NANYANG TECHNOLOGICAL UNIVERSITY

# Wearable Sensing System for Improvement in Sports Training

**Submitted by: Ank Khandelwal**
**Matriculation Number: G1402211H**

**Supervisor: A/P Wang Ping**

School of Computer Engineering

A final year project report presented to the Nanyang Technological University
in partial fulfilment of the requirements of the degree of
Masters of Science

**2015**

# Table of Contents

# Abstract

With the rise of wireless sensor technology, it is gaining popularity from consumer electronics to health-care applications; these light weight and power efficient embedded sensory devices are also finding their way into the sports industry. With the rising trend of using different methods by trainers to improve the performance of the athletes, wearable sensors are gaining huge popularity amongst the others. Wearable sensors being light weight can be worn by the user without any razzle-dazzle and can also perform very complex algorithm calculation which might help in improving their performance.

We propose to develop a system for improvement in sports training focusing on cycling which will help the cyclist and trainers to extract important information which can help them improve in their training. This contribution is concerned with the knee flexion angle calculation, pedal stroke calculation and the power measurement using the cycling ergometer. The angle calculation is based on inertial measurement unit data in the context of human motion analysis while cycling, and the number of pedal strokes calculated are also calculated using the same data. The gathered data from the in-house developed IMU sensors is sent to the host PC, processed and sent to the cloud for further processing and visualization. This IMU system and the power calculation system are a part of a larger system which is developed by the Wireless Wearable Sensing Technology Laboratory at Institute for Infocomm Research (I2R).

Cycling ergometer CompuTrainer$^{TM}$ is used to get the power output from each session and customize the track according to the users need. The power output available is not in a format which can be sent to the server and mapped with the data obtained from other systems. So a system is designed which extracts the power values and send it to the web server with the correct timestamp in order to map it with the data received.

The final system provides real time visualization on a web application of the angle and the power measurement and how it varies with the pedaling strokes.

# Acknowledgements

The satisfaction and euphoria that accompany the successful completion of the project would be incomplete without mentioning the people who made it possible; whose constant guidance and encouragement crowned my efforts with success.

I take this opportunity to express my profound gratitude and deep regards to everyone at the Wireless Sensing Technology, Institute for Infocomm Research (I2R), Singapore. I would further like to extend to my advisor Dr. Ge Yu for monitoring me and constantly encouraging me throughout the course of the project. The help and guidance given by her helped me both professionally and personally to complete the project.

I also take this opportunity to thank Wu Dajun and Jaya for their constant feedback and support throughout the project.

I would like to thank my project advisor Associate Professor Wang Ping for giving me the opportunity to work on such an exciting project and also for her exemplary guidance.

Lastly, I would like to thank my family and friends for their constant encouragement without which the assignment would not be possible.

Ank Khandelwal
15 June 2015

# Acronyms

| | |
|---|---|
| ADC | Analog to Digital Convertor |
| API | Application Program Interface |
| BDC | Bottom Dead Center |
| COM | Communication Port |
| dBm | Decibel-miliwatts |
| DMP | Digital Motion Processor |
| Dps | degree per second |
| EMG | Electromyography |
| FIFO | First In First Out |
| FPU | Floating Point Unit |
| FTDI | Future Technology Devices International |
| GHz | Giga hertz |
| GUI | Graphical User Interface |
| HR | Heart Rate |
| HTTP | Hypertext Transfer Protocol |
| I2R | Institute for Infocomm Research |
| IDE | Integrated Development Environment |
| IMU | Inertial Motion Unit |
| ISM | Industrial Scientific and Medical |
| JSON | JavaScript Object Notation |
| Mbps | Megabits per seconds |
| MCU | Microcontroller Unit |
| MEMS | Micro-Electro-Mechanical Systems |
| MPU | Memory Protection Unit |
| OTG | On-The-Go |
| PC | Personal Computer |
| RF | Radio Frequency |
| RISC | Reduced Instruction Set Computer |
| RPM | Revolutions per minute |
| RSET | Representational State Transfer |
| RX | Receiver |
| SPI | Serial Peripheral Interface |
| SRAM | Static Random Access Memory |
| TDC | Top Dead Center |
| TX | Transmitter |
| UART | Unified Modeling Language |
| UML | Universal Asynchronous Receiver/Transmitter |
| USB | Universal Serial Bus |
| WBSN | Wearable Body Sensor Network |

# Symbols

| | |
|---|---|
| $G_{gyro}$ | computed gyroscope value |
| $Raw_{gyro}$ | raw gyroscope value |
| $\alpha_{gyro}$ | angle calculated using gyroscope |
| $\alpha_{acce}$ | angle calculated using gyroscope |
| $Raw_{ace\_x}$ | raw accelerometer reading in x-axis |
| $Raw_{ace\_z}$ | raw accelerometer reading in z-axis |
| $\alpha_{final\_x}$ | final angle in x-axis |
| $\alpha_{final\_y}$ | final angle in y-axis |

# List of Figures

# List of Tables

# Chapter 1 Introduction

Competitiveness in sports has been gaining ever since and so is the desire of athlete to perform better than other. Training is one of the most important aspects for better performance. Sport’s training is defined as the body’s adaptation to the conditions of certain exercise. With the rising trend of using different methods by coaches to improve the performance of their athletes, wearable sensors are gaining huge popularity amongst others. Advancements in technology have enabled the design of light weight and power efficient embedded sensory devices. Along with this ability of wireless sensor platform to perform complex algorithm computation, simultaneously store the data, and communicate with the host systems make them lucrative for the development of wearable systems [1].

Micro-Electro-Mechanical Systems (MEMS) inertial sensors can be found in every consumer electronic devices.  These sensors have a limited life-cycle and are manufactured in abundance under tight cost constraints. An inertial measurement unit (IMU) is a device that measures and accounts velocity, orientation, and gravitational forces, using a combination of accelerometers, gyroscopes and magnetometer values [2].

Cycling is no different from other competitive sports, traditional training methods constraints the coaches for gathering biometric data from the athlete which can immensely help the athlete to perform much better and avoid injuries. This report will focus on the development of a subsystem for a product being developed by Wireless Wearable Sensing Technology at the Institute for Infocomm Research (I2R), A*Star in Singapore. The overall system is one which will collect different information from an ergometer and the collected data in processed and presented in a visualized form for the use by the instructor.

Although cycling is constrained by the circular trajectory of pedals, it is not a simple movement. The coaches can help the athletes to improve on various factors but there are certain factors which are out of reach for the coach to visualize physically. These certain factors if taken casually can cause many problems for the cyclist on a longer run and these are also very vital in affecting once performance. But cycling outdoors makes it very difficult for the data acquisition through the wireless sensor network. For this purpose a cycling ergometer is used.

We propose a system of the wireless sensors (EMG and IMU) and the cycling ergometer to study the lower limb muscle activation pattern during pedaling using the EMG sensors and measure the knee flexion angle using the IMU subsystem. The system will help the cyclist in improving their pedaling efficiency. The complete high level diagram of complete system is shown in Figure 1.

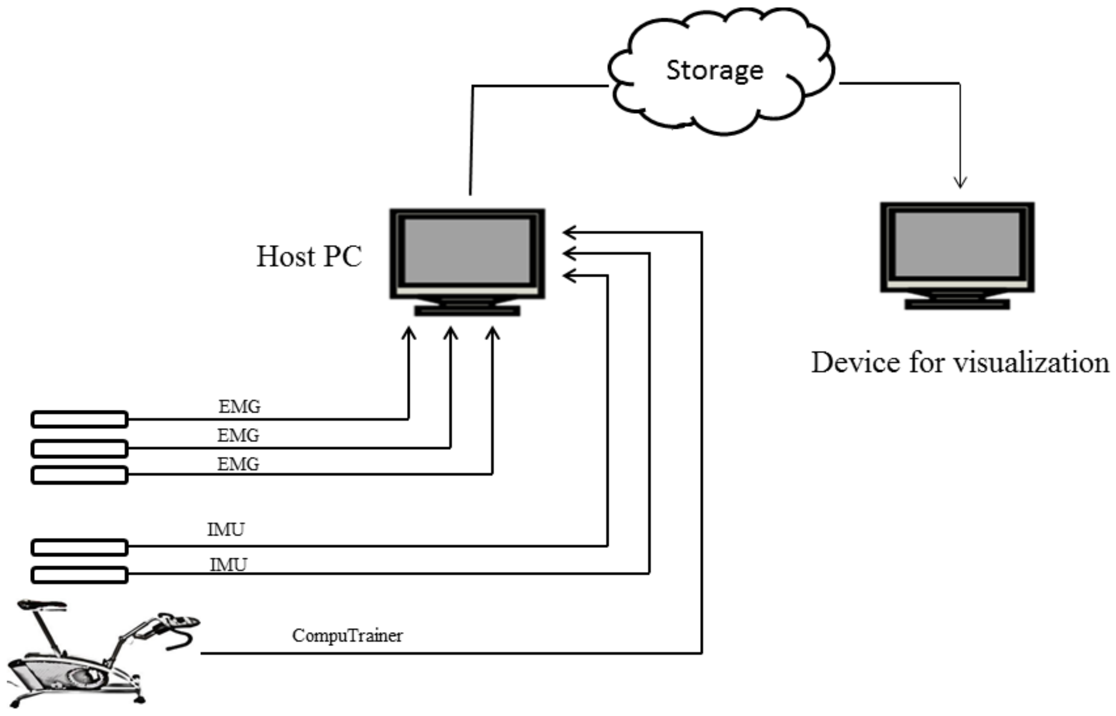


Figure 1 High Level diagram of complete system

The overall system can be broken into following sections:

*Sensor system:* Sensors gather different types of input data and send it to the host for processing

*Host:* Gather all the data and do the necessary processing and storage

*Server:* Receives data from host and stores it remote retrieval and visualization

To ensure the reliability of the system there needs to be a proper synchronization between the data received from the sensors and between the host and the web server. This is necessary so that the data is available real time and can also be visualized by the training professionals. The overall system gathers data from IMU & EMG sensors and from ergometer. Inertial Motion Unit (IMU) sensors give the position of pedal, rpm and knee angle, EMG sensors measure the power output of 6 different muscles and ergometer measures the complete power output.

## 1.1 Motivations

Wearable sensors are gaining popularity in sports, and a lot of questions are being raised on how the research on wearable sensor can be exploited to help the athletes improve their performance. Cycling is one of the competitive sports which can use the wearable sensor technology to its advantage. There is a lot of research going on, which focusses on the biomechanics of leg where the cyclist's lower limb movement as its position keeps on changing during the pedaling stroke and how can a better sitting posture help in maximizing efficiency of a pedaling stroke of a cyclist. There are a lot of factors which needs to be taken care of in order to maximize the performance. Various techniques have been proposed to carry out research in this area and they are categorized into two main categories namely video-based and wearable sensor based [3]. But there are a lot problems with the video based as numerous markers have to be used. The movement of the lower limb is captured by the camera, as light has a significant effect on detection effectiveness which makes this approach not so user friendly. On the other hand as the size of wearable sensors are becoming more compact and powerful day by day, they are not affected by the surrounding noises, light condition and it also provides more freedom in terms of human movements.

## 1.2 Objectives and Scope

The primary objective of the dissertation was to develop a cycling system which can be used by the user to interact with the whole system network and use this wearable body sensors network (WBSN) to extract useful information in order to maximize their pedaling efficiency and thus improve their performance. The secondary objective of this system is to help the trainer detect incorrect postures which might be affecting the performance and which may be the cause for injury in future. All these objectives were achieved by:

- Proper synchronization between various sensors.
- Logging all the data from sensors and ergometer on the server for server visualization.
- Creating a GUI for the user to interact with the whole system.
- Developing a webpage for real time output from all the WBSN in order to make a meaning conclusion.

The scope of this thesis extends to the design and development of a system which can help the user to maximize their cycling performance.

## 1.3 Organisations

This report is divided into 8 chapters.

Chapter 1 describes its introduction to the project, motivation behind the project and objectives and scope for this project. This discusses how we can use the wearable sensor technology in cycling and what all is expected from the project.

Chapter 2 discusses about the research which has already been carried out in this field. It also discusses some of the methods which already have been implemented. It also gives us an idea of why this project is necessary and how it can be useful.

Chapter 3 discusses about the system architecture which explains the hardware and software architecture of the complete system.

Chapter 4 discusses about the design and implementation of the IMU system and the CompuTrainer$^{TM}$ system. It discusses in detail the algorithm which is needed to calculate the knee angle, calculate the number of pedal strokes and how to extract information from the CompuTrainer$^{TM}$ which is usable for the system analysis. This chapter also discusses about the detailed implementation of both the systems.

Chapter 5 covers the testing and results section which includes testing of whole integrated system and discusses the problems which are observed during this stage.

Chapter 6 is the final section of this report and it is about the conclusion, future work and about the limitations of the system and what measures can be taken to improve it.

# Chapter 2

# Literature Review

## 2.1 Robotic Hinge Joint and Human Knee Joint

Thomas Seel et al. in their paper discusses about the major challenges faced for IMU- based human gait analysis [4]. Their argument is regarding the hinge joint (knee joint) i.e. joints with one rotational degree of freedom as depicted in Figure 2 (b) [5] . Peng Cheng et al. in the paper titled “Joint-Angle Measurement Using Accelerometer and Gyroscopes – A Survey” has demonstrated that the inertial measurement data can be used to calculate hinge joint angles when at least one IMU sensor is attached to each side of the joint [6]. In most robotic and mechanical with hinge joint axis as shown in Figure 2 (a), the hinge joint angle can be measured by integrating the difference of angular rate of both the sensors around their corresponding co-ordinate system. Since even the most precise calibration will yield a non-zero bias, this calculated angle will be subjected to drift.

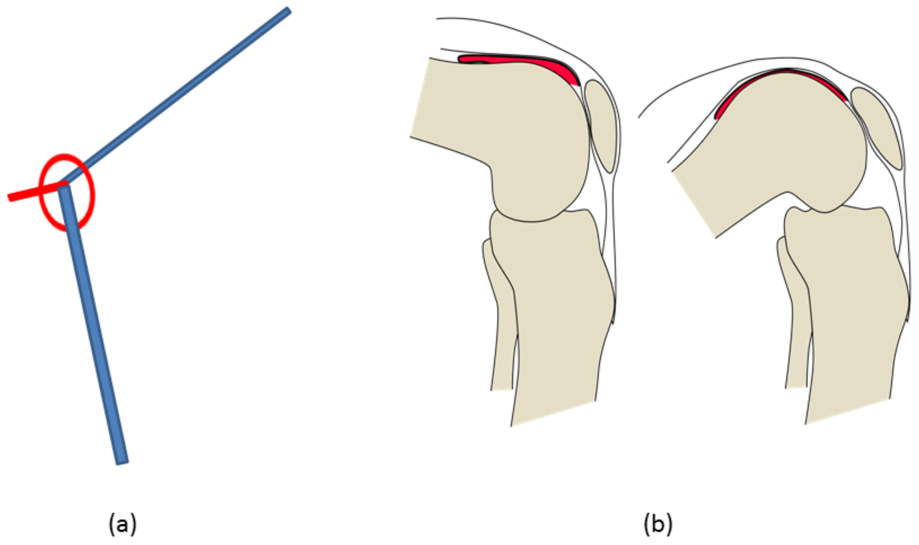


Figure 2 (a) Robotic hinge joint (b) Knee joint

Using the principle discussed above inertial measurement units (IMU's) can be used to calculate the angle of hinge joints on the human body, for example on the knee joint or the elbow joint. However it is not as simple, as there is very important difference between the human leg and most robotic setups: It is very difficult to attach IMU's to the leg in such a way that one of the local coordinate axes coincides exactly with the knee joint axis, still accuracy is limited.

