IoT BASED IMU SENSOR NETWORK FOR HUMAN ARM POSE ESTIMATION

JAHANGIR HASSAN KHAN

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

IoT BASED IMU SENSOR NETWORK FOR HUMAN ARM POSE ESTIMATION

JAHANGIR HASSAN KHAN

RESEARCH REPORT SUBMITTED TO THE FACULTY OF ENGINEERING UNIVERSITY OF MALAYA, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING (MECHATRONICS)

2019

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Jahangir Hassan Khan

Matric No: KQF 170001

Name of Degree: MASTER OF MECHATRONICS ENGINEERING

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

IoT BASED IMU SENSOR NETWORK FOR HUMAN ARM POSE ESTIMATION Field of Study: Engineering Sciences, Mechanical Engineering, Robotics

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

Abstract

Human arm pose estimation or tracking is becoming an increasing sought after field of research due to the multitudes of uses it offers. With the start of Industry 4.0, Internet of Things (IoT) concept has made wireless data transfer into reality and hence, remote monitoring and control can be performed relatively inexpensive and portable. The applications can range from Virtual Reality gaming to training industrial robots as well as controlling surgical robots and stroke rehabilitation. For example: In the past, patients had to go periodically for rehabilitation to clinics and hospitals, can now do those same exercises at home using sensors that can send progress data over the internet to their doctors who can analyse the data and provide feedback. However, this technology is still not well-established and thus requires more research. This motivates the current study to work with this research topic. The objective of this project is to design and build an IoT based sensor system that can send its data wirelessly to a computer, which can then analyse the data and estimate the position of the human arm at remote location. The sensor used in this project is the BNO055 Inertial Measurement Unit, which is a low cost sensor incorporating a gyroscope, accelerometer and magnetometer to provide orientation and acceleration data. The medium for wireless data transfer is a Wi-Fi enabled Node MCU micro-controller. The sensor node will transfer data to a computer via Wi-Fi using a custom designed network protocol. A multi-threaded program running on the computer will perform Forward Kinematics on the wirelessly transferred data using an arm model created using the Denavit-Hartenberg convention. The Forward Kinematics approach has great advantage such as being easy to program and its faster computational speed, as compared to the conventional arm pose estimation algorithms. By using the developed arm pose tracking program, several important parameters such as positions of the elbow and wrist with reference to the shoulder joint, moving distance, angular movement speed and movement pattern of arm are successfully measured and estimated. The results of this

research show that for basic movements such as shoulder flexion, contraction or elbow flexion and contraction, the accuracy of tracking distance is more than 80 percent. For complicated movements like tracing shapes the accuracy ranges from 60 to 70 percent. For the purpose of tracking arm movements of stroke patients, the accuracy is adequate since the arm movement exercises for stroke rehabilitation consist of basic arm movements for which the proposed design has high accuracy. The proposed system also has high enough sampling and transmission frequency to accurately track fast arm movements which makes it ideal for the purpose of arm position estimation.

ABSTRAK

Anggaran postur lengan manusia sedang menjadi satu bidang penyelidikan yang semakin popular kerana aplikasinya yang pelbagai. Dengan permulaan konsep Industri 4.0 dan Internet Benda, pemindahan data tanpa wayar telah menjadi realiti, maka pemantauan dan kawalan terpencil boleh dijalankan dengan mudah dan secara mudah alih. Aplikasinya pelbagai, seperti permainan Realiti Maya, latihan robot lengan industri, malahan juga untuk mengawal robot penbedahan dan robot pemulihan strok. Sebagai satu contoh: Pada seketika dahulu pesakit yang mengalami strok perlu menghadiri sesi pemulihan di hospital dan klinik dari masa ke semasa. Namun, sekarang pesakit-pesakit tersebut dapat menjalankan latihan yang sama di rumah sendiri dengan menggunakan sensor yang boleh menghantar data kemajuan seseorang pesakit kepada para doktor memalui internet. Para doktor tersebut kemudiannya boleh membuat analisis pada data yang diterima dan sejurusnya memberikan nasihat yang berpatutan. Namun demikian, bidang teknologi ini masih pada tahap pucuknya, dan memerlukan penyelidikan yang selanjutnya. Hal ini memberikan motivasi untuk menjalankan usaha penyelidikan yang digaris besarkan di dalam laporan projek ini. Projek ini mempunyai objektif untuk mereka bentuk dan membina satu system sensor Internet Benda yang boleh menghantar datanya secara tanpa wayar ke sebuah computer, yang kemudian boleh menganalisa data tersebut dan memberikan satu anggaran posisi lengan tangan manusia. Sensor yang digunakan dalam projek ini ialah Unit Pengukur Inertia BNO055 yang merupakan satu sensor berkos rendah yang mengandungi giroskop, akselerometer, dan magnetometer. BNO055 mampu digunakan untuk memberikan data orientasi dan pecutan. Medium yang digunakan untuk menghantar data secara tanpa wayar ialah noda MCU mikro-kawalan yang dilengkapkan dengan Wi-Fi. Noda sensor ini boleh menghantar data ke computer melalui Wi-Fi dan protocol rangkaian yang direka bentuk secara khas. Sebuah program yang dijalankan dalam computer pula akan melalukan pengiraan terjemahan ke hadapan pada data yang

diterima, dengan menggunakan konvensi Denavit-Hartenberg. Kaedah ini mempunyai kelebihan iaitu senang diprogramkan, dan lebih pantas jika dibandingkan dengan kaedah algoritma anggaran posisi lengan tangan manusia yang biasa. Dengan menggunakan program penjejakan posisi lengan tangan ini, beberapa parameter yang penting boleh diukurkan dan dianggarkan. Contohnya, posisi siku dan pergelangan tangan dari bahu, jauh pergerakan, kelajuan sudut pergerakan, dan corak pergerakan tangan. Hasil penyelidikan ini menunjukkan bahawa untuk pergerakan asas seperti pergerakan bahu atau siku, ketepatan jarak pergerakan yang diukur adalah lebih daripada 80 peratus. Bagi pergerakan yang lebih rumit pula peritus ketepatannya adalah dalam kadar 60 sehinggan 70 peratus. Bagi tujuan mengukur pergerakan tangan pesakit strok, tahap ketepatan ini adalh mencukupi, kerana latihan pergerakan tangan untuk pemulihan strok hanya terdiri daripada pergerakan tangan yang asas di mana alat ini mempuyai ketepatan yang tinggi. Sistem yang dicadangkan juga mempunyai kadar pensampelan dan kadar penghantaran yang cukup tinggi untuk mengukur pergerakan tangan yang cepat. Dengan itu, hal ini menjadikan system yang direka bentuk ini sesuai untuk digunakan bagi tujuan penganggaran posisi tangan.

ACKNOWLEDGEMENTS

Firstly, I am grateful to Allah, the Merciful and the Almighty for all His Blessings without which I wouldn't be where I am today.

I would also like to express my gratitude to my supervisor Dr Khoo Shin Yee, at Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, who has provided me with valuable insights, suggestions and guidance for the fulfilment of this project. I am profoundly grateful for his ever willingness and availability to help me whenever needed, and to guide me whenever I encounter a problem.

I want to thank my Parents and family for their constant support and motivation which got me through all the roadblocks in my life. And for being my safety net when things weren't working out in life.

I would also like to thank all the amazing friends I made during my study and stay in Malaysia for all their support and help along the way, and from whom I learnt a lot from. Gratitude and appreciation also go to all my lecturers in this university, who made this learning experience all the more enjoyable.

Table of Content

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DOF	:	DEGREE OF FREEDOM
DH	:	DENAVIT-HARTENBERG
IMU	:	INERTIAL MEASUREMENT UNIT
IP	:	INTERNET PROTOCOL
MCU	:	MICRO-CONTROLLER UNIT
SMD	:	SURFACE MOUNT DEVICE
VR	:	VIRTUAL REALITY
GUI	:	GRAPHICAL USER INTERFACE
PCB	:	PRINTED CIRCUIT BOARD

LIST OF APPENDICES

Chapter1: Introduction

1.1 Introduction

Human arm pose estimation refers to the ability of a software with the help of sensors to estimate or track the movements of a human arm. Human arm position or pose tracking has recently garnered much research effort. Teaching a computer to recognise and tracking human arm positions have widespread applications. For example, Gaming industry has always shown great interest in this field due to the shift in focus of customers from traditional controller games to Virtual Reality games. Virtual Reality games aim to provide an extremely immersive gaming experience. This experience relies on the customer to be able to control their game characters using their own limbs like arms. This requires the VR software to be able to track human limb movements. In industry arm estimation and tracking together with artificial intelligence is being used to teach robotic manipulators complicated movement sets that previously were very difficult to program. In the field of medicine, human arm tracking is being sought after as a viable tool to control surgical robots for when the surgeon is not physically present.

1.2 Problem Statement

In the past the only sensors being used in home rehabilitation of stroke patients were accelerometers. These sensors were used to measure activity count of daily activities. For a long time, this was the main rubric for improvement of stroke arm during home rehabilitation. In recent years, inertial measurement units (IMU) have been used to develop systems to more accurately measure range of motion and arm positions. But these systems have been developed more for the purpose of gaming and industry. Those that are developed for medical studies are not portable or wireless. In some cases, the human arm tracking system is portable but require the use of storage devices to log data which then have to be periodically uploaded so that analysis can be done. This deters active

monitoring and require technical knowledge on the part of the patient using such system at home. There is a need to develop a wireless arm tracking system for home rehabilitation monitoring for stroke patients that is portable, easy to use and more importantly low cost.

1.3 Objectives of Research

The objectives of this research are as follows:

1. To develop a Wireless Internet of Things sensor system using the BNO055 Inertial Measurement Unit (IMU)
2. To evaluate the performance of the human arm pose tracking program with forward kinematics algorithm.

Chapter 2: Literature review

2.1 Use of sensors in medical rehabilitation

Stroke is the leading cause of morbidity and Mortality in the world. There are almost 50000 deaths per year in industrialized countries. Around half the survivors are left with disabilities. These impairments can range from loss of strength, change in muscle tone and loss or reduced movement. 30 % of the patients are left with loss of motor and sensory function in their upper limbs (Benjamin et al., 2017; Staff, 2019). In United States alone, 600000 individuals have first-ever stroke. Fifty percent of survivors are older than 60 years and 26 percent can't perform daily life activities. This results in decreased activation and movement that can greatly affect everyday life (Oujamaa et al., 2009).

Rehabilitation is the main recourse for stroke patients. Rehabilitation involves exercise routines that help the patient get back some of the movement and skills that they may have lost as a result of stroke. One of the therapies involved in rehabilitation is the range of motion therapy. This session involves certain exercises and treatments that ease the tension in the muscles, thereby increasing range of motion (Oujamaa et al., 2009).

A study was conducted on the monitoring of use of upper extremities such as arm after transition from post stroke rehabilitation to discharge (Rand & Eng, 2015). This study monitored the functional abilities and use of affected arm through a period of 12 months after discharge. The premise of this study was that in previous studies based on the results of four weeks of post stroke therapy a recovery time of 8 months was predicted based on certain measurable parameters such as range of motion and muscle strength. But there was a disparity between the amount of daily use predicted and the actual amount of daily use. This disparity was due to the lack of studies monitoring daily use after post stroke rehabilitation patients are discharged. Traditional methods of monitoring arm use were

limited number of observations, self-reporting by patients using Motor activity logs and the REACH (Rating of everyday arm use in the community and home) scale. This study proposed use of accelerometers to accurately measure affected arm daily use and therefore correlate the predicted recovery time with the actual recovery time. In this study, patients who suffered stroke, were admitted to a rehabilitation program and were of age 19 and above volunteered to be a part of this study. These subjects wore an accelerometer on each wrist as well as an accelerometer on the hips. The hip accelerometer is to measure the hip movement which can be used to identify arm movement due to walking. This can thus be eliminated to give arm movement from arm related activities. To measure activity the accelerometer integrates acceleration over windows of 15 seconds. Daily records were visually examined to insure no irregular or unexplainable activity was recorded. Daily activity of each arm was measured by taking the total activity of each arm in three days and dividing that by three. Daily use was also measured using self-questionnaires such as Motor Activity Logs (MAL). This study showed that activity of the stroke affected arm was only 35 percent of the daily activity of the non-affected arm (Rand & Eng, 2015).

Gebruers et al. (2010) also discusses the use of accelerometers to measure the daily activity count of affected vs non affected arms in a stroke patient (Gebruers et al., 2010). Their study also brings attention to the traditional method of measuring daily activity which is via questionnaires such as MAL. The main disadvantage of such methods is that they depend on the patient's honesty and ability to remember the activities done in a day, which represents a subjective evaluation. Accelerometers with threshold filters to filter out erratic movements have been used in previous studies to measure or log daily activity of arms of patients at home. These accelerometers have built in storage which stores the activity data. Another use of accelerometers with stroke patients was to measure the rate

of finger tapping. The rate of finger taps was measured by mounting a tri-axial accelerometer onto the index finger using a Velcro strap (Calautti et al., 2006).

Patel et al. (2012) explored the use of wearable sensor systems for use in monitoring rehabilitation process of patients (Patel et al., 2012). Wearable sensor systems have the potential to help patients in areas where there is limited access to healthcare or medical services are far away. In united states 20 percent of patients live in rural areas whereas only 9 percent of the doctors work in rural areas.

As shown in the Figure 2.1, wearable sensors can monitor physiological as well as movement data of the patients and transmit that data wirelessly to their physician, family members as well as emergency services.

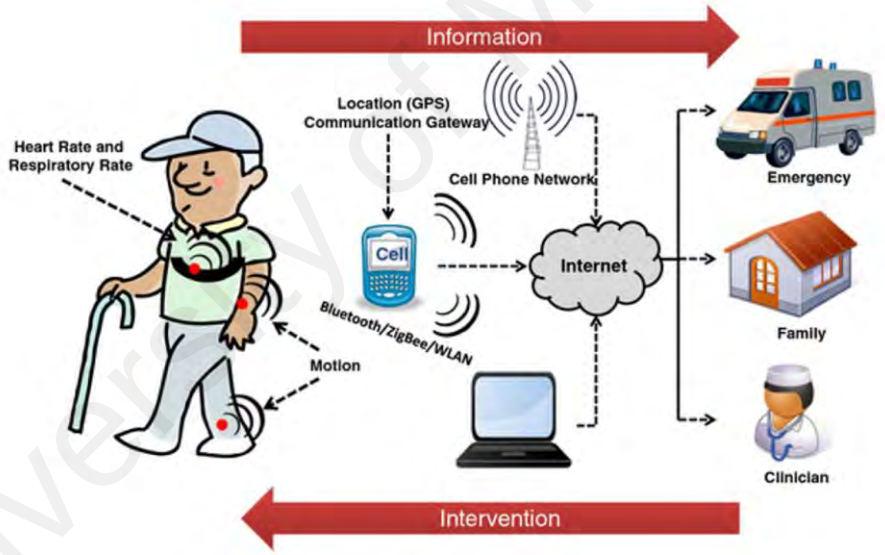


Figure 2.1: Wearable Sensor Monitoring System (Patel et al., 2012)

Previously wearable sensor systems were big and cumbersome which made it uncomfortable and therefore not practical for long term use. Technology over the years have advanced enough that sensor systems have been made smaller, wireless, easy to use and comfortable to wear. For example, Figure 2.2 shows a very small and flexible sensor system that incorporates a micro-controller, an ECG sensor and a radio chip on a single PCB.

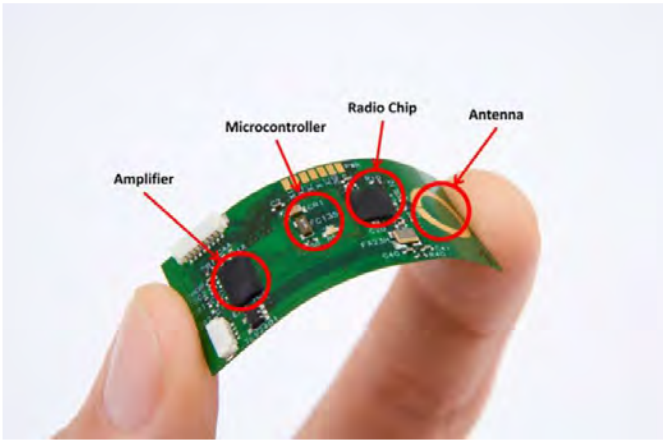


Figure 2.2: Wireless Micro-controller with ECG Sensor (Patel et al., 2012)

Sensors can also now be incorporated into garments and clothing making it yet more convenient to wear these systems and therefore help in patient monitoring. For example, Figure 2.3 shows a comfortable body suit that incorporates a wide variety of sensors that can be worn under normal clothing.

