DEVELOPMENT OF HAND GESTURE GLOVE TO CONTROL ROBOT ARM

GHEE YIH FARN
B071310234

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2016

B071310234 BACHELOR OF ELECTRICAL ENG. TECH. (IND. AUTOMATION AND ROBOTICS) 2016 UTeM

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT OF HAND GESTURE GLOVE TO CONTROL ROBOT ARM

This report submitted in accordance with requirement of the Universiti Teknikal Malaysia Melaka (UTeM) for the Bachelor Degree of Engineering Technology (Industrial Automation and Robotics) with Honours

by

GHEE YIH FARN

B071310234

FACULTY OF ENGINEERING TECHNOLOGY

2016

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BORANG PENGESAHAN STATUS LAPORAN PROJEK SARJANA MUDA

TAJUK: **Development of Hand Gesture Glove to Control Robot Arm**

SESI PENGAJIAN: **2016/17 Semester 1**

Saya **GHEE YIH FARN**

mengaku membenarkan Laporan PSM ini disimpan di Perpustakaan Universiti Teknikal Malaysia Melaka (UTeM) dengan syarat-syarat kegunaan seperti berikut:

1. Laporan PSM adalah hak milik Universiti Teknikal Malaysia Melaka dan penulis.
2. Perpustakaan Universiti Teknikal Malaysia Melaka dibenarkan membuat salinan untuk tujuan pengajian sahaja dengan izin penulis.
3. Perpustakaan dibenarkan membuat salinan laporan PSM ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (✓)

☐ SULIT (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☐ TERHAD (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia sebagaimana yang termaktub dalam AKTA RAHSIA RASMI 1972)

☐ TIDAK TERHAD

Disahkan oleh:

____________________ ____________________

Alamat Tetap

Cop Rasmi:

____

____

____

** Jika Laporan PSM ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan PSM ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I hereby, declared this report entitled "Development of Hand Gesture Glove to Control Robot Arm" is the results of my own research except as cited in references.

Signature	:	…………………………………………….
Name	:	GHEE YIH FARN
Date	:	09 December 2016

APPROVAL

This report is submitted to the Faculty of Engineering Technology of UTeM as a partial fulfillment of the requirements for the degree of Bachelor of Engineering Technology (Industrial Automation and Robotics) with Honours. The member of the supervisory is as follow:

……………………………………

(EN MOHD HANIF BIN CHE HASAN)

ABSTRAK

Sistem Gerak Isyarat Tangan (HGG) adalah alat kawalan untuk menggunakan Interaksi Manusia-Mesin konsep (HMI). Bagi HMI adalah teknologi interaksi yang terbagus antara seorang manusia dan mesin. Yang pentingnya bagi konsep HMI adalah menyebabkan mesin atau robot boleh menggantikan manusia untuk melakukan lebih banyak kerja di kawasan berbahaya dan juga di kawasan-kawasan yang tidak boleh dilarangkan oleh manusia. Bagi Gerak Isyarat Tangan peranti kawalan adalah kawalan dengan isyarat tangan manusia. Konstruk ini adalah berdasarkan 3 Axis meter pecutan, sensor flex dan termasuk teknologi antara muka wayarles. Dan kawalan tindakan peranti atau gerakan ini adalah kebebasan dan fleksibel daripada pengawal peranti lain seperti "keyboard", "joystick" dan "PC mouse". Kegunaan untuk projek ini adalah untuk Robotic Arm.

ABSTRACT

A Hand Gesture Glove (HGG) is a control device, using Human-Machine Interaction (HMI) concept. A HMI is great interaction technology between a human being and a machine. An important the HMI in Robot arm that is because the machine or robot can replace human to do more work in dangerous areas and also in the areas which cannot be accessed by human. The Hand Gesture Glove control device is control by human hand gesture. This is construct based on 3 Axis Accelerometer, flex sensor and including the wireless interface technology. And this control device action or motion movement is freedom and flexible than the other device controller such a keyboard, joystick and PC mouse. The developed devise is applied to Robotic Arm and further evaluated it performance.

DEDICATIONS

To my beloved parents

To my beloved family members

To my trusty friends

ACKNOWLEDGMENTS

By the way, I wanted to thank a lot for him my supervisor, En Mohd Hanif Bin Che Hasan encouragement, suggestions and trusted during the period of this first semester Final Year Project (FYP). My supervisor, is very smart person is because him are try to explain more and give more clearly example with new ideal for me. My supervisor, discuss with me how to write a good FYP report and also get the ideal about how to develop hardware project. Last, I also want to thank my trusted friends. They also given a good response when I asking a question during my FYP section and also example the ideal for me to smoothly finish this first semester FYP project this semester.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF EQUATION

LIST OF SYMBOLS AND ABBREVIATIONS

HGG	=	Hand Gesture Glove
HMI	=	Human-Machine Interaction
FYP	=	Final Year Project
IR	=	Industrial Robot
MEMS	=	Micro Electro Mechanical System
3D	=	Three Dimension
X	=	X-Axis
Y	=	Y-Axis
Z	=	Z-Axis
DOF’s	=	Degree of Freedoms
LED	=	Light Emitting Diode
ARM	=	Acorn/Advanced RISC Machine
PIC	=	Peripheral Interface Controller
SR	=	Synchronous Sign Magnitude Rectification
DC	=	Direct Current
PWM	=	Pulse Width Modulation
AVR	=	Modified Harvard Architecture 8-bit RISC
RISC	=	Reduced Instruction Set Computing
ROM	=	Read-Only Memory
EPROM	=	Erasable Programmable Read-Only Memory
EEPROM	=	Electrically Erasable Programmable Read-Only Memory
KHU-1	=	3D Hand Motion Tracking and Gesture Recognition System
RC car	=	Radio Controlled
PC	=	Personal Computer
NXT	=	Next Generation
RF	=	Radio Frequency
PID	=	Proportional Integral Derivative
UTeM	=	Universiti Teknikal Malaysia Melaka
IDE	=	Integrated Development Enviroment

USB	=	Universal Serial Bus
COM Port	=	Serial Port
EMA	=	Exponential Moving Average
UNO	=	Arduino UNO
WF	=	World Frame
G	=	Gripper Frame
W	=	Wrist Frame
S	=	Shoulder Frame
E	=	Elbow Frame
B	=	Base Frame
T	=	Transition
T1	=	Transition 1
T2	=	Transition 2
T3	=	Transition 3
DH-Table	=	Denavit-Hartenberg Table
TX	=	Transmission Bluetooth
RX	=	Receiver Bluetooth

CHAPTER 1

INTRODUCTION

What is Robotics? When mentioning robots, many people will think about machines with hands and feet. That is because many type of scientific movies, entertainment and toy stores are showing this kind of robot. Actually, robotic engineer can develop many shape or type of robot to helping people to do more difficulty work such as robotic arm, car, quadcopter and other type of robot. Difference type of robot got deference types of robot application. There are Industrial robots, Domestic or household robots, Medical robots, Service robots, Military robots, Entertainment robots, Scape robots, Hobby and Competition robots. In this paper, just want to show only the type of robot application is Industrial robots. The Industrial robots are robots used in an industrial manufacturing environment. Usually these are articulated arms specifically developed for such applications as welding, material handling, painting and other.

1.0 Background Project

A gesture is a non-verbal or non-vocal communication which produced by our human body. Gesture also can call that as a human body language. To study human gesture also given many type of gesture such as hand, arm, leg, body and face.

In this paper, just focusing how to use the human gesture to make an instruction to ask the robot to do work. Mean that to convert the gesture signal produced by human hand or arm to a form of digital signal to control robot to do work like similar with human. The concept used in this paper is called as Human-Machine Interaction (HMI). The hand gesture includes several type gesture there are Palmar or Tripod grasp, Lateral grasp, Wrist Extension, Wrist Flexion, Move Hand or Arm Forward or Backward, Move Arm Left and Right, Cylindrical grasp, and Grasping.

The human gesture can easily the snap or capture using the type of glove including some electronic sensor devices with the high performance microcontroller and to communication with Industrial Robot Arm in professional can apply high technology wireless interface. The electronic sensor devices are only 3 Axis Accelerometer Sensor and Flex Sensor is choosing. Because the 3 Axis Accelerometer can provide 3D orientation and movement that is slightly similar with human gesture movement or orientation. Flex Sensor is using for controlling the gripper or other tool can apply at Industrial Robot Arm. This system can recognize any sampled data saved in the database while promoting maximum portability and mobility to the user via wireless Bluetooth technology rates relationships.

1.1 Objectives

The objectives of the project are as follows:

a. To study the concept of hand gesture glove to control robot arm wireless interface.
b. To design the hand gesture glove to control robot arm.
c. To analysis the performance whether the hand gesture glove can be smoothly and directly replacing other devices controller to control robot arm.

1.2 Scope

The scopes covered for this project is the wireless hand gesture control device. It device is construct based on accelerometer sensing in 3D environment. Implementation allows user to wear a set of hardware (glove) and control a robot movement through different hand movement and orientation. The Robot Arm movement directions including move in up, down, left or right and pick up some objects form another place and beside that the movements of the human hand gesture by using accelerometer sensor to scan and follow the acceleration of human hand movement. The scope of accelerometer sensor can be provided 3D movement similar with human gesture.

For the robot arm gripper that will be using flex sensor to control whether grip open or close because flex sensor is a passive resistive device that can be used to detect bending. There is a change in resistance depending on the amount of bend on the sensor. As the amount of bend increases the resistance increases and the gripper will be closed.

The type of wireless interface is using Bluetooth device to communicate in between hand gesture glove and robot arm. Every signal receives or transmit that will be manipulate by microcontroller Arduino Uno.

The performance analysis is focusing to get the best wireless communication between hand gesture glove and robot arm. Next, is get the best performance analysis when the position movement of robot arm is exactly similar will the hand gesture glove position movement are provided.

1.3 Problem Statement

Now, the world wide is near to the new technology world. The new technology world is coming to changing our human to be stress and hard to control the new technology for certain people are no scholar and also focus to make human become lazier to do more work. For example, the new technology now is controlled by using keyboard, joystick, switch button and include touch screen sensor. That will become more difficult for our human are no scholar to control. Since, now many of people is still having a problem to find permanent joy for their life because of not enough of knowledge and less confidence for controlling the Robot or Machine. That will be a big problem statement because of coming a new technology world.

At the same time, our robotic engineer is developing some new controller to replacing the traditional controller devices. The new controller device is using Human-Machine Interaction concept. This Human-Machine Interaction concept has called as a HMI concept. The structure is using combination of our human and machine to interaction. The functionality of this HMI concept is using our human gesture to communicate with the machine or robot. Other way to describe in this HMI concept is the development in science and robotic combine. Gesture recognition can be considered as a way for computer to understand human body language.

An important the Human-Machine interaction that is because the robot can replace human to do work in dangerous areas and also in the areas which cannot be accessed by human. The robot arm is controlled by hand gesture is because the action or motion movement is freedom than the robot arm where controlled by keyboard or computer. It controlled by keyboard or computer are not a flexible motion are providing and cannot provider a good work at the nature environment. For example, robotics with wireless hand gesture controller can be applied to control wheelchairs for the disabled or robotics arm for bomb diffusion.