## 2.2 Importance of knee angle

Muscle architecture of the human lower limb and the optimum angle of torque development have a crucial role to play in skeletal muscle function. All of the major muscles of the legs are used at one point of another during the pedaling stroke but the major muscles that are in action for generating power are the quadriceps group. During the push phase when the pedal goes from Top Dead Centre (TDC) to Bottom Dead Centre, (BDC) quadriceps is mainly in action as you push the pedal down and straighten your leg while the hamstrings at the back of the thigh work to bent the knee [7].

With the bike correctly set up during one complete rotation of the pedal the knee travels from 30 to approximately 110 degrees of flexion. The position and body posture of the cyclist on a bicycle can determine how well the body performs during a cycling task [8] [9]. In a competitive cycling, setting up the proper saddle height is important for both performance and injury prevention. According to BikeDynamics ideal saddle height (shown in Figure 3 [10]) can be described by the angle of the knee at full extension which is 143.6 degree and once this has been established angles below 70 should be avoided as this can result in sheer force into the knee joint at the top of the pedal stroke and can cause discomfort in the hip and lower back [11].

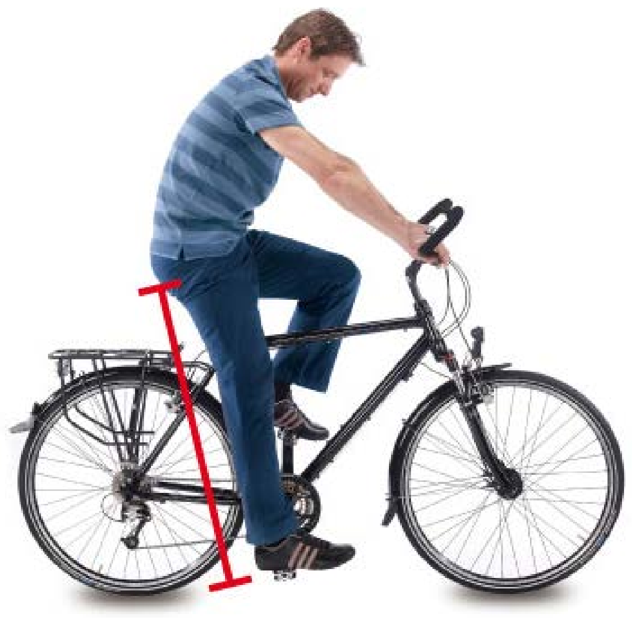


Figure 3 Saddle Height

## 2.3 Ergometer

Cycling coaches and sports scientists around the world uses field and laboratory –based cycling tests to assess the performance level of competitive cycling and to investigate nutritional strategies, and other factors that might affect performance level of a cyclist. Time-trials and other field-based tests that reproduce the demands of competitive events are generally better than the laboratory-based tests for these purpose [12], but cycling outdoors and measuring the performance level of the cyclist using wearable sensor is difficult in outdoor condition due to environmental conditions and the wireless range of sensors. Due to all these limitations stationary cycle ergometers like CompuTrainer$^{TM}$ are in widespread use which can mimic the outdoor conditions and produce some of the best measures of performance in the laboratory. The indoor cycle ergometer allows cyclists to record their speed, power output, and pedaling efficiency while relaying the information to a computer. In addition, the cycle ergometer’s software can simulate a virtual course through which a cyclist can pedal or simulate a cycling event. But there is minimal research that supports or disputes simulation of cycling with the use of an indoor cycle ergometer as a training practice for bikers. [13, 14]

# Chapter 3

# System Architecture

The high level diagram of the whole system is shown in Figure 1. As discussed the system is divided into 3 parts sensor system, host and server. We will discuss the hardware and software components of each in detail in this section.

## 3.1 Sensor System

Sensor system comprises of a transmitter unit which is a wearable sensor which is connected to the body of the user and receiver unit is connected to the host computer. This can be visualized in the Figure 4.

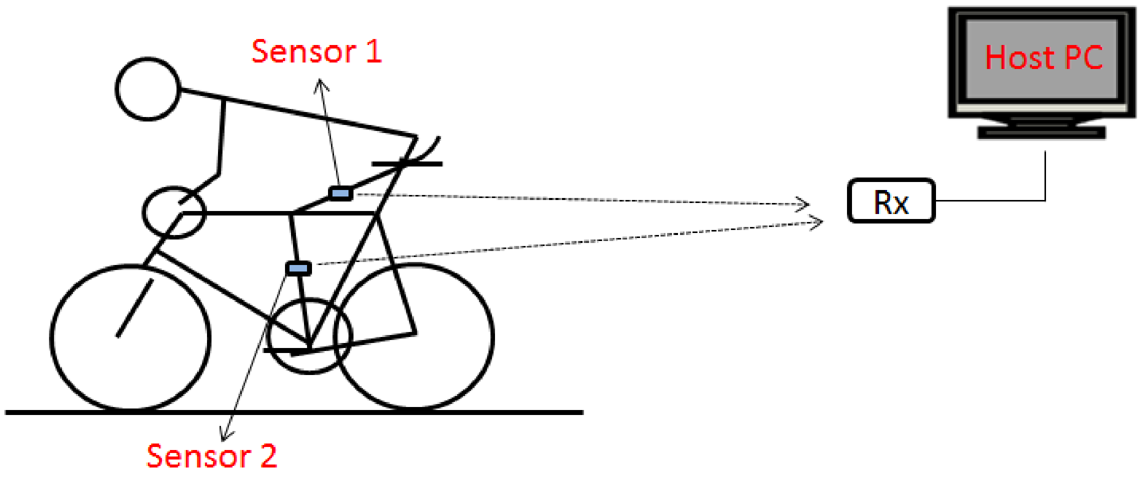


Figure 4 Sensor System

### 3.1.1 Microcontroller Unit

The processing unit of the system is microcontroller (MCU), STM 32F401xD from ST Microelectronics. Due to its low power consumption it is a suitable choice for biomedical signal acquisition systems. Device has a 12-bit ADC, 512 kB flash memory, 96 Kbytes SRAM embedded inside it and needs a current between 1-146 uA when operated at 26 MHz [15]. The microcontroller (MCU) uses serial communication port at a baud rate of 115,200 to connect to the Nordic nRF24L01 transceiver. The IMU information is output to the MCU using the I2C protocol. The processing unit has a C code dumped on it and it is programmed as transmitter or receiver depending upon its usage. Figure 5 shows the processing unit of the platform.

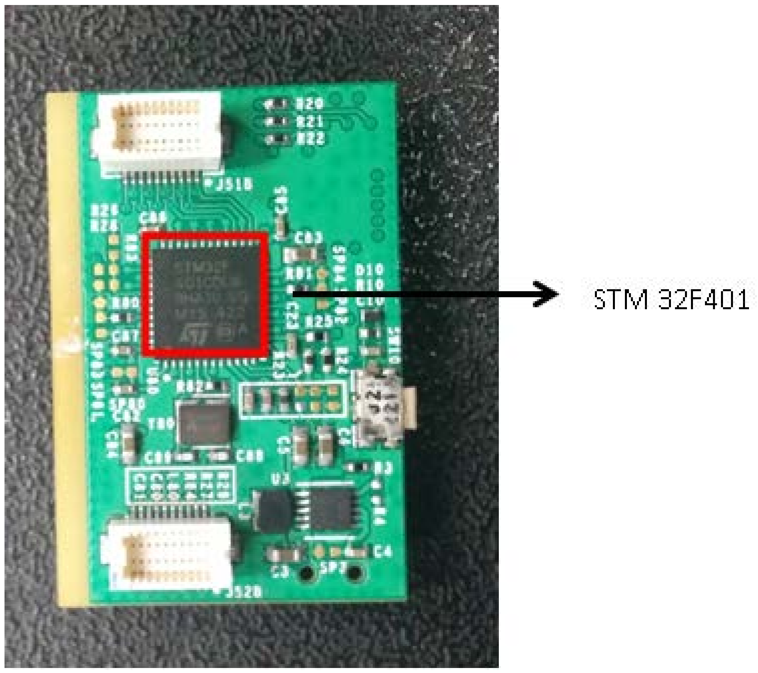


Figure 5 Processing Unit of Wireless sensing platform

### 3.1.2 Transmitter Unit

Nordic nRF24L01 ultra low power 2.4 GHz Transceiver IC was chosen as the wireless transmission interface due to its low cost and low-power radio frequency transmission. The chip is energy saving and has a high transmission capacity of 2 Mb/s. It is also easy to integrate into the low-level circuit requirements.

Figure 6 shows the transmitter model of an in-house developed wireless sensing platform. It consists of a rechargeable lithium-ion battery (3.7 V and 550 mAh). It is also attached to the processing unit, IMU and Nordic nRF24L01 for the wireless transmission. Mode of communication between MCU, IMU and RF is using Serial Parallel Interface (SPI) bus. The lithium-ion battery can be charged through USB OTG cable.

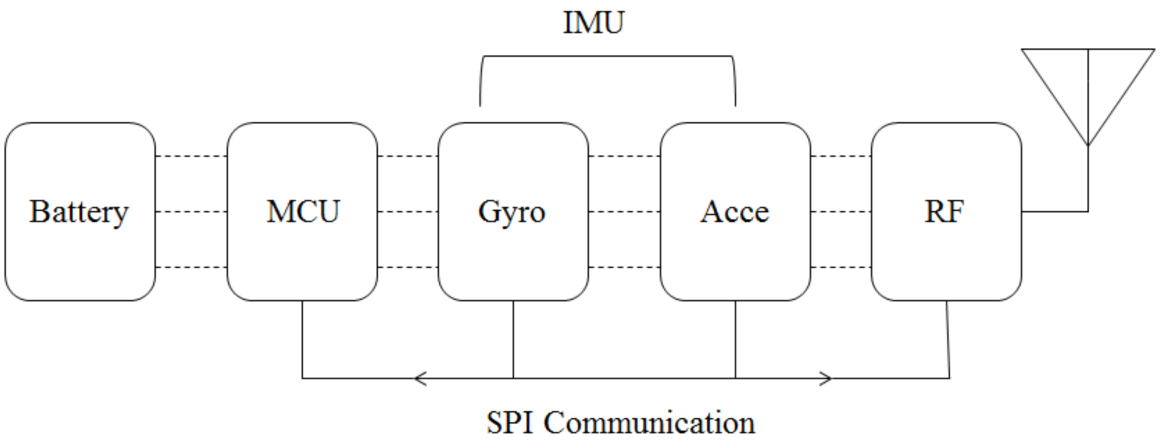


Figure 6 Transmitter Unit

The Figure 7 and Figure 8 show the top-view and side-view of the wireless sensing platform respectively.

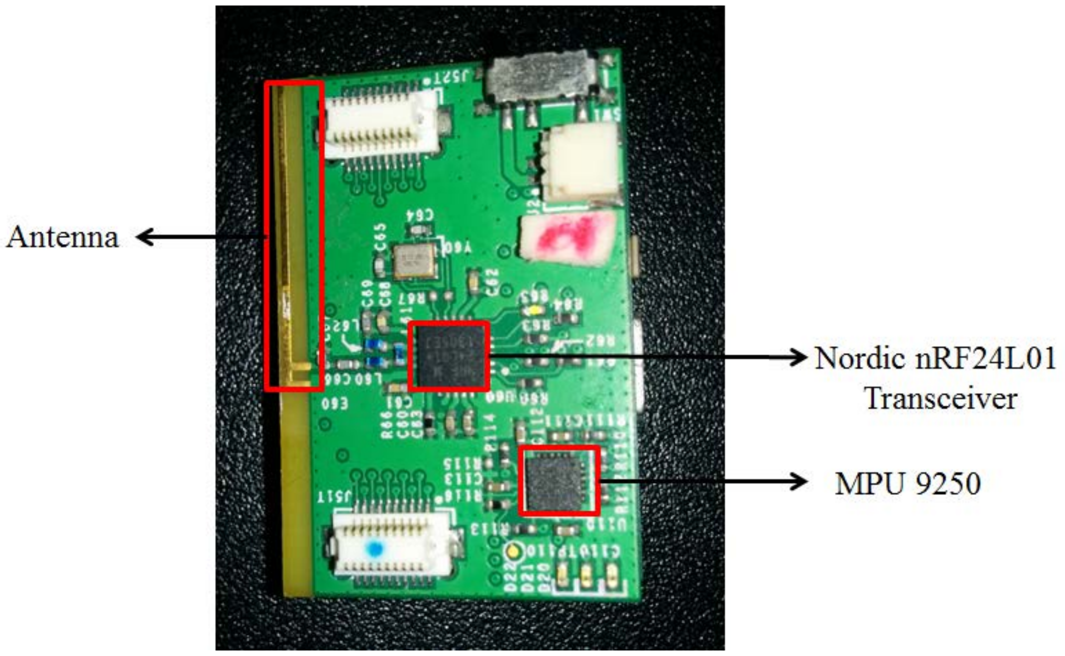


Figure 7 Top-view of Wireless sensing platform

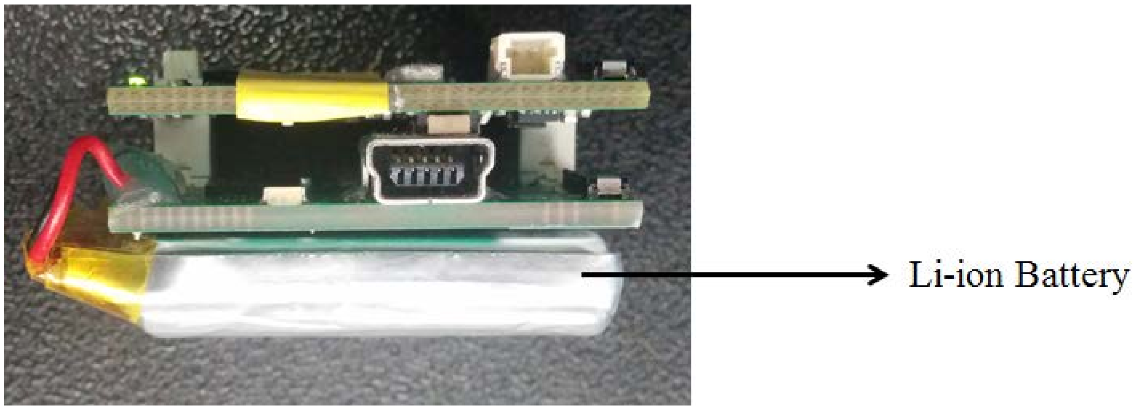


Figure 8 Transmitter unit

### 3.1.3 Receiver Unit

The receiver unit is shown in the Figure 9; it consists of Nordic nRF24L01 which is able to receive data from 1 or more transmitter unit. The transmitter decodes on the basis of their ID. The RF chip is a bare metal chip, and the channel for its communication is programmed using MDK-ARM. For proper communication between the transmitter and receiver unit both has to be configured with the same information channel so there is minimal interference from the other sensors and valid packets are received. The MCU unit is used to receive the data packets from the air interface and it send it to the USB to UART using FTDI which can then be sent to the host PC using COM Port. Figure 10 and 11 shows the top and side view of the receiver unit respectively.