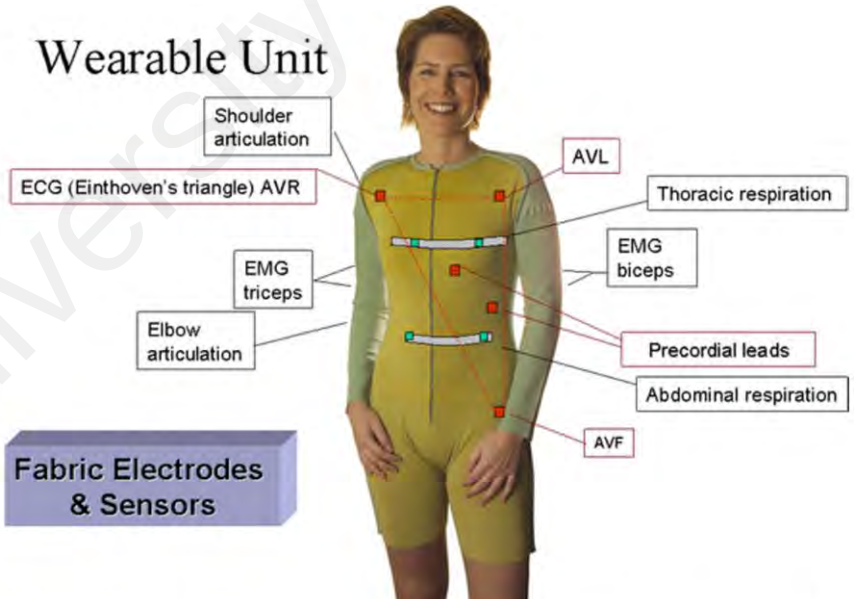


Figure 2.3: Sensors Incorporated in Clothing (Patel et al., 2012)

The most important parts of any wearable sensor systems are the sensors which can measure a variety of physiological data and the wireless transmission hardware. Advancement in MEMS technology has enabled the size of sensors to be reduced. MEMS

batch fabrication process has also reduced the cost of these sensors. With advancements in communication technologies such as Radio Frequency (RF), Bluetooth and Wi-Fi sensor systems no longer need to be connected via wires. The gathered data can be sent via a gateway such as a home computer or mobile phones through the internet to the patient's physician who can monitor and analyse the data to provide meaningful feedback. Mobile applications can also log data from sensor systems and have enough computing power to process and analyse according to programmed logic the said data and provide meaningful feedback.

In the past years sensor technology has come a long way. Sensors to measure heart rate and blood oxygen level have been developed. For example the heart rate and blood oxygen monitoring sensor developed by Asada et al. (2003) comes in the form of a ring that can be worn on a finger. The sensor is completely self-contained and transmits its data via an RF transmitter (Asada et al., 2003). Corbishley and Rodriguez-Villegas (2008) developed a system in which a microphone mounted on a person's neck can measure acoustic signals that are produced by breathing and use this data to calculate the respiratory rate with more than 90 percent accuracy (Corbishley & Rodriguez-Villegas, 2008). Patterson et al. (2009) used flexible circuits to develop an ear worn low power PPG sensor to monitor heart rates (Patterson et al., 2009).

Bio-chemical sensors have also been integrated into the field of wearable sensor systems. Dudde et al. (2006) developed a wearable minimal invasive sensor system that measures blood glucose level and administers the appropriate level of insulin. The system also incorporates Bluetooth so that data such as the amount of insulin administered and blood sugar levels can be sent to smart phone applications or computer for logging and analysing (Dudde et al., 2006).

Curone et al. (2010) developed a garment for fire fighters, fitted with various sensors that can measure heart and respiration rate, blood oxygen levels, body

temperature and movement, as well as external conditions such as carbon dioxide and monoxide levels, temperature, humidity, etc. All this data is then analysed by the system which then warns the firefighter of dangerous elements in the environment. Also the system sends data to a control center so that they can be informed about the wellbeing of the fire fighter (Curone et al., 2010).

In the field of rehabilitation wearable sensor systems, mostly include inertial sensors which can detect and track movements. Such sensors thanks to advancement in MEMS technology are cheap and use low power which makes them ideal for monitoring purposes. Home rehabilitation is an emerging field. In the previous years inertial sensors were the main sensors used in monitoring patient's rehabilitation and exercises at home. Recently virtual reality gaming is being used to make home rehabilitation easier and more motivating. For example the valedo system developed by hacoma AG is a system of devices that help patients who are recovering from back injuries train their back. The system incorporates a game which keeps the patient motivated via game achievements and verbal and visual encouragement. Efforts like the myHeart Initiative have also been made to incorporate sensors in garments to make it more comfortable to be used in a home environment (Patel et al., 2012).

For stroke rehabilitation at home, inertial sensors have been vastly used because of their compact size and cost effectiveness. In recent years with the widespread use of smart devices such as smartphone, smart wrist bands, etc, these devices are also a viable tool for long term monitoring. The reason behind this is that people use these devices every day and have them on their person during most times of the day. And these devices are equipped with inertial sensors such as accelerometers, gyroscopes and magnetometers. Therefore it is convenient to design monitoring applications around these devices (Šimaitytė et al., 2019).

2.2 Use of IMU in limb tracking

IMUs or inertial measurement units have seen much use for the purpose of orientation and position detection because of their compactness and because they are relatively inexpensive.

Atrsaei et al. (2018) utilized two MTx inertial sensors for the purpose of arm tracking. They mounted one of the sensors on the upper arm and one on the forearm as shown in Figure 2.4.

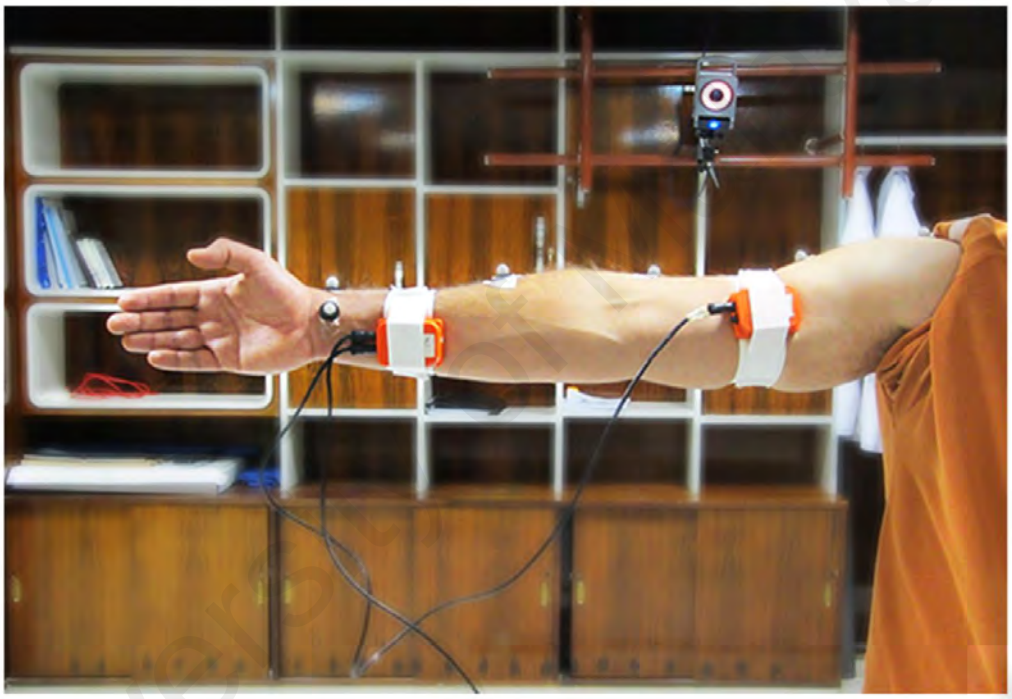


Figure 2.4: IMU Sensors mounted on the arm (Atrsaei et al. 2018)

To avoid singularities such as the gimbal lock, they used quaternions from the sensors to determine their orientation in space. They used the quaternion output with the accelerometer and magnetometer output to determine the linear velocities of each of the sensors. Using these velocities, the velocities for the elbow and the wrist joint are calculated with the assumption that the shoulder joint is rigid. Hence using the velocities of the elbow and wrist their relative positions are estimated. The results were compared with the results of a motion capture camera. The mean error between the position captured

by the camera and the position estimated by the IMU system was less than 6 cm (Atrsaei et al., 2018).

Upper limb tracking in recent years have gained popularity because of an increased research focus due to its many uses in fields such as sports, gaming, robotics and medicine and rehabilitation. Different technologies have been used for the purpose of tracking including optical, magnetic, mechanical, acoustic and inertial measurement tracking systems.

Mechanical tracking involves use of an exoskeleton mounted on the limb to be tracked and then using encoders measure the various joint angles. For example, Gu et al. (2016) developed a lightweight mechanical hand exoskeleton that can be attached to a human hand. A force feedback unit attached to each finger of the exoskeleton measures the movement of each finger of the human hand. (Gu et al., 2016)

Acoustic tracking uses ultrasonic sensors to measure time of flight and triangulation to estimate the position of the entity to track. Wang et al. (2003) developed a system in which multiple small wireless acoustic sensors were placed in the vicinity of the object to track. And using the data from these sensors when the target moved, the position of the target was triangulated. The data from the sensors was wirelessly transmitted to a computer using 'sink tree' routing algorithm.(Wang et al., 2003)

In magnetic tracking mini coils are mounted at various points on the limb to be tracked. A field transmitter is used to induce currents in these mini coils. The current intensity in these sensor coils is proportional to the distance of these sensor coils from the field transmitter. Therefore, the position in space of these sensors can then be calculated and hence the position of the limb.

Optical tracking uses multiple cameras to track marked position on a moving object and use 3D reconstruction algorithms to track objects. Multiple cameras are placed around the target to track. Markers are placed on various points on the target body. Each camera records the target from a different angle. When all the data is collected a computer algorithm 'stitches' all the different data into a single 3D point cloud. By identifying the markers, the algorithm can then track the movement of the different points on the target's body.

Inertial sensors make use of a gyroscope, accelerometer and a magnetometer mounted on a single chip to get orientation and acceleration data, in order to calculate the velocities and position of the object or entity to track.

Zhu & Liang (2016) from the Stanford research group used one IMU sensor and one flex sensor to estimate the joint angles. Using a complementary filter to fuse the accelerometer and gyroscope readings from the IMU, they got the orientation of the IMU mounted on the upper arm and therefore the joint angles of the shoulder. They got the rotation angle of the elbow using a flex sensor. They then used a human arm model in unity software to track the actual arm using the orientation data from the sensors. Their error depended on proper placement of the sensors on the arm, where wrong placement would result in an offset between the actual position and estimated position (Zhu, 2016).

Tian et al. (2015) aimed to eliminate the drift in the orientation and positional data given by the IMU sensors by fusing the IMU data with the sensor data from a Kinect device. An IMU was mounted on the subjects' forearm and upper arm. An unscented Kalman filter is used to fuse the data from the accelerometer, gyroscope and magnetometer to provide a relatively accurate orientation and acceleration data. Positional data is estimated by double integrating the acceleration data with respect to time. This process causes the positional data to accumulate error over time, which is known as drift error. This error can be very large as demonstrated in this study. To help

reduce this error, the IMU data was fused with some geometrical constraints. One of these constraints was that since the elbow has only two degrees of freedom the angle of elbow abduction and adduction can be assumed to be very small. This assumption improved the result, but the accuracy was still not good enough. Therefore, the IMU data is fused with Kinect data to improve accuracy and remove the drift error. At the start the Kinect coordinate frame and global reference coordinate frame are not aligned. Therefore, before fusion a rotation is applied to the Kinect sensor data to align the Kinect coordinate frame with the global reference frame.

Data flow model for the IMU-Kinect fusion system is shown in Figure 2.5. First, data are measured from both the Kinect and IMU and fused. Geometrical constraints are then applied to the resulting data. And using the constrained data positions of the upper and lower limb are estimated. Results for position estimation is obtained by doing the experiment once with only IMU and the second time with both IMU and Kinect. The results with Kinect were smoother and suffered less from drift errors (Tian et al., 2015).

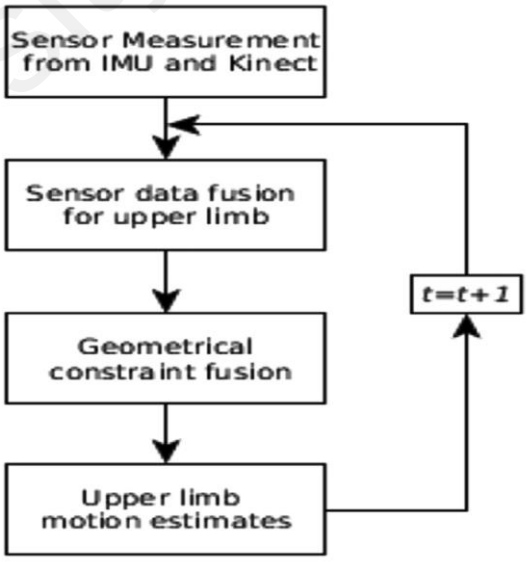


Figure 2.5: Data Flow Model for IMU-Kinect Fusion System (Tian et al 2015)

2.3 Forward Kinematics

Kinematic is a branch of mechanics that involves defining a motion of an object without considering the cause of motion that is force. It is the use of geometry of the points on the object to find the position of those points after some motion. It involves a series of equations that uses said geometry to output Euclidean position of said points. Forward kinematics is the use of those kinematic equations to determine the position and orientation of an end-effector in a robot with reference to its base frame from its joint parameters. The position of the end-effector with reference to its base frame is known as the Euclidean position and it is written as a vector such as Eq. (2.1) (Spong et al., 2006).

$$P = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \tag{2.1}$$

Usually robots consist of a series of links connected by joints which can either be prismatic or revolute. Revolute joints are joints that rotate about an axis. Prismatic joints are joints that move linearly along an axis. The base of the robot is usually rigid, and all the links are connected to the base. Therefore, if geometry is being used to determine the position of the end-effector, the position will be with reference to the base frame since the end-effector is connected to it through the links and joints. Since the links in the robots have one degree of movement, they can move relative to the axis of rotation of its joint in case of revolute joint or linear along the axis in case of a prismatic joint. Therefore, to define the position and orientation of the link a separate coordinate frame is attached to the joint to which the link is attached. Since a robot comprises a series of connected links, each of the coordinate frames can be mapped onto the coordinate frame of the previous joint via a transformation. Since all the joints in the arm model used in this project are revolute joints, further discussion would be with respect to revolute joints only. A transformation of coordinate frame B to coordinate frame A will also map all the points

in frame B to frame A. The transformation can be described as a series of rotations and translations (Spong et al., 2006).

A translation is the linear movement of a point in 3D space. A translation vector is of the same kind as in Eq. (2.1). To translate a point this translation vector is added to the point. A rotation involves multiplying a rotation matrix with a point to rotate that point about the origin of its coordinate frame. A rotation matrix is a combination of rotation about the three principal axes of the coordinate frame. To rotate a point B to a new point A the rotation matrix of the form in equation Eq. (2.2) is used.

$$ {}^{A}_{B}R = \left[{}^{A}_{B}\hat{X}\ {}^{A}_{B}\hat{Y}\ {}^{A}_{B}\hat{Z} \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.2} $$

Since a transformation is a combination of both rotations and translations, a transformation matrix is matrix combines these two operations in a single operation. A transformation matrix is usually in a format given in equation (3).

$$ {}^{A}_{B}T = \begin{bmatrix} & {}^{A}_{B}R & & {}^{A}_{B}P \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

Coming back to forward kinematics, finding the coordinates of the end-effector involves transforming the coordinate frame of the end-effector into the frame of the joint preceding it and repeat this process until the base frame is reached. This involves multiplying all the frame transformation matrices from the end-effector frame to the base frame. The combined transformation matrix will map the coordinates of the end-effector from its own coordinate frame to the base coordinate frame.

Calculating the transformation matrices for each link/joint can be a tedious and lengthy, especially if there are three axes. To make assigning coordinate frames and matrix calculations, easier we will make use of the Denavits-Hartenberg Convention. In 1955 Jacques Denavits and Richard Hartenberg came up with a standardized method of assigning coordinate frames to joints in a robotic manipulator (Lipkin, 2005). This

convention has a fixed set of rules to assign coordinate frames to the joints in a robotic manipulator. Once the coordinate frames are assigned to each joint, a set of four parameters can be determined for each frame. These parameters are known as DH parameters and they represent the rotations and translations needed to transform the frame into the coordinate frame of the previous frame. These parameters are namely 'θ', 'd', 'α' and 'a'. θ_i denotes the amount of rotation required about axis $Z_{n-1}$to align axes X_{n-1} to X_n with 'n' representing the frame number. The value d_i represents a translation along Z_{n-1} axis. α_i represents the rotation about axis X_n to align axes Z_{n-1} with Z_n. The value a_i represents a translation along the axis X_n. These parameters can be calculated as seen in Figure 2.6.

