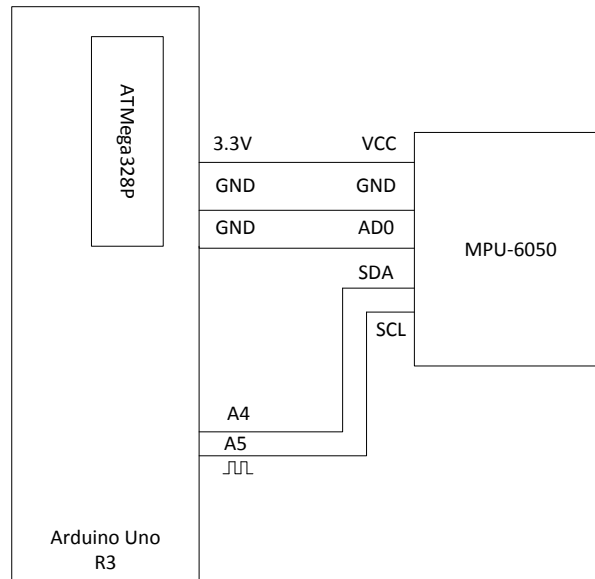
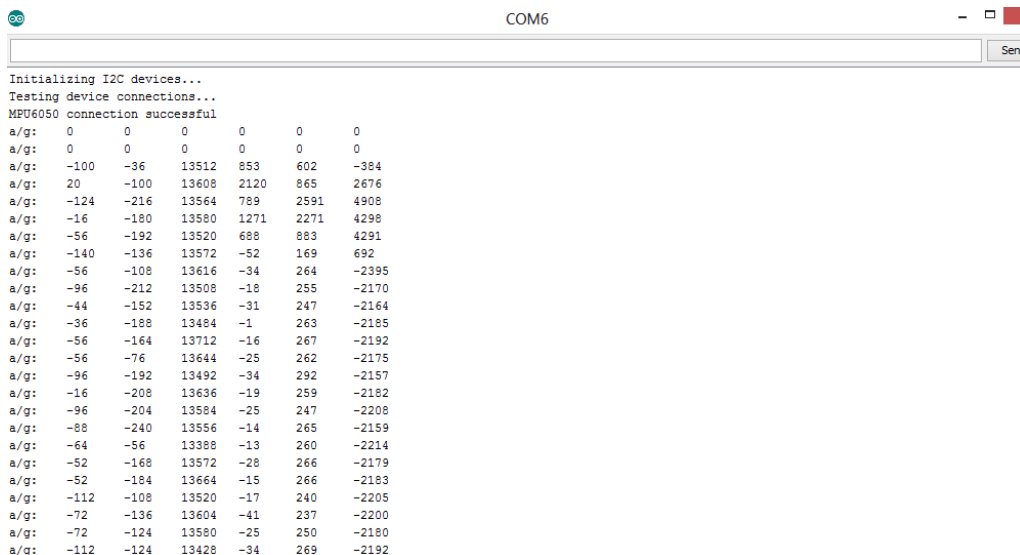


Figure 3.7: Connection between Arduino Uno R3 and MPU-6050.

The coding of Arduino to use MPU-6050 is as referred from open source code by Rowberg (2011). The initial result as obtained from the Arduino sketch is shown in the Figure 3.8. The sensor is placed on a table with no movements to test the validity of the sensors. However we noticed that there is an offset on the z-axis of the gyroscope which might be caused by the error calibration of the hardware. Thus we needed to offset a value of 2000-bits from it as a correction. The result after offset is shown in Figure 3.9. The frequency is set to 115.2 kHz which is the maximum bit rate that can be monitored on Arduino.

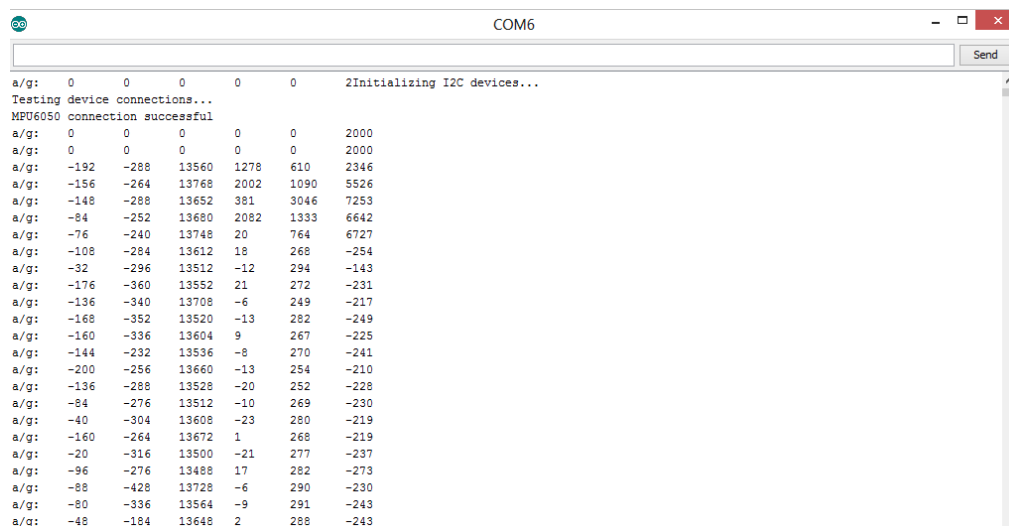


```

COM6
Initializing I2C devices...
Testing device connections...
MPU6050 connection successful
a/g: 0 0 0 0 0 0
a/g: 0 0 0 0 0 0
a/g: -100 -36 13512 853 602 -384
a/g: 20 -100 13608 2120 865 2676
a/g: -124 -216 13564 789 2591 4908
a/g: -16 -180 13580 1271 2271 4298
a/g: -56 -192 13520 688 883 4291
a/g: -140 -136 13572 -52 169 692
a/g: -56 -108 13616 -34 264 -2395
a/g: -96 -212 13508 -18 255 -2170
a/g: -44 -152 13536 -31 247 -2164
a/g: -36 -188 13484 -1 263 -2185
a/g: -56 -164 13712 -16 267 -2192
a/g: -56 -76 13644 -25 262 -2175
a/g: -96 -192 13492 -34 292 -2157
a/g: -16 -208 13636 -19 259 -2182
a/g: -96 -204 13584 -25 247 -2208
a/g: -88 -240 13556 -14 265 -2159
a/g: -64 -56 13388 -13 260 -2214
a/g: -52 -168 13572 -28 266 -2179
a/g: -52 -184 13664 -15 266 -2183
a/g: -112 -108 13520 -17 240 -2205
a/g: -72 -136 13604 -41 237 -2200
a/g: -72 -124 13580 -25 250 -2180
a/g: -112 -124 13428 -34 269 -2192

```

Figure 3.8: Raw Sensor Values of MPU-6050.



```

COM6
Initializing I2C devices...
Testing device connections...
MPU6050 connection successful
a/g: 0 0 0 0 0 2000
a/g: 0 0 0 0 0 2000
a/g: -192 -288 13560 1278 610 2346
a/g: -156 -264 13768 2002 1090 5526
a/g: -148 -288 13652 381 3046 7253
a/g: -84 -252 13680 2082 1333 6642
a/g: -76 -240 13748 20 764 6727
a/g: -108 -284 13612 18 268 -254
a/g: -32 -296 13512 -12 294 -143
a/g: -176 -360 13552 21 272 -231
a/g: -136 -340 13708 -6 249 -217
a/g: -168 -352 13520 -13 282 -249
a/g: -160 -336 13604 9 267 -225
a/g: -144 -232 13536 -8 270 -241
a/g: -200 -256 13660 -13 254 -210
a/g: -136 -288 13528 -20 252 -228
a/g: -84 -276 13512 -10 269 -230
a/g: -40 -304 13608 -23 280 -219
a/g: -160 -264 13672 1 268 -219
a/g: -20 -316 13500 -21 277 -237
a/g: -96 -276 13488 17 282 -273
a/g: -88 -428 13728 -6 290 -230
a/g: -80 -336 13564 -9 291 -243
a/g: -48 -184 13648 2 288 -243

```

Figure 3.9: Corrected Raw Sensor Values of MPU-6050

3.2.3 True Acceleration Accelerometer Model

As mentioned earlier, the gravity component will act on the accelerometer sensor when it is tilted towards that axis. This means that even though the device does not experience any acceleration, the accelerometer will still output a value of $1g$ on either one of the axis depending on the orientation of the accelerometer. This phenomenon will cause an inaccurate reading when the device is trying to monitor the acceleration produced by leg gestures. Thus, the gravity component must be removed from the accelerometer raw values. The true acceleration model reduces the gravity component by subtracting the original accelerometer reading by $1g$ when that particular axis is titled. This removes any unwanted gravity component acting on the accelerometer and gives the true acceleration value in “g” when the user performs any kind of gesture in the experiment. The true accelerometer can be seen in the figure below. The figure is an imaginative view of the starting point of the accelerometer.

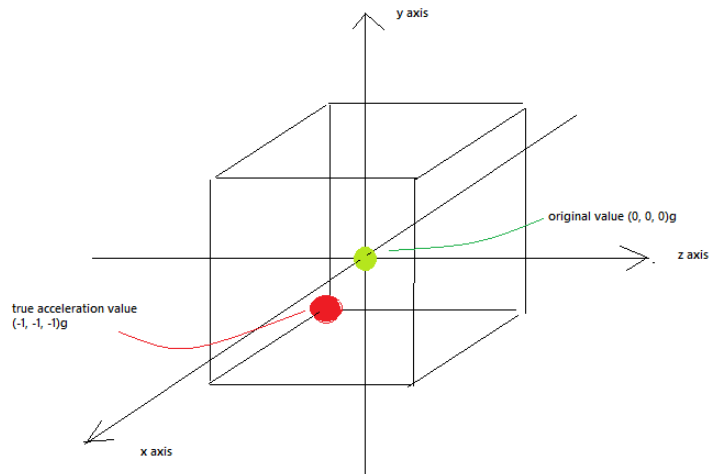
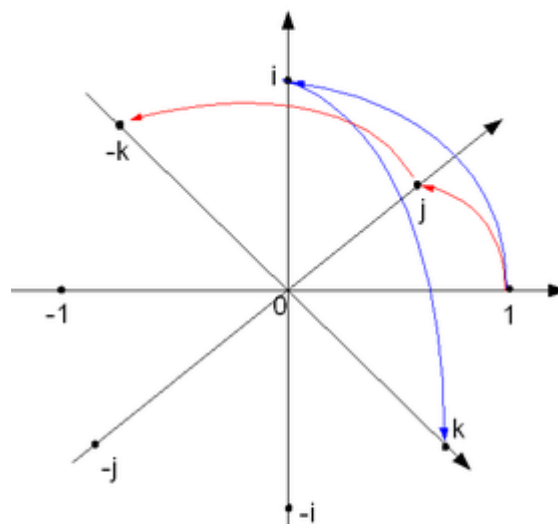


Figure 3.10: True Acceleration Model.

The acceleration value here is represented in bits. However the system outputs it into raw bit value with a 16 bit signed maximum thus reducing by $1g$ is equals to about 8192 bits in the raw accelerometer value.

3.2.4 Quaternions

Quaternions are complex numbers that are used to represent the orientation of a three dimensional rotation. They are four-dimensional that requires a division of real numbers. They are a much more efficient representation of a rotation as compared to methods like Euler Angles because the Euler Angle suffers from the Gimbal Lock problem which is essentially loss of one degree of freedom during rotation. Quaternion is often represented with one real number (in this project it is denoted as $q.w$) and three imaginary number (denoted as $q.x$, $q.y$, $q.z$). The three imaginary numbers are orthogonal to one another. The concept of the quaternion imaginary numbers are shown in the figure below.



Graphical representation of
quaternion units product as
90°-rotation in 4D-space

$$\begin{aligned} ij &= k \\ ji &= -k \\ ij &= -ji \end{aligned}$$

Figure 3.11: Relationship of the three imaginary numbers of quaternion.

Some essential equations of quaternion are shown below.

$$q = xi + yj + zk + w \quad (3.1)$$

$$w^2 + x^2 + y^2 + z^2 = 1 \quad (3.2)$$

The quaternions are represented by a 3 by 3 matrix during rotation as shown in the equation below (Wikipedia, 2014a).

$$\begin{pmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2wy + 2xz \\ 2wz + 2xy & w^2 - x^2 + y^2 - z^2 & 2yz - 2wx \\ 2xz - 2wy & 2wx + 2yz & w^2 - x^2 - y^2 - z^2 \end{pmatrix} \quad (3.3)$$

The important part of understanding the quaternion lies on the equation 3.2. The value w in the equation is a real number that shows when either one of the imaginary number exceeds the maximum value (which is the value of 1) it will become a negative value and either two of the imaginary numbers will exchange values in the matrix. Take example, if you take a box faced up as shown in the figure below and turning it 90 degrees left, the z axis has taken the value of x whilst the x axis has taken the value of negative z . The denotation could not be changed and therefore requires a 3 by 3 rotation to represent the figure shown below correctly.

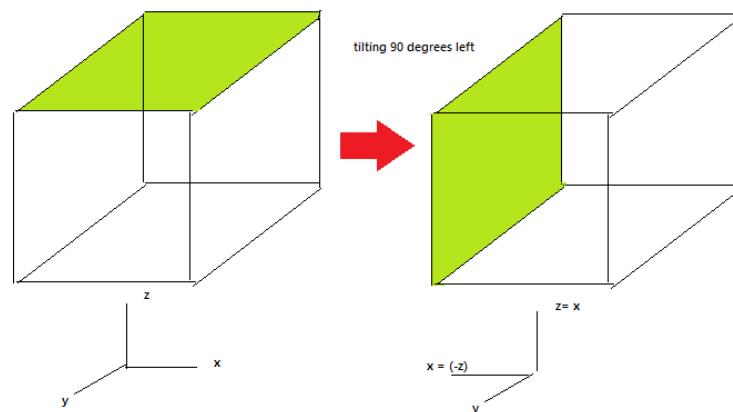


Figure 3.12: Before and After turning the box.

CHAPTER 4

PROJECT IMPLEMENTATION

In this chapter, the overall progress of setting up the sensor device and the development of the Android mobile application are discussed. Throughout developing the prototype device each precaution is noted and shown in this chapter too.

4.1 Device Setup

The device is setup by placing the whole circuit onto an Arduino prototyping breadboard as shown in Figure 4.1 as the whole circuit only consists of two components which are the Bluetooth HC-05 and also the Inertial Measurement Unit. This allows a much more convenient testing condition when the device is attached to the user's ankle.

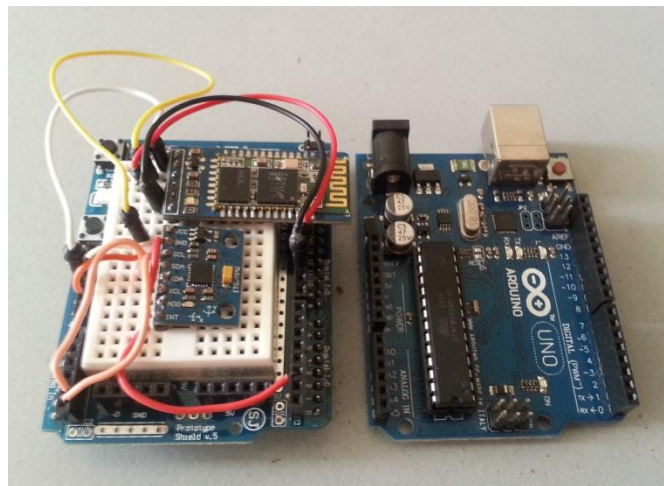


Figure 4.1: Sensors on prototyping board (left) & Arduino Microcontroller (right).

The device is then placed into a sports arm band to allow attachment to the user's ankle. The Arduino microcontroller is then connected to a portable battery bank with a capacity of 11200mAh with an output voltage of 5V and 1A which matches the operating voltage level of Arduino microcontroller. As of the current prototyping phase, the size and capacity of the power source is not an important consideration and we will therefore relegate the address of this issue in future development. The device is shown in Figure 4.2.



Figure 4.2: Device in black sport band and portable battery bank.

The measuring device is then placed on the right side of the user's ankle regardless of left or right leg of the user. The reason behind this is for the ease of understanding the orientation of the accelerometer in such a way that positive X axis depicts forward direction; positive Y axis depicts upwards direction and positive Z depicts sideways direction to the right. The device attached to the user's ankle is shown in Figure 4.3. However, throughout the development of the device, the device has to be placed onto the user's left leg and right side of the ankle. This is to ensure that the calculated side step distance is accurate according to the results. This issue is addressed in **Chapter 5**.



Figure 4.3: Device attached to user's leg.

The measured results of each test are logged into text file format by using the terminal software TeraTerm which allows the connection from computer to the HC-05 Bluetooth. The software allows communication of Bluetooth devices to receive and send data which could be recorded down and analysed through Microsoft Excel and display data in graphs and tables. A demonstration of using TeraTerm to collect data is shown in Figure 4.4 below.

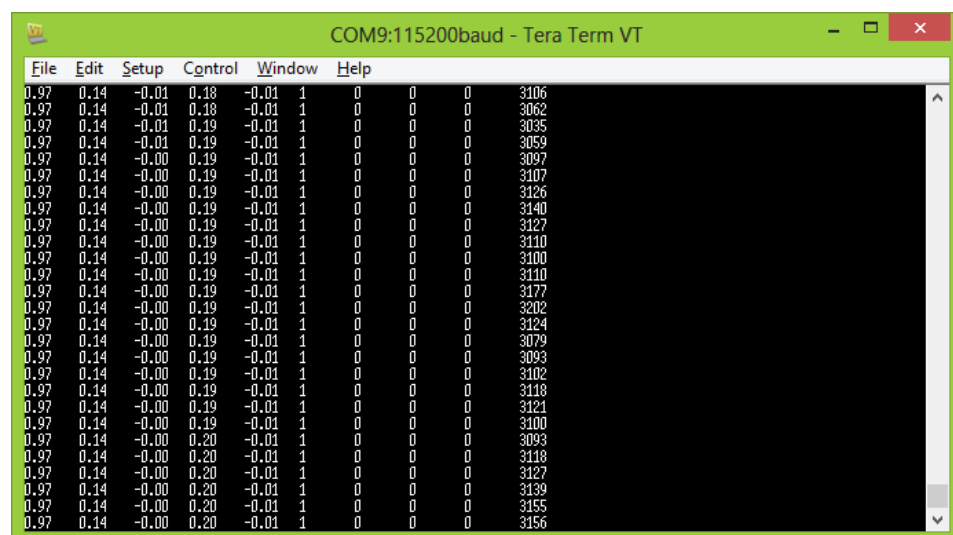


Figure 4.4: Using TeraTerm to log in results.

4.2.2 Step Distance Calculation

The flowchart below shows the sequence of the program to determine the individual steps distance made by user. This section is discussed in **Chapter 5.2**.

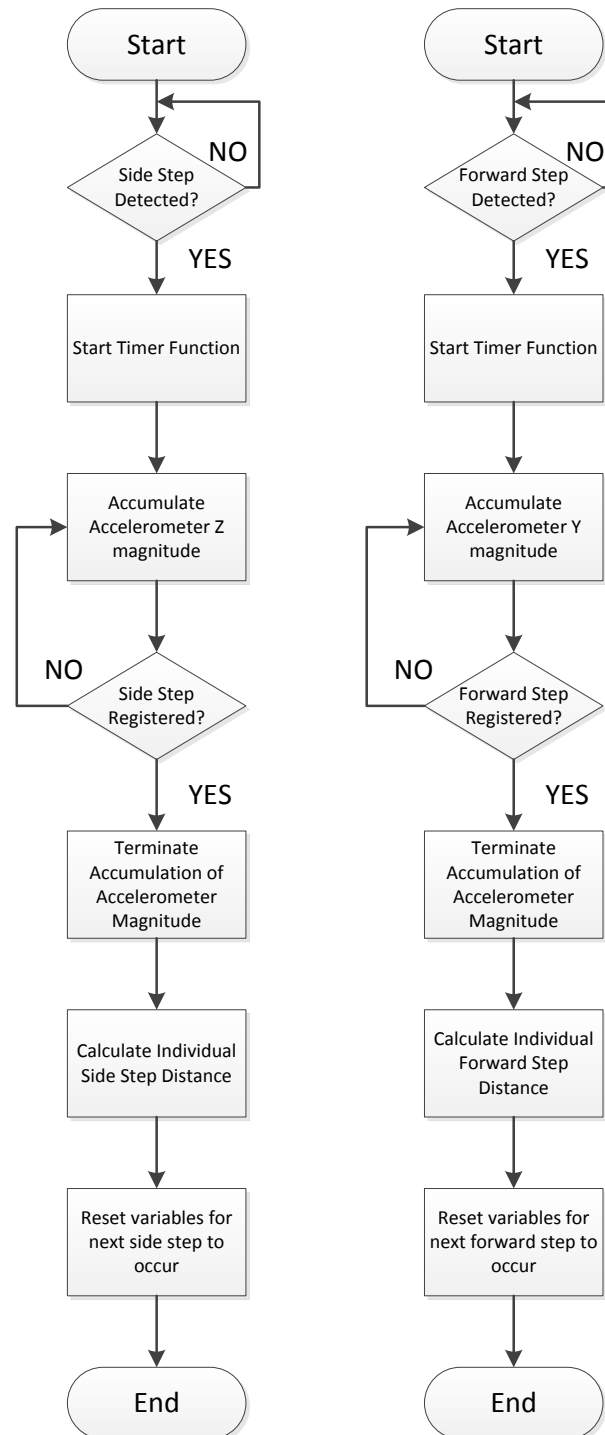


Figure 4.6: Flowchart of Step Distance Calculation.