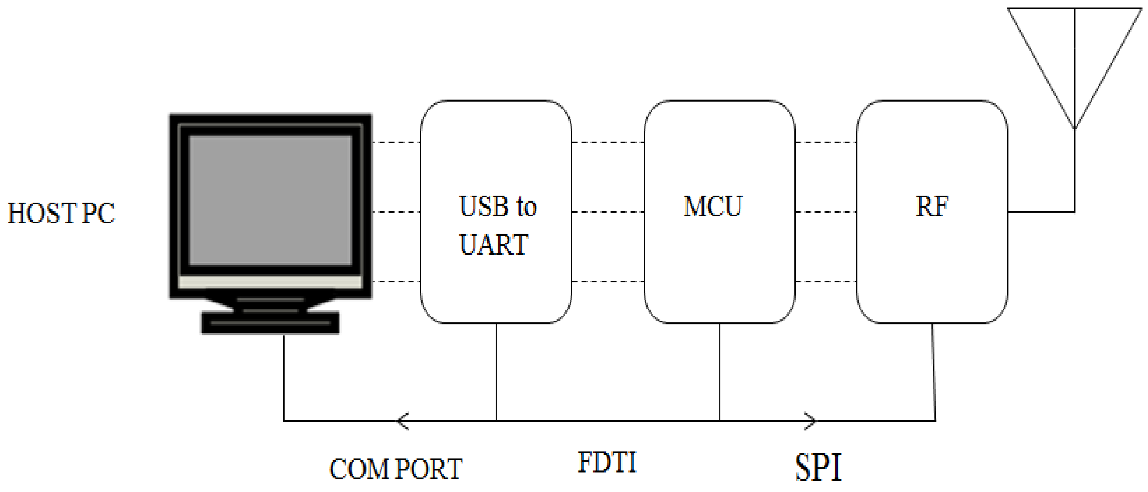


Figure 9 Block diagram of Receiver Unit

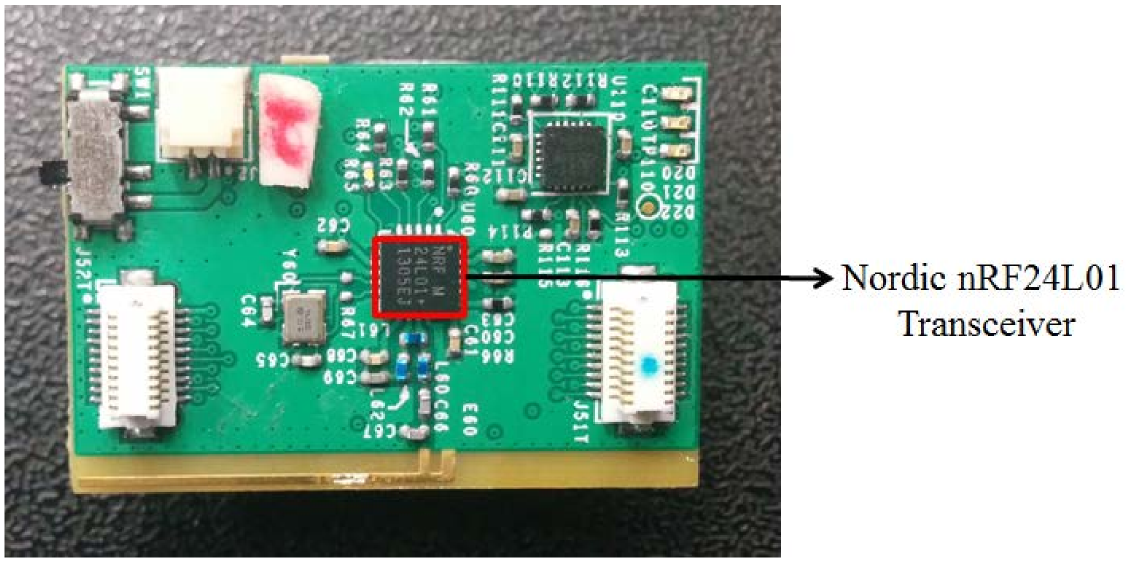


Figure 10 Receiver unit

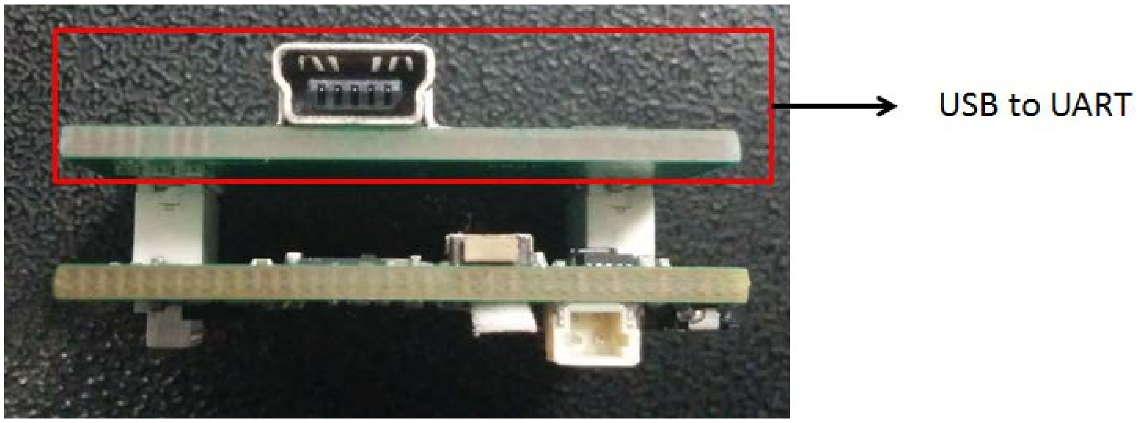


Figure 11 USB to UART

## 3.2 Hardware components

### 3.2.1 Nordic nRF24L01 Transceiver

Nordic nRF24L201 is a 2.4 GHz transceiver which consists of an embedded baseband protocol engine (Enhanced ShockBurst™). The transceiver is designed for applications which require very low power consumption. It is designed to function in the frequency band of 2.400 – 2.4835 GHz. A MCU and very few external passive components are needed in order to design a radio system using nRF24L01. It is operated using Serial Peripheral Interface (SPI). The embedded baseband protocol engine i.e. Enhanced ShockBurst$^{TM}$ is based on packet communication and supports various modes from manual operation to advanced autonomous protocol operation. A smooth flow of data is ensured between the radio front end and the system's MCU using internal FIFOs. Enhanced ShockBurst$^{TM}$ also reduces the system cost by handling all the high-speed link layer operation. The air data rate supported by the nRF24L01 is configurable to 2Mbps [16]. The very high data rate combined with 2 power saving modes makes this chip a favorable choice for our application.

Some of the features of nRF24L01 include:

**Radio**

- Worldwide 2.4GHz ISM band operation
- Common RX and TX pins
- 1 and 2 Mbps air data rate
- 2 MHz non-overlapping channel spacing at 2 Mbps

**Transmitter**

- Programmable output power: 0, -6, -12 or -18 dBm
- 11.3 mA at 0 dBm output power

**Receiver**

- Integrated channel filters
- 12.3 mA at 2 Mbps
- -82 dBm sensitivity at 2 Mbps
- -85 dBm sensitivity at 1 Mbps

**Enhanced ShockBurst™**

- 1 to 32 bytes dynamic payload length
- Automatic packet handling
- Auto packet transaction handling

**Power Management**

- Integrated voltage regulator
- 1.9 to 3.6 V supply range
- Idle modes with fast start-up times for advanced power management

### 3.2.2 STM 32F401xD

STM32401xD device is based on the high-performance ARM® Cortex® -M4 32- bit RISC core operating at a frequency of up to 84 MHz. It also features a floating point unit (FPU) which supports all ARM single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security. Some of the other features of STM32F401xD are:

- 512 Kbytes of Flash memory
- 96 Kbytes of SRAM
- 12-bit ADC
- Three I2Cs
- Up to four SPIs
- USB 2.0 OTG full-speed interface

A comprehensive set of power-saving mode allows the design of low-power applications which makes it a suitable choice to integrate it with wearable sensor technologies [15].

### 3.2.3 InvenSense IMU Sensor

InvenSense MPU-9150 Intertial motion sensor is used as a 9-axis Motion Tracking MEMS device because of the low power and high performance requirements of the wearable sensors. It contains a 3-axis accelerometer, 3-axis gyroscope and a 3 axis compass. The Digital Motion Processor (DMP) gives the quaternion representation of the movement [17]. Some of the features of the Invensense IMU Motion sensor are as follows:

**Gyroscope Features**

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec and integrated 16-bit ADCs
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.2mA
- Factory calibrated sensitivity scale factor

**Accelerometer Features**

The triple-axis MEMS accelerometer in MPU-9250 includes a wide range of features:

• Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and
±16*g* and integrated 16-bit ADCs

• Accelerometer normal operating current: 450µA

**Magnetometer Features**

The triple-axis MEMS magnetometer in MPU-9250 includes a wide range of features:

• 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator

• Wide dynamic measurement range and high resolution with lower current consumption.

**Additional Features**

The MPU-9250 includes the following additional features:

• Auxiliary master I2C bus for reading data from external sensors (e.g. pressure sensor)

• 3.5mA operating current when all 9 motion sensing axes and the DMP are enabled

• VDD supply voltage range of 2.4 – 3.6V

• VDDIO reference voltage for auxiliary I2C devices

• Minimal cross-axis sensitivity between the accelerometer, gyroscope and magnetometer axes

• 512 byte FIFO buffer enables the applications processor to read the data in bursts

• User-programmable digital filters for gyroscope, accelerometer, and temp sensor

• 400 kHz Fast Mode I2C for communicating with all registers [17]

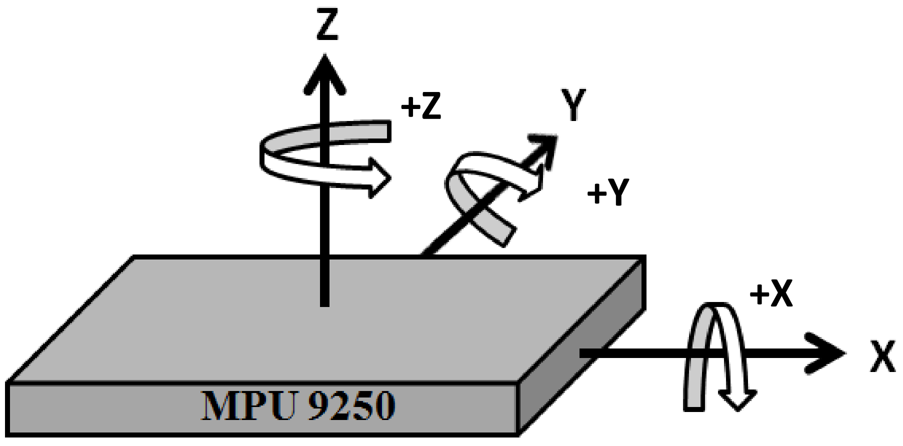


Figure 12 Orientation of Axis for Accelerometer and Gyroscope

## 3.3 Software

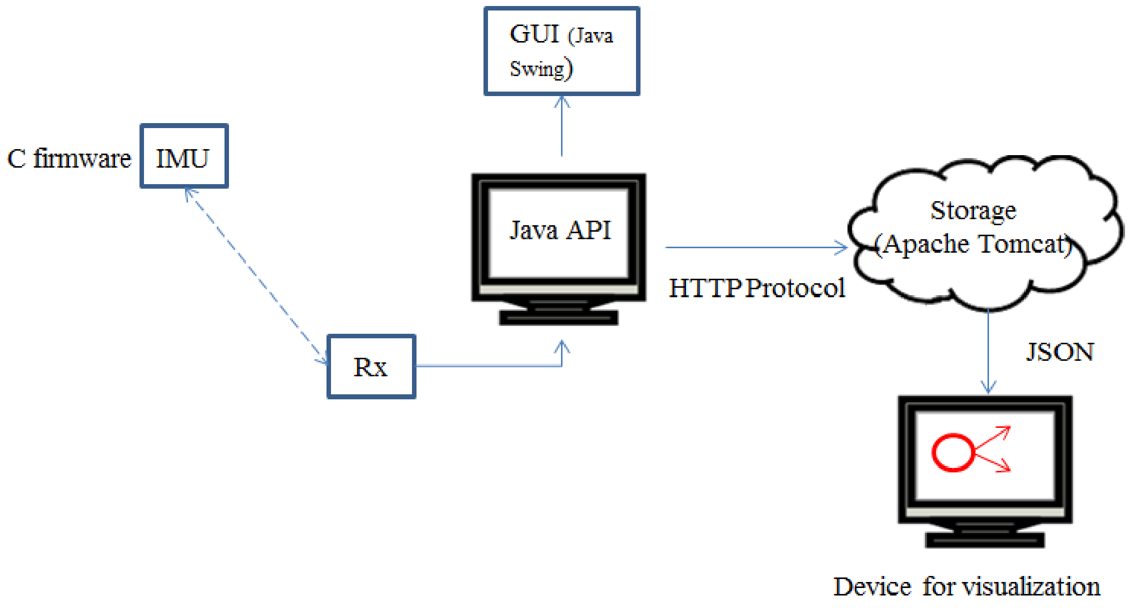


Figure 13 Block diagram of Software Architecture

Software part of the whole system is shown in the Figure 13. It consist of the following part

- IMU sensor unit which consist of MCU is loaded with a firmware in C language.

- The host PC consist a Java API which is used to decode the data received for the receiver in the proper format and apply the necessary algorithm based on the application
- Also at the host side, Java SWING GUI is there to interact with the whole system and visualize the whole data.
- The computed data using the algorithm is sent to the server using the HTTP protocols and in the cloud data from the multiple sensors are stored in the database using the Apache Tomcat.
- Now this data in order to be visualized is sent to another system, JavaScript Object Notation (JSON) is used to transmit data between a server and a web application.

### 3.3.1 Host to Server Communication

Host to server communication is a critical part of any application because it needs to store all the incoming data from different sources and visualize it to the user, so that they can reuse the data for analysis. Therefore it was equally important to design a structure of communication between the host and server which portable. So much of thought was put into this because the server needs to receive data from different systems and platforms. Initially the system was designed to use Representational State Transfer (REST) style of requests, using the HTTP verb “GET”. However, with the existing server architecture this only allowed a set of data points from one instant in time to be sent per request. This meant that there was a limit to sampling rate which could be sent to the server.

There were a number of different ways by which we could approach this problem, but it was concluded that the most simplest and effective method to solve this problem would be to change the format of requests and send the data in the JavaScript Objet Notation (JSON) format instead. This allowed an arbitrary number of sets of data points to be transmitted with every request; also this meant that the only limit to sampling rate would be the time taken by the server to parse the JSON request.

### 3.3.2 Timing Considerations

Timing was a critical issue to consider as the overall system aims to collect process and log the real-time data with the smallest delay possible. Problems relating to concurrency and part of the system which needed to run at the same time namely data collection, processing and logging to the server (all in different threads) was resolved using Java as all the shared variables are thread safe by default.