The comparison with the traditional controller devices even those have performed transition signal but still limit distance of control the Robot Arm. The maximum distance for using wire interface is around 100m only. When the Robot Arm controller is upgrade to use wireless interface. The control distance will be increase between 1km or will be more depending which type the wireless interface devices are using. The advantage for using this wireless interface is to reduced extra work to arrange back the wire after using. They have several type of wireless devices such as RF modular, Bluetooth and Network communication.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

This chapter discussed about the literature review. A literature review discusses published subject area, and sometimes information in a particular subject area within a certain time period. A literature review is to simple summary of the sources, but it usually has an organizational pattern and combines both summary and synthesis before the new project coming out. This chapter contains the theory and implementation of the components, equipment and programming language used in the previous project.

2.1 Method of Control

Many type of method control are mentioned previous project; they still use Human-Machine Interaction concept to do their project. The first method is used human hand gesture to control or communicate their Robotic Arm with wirelessly. Which is the transmission controller device is fixed on the gloves and wear by human hands than can easily to controlling it freedom like a conventional remote controller [1]. Actually, the construction of the human hand gesture control Robot Arm consists of two parts. One of the part is develop as a transmitter part and other part is receiver part [2]. When the comparison in between using gesture recognition and type of traditional devices controller is showed that the traditional devices is more stable to control the Robot arm. Well, because of new technology most of people trying to use the human gesture as a controller to replace the traditional devices controller. The gesture recognition actually can be controlling the complete Robotic arm by providing the gesture motion such a movement and orientation [3], [4], [6]. Another method will be used an Artificial Neural Network (ANN) trained with a back-propagation algorithm was used to recognize arm gestures and postures, which then will be used as input in the control of the robot [5].

Other method will be used accelerometer sensor represented as a human gesture recognition to control wirelessly with different number of degree of freedoms (DOF's) [7]. Last method is to improve the stability and smooth of gesture recognition controller device to control the Robot Arm movement. The method is to interpret movement of gesture recognition via algorithm. Convert or interpret the gesture recognition provided to a form of software, hardware or combination of both type [8]. Next, the goal of research is to develop an application system which control its movement by the merely banding of a person's fingers [9]. The actually transmits the encoded signals representing the hand postures and dynamics gesture to the robot via RF link or Bluetooth [1-12], [14-16].

2.2 Controller Used

Arduino microcontroller is a controller device. Arduino board are able to read inputs such as light on a sensor, a finger on a button, or a Twitter message and turn it into an output such as activating motor, turning on a LED, publishing something online. While, can tell the board what to do by sending a set of instruction to the microcontroller on the board.

Paper	Transmitter	Receiver
[1]	Arduino Duemilanove	Arduino Mega
[2]	Arduino Uno	-
[3], [7],[24]	Arduino Uno	Arduino Uno
[4]	Arduino Uno	Arduino Leonard
[9]	Arduino Funnel I/O	Arduino Uno
[14]	Arduino Uno	Arduino Mega
[16]	Arduino ATMEGA8	ARM Cortex-M3
[18]	Arduino Funnel I/O	Arduino ATMEGA 328P
[19]	Arduino	-
[20],[21]	-	Arduino
[22]	Arduino ATMEGA328P	-
[26]	Arduino ATMEGAL168P	-

Table 2.1: Comparison for each paper used the type of controller to control robot.

The application is to read or given instruction movement of the Robot Arm. MotomanLib can read the input instruction produced from the accelerometers signal and interprets the received data through software application and then transmitted movement signal to the Robot Arm. This application is a Data Link Library created to control the Robot Arm wirelessly via Ethernet [5].

The Keil MicroVision controller selected to control fast enough and given respond or performance of operation in real-time. Here the controller does sinking the input data, processing it and running the per-defined algorithm stored in memory and accordingly provides the signals the outside world. This is very basic requirement and the system has a dedicated and specific task. So a controller that satisfies the basic requirement as well as have the desired peripherals like ADC (>8bit) and PWM generator can be used [6].

PIC is a family of modified Harvard architecture microcontrollers made by Microchip Technology. PIC originally developed by General Instrument's Microelectronics Division. The name PIC stand to Peripheral Interface Controller. When PIC microcontroller is used to control some Robot Arm. The term of electronic control system is called as SMART ARM. That is because the Robot Arm can perform a motion of movement controlled by the high performance PIC model series controller [13].

The Talon Controller is a simple controller, but very robust, PWM interface only, brake or coast jumper and Fan optional. When data processing unit sends the commands to the Talon's controller wirelessly via radio frequency. The Talon controller assembles all commands from the data glove and other sources into a package and sends them to the robot for execution. This architecture has the advantage of allowing traditional control of the robot as well as conforming to the robot's existing interface [13].

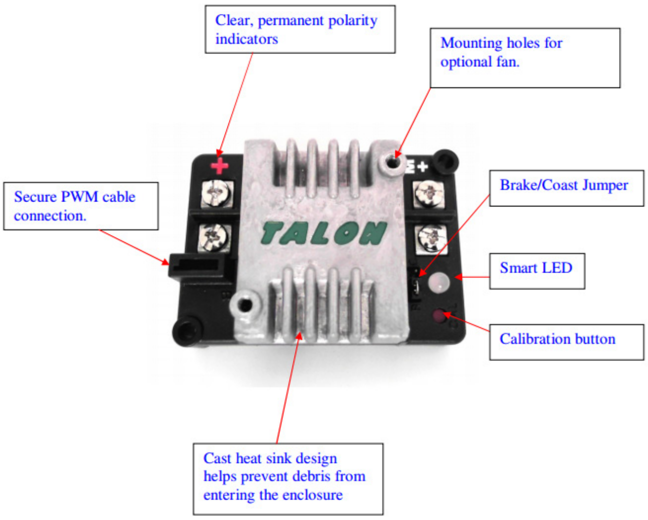


Figure 2.1: Talon Device Overview [13].

The AVR is a modified Harvard architecture 8-bit RISC single-chip microcontroller. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage. An Atmel AT89C51 microcontroller is use to process the most important part of the robot. Tills now to get the data from the decoder. Based on that data decisions have to be made. So here the role of microcontroller is used [11]. An Atmel AVR ATmega128 used as a controller for development of the KHU-1 data glove [12]. An Atmel AVR AT90LS8535 microcontroller on the forearm controller converts the analogue signal for the accelerometer to a digital signal [15].

2.3 Algorithms

Step by step procedure designed to perform an operation, and which like a map or flowchart will lead to the sought result if followed correctly. Algorithms is a system to calculate or produces the similar output information when given the similar input information, while the several simple algorithms can be combined to perform complex tasks such as writing a computer program.

The gesture recognition system is a system can be interpreting the gesture movement via algorithms. This type of algorithms method is to convert the gesture given by human movement to a form of software, hardware or combination of both

[8]. The human with computer interaction system is a method of algorithms. This algorithm for human gesture recognition can be calculating the coordinate of sensor movement parameter, division by default angle and resultant the motion of human gesture is provided. While this algorithm for human hand gesture recognition is for noise reduction. The algorithms can be average filter by using formula to control the noise of motion and get the original smooth hand gesture recognition result [12].

The smart arm is named by people. Because the robotic arm design included the control solution is implementation by self-develop computer software. Form the 3D virtual arm simulation, the robotic arm is able to reach specified coordinates through the means of direct kinematics and inverse kinematics. Therefore, both methods were reciprocal to each other is joint, reciprocal, XYZ and simulation [13].

Other algorithms to improve the stability movement and to detect the orientation of 6-DOF Robotic Arm movement by adding gyro meter with accelerometer. The displacement angle of human hand or arm is an important motion of movement for 6-DOF Robotic Arm. Then can transform it with geometrical calculation for accelerometer and gyro component signal. The algorithms for accelerometer is to calculate gravitation acceleration Z axis. For gyro component is to calculate the rotation X and Y axis [14]. A glove structure including with 2-axis accelerometers on the finger tips and back of the hand has been built using commercial off the shelf component. For the algorithms to plot of magnitude accelerometer and angle over time for various hand gestures [15]. On one paper research mention for future research wireless data glove will have an input-filtering algorithm to minimize unintended commands originating form noise that produce shock and vibration, which can perturb the natural of the hand and fingers [10].

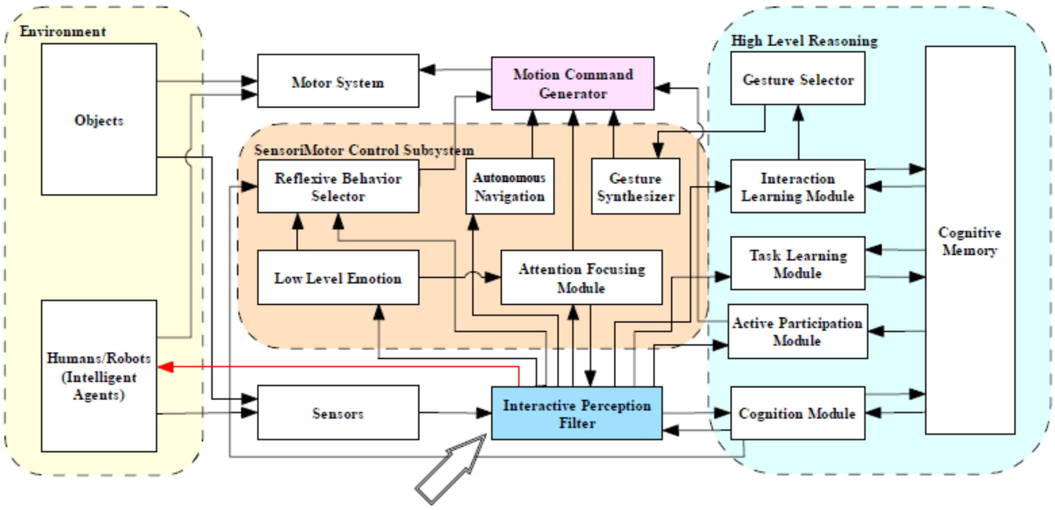


Figure 2.2: The Overall Design for Interactive Perception Filter [17].

The problem is to delegate this task to the high level reasoning subsystem, but this technique will complicate the job of this high level module significantly because it has to deal with three problems at the same time there is sensor noise attenuation, unintended behaviour attenuation in order to focus on intended behaviour only, contest analysis, interaction atmosphere detection, and alignment with the actions of other intelligent agents. This interactive perception module can overcome some of the limitations of classical passive perception modules [17].

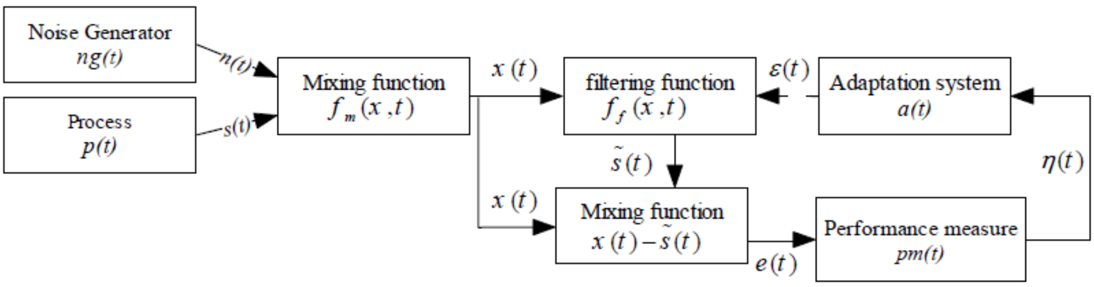


Figure 2.3: Passive Noise Filtering [17].