4.3 Android App Development

In order for users to receive important information such as the step counts and step distance, an Android mobile application is created to allow users to view their results and to sort out the information received by the Arduino microcontroller. However, before we start to create the application, an application called SENA BTerm is used to communicate with Arduino. The application can be found in Google Play Store. The application is able to receive the data packets sent by Arduino correctly. However it lacks a friendly user interface and is difficult to understand by a common user without Bluetooth knowledge. Thus, this application gives a general direction of how our own Android application in this project should be developed. Our application will be developed in the sense that it provided a simple user interface. This will ensure that anyone can use the app with the prototype without any knowledge required. The application is shown in Figure 4.7.

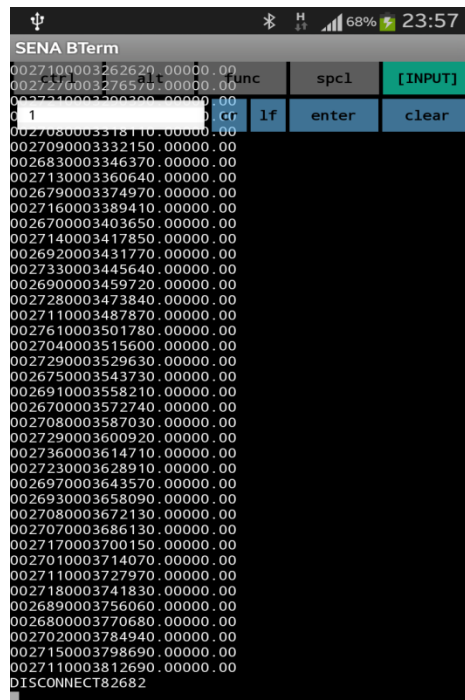


Figure 4.7: Data collection using SENA BTerm.

The software used to develop the mobile application is Eclipse. Android developers have provided dedicated plugin for the software Eclipse for the ease of development process. The overview of the software is shown in Figure 4.8.

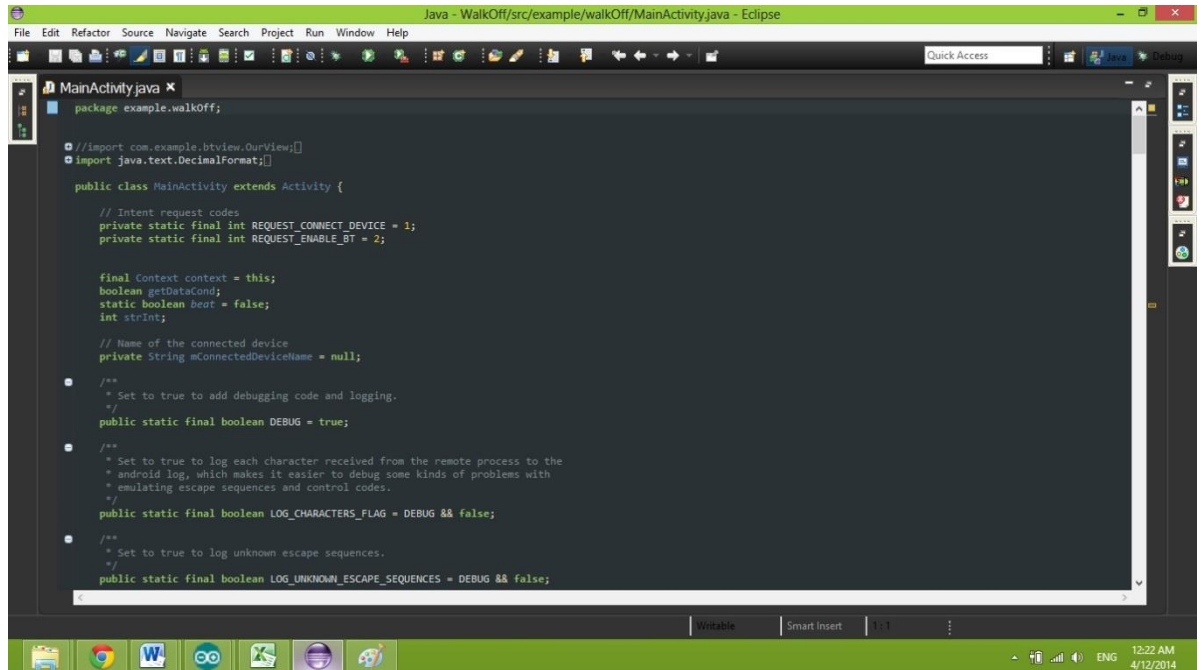


Figure 4.8: Eclipse with Android Plugin.

The final product of the application we have created is shown in the figures below. Upon opening the application, the user will be prompted to turn on their Bluetooth function in their mobile (Figure 4.9). The application then goes to the main interface (Figure 4.10) where users can view all the information. However, users are required to manually connect to the sensor device (Figure 4.11). The timer function allows users to track their information in real time and upon pressing the “Start” button in the application, it will send a signal to allow the sensor device to start detecting user’s leg gestures with the condition that the device is paired with the mobile’s Bluetooth (Figure 4.12).

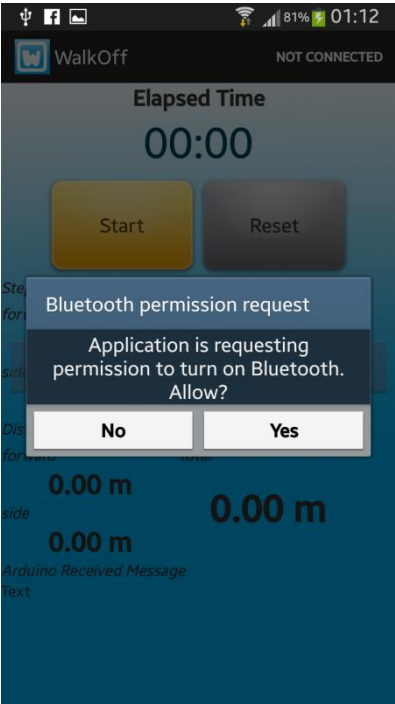


Figure 4.9: Application request Bluetooth.



Figure 4.10: Main Page



Figure 4.11: Manually connect device.

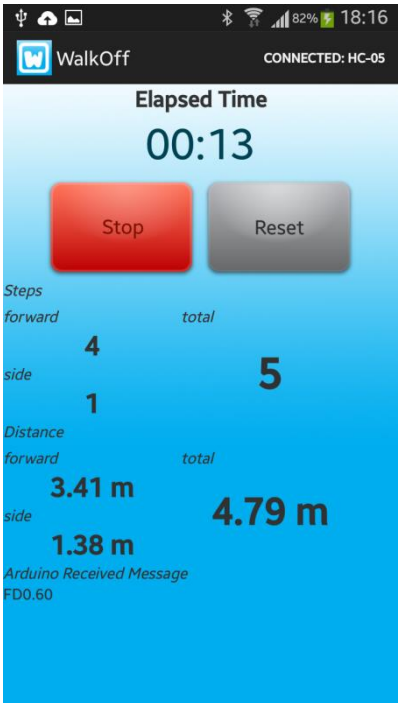


Figure 4.12: Data Collection.

4.3.1 Application Publication

The application is available in Google Play Store at <https://play.google.com/store/apps/details?id=example.walkOff> is as shown in the figure below. The application is published under the name “Walk Off” and it is free and available for all users to download. However, this application requires the user to have the device in order for the application to be fully functioned. Users who are interested in this application may contact the original author to purchase or request for the sensor device.

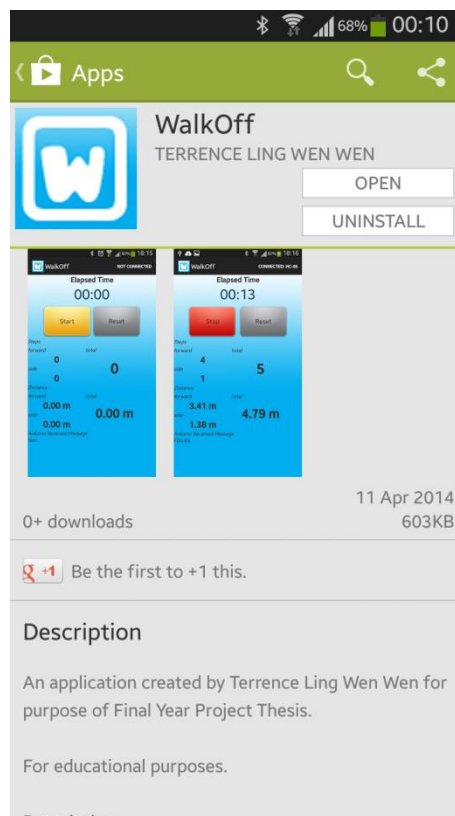


Figure 4.13: Application on Google Play Store.

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, results from different experiments are collected and analysed to allow the device to differentiate between different leg gestures from its respective user, the results and the challenging issues would then be discussed. The algorithm that is used to differentiate leg gestures are then tested in an empirical experiment to show how the result is affected by different users. The last section of this chapter illustrates how the developed prototype is compared against a commercial product available in the market. The advantages and disadvantages are then discussed in detail to show the potential of the developed prototype.

5.1 Motion Differentiation

As earlier mentioned in **Chapter 3.4**, the method used to differentiate a user's leg gesture will be through the quaternion rotation of the inertial measurement unit. Quaternions are obtained directly through the Digital Motion Processing (DMP) of the inertial measurement unit at a rate of 50Hz and for raw accelerometer and gyroscope data at a rate of 1000Hz. Both data are combined into a FIFO packet with a size of 42 bytes and transferred through the I2C bus at 115200 baud rate.

5.1.1 Static Motion / No Movement

In measuring a user's static motion which includes sitting, standing or no movement from the leg of attached device, we can expect a flat out straight line from the entire

axis in the quaternion rotation as shown in Figure 5.1. The result is also used as a reference to the initial position of the sensor to determine subsequent motions from user.

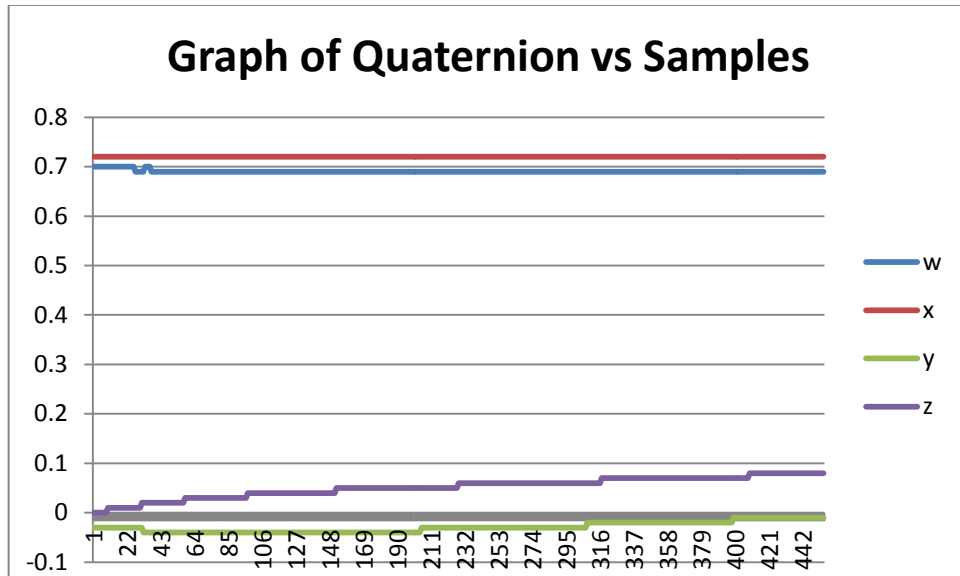


Figure 5.1: Static movement result.

5.1.2 Forward Step Detection

The next step of the project is to detect the most basic leg gesture which is a forward step. A simple forward step can be categorized into two motions which are lifting the leg and extending it to a certain landing point. A simple test is conducted by walking in a straight line for a certain amount of forward steps and analysing the data obtained.

Figure 5.2 below presents the result obtained. The actual steps performed by the user are 7 steps on the attached leg. It could be observed through the result that 7 peaks are obtained in particular the x, y and z axis of the quaternion rotation. An extra observation made is the time difference of the peak of the quaternion axis y and z. Axis y peaks are followed closely by axis z peaks which explain the motion of lifting the leg first that give rise of axis y and landing horizontally happens right after lifting the leg give rise to axis z. Another observation is the timing of axis x and axis

y peaks about the same time. This is because the lifting motion is not perfectly vertical which involves sideways motion that forms a vector of the direction of lifting the leg.

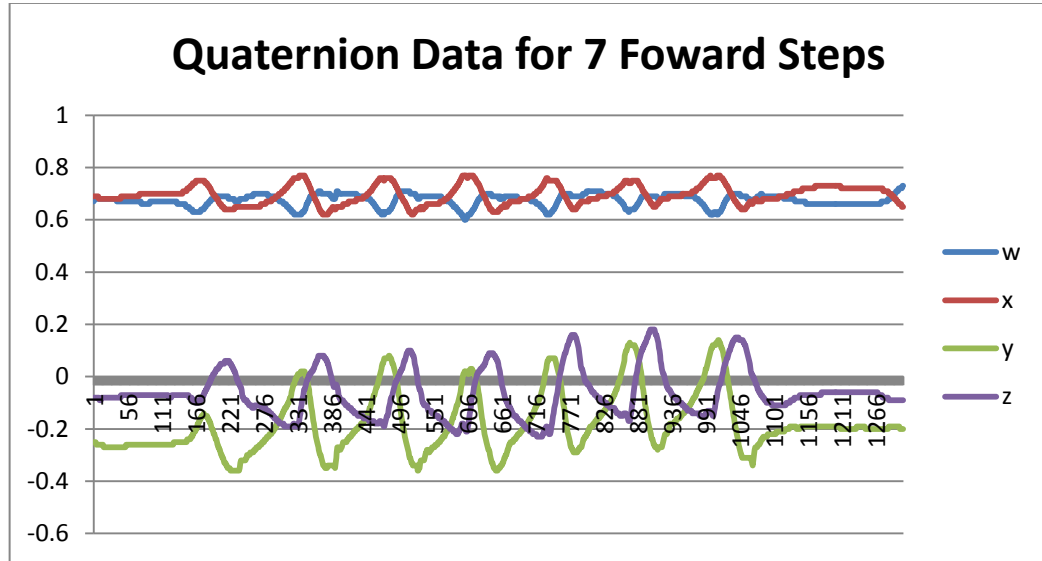


Figure 5.2: Result collected for 7 forward steps.

The motion of lifting a leg gives enough information to detect a forward step made. This can be done by drawing a threshold line across half of the sinusoidal signal. If quaternion axis y crosses the threshold for a certain period of time then a forward step is registered. The implementation of the threshold line can be done by using a moving average filter of a certain sample size. A moving average filter is a filter that determines the general trend of a particular sample size and the general equation of the filter is shown in the equation below. In this particular problem, a sample size of 50 is chosen such that the sample size is large enough to determine the overall trend and small enough so that it is not affected by the noisy raw quaternion raw data. The result is shown in Figure 5.3. We can observe that there are 7 peaks above the moving average filter and 7 troughs below the threshold line.

$$MovingAverage = (q.y / n) + MovingAverage - (MovingAverage / n) \quad (5.1)$$

where

q.y = Quaternion Axis Y

n = sample size

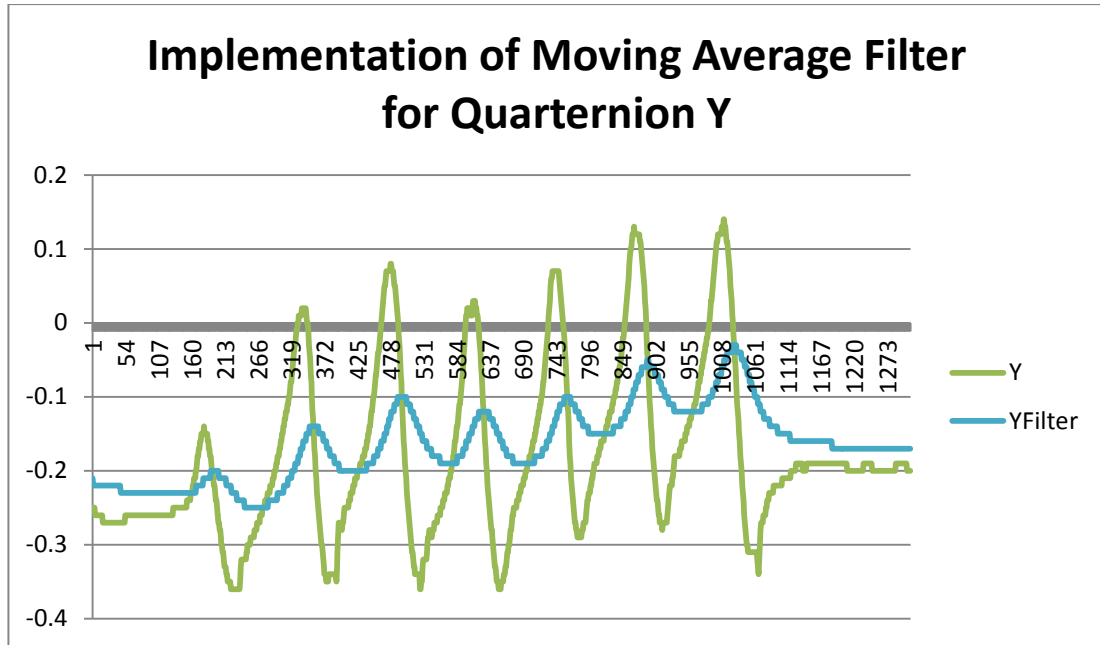


Figure 5.3 : Result of 7 forward steps with Moving Average Filter.

In order to further improve the forward step detecting condition, a range of 30% is added above the detection threshold, this can be seen in Figure 5.4. The figure shows that the threshold is slightly higher in value than the original moving average filter. This eliminates all the unwanted noise which might give rise to the detection of a forward step, which is not accurate. The equation is shown in 5.2 below. A period of 10 samples above the threshold is required for a forward step to be registered as a valid step. This is based on the flight time of lifting the leg motion. By using the timer functions in the program, the main loop of the program requires about 0.02s to obtain the next sample, which means that 0.2s of time is required for a step to be valid. The reason of selecting 10 samples as the requirement is intuitively based on the fastest record made by the athletes 100m sprinting world record. A world record of 9.58s made by Usain Bolt (Wikipedia, 2014) to complete 100 meter translate to a sample size of 479 samples and the athlete only used 21 strides (22.81 samples per step). Thus, it is a safe assumption that 10 samples is able to detect the fastest step made and clear out negligible ones.

$$ThresholdY = 1.3 \times MovingAverage \quad (5.2)$$

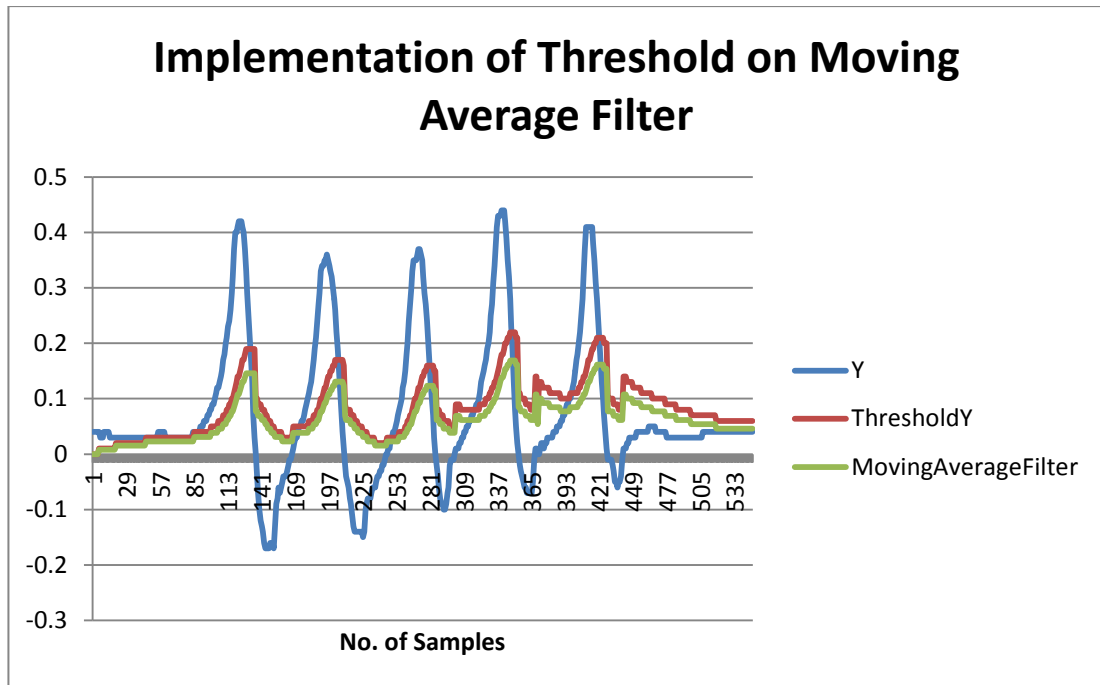


Figure 5.4: Result of 7 forward steps with Moving Average Filter.