Synchronization was also an issue which was encountered during the trials conducted when all the systems were integrated. All the systems ran on different systems and the data collected was sent to the server using their own system timing, which differed from platform to platform which resulted into improper mapping of data received from different sources thus losing valuable critical information. Cycling occurs at a very fast rate so it became even more critical for the data to be received and reduce the delay as much as possible. Also the Computrainer™ provided the data only at the end of each session which is needed to be mapped with all the data received, so a meaningful conclusion can be drawn from the data collected from all the sensors. Even if the system timing of all the systems were matched the received data might not actually have been recorded at the same point in time (since devices might have been started at slightly different times).

The issue was resolved by logging the Unix time which each data packet as it provides precision in milli seconds, so it became easy to map the data received from different sources and produce the overall result for display for the user. The data logged by Computrainer™ did not allow the user to alter information in a way to insert Unix timestamps with the data samples collected. Also the log file did not provide any information when the session started or ended. In order to overcome this issue the last modified time for the file was checked which was the time when the session ended and this time was converted to Unix time and with the difference in the duration between the consecutive samples, data was sent to the server with the computed timestamp to properly map with the other data collected from different sensors.

One issue which came up during implementation was the time taken to actually process a request to the server. The time taken due to the HTTP overhead was much larger than any other part of the logging process. Therefore data was arriving faster than we could send it to the server. We solved this issue through the implementation of further multithreading where data to be sent to the server would be allocated its own thread from inside the logging thread. This means that more than one HTTP request could be initiated at once and while these were occurring we could also be dealing with new data to be logged.

### 3.3.3 Integrated Development Environment (IDE)

In order to aid in the development of the subsystem an Integrated Development Environment (IDE) is used to develop the solution. The IDE used for this project was Eclipse [18] which is a multiplatform plug-in based environment used with a number of different languages. It was used as it has good Java integration, good user interface and useful debugging tools.

Other details regarding the java implementation and GUI development will be discussed in later sections.

# Chapter 4

# System Design and Implementation

As discussed before the thesis focusses on 2 systems and details about their design will be discussed in this chapter.

- IMU system: We will discuss about the algorithm which is required in order to calculate the angle and to compute the number of pedaling strokes by the user. After this we will discuss about the implementation of IMU system.
- Computrainer™ system: In this we are going to discuss about the algorithm and implementation to extract power from the log file.

## 4.1 IMU System

### 4.1.1 Computational Algorithm

This part of the section will discuss about the algorithm for angle calculation and speed calculation using the IMU sensor.

#### 4.1.1.1 Angle Calculation

In order to compute the angles values from the IMU sensors, both the gyroscope and accelerometer values are necessary, this is because using either one of them will result in inaccurate readings. Gyroscope as discussed earlier in the background section computes the rate of rotation around the 3 axes of space in degree/s shown in Figure 14 [19]. This information is required to be traced over time in order to compute the current angle. There are a few problems associated with gyroscope; firstly it just provides the angular rate not the absolute measure. Initial values of pitch and roll (Figure 14) will always be zero for any case. Secondly tracking the information over time results in drift, which means even if someone stands still, the sensor, will output values different from zero. But gyroscopes are better for measuring sharp movements unlike accelerometer [20].

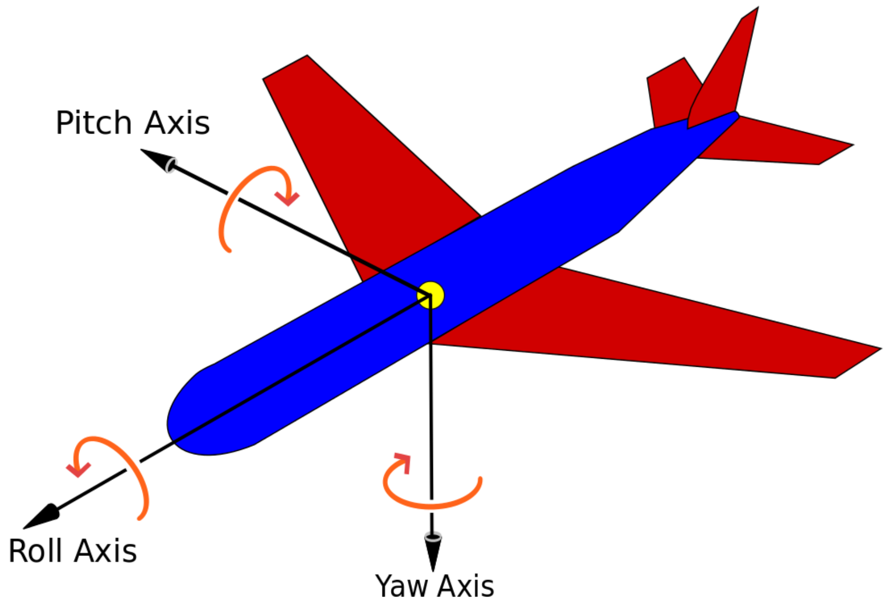


Figure 14 Roll, Pitch and Yaw calculation

Accelerometer measures combined acceleration (static and dynamic), static acceleration is the acceleration due to gravity and the dynamic acceleration is due to the sudden movements. Acceleration is measured as *+1g* on the z-axis when both the pitch and roll angles are zero, but when the sensor is tilted in the other axes it experiences a component of the upward acceleration whose magnitude depends on tilt angle [21]. Unlike gyroscope accelerometer don't need to be traced and can compute the current angle at any instant of time. But accelerometer values alone cannot be used to compute angles because the readings are very noisy and are only useful for tracking angle over a long period of time and it is recommended to use gyroscopes value for the application that is discussed in the thesis.

Angle can be calculated from the gyroscopes by knowing how fast it is rotating in terms of degree per seconds (dps). This is done by multiplying the raw value with the sensitivity level that was used when enabling the gyro.

$$G_{gyro} = Raw_{gyro}*(Range/2^{Resolution}-1)$$

Where, $Raw_{gyro}$ is the raw sample values from the gyro

To compute the angle from the above information we need to track the above gyro value overtime.

$$\alpha_{gyro} += G_{gyro} * dt$$

where, $dt$ = loop period and $\alpha_{gyro}$ is the current angle calculated from the gyro data.

To conversion of accelerometer values into usable angle is done by using trigonometric function *Atan2* function.

$$\alpha_{acce} += (atan2(Raw_{ace\_x}, Raw_{ace\_z}) + \pi) * (180/\pi)$$

where, $\alpha_{acce}$ is the angle calculated from the acceleration and $Raw_{ace\_x}$ & $Raw_{ace\_z}$ are the raw values of acceleration in x and z direction.

After calculation of both the values, these values are combined to eliminate drift caused by gyroscope and noise from the accelerometer. To eliminate the noise, filter is used which will calculate the final output by taking gyroscope values for shorter period and accelerometer for longer periods. Complimentary filter is used because, as told earlier the accelerometer value of the angle is not subjected to drift so we use this value to correct the value given by gyroscope which is more precise and less subjected to the noise caused by vibration. Therefore we combine the two values as shown in the equation below.

$$\alpha_{final\_x} = (1-GyroPercentage)*(\alpha_{final\_x} + \alpha_{gyro\_x}) + (GyroPercentage*\alpha_{acce\_x})$$

$$\alpha_{final\_y} = (1-GyroPercentage)*(\alpha_{final\_y} + \alpha_{gyro\_y}) + (GyroPercentage*\alpha_{acce\_y})$$

For our case we take the value of *GyroPercentage* as 0.02.

Complimentary filter is used instead of Kalman filter as the later one is very processor intensive and both give the same output results, hence complimentary filter is the ideal choice for wearable sensors.

Figure 15 given below shows the block diagram representation for the angle calculation.

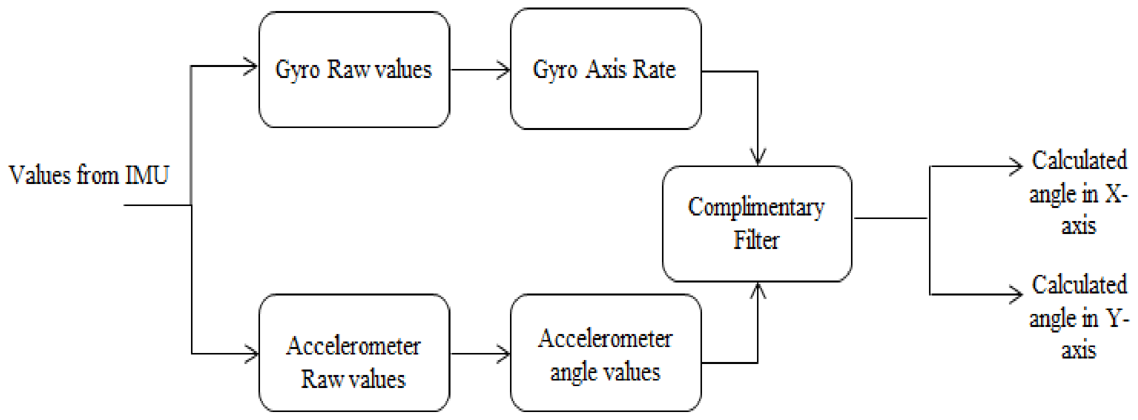


Figure 15 Block diagram of Angle Calculation

The values from the sensor are collected using the host PC and the raw values are multiplied with the sensitivity values as mentioned in the data sheet. The plots for raw gyroscope values and usable gyroscope values are given in Figure 16 and Figure 17.

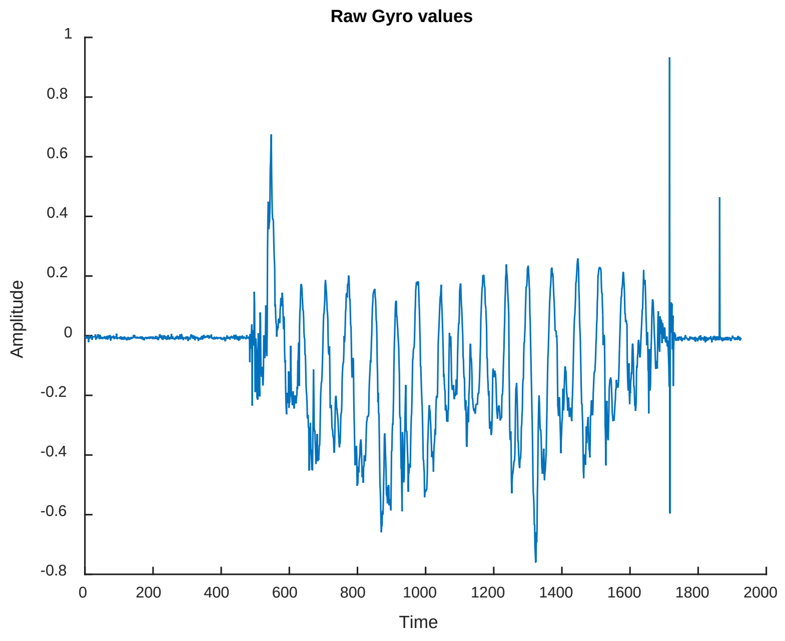


Figure 16 Raw Gyroscope values

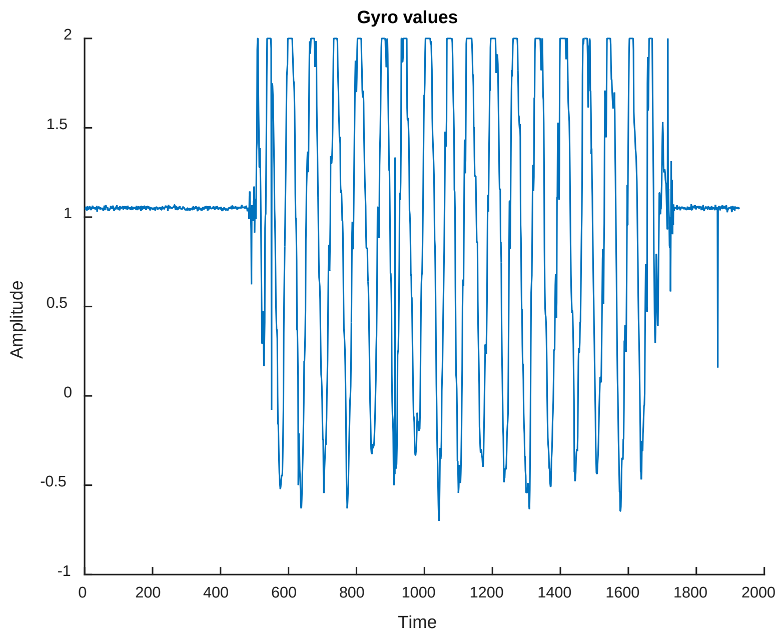


Figure 17 Computed Gyroscope values

The plots for raw gyroscope values and usable gyroscope values are given in Figure 18 and Figure 19.

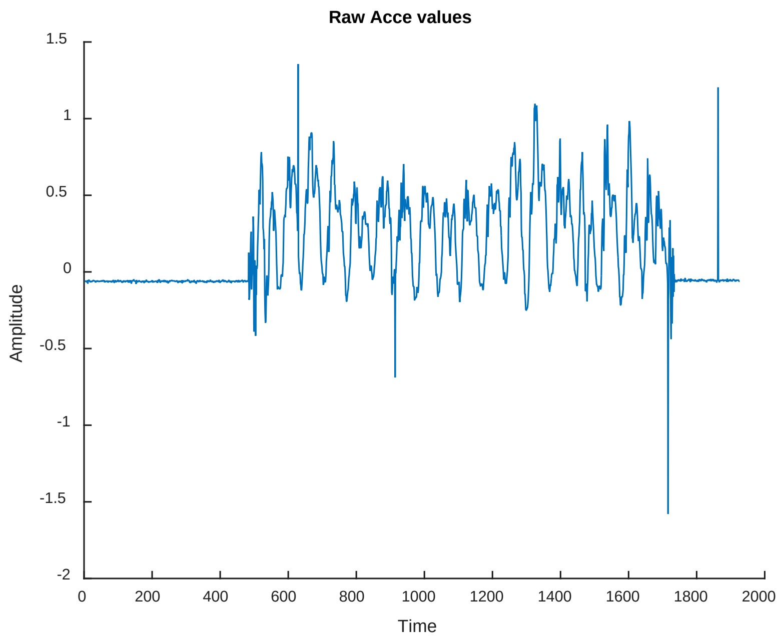


Figure 18 Raw acceleration values

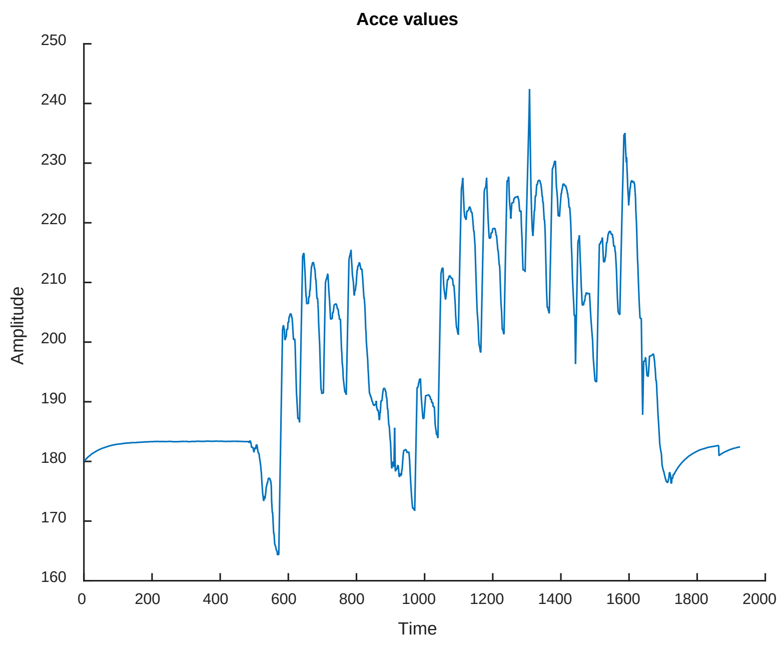


Figure 19 Calculated acceleration values

### 4.1.1.2 Flexion Knee Angle Calculation

In the previous section we discussed the algorithm required for angle calculation. In order to calculate the knee flexion angle, two IMU sensors are required to be mounted on the user's knee. The high level block diagram for flexion knee angle calculation is shown in Figure 20.