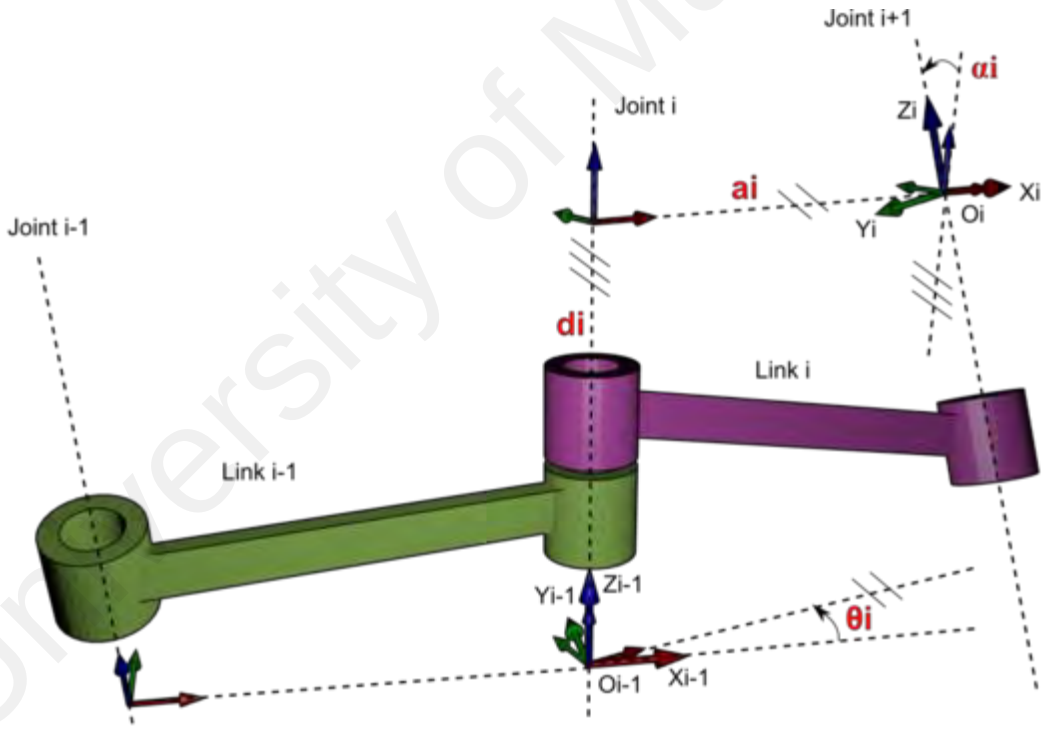


Figure 2.6: Determining DH Parameters

After determining these four parameters for each of the coordinate frames, DH convention provides a standard matrix format in which the DH parameters can be inserted to derive the transformation matrix which will transform one coordinate frame onto the previous frame. The standard matrix format is given by Eq. (2.4).

$$T_i = \begin{bmatrix} Cos\theta_i & -Sin\theta_i Cos\alpha_i & Sin\theta_i Sin\alpha_i & a_i Cos\theta_i \\ Sin\theta_i & Cos\theta_i Cos\alpha_i & -Cos\theta_i Sin\alpha_i & a_i Sin\theta_i \\ 0 & Sin\alpha_i & Cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

Using this format, transformation matrices can be derived and then multiplied together to get the overall transformation matrix which will map the end-effector in the reference axes of the base frame, as shown in Eq. (2.5).

$$T_{combined} = \prod_1^n T_n \quad (2.5)$$

$$Position_{reference} = T_{combined} \cdot Position_{local} \quad (2.6)$$

Once the overall transformation matrix is obtained. Eq. (2.6) is used to dot product the overall transformation matrix with the coordinates of the end effector in its local coordinate frame as shown in Figure 2.7 to get its coordinates in the reference coordinate frame.

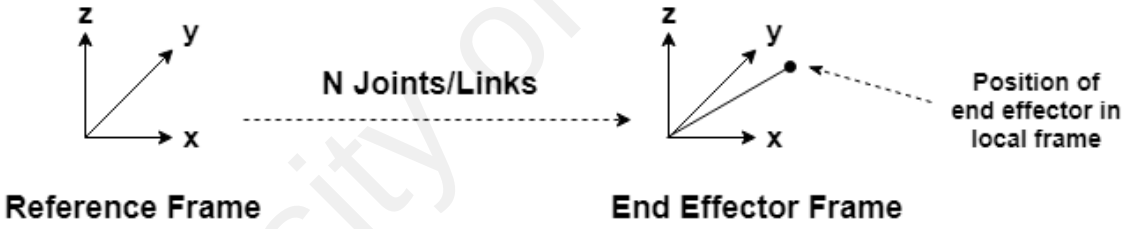


Figure 2.7: End effector relation to reference frame

Using the DH convention simplifies the process of calculating the transformation matrices for each linkage and joint. Without this convention, the transformation matrices would need to be calculated manually using trigonometry. This might be simple for systems with 2 or 3 degrees of freedom. But for systems with 6 to 7 degrees of freedom the manual process becomes extremely long and cumbersome.

Determining position and orientation using forward kinematics works when the reference frame is fixed. This is the case for this research in which the shoulder is the reference and assumed to not be moving. In such cases forward kinematics method of position and orientation determination of end effector or wrist joint is preferred over

traditional methods like getting position from acceleration. The reason being that forward kinematics is easier to program in computer and it is less computationally intensive as compared to the traditional methods.

Chapter 3: Methodology

One of our objectives in this project was to design a wearable IMU sensor module that can send its data wirelessly to either computer or smartphone. The methodology of developing the wireless IMU will be described next.

3.1 Inertial Measurement Unit (IMU)

IMU is a 3-in-1 sensor unit. An IMU contains an accelerometer, gyroscope and a magnetometer all on a single chip. The output data consists of quaternions, Euler angles, rotation vector, linear acceleration, gravity and heading. The IMU used in this project is the Adafruit BNO055 absolute orientation sensor as shown in Figure 3.1.

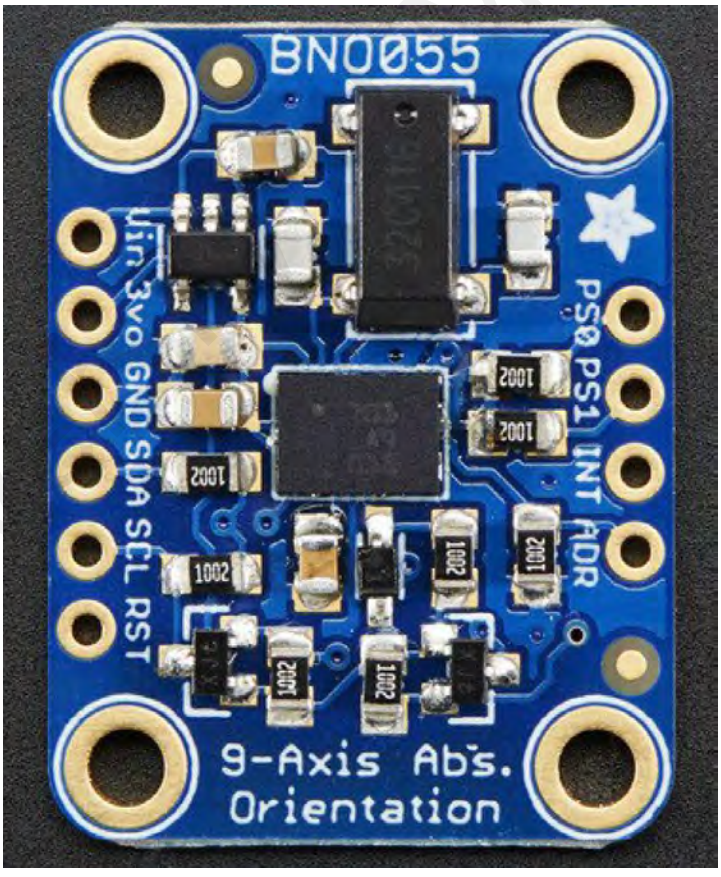


Figure 3.1: BNO055 IMU Sensor

It contains a 16 bit gyroscope, 14 bit accelerometer and a magnetometer. Along with these sensors this module also contains the complementary electronic components needed to make it an easily interfaceable module. This module also contains a high-speed ARM

Cortex M0 based processor which is tasked with taking the raw data from the accelerometer, gyroscope and magnetometer and fusing this data together into actual 3D space orientation. With other commercially available sensors only the raw data is available. This results in users spending weeks or months trying different filter algorithms to find one with required accuracy and speed. The BNO055 IMU module comes with a proprietary sensor fusion algorithm that can readily provide meaningful orientation data. Also, unlike other commercially available IMU sensors the BNO055 does its own calibration routine on start-up. This means that although a calibration run is recommended, it is not absolutely necessary and the sensor can be used as is. The BNO055 module is connected to a microcontroller by connecting its Vin pin to the 3.3V and the GND pin to the GND pin of the microcontroller. Then the SDA and SCL pins of the module are connected to the I2C pins of the microcontroller.

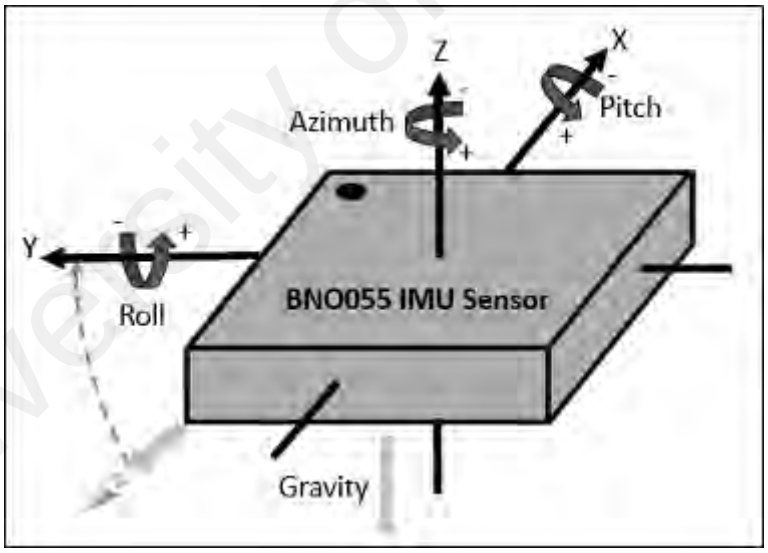


Figure 3.2: BNO055 Axes Arrangement

Figure 3.2 shows the axes of the BNO055 IMU sensor. The range for the X axis is from 0° to 360°. The range for the Z axis is from -180° to 180°. For the purpose of this project X and Z axis can be used as is since mathematically they are the same. However, the Y range is from 0° to 90° to 0° for 180° counterclock rotation, and 0° to -90° to 0° for 180° clockwise rotation. This is maybe due to the inherent function of BNO055 to prevent gimbal lock from happening. So this limits the rotation measurement to 90° in the Y axis.

Things become worse when the initial angle is not 0° which will further reduce the rotational measurement range. In this study, this issue can be tackled by carefully assigning this axis to suitable arm joint. This will be explained in the next section.

3.2 The Micro-controller

The micro-controller that was used in this project was the Node MCU as shown in Figure 3.3.

Figure 3.3: NodeMCU Micro-controller

The Node MCU is an open source Internet of things (IoT) platform which integrates a Tensilica Xtensa LX106 core with an Esp-12E wifi module. This makes the Node MCU a powerful standalone wifi solution for most IoT applications. The Node MCU is a 32 bit low-power micro-controller uni. It has a 10 bit analog to digital convertor, 2.4 Giga Hertz Wifi, an integrated TCP/IP protocol stack, an integrated power management unit that can incorporate a sleep mode and quick wakeup mode to save power and a lot of other useful features. It is a low power device, which can be powered via a small 9V battery which in addition to the compact size of the Node MCU makes it an ideal choice for portable wifi

applications. Another important aspect that made this micro-controller ideal for this project was its low cost.

3.3 Power source

For this project, a 9V rechargeable battery was used. This battery is easily rechargeable via a cellphone USB. To provide the Node MCU and IMU with a stable 5V, an LM2596 DC-DC converter was used, its circuit is shown in Figure 3.4.

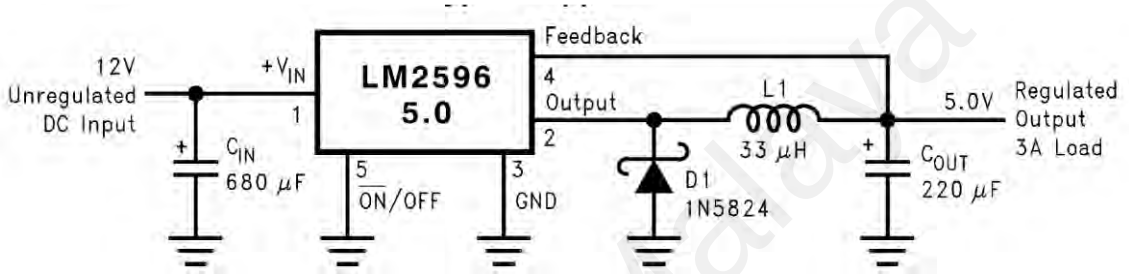


Figure 3.4: DC-DC Step-down Voltage Regulator Circuit

The LM2596 is a step-down voltage regulator which takes a higher unregulated voltage and steps it down to a regulated lower voltage. In our case a 9V DC input via the battery was stepped down to a stable 5V DC output which was then fed to the Node MCU and the IMU. This regulator can maintain this stable 5V DC output as long as the battery input stays above 6V considering the regulator's dropout voltage of 0.9V. Since micro-controller and the IMU both consumed about 80 mA of current during peak operation, the power draw from the battery is low and therefore the battery can last more than 8 hours of continuous use which is sufficient for the purpose of this project.

3.4 Programming languages

In this project, two main programming languages were used to design the software for the wireless IMU based arm position estimation system. C++ for programming the micro-controller and Python for the computer program.

The IMU sensors are interfaced with the Node MCU Wi-Fi enabled micro-controller. This micro-controller uses C++ as its main programming environment. C++ is often the

programming software for multi device application development (Evans, 2011). This is due to the portability of C++. It has an extensive selection of libraries that handle all the low-level mechanisms and provides the user with an easy to understand user interface. The biggest benefit of C++ is that since it is a middle level language and can deal with system's memory registers directly without needing an intermediary, it is faster than other programming languages. This makes it an ideal programming language for embedded applications.

For the computer-side application, there are a wide variety of programming languages to choose from like Java, python, C++, JavaScript, etc. After going through the pros and cons, python was chosen as the programming language for the computer-side application. There are many positives about python that made it stand out. Firstly it is an extremely easy to use language. Programming in python is like writing pseudo-code in English. Unlike C++ or Java variables do not have to be implicitly declared. If we write A equals 2 then the program knows that A is of the integer type. This makes it easy to program for beginner programmers. Python's syntax allows programmers to use fewer lines of code, as compared to Java or C++. Python also has extensive library support. There are a multitude of libraries available that can provide users with easy to use functions for the purpose of applications ranging from mathematics to signal processing to robotics. Hence, Python was chosen as the programming language for the computer-side application of this project.

For the purpose of presenting the result diagrams in a clear and well labelled manner MATLAB was used. Since the results from the python program was stored in a .csv format MATLAB was used to plot these results since MATLAB has good plotting and display options.

3.5 Design of the wearable sensor system

3.5.1 Prototype Version 1

The first prototype consisted of three IMUs connected to a single Node MCU micro-controller. The idea was to reduce the overall cost by using a single micro-controller. Since the micro-controller only had one I2C input and each of the IMUs required an I2C connection, we used an I2C expander as shown in Figure 3.5.

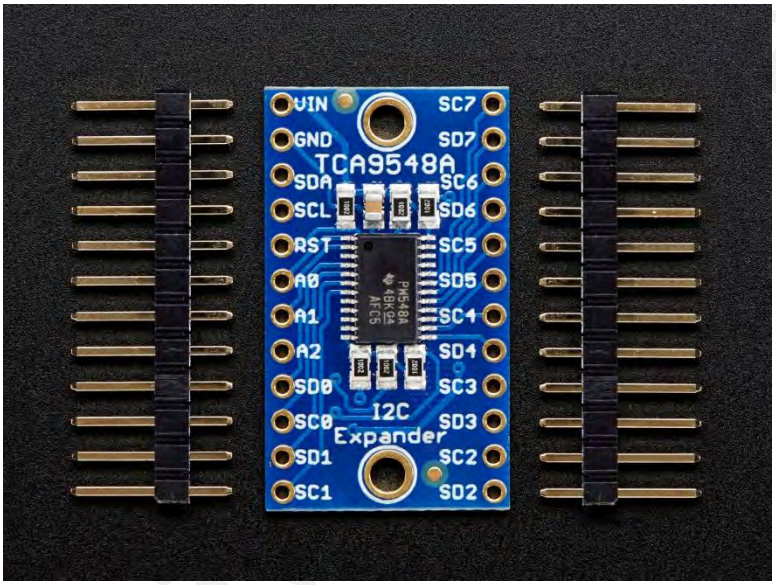


Figure 3.5: TCA9548A I2C Expander

This TCA9548A I2C expander allows a micro-controller with a single I2C port to communicate with up to 8 I2C devices such as an IMU. In our case the Node MCU micro-controller used this device to communicate with three BNO055 IMUs. This device works on the principles of multiplexing. The micro-controller first sends an address to the TCA9548A of the I2C device it wants to communicate with. That device can be connected to any of the eight ports ranging from SD0 SC0 to SD7 SC7. So for example the micro-controller wants to communicate with the IMU connected to port 1 then the micro-controller will first send the address of port 1 (SD1 SC1) to TCA9548A which will then redirect further communication from micro-controller (SDA SCA) to IMU on port 1 (SD1 SC1). This procedure is repeated for all three IMUs.

Initial prototype of the wearable sensor system had the micro-controller with the power supply strapped to the part of the forearm that was closest to the wrist. The micro-controller and the TCA9548A I2C expander were soldered together onto a double-sided PCB. This together with the power supply was attached to the arm using a small wrist band with buckles. The first IMU was attached to the upper arm close to the shoulder joint. The second IMU was attached to the upper forearm close to the elbow joint. And the third IMU was attached to the palm of the hand close to the wrist joint. The IMUs were attached to the arm using a mild adhesive double-sided tape for easy placement and removal. All of the IMUs were connected to the micro-controller via the TCA9548A chip using thin wires.