5.1.3 Side Step Detection

The same procedure to detect forward step motion is also applied to detect side steps. Users performs side steps in a straight line for a certain amount of steps and the data is collected and analysed, the figure below shows the results collected.

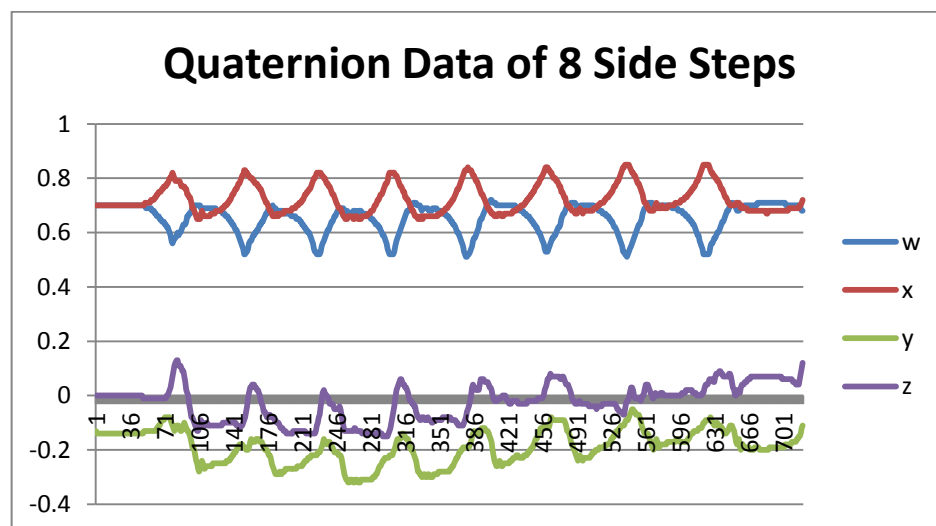


Figure 5.5: Results of 8 side steps.

From the data collected, the first observation we made is the peaks and troughs of quaternion axis w and x are much more significant as compared to forward steps. The sinusoidal waveform observed in the quaternion y and z axis is much smaller as compared in forward steps data. This could be used as the condition to differentiate between forward steps and side steps. By analysing from Figure 5.5, we found that the minimum of quaternion axis w is about 0.6 and a maximum of 0.78 for axis x in forward steps. As compared to forward steps, side steps has a minimum of 0.51 for axis w and maximum of 0.85 for axis x as presented in the figure above. Thus a double condition of when axis w is less than 0.6 and axis x is more than 0.78 is detected, a side step is registered. A sample size of 10 is also required for a side step to be validly registered, reason being as mentioned previously in Section 5.1.2. By applying the algorithm above, the side step count are shown in the figure below, which matches the actual side steps made in Figure 5.5.

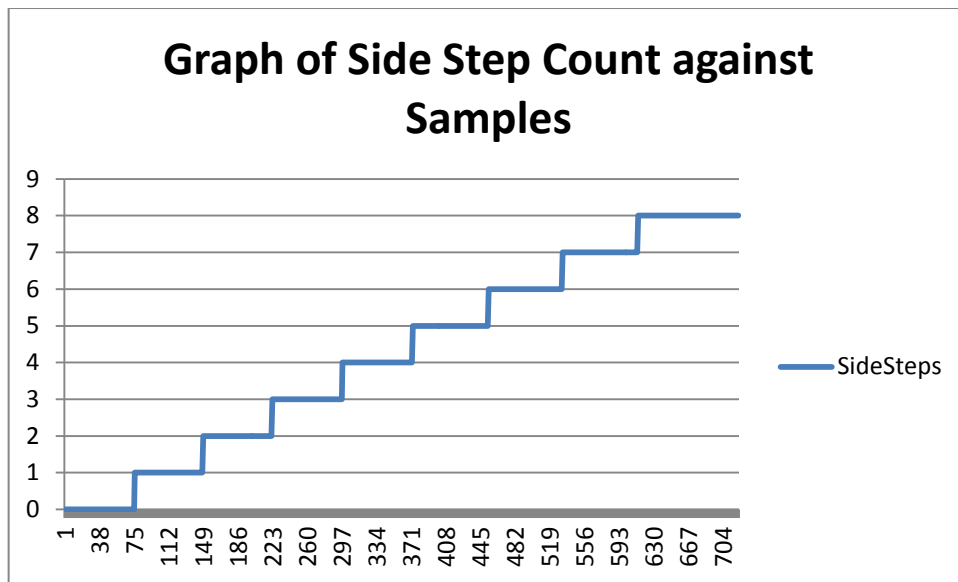


Figure 5.6: Result of Side Step Count.

However, if we look closely at the result shown in Figure 5.5 the quaternion axis y which is used to detect forward step also presents sinusoidal waveform but not in a big magnitude as forward steps. This may cause false positive detection of forward steps performed at the same time with side steps. An example of the false positive detection is shown in Figure 5.7 below. In order to simplify data collection,

a conditional variable Step Bit is assigned to be 1 when forward step condition is detected and Side Step Bit equals to 1 when side step condition is detected.

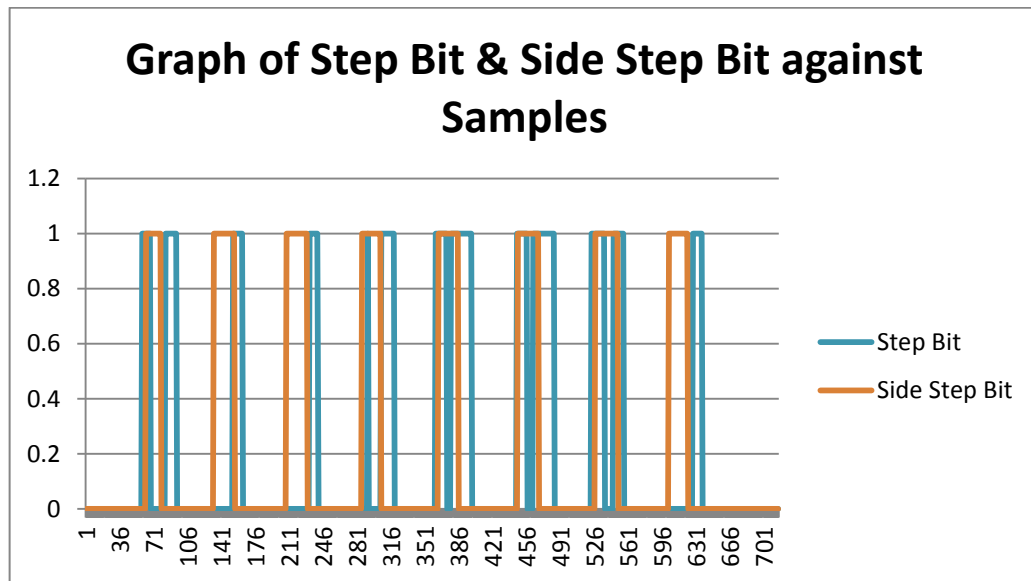


Figure 5.7: Result of 8 Side Steps with False Positive Error.

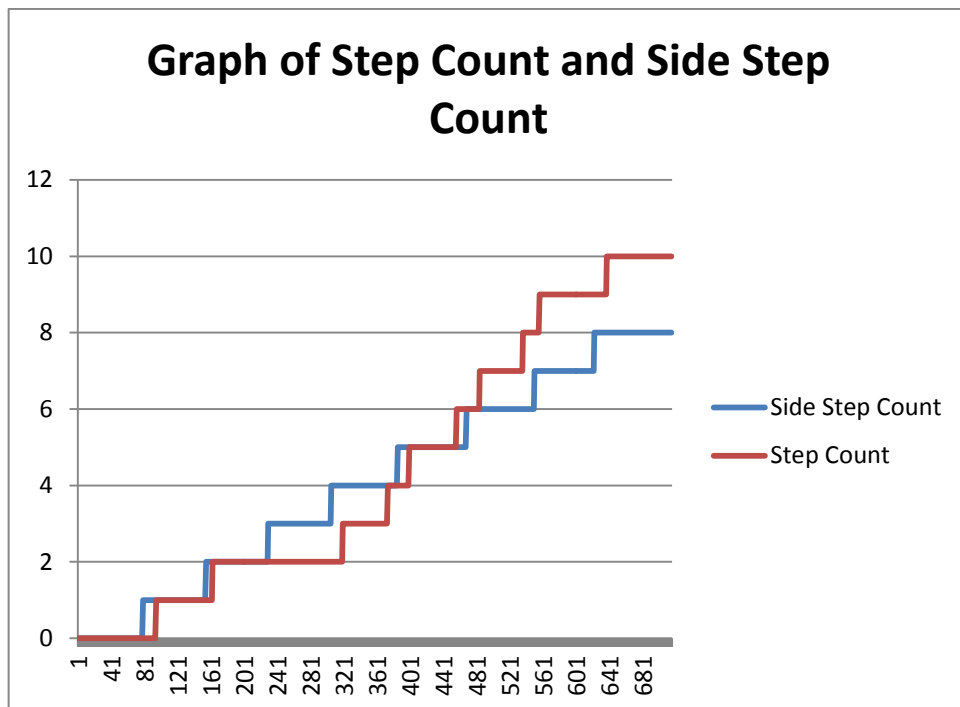


Figure 5.8: Result of Side Step Count with False Positive Error Forward Count.

As shown in Figure 5.7, 8 side steps are performed which the condition is able to detect it accurately as shown in Figure 5.8. However, an extra 10 steps of forward step count is accounted for the small sinusoidal data in quaternion axis y, which are all false positive gesture detected.

In order to eliminate the false positive forward steps detected, a First In First Out (FIFO) Buffer is created to register the previous 3 sample number of the forward steps detected. We can use this information to eliminate the forward steps made closely after a side step has occurred. The condition set here is that if any forward steps are registered within 30 samples of the rising edge of the valid side step, the particular forward step count is deducted from the total step count. An additional cool down period of 30 samples is also added after a side step is registered before any motion can be detected. The result is shown in Figure 5.9 below.

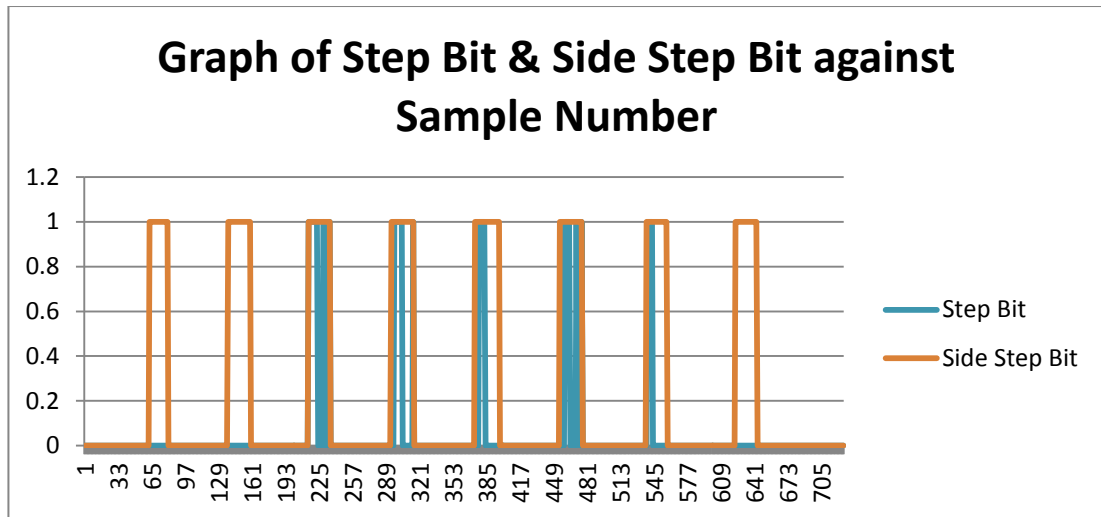


Figure 5.9: Result of Side Step Data with False Positive Error removed.

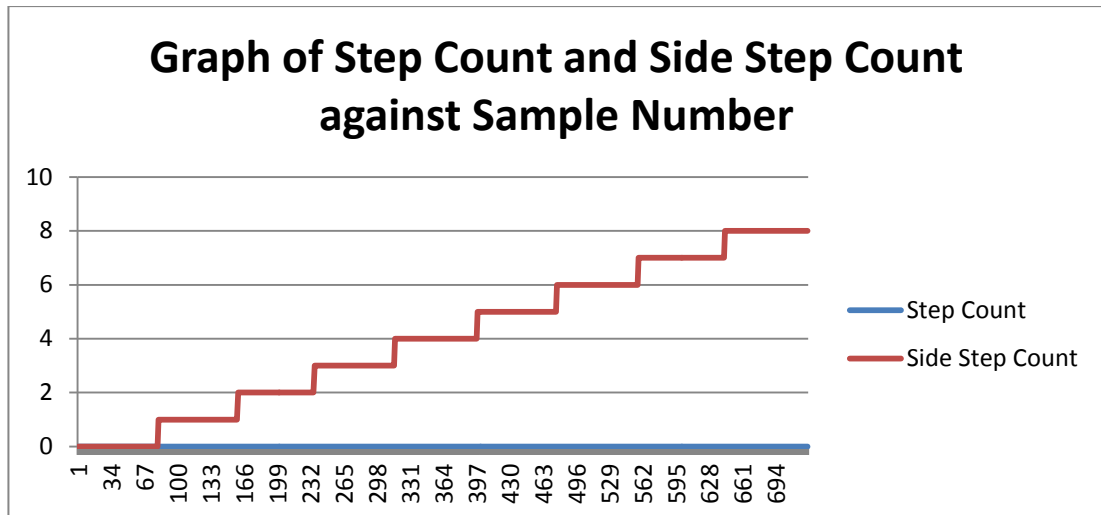


Figure 5.10: Result of Side Step Count with False Positive Error removed.

From the results collected, we can see that the false positive error is eliminated as user performs 8 side steps and the device is able to give correct count of both side steps and forward steps.

5.1.4 Combination Differentiation

Given that both conditions are able to detect individual leg gestures effectively, we need to combine both leg gestures to ensure effective combination recognition that does not give false positive gesture recognition. User performs either two of the gestures in a single experiment to test the effectiveness of the condition. The results collected are shown in Figure 5.11.

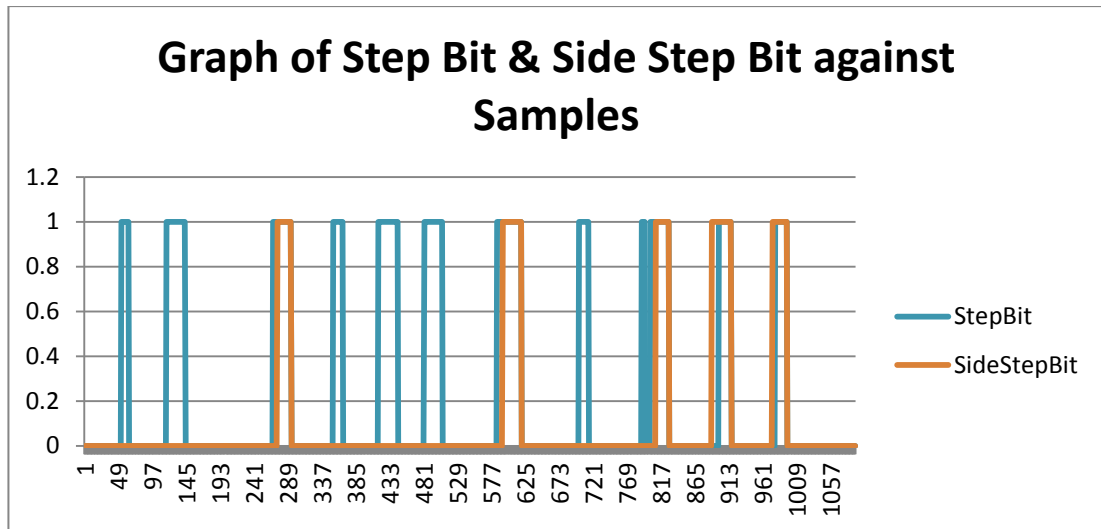


Figure 5.11: Results of combined gestures.

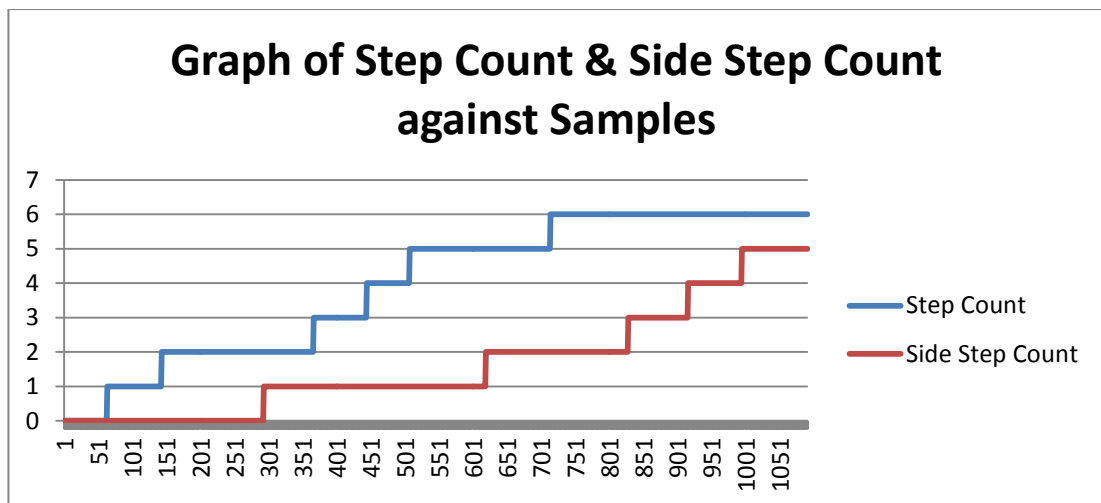


Figure 5.12: Step Count and Side Step Count of combined gestures.

In Figure 5.11, user performs 2 forward steps followed by 1 side step, 3 forward steps, 1 side step, 1 forward step and 3 side steps. Thus a total of 6 forward steps and 5 side steps performed in mixed combination. The algorithm is able to detect accurately as shown in Figure 5.12. Although there are mixed bits detected before, after, and in between the Side Step Bit the algorithm is able to disregard them and give accurate information of which gesture is actually performed.

An empirical experiment is conducted on 4 different users to test out the accuracy of the measurements, as different users have different leg gestures during

forward steps and side steps which might cause the results to vary. The results are shown from Figure 5.13 to Figure 5.33. Table 5.1 and Table 5.2 shown below summarizes the results of each different user.

Table 5.1: Forward Step Result of 4 Different Users.

User	Measured Steps	Actual Count	Error Count	Error Percentage (%)
1	42	43	1	2.33
2	56	56	0	0
3	49	50	1	2
4	45	45	0	0
Total	192	194	2	1.03

Table 5.2: Side Step Result of 4 Different Users.