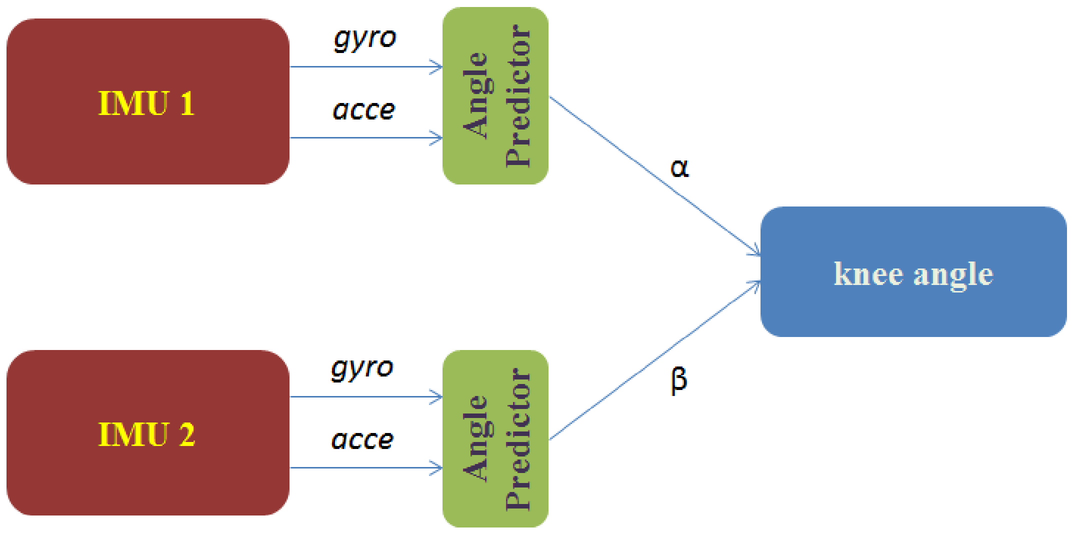


Figure 20 Knee angle calculation

The gyroscope and acceleration values are received from the individual sensors and using the algorithm discussed for angle prediction is used to calculate the angle. Subtracting one value from the other gives the knee angle. The proper positioning of the sensor on the body will be discussed in the next section.

The flow chart for knee angle calculation is shown in Figure 21. The algorithm for the same is as follows

1. Firstly the values are taken from the COM Port, each data packet which is received from the IMU sensor is 32 bytes in size, which has to be decoded to get accelerometer and gyroscope values in x,y and z axis. This is discussed in detail in next chapter.
2. According to the sensor ID which is present with each packet, usable values for accelerometer and gyroscope is computed using the sensitivity values provided in the data sheet.
3. After that depending upon the gyroscope and accelerometer values different computation is applied to calculate the angle.
4. That angle is passed through the complimentary filter and combined to get the final usable angle.
5. Flexion knee angle is the difference between both the angles.
6. This angle is updated on the GUI in real time and based on the value "Good Angle" or "Bad Angle" is shown on the GUI.

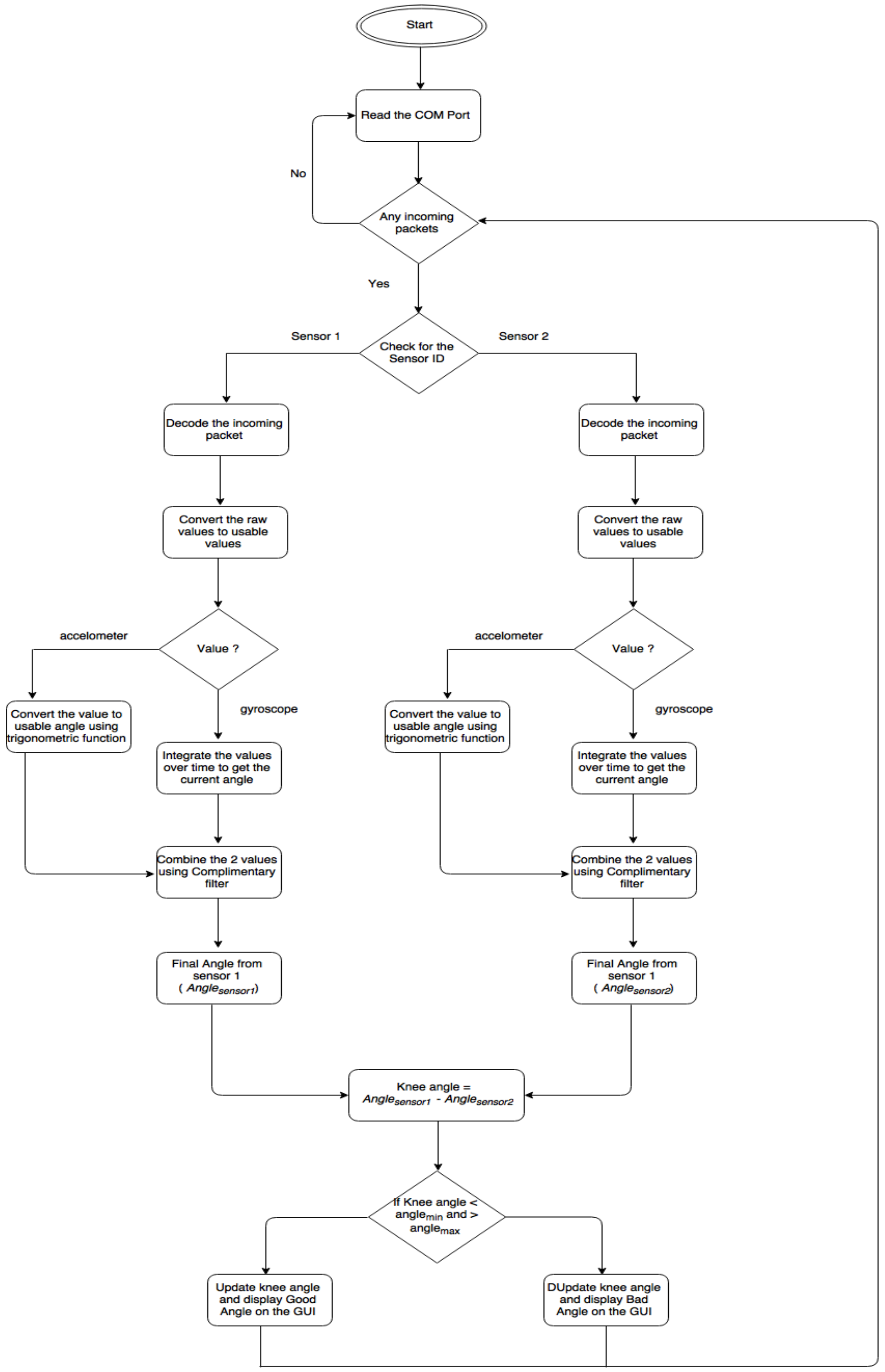

Start
Read the COM Port
No
Any incoming packets
Yes
Sensor 1
Check for the Sensor ID
Sensor 2
Decode the incoming packet
Decode the incoming packet
Convert the raw values to usable values
Convert the raw values to usable values
accelometer
Value ?
accelometer
Value ?
Convert the value to usable angle using trigonometric function
gyroscope
Convert the value to usable angle using trigonometric function
gyroscope
Integrate the values over time to get the current angle
Integrate the values over time to get the current angle
Combine the 2 values using Complimentary filter
Combine the 2 values using Complimentary filter
Final Angle from sensor 1 ( $Angle_{sensor1}$)
Final Angle from sensor 1 ( $Angle_{sensor2}$)
Knee angle = $Angle_{sensor1}$ - $Angle_{sensor2}$
If Knee angle < $angle_{min}$ and > $angle_{max}$
Update knee angle and display Good Angle on the GUI
DUpdate knee angle and display Bad Angle on the GUI

Figure 21 Flowchart for knee angle calculation

### 4.1.1.3 Number of Pedal Strokes

The IMU can also be used to calculate the number of pedal strokes a user performs while cycling, whenever a pedal stroke is performed a sinusoidal type of waveform is observed from the acceleration data. A plot of acceleration in x direction vs time is shown in Figure 22.

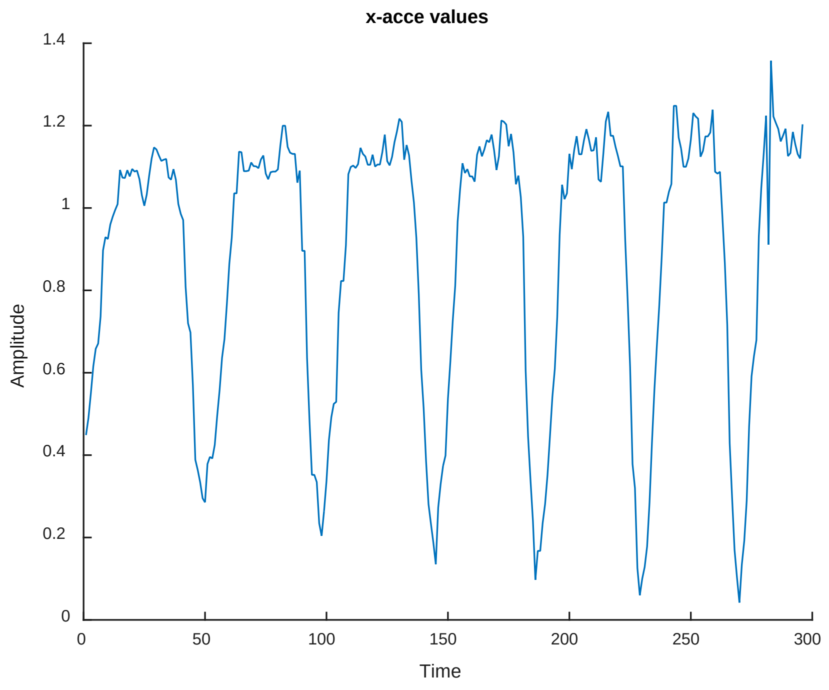


Figure 22 Change of acceleration in x direction during cycling

On further observation it was found out that each peak on the negative y-axis (due to the orientation of the IMU sensor mounted on the body) of the plot shown in the figure above represents a pedaling stroke. The amplitude of the peak depends upon the force by which pedal action is performed. This observation was used to design a simple algorithm to compute the number of pedaling strokes performed during the cycling session. To eliminate the noise moving window averaging filter was used and window size was decided accordingly. It was necessary to necessary to remove the unwanted noise which can affect the calculation and result in additional peaks. Two methods were used to compute the number of pedaling strokes -

- In the first method static threshold was used, in this method counter increases each time whenever the threshold value is crossed. As discussed the magnitude of each peak depends upon the force which may vary from person to person. So it may sometime happen that the magnitude of peak is below the threshold which will not be detected as a pedaling stroke and will result in incorrect calculation.
- In the second method dynamic threshold was used, in this the user is asked to perform cycling at the slowest rate possible and the threshold value is set accordingly. Whenever that threshold value is crossed the counter is increased and it is shown on the GUI. This method is better than the method discussed previously and the chances of error are also less.

### 4.1.2 IMU System Design

IMU system was designed to be flexible as possible so that it can be connected to multiple IMU sensors at a time, gather the data, process them and provide real time visualization for the user. Therefore with this in mind along with portability with the other systems it was decided to design using an object oriented language in mind, Java in particular. Any other design considerations will also be discussed in the report which had to done during the implementation stage due to the issues which occurred.

### 4.1.3 IMU System Specification

In order for the IMU system to function properly, following are the functional requirements for the system:

- The system should be able to connect to 2 IMU sensors at a time and more if required.
- The received data from both the sensors should be correctly differentiated so it does not interfere with their respective calculations.
- Data should be properly filtered in order to remove noise.

- After filtration data should be processed in such a manner that useful information can be extracted.
- The IMU sensors should communicate at a very high sampling rate without any loss of data so the information is not lost.
- The useful information should be sent to the server at the highest sampling rate possible.
- Information sent to the server should be time-stamped in Unix epoch measured to milliseconds accuracy.
- Data received from the Computrainer™ should be properly logged and correctly mapped with the data received from the IMU sensor using the Unix time stamp.
- Data should also be logged to a local file on the host PC.
- To allow the user to interact the system through a Graphical User Interface.
- Provide real time graph on the graphical user interface, so it is easy for the user to analyze the data.

### 4.1.4 Arbitrary Mounting Orientation and Position

One problem with the IMU based human gait analysis is that the local coordinate axes of IMU are not aligned with any existing meaningful coordinate system. But this problem can be ignored by assuming that IMU's can be attached precisely in a predefined position as visualized in the Figure 23 [4] for an illustration.

As seen in the figure, the sensor mounting positions are characterized by the local coordinates of the joint's axis and position which can be measured manually. There is no need for calibration as in Outwalk protocol [22] [23] which employs pure flexion/extension motions and static poses

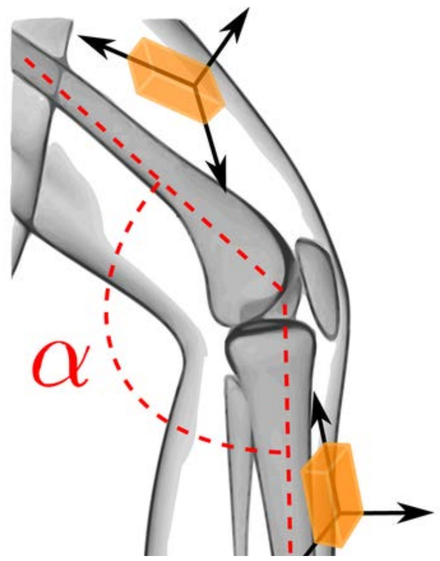


Figure 23 Mounting of IMU sensor

to find the local coordinates of joint-related axes. Also to note that the accuracy is limited by the precision with which the subject can perform the desired postures or motions.