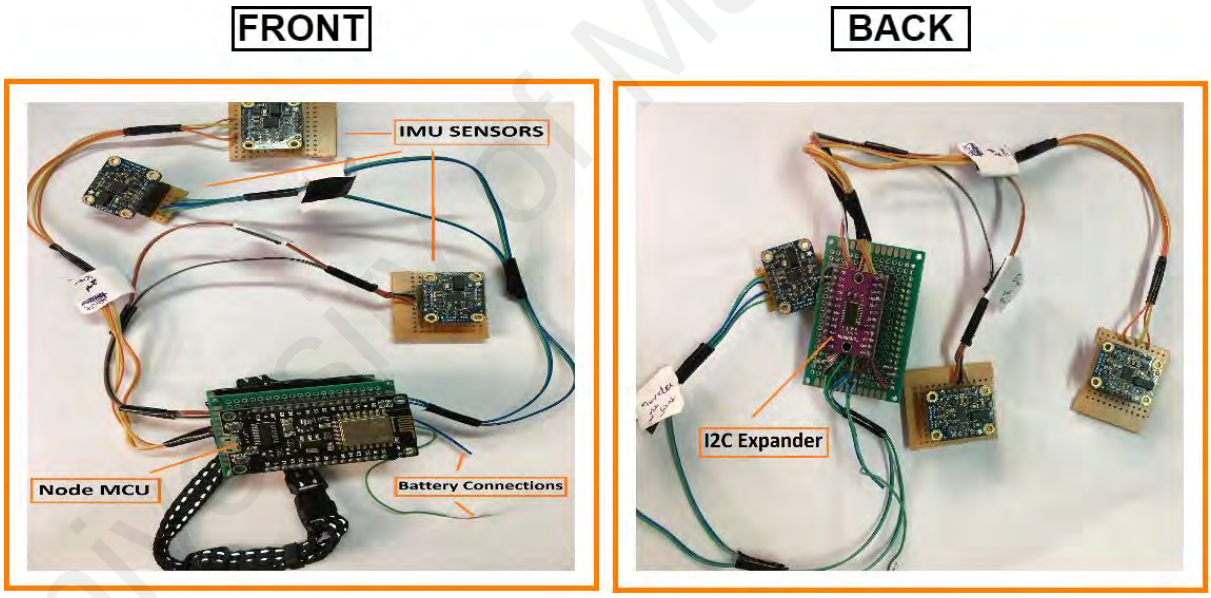


Figure 3.6: Wearable Sensor Prototype Version 1 (a) Front View (b) Back View

The idea behind this design was to limit the number of Wi-Fi micro-controllers to one in order to reduce cost as well as simplify the network protocol problem that will be discussed later. There were many issues with this design which later led to the development of another prototype with a different design. One of the issues was that there were too many parts in this wearable system that made wearing it time-consuming and difficult. Second was that there were too many wires going to and from the sensors. This

made debugging the circuit difficult in the case of electrical problems. The wires were also susceptible to breakage at the solder joints if the wires were moved around a lot. Also, due to the wires from the sensors spanning the length of the upper arm and forearm, the data coming from the sensors was susceptible to attenuation and picking up interference. Also, since the micro-controller had to multiplex between the three IMU sensors, this induced a delay between successive transmission which was around 0.01 seconds. This delay is regardless of the transmission between the system and a computer server. Due to these issues a new prototype was made with a more compact design.

3.5.2 Prototype Version 2

Keeping the issues of the previous design in mind, this design revolved around making each sensor a modular sensor node. Each sensor node would contain an IMU sensor, a Node MCU micro-controller and a power source. All of these components were soldered onto a double-sided PCB. The power source for each node consists of a 9V battery and a 5V regulator. The circuit is shown in Figure 3.7.

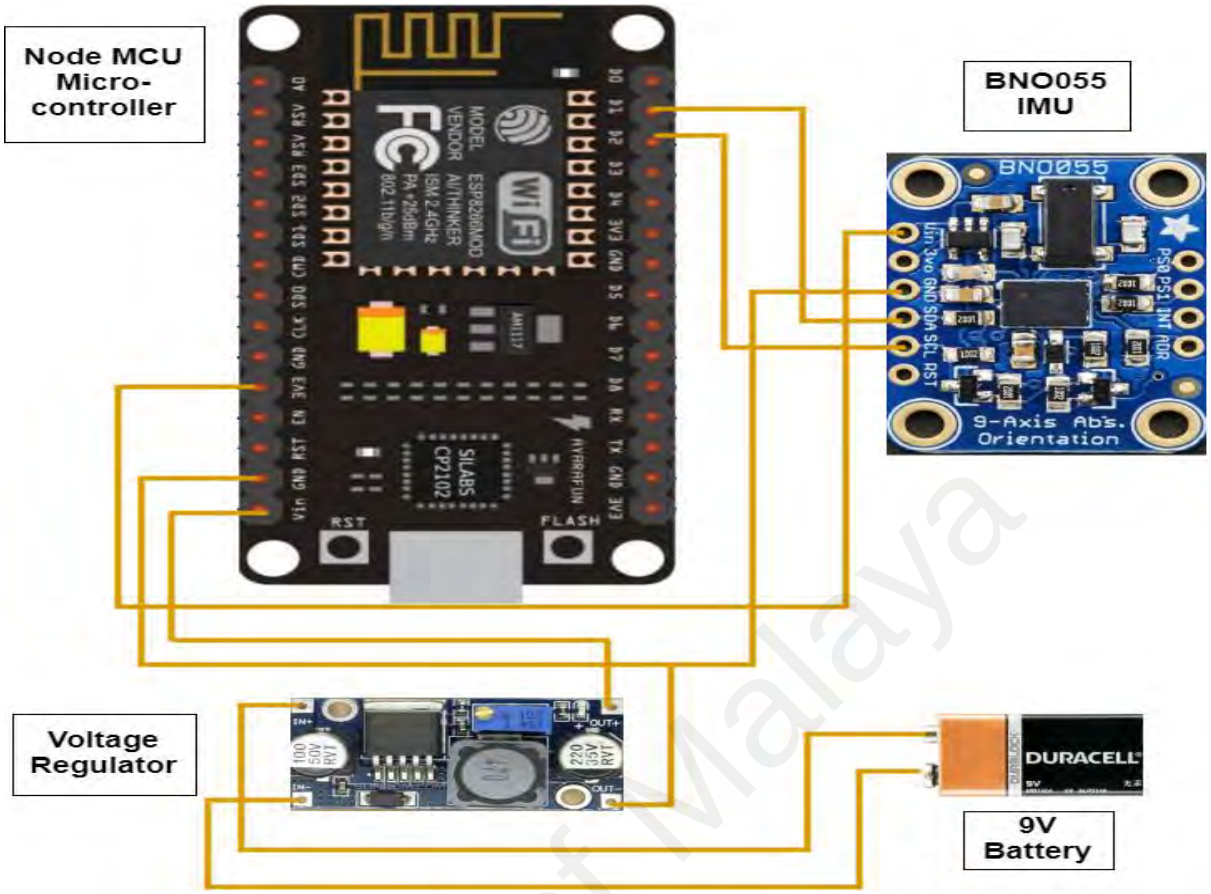


Figure 3.7: Electrical Connections for a IMU Sensor Node

Although three of these nodes were made, only two of them are needed for arm tracking for the purpose of monitoring stroke rehabilitation. This was based on the number of sensors used in various papers and articles in the literature review. The benefit of this modular system is that adding the third node into the system is not difficult. Like the first design the first sensor node will be attached to the upper arm and the second node will be attached to the forearm. The nodes are attached via Velcro straps. The complete sensor node as well as mounting procedure can be seen in Figure 3.8.

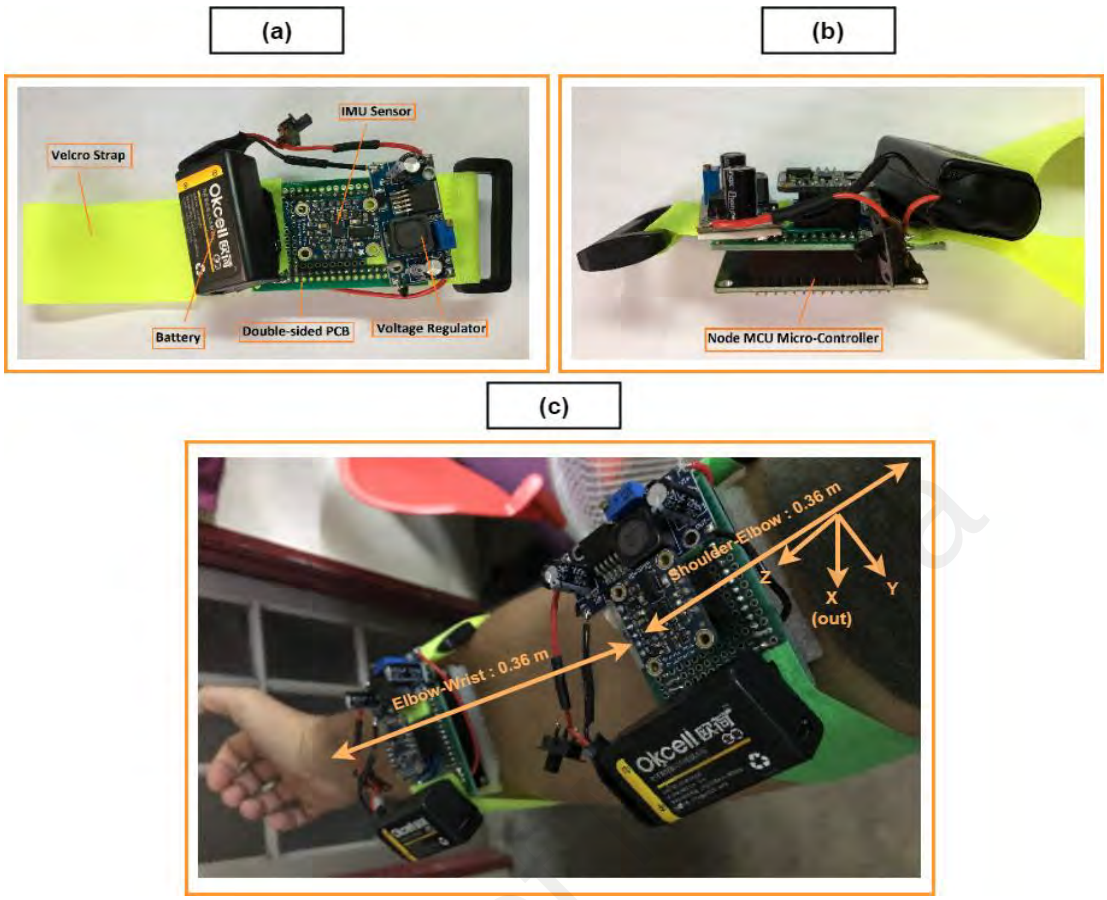


Figure 3.8: Wearable Sensor Node Version 2 (a) Front View (b) Back View (c) Sensor nodes mounted on the arm

This design is an improvement over the previous design because it has only two wearable parts and since everything was on a single PCB there were no long wires. This solved the issues that long wires caused in the previous design. The issues with this design was that there was a need for synchronisation between the nodes and the computer. If there was no synchronisation depending on which node was switched on first, there would be a delay between the positional data of the first node and the second node. This issue was handled in the software which would be discussed later. Also, the cost of this design is more than the previous one because we are using more micro-controllers and more power supplies. But this is offset by the ease of use and more compact design. Also, since the sensors are the most expensive parts of this project the additional cost of the micro-controller and power supply are not that big of an investment. The construction of each

node costed 40 RM while the sensor cost about 50 RM. This is a lot cheaper than for example the MTx inertial sensor which costs about $1000 each. This modular design also eliminated the use of the I2C expander since each IMU had its own micro-controller. This therefore removed the delay between successive data transmission that was induced by the multiplexing between the sensors in the previous design. Hence enabling the sensor nodes to send data at a higher frequency than the previous design.

3.6 Network Protocol

For this project, since data is being transmitted wirelessly it involves two devices to communicate with each other over Wi-Fi. In our case it involves communication between the computer and the IMU sensor nodes. So, we have a Wi-Fi router place in the vicinity of the sensor nodes and the computer. We first start the server program on the computer. The server listens for incoming connections. Then we switch on the nodes attached to the subject's arms. The nodes have the IP address of the computer already programmed into it as well as the username and password of the Wi-Fi router. When the sensor nodes are switched on they connect to the Wi-Fi and then connect to the Master Computer. This is illustrated in Figure 3.9. Till this point this method of network protocol was same for both versions of the prototypes. The good aspect of the previous design was that there was only one Wi-Fi micro-controller which meant that once the single micro-controller connected to the computer, it could start sending positional data from all the IMU sensors in single data messages. The issue with our current design of multiple Wi-Fi enabled IMU nodes is that of synchronisation. If like the previous design the nodes start data transmission as soon as they connect to the computer, the data from the node that was switched on last would lag the data from the node that switched on before it. So, to rectify this issue a synchronisation part was added to the network protocol. Once the sensor nodes are connected to the computer, they waits for a "START" signal from the computer after which the sensor nodes will start data transmission. On the computers end the software

will first wait for all the nodes in the system to connect and only then it will broadcast a "START" signal to all the nodes in the system (Rhodes & Goerzen, 2014). All the nodes will receive this signal at roughly the same time and will start their individual data transmission. Hence, since they all start data transmission at roughly the same time, they will all be synchronised. The codes for the network protocol will be discussed further. Program codes are attached in Appendix B and C.

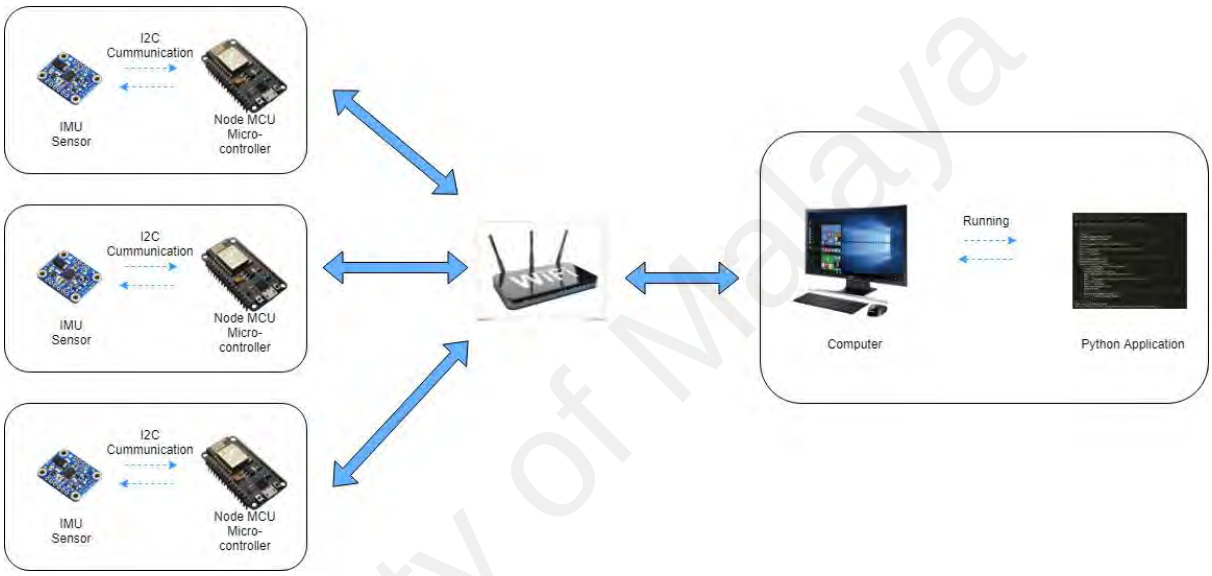


Figure 3.9: Communication Flow in the System

3.7 Forward Kinematics

Forward Kinematics is a method of translating joint angles into the final end-effector position for a robotic manipulator. As discussed in the literature review a popular method of performing forward kinematics is to use Denavit-Hartenberg Convention. We want to model a human arm using DH Convention. DH convention involves dividing the manipulator (human arm in our case) in a series of joints and links. Frames of references and axes are attached to each joint. Based on the frame assignments, four parameters known as DH parameters are calculated for each joint in the manipulator. Using these four parameters, transformation matrices for each link/joint are determined. Each transformation matrix transforms the axis of one frame with the link to the reference

frame of the previous link/joint. So, if we multiply all the transformation matrices of all the joints/links we can get a combined matrix that can give the position of the end-effector in the reference frame of the base joint.

Usually in a robotic manipulator, the base joint is fixed and assumed to not be moving. Since we are tracking arm movement with reference to the shoulder joint, we can assume that the shoulder is not moving linearly. Using this assumption, we can model the human arm as a robotic manipulator with the shoulder acting as the base.

The human arm has 7 degrees of freedom of movement. The shoulder joint is a ball and socket joint that can rotate about its three principal axes and therefore has three degrees of freedom. The elbow joint is a hinge joint which can only rotate about one axis and hence has one degree of freedom. The wrist joint is again a ball and socket joint that can rotate about its three principal axes and therefore has three degrees of freedom. This gives a human arm a total of 7 degrees of freedom.