User	Measured Steps	Actual Count	False Positive Error Count	Error Percentage (%)
1	32	32	1	3.13
2	24	24	0	0
3	36	36	1	2.78
4	30	30	0	0
Total	122	122	2	1.64

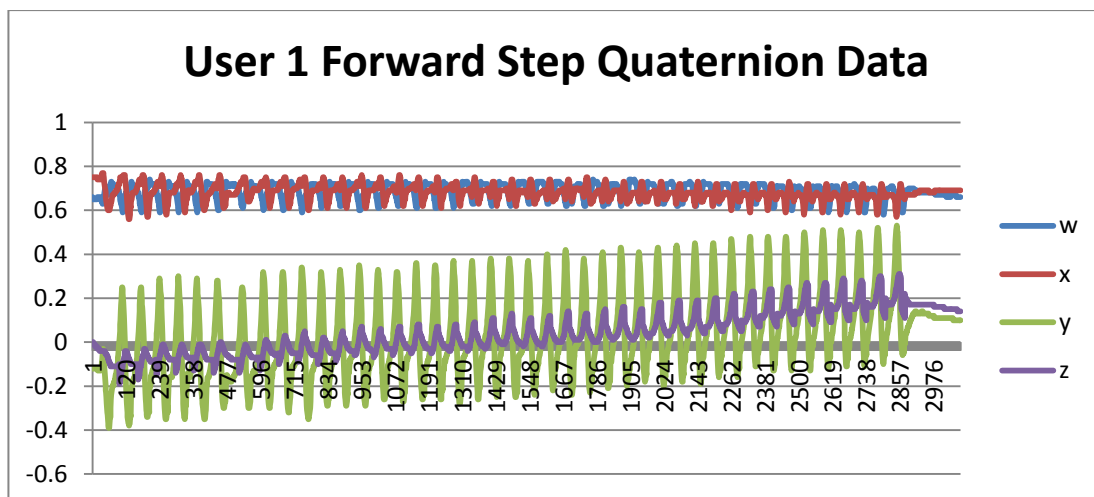


Figure 5.13: Result of User 1 Forward Steps.

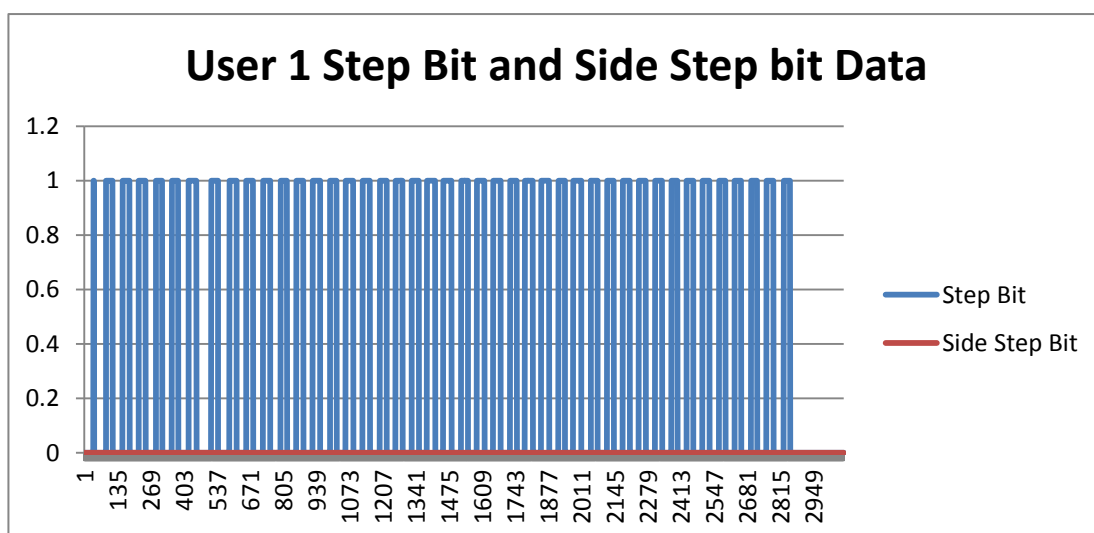


Figure 5.14: Result of User 1 Step Bit and Side Step Bit.

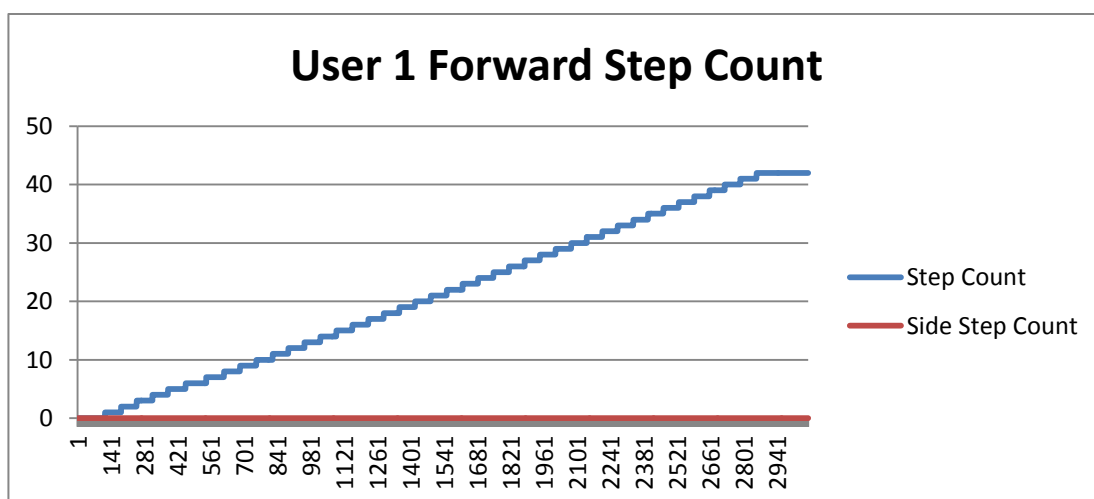


Figure 5.15: Result of User 1 Total Forward Step Count.

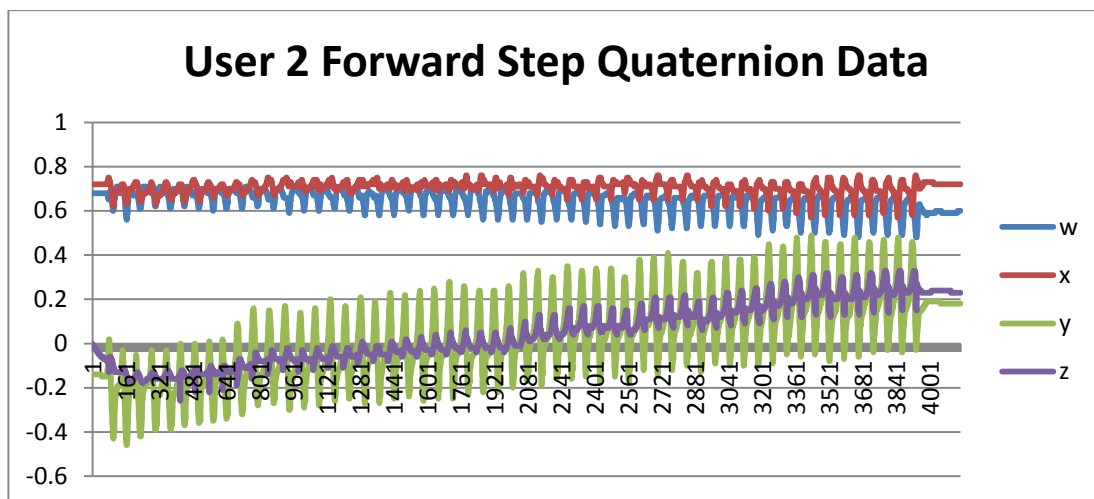


Figure 5.16: Result of User 2 Forward Steps.

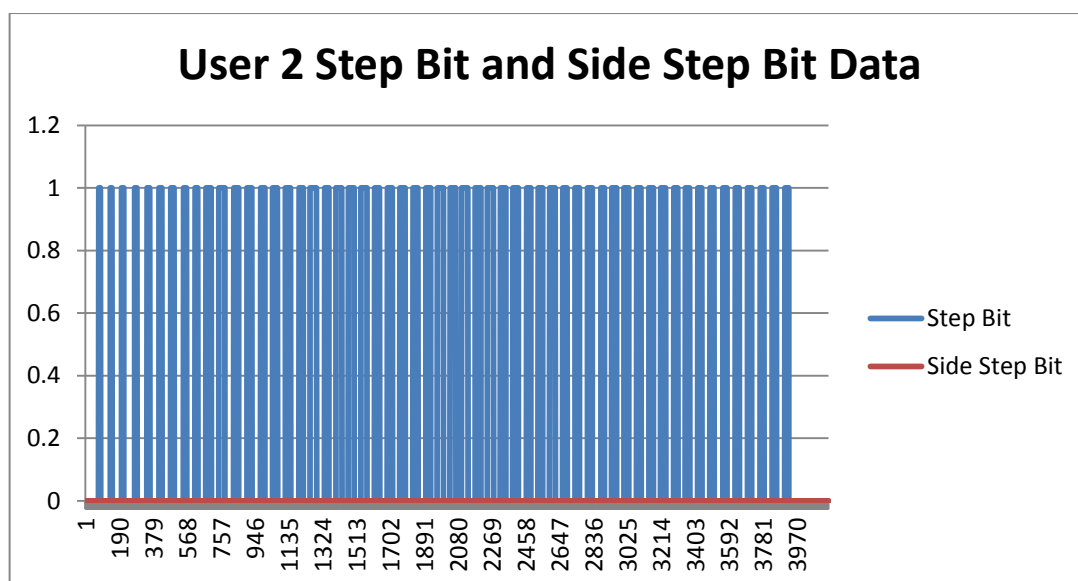


Figure 5.17: Result of User 2 Step Bit and Side Step Bit.

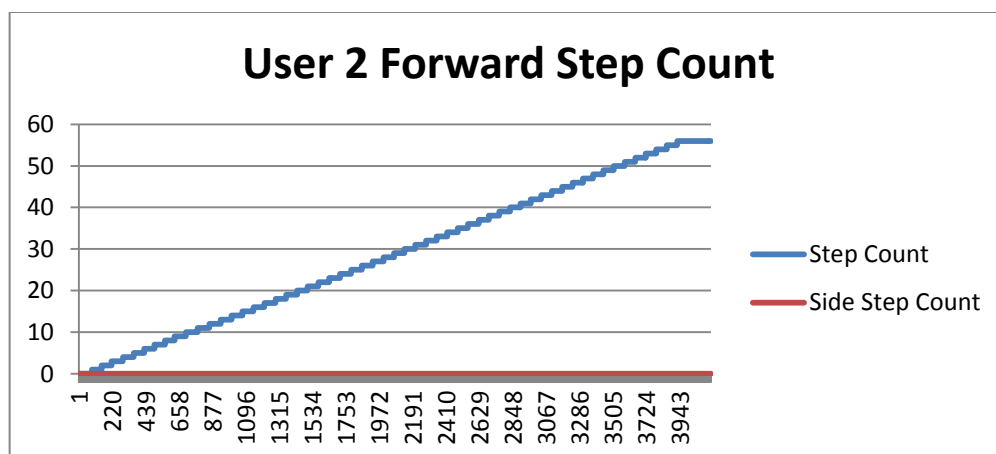


Figure 5.18: Result of User 2 Total Forward Step Count.

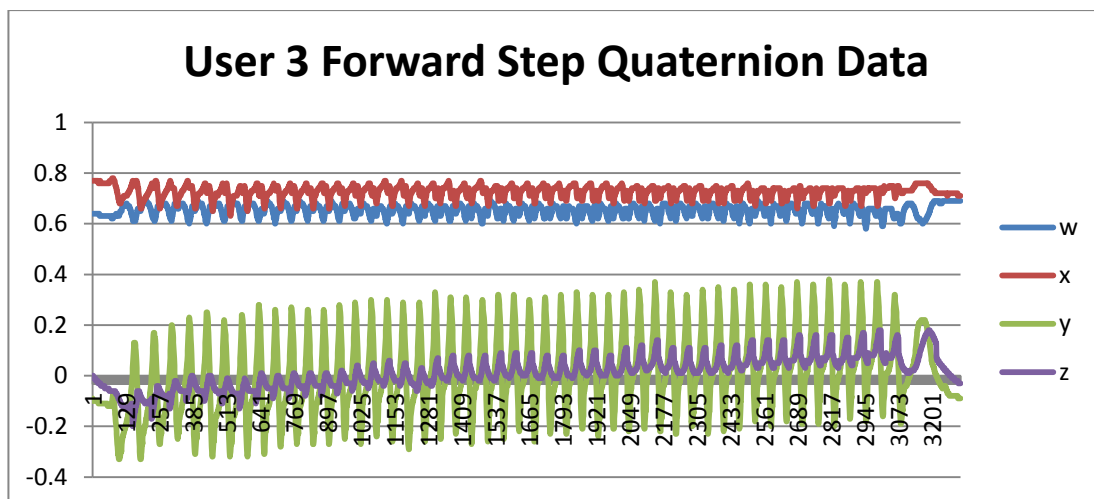


Figure 5.19: Result of User 3 Forward Steps.

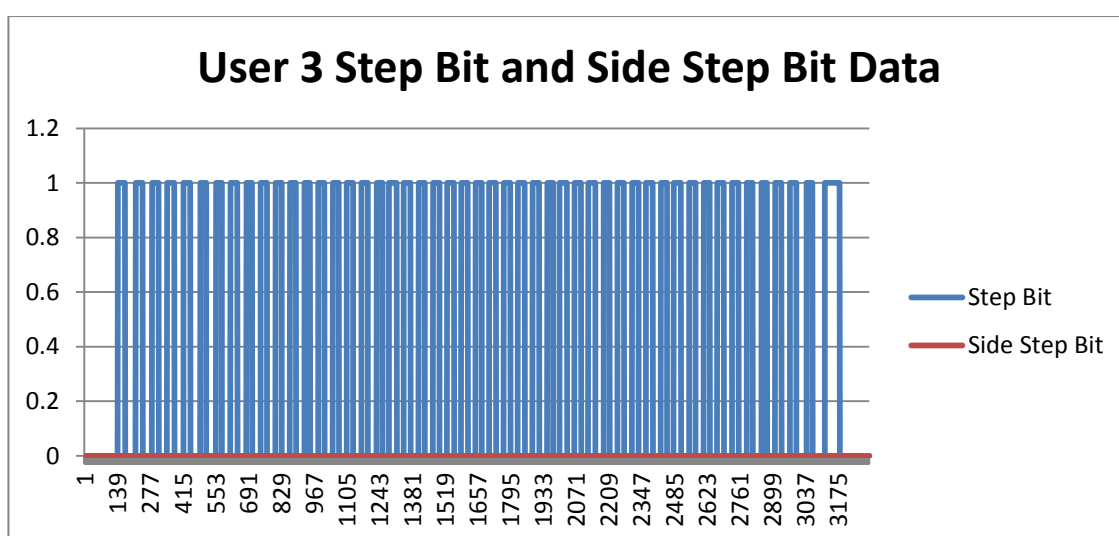


Figure 5.20: Result of User 2 Step Bit and Side Step Bit.

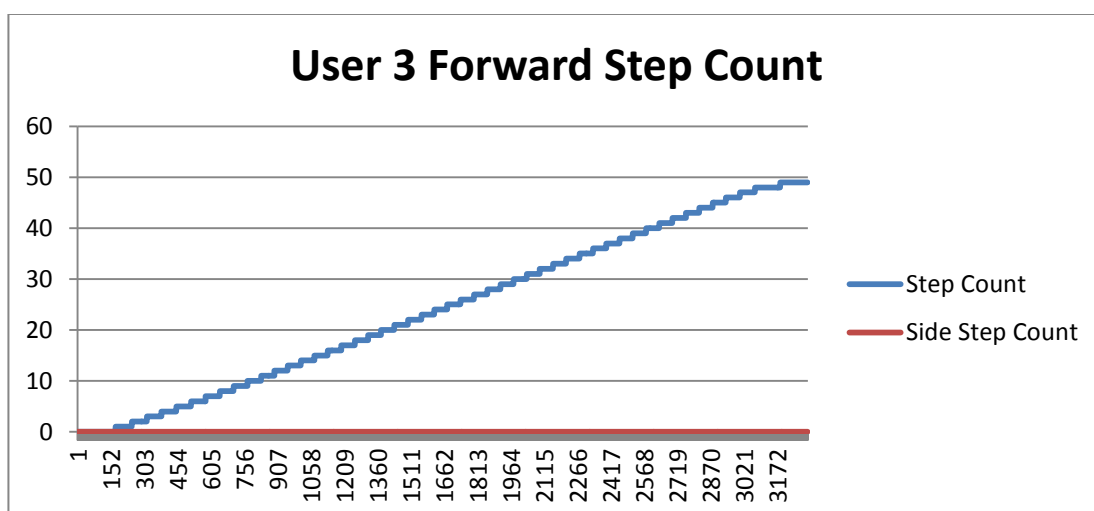


Figure 5.21: Result of User 3 Total Forward Step Count.

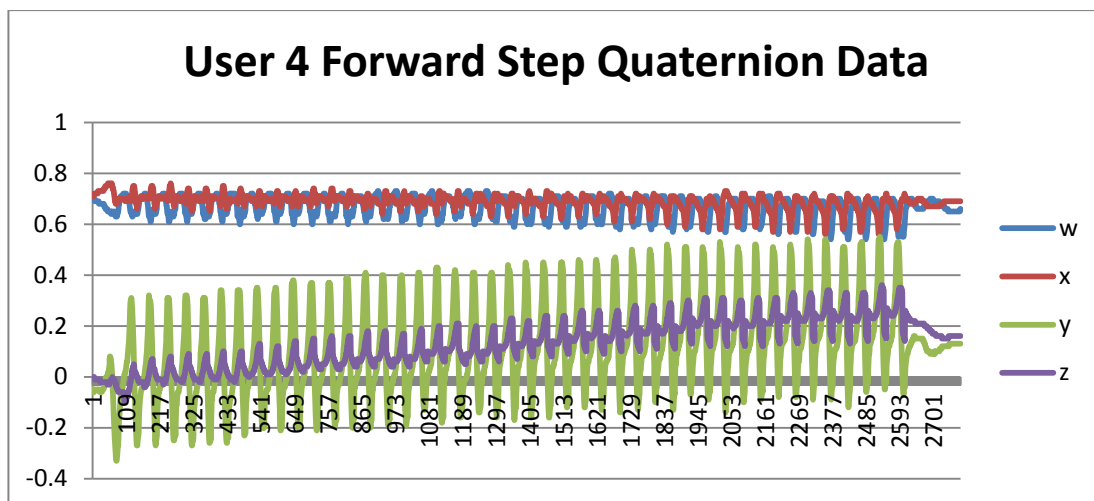


Figure 5.22: Result of User 4 Forward Steps.

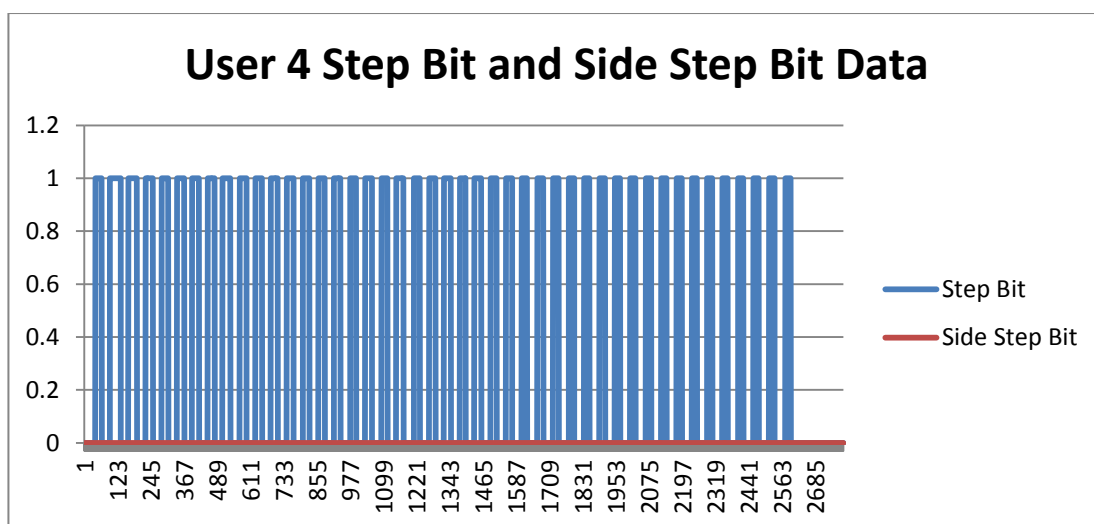


Figure 5.23: Result of User 4 Step Bit and Side Step Bit.