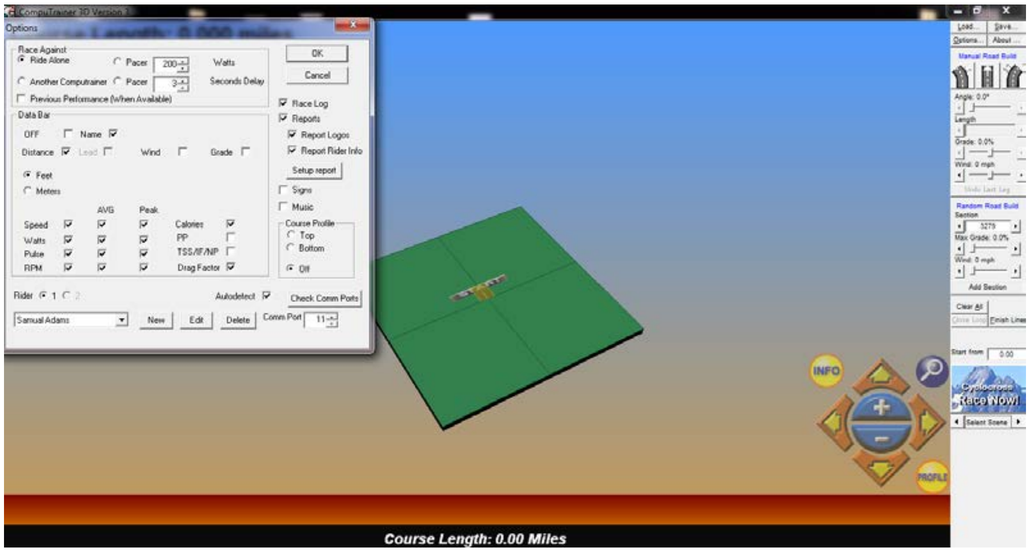
There are quite a few algorithms which have been suggested for IMU based knee angle estimation and defines the flexion/extension angle of the knee joint as the angle between the upper and lower leg along the main axis of relative motion, i.e., the knee joint axis [24-27]. Abduction /adduction and internal/external rotation which leads to three-dimensional knee joint angle as in [26-28] are ignored because they never exceed a range of ±10◦ [26, 29].

As mentioned above, IMU’s are attached such that one of the local coordinate axes is aligned with the joint axis. Integrating the difference between the upper and lower sensor’s angular rate (as discussed in the background research section) around that axis will give the desired angle. One of the issues is how to differentiate between the data values received from different sensors and compute the real time angle while cycling. This was solved by assigning each IMU sensor a unique sensor id, each data packet consists of a sensor id which the each packet of data received so it is easy to differentiate.

## 4.2 CompuTrainer™ System

### 4.2.1 Power Calculation

For the power calculation part from using the cycling ergometer, CompuTrainer™ is used to get the power output. CompuTrainer™ is a high performance indoor ergometer and trainer shown in Figure 25, with its programmable, interactive software CompuTrainer 3D provides almost unlimited capability to fashion our workouts according to one’s need. It is connected to the rear end of the bicycle as shown in Figure 25. It generates load (wattage) using load generator which can be configured using the software provided, and provides information like HR/RPM/Power (watts). The CompuTrainer™ system uses the bicycle rear wheel to drive a copper flywheel, spinning in the field of an electromagnet to simulate cycling conditions [30]. One of the main features is that one can ride virtualized courses of anything in the world. The power output provided by the CompuTrainer™ is available real time only if we are using the software provided by them shown in Figure 24. After each session the software provides a log file which is converted into a proper format using the java and the data is sent to the cloud using the appropriate time stamp. This part will be explained later in the next chapter.



Race Against: [✓ Ride Alone] Pacer 200 Watts; Another Computrainer, Pacer 3 Seconds Delay; Previous Performance (When Available) ☐

Data Bar: OFF ☐ Name ✓ Distance ✓ Lead ☐ Wind ☐ Grade ☐
[✓ Feet] Meters

| | | AVG | Peak | | |
|---|---|---|---|---|---|
| Speed | ✓ | ✓ | ✓ | Calories | ✓ |
| Watts | ✓ | ✓ | ✓ | PP | ☐ |
| Pulse | ✓ | ✓ | ✓ | TSS/IF/NP | ☐ |
| RPM | ✓ | ✓ | ✓ | Drag Factor | ✓ |

Race Log ✓ Reports ✓ Report Logos ✓ Report Rider Info ✓ Signs ☐ Music ☐
Course Profile: Top, Bottom, [✓ Off]
Rider [✓ 1] 2 Autodetect ✓
Samual Adams Comm Port 11

Figure 24 Screenshot of CompuTrainer™ 3D v3

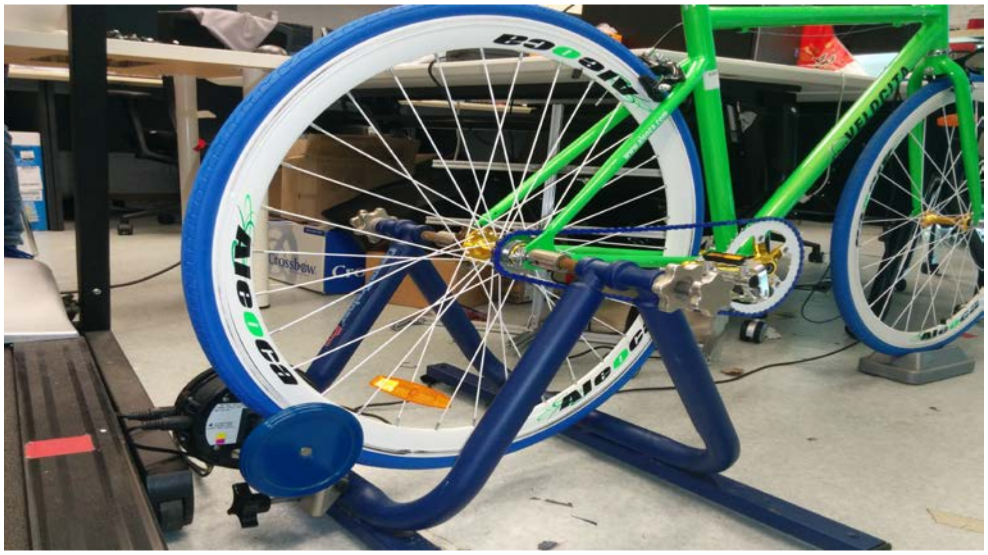


Figure 25 Cycling Ergometer

## 4.3 Implementation

Once the overall system was designed, it was time to go ahead with the implementation stage. If any issues were raised during the implementation stage, it was necessary to incorporate these issues in the design stage and apply suitable changes necessary for proper functioning of the whole system. This chapter will discuss in detail the implementation of each class.

The IMU system has a number of components which operate over a number of threads in the host PC. UML for the Computrainer™ and IMU system are shown in Figure 26 and Figure 30 respectively.

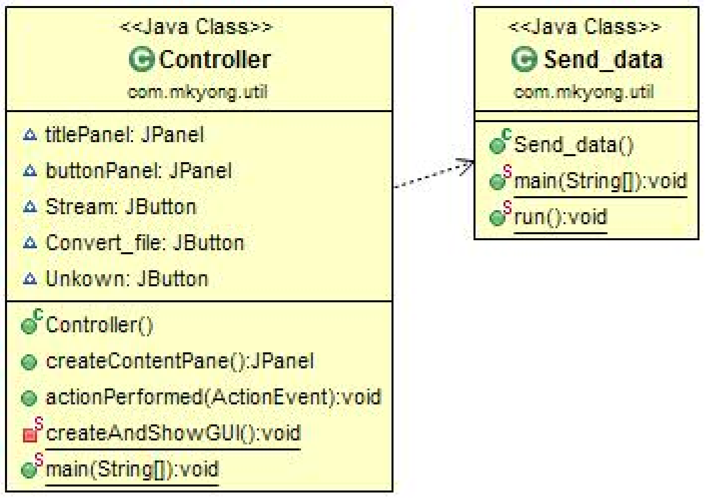
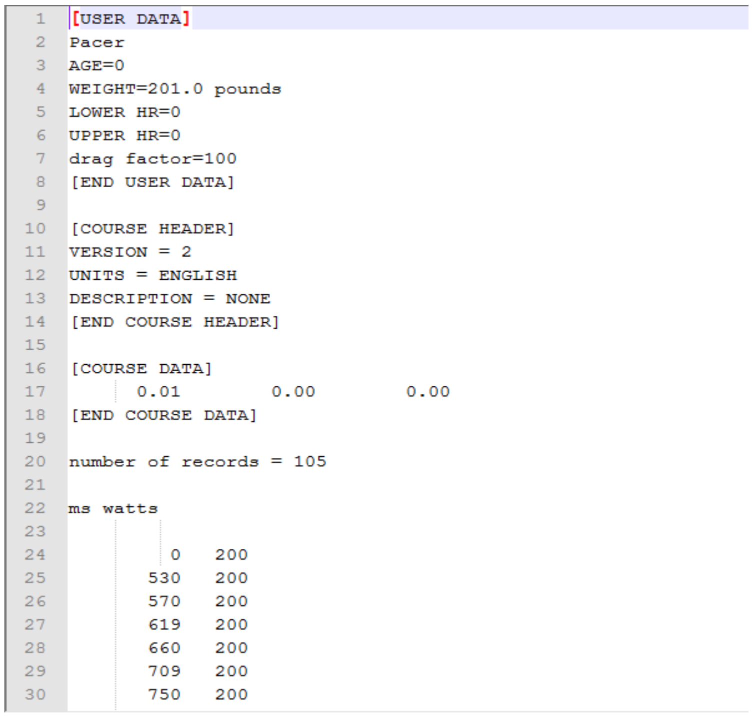


Figure 26 Class diagram of Computrainer™ system

### 4.3.1 Computrainer™ System

The Computrainer™ has two java classes namely *Controller* and *Send_data.*

- ***Send_data:*** The information gathered after using the Computrainer™ is saved in a .3dp file in the host system. That host file can be converted into a .txt format by using the software provided by the company. The output of that conversion is shown in Figure 27. The only required information from this file is the time and the power other information is redundant. The power output gathered from this file is sent to the server with the Unix timestamp. One of the issue as was disused in the previous section was about the synchronization. The log file only gives the time period between the different samples and there is no information about the start time of the session. As the information needs to be mapped with the data collected from other sensor this issue was very important to resolve. Various changes were made in the design but only one method gave promising results. Whenever the session ends the “.3dp” file is modified. So this information was used to get the Unix timestamp whenever the session ended, this was used to find the starting time when the session started using the time period (from the log file). The extracted was sent to the server using with their corresponding timestamp so that it is easy to map the information with the data received from other sensors. The flowchart in Figure 28 shows the overall flow of data extraction from the log file.



```
[USER DATA]
Pacer
AGE=0
WEIGHT=201.0 pounds
LOWER HR=0
UPPER HR=0
drag factor=100
[END USER DATA]

[COURSE HEADER]
VERSION = 2
UNITS = ENGLISH
DESCRIPTION = NONE
[END COURSE HEADER]

[COURSE DATA]
        0.01        0.00        0.00
[END COURSE DATA]

number of records = 105

ms watts

          0   200
        530   200
        570   200
        619   200
        660   200
        709   200
        750   200
```

Figure 27 Log file from Computrainer™

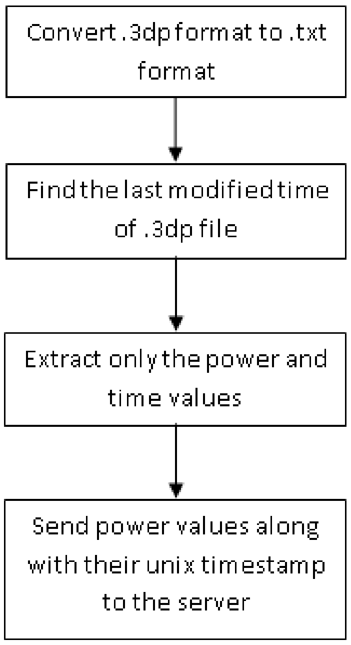

Convert .3dp format to .txt format
Find the last modified time of .3dp file
Extract only the power and time values
Send power values along with their unix timestamp to the server

Figure 28 Block diagram of data extraction from log file

- ***Controller:*** As the name suggests this class help to automate the process and make it easy for the user to interact using a GUI. Figure 29 shows the screen capture of the GUI. *Convert File* button automates the full process of converting the .3dp file to .txt and the data extraction. *Send data* button send all the data to the server.

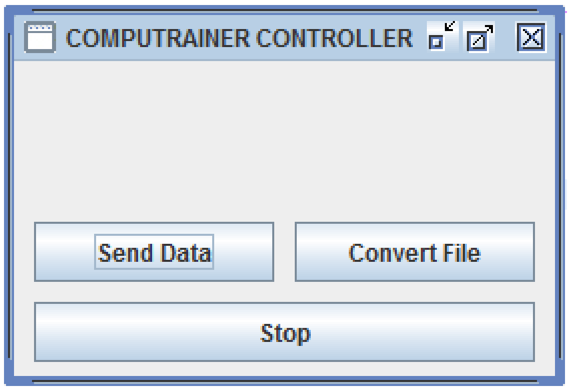


Figure 29 GUI for Computrainer™ controller

### 4.3.2 IMU System

The class diagram for the IMU system is shown in Figure 30.The class diagram shows the dependencies between the various classes.

- ***CalculQuat:*** This class is responsible for all the backend calculation required to process the raw data samples received from the IMU sensor. The ListenComPort class receives data from the IMU sensor using the receiver connected to COM Port depending upon the frequency at which the transmitter is sending.

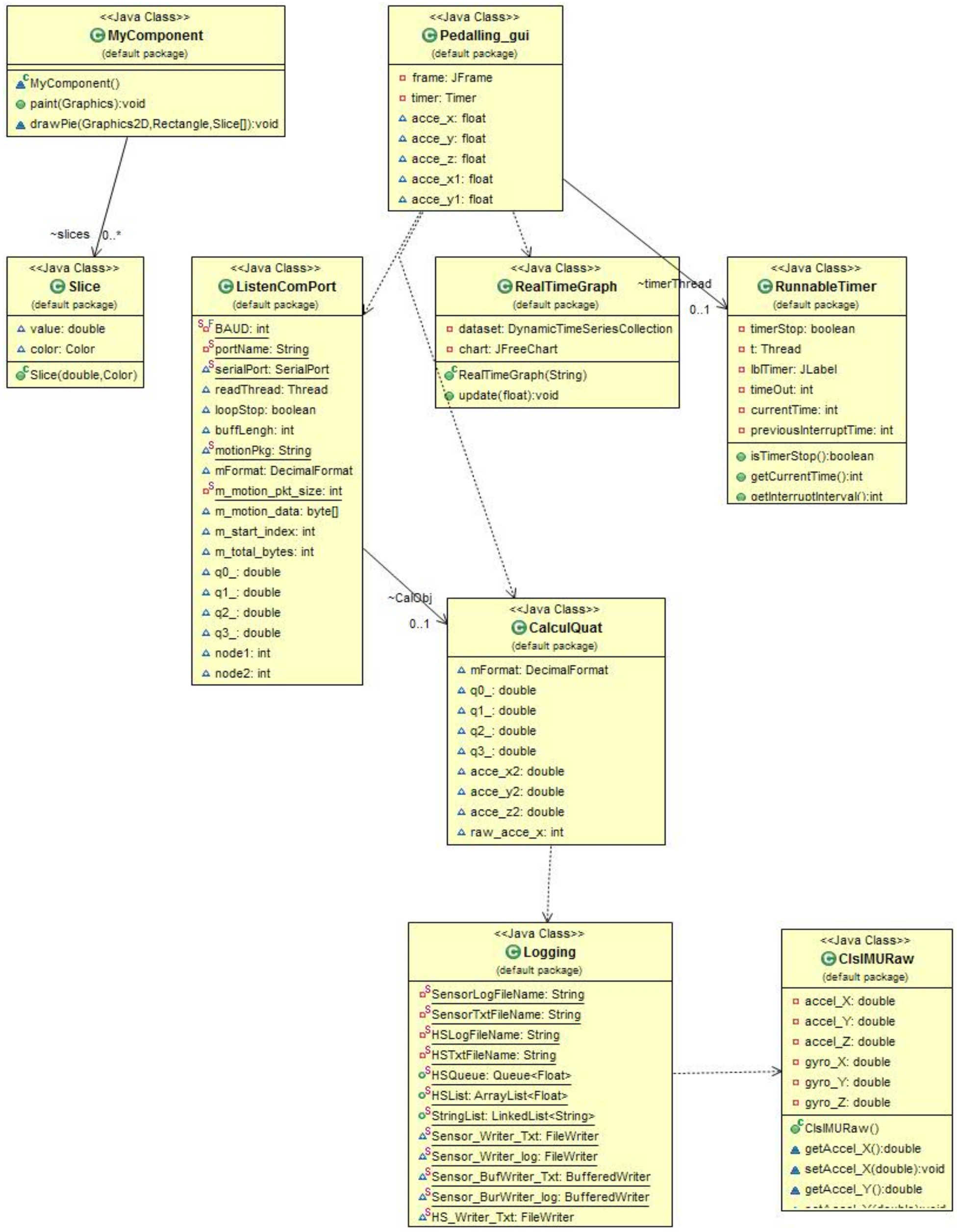
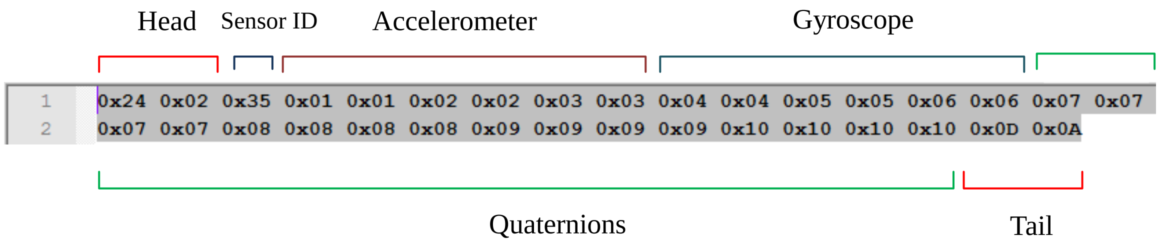