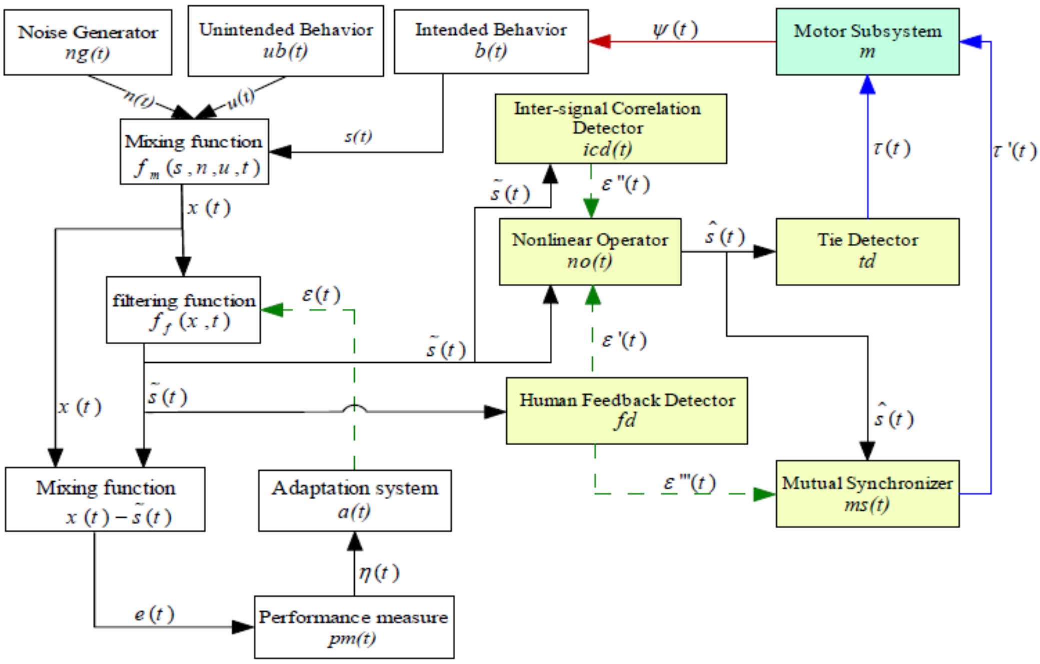


Figure 2.4: Active Perception Block Diagram [17].

2.4 Sensor Glove

Based on literature review so far, we can summarize type of sensor used in various researchers. The type of sensor including Accelerometer, Gyroscope, Flex, Force, Bluetooth, RF module, Glove, and Computer. We also can categories sensor part including Accelerometer, Flex and Force sensor. Communication part including Bluetooth and RF module. Location apply in between Glove or Computer.

Component / Paper	Accelerometer	gyroscope	Flex sensor	Force sensor	Bluetooth	RF module	Glove	Computer
[1]	✓	✓	✓			✓	✓	
[2]	✓					✓	✓	
[3]		✓	✓	✓	✓		✓	
[4]	✓		✓			✓	✓	
[5]	✓				✓		✓	
[6]	✓						✓	
[7]	✓					✓	✓	
[8]	✓				✓		✓	
[9]			✓		✓		✓	
[10]	✓	✓					✓	
[11]	✓				✓		✓	
[12]	✓				✓		✓	
[13]								✓
[14]					✓		✓	
[15]	✓				✓		✓	
[16]	✓		✓		✓		✓	
[17]								✓
[18]			✓		✓		✓	
[19]	✓		✓		✓		✓	
[20]								✓
[21]								✓
[22]					✓			
[23]							✓	
[24]			✓		✓		✓	
[25]	✓	✓				✓	✓	
[26]					✓			

Table 2.2: Comparison for each paper used the type of sensors.

2.5 Location Applied

Based on literature review so far, we can summarize the location applied used by various researchers and sort by according the years. The years including from 1999 to 2015 is separated according by colours.

Paper	Location Applied	Years
[1]	Glove and robot arm	2013
[2]	Glove and Robot	2015
[3]	Glove and RC car	2014
[4]	Glove and Mouse PC	2014
[5]	Glove and Industrial robot	2009
[6]	Glove and Robotic arm	2012
[7]	Glove, Robot arm and Platform	2013
[8]	Glove and Computer (Database)	2014
[9]	Glove and Wheel chair	2013
[10]	Glove and Talon Robot	2009
[11]	Glove and Robot	2013
[12]	Glove and Computer (Data Glove)	2009
[13]	Robot arm	2011
[14]	Glove and Robotic arm	2014
[15]	Glove and Human computer interaction	1999
[16]	Gesture recognition and Robotic arm	2013
[17]	Robot, Human, and computer	-
[18]	Glove and Wheel Chair	2011
[19]	Glove and NXT Robot	2014
[20]	Computer and Robot Car using DC Motor	2013
[21]	Computer, DC motor and Robocar	2014
[22]	EMG electrodes and CM-530 humanoid robot	2014
[23]	Glove, Handroid	2012
[24]	Glove, Robot Arm and Hand (development)	2015
[25]	Wearable Input Device	2005
[26]	Humanoid Robot	2013

Table 2.3: Comparison for each paper which location applied.

2.6 Advantage

Advantage of each project is very important for each engineer, that will cause the benefit for user to use their project or product. The main advantage for the hand gestures remote controlled robotic arm is to interface wirelessly in between 1000 meter of communication distance. The interface in between human and robot if far communication distance provided, that will cause the project more effective to do more work at far distance [1]. Accelerometer base gesture controlled robot using Arduino that will provided a real time palm gesture recognition, this will become more effective and get the natural way of controlling robots. By the way, many existing system controllers have used Bluetooth wireless control device which to replace by RF modules is because the communication distance for both devices are differences. RF modules can achieve the requirement for control for long distances, that covers 3KHz to 300GHz, so it basically includes Bluetooth is only covers 2.4GHz [2]. Magic glove-wireless hand gesture hardware controller included various of sensors to get the human hand movement and to control the robot [3].

The main advantage of the wireless gesture controlled systems can be easily control the various of sensors to capture all the action and motion of an arm thereby transmit all the digital signal using wireless interface to control the robot [4]. The achieved recognition rate of gestures and postures 92% should be improved in future, keeping the compromise with the system response time 160ms [5]. For the design and implementation of a wireless gesture controlled robotic arm with vision not only using hand gesture but also can controlled by human leg gesture [7]. Some wireless gesture recognition system using MEMS accelerometer to controlled robot, but before that still need to interpret the gesture are in the form of software, hardware or combination of both. This is the one of advantage include database system embedded in hand held device for more mobility and portability [8], [12], [13]. Design and development of a hand-glove controlled robot based on MEMS, the objective for this design and development hand-glove which is believed to provide better control to people with severe disabilities in comparison to the traditional joystick-controlled because of the lesser amount of force required to manipulate the integration of an obstacle avoidance and collision detection system [9], [18].

Wireless data glove for gesture based robotic control is already a completed control device by using wireless interface and human-machine interface, but for future research this type of controller such be able to have an input-filtering algorithm is to minimize unintended commands originating form noise that produce shock and vibration, which can perturb the natural of the hand and fingers [10]. Gyro-accelerometer based control of a robotic arm using AVR microcontroller. But the advantage for using this 3 axis gyro-meter which can program and find out Z-axis rotational angle to makes the control system easier. This Gyro system can have improved this control system more stable and synchronous response [14]. The acceleration sensing glove which is taking an advantage of gravity induced acceleration offset, have able to detect and identify unwanted statics gesture [15]. Mostly the potential application for the using accelerometer sensor, they are a wireless wearable mouse pointing device, a wireless wearable keyboard, hand motion and gesture recognition tools, virtual musical instruments, computer sporting games, and work training in a simulated environment. Comparison for other common input devices tech pendant, this approach using accelerometer over wireless medium is more easy to work [16].

2.7 Conclusion

This chapter is to study the related research for the scope of this project Development of Hand Gesture Glove to Control Robot Arm. Here, the conclusion for the method of control is using human hand gesture interaction to control robot arm with wireless interface technology. All of this is controlled by 2 Arduino UNO Microcontroller. One of the Microcontroller will paired with Accelerometer and flex sensor to develop a glove to become a transmit controller system. Other one is become a receiver controller system to receive the transmit signal and convert to digital signal for robotic arm. In between the transmission of this control system, the Bluetooth device are choosing as a wireless interface technology.

Last, the development of the Hand Gesture Glove to Control Robot Arm is also including some algorithm to feel the motion of Robot Arm move with smoothly and work as high performance. All of this selection component will be depend to the costing, quality, effective and expectation for this project development.

CHAPTER 3

PROJECT METHODOLOGY

3.0 Introduction

This chapter is to explained the method to developing the Hand Gesture Glove to Control Robot Arm by present in flow chart. To make sure that progress of this project can be achieved. The overall of this process are divided into a four sequence of program.

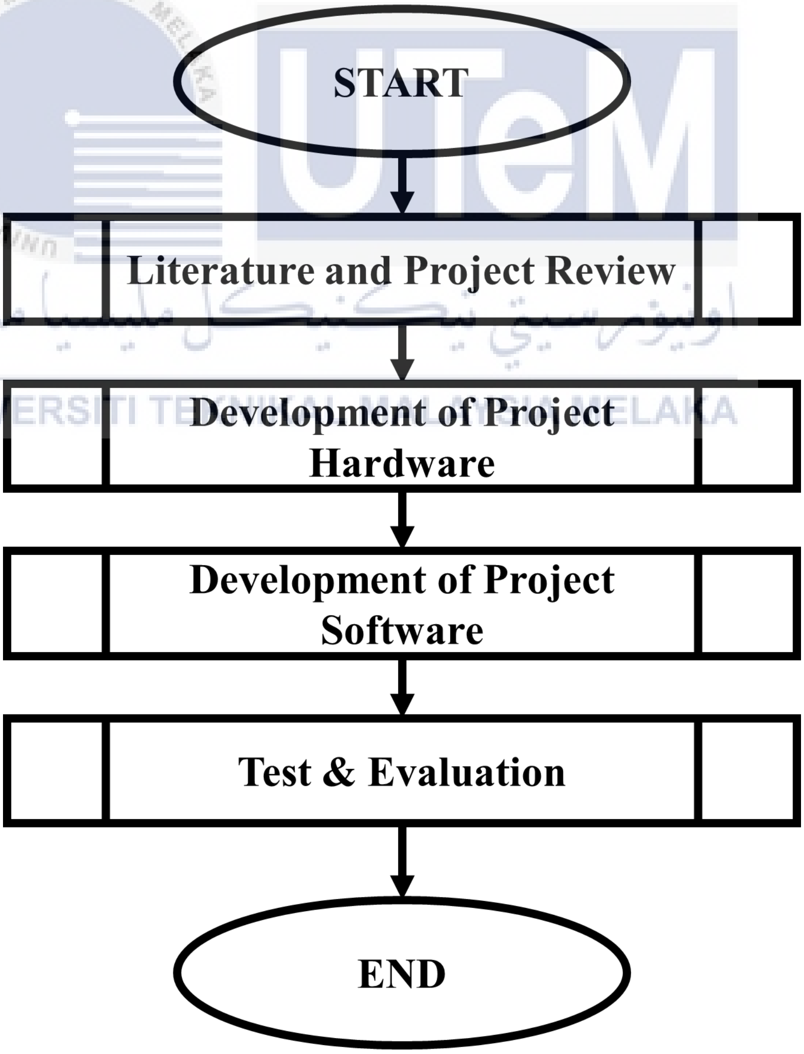


Figure 3.1: Methodology Flow Chart.