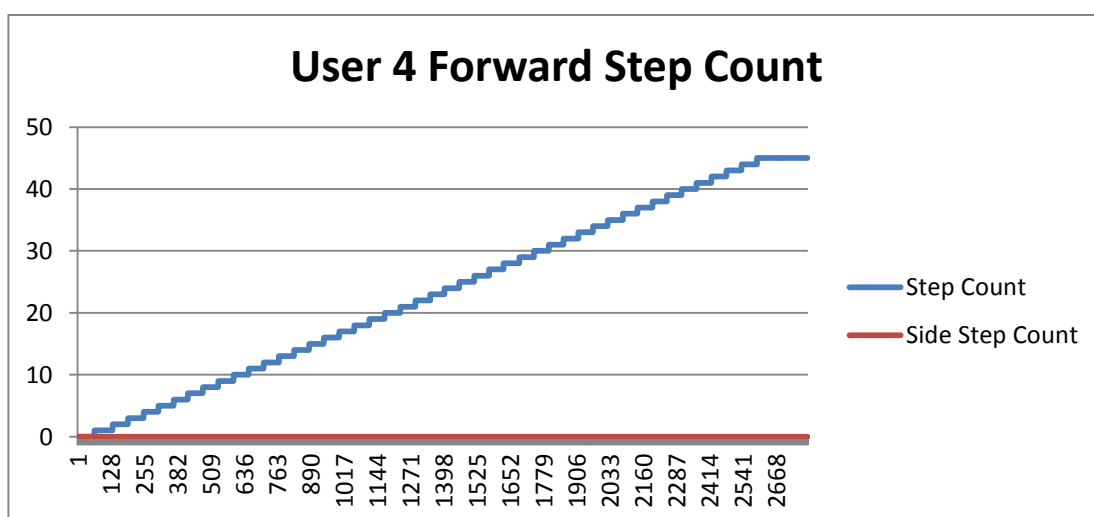


Figure 5.24: Result of User 4 Total Forward Step Count.

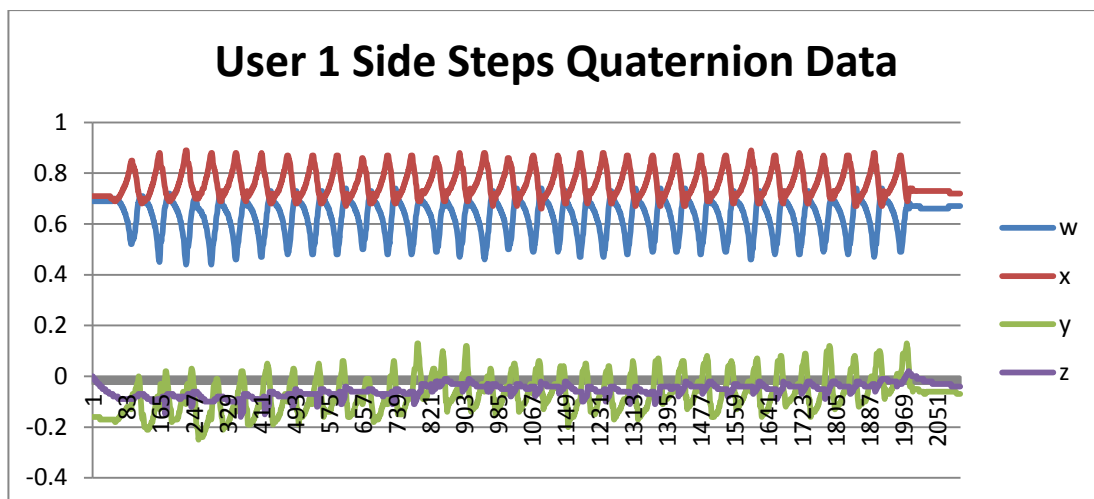


Figure 5.25: Result of User 1 Side Steps.

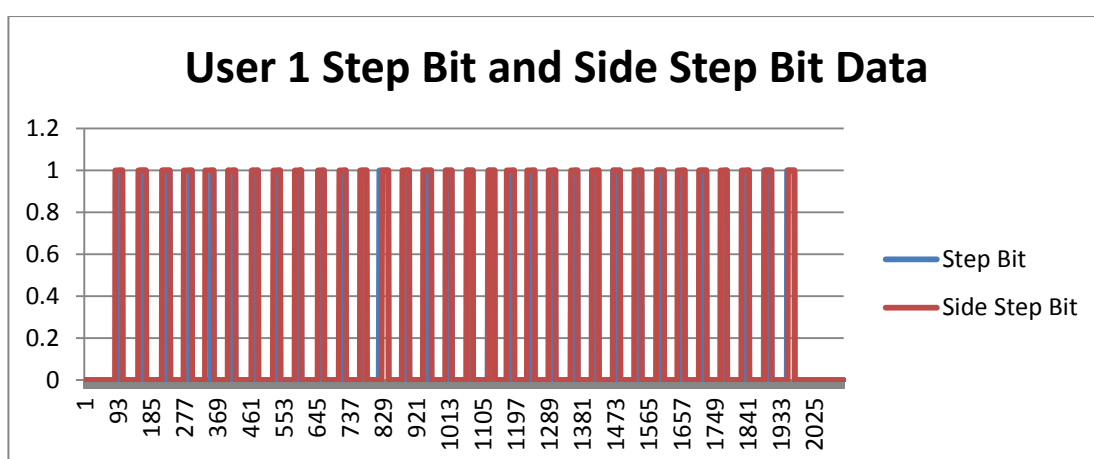


Figure 5.26: Result of User 1 Step Bit and Side Step Bit.

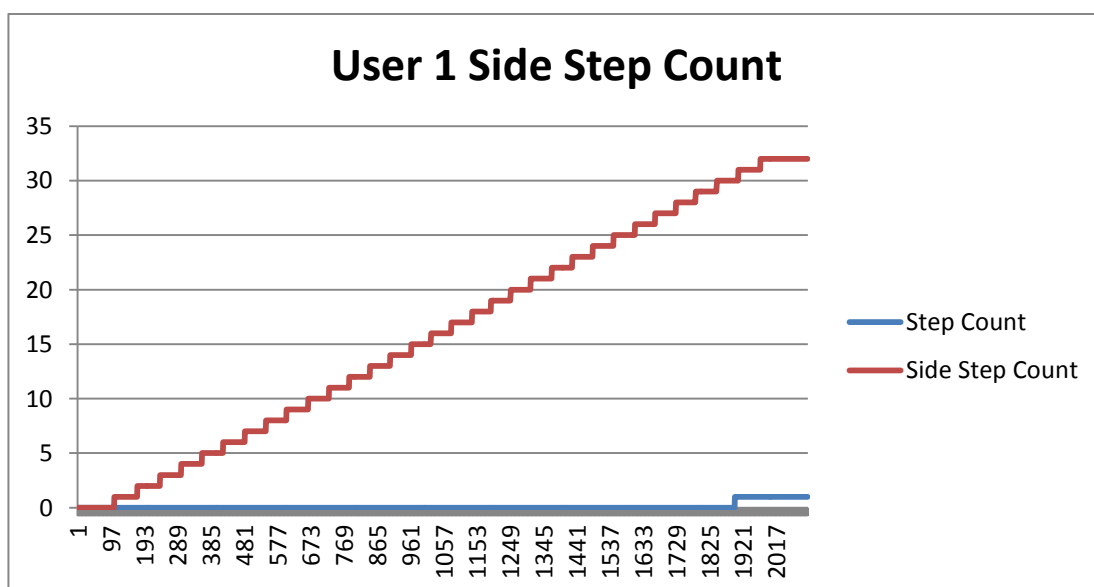


Figure 5.27: Result of User 1 Total Side Step Count.

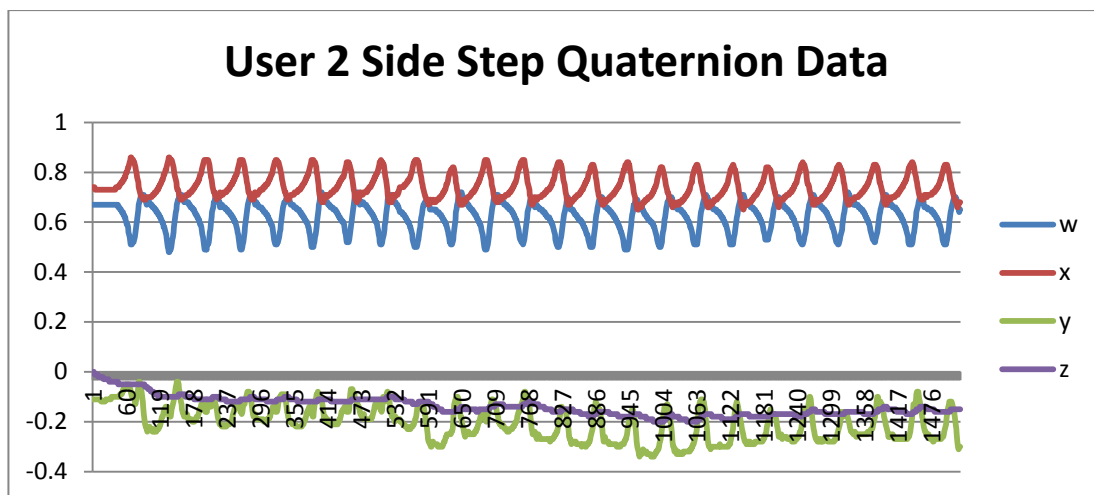


Figure 5.28: Result of User 2 Side Steps.

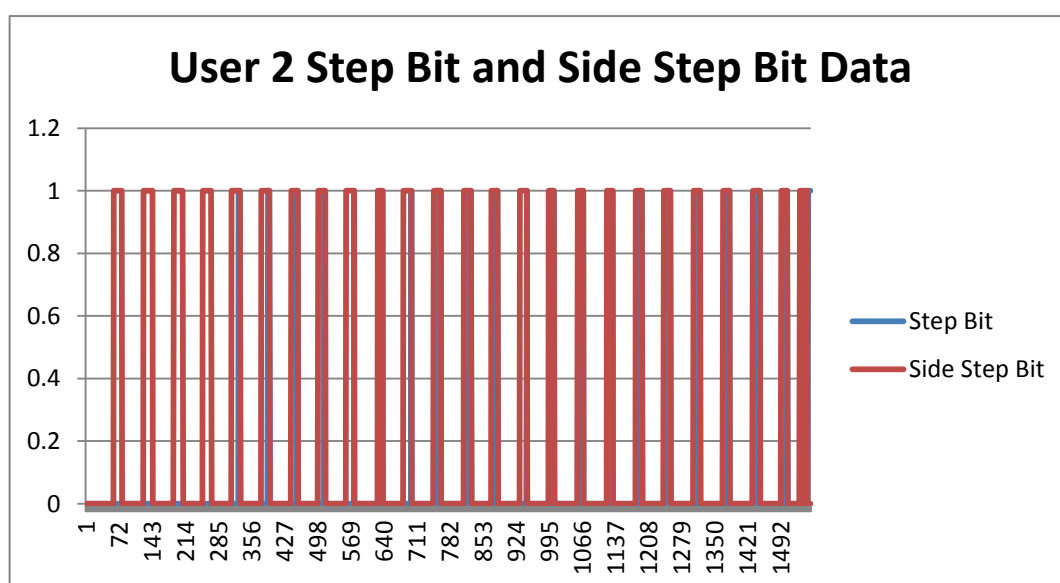


Figure 5.29: Result of User 2 Step Bit and Side Step Bit.

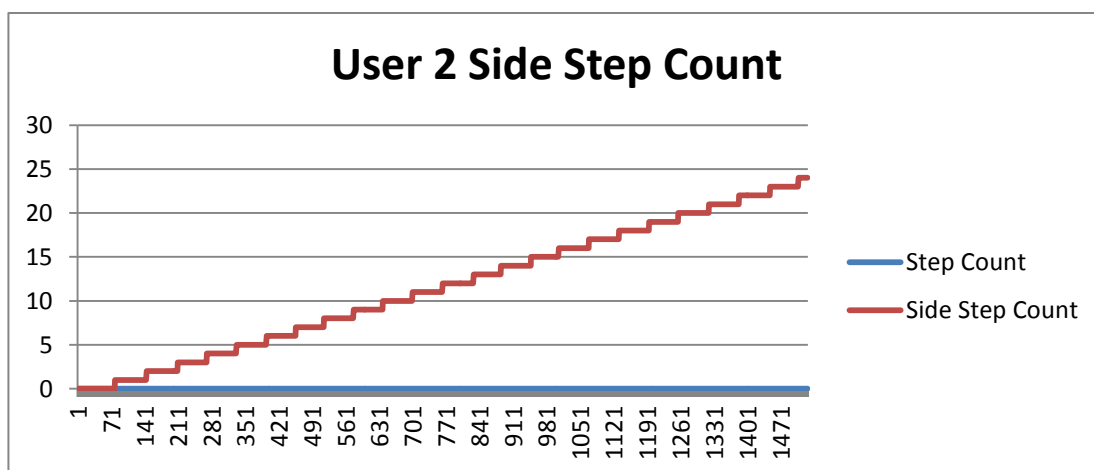


Figure 5.30: Result of User 2 Total Side Step Count.

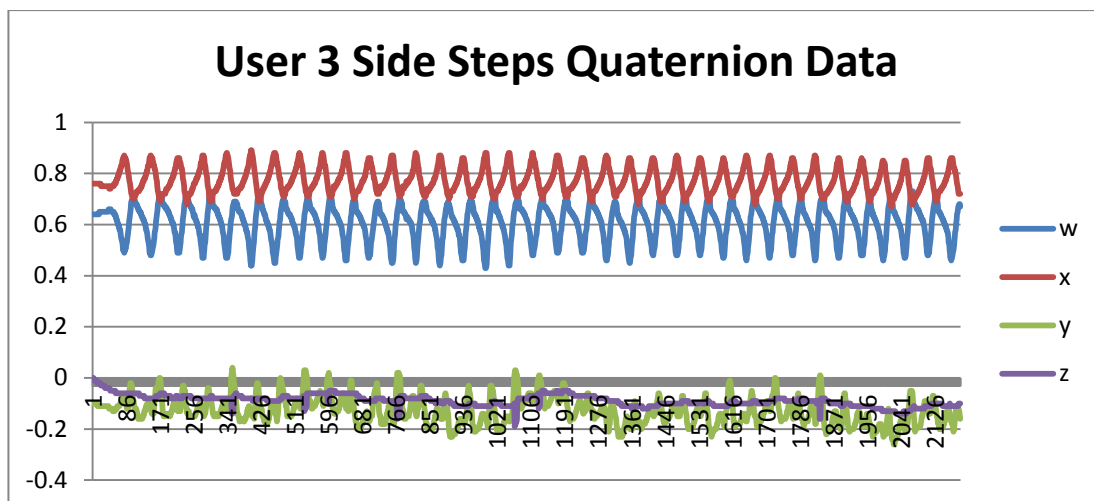


Figure 5.31: Result of User 3 Side Steps.

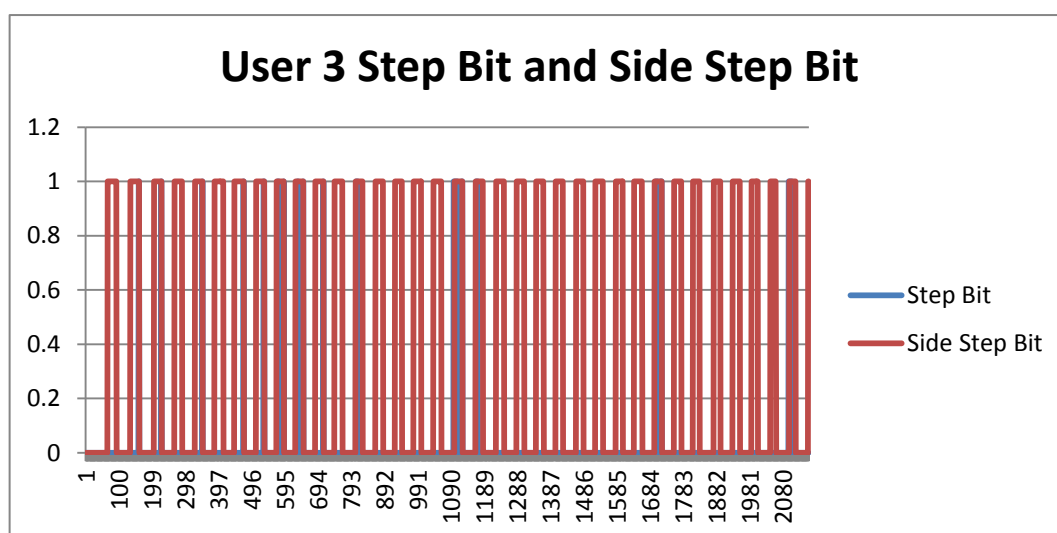


Figure 5.32: Result of User 3 Step Bit and Side Step Bit.

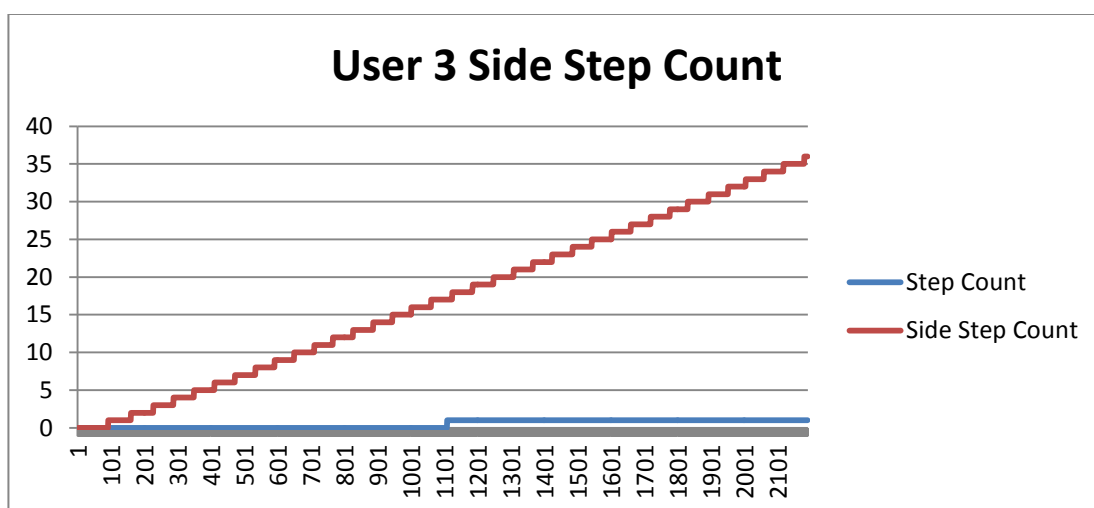


Figure 5.33: Result of User 3 Total Side Step Count.

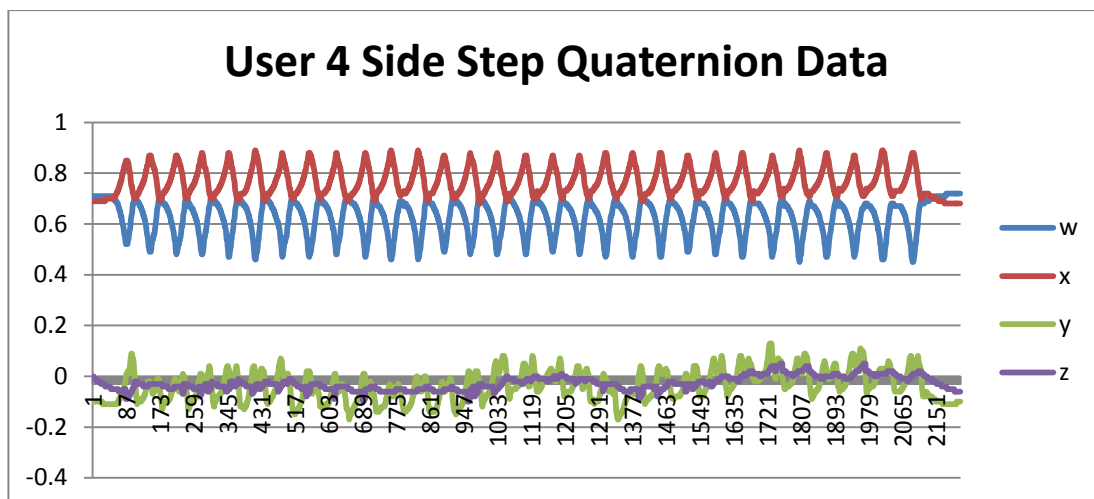


Figure 5.34: Result of User 4 Side Steps.