Figure 30 Class diagram of IMU system

At the receiver side, data is captured wireless unit and is sent to the microcontroller and these are then forwarded to the COM Port using UART protocol. The complete data received is shown in Figure 31 below. The figure shows one full packet containing accelerometer, gyroscope and quaternions values of length 32 bytes including 4 bytes for header and tail, 6 bytes each for accelerometer and gyroscope values (2 bytes each for x,y,z direction) and the remaining 16 bytes are for Quaternion values. Size of each parameter can change based on the sensitivity value. The packets received are decoded and different sensor readings are separated out. Using the sensitivity values from the data sheet, the reading obtained are calculated accordingly and the final values are obtained. These values are then used to calculate the angle values from the packets received from the 2 IMU sensors as discussed in the previous section.



```
0x24 0x02 0x35 0x01 0x01 0x02 0x02 0x03 0x03 0x04 0x04 0x05 0x05 0x06 0x06 0x07 0x07
0x07 0x07 0x08 0x08 0x08 0x08 0x09 0x09 0x09 0x09 0x10 0x10 0x10 0x10 0x0D 0x0A
```

Figure 31 Data packet from the IMU sensor

- ***Logging:*** Logging class is used to log the data from the 2 sensors. This class saves raw acceleration and raw gyroscope and calculated accelerometer and gyroscope values. These values are logged with their timestamp so that the user can further analyze the data.
- ***Pedalling_gui:*** The GUI for the pedaling was the last part of the system to be implemented although it did not have impact on the operation correctness of the system, but its functionality was equally important. The GUI consist an instance of *ListenComPort* Class and it is using this it receives the data from the IMU sensors using the COM Port. The GUI implemented can be seen in the Figure 32. The GUI was designed to be as simple as possible, allowing the user to perform the following actions:

1. To allow the user to start receiving the data from the COM Port.
2. To *start*, *stop* and *reset* the count i.e. the number of pedal strokes per min.
3. A Timer.
4. The real time graph for acceleration, angle and gyroscope value (or whichever is required by the user)
5. The real time angle measurement value.
6. And to determine if the flexion angle was good or bad the GUI shows whether it's a Good angle or a Bad angle.

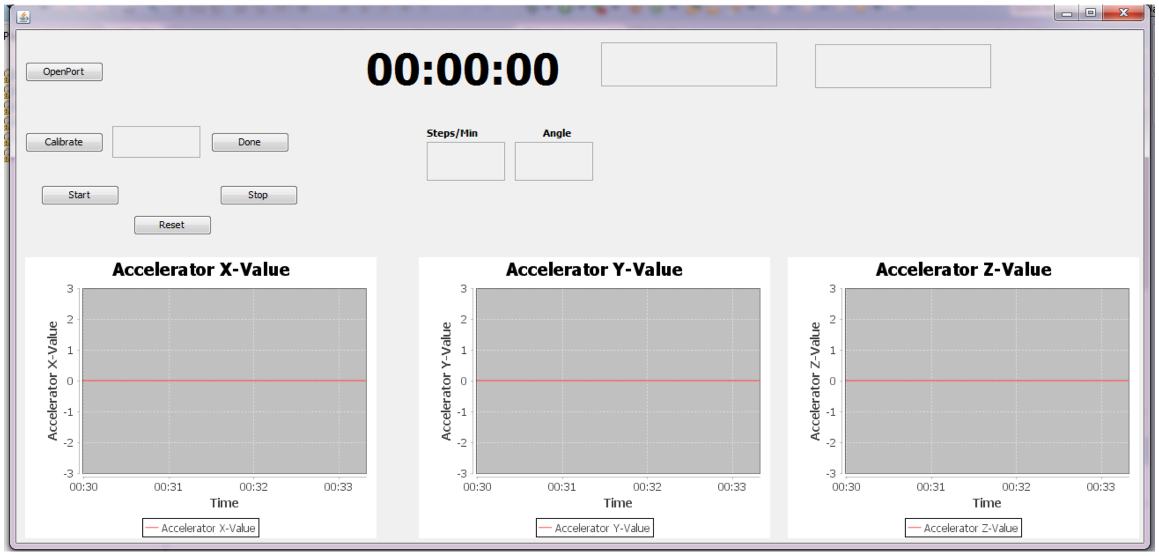


Figure 32 GUI for IMU system

Timer is implemented using the *RunnableTimer* class as shown in the class diagram in Figure 32. The GUI class was implemented using Java AWT and Swing which is a GUI widget toolkit for Java. The interactivity between the classes and GUI *actionListerns* are used. GUI helps in separating the display (GUI) from the Model (*CalculQuat* class) which allow the user to add additional features to the GUI without changing the Model.

# Chapter 5

# Testing and Results

## 5.1 Power

After the implementation stage it was necessary to integrate the IMU system and CompuTrainer™ system with the overall system to check its correctness. Before system integration all the sub-systems were tested individually to check if the correct data is reaching the server or not. It was easy to check as the data received at the server side can be compared with the log file on the host system. Dummy IMU and EMG data was also used in order to visualize the functionality of the whole system. Figure 34 shows the server visualization which shows the power received from the EMG sensor of 5 different muscles and the power output from the CompuTrainer™. The circular plot gives the power output from the 5 muscles corresponding to the rotation angle of the pedal which can be seen in Figure 33.

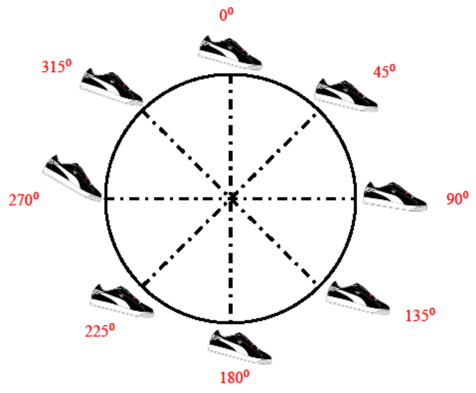
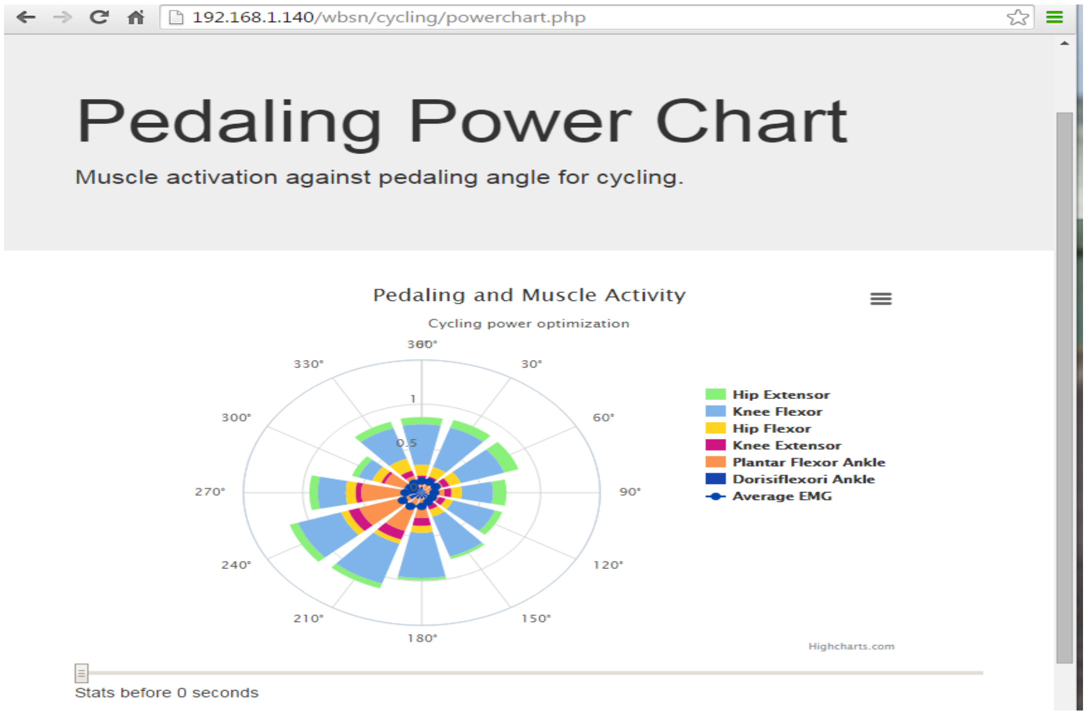


Figure 33 Pedaling angle



192.168.1.140/wbsn/cycling/powerchart.php

Pedaling Power Chart

Muscle activation against pedaling angle for cycling.

Stats before 0 seconds

Figure 34 Server Visualization

The power plot was divided into 12 parts of 30 degree each. The data received from IMU, EMG and Computrainer™ was mapped according to the timestamp and were shown on the server visualization.

## 5.2 Knee Angle

The next part of the testing was to measure the correctness of angle calculated by the IMU sensor. The data gathered for different leg postures is summarized in Table 1 below. The table shows the angle computed by the IMU sensor and the correct angle which was measured by using angle measuring device. The percentage error for each of the recorded data is also shown in Table 1.

| S No. | Actual Angle | Calculated Angle | Error (%) |
|---|---|---|---|
| 1 | $91^0$ | $89^0$ | 2.19 |
| 2 | $156^0$ | $160^0$ | -2.56 |
| 3 | $82^0$ | $79^0$ | 3.65 |
| 4 | $180^0$ | $178^0$ | 1.11 |
| 5 | $78^0$ | $76^0$ | 2.56 |
| 6 | $34^0$ | $35^0$ | -2.94 |
| 7 | $124^0$ | $124^0$ | 0 |

Table 1 Knee flexion angle for different postures

From the table it can be seen that there is an error rate of $\pm$ 3% on an average. The angle measured in a still position is not as intricate when compared to cycling. While cycling the angle is changing at a very fast rate so the angle calculated real time will also depend upon the frequency at which IMU sensor will send the data. The variation of calculated knee angle for a cycling session is shown in Figure 35.

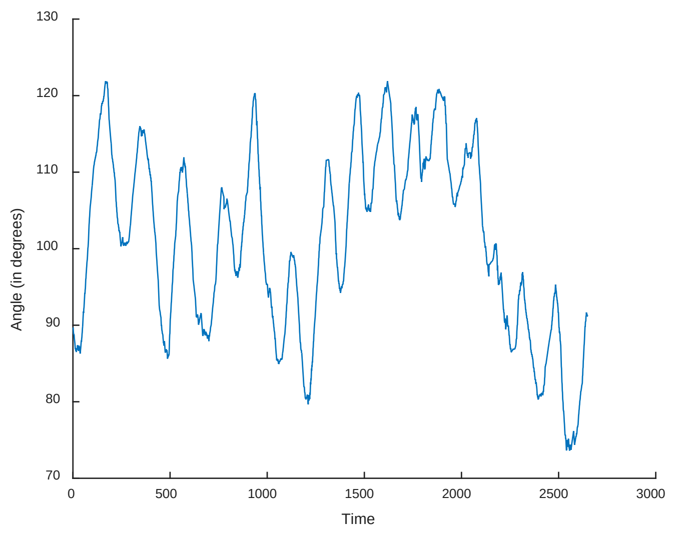


Figure 35 knee flexion angle vs time

For the next set of tests participants were asked to cycle for 5 minutes at a normal speed. The Computrainer™ 3D software provided with the Computrainer™ was used to customize the cycling track, set the grade. The software provides a wide range of option to customize according to user need, a screenshot of the software is shown in Figure 36. The orientation and position of sensor on the knee is shown in Figure 37 which can be identified by red box. After each session the power output was extracted from the *Computrainer*™ system and the angle values were obtained from the IMU system and the values were mapped based on their timestamp. A screenshot of the GUI of the IMU system is shown in Figure 38. The data collected has been summarized in Table 2.