In our model each of these degrees of freedom can be modelled as a revolute joint connected together with links. Figure 3.10 shows our human arm modelled as a series of kinematic joints and links. The Z axes represent the axes of rotation for each joint.

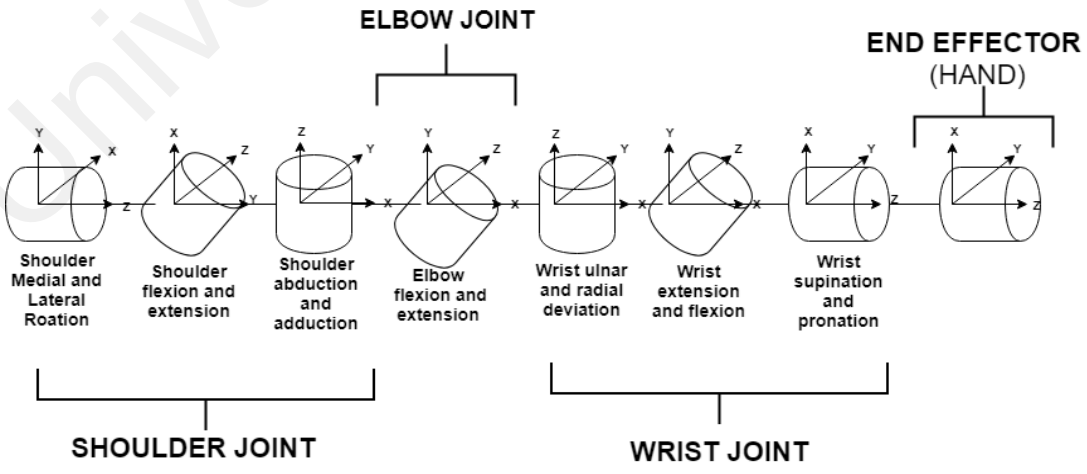


Figure 3.10: DH Model for the Human Arm

Each of these joints have been assigned a coordinate frame whose axes were determined using the rules of the DH convention. The link lengths connecting the three

shoulder joints are zero since physically the shoulder joint is one joint. Using DH convention shoulder and wrist joints can be modelled as three revolute joints lumped together with no link lengths in between. The links connecting the elbow joint with the shoulder and the wrist joints have lengths and they are the upper arm length and the forearm length respectively.

As discussed in the literature review using this model and following the Denavit-Hartenberg convention a set of four parameters known as DH parameters can be determined for the seven frames in the model. These parameters can be determined using a set of rules of the DH convention. Table 3.1 shows all the DH parameters for all the frames in our model.

Table 3.1: DH Parameters for the Arm model

Frame number, n	Joint Angle θ (°)	Frame alignment angle α (°)	Frame translation along X_n axis a (m)	Frame translation along Z_{n-1} d (m)
1	θ_1	90	0	0
2	θ_2	90	0	0
3	θ_3	-90	L_{se}	0
4	θ_4	90	L_{ew}	0
5	θ_5	-90	0	0
6	θ_6	-90	0	0
7	θ_7	0	0	0

θ_1 to θ_7 represent the joint variable which in our case are the joint angles or the angular position of the joints in their respective coordinate frames. L_{se} represents the length of the upper arm from the shoulder joint to the elbow joint and this value is in meters. L_{ew} represents the length of the forearm from the elbow joint to the wrist joint. For the sake

of uniformity, the upper arm and forearm lengths namely L_{se} and L_{ew} have been chosen to be 0.36 m in this project. This can be modified in the program code. The joint variables are obtained using the orientation of the sensor nodes mounted on the human arm. As mentioned before, each sensor node has three degrees of freedom in terms of orientation. So, for the shoulder joint which requires three joint variables we can use the three axes output from the sensor node as those joint variables. For the yaw and pitch we used the X and Z axes since they have full range. For the roll we used the Y axis of the sensor since it had limited range and for our purpose the roll is kept to the minimum. For the elbow joint variable which has only one degree of freedom we can use either X or Z axis of the IMU. For the wrist joint which is the same as a shoulder joint we use the same convention used for the shoulder joint. Using these parameters, a transformation matrix can be determined for each frame that will transform that frame into the coordinate system of the preceding frame. Matrices in Eq. (3.1) - (3.7) are the transformation matrices for the seven joints/links in our model. For example, T_1represents the transformation matrix that will map the coordinate frame of the first revolute joint in our model onto the coordinate frame of the shoulder base or reference coordinate frame.

$$T_1 = \begin{bmatrix} cos\theta_1 & -sin\theta_1 cos90^0 & sin\theta_1 sin90^0 & 0 * cos\theta_1 \\ sin\theta_1 & cos\theta_1 cos90^0 & -cos\theta_1 sin90^0 & 0 * sin\theta_1 \\ 0 & sin90^0 & cos90^0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$T_2 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 cos90^0 & sin\theta_2 sin90^0 & 0 * cos\theta_2 \\ sin\theta_2 & cos\theta_2 cos90^0 & -cos\theta_2 sin90^0 & 0 * sin\theta_2 \\ 0 & sin90^0 & cos90^0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_2 & 0 & sin\theta_2 & 0 \\ sin\theta_2 & 0 & -cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$T_3 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 cos(-90^0) & sin\theta_3 sin(-90^0) & L_{se} * cos\theta_3 \\ sin\theta_3 & cos\theta_3 cos(-90^0) & -cos\theta_3 sin(-90^0) & L_{se} * sin\theta_3 \\ 0 & sin(-90^0) & cos(-90^0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_3 & 0 & -sin\theta_3 & L_{se} * cos\theta_3 \\ sin\theta_3 & 0 & cos\theta_3 & L_{se} * sin\theta_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$T_4 = \begin{bmatrix} cos\theta_4 & -sin\theta_4 cos(90^0) & sin\theta_4 sin(90^0) & L_{ew} * cos\theta_4 \\ sin\theta_4 & cos\theta_4 cos(90^0) & -cos\theta_4 sin(90^0) & L_{ew} * sin\theta_4 \\ 0 & sin(90^0) & cos(90^0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_4 & 0 & sin\theta_4 & L_{ew} * cos\theta_4 \\ sin\theta_4 & 0 & -cos\theta_4 & L_{ew} * sin\theta_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$T_5 = \begin{bmatrix} cos\theta_5 & -sin\theta_5 cos(-90^0) & sin\theta_5 sin(-90^0) & 0 * cos\theta_5 \\ sin\theta_5 & cos\theta_5 cos(-90^0) & -cos\theta_5 sin(-90^0) & 0 * sin\theta_5 \\ 0 & sin(-90^0) & cos(-90^0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_5 & 0 & -sin\theta_5 & 0 \\ sin\theta_5 & 0 & cos\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$T_6 = \begin{bmatrix} cos\theta_6 & -sin\theta_6 cos(-90^0) & sin\theta_6 sin(-90^0) & 0 * cos\theta_6 \\ sin\theta_6 & cos\theta_6 cos(-90^0) & -cos\theta_6 sin(-90^0) & 0 * sin\theta_6 \\ 0 & sin(-90^0) & cos(-90^0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_6 & 0 & -sin\theta_6 & 0 \\ sin\theta_6 & 0 & cos\theta_6 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$T_7 = \begin{bmatrix} cos\theta_7 & -sin\theta_7 cos(0^0) & sin\theta_7 sin(0^0) & 0 * cos\theta_7 \\ sin\theta_7 & cos\theta_7 cos(0^0) & -cos\theta_7 sin(0^0) & 0 * sin\theta_7 \\ 0 & sin(0^0) & cos(0^0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_7 & -sin\theta_7 & 0 & 0 \\ sin\theta_7 & cos\theta_7 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

Since the position of the end-effector is in the frame of the last joint, in order to determine the position of the end-effector in the coordinate frame of the reference frame i.e. shoulder frame all the transformation matrices $T_1$to T_7 need to be multiplied together to give an overall transformation matrix as shown in Eq. (3.8). When a dot product is taken of this matrix with the coordinates of the end-effector in its frame, the result is the coordinates of the end-effector in the base or reference frame i.e. the shoulder frame. This is shown in Eq. (3.9).

$$T_{combined} = \prod_1^7 T_n = T_1 T_2 T_3 T_4 T_5 T_6 T_7 \quad (3.8)$$

$$Position_{reference} = T_{combined} \cdot Position_{local} \quad (3.9)$$

3.8 Sensor node application program

As mentioned earlier, a sensor node is an IMU interfaced with a Node MCU Wi-Fi enabled micro-controller. To get useful orientation data from this node to the computer, the micro-controller is programmed with an application written in C++ language (Evans, 2011). Figure 3.11 shows the pseudocode for the sensor node application program.

```
1. Begin
2. Set the unique ID
3. Initialise the IMU sensor
4. Connect to the router
5. Connect to the computer with application server running
6. Wait for the start signal from the computer
7. While True (forever)
8.     Get the orientation data from IMU
9.     Get the current time
10.    Form a message with the unique node ID, orientation data and current time stamp
11.    Transmit this message along with message length to the computer
12. End While
```

Figure 3.11: Pseudocode for Sensor Node

As can be seen in the pseudocode, each sensor node is assigned a unique ID. This ID is used by the program running on the computer to know which sensor node is sending its data and how to organise the data. On start-up the micro-controller initialises the inertial measurement unit (IMU) as well as connect to the Wi-Fi router. This is done by programming in the Wi-Fi username and password. The node then waits for the signal from the computer program to start transmission. It does this by entering a listen mode where it monitors all incoming messages for a specific message which in our case is "start". After the node has received this message it exits the listen mode and starts transmitting data continuously. The data consists of a message along with the message length. The message consists of the unique ID for sensor node, orientation data and the current time at which the message is sent. This timestamp is used by the computer program to determine the transmission delay and frequency.

3.9 Computer side application program

The computer side program is slightly more complicated than the sensor node program. Usually in programming the program or application executes the code sequentially starting from the top to the bottom as can be seen in the program for the sensor nodes. The issue with this sequential execution is that since out computer program

has a lot of ‘moving’ parts to it, a whole iteration of getting a sensor node message, deciphering it and performing the required analysis on it would take too long. This will lead to long delays before other messages go through the same process. Therefore, the computer side program will start to lag the sensor nodes a lot which is not preferred.

To rectify or at least improve the performance a lesson from large scale manufacturing can be implemented. In large scale factories to improve speed and efficiency large tasks are broken down into small tasks and there is a specialised operator for that task. This leads to a lot of small tasks being done simultaneously and then later combined to complete the original large task. In programming this parallel execution of tasks is known as multi-thread programming or multi-process programming. A thread can be thought of as that specialised worker in an assembly line who is given one aspect of a large product to work on. Threads can work on one aspect of a program in parallel with other threads to increase the speed and efficiency of the program (Zaccone, 2015). For this purpose, multi-thread programming was used for the computer side application program.

The main program of the computer side application can be broken down into four subsections. They are receiving messages from sensor nodes, deciphering those messages, performing the required analysis on data from the messages and handling any user input. To cater to each of these four subsections a program thread is assigned to each of them. These four threads will work in parallel and pass data to each other using a programming structure known as queues. These queues are used to store data. And like real life queues the first data to be stored is also the first data to be retrieved later. Figure 3.12 shows an overview of the computer side application program.

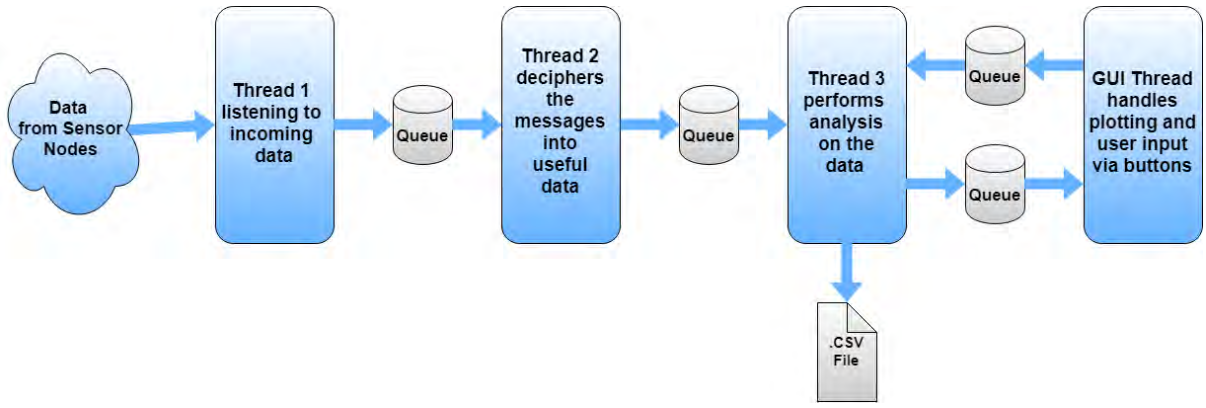


Figure 3.12: Computer Program Overview

The start-up pseudocode for the compute side program can be seen in Figure 3.13. The program on initialisation waits for the server to listen for incoming connections. The program waits for all the sensor nodes to first connect to the computer. As the sensor nodes connect the program saves their connection information in an array based on the sensor nodes unique ID. Once all the nodes have connected to the computer the computer, it will broadcast a "start" transmission signal to all the nodes that are connected via the connection information stored earlier. Then, the program starts all the threads handling the four main aspects of the computer side application program.

```
1.  Begin
2.  For (all sensor nodes in system)
3.      Wait for all the nodes to connect
4.      Save the connection information for each node based on unique ID
5.  End For
6.  After all nodes have connected
7.  For (all sensor nodes in system)
8.      Send "Start" transmission signal to all nodes connected
9.  End For
10. Begin the listener thread (Thread 1)
11. Begin the reader thread (Thread 2)
12. Begin the analysis thread (Thread 3)
13. Begin the GUI thread (Thread 4)
```

Figure 3.13: Pseudocode for Computer Program Startup

The pseudocode for the listener thread is given in Figure 3.14. This tread actively listens for all messages that are received in the network stream. It first looks for the message header preceding the message and it contains the length of the incoming message. The thread then reads that specified length of the message which is in form of bytes. The thread then decodes those bytes into a readable or parse able format. It then stores this message in a message queue for the reader thread to access.

```
1.  Begin
2.  While True (forever)
3.      For (all sensor nodes in system)
4.          Check if any messages have been received
5.          Read the first two bytes containing the length of the data message
6.          Read this specified length of incoming message and label it 'data'
7.          Decode this 'data' message from bytes into a String data type
8.          Put this decoded message at the end of a Queue
9.      End For
10. End While
```

Figure 3.14: Pseudocode for Listen Thread

The pseudocode for the reader thread is given in Figure 3.15. The reader thread first checks if there are any pending messages in the message queue. Since the listener thread has lesser operations than the reader thread it is slightly faster. Therefore, as long as the sensor nodes keep transmitting there will always be messages in the message queue. The reader thread retrieves the first message from the message queue and splits the message based on comma and '&' separation. The thread then reads the tidied-up message to get the unique sensor ID, sensor orientation data and time stamp. The reader then sorts and stores this data in the data queue according to the unique ID, so that the thread doing the analysis knows which sensor node sent which data.

```
1.  Begin
2.  While True (forever)
3.      For (all sensor nodes in system)
4.          Check if message Queue has messages in it
5.          Split the message string into smaller strings based on comma separation
6.          Extract useful data from the shortened message string
7.          Put the positional and time data from sensor nodes in a variable
8.          Put that variable at the end of a Queue
9.      End For
10. End While
```

Figure 3.15: Pseudocode for Reader Thread

The pseudocode for the analyser thread is given in Figure 3.16. This thread does all the analysis on the data from the sensor nodes. This thread is the most computation intensive thread of the four. This thread first checks if there is new data in the data queue. If there is this thread retrieves new data from the data queue. If the data is the first one, then it is stored as the reference orientation. Transformation matrices are then calculated using the orientation data from all the sensor nodes as follows Eqs. (3.1) - (3.7). The orientation of the sensor nodes is the absolute orientation of the node itself. For the forward kinematics the angles used in the determination of the transformation matrices of the joints are the angles from the principal axes of the coordinate frame of the joint itself. In Figure 3.17 for the shoulder joint angle 'A' is used as the joint parameter in determining one of the transformation matrices. This is obtained directly from the sensor node mounted on the upper arm. The issue comes with the sensor node mounted on the forearm. The sensor node will give the angle 'B' while to calculate the matrices for the wrist position angle 'C' is required. Angle 'C' can be calculated using equation (3.10).

$$\angle C = \angle B - \angle A \qquad (3.10)$$

Firstly, the transformation matrix is determined that will be used to transform the elbow joint position from its local frame to the shoulder base frame. This is shown in Eq. (3.11).

$$^{shoulder}_{elbow}T = \prod_1^3 T_n = T_1 T_2 T_3 \qquad (3.11)$$

Taking a dot product of this matrix with the elbow joint position in its own or original frame will give the position of the elbow joint with reference to the shoulder joint. Another transformation matrix is determined that will transform the wrist joint position from its own frame into the shoulder base frame. This is shown in Eq. (3.12).