3.1 Literature and Project Review

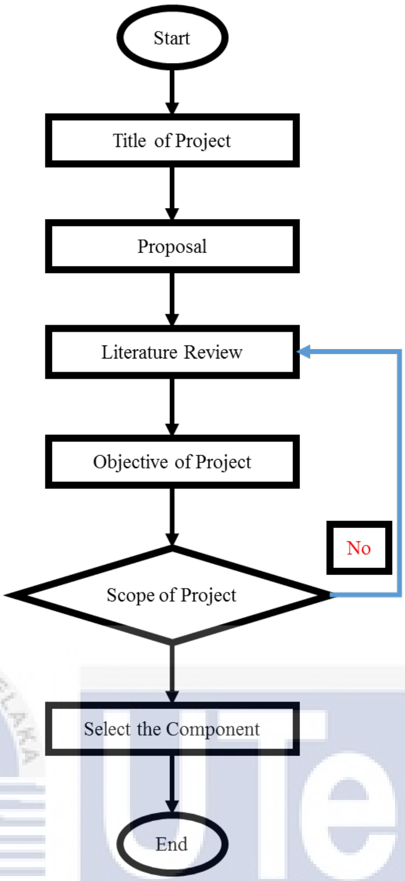


Figure 3.2: Flow Chart for Literature and Project Review.

Literature and Project Review	
Activity 1	Introduction of PSM 1 and find the suitable tittle with your supervisor are choose.
Activity 2	Provide proposal including project briefing, problem statement, work scope, project methodology, result expectation, reference and project planning.
Activity 3	Literature review
Activity 4	Select component

Table 3.1: Literature and Project Review for Methodology Flow Chart.

3.2 Development of Project Hardware

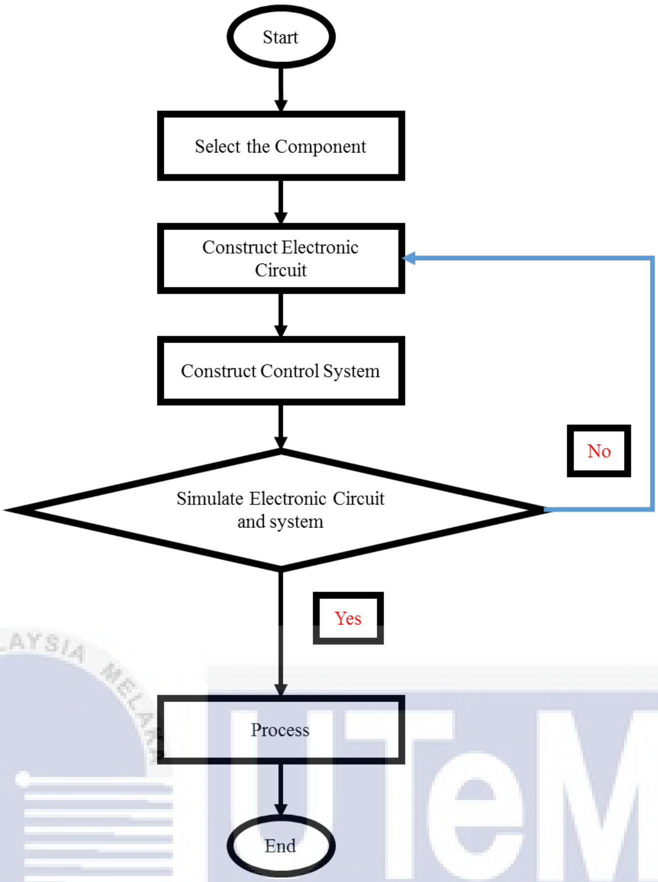


Figure 3.3: Flow Chart for Development of Project Hardware.

Development of Project Hardware	
Activity 5	Electronic Circuit Design
Activity 6	Simulation both mechanical and electronic design

Table 3.2: Development of Project Hardware for Methodology Flow Chart.

3.3 Development of Project Software

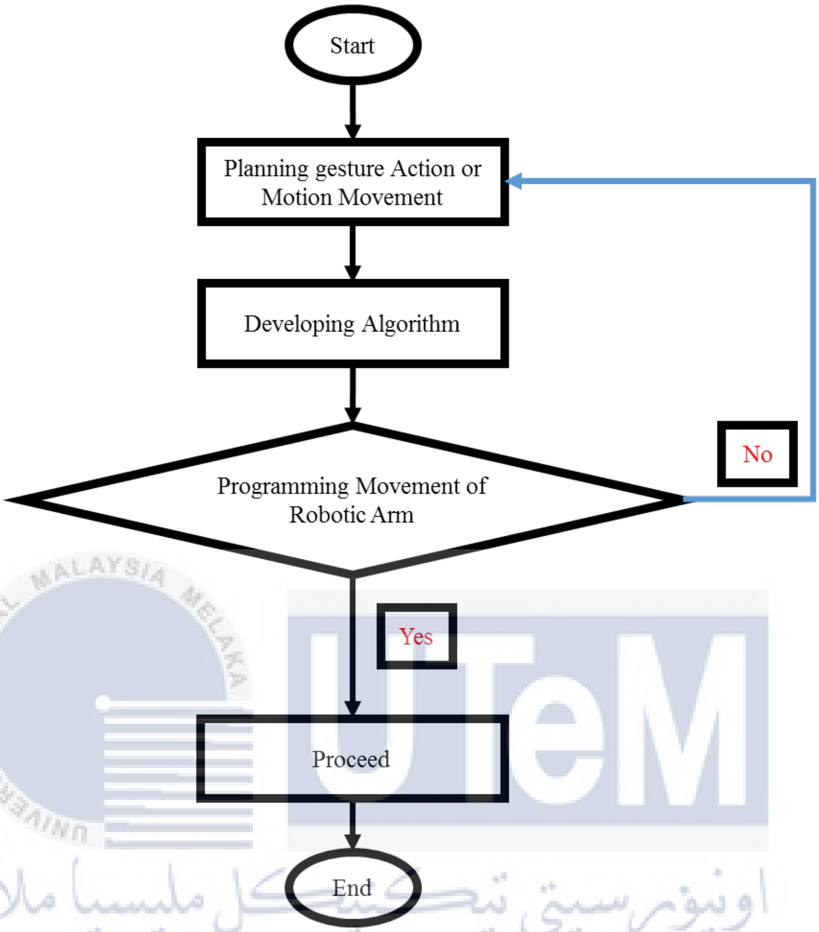


Figure 3.4: Flow Chart for Development of Project Software.

Development of Project Software	
Activity 7	Develop Algorithm and Robot Arm motion.
Activity 8	Program coding for the movement of the Robot Arm using Hand Glove.

Table 3.3: Development of Project Software for Methodology Flow Chart.

3.4 Test & Evaluation

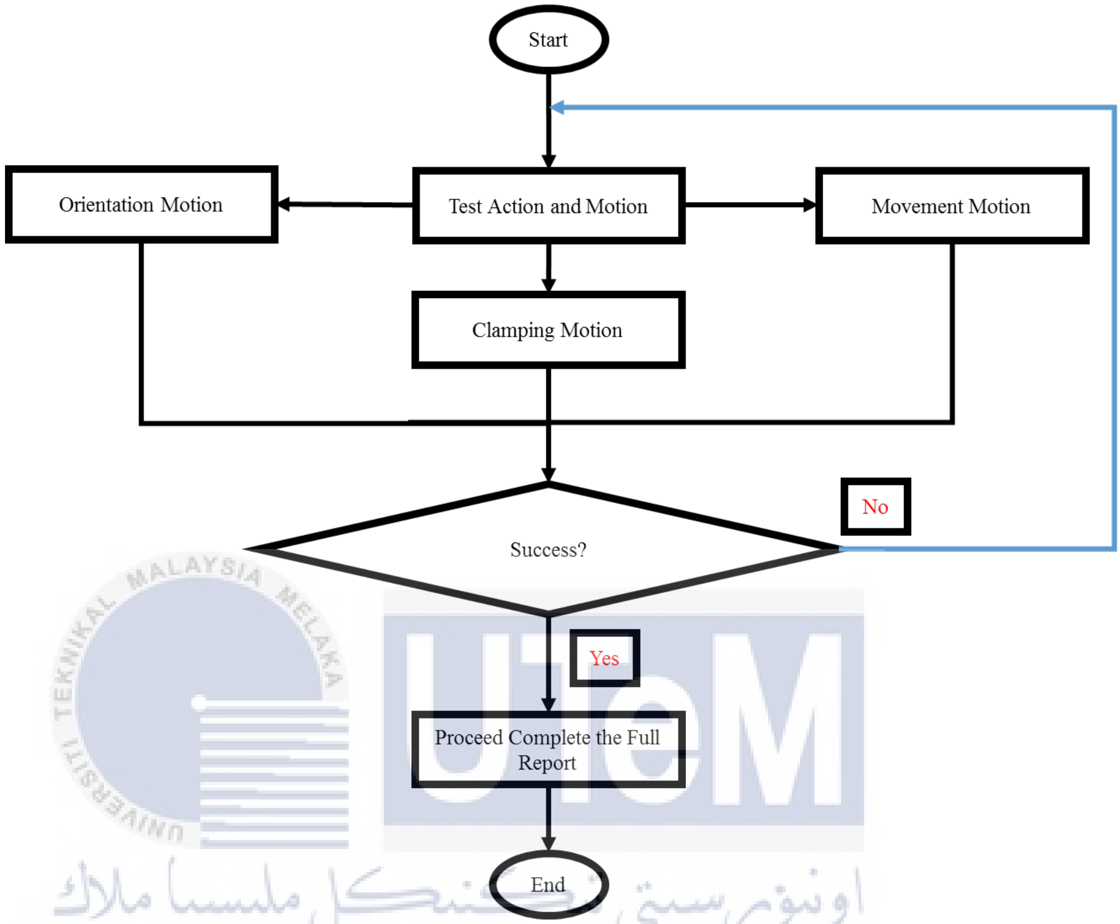


Figure 3.5: Flow Chart for Test & Evaluation.

Test & Evaluation	
Activity 9	Testing and Analysis the performance of the Hand Glove Control Device.
Activity 10	Complete the full report.

Table 3.4: Test & Evaluation for Methodology Flow Chart.

3.5 Method for Development of Hand Gesture Glove to Control Robot Arm

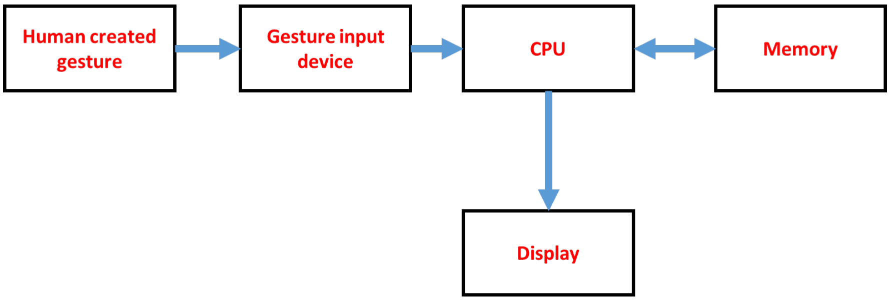


Figure 3.6: Hand Gesture Glove to Control Robot Arm (wired interface).