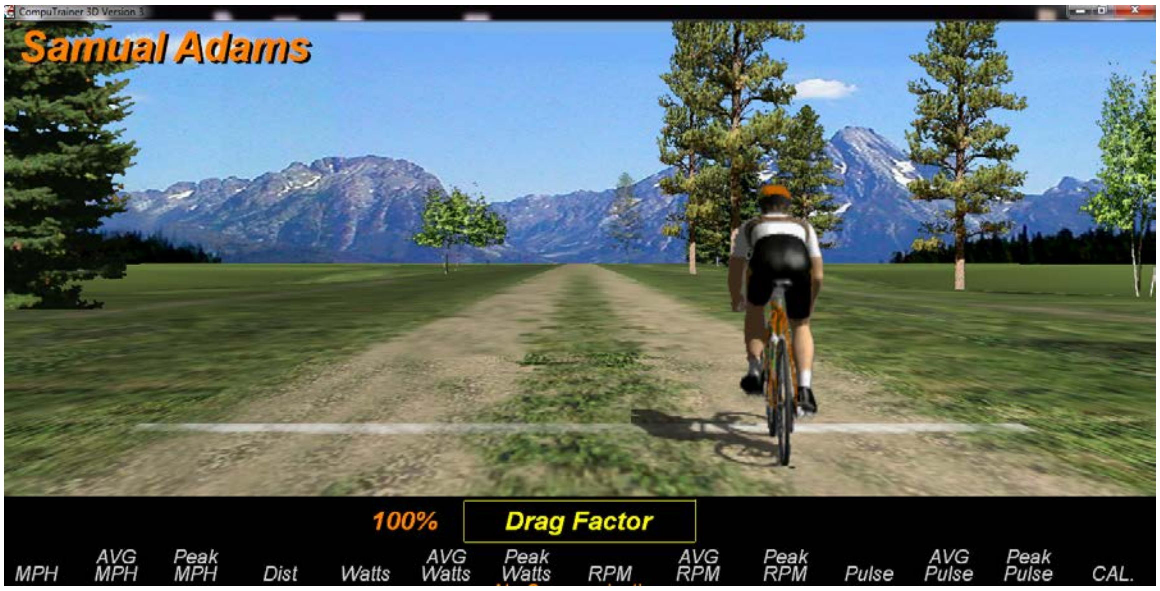


Figure 36 Computrainer™ 3D software

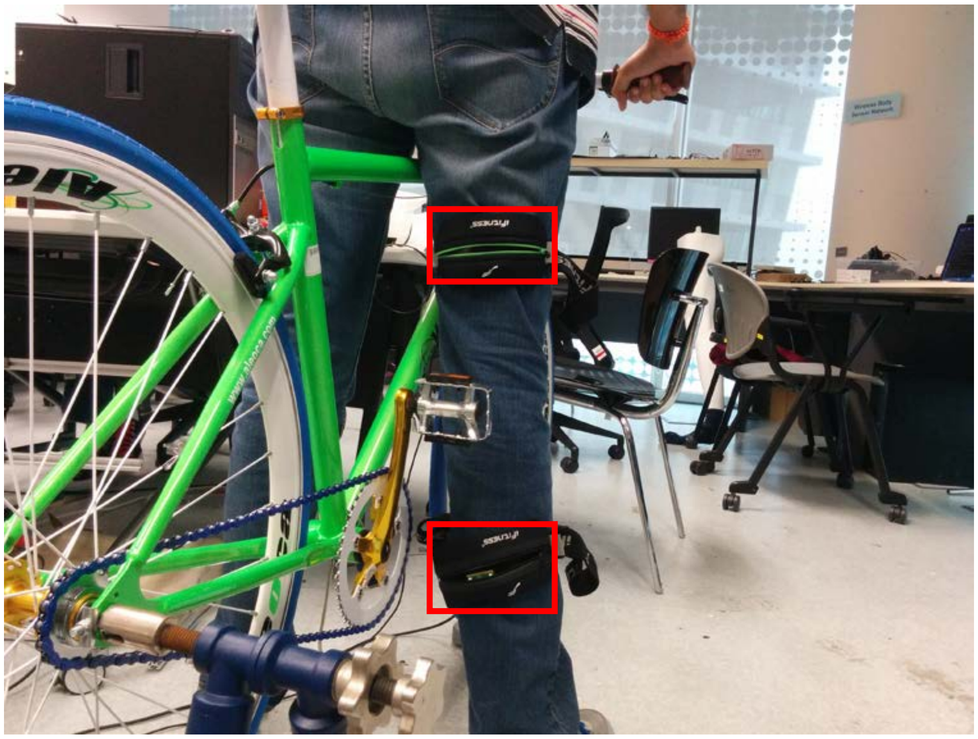


Figure 37 Sensor Position

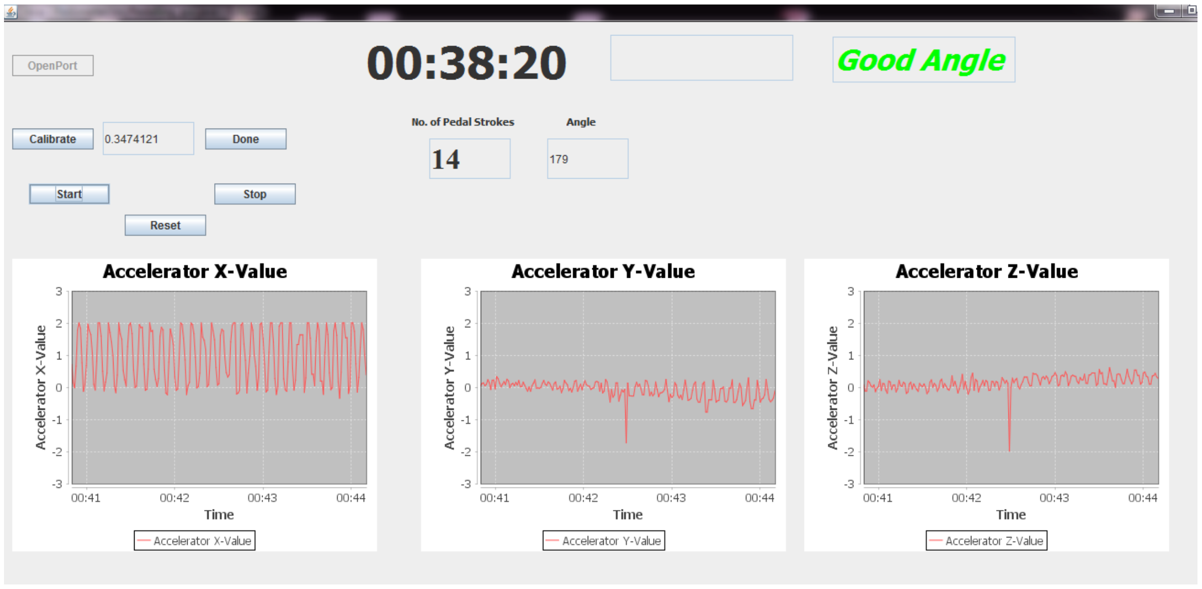


Figure 38 Pedaling GUI

| Participant | Grade = 0% | Grade = 1% | Grade = 2% | Grade = 4% |
|---|---|---|---|---|
| Maximum Power | 81 W | 81 W | 71 W | 46 W |
| Average Power | 56.31 W | 50.20 W | 48.77 W | 38.05 W |
| Maximum Flexion Angle | 125.50$^0$ | 121.49$^0$ | 119.57$^0$ | 120.49$^0$ |
| Minimum Flexion Angle | 73.61$^0$ | 74.66$^0$ | 82.64$^0$ | 81.03$^0$ |
| Average Flexion Angle | 100.27$^0$ | 96.17$^0$ | 98.87$^0$ | 98.62$^0$ |

Table 2 Power and angle output for different grade

The knee flexion angle vs time duration for different grades are given in Figure 39.

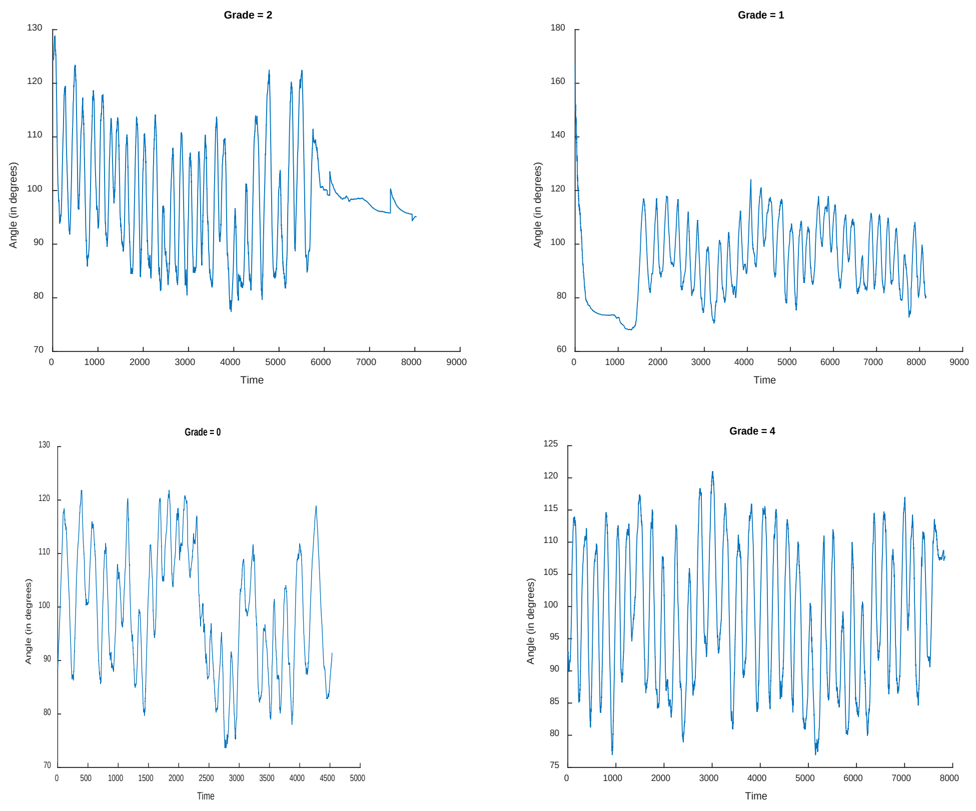


Figure 39 Flexion knee angle vs time

The system performed well in the testing situations described above and the overall system performed as expected. Each part of the system was fully functional and the system was also tested by a cycling training professional. He described that the data gathered by the system can be very useful for the cyclists. The system will continue to develop and improve was the system was capable of performing well in each of the testing scenarios described, acting as expected in each. Each part of the functional specification was fully implemented and tested.

# Chapter 6

# Future Work and Conclusion

Although we have tested the basic functionality of the system in various testing scenarios and it was able to deliver the performance we expected but there is an always a scope for improvement. The development will continue to take place and it will make the system more robust and improve the functionality.

There are quite a few ways by which the system can be improved, one of the issue which is still there with the system was the power output from the CompuTrainer™ was only available after the end of each session and not real time. Real time power can only be visualized in real time by using the software provided by the company. But with the newer version of Computrainer™ it is possible to get the power output values real time and integrate it with the API's developed. This will enable the user to analyze their performance more efficiently in real time. Furthermore there is a long term goal of developing in-house developed sensors (EMG) can ease the implementation of these sensors.

To conclude we have managed to develop and implement a fully functioning IMU and Computrainer™ system which successfully gathers and process the data using the algorithm before sending it to the server. Overall system worked as desired and provided the functionality which was discussed during the design phase of the project. This tool can be used by cycling professionals around the world to improve their pedaling efficiency, reduce injuries and maximize their performance. Development of the system will continue with further testing and improvements in future.

# References

1. Hassan Ghasemzadeh, V.L., Roozbeh Jafari, *Sport Training Using Body Sensor Networks: A Statistical Approach to Measure Wrist Rotation for Golf Swing*. The Fourth International Conference on Body Area Networks (BodyNets), 2009.
2. Skog, I., J.O. Nilsson, and P. Handel. *An open-source multi inertial measurement unit (MIMU) platform*. in *Inertial Sensors and Systems (ISISS), 2014 International Symposium on*. 2014.
3. Muro-de-la-Herran, A., B. Garcia-Zapirain, and A. Mendez-Zorrilla, *Gait analysis methods: an overview of wearable and non-wearable systems, highlighting clinical applications*. Sensors (Basel, Switzerland), 2014. **14**(2): p. 3362-3394.
4. Seel, T., J. Raisch, and T. Schauer, *IMU-based joint angle measurement for gait analysis*. Sensors (Basel, Switzerland), 2014. **14**(4): p. 6891-6909.
5. Addingrefs, *A simple diagram showing an unfolding suprapatellar synovial recess in a flexing knee*, in *InkSpace*, Knee-unfolding-recess-diagram.svg, Editor. 2008, WikiMedia: WikiMedia Commons.
6. Cheng, P. and B. Oelmann, *Joint-Angle Measurement Using Accelerometers and Gyroscopes A Survey*. IEEE Transactions on Instrumentation & Measurement, 2010. **59**(2): p. 404-414.
7. Hambly, K. *Cycling for Knee Rehabilitation*. 2009; Available from: http://www.cartilagehealth.com/cycling.html.
8. Hug, F. and S. Dorel, *Electromyographic analysis of pedaling: A review*. Journal of Electromyography and Kinesiology, 2009(2): p. 182.
9. Ashe, M.C., et al., *Body position affects performance in untrained cyclists*. British Journal of Sports Medicine, 2003. **37**(5): p. 441.
10. Ergotec, *Vorbau_BEschriftung.indd*, in *Adobe InDesign*, S. Height, Editor.
11. Veal, M., *BikeDynamics - Bike Fitting Specialists*, in *DIY Dynamic Bike Fitting Using Simple Tools and Observations*. 2015, BikeDynamics Ltd.
12. Paton, C.D. and W.G. Hopkins, *Tests of Cycling Performance*. Sports Medicine, 2001. **31**(7): p. 489-496.
13. Faria, E.W., *Recent advances in specific training for cycling : review article*. International SportMed Journal, 2009(1): p. 16.
14. Faria, E.W., D.L. Parker, and I.E. Faria, *The Science of Cycling: Factors Affecting Performance – Part 2*. Sports Medicine, 2005. **35**(4): p. 313-337.
15. STMicroelectronics, *STM32F401xD STM32F401xE Datasheet*. 2015.
16. Nordic Semiconductor, *nRF24L01 Single Chip 2.4GHz Transceiver Product Specification*. nRF24L01 Product Specification.
17. InvenSense, *MPU-9250 Product Specification, Revision 1.0*. MPU-9250 Datasheet, 2014.

18. Eclipse. *Eclipse Home Page*. Eclipse 26 June 2015]; Available from: https://eclipse.org/.

19. Auawise, Y.A.s., *Axes to control the attitude of a plane*, Y.A. Corrected.svg, Editor. 2010, Wikipedia. p. An image showing all three axise.

20. theboredengineers. *The quadcopter : how to compute the pitch, roll and yaw*. 2012 [cited 2015; Available from: http://theboredengineers.com/2012/09/the-quadcopter-get-its-orientation-from-sensors/.

21. Continum, T.C. *Arduino IMU: Pitch & Roll from an Accelerometer*. 2012; Available from: http://theccontinuum.com/2012/09/24/arduino-imu-pitch-roll-from-accelerometer/.

22. Cutti, A.G.A.P.M.A.A., *'Outwalk': a protocol for clinical gait analysis based on inertial and magnetic sensors.* Medical & Biological Engineering & Computing, 2010. **48**(1): p. 17-25.

23. Ferrari, A.A.G.P.M.M.A.A., *First in vivo assessment of "Outwalk": a novel protocol for clinical gait analysis based on inertial and magnetic sensors.* Medical & Biological Engineering & Computing, 2010. **48**(1): p. 1-15.

24. Kun, L., et al. *Ambulatory measurement and analysis of the lower limb 3D posture using wearable sensor system*. in *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*. 2009.

25. Seel, T., T. Schauer, and J. Raisch. *Joint axis and position estimation from inertial measurement data by exploiting kinematic constraints*. in *Control Applications (CCA), 2012 IEEE International Conference on*. 2012.

26. Favre, J., et al., *Ambulatory measurement of 3D knee joint angle.* Journal of Biomechanics, 2008. **41**: p. 1029-1035.

27. Cooper, G., et al., *Inertial sensor-based knee flexion/extension angle estimation.* Journal of Biomechanics, 2009. **42**: p. 2678-2685.

28. Favre, J., et al., *A new ambulatory system for comparative evaluation of the three-dimensional knee kinematics, applied to anterior cruciate ligament injuries.* Knee Surgery, Sports Traumatology, Arthroscopy, 2006. **14**(7): p. 592-604.

29. *Gait analysis; normal and pathological function, 2d ed*. 2010, Book News, Inc.

30. RacerMate, *CompuTrainer 3D Software Users Guide.* 2009.