$$^{shoulder}_{wrist}T = \prod_1^7 T_n = T_1 T_2 T_3 T_4 T_5 T_6 T_7 \qquad (3.12)$$

If the user wants to save this data, then this thread saves the calculated elbow position, wrist position and current time to a '.csv' file. This thread also calculates the Euclidean distance travelled by the end effector. The thread does this by calculating the magnitudes of the vectors connecting the points in the path the end-effector travelled on. And then the total distance is calculated by combining all these small steps. This thread also calculates the time taken in each movement by making use of the time stamps. This data is also added to the plotter queue for the GUI thread.

```
1.  Begin
2.  While True (forever)
3.      For (all sensor nodes in system)
4.          Check if sensor data Queue has data in it
5.          Save the first data as reference position of our virtual arm model
6.          Retrieve sensor nodes data from queue
7.          Use that data to determine the transformation matrices for the 3 shoulder joints
8.          Multiply all these matrices to get overall transformation matrix
9.          Calculate dot product of this matrix with initial position of elbow link
10.         This gives elbow position with respect to shoulder joint
11.         Determine the transformation matrices for all the 7 joints
12.         Multiply all these matrices to get overall transformation matrix
13.         Calculate dot product of this matrix with initial position of wrist link
14.         This gives wrist position with respect to shoulder joint
15.         Calculate overall distance travelled using these coordinates
16.         Save these coordinates and distance travelled along with time in a .csv file
17.         Also put these coordinates in the plotter Queue
18.     End For
19. End While
```

Figure 3.16: Pseudocode for Analyser Thread

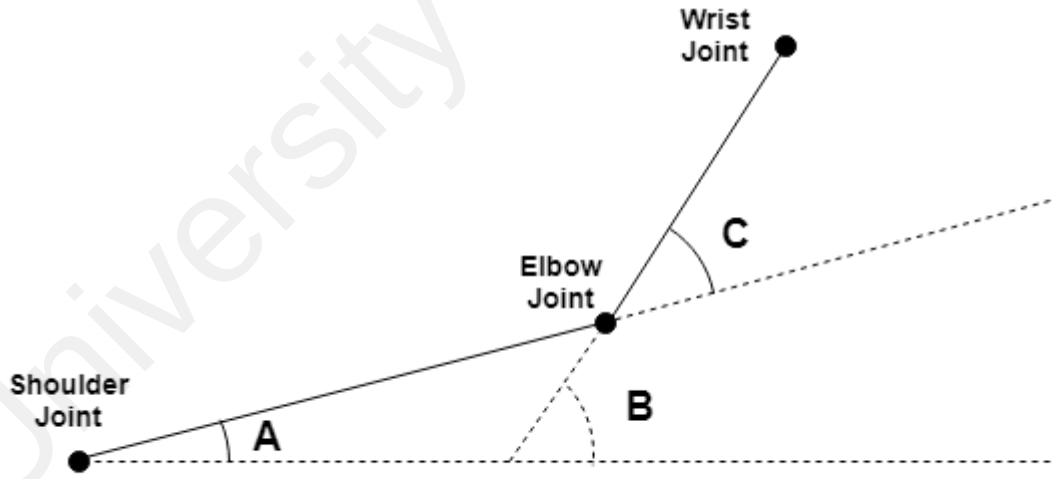


Figure 3.17: Geometry of Arm Extension

The GUI thread handles inputs from user as well as plot the data in form of a 3D graph. The GUI thread hosts a graphical user interface that can be seen in Figure 3.18.

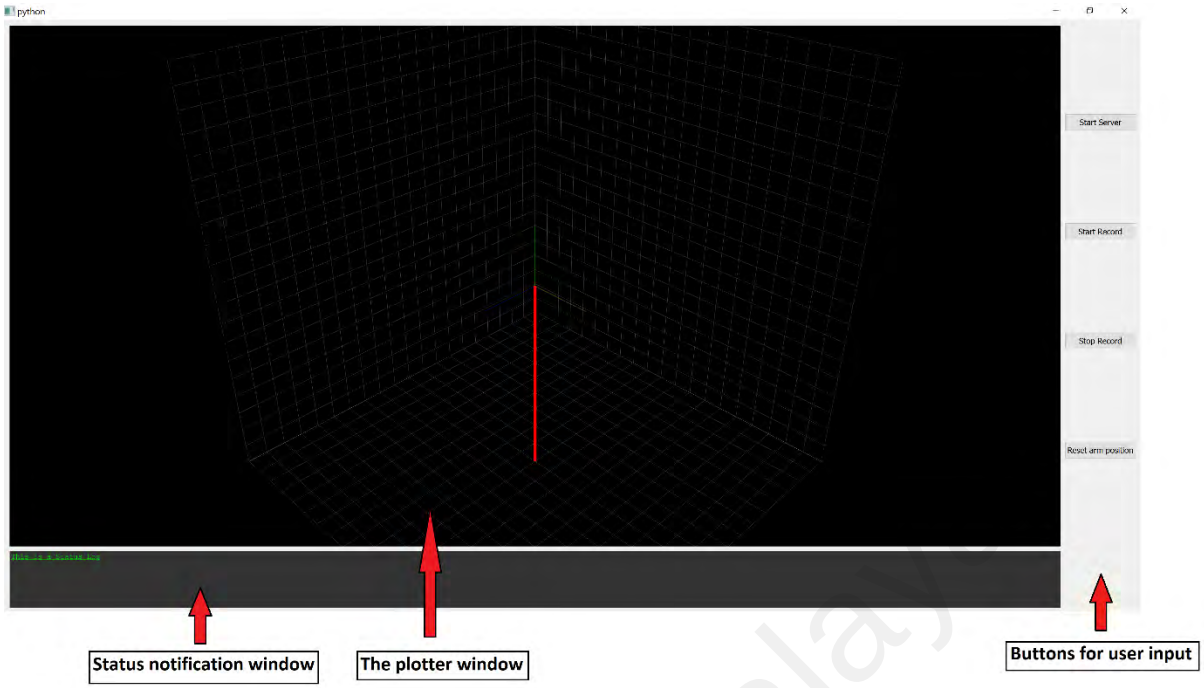


Figure 3.18: The Graphical User Interface (GUI)

The buttons allow user to start the program by the "start server button". The user can reset the arm starting reference position by using the "reset arm position button". The user can also save the wrist positions, elbow positions and time stamps by pressing the "Start record" and "Stop record" buttons. The status window shows useful information like which sensor nodes have connected and which buttons have been pressed. The GUI thread gets the data to plot from the Queue to which the analyser thread saved the arm positions.

For the purpose of plotting the results, Matlab is used. The data stored in the '.csv' files is used to plot the output of the tracking algorithm along with calculating the distance and the amount of time taken. The Matlab plotting code is attached as Appendix D.

Chapter 4: Results and discussion

Our purpose for this project is to have the wearable sensor nodes track the arm movement of the wearer with decent accuracy. The expected accuracy is not too high since the output is not used for any position and orientation critical tasks. The function is to check the effectiveness of rehabilitation exercises done in a home setting. So, after every few days or weeks as per physician's instructions, the patient will first wear the setup on their good arm and perform specific movements and then repeat the process on the stroke arm. The results can be used by physician to determine if any improvement has been made. This system can give useful data such as the total distance travelled during the movement, time taken and the angular speed.

4.1 Transmission delay

Figure 4.1 shows the transmission delay as the time taken for the sensor orientation data to be acquired from the IMU, transmitted over Wi-Fi and be available for use in analysis. As mentioned before this time calculation is possible because in every message of a sensor node we embedded a time stamp. This timestamp together with the receive time stamp can be used to calculate the transmission time or transmission delay in between consecutive messages received. Figure 4.1 shows a graph of transmission delays for several messages from the sensor nodes. It can be seen from the graph that the maximum delay is 0.2397 seconds, minimum delay is 0.0212 seconds and average delay is 0.0894 seconds. In the first prototype the I2C expander took 0.01 seconds to multiplex between each sensor. Therefore, a further delay of 0.02 seconds was being added which was increasing the average delay. Therefore, the modular sensor nodes are faster since there is no need for multiplexing.

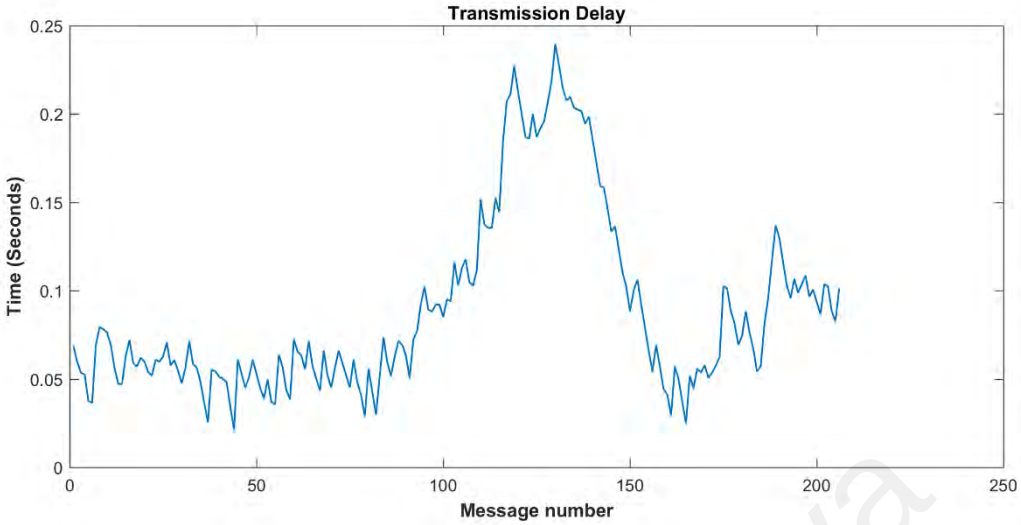


Figure 4.1: Transmission Delay in Communication

In the worst-case scenario, we will see the result of the arm movement about 0.2 seconds after the movement has taken place. And on average the delay less than 0.1 seconds which is almost negligible and acceptable for our purpose.

4.2 Transmission Frequency

Transmission frequency was calculated by taking the time taken to transmit ten messages from the sensor node to the computer. This time was then divided by ten to get the transmission time for one message and then the frequency is $\frac{1}{Transmission_{time}}$. Figure 4.2 is the transmission frequency graph.

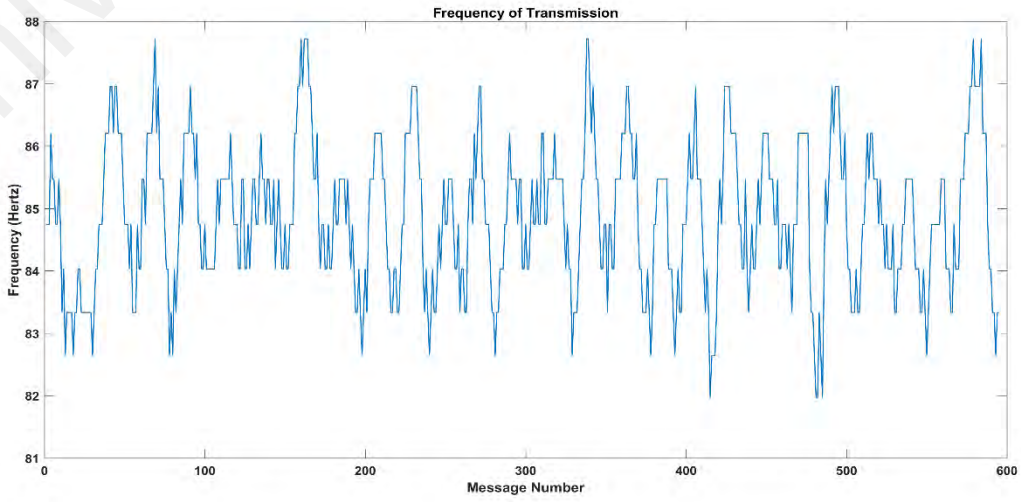


Figure 4.2: Frequency of Transmission from Sensor Node

Maximum frequency achieved is 87.72 Hz, minimum frequency noted is 81.97 Hz and the average transmission frequency is 84.86 Hz. This means that on average the arm movement is sampled 84 times in one second. Since our purpose is to track the arm movements of a stroke patient this frequency is more than enough. Since, normal arm movement speed of a healthy person is not fast enough to lead to a loss of data at this sampling frequency. Therefore, if the frequency of transmission is enough for the normal arm it is more than enough for a stroke affected arm.

4.3 Tracking ability

4.3.1 Arm raised from side to front

In this experiment the starting position of the human arm with the sensor nodes attached was vertical and by the side. The started position of the end effector or wrist joint is coordinates (0,0,-0.72) and this is because the length of the arm from shoulder to wrist is 0.72 m and the Z axis of our system is perpendicular to the ground. Then the arm was raised steadily to the front through a shoulder rotation of approximately 90 degrees. The movement to perform is the shoulder extension (figure 4.3) of close to 90 degrees. Figure 4.4 shows the results of the arm tracking as viewed from the X axis or the side. The red lines show the arm positions at intervals of time. The red lines connect the shoulder joint with the elbow position and the wrist position. The origin denotes the shoulder joint which is the reference point for the elbow and wrist positions. The blue line shows the path taken by the end-effector or the wrist joint. As can be seen the system can track linear movements like the arm raise fairly accurately.

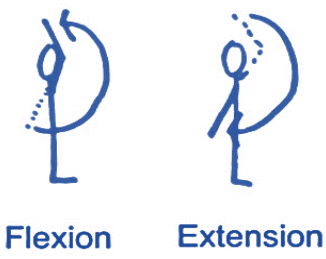


Figure 4.3: Shoulder Flexion and Extension

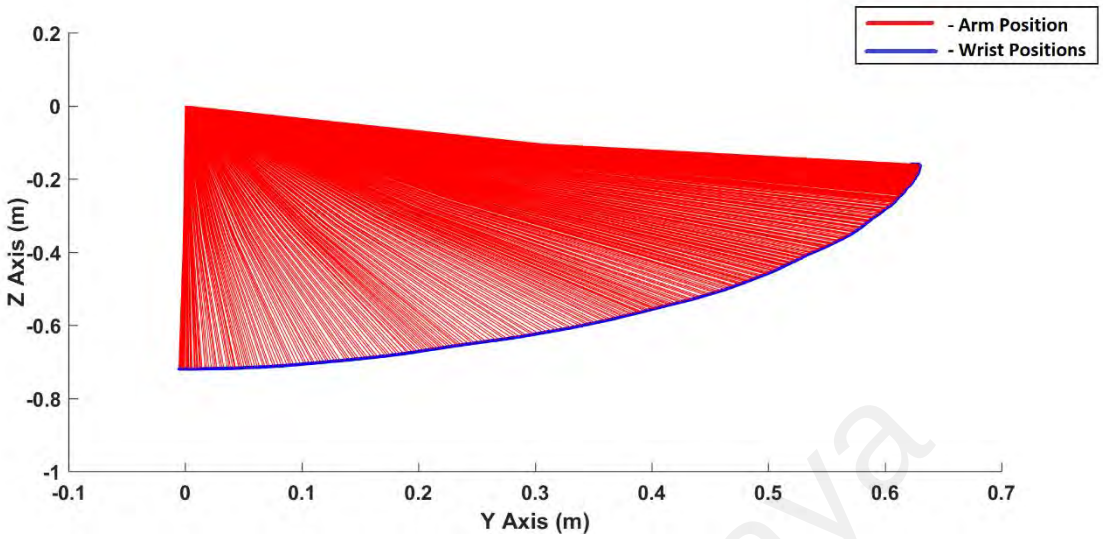


Figure 4.4: Tracking Arm Raised (Side View)

4.3.2 Arm raised to overhead

The previous experiment was repeated with the arm being raised almost two thirds of the way overhead. Figure 4.5 shows the results of the arm tracking. The results can be seen tracing an arc as the human arm moved. This is as expected. As the arm moved past 90 degrees the elbow joint started to flex which resulted in the wrist position and elbow position to no longer be parallel with reference to the shoulder joint. This can also be seen in figure 4.5.

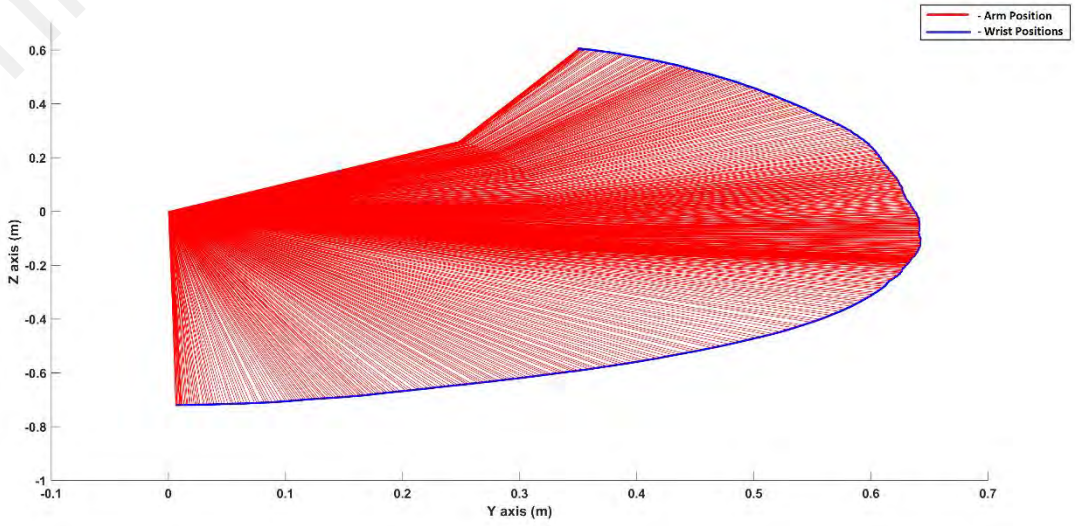


Figure 4.5: Tracking Arm Raised more than half way (Side View)

4.3.3 Elbow flexion

The figure 4.6 shows the result of the tracking algorithm as it tracks the elbow flexing through almost 90 degrees from the side (X axis). The arm started with the arm vertical by the side. Then as the elbow was flexed the elbow joint slowly moved back a small amount and a bit outwards laterally similar to a bicep curl. This can be seen in the figure as the elbow position in the middle of the figure as moving back while the wrist position is shown to be tracing an arc. Figure 4.7 shows this movement more clearly.