3.6 Robotic Glove

Robotic glove is develop to control the Robotic Arm. It consists of Arduino CPU as a microcontroller to control and manage the input signal reading from sensor provided. The analog signal provided from sensors that will convert to high frequency signal and communicate with the Robotic Arm via Bluetooth device. The accelerometer and flex sensor installed and connected with microcontroller is to represent as a movement of human gesture recognition. Because the accelerometer can capture the angle of X, Y, Z axis and also the acceleration of the 3 axis movement directions. The flex sensor can be control the gripper of the Robotic Arm. That is because the flex sensor can be bending and provided the variation of resistance to represent the human finger gesture to do work as a pick and place the objects from initial location to other location.

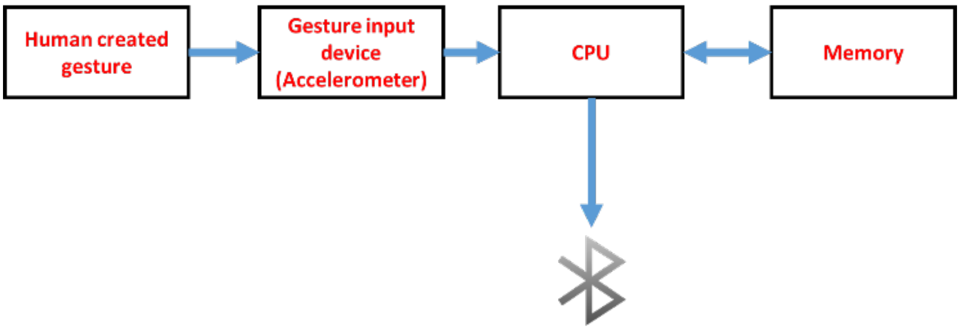


Figure 3.7: Robotic Glove (wireless interface – Bluetooth).

3.7 Robotic Arm

Lynxmotion AL5A are using to be control by wireless hand gesture glove.

Figure 3.8: Robotic Arm (Lynxmotion AL5A).

CHAPTER 4

PROJECT DEVELOPMENT

This chapter is to cover overall process in developing a project prototype of a Hand Gesture Glove. The design process of this project had been included into 6 categories, which are Robot Arm, Electrical design part, Programming design part, Inverse Kinematics algorithm part, Analysis data input accelerometer, and Development of hand gesture glove.

4.0 Introduction Lynxmotion AL5A Robot Arm

Figure 4.1 this is a robotic arm can provide fast, accurate, and repeatable movement but not include wireless communication system. The feature for this robot included optional wrist rotate, a functional gripper, wrist motion, elbow, single plane shoulder and last is supported overall part of robotic arm by base rotation.

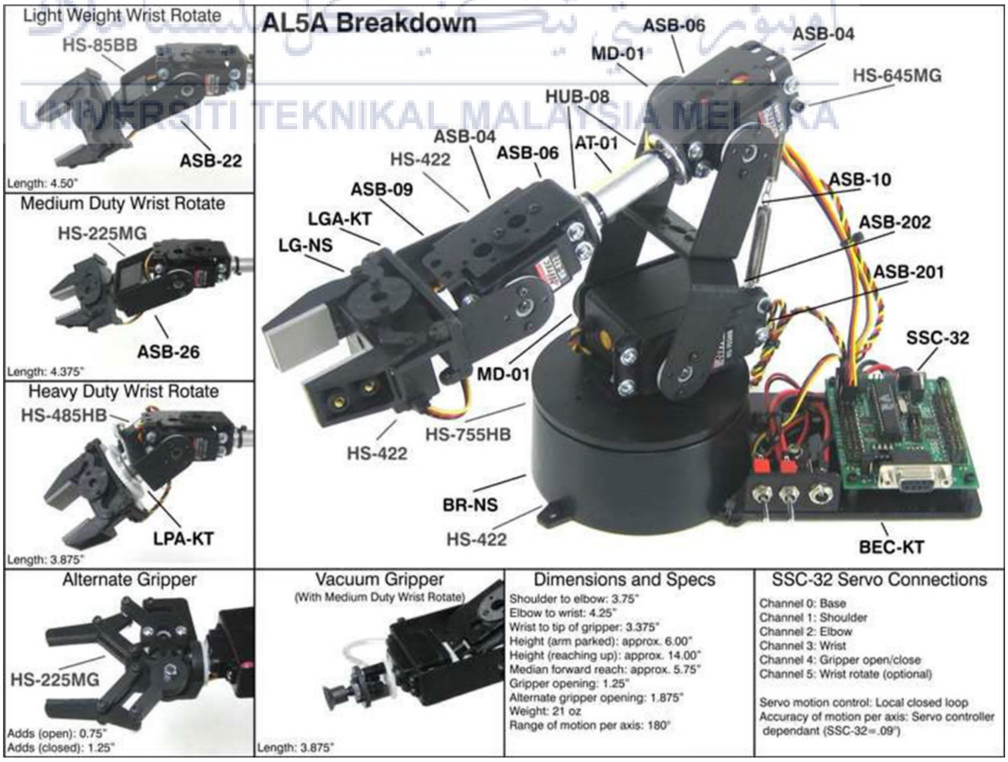


Figure 4.1: Part of Robotic Arm (Lynxmotion AL5A).

4.1 Electrical Design Part

Electronic and electrical parts is a main internal system for this project. The importance for this electrical design part is to control or communicate in between human and robot. The electrical design for this project included few main component, which is for overall project is powered by Arduino, Accelerometer and the Bluetooth communication system. The overall electronic circuit is divided into both part, which is Robot Arm control system and Hand Gesture control system. This electronic circuit is designed by using Fritzing software.

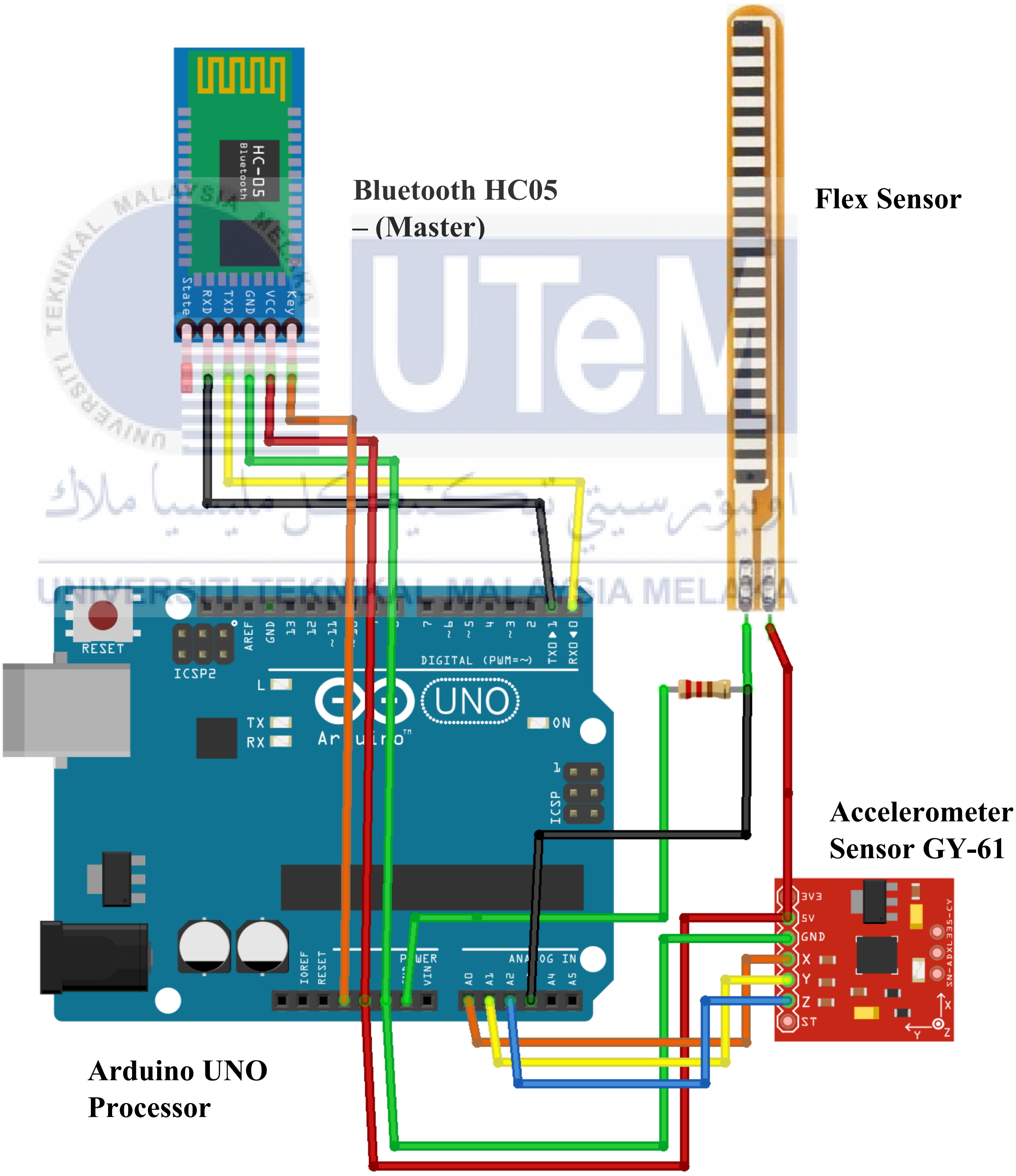


Figure 4.2: The Electronic Circuit for Hand Gesture Control System.