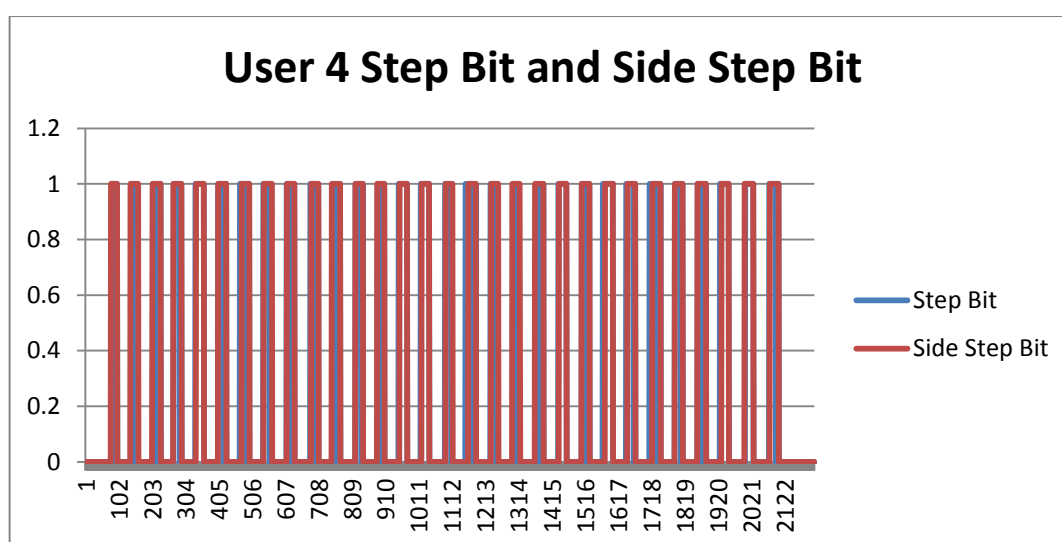


Figure 5.35: Result of User 4 Step Bit and Side Step Bit.

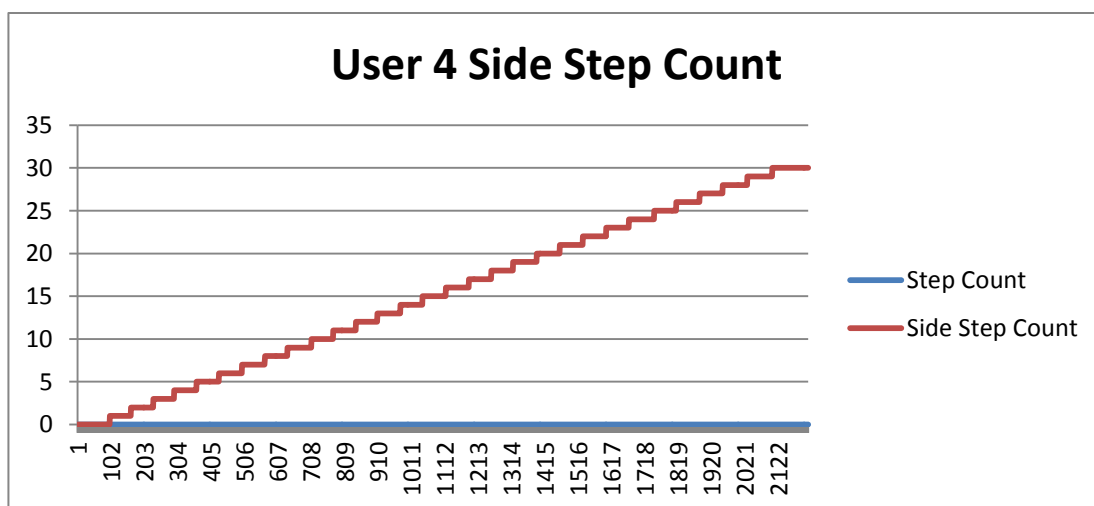


Figure 5.36: Result of User 4 Total Side Step Count.

5.2 Distance Calculation

Once the device is able to differentiate between forward and side steps using quaternion, the distance travelled for each individual leg gestures should be calculated using the raw accelerometer. This function can also serve as an input to offline route map tracking for navigation purposes. The quaternion measures the change in orientation of the sensor, which can be used to detect the user's leg gesture while the accelerometer measures the change in gravity of the sensor itself. This information can be used to determine the users speed and distance travelled.

5.2.1 Forward Step Distance

As mentioned earlier in **Chapter 3.3**, the compensated accelerometer model will be used to determine the user's true acceleration without gravity component. This allows the calculation by simply obtaining the time elapsed from the start of a forward step and the end of the step. The start of a forward step is when the user lifts his leg vertically and the end of the step is where the foot lands on the ground.

The distance that is measured here is the horizontal distance the foot travel, which is also the distance travelled by the user. Thus the accelerometer axis that we are interested in is the X axis. A sample of the data collected for the X axis is collected from performing 3 forward steps and the result is shown in Figure 5.37.

From the result we can see there are 3 distinguishable signals upon performing 3 forward steps. As we analyse each individual forward step signal, we can see that the accelerometer value (in bits) increases positively and suddenly drops to a negative maximum and returns to a value close to zero over time. This can be explained as when a forward step is performed, a person usually lifts the leg vertically and moves it horizontally simultaneously, which increases the accelerometer's positive due to acceleration. The signal then drops to a negative maximum is because the forward step is completed and the foot of the user has

landed on ground thus giving a stop to the acceleration in a short span of time. Lastly, the signal dies off over time because no motion is performed during the period.

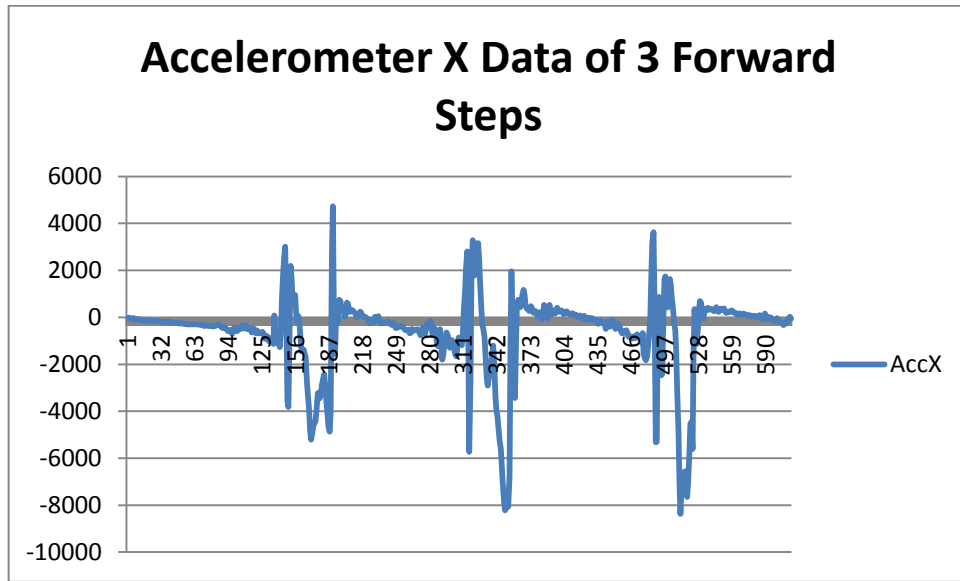


Figure 5.37: Accelerometer Data of 3 Forward Steps.

However, the signal produces negative acceleration which does not make sense in calculation of the distance, because a person can never travel in a negative distance but only in a negative direction. Thus, the accelerometer data must be converted into magnitude form in order to obtain the distance travelled accurately. The equation used to convert the accelerometer axis is shown below. The converted accelerometer data of Figure 5.37 is shown in magnitude form in Figure 5.38.

$$MagnitudeX = \sqrt{(AccelerometerX)^2} \quad (5.3)$$

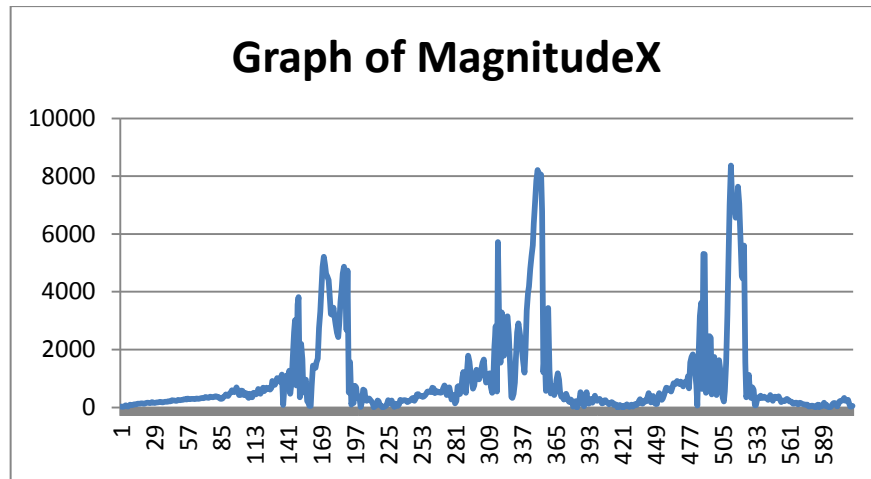


Figure 5.38: Magnitude of Accelerometer X after conversion.

In **Chapter 5.1.2** we explained that the forward step detection algorithm is able to detect the lifting of the leg, and thus, conclude that a forward step has been performed. This is shown in Figure 5.39 where the Step Bit is compared against the magnitude of the accelerometer. The Step Bit is set to a value of 10000 just to show the area it covers. The negative maximum of the accelerometer data due to landing the foot on the ground is not under the area of the Step Bit. The maximum of the data, which is 7240 bits is the point where the foot lands on ground where a high deceleration occurs, we can see that the signal fluctuates for a moment and the magnitude decreases slowly. This information can be used as a condition to determine the point when the forward step has completed and the distance should be calculated at this point.

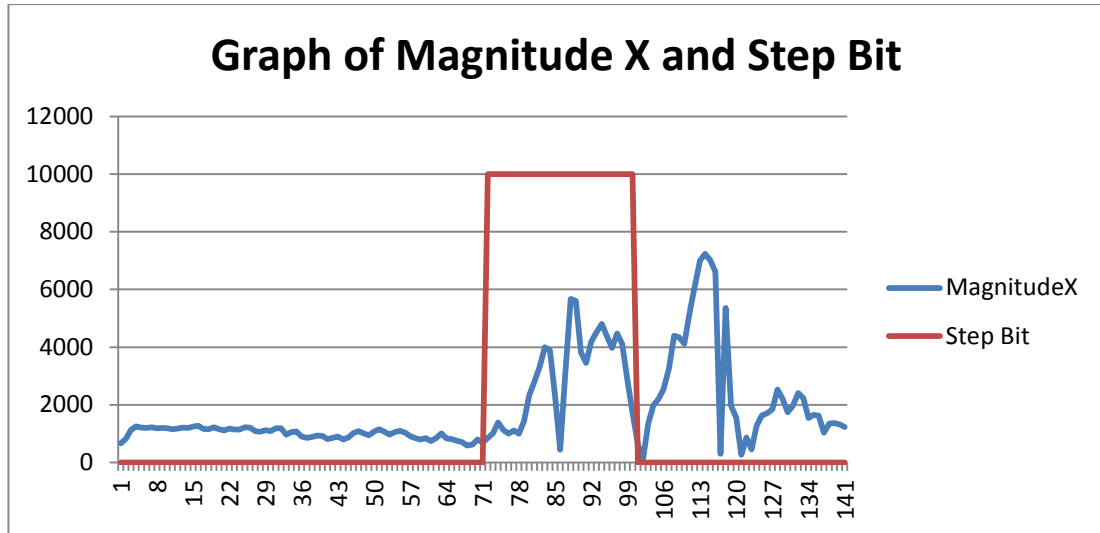


Figure 5.39: Graph of Magnitude X and Step Bit.

5.2.2 Forward Step Distance Threshold and Calculation

Since we know the condition that the accelerometer magnitude must reach a maximum and drops below to certain point to calculate the distance, we need to determine the threshold where the program should stop monitoring the accelerometer magnitude and calculate the distance travelled on that individual step.

The distance is calculated by the equation that distance equals acceleration multiplied by time. Thus, aforementioned in **Chapter 5.1.2**, each collection of the sample requires 0.02s to be collected and the time elapsed for one step will be multiplied from 0.02s to the number of samples collected to calculate the total time of one step. The equation of calculating the individual forward step distance is shown below. The Accumulated Magnitude is the average sum of magnitude until the point where the threshold percentage is reached.

Forward Step Distance

$$= \left(\frac{\text{Accumulated Magnitude}}{32768} \right) \times 2g \times (0.02s \times \text{No. of Samples})$$

(5.4)

The threshold is determined by performing forward step in a straight line of 15m on different threshold percentage by the multiple of 5. This is to find out the range where the threshold percentage gives the least error and from that range we can continue to find the exact threshold percentage that gives the least error. The result of using the threshold percentage by multiples of 5 is shown in the table and figure below.

Table 5.3: Result of Threshold Percentage and Error Produced.

Threshold Percentage (%)	Actual Distance (m)	Measured Distance (m)	Error Percentage (%)
5	15	19.64	30.93
10	15	17.05	13.66
15	15	14.68	2.13
20	15	16.44	9.6
25	15	14.07	6.2
30	15	13.65	9

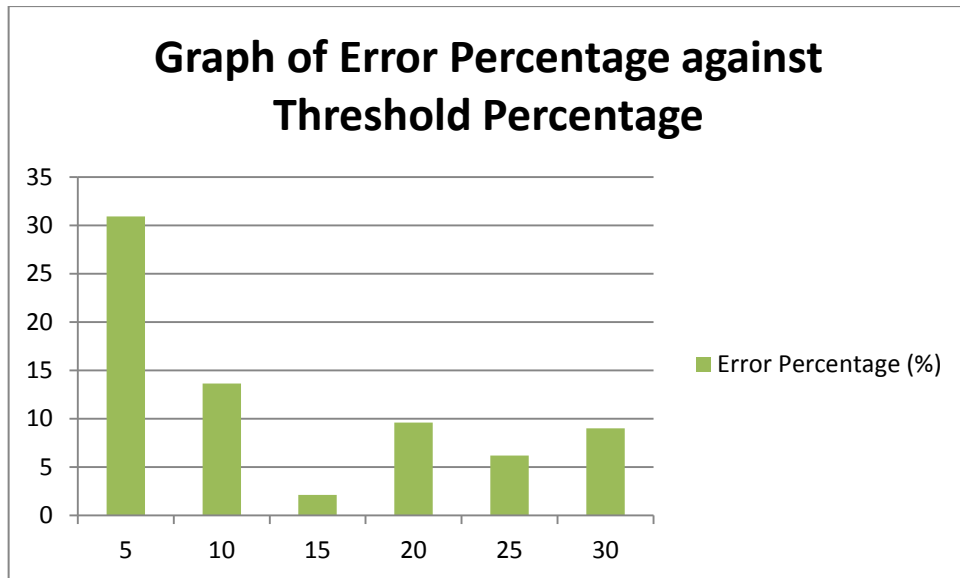


Figure 5.40: Graph of Error Percentage against Threshold Percentage.

From the result produced, we can see that around the threshold percentage range of 15% produced the least error. We continue to find out the range by performing the test for the threshold percentage from 11% till 19%. The result is shown in the table and figure below. The result shows that the most accurate range of the threshold percentage is in the range from 15% till 19% with 19% as the least error of only 0.33%.

Table 5.4: Result of Threshold Percentage and Error Produced.

Threshold Percentage (%)	Actual Distance (m)	Measured Distance (m)	Error Percentage (%)
11	15	16.14	7.6
12	15	16.21	8.06
13	15	14.37	4.2
14	15	13.8	8
15	15	14.68	2.13
16	15	15.46	3.06
17	15	14.66	2.26
18	15	15.18	1.2
19	15	15.05	0.33

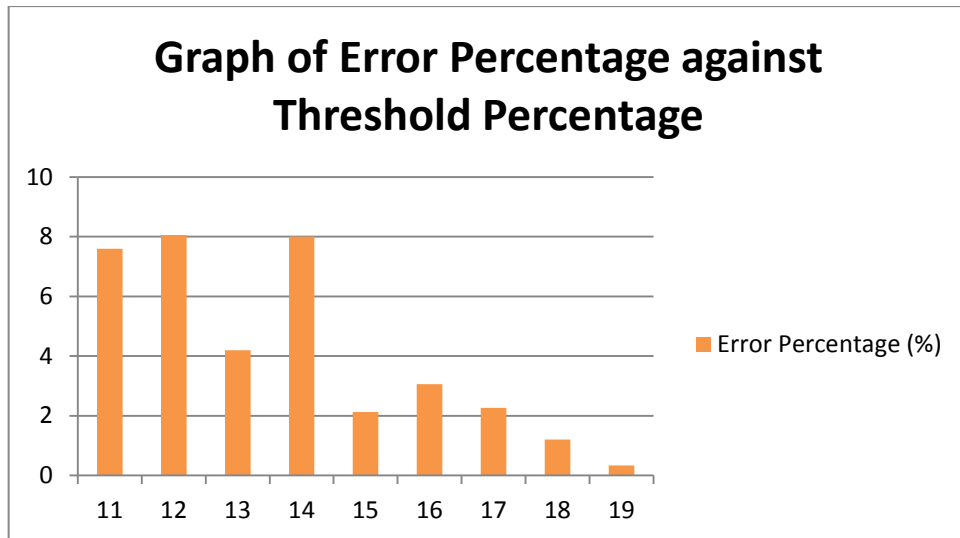


Figure 5.41: Graph of Error Percentage against Threshold Percentage.

Thus we have found out that the threshold range that should be applied in the detection of forward steps should be between 15% to 19%. Since 19% produces the least error percentage, this threshold will be used as the threshold percentage in the program. The accuracy of the distance calculation will be further compared in the empirical experiment in **Chapter 5.3.1**.

5.2.3 Side Step Distance

The general method of calculating a side step distance is the exact same method as in forward step distance. Thus this subsection will be described briefly for similar explanation.

The accelerometer axis that we are interested in determining the side step distance is the Z axis of the accelerometer. A result of 3 side step performed is shown in the figure below. The principal explanation is also similar to the explanation in forward step. The accelerometer value increases due to the leg is lifted and when it lands in the ground it gives a negative maximum which stops the acceleration. The only difference here as compared to the forward step is the direction of the step which happens to be in the Z axis instead of the X axis. The accelerometer data is then converted into magnitude by using the same equation in 5.4 for the purpose of calculating true acceleration. The result is shown in Figure 5.43.

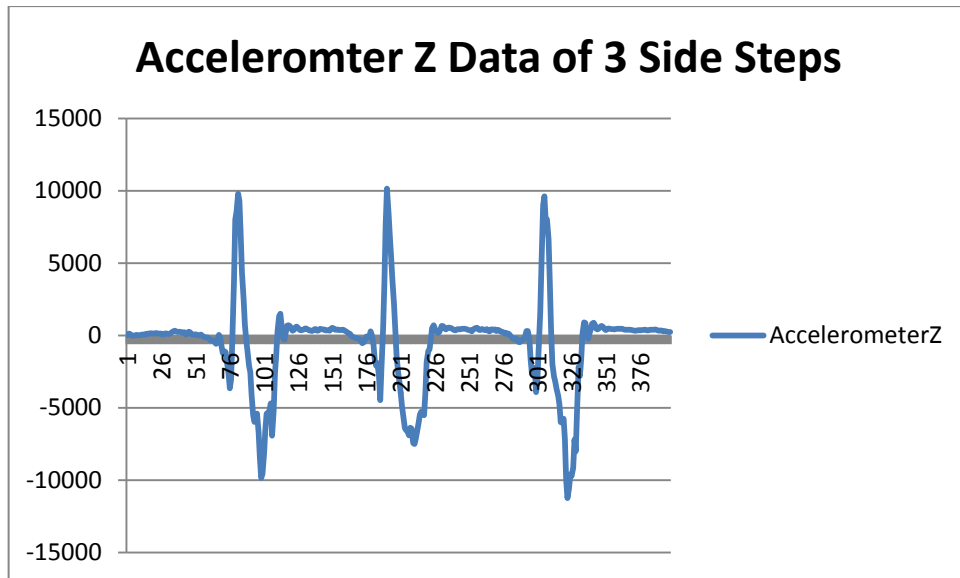


Figure 5.42: Accelerometer Data of 3 Side Steps.