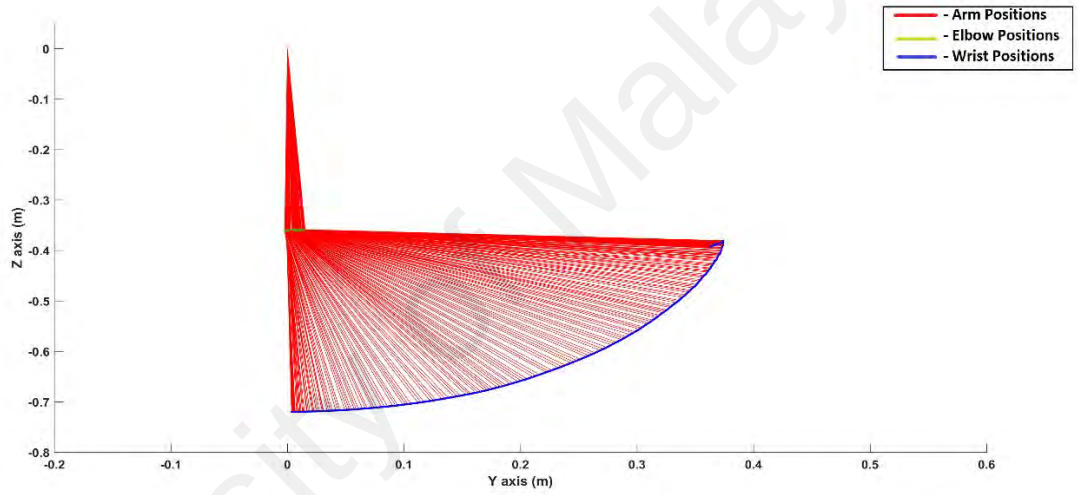


Figure 4.6: Tracking Elbow Flexion (Side View)

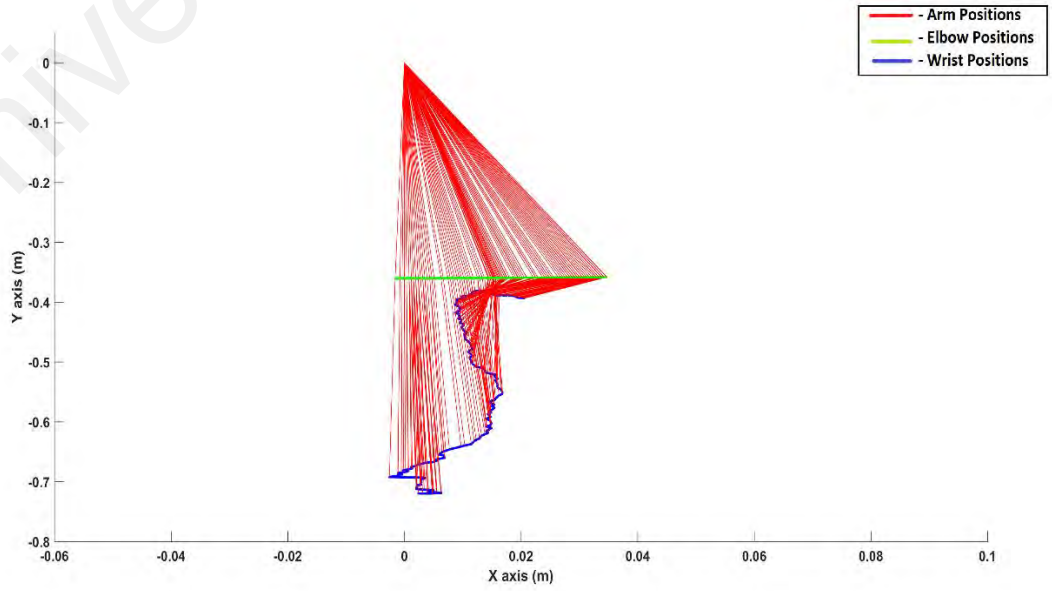


Figure 4.7: Elbow Flexion (Back View)

The view point in figure 4.7 is as seen from the Y axis or from behind the elbow. The blue line again denotes the wrist positions. The green line denotes the elbow positions. As can be seen the elbow joint and wrist joints move to the side slightly as the elbow joint is flexed and wrist joint moves forward as well.

4.3.4 Elbow flexion and shoulder lateral and medial rotations

In this experiment the elbow was again flexed about almost 90 degrees and then the shoulder joint was rotated about the Y axis which is also called lateral or medial rotation as shown in figure 4.8.

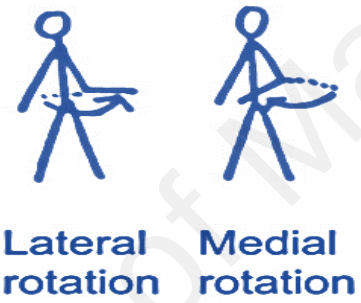


Figure 4.8: Shoulder Medial and Lateral Rotations

The results of the arm tracking algorithm can be seen in figure 4.9. This view is from the side or X axis. As can be seen from the figure when the elbow is flexed the wrist position follows an arc as shown by the blue line in the figure. Once the flexion is complete the shoulder joint goes through a medial rotation which moves the elbow and wrist joint side to side. This can be seen in figure 4.10 which shows the movement from the Y axis or from behind the elbow. The elbow joint positions and wrist positions can be seen in the green and blue lines respectively. Once the medial rotation is done the arm is brought back down to starting position.

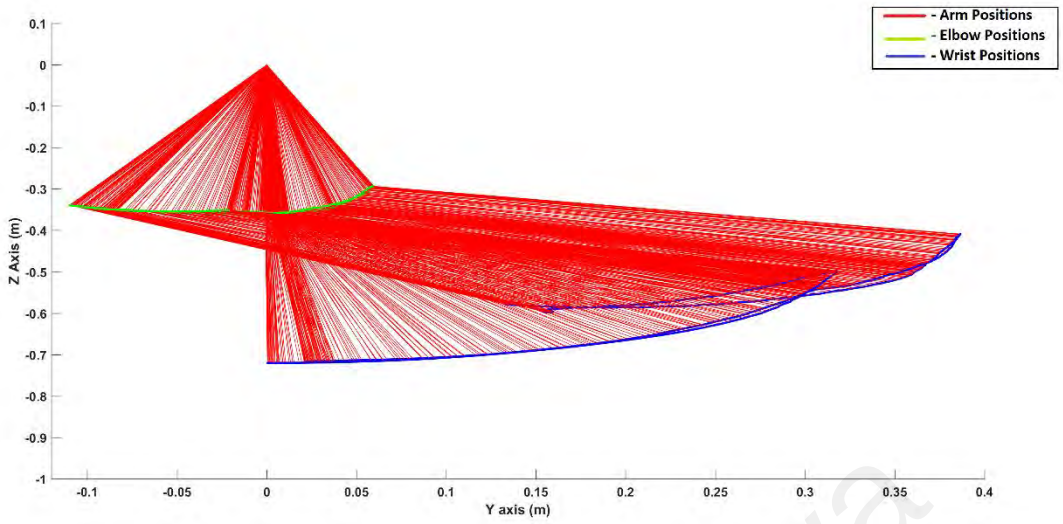


Figure 4.9: Elbow flexion and Shoulder Medial and Lateral Rotations (Side View)

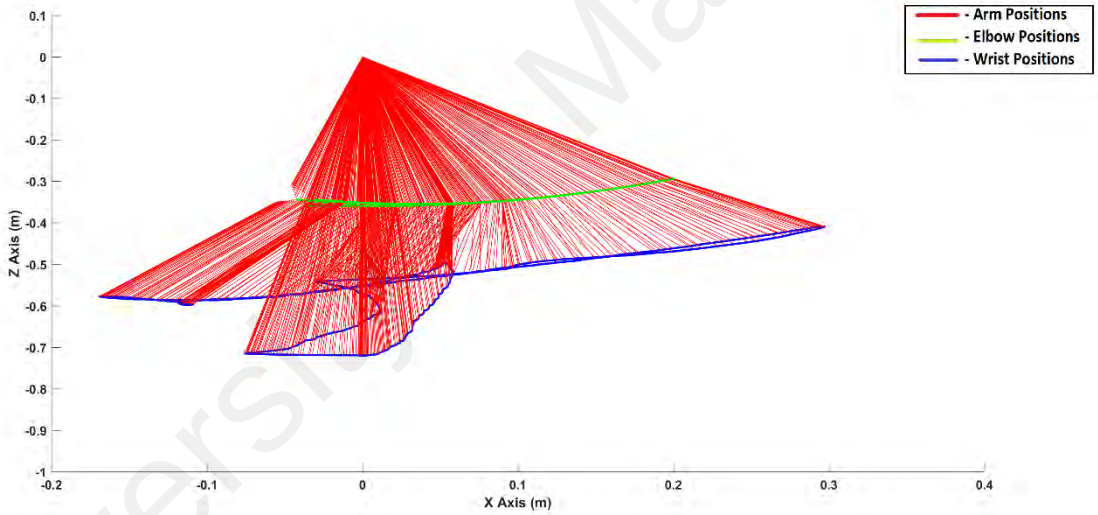


Figure 4.10: Elbow Flexion and Shoulder Rotations (Back View)

4.3.5 Tracing a triangle in air

This experiment although not necessary for the scope of our purpose, it was done to demonstrate a general tracking ability of this system. In this experiment the arm was raised from start position which is vertically by side of the body though about 90 degrees and then a rough triangle path was followed after which the arm was lowered. Figure 4.11 shows the arm positions tracked by the wearable sensor nodes. The view is from the front which is the Y axis.

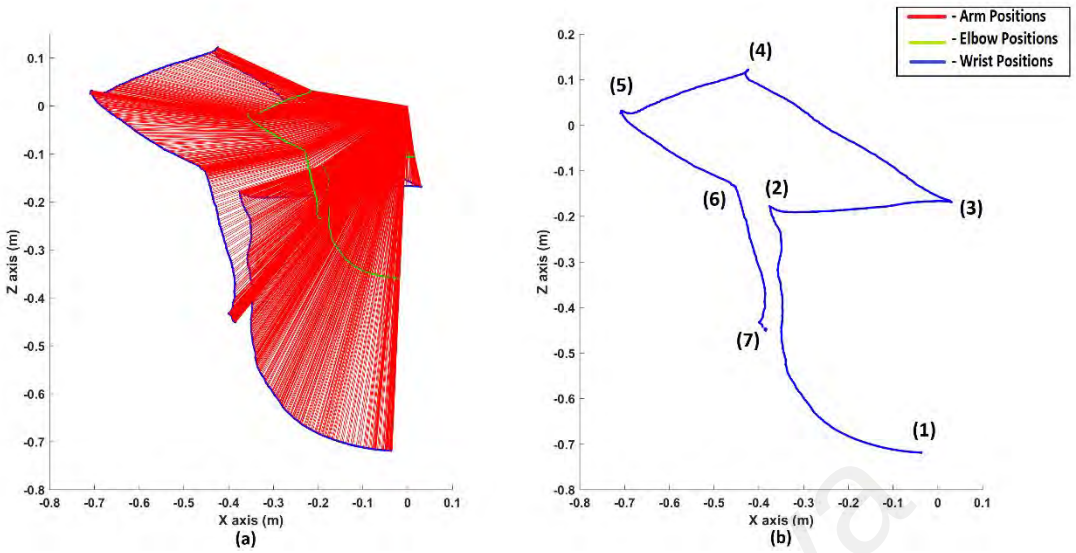


Figure 4.11: Making a Triangle in air (a) With arm positions shown (b) With end effector positions shown only

As can be seen in the figure the virtual arm started from coordinates (0,0,0.72) or point (1) marked in the Figure 4.11 (b) and was raised via the shoulder joint to the front which is point (2). Then a rough triangular shape was traced in air which is from point (2) to (6), after which the virtual arm is lowered. This demonstrates basic tracking abilities of the system in which the vertical and horizontal extension and flexion of the shoulder joint is captured with acceptable accuracy.

4.4 Accuracy of Tracking Distance

In this experiment the distance measuring capability of the system was tested. The system can track distance travelled as well as the time taken by the exercise. Three tests were conducted.

In the first test the human arm was extended from side to front. Figure 4.12 (a) shows the path followed by the arm on paper. This was achieved by standing next to the paper and moving the arm while holding a marker. The arc length or the actual distance traced by the wrist joint was 1.19 m.

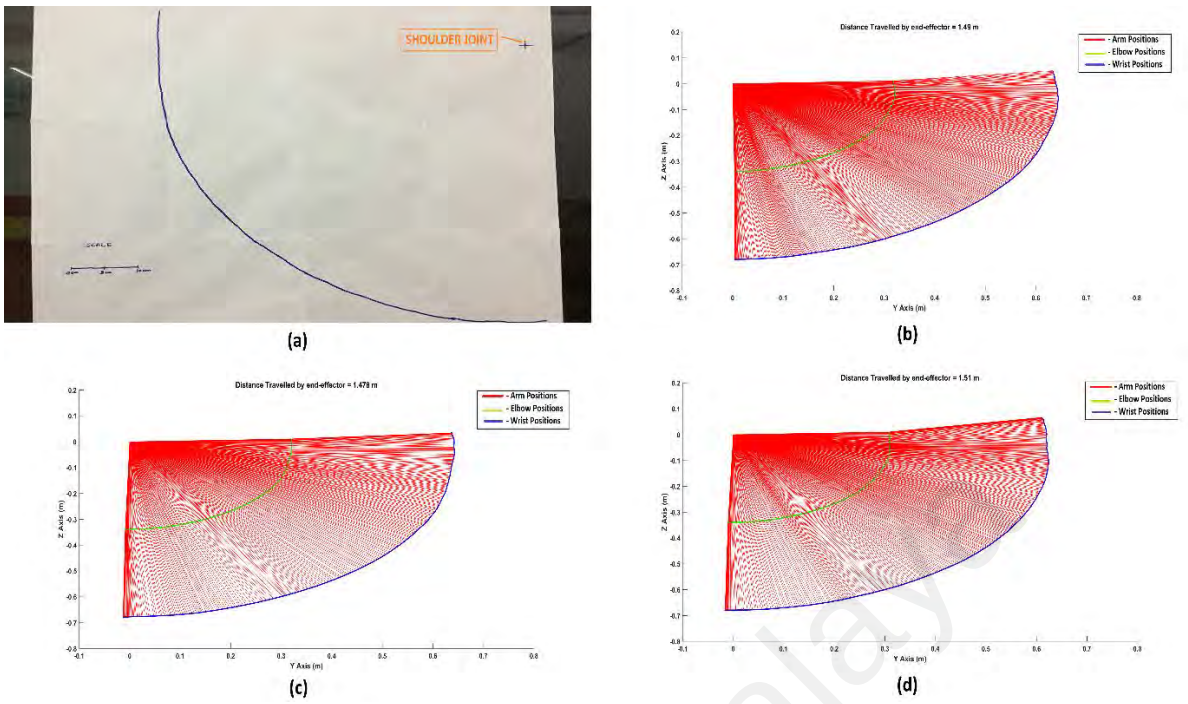


Figure 4.12: Tracking Shoulder Flexion (a) Reference Path to Follow (b - c) Tracking Algorithm Output for Three Iterations

This experiment was repeated three times and Figures 4.12 (b) to (d) show the results of the arm tracking algorithm. The average distance moved by the end-effector across the three experiments as calculated by the algorithm is 1.49 m. This gives an accuracy of 80 percent.

In the second test a parallelogram of perimeter 0.9 meters was traced by the human arm wearing the sensor nodes. Figure 4.13 shows the results of the tracking algorithm.