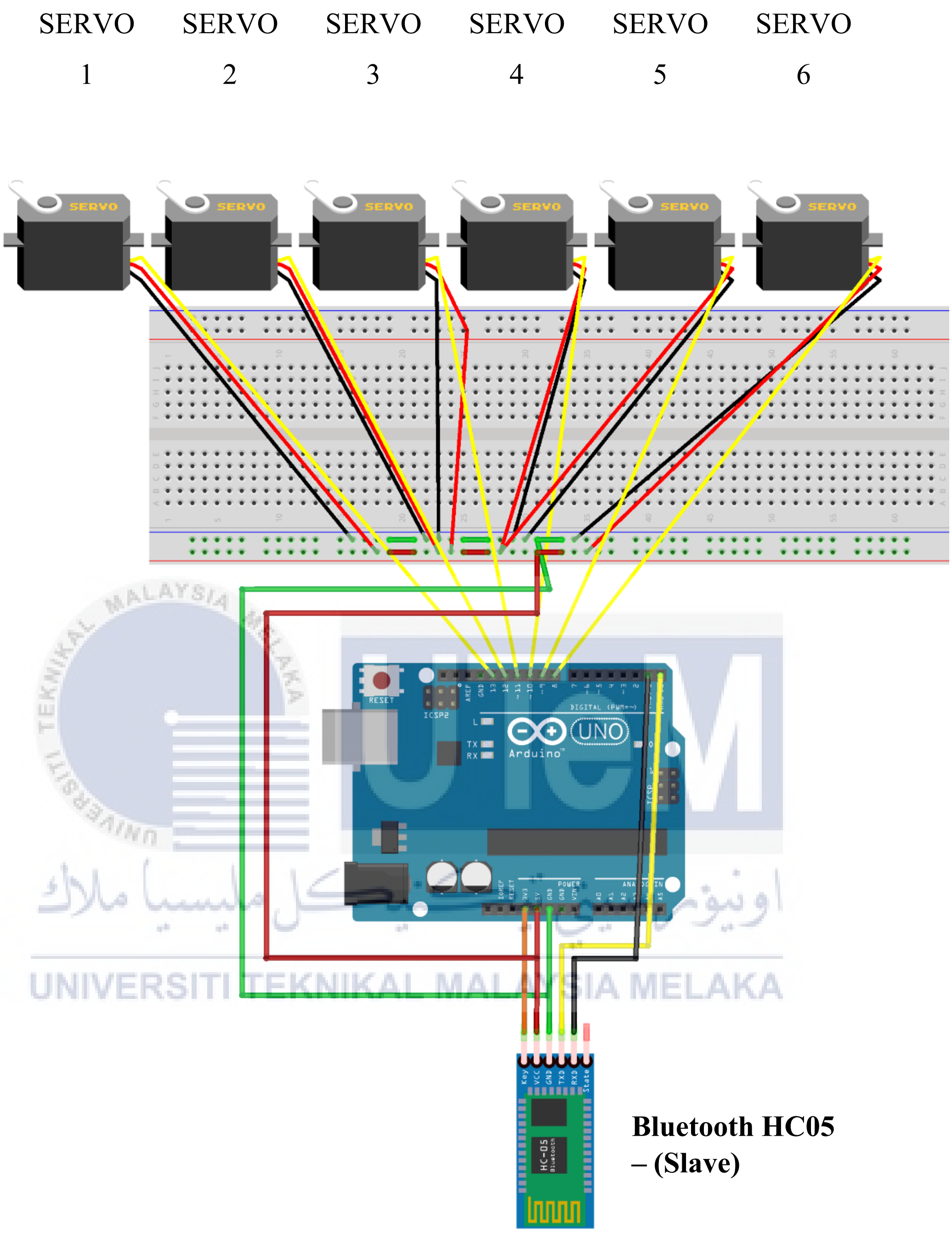


Figure 4.3: The Electronic Robot Arm Control Circuit.

4.1.1 Arduino Uno Processor

Arduino is an open source prototyping platform based on easy to use for building electronics projects such as hardware or software. This Arduino make life simple and accessible user experience. Arduino is simplifying the process of working with microcontrollers but it offers some advantages over other systems. Simple and clear programming environment, free library easy to getting and providing some getting started guide and also some great variety of tutorial on Arduino Project Hub.

SPECIFICATIONS	DESCRIPTIONS
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14(of which 6 provide PWM)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	32kb (ATmage328) of which 0.5KB used by bootloader
SRAM	2KB (ATmega328)
EEPROM	1KB (ATmega328)
Clock Speed	16MHz

Table 4.1: The Specifications of the Arduino UNO.

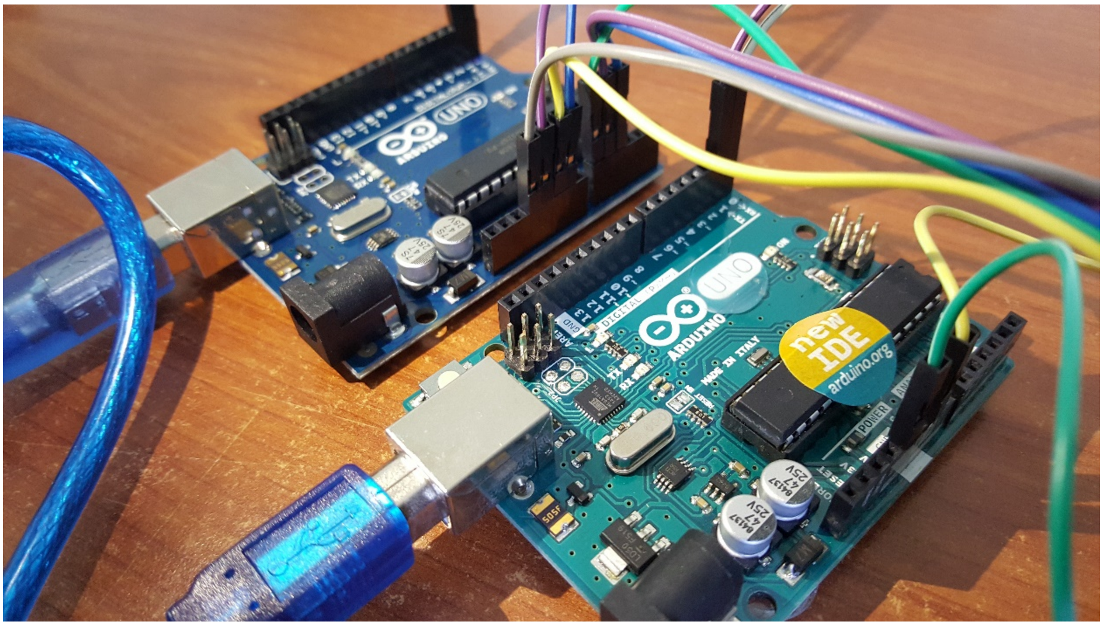


Figure 4.4: Arduino UNO Board Processor.

4.1.2 Accelerometer Sensor

The Accelerometer Sensor is used in this control system, that is to capture or snap the human gesture movement or orientation. To convert and represent that human gestures in to data signal from for the control system process. The features in this sensor is to control 3 axis sensing, small and low profile package, low power, single supply operation, and excellent temperature stability. The applications for accelerometer sensor is cost sensitive, low power, motion and tilt-sensing applications, mobile devices, gaming systems, disk and protection, image stabilization and last is sport and health devices.

SPECIFICATIONS	DESCRIPTIONS
Operating Voltage Range	1.8V to 5V
Supply Current	350uA
Interfaces	Analog
Operating Temperature	-40ºC to +85ºC
Dimension	20.3mm x 15.7mm x 11.6mm

Table 4.2: The Specifications of the Accelerometer Sensor.

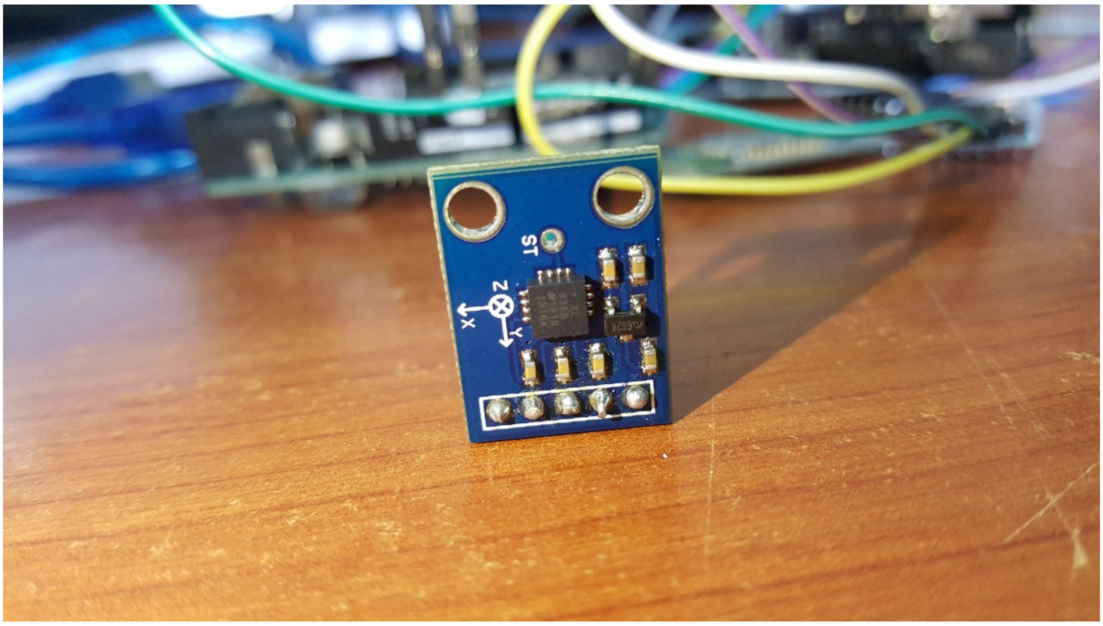


Figure 4.5: Accelerometer Sensor GY-61.

4.1.3 Flex Sensor

This Flex Sensor is very useful to control system. The Flex Sensor provided some features for us. The Flex sensor can provide angle displacement measurement, the bends and flexes physically with motion device. Last, this sensor possible uses for control Robotics, Gaming (Virtual Motion), Medical Devices, Musical Instruments, Physical Therapy, Simple Construction, and Low Profile systems. The function of resistor is to limit the current flow through the sensor used. That only flex sensor is using resistor for voltage divider.

SPECIFICATIONS	DESCRIPTIONS
Mechanical Specifications	
Life Cycle	>1 million
Height	≤ 0.43mm (0.017”)
Temperature Range	-35°C to +80°C
Electrical Specifications	
Flat Resistance	25K Ohms
Resistance Tolerance	± 30%
Bend Resistance Range	45K to 125K Ohms (depending on bend radius)
Power Rating	0.50 Watts continuous. 1 Watt Peak

Table 4.3: The Specifications of the Flex Sensor.

Figure 4.6: Flex Sensor.

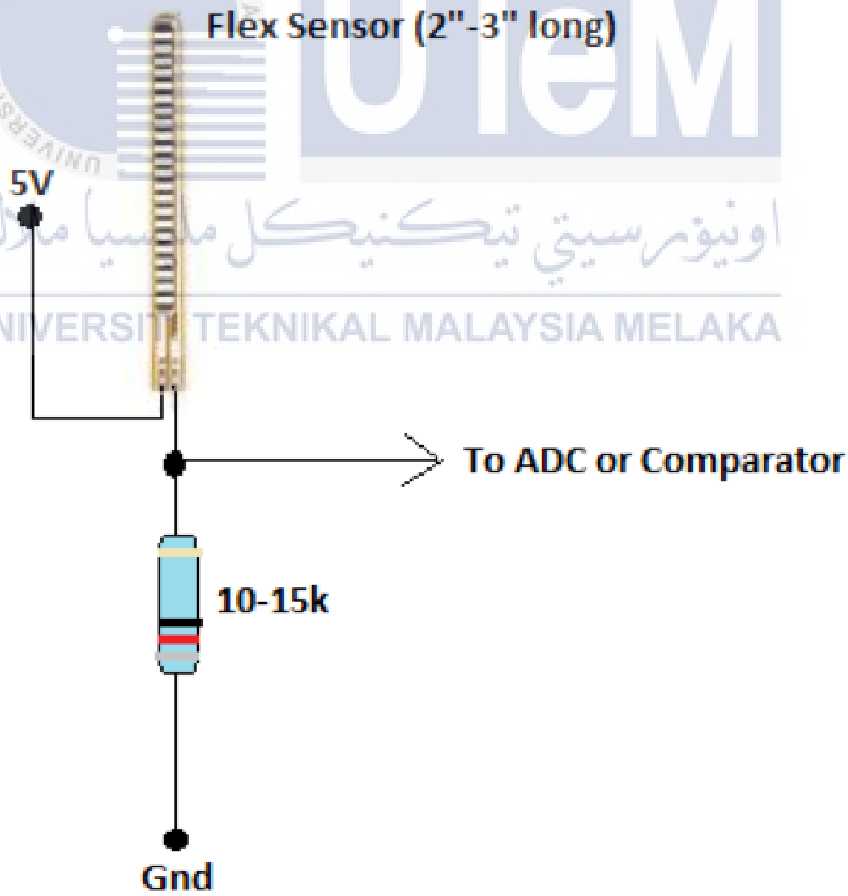


Figure 4.7: Schematic Connection Between Resistor and Flex Sensor.

4.1.4 Bluetooth Communication System

Bluetooth is the one type of communication system for transferring data form other device to another device. The module used in this control system is a HC-05, because this type of module can provide both communications such as transmit and receive data.

SPECIFICATIONS	DESCRIPTIONS
Voltage Supply	5V
Current Supply	50mA
ISM Frequency	2.4GHz
Asynchronous Speed	160Kbps – 2.1Mbps
Synchronous Speed	1Mbps
Dimension	26.9 mm x 13mm x 2.2mm

Table 4.4: The Specifications of the Bluetooth HC-05.

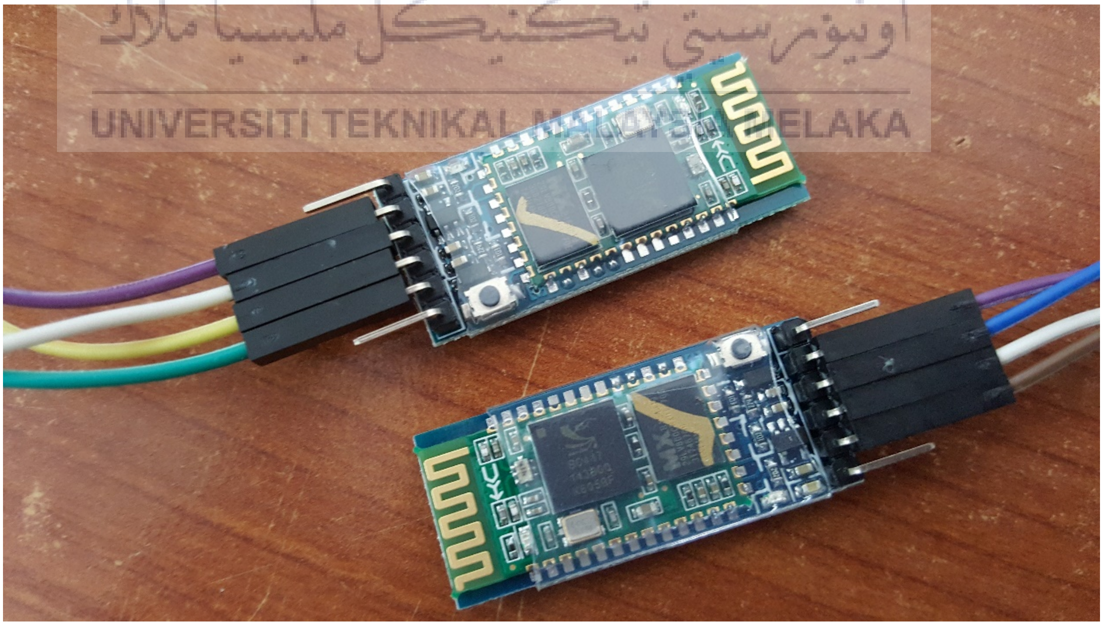


Figure 4.8: Bluetooth HC-05.

4.1.5 Wire Connector

This wire connector is used to connect in between Arduino and Sensor. Following this project, the wire connector is used to connect Bluetooth device, Accelerometer and Flex sensor to Arduino Board Processor.

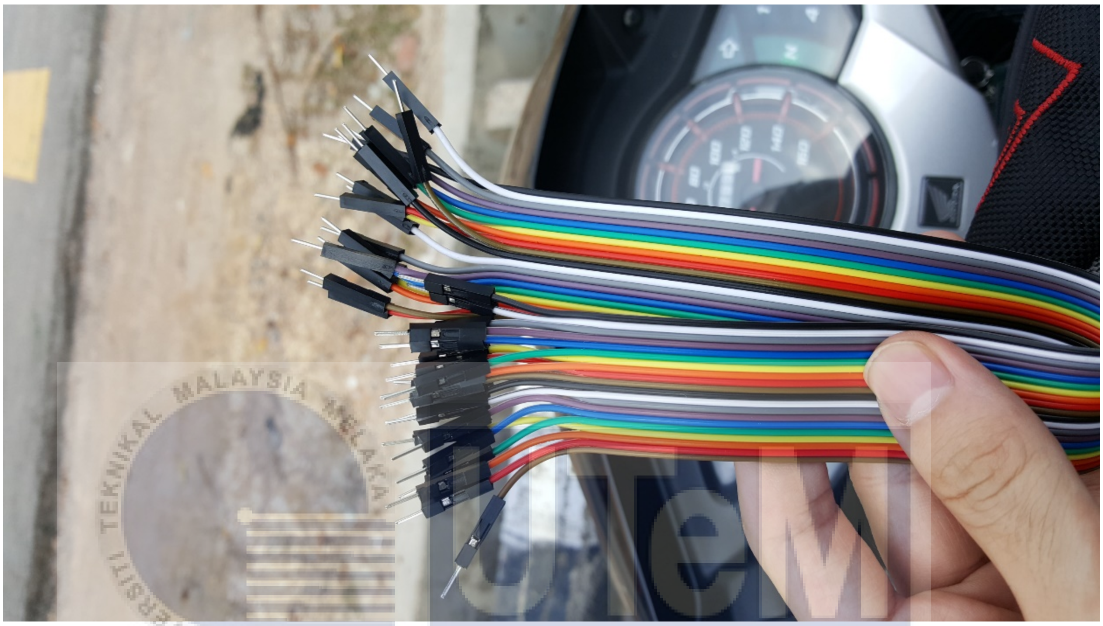


Figure 4.9: Wire Jumper Connector.

4.1.6 Power Supply Regulator 5V

This power supply regulator is use to supply voltage to servo motor. This electronic board providing stable voltage supply to prevent over flow voltage to broken each servo motor.

SPECIFICATIONS	DESCRIPTIONS
Locking ON/OFF Switch	Yes
LED Power Indicator	Yes
Input Voltage	6.5 V to 12V (DC)
Output Voltage	3.3 V / 5V
Maximum Output Current	700mA

Power Rails	0V, 3.3V, 5V on Breadboard
Two Groups of Header Pins	Yes
Size	5.3cm x 3.5cm

Table 4.5: The Specifications of the Power Supply Regulator 5V.

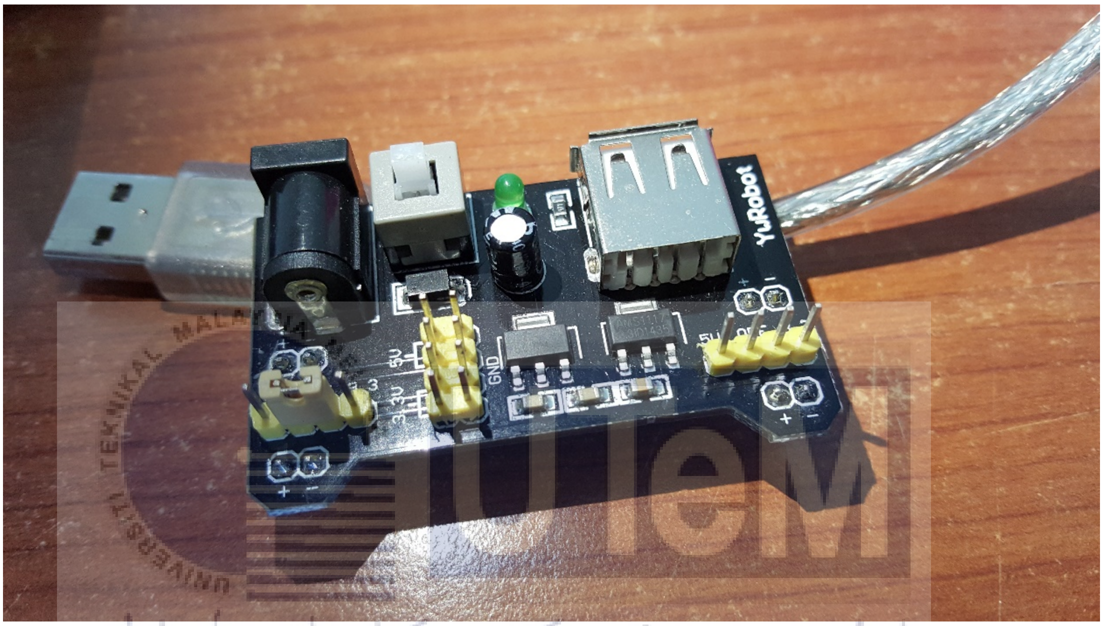


Figure 4.10: Power Supply 5V regulator.

4.1.7 Power Bank 5V

This power bank 5V is use to supply human gesture glove controller. This human gesture glove controller need 5V voltage to active the overall control system.

SPECIFICATIONS	DESCRIPTIONS
Input Voltage	5V (DC) – 1.0A
Output1 Voltage	5V (DC) – 2.0A
Output2 Voltage	5V (DC) – 2.0A
Power Storage	18000mAh

Table 4.6: The Specifications of the Power Bank 5V.

Figure 4.11: Power Bank 5V 18000mAh.

4.2 Programming Design Part

Before, to develop some robotics system, should learn how to program the programming using Arduino IDE. Programming design must follow the structure of the Arduino programming language. The basic structure of the Arduino programming language is fairly simple and runs in at least two parts. The following Table 4.7 is the basic example programming design is used to program hand gesture glove controller to control robotic arm.

Basic Example Programming Design	

Structure	`Void setup()`
	`{`
	`  Statements;`
	`}`
	`Void loop()`
	`{`
	`  Statements;`
	`}`

setup()	`Void setup()`

	{
	 pinMode(pin, OUTPUT);
	}

loop()	Void loop()
	{
	 digitalWrite(pin, HIGH);
	 delay(1000);
	 digitalWrite(pin, LOW);
	 delay(1000);
	}

functions	type functionName (parameter)
	{
	 Statements;
	}

{} curly braces	type function()
	{
	 Statements;
	}

; semicolon	int g = 0;
/*……*/ block comments	/* hello hello */
// line comments	// hello
variables	int g = 0; g = analogRead(2);
Variable declaration	int g = 0;
int	int g = 150;
float	float g = 150;
double	double g = 150;

arithmetic	y = y + 1;
	x = x - 2;
	g = z * 3;
	r = r / 4;

<table>
<tr><td>Compound assignment</td><td>g++ // g = g + 1
g-- // g = g - 1
g+=y // g = g + y
g-=y // g = g - y
g*=y // g = g * y
g/=y // g = g / y</td></tr>
<tr><td>Comparison operators</td><td>g == y // g is equal to y
g != y // g is not equal to y
g < y // g is less than y
g > y // g is greater than y
g <= y // g is less than or equal to y
g >= y // g is greater than or equal to y</td></tr>
<tr><td>Logical operators</td><td>//Logical And
if (g>0 && g<4)
//Logical Or
if (g>0 || y>0)
//Logical NOT
if (!x>0)</td></tr>
<tr><td>If</td><td>if (someVariable ?? value)
{
 Dosomething;
}</td></tr>
<tr><td>If ….else</td><td>if(inputPin == HIGH)
{
 dothingA;
{
else
{
 dothingB;
}</td></tr>
<tr><td>for</td><td>for(initialization; condition; expression)
{
Dosomething;
}</td></tr>
</table>

anologRead(pin)	`Value = analogRead(pin);`
delay(ms)	`delay(1000);`
millis()	`Value = millis();`

Serial.begin()	`Void setup()`
	`{`
	` Serial.begin(9600);`
	`}`

Serial.println()	`Serial.println(analogValue);`

Table 4.7: Example Programming Design Part.