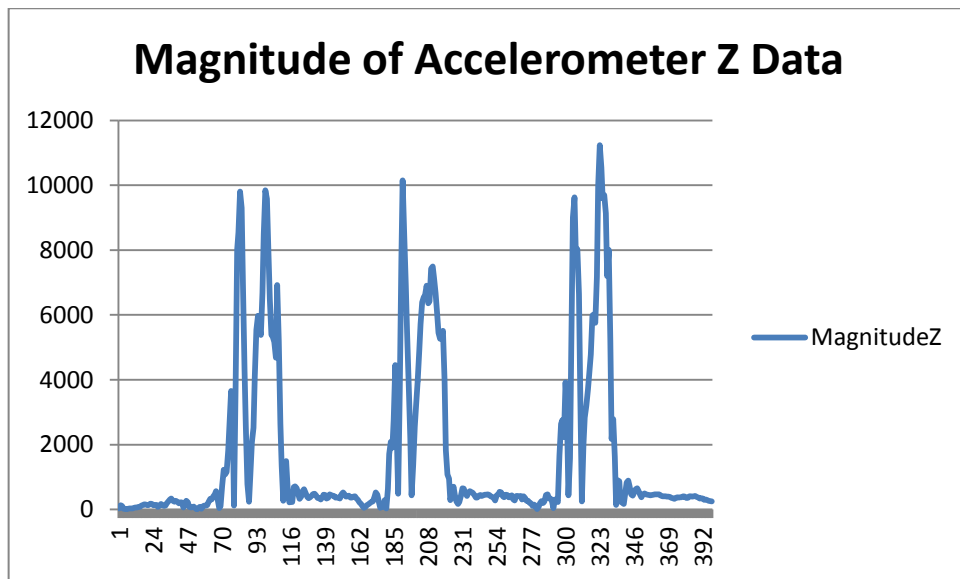


Figure 5.43: Magnitude of Accelerometer Z after conversion.

The conversion of the negative maximum in the accelerometer data is also not covered in the Side Step Bit period and thus by using similar method, upon registering a valid side step, the program will monitor the Z axis of the accelerometer to determine the maximum peak in magnitude Z and determine the individual side step distance once the magnitude drops under the threshold percentage. Figure 5.44 shows the area covered by the side step bit and the peak that follows right after the

side step bit drops to zero, subsequently the signal drops to a small value after the peak. This will be the condition used to extract the features of the signal and calculate the distance.

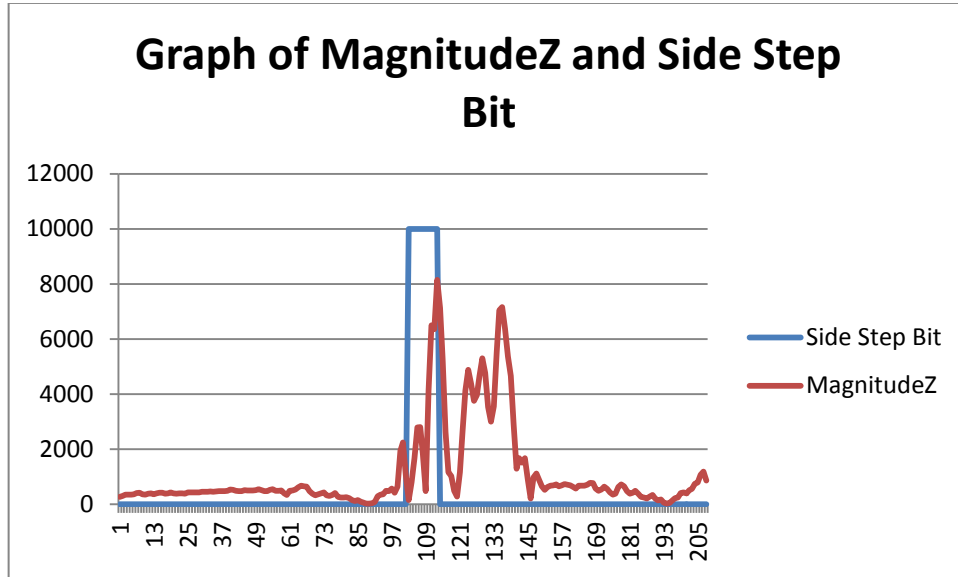


Figure 5.44: Graph of Magnitude Z and Side Step Bit

5.2.4 Side Step Distance Threshold and Calculation

The equation to calculate individual side step distance is similar to forward step distance, whereby the program starts accumulating the accelerometer magnitude when a motion is detected, and if the motion is registered as a valid one the program looks for a maximum peak and terminates the accumulation and calculates the distance once the magnitude drops below a certain threshold percentage. The equation to calculate side step distance is shown below.

Side Step Distance

$$= \left(\frac{\text{Accumulated Magnitude}}{32768} \right) \times 2g \times (0.02s \times \text{No. of Samples})$$

(5.5)

Thus similar tests are performed for side step distance to see which threshold percentage produces the least error percentage and a further test is conducted to investigate which threshold percentage will produce the least error in that range. The table below shows side step tested on different threshold percentage.

Table 5.5: Result of Threshold Percentage and Error Produced on Side Step.

Threshold Percentage (%)	Actual Distance (m)	Measured Distance (m)	Error Percentage (%)
5	15	14.77	1.53
10	15	12.66	15.6
15	15	11.97	20.2
20	15	10.78	28.13
25	15	8.77	41.53
30	15	7.12	52.53

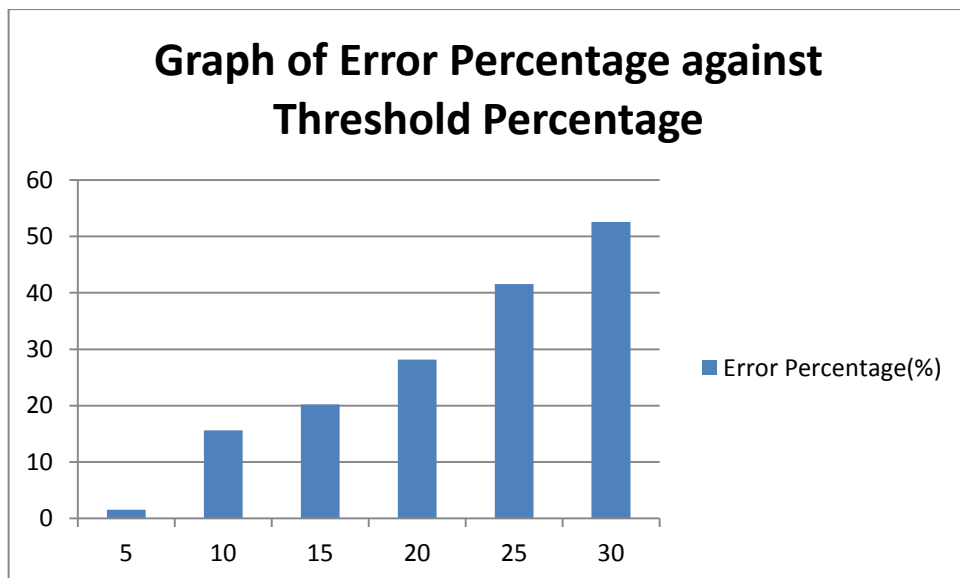


Figure 5.45: Graph of Error Percentage against Threshold Percentage for Side Steps.

From the chart shown above, we can see that the most accurate range is within the threshold of 5% which produces only 1.53% of error and the error percentage continues to increase as the threshold is increased. The test is further

conducted to find the accuracy of the threshold from the range 2% to 9 % to see which percentage produces the least error. The table and figure below shows the result collected. We can see that the range from 2% to 5 % of the threshold percentage produces less error as compared to the others in the figure with 3% producing an error of only 0.93%. Therefore the 3% will be used as the threshold percentage and tested by different users to confirm the accuracy and consistency in **Chapter 5.3.2.**

Table 5.6: Result of Threshold Percentage and Error Produced for Side Step.

Threshold Percentage (%)	Actual Distance (m)	Measured Distance (m)	Error Percentage (%)
2	15	16.14	1.4
3	15	16.21	0.93
4	15	14.37	3.67
5	15	13.8	1.53
6	15	14.68	17.47
7	15	15.46	10.2
8	15	14.66	38.73
9	15	15.18	45.3

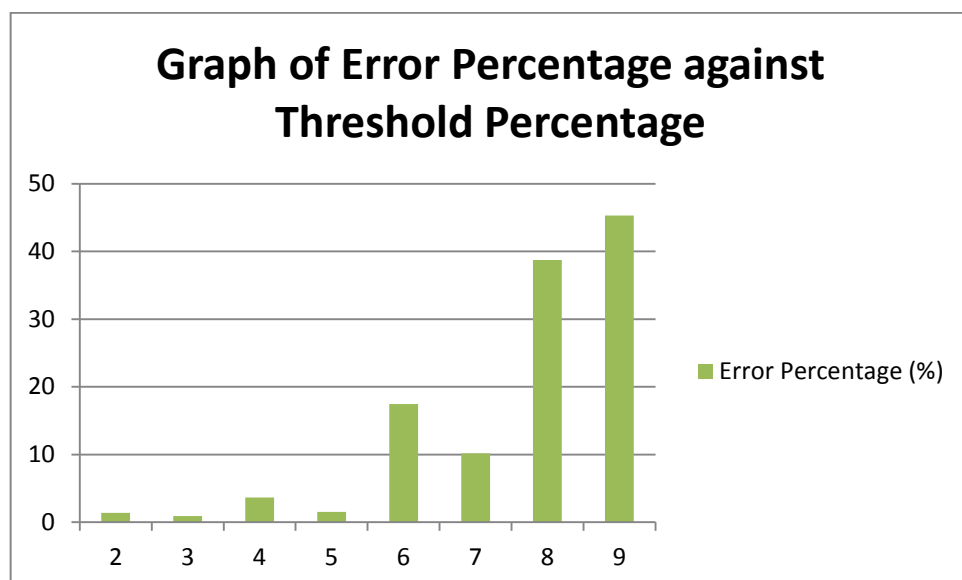


Figure 5.46: Graph of Error Percentage against Threshold Percentage for Side Steps.

5.2.4.1 Side Step Left Issue

The entire series of tests performed above are the side step to the right direction of the user. This is because side step left produces inaccurate result. This is an important issue that should be addressed and explained because it affects the result obtained when user performed side step left.

The figure below explains the phenomena when user side step left as compared to side step right when the sensor device is attached to the users left ankle. When side step right is performed, the right foot is lifted and landed on the ground horizontally before the left foot is lifted and placed on the ground. During this motion, the body weight of the user will sway towards the right foot and the right foot is used to support the user's body weight. This reduces the gravitational force on the left foot when it is moved right, thus producing more acceleration as compared to side step left. As for side step left, the body sway towards the left foot first and the body weight is supported by the left foot before the right foot is moved left. Thus this creates more gravitational force on the sensor device and less acceleration will be produced because of the weight to be supported by the left foot. The results of the side step left is shown in **Chapter 5.3.3** to show that this phenomena results in the side step distance increases by an error percentage of more than 50%.

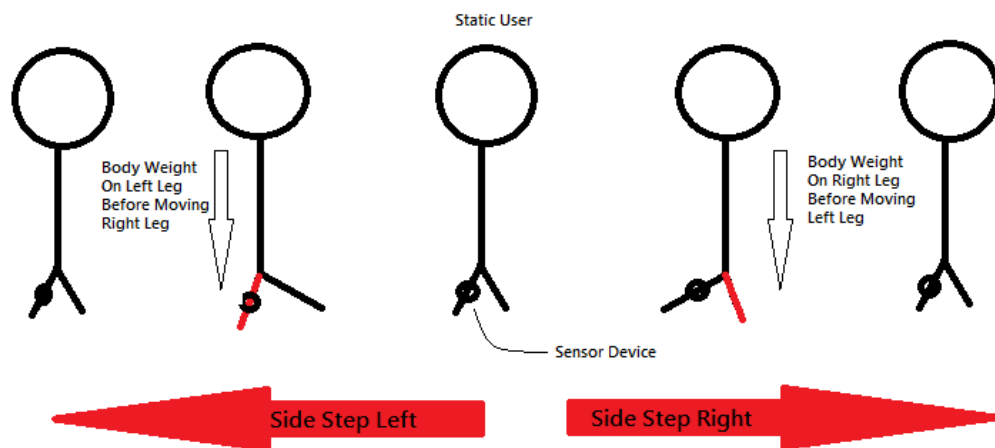


Figure 5.47: Mechanism of User Side Step Left and Right

5.3 Empirical Test and Comparison with Commercial Product

In this section, the developed prototype is tested and compared against commercial product available in the market. The commercial product we have obtained in this project is Fitbit Flex and its function is explained in **Chapter 2.1.4**.

The empirical test performed here is to request 5 male and 5 female to perform forward steps, side steps left and side steps right for 15m. The objective here is to confirm the threshold found in **Chapter 5.2** (19% for forward step and 3 % for side step) is accurate and consistent to be used. The results are separately shown in the sections below.

5.3.1 Forward Step Distance

In this test, 10 users perform forward step in a straight line for 15m. They are required to perform it the same time while wearing FitBit Flex and our sensor system. This is to ensure that the results produced are comparable. The results are shown in the table below. Due to large amount of data collected, it is impossible to show every single graph. Thus only the first two users result will be shown and the rest will be shown in the additional files in the CD.

Table 5.7: Result Comparison of Sensor Device against FitBit Flex for Forward Step Distance.

User	MPU 6050			FitBit Flex		
	Measured Steps	Measured Distance(m)	Error Percentage (%)	Measured Steps	Measured Distance(m)	Error Percentage (%)
1	11	15.76	5.07	11	20	33
2	10	15.0	0	8	10	33
3	10	14.38	4.13	8	10	33
4	10	15.09	0.6	9	20	33
5	10	15.23	1.53	10	10	33
6	10	13.37	10.87	8	10	33

7	10	13.49	10.07	9	10	33
8	11	15.35	2.33	10	20	33
9	10	16.68	11.2	11	20	33
10	11	15.41	2.73	10	10	33
		Average Error (%)	4.86	Average Error (%)		33

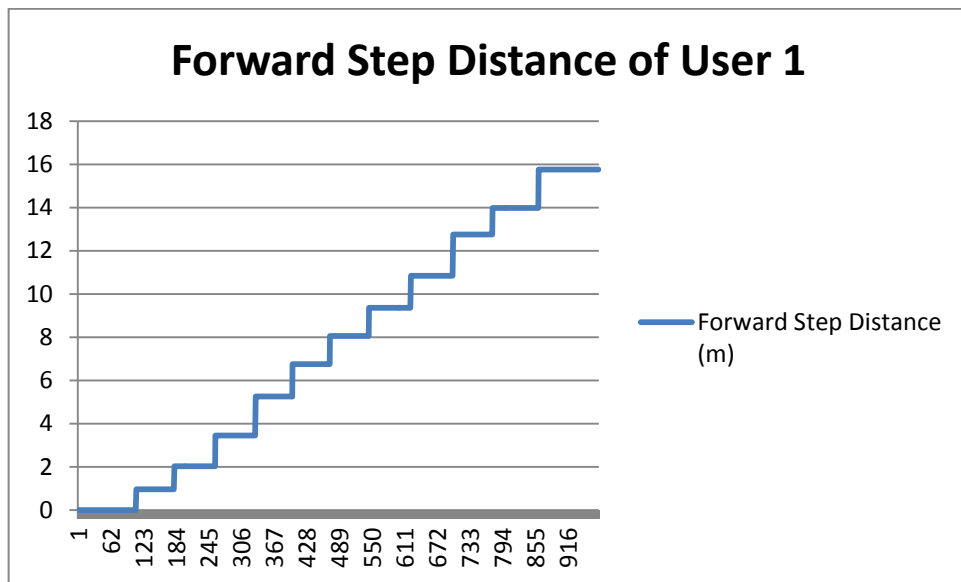


Figure 5.48: Forward Step Distance of User 1.

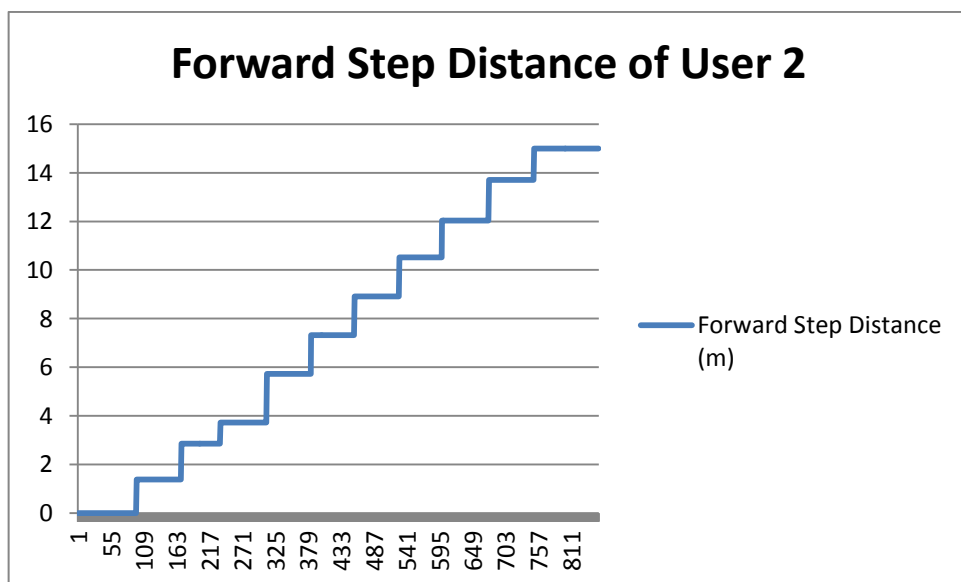


Figure 5.49: Forward Step Distance of User 2.

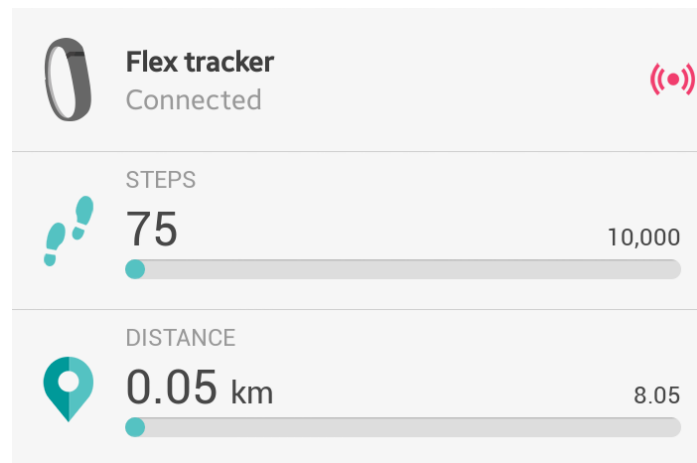


Figure 5.50: Forward Step Distance of User 1 before Testing Fitbit Flex.

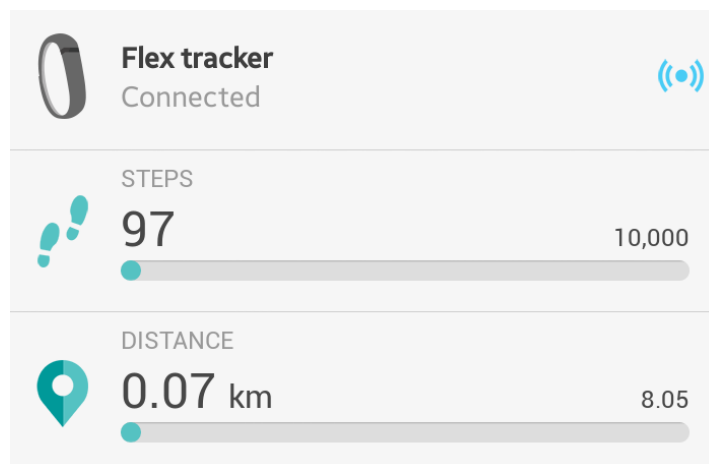


Figure 5.51: Forward Step Distance of User 1 after Testing Fitbit Flex.

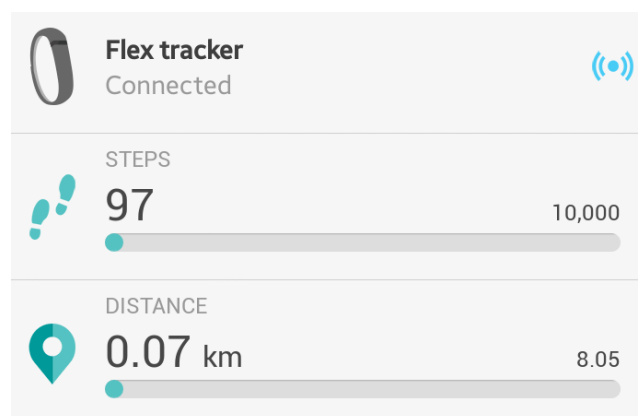


Figure 5.52: Forward Step Distance of User 2 before Testing Fitbit Flex.

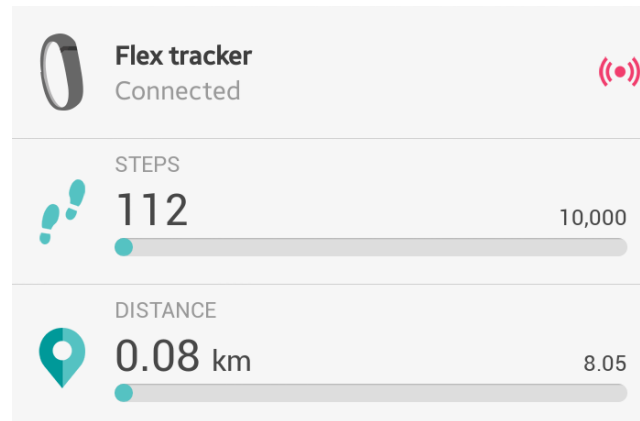


Figure 5.53: Forward Step Distance of User 2 after Testing Fitbit Flex.