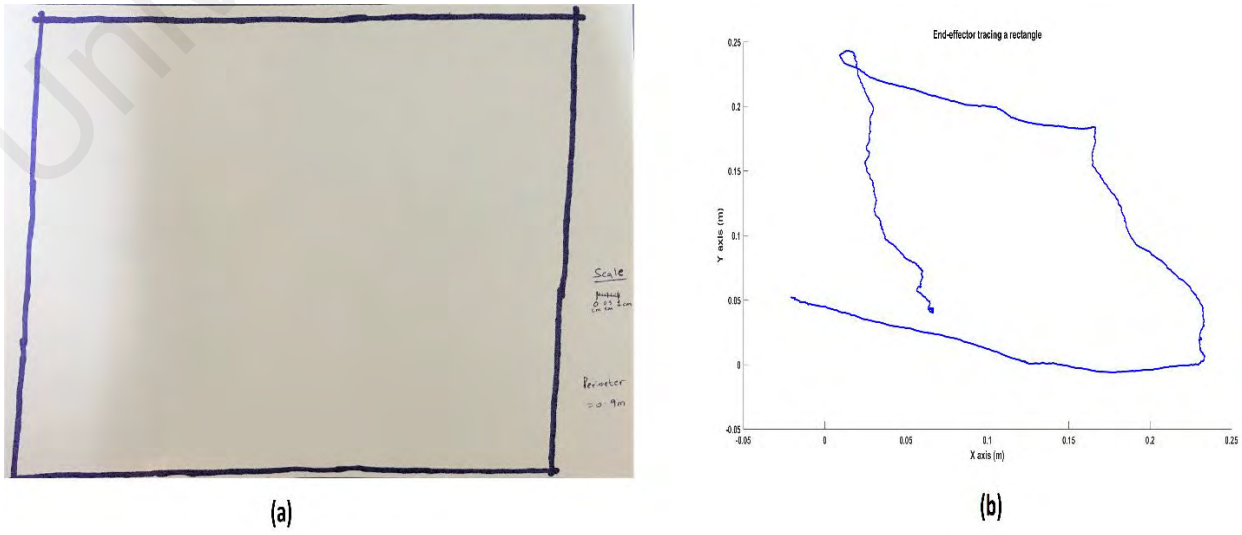
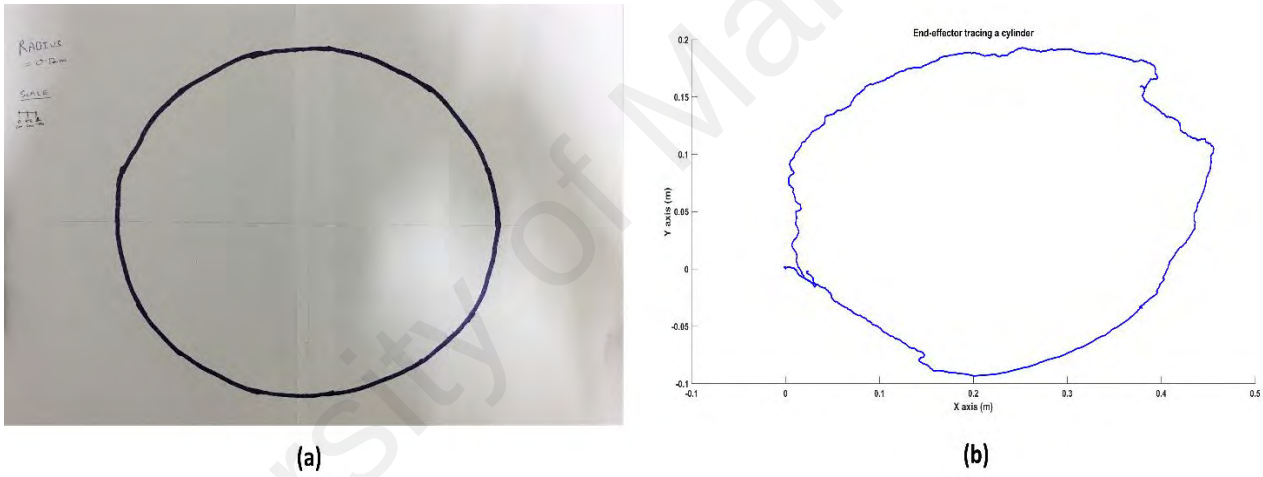


Figure 4.13: Tracing a Parallelogram (a) Reference Parallelogram (b) Tracing Algorithm Output

The perimeter of the rectangle measured by the algorithm was 1.24 meters and the time taken to do this was 11.7 seconds. This time is calculated using the receive times of the messages from the sensor nodes. As can be seen the outline traced is very rough. This was due to unsteady hand movement while tracing with an out stretched arm. Nevertheless, the system achieved an accuracy of around 72 percent in this test.

In the third test the outline of a cylinder of radius 0.12 meters was traced by the human arm with the sensor nodes. Expected should be a circle of radius 0.12 meters. Figure 4.14 shows the results of the tracking algorithm.



(a) (b)

Figure 4.14: Tracing the outside of a Cylinder (a) Reference Circle Outline (b) Tracing Algorithm Output

The result shows a very rough circle. The reason being that the straightened human arm was unsteady while tracing the outline. And the medial rotation about the shoulder may also account for some of the error. Tracing was done by moving the closed fist around the surface of the cylinder. This led to a shaky outline of the cylinder. The approximate radius of the traced circle after accounting for the offset of the sensor by the fist is 0.21 meters. Time taken to complete this test was 12.6 second. Once again this time is calculated using the receive times of the messages from the sensor nodes. So, for this case the accuracy is 60 percent.

Given the accuracy in the second and third test are a bit low which can be improved, the tests were outside the scope of the intended purpose. The intended purpose is the range of motion tests which usually involve arm movement in a single direction like the first test which has acceptable accuracy.

4.5 Speed test

This experiment was conducted to show the tracking results when the arm is moved at increasing speed. The arm again started vertical and hanging by the side. Then the arm was raised to the front. This movement was repeated three times at increasing arm movement speed. The first movement is considered as the reference speed and the subsequent movements are faster than this reference movement. The angular speed was calculated by the algorithm using Eq. (4.1).

$$Calculated\ Angular\ Speed = \frac{Distance\ travelled\ by\ end{-}effector}{ArmLength * Time\ taken} \tag{4.1}$$

Figure 4.15 shows the results of the tracking algorithm.

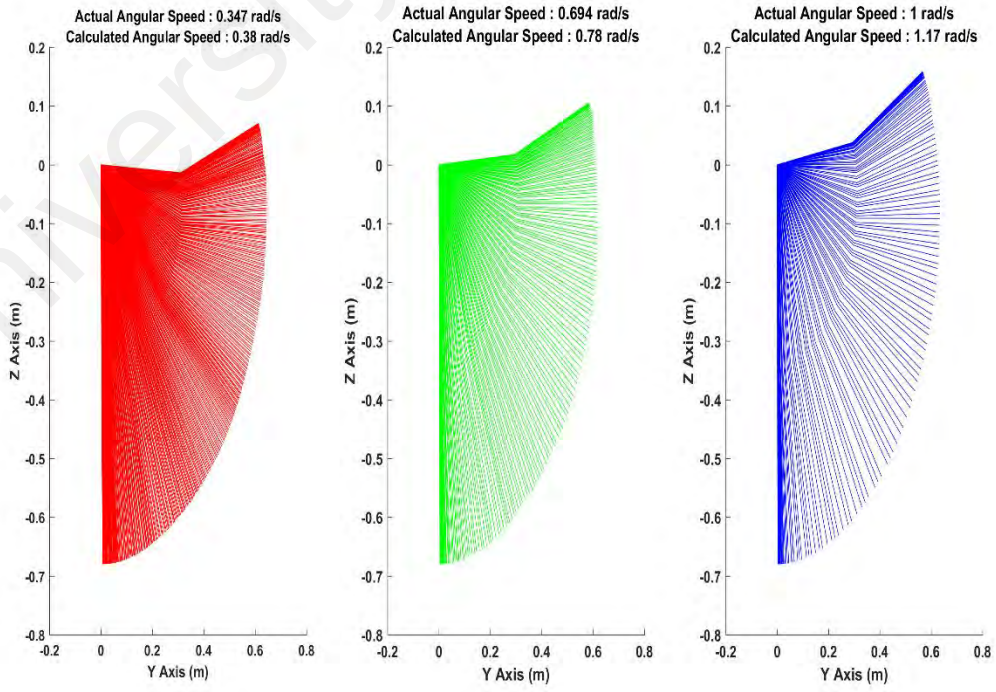


Figure 4.15: Shoulder Flexion at different speeds

The actual speed was calculated by physically measuring the total angle the arm was rotated and then dividing it by the time taken for the movement. This is shown in Eq. (4.2).

$$Actual\ Angular\ Speed = \frac{Acual\ Angle\ of\ Rotation}{Time\ taken\ for\ movement} \quad (4.2)$$

As can be seen from the results that at lower speeds the resolution of the result is very fine. The resolution decreases at increasing speeds. Nevertheless, the resolution of tracking is enough to make sense of the movement. This is enough for our purpose where the stroke patient is expected to conduct these movements at a normal pace.

4.6 Limitations

As mentioned in the previous section the differentiable range of motion of the Y axis of the IMU sensor node is limited to theoretically 180 degrees which is -90° to 90°. But depending on the start orientation and positioning of the sensor node the range of motion can be even more limited. We assigned this axis for the medial and lateral rotation (figure 4.8) of the shoulder since the stroke rehabilitation exercises use less of this rotation and more of the other rotations.

Another limitation with our sensor node design is that our sensor node is thick. The reason being that since this was intended as a prototype through hole components were used which have larger dimensions than SMD components. This causes the IMU sensor to be slightly elevated from the mount point due to the other components being underneath. This causes un-wanted motion in the case of elbow flexion and shoulder lateral and medial rotations which give an imprecision in the results.

Chapter 5: Conclusion

5.1 Summary

The first objective of this project was to design a modular IMU based wireless sensor system. The IMU sensor namely BNO055 was interfaced with the Node MCU micro-controller. The power source for this was a 9-volt battery together with a voltage regulator to regulate the voltage at 5 volts for the sensor node. All these components were soldered onto a double-sided PCB and Velcro straps were attached to make the node wearable. Two of these sensor nodes were built. The Node MCU micro-controller has a built in Wi-Fi chip that was used to connect to a computer via a Wi-Fi router. Network analysis results show a high frequency of message transmission and low transmission delay for our developed system. These results make our wireless sensor network system suitable for human arm movement tracking.

The second objective of this project was to develop a human arm position tracking algorithm. When the sensor nodes are switched on, they are connected to the computer via a Wi-Fi router. A program written in Python receives and deciphers the messages sent by the sensor nodes. To ensure high frequency of transmission from the sensor nodes all the analysis is done on the computer using Python. To track the human arm movement using the orientation data from the sensor nodes, a mathematical model was developed for the arm using Denavit-Hartenberg convention. Using this model and data from the sensor nodes on the arm, Forward Kinematics was performed on the data. The results of all the various movement tests show that this sensor system can track arm movements well for two of the three degrees of freedom for the shoulder joint. The lateral and medial rotations of the shoulder become erroneous past a certain point during rotation. This is due to the limited range of motion issue with the Y axis output of the BNO055 IMU sensor. For the purpose of tracking range of motion exercises for stroke patients, this issue

is mitigated because the shoulder lateral and medial rotations are kept to the minimum. The accuracy of tracking is about 80 percent for basic arm movements such as shoulder and elbow flexion and contraction and shoulder abduction and adduction. Accuracy for tracing complicated shapes is between 60 and 70 percent which is primarily due to human error in manually tracing objects and due to the limitations in the shoulder medial and lateral rotations.

5.2 Future Recommendations

The dimensions of the sensor nodes can be reduced by making a custom printed circuit board (PCB) with the IMU sensor chip along with the micro-controller chip and voltage regulator and complementary electronics. A battery with a smaller form factor like a Lithium polymer battery can be used. These changes will make the sensor node thinner which will make the IMU sensor attach more securely to the arm and prevent error from shaking and slipping.

A higher accuracy IMU can also be used. Also, the DH parameters can be converted to use quaternions which are more accurate. Quaternions are better because they avoid problems like the gimbal lock which is linked to rotations using Euler angles.

Artificial Intelligence can also be implemented. A neural network model can be trained with a large dataset of arm movements so that the AI can recognise what movement the sensor node wearer has performed. An AI model trained with dataset of stroke patient arm movements can also be used to identify the type of stroke suffered by the patient and its severity. This can be used to relay diagnostic data to a doctor in the case that the doctor and patient are geographically separated.

Automatic data logging can also be implemented later. This can be done by using accelerometer data to determine if the arm has moved. If so, data logging can start and stop when the arm stops moving.

References

Asada, H. H., Shaltis, P., Reisner, A., Rhee, S., & Hutchinson, R. C. (2003). Mobile monitoring with wearable photoplethysmographic biosensors. *IEEE engineering in medicine and biology magazine, 22*(3), 28-40.

Atrsaei, A., Salarieh, H., Alasty, A., & Abediny, M. (2018). Human arm motion tracking by inertial/magnetic sensors using unscented Kalman filter and relative motion constraint. *Journal of Intelligent & Robotic Systems, 90*(1-2), 161-170.

Benjamin, E. J., Blaha, M. J., Chiuve, S. E., Cushman, M., Das, S. R., Deo, R., Floyd, J., Fornage, M., Gillespie, C., & Isasi, C. (2017). Heart disease and stroke statistics-2017 update: a report from the American Heart Association. *Circulation, 135*(10), e146-e603.

Calautti, C., Jones, P. S., Persaud, N., Guincestre, J. Y., Naccarato, M., Warburton, E. A., & Baron, J. C. (2006). Quantification of index tapping regularity after stroke with tri-axial accelerometry. *Brain Res Bull, 70*(1), 1-7. doi:10.1016/j.brainresbull.2005.11.001

Corbishley, P., & Rodriguez-Villegas, E. (2008). Breathing detection: towards a miniaturized, wearable, battery-operated monitoring system. *IEEE Trans Biomed Eng, 55*(1), 196-204. doi:10.1109/tbme.2007.910679

Curone, D., Secco, E. L., Tognetti, A., Loriga, G., Dudnik, G., Risatti, M., Whyte, R., Bonfiglio, A., & Magenes, G. (2010). Smart Garments for Emergency Operators: The ProeTEX Project. *IEEE Transactions on Information Technology in Biomedicine, 14*(3), 694-701. doi:10.1109/TITB.2010.2045003

Dudde, R., Vering, T., Piechotta, G., & Hintsche, R. (2006). Computer-aided continuous drug infusion: setup and test of a mobile closed-loop system for the continuous automated infusion of insulin. *IEEE Trans Inf Technol Biomed, 10*(2), 395-402.

Evans, B. (2011). *Beginning Arduino Programming*: Apress.

Gebruers, N., Vanroy, C., Truijen, S., Engelborghs, S., & De Deyn, P. P. (2010). Monitoring of physical activity after stroke: a systematic review of accelerometry-based measures. *Arch Phys Med Rehabil, 91*(2), 288-297. doi:10.1016/j.apmr.2009.10.025

Gu, X., Zhang, Y., Sun, W., Bian, Y., Zhou, D., & Kristensson, P. O. (2016). *Dexmo: An Inexpensive and Lightweight Mechanical Exoskeleton for Motion Capture and Force Feedback in VR*. Paper presented at the Proceedings of the 2016 CHI

Conference on Human Factors in Computing Systems, San Jose, California, USA.

Lipkin, H. (2005). *A note on Denavit-Hartenberg notation in robotics.* Paper presented at the ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

Oujamaa, L., Relave, I., Froger, J., Mottet, D., & Pelissier, J. Y. (2009). Rehabilitation of arm function after stroke. Literature review. *Annals of Physical and Rehabilitation Medicine, 52*(3), 269-293. doi:https://doi.org/10.1016/j.rehab.2008.10.003

Patel, S., Park, H., Bonato, P., Chan, L., & Rodgers, M. (2012). A review of wearable sensors and systems with application in rehabilitation. *Journal of NeuroEngineering and Rehabilitation, 9*(1), 21. doi:10.1186/1743-0003-9-21

Patterson, J. A. C., McIlwraith, D. C., & Yang, G. (2009, 3-5 June 2009). *A Flexible, Low Noise Reflective PPG Sensor Platform for Ear-Worn Heart Rate Monitoring.* Paper presented at the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks.

Rand, D., & Eng, J. J. (2015). Predicting daily use of the affected upper extremity 1 year after stroke. *J Stroke Cerebrovasc Dis, 24*(2), 274-283. doi:10.1016/j.jstrokecerebrovasdis.2014.07.039

Rhodes, B., & Goerzen, J. (2014). *Foundations of Python network programming*: Apress.

Šimaitytė, M., Petrėnas, A., Kravčenko, J., Kaldoudi, E., & Marozas, V. (2019). Objective evaluation of physical activity pattern using smart devices. *Scientific Reports, 9*(1), 2006. doi:10.1038/s41598-019-38638-z

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*.

Staff, M. C. (2019, 17/4/2019). Stroke rehabilitation: What to expect as you recover. Retrieved from https://www.mayoclinic.org/diseases-conditions/stroke/in-depth/stroke-rehabilitation/art-20045172

Tian, Y., Meng, X., Tao, D., Liu, D., & Feng, C. (2015). Upper limb motion tracking with the integration of IMU and Kinect. *Neurocomputing, 159*, 207-218.

Wang, Q., Chen, W.-P., Zheng, R., Lee, K., & Sha, L. (2003). *Acoustic Target Tracking Using Tiny Wireless Sensor Devices*, Berlin, Heidelberg.

Zaccone, G. (2015). *Python parallel programming cookbook*: Packt Publishing Ltd.

Zhu, k. S., Liang. (2016). *Motion Control in VR — Real-time Upper Limb Tracking via IMU and Flex Sensor*.