4.3 Inverse and Forward Kinematics Algorithm Part

The Inverse Kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector. The Forward Kinematics is to get co-ordinate of end-effector from given angles of all joints.

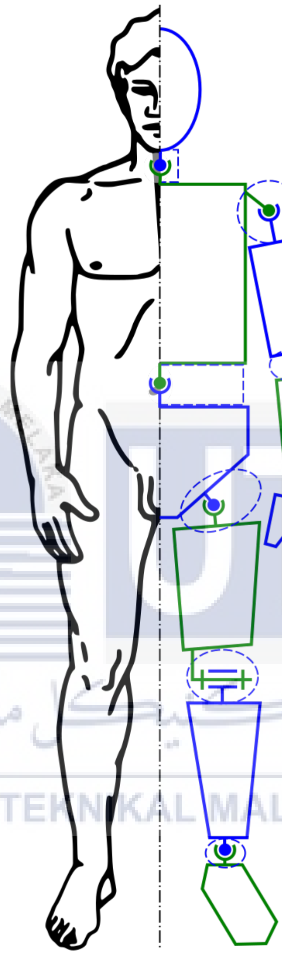


Figure 4.12: A model of the human skeleton as a kinematics chain allows positioning using inverse kinematics.

4.3.1 Inverse Kinematics Calculation for Robot Arm Control System

Direct measure the base height, length of humerus and ulna of the robot arm. This measurement length and height of robot arm is an importance part in this inverse kinematics calculation. To determine the angle movement for each joints of robot arm is using, when Base height 6cm, Humerus 9.5cm, Ulna 10.5cm, and Hand 8.5cm.

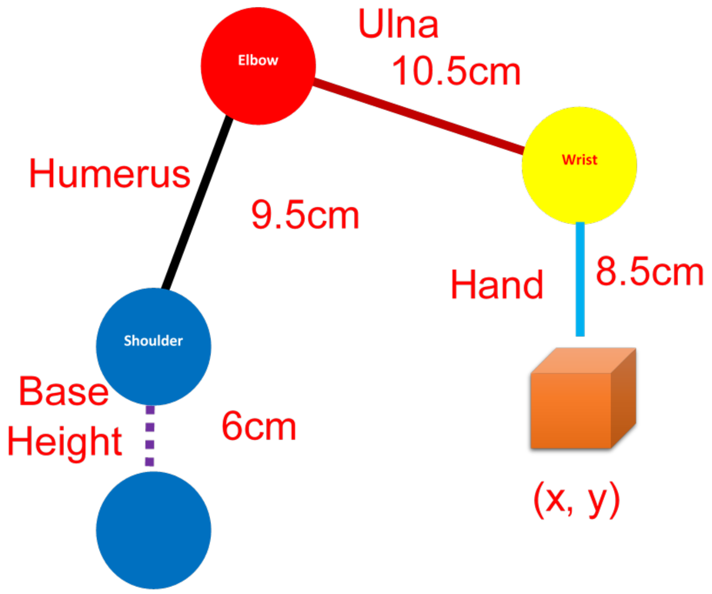


Figure 4.13: Robot Arm with Height and Length Measurement.

Solution

Convert the Hand -90 degrees to radians.

$$\text{Hand} = -90^{\circ}$$

$$\text{Hand} = -1.570796327 \text{ rad} \quad (4.1)$$

Calculate the Offset when Hand set to -90 degrees

$$\text{Offset} = (\sin(-1.570796327)) * 8.5 \text{ cm}$$

$$\text{Offset} = (-1)*8.5$$

$$\text{Offset} = -8.50 \text{ cm} \quad (4.2)$$

Wrist is given the in this calculation, when (x, y) = (13.7492, 6.4271)

$$\text{Wrist } x = 13.7492 \text{ cm} \quad (4.3)$$

And

 (13.7492, 6.4271)

$$\text{Wrist } y = 6.4271 – (-8.50) – 6$$

$$\text{Wrist } y = 8.9271 \text{ cm} \quad (4.4)$$

Calculate the average *d1* for the point given (13.7492, 6.4271), where *d1* is the length of the first joint of the robot arm.

$$d1 = \sqrt{(wrist\ y^2) + (wrist\ x^2)}$$

$$d1 = \sqrt{(13.7492^2) + (8.9271^2)}$$

$$d1 = 16.39309657\ cm \tag{4.5}$$

Calculate the angle of the first joint A1

$$A1 = \text{Atan2}(\text{wrist } x, \text{ wrist } y)$$

$$A1 = \tan^{-1}(\text{wrist } y / \text{ wrist } x) \text{ rad}$$

$$A1 = \tan^{-1}(8.9271/13.7492) \text{ rad}$$

$$A1 = 0.575869896 \text{ rad} \tag{4.6}$$

Calculate the angle of the second joint A2

$$A2 = \cos^{-1}[(\text{humerus}^2 - \text{ulna}^2 + d1^2)/ 2(\text{humerus})(d1)]$$

$$A2 = \cos^{-1}[(9.5^2 - 10.5^2 + 16.39309657^2)/ 2(9.5)(16.39309657)]$$

$$A2 = 0.645859611 \text{ rad} \tag{4.7}$$

Finally, calculate the angle of the Shoulder, Elbow and Wrist in the robot arm

$$\text{Shoulder} = A1 + A2$$

$$\text{Shoulder} = 0.575869896 + 0.645859611$$

$$\text{Shoulder} = 1.221729507 \text{ rad}$$

$$\text{Shoulder} = 70.0^\circ \tag{4.8}$$

$$\text{Elbow} = \cos^{-1}[(\text{humerus}^2 + \text{ulna}^2 - d1^2)/ 2(\text{humerus})(\text{ulna})]$$

$$\text{Elbow} = \cos^{-1}[(9.5^2 + 10.5^2 - 16.39309657^2)/ 2(9.5)(10.5)]$$

$$\text{Elbow} = 1.919865359 \text{ rad}$$

$$\text{Elbow} = -(180^\circ - 110^\circ)$$

$$\text{Elbow} = -70.0^\circ \tag{4.9}$$

$$\text{Wrist} = -90^\circ - (-70.0^\circ) - (70^\circ)$$

$$\text{Wrist} = -90^\circ \tag{4.10}$$

4.3.2 Forward Kinematics Calculation for Robot Arm Control System

Direct measure the base height, length of humerus and ulna of the robot arm. This measurement length and height of robot arm is an importance part in this inverse kinematics calculation. To determine the distance movement of the overall part of robot arm is using, when Base height 6cm, Humerus 9.5cm, Ulna 10.5cm, and Hand 8.5cm. And the angle for each joints of the robot arm is given in degrees.

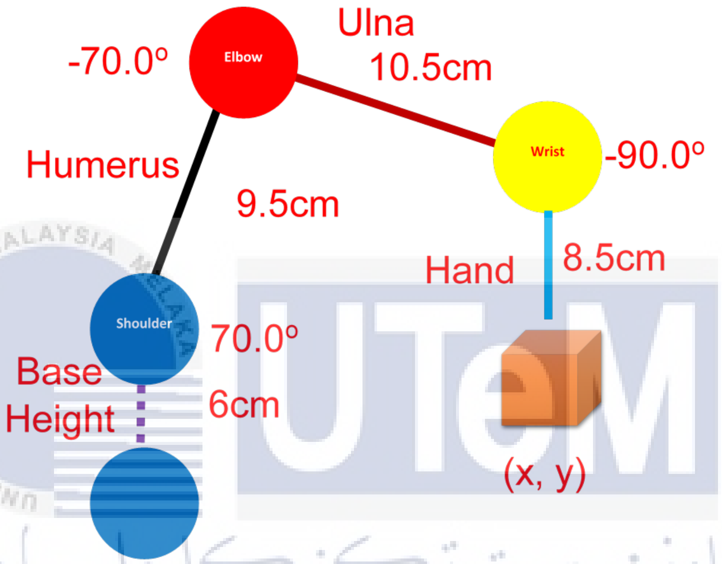


Figure 4.14: Robot Arm with Angle for each Joint, Height and Length Measurement.

Solution

Convert angle Shoulder, Elbow, and Wrist degrees given to radians. Calculate the sin and cos value for each joints.

Shoulder	70.0°	1.221729507 rad
Sin		Cos
Sin 1.221729507 = 0.939692289 rad		Cos 1.221729507 = 0.342021054 rad

Table 4.8: Convert Degrees to Radians and Calculate sin and cos value for Shoulder Robot arm.

Elbow	-70.0°	-1.221730476 rad
Sin	Cos	
Sin (-1.221730476 + 1.221729507) = - 0.000000969	Cos (-1.221730476 + 1.221729507) = 1	

Table 4.9: Convert Degrees to Radians and Calculate sin and cos value for Elbow Robot arm.

Wrist	-90.0°	-1.570796327 rad
Sin	Cos	
Sin (-1.221730476 + 1.221729507-1.570796327) = -1	Cos (-1.221730476 + 1.221729507-1.570796327) = -0.000000969	

Table 4.10: Convert Degrees to Radians and Calculate sin and cos value for Wrist Robot arm.

Calculate the y value for each joint in robot arm given in Figure 4.15

Y, humerus = (Sin 1.221729507) * (humerus)

Y, humerus = (0.939692289) * (9.5cm)

Y, humerus = 8.927076746 cm (4.11)

Y, base height = 6 cm (4.12)

Y, ulna = (Sin (-1.221730476 + 1.221729507)) * (arm))

Y, ulna = (- 0.000000969) * (10.5)

Y, ulna = -0 cm (4.13)

Y, tip = (Sin (-1.221730476 + 1.221729507-1.570796327))*(tip)

$$Y, tip = (-1) * (8.5)$$

$$Y, tip = -8.5 \text{ cm} \tag{4.14}$$

Combination or Summation for all y value for each joint in robot arm

$$Y = 8.927076746 \text{ cm} + 6 \text{ cm} + (-0 \text{ cm}) + (-8.5 \text{ cm})$$

$$Y = 6.427076745 \text{ cm} \tag{4.15}$$

Calculate the x value for each joint of robot arm given in Figure 4.15

$$X, humerus = (\text{Cos } 1.221729507) * (humerus)$$

$$X, humerus = (0.342021054) * (9.5cm)$$

$$X, humerus = 3.249200013 \text{ cm} \tag{4.16}$$

$$X, base\ height = 0 \text{ cm} \tag{4.17}$$

$$X, ulna = (\text{Cos } (-1.221730476 + 1.221729507)) * (arm))$$

$$X, ulna = (1) * (10.5cm)$$

$$X, ulna = 10.5 \text{ cm} \tag{4.18}$$

$$X, tip = (\text{Cos } (-1.221730476 + 1.221729507-1.570796327))*(tip)$$

$$X, tip = (