5.3.2 Side Step Right Distance

Similar test is also performed for side step distance right and the results are shown below.

Table 5.8: Result Comparison of Sensor Device against FitBit Flex for Side Step Right Distance.

User	MPU 6050			FitBit Flex		
	Measured Steps	Measured Distance(m)	Error Percentage (%)	Measured Steps	Measured Distance(m)	Error Percentage (%)
1	23	15.32	2.09	14	20	33
2	19	15.11	0.73	14	20	33
3	22	15.65	4.15	18	30	50
4	19	15.58	3.72	16	10	33
5	21	14.94	0.40	15	10	33
6	18	15.17	1.12	14	20	33
7	17	14.8	1.35	17	10	33
8	16	15.62	3.97	18	20	33
9	20	14.98	0.13	14	20	33
10	21	15.61	3.9	14	30	50
	Average Error (%)		2.16	Average Error (%)		36.4

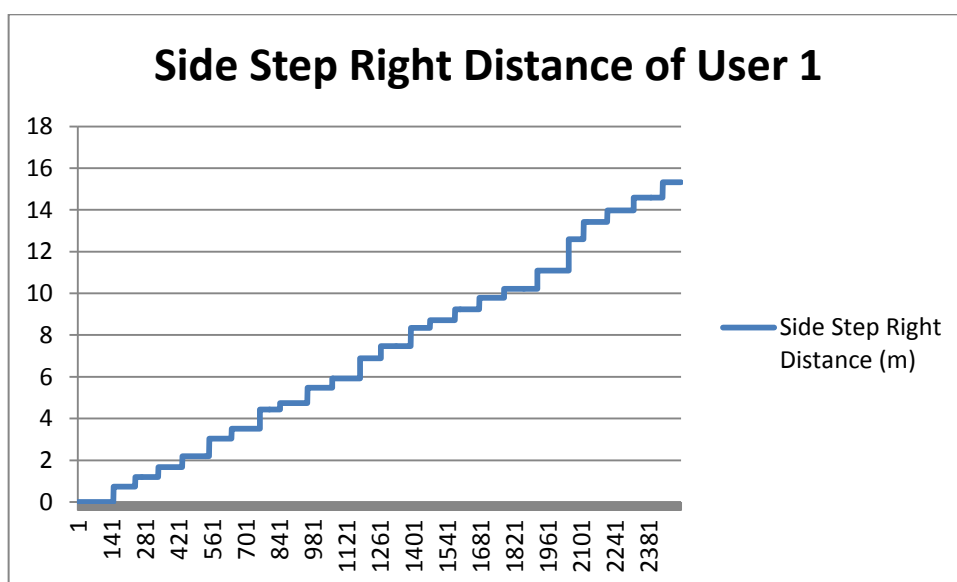


Figure 5.54: Side Step Right Distance of User 1

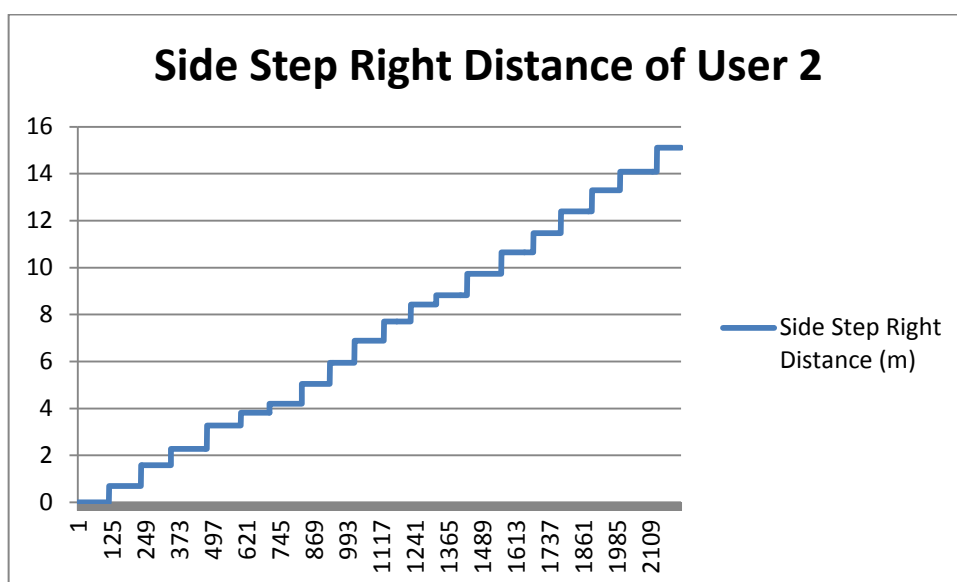


Figure 5.55: Side Step Right Distance of User 2.

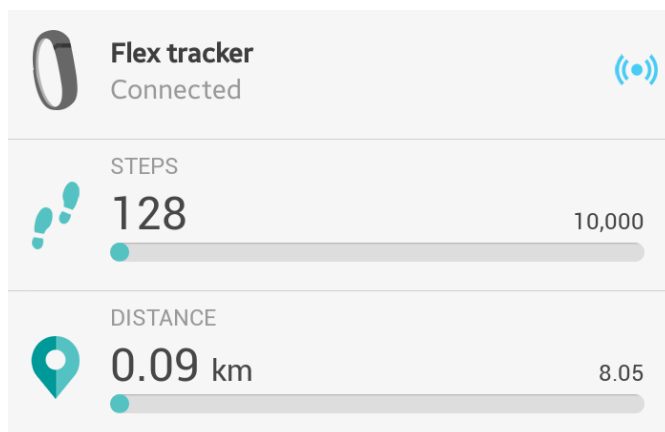


Figure 5.56: Side Step Distance of User 1 before Testing FitBit Flex

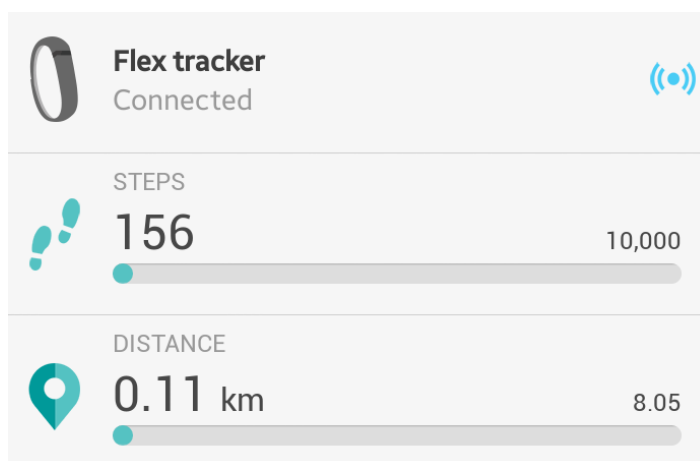


Figure 5.57: Side Step Right Distance of User 1 after Testing FitBit Flex.

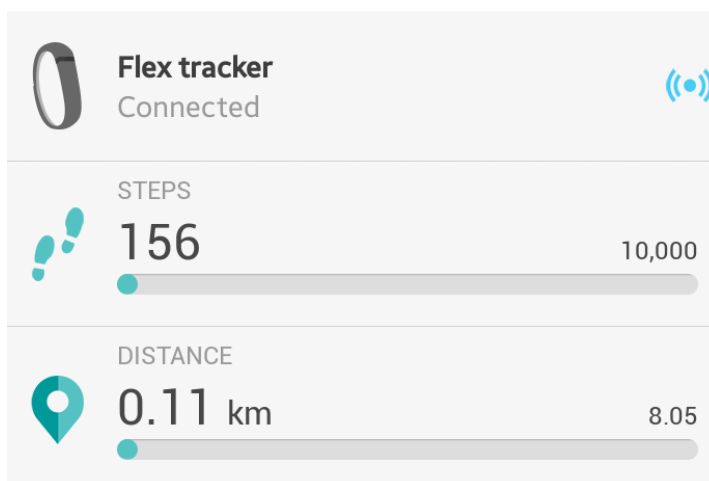


Figure 5.58: Side Step Right Distance of User 2 before Testing FitBit Flex

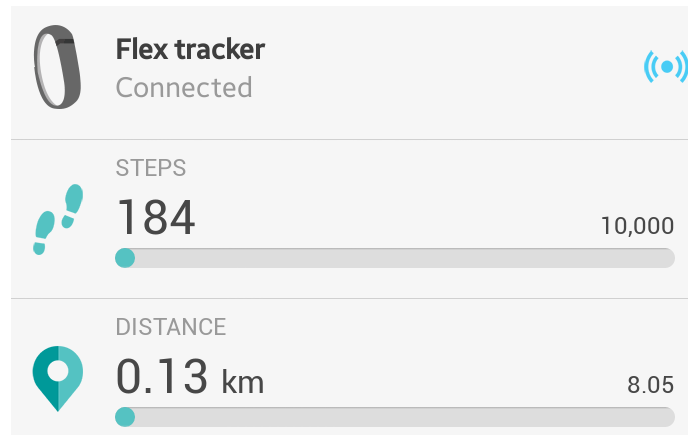


Figure 5.59: Side Step Right Distance of User 2 after Testing FitBit Flex.

5.3.3 Side Step Left Distance

Similar test are performed in determining the accuracy of side step left distance. However the result will not be compared to FitBit Flex because the result already has a large amount of error which was explained in **Chapter 5.2.4**.

Table 5.9: Result Comparison of Sensor Device against FitBit Flex for Side Step Right Distance.

User	Measured Steps	Measured Distance(m)	Error Percentage (%)
1	19	6.88	54.27
2	17	5.06	66.27
3	17	6.57	56.2
4	17	6.14	59.1
5	17	6.2	58.67
6	16	7.21	51.93
7	17	7.49	50.07
8	16	6.39	57.4
9	18	5.77	61.53
10	18	5.76	61.6
Average Error (%)			57.7

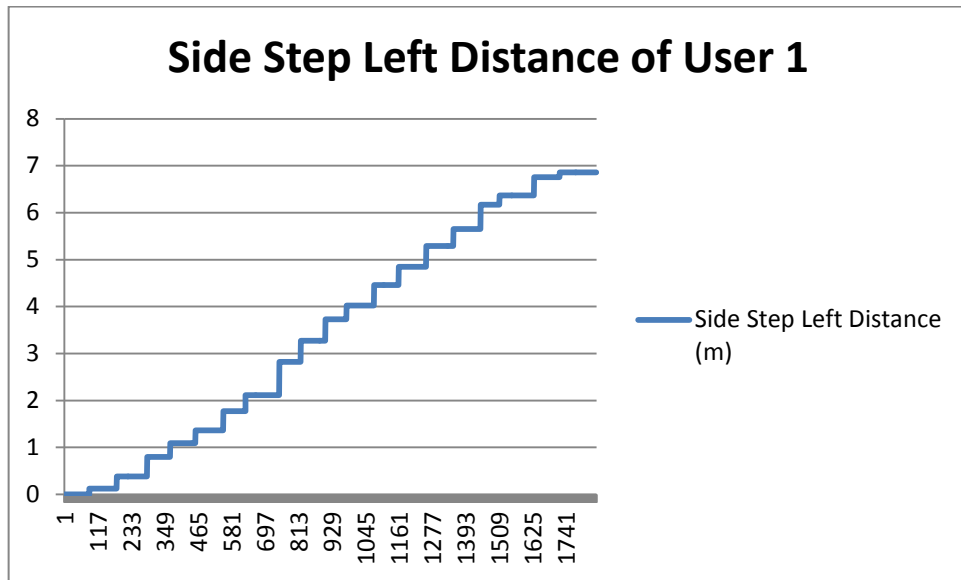


Figure 5.60: Side Step Left Distance of User 1

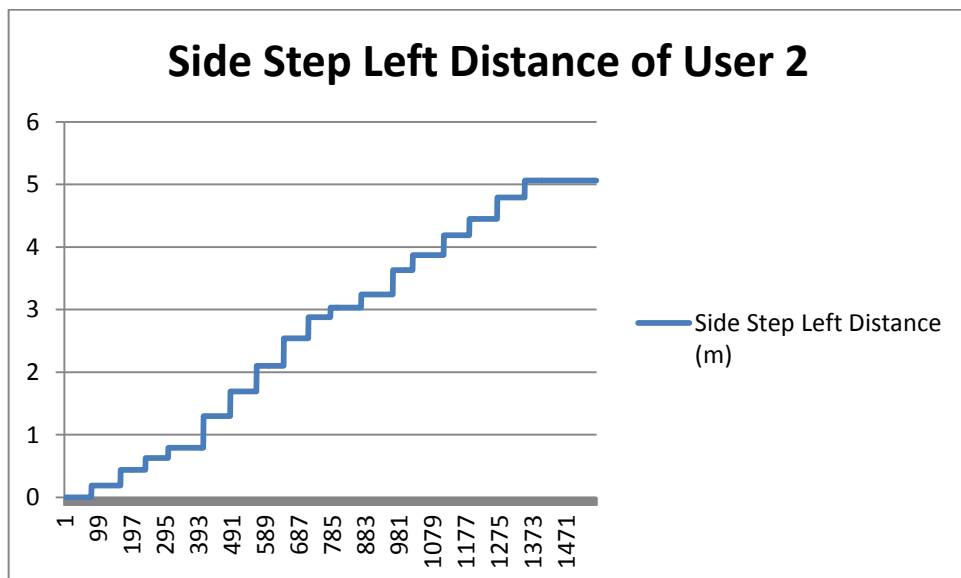


Figure 5.61: Side Step Left Distance of User 2

5.3.4 Summary of Results and Comparison

By comparing the results, we can see that the device developed is much more accurate as compared to FitBit Flex, in terms of forward step distance and also side step right distance. Our device shows an error of only 4.86% while Fitbit Flex shows an error of 33%. For side step right distance, our device recorded only an average of 2.16% while FitBit recorded 36.4%. The reason for this is that FitBit Flex does not use sensors to detect the distance travelled for each leg gesture. Fitbit Flex only uses the accelerometer to determine the step count, where the step count is synchronised into the mobile phone and the distance travelled is based on user's height and weight which will be fixed. Thus if a user travels in a random distance for each step counted, FitBit Flex will not be able to detect it but is only able to give a fixed distance by multiplying the step counted against the fixed distance that is programmed in the FitBit mobile application.

The advantage of our prototype as compared with FitBit Flex is it's ability to differentiate two different leg gestures where FitBit cannot. The prototype is also able to detect individual step distances that does not depend on user's weight and height while FitBit requires both.

The issues regarding side step left distance which produces the large margin of error could be solved by adding an external magnetometer sensor to the inertial measurement unit. An external magnetometer sensor will be able to detect the direction of the user and if it detects that a side step left has been performed, the 57.7% error percentage in the side step left distance result could be used as a factor to correct the distance.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

In conclusion, the objectives of this project have been achieved. The device is able to differentiate two basic leg gestures, namely forward step and side step. The device is also able to calculate individual step distance and when compared with the commercial product Fitbit Flex, the device shows a much better accuracy and consistency when tested on 10 subjects. Different programming language and techniques are also applied in order to make this project achievable. The main communication method used in this project is achieved by using Bluetooth standards. It is able to communicate with any mobile devices provided that the user has downloaded the application Walk-Off in Google Play Store. However, the system is not considered to be perfect and certain functions require improvement, this is further discussed in the next section.

6.2 Future Implementation

6.2.1 Recognition of Other Leg Gesture Motion

The prototype is only able to recognize two different leg gestures, and this would not be sufficient in real world where more leg gestures are used on a daily basis. The next stage of this project may include gesture recognition such as vertical jumps,

incline walks, decline walks, climbing stairs, and many more. This would allow users to obtain more accurate result of their daily life and travel pattern. The inclusion of more leg gestures also means the completion of the system so that it might be used as a foundation for researchers to study on human motion.

6.2.2 Bluetooth Low Energy

In this project, the Bluetooth module used is HC-05 which is a Bluetooth 2.0 technology. The Bluetooth LE as mentioned in the methodology could not be used due to the high cost required to obtain one. Moreover, certain mobile phones do not include this technology as Bluetooth LE is mainly used in the Android 4.3 operating system and above. Thus, it can be foreseen that the prices of Bluetooth LE will drop when the technology becomes more common in mobile devices. By then, it can be implemented in our device. By using Bluetooth LE, the advantage is that it will allow is to communicate with lower power consumption.

6.2.3 Mobile Application Platform

The application can be developed for other mobile operating systems which widely used around the world such as Apple IOS and Windows Mobile Phone. This would establish our mobile application and device so that more people would know about it and find it useful.

REFERENCES

Roetenberg, D., 2006. Inertial and Magnetic Sensing of Human Motion. [online] Available at: www.xsens.com_images_stories_PDF_InertialandMagneticSensingofHumanMotion [Accessed 13 June 2013].

Wikipedia the free encyclopedia, 2013a. Sport Science. Available at www.en.wikipedia.org/wiki/Human_movement [Accessed 4 July 2013].

Altun, K., Barshan, B. and Tuncel, O., 2010. Pattern Recognition: Comparative Study on Classifying Human Activities with Miniature Inertial and Magnetic Sensors, [e-journal] 43(2010) pg. 3605-3620. Available through Science Direct website www.sciencedirect.com [Accessed 29 July 2013].

Zhou, H. and Hu, H., 2008. Human Motion Tracking for Rehabilitation – A Survey, [e-journal] 1(18). Available through Science Direct website www.sciencedirect.com [Accessed 29 July 2013].

MEMSIC, 2011. MEMISC Introduces World's Smallest and Most Robust Digital Accelerometer with Features Never before Available at This Price Point. [online] Available at www.investor.memsic.com/releasedetail.cfm?ReleaseID=619035 [Accessed 4 July 2013].

Robertson, G., 2009. How powerful was the Apollo 11 computer? [online] Available at www.downloadsquad.swithced.com/2009/07/20/how-powerful-was-the-apollo-11-computer [Accessed 4 July 2013].

Zhu, R. and Zhou, Z., 2004. A Real-Time Articulated Human Motion Tracking Using Tri-Axis Intertial/Magnetic Sensors Package, [e-journal] 12(2). Available through: IEEE Xplore Digital Library website <http://ieeexplore.ieee.org/Xplore/home.jsp> [Accessed 1 August 2013].

Ha, P., 2012. Jawbone UP (2012) Review: Still Not Fit To Buy. [online] Available at <http://gizmodo.com/5965750/jawbone-up-2012-review-still-not-fit-to-buy> [Accessed 2 August 2013].

Goode, L., 2013. Comparing Wearables: Fitbit Flex vs. Jawbone Up and More. [online] Available at <http://allthingsd.com/20130715/fitbit-flex-vs-jawbone-up-and-more-a-wearables-comparison/> [Accessed 2 August 2013].

Fankhauser, D., 2013. The Tiny, Powerful Brain Inside Nike's FuelBand. [Online] Available at <http://mashable.com/2013/01/31/nike-fuelband/> [Accessed 2 August 2013].

Basis, 2012. What's Inside Basis. [Online] Available at <http://www.mybasis.com/basis-healthy-habits-technology/> [Accessed 2 August 2013].

Gadget Gangster, 2010. Accelerometer & Gyro Tutorial. [Online] Available at <http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/> [Accessed 18 July 2013].

Invensense, 2012. MPU-6000 and MPU-6050 Product Specification Revision 3.3. [Online] Available at: www.invensense.com/mems/gyrp/documents/RM-MPU-6000A.pdf [Accessed 18 July 2013].

Wikipedia the free encyclopedia, 2013b. I2C. [Online] Available at: <http://en.wikipedia.org/wiki/I%C2%B2C> [Accessed 18 July 2013].

Google, 2013. Bluetooth Low energy. [Online]. Available at: <http://developer.android.com/guide/topics/connectivity/bluetooth-le.html> [Accessed 24 August 2013].

Wikipedia the free encyclopedia, 2013c. [Online] Available at: http://en.wikipedia.org/wiki/Bluetooth_low_energy [Accessed 24 August 2013].

Rowberg, J., 2011. Example code for MPU-6050. [Online] Available at: https://github.com/jrowberg/i2cdevlib/blob/master/Arduino/MPU6050/Examples/MPU6050_raw/MPU6050_raw.ino [Accessed 18 July 2013].

Wikipedia, 2014. Usain Bolt. [Online] Available at: http://en.wikipedia.org/wiki/Usain_Bolt [Accessed 24 February 2014].

Wikipedia, 2014a. Quaternion. [Online] Available at: <http://en.wikipedia.org/wiki/Quaternion> [Accessed 30 March 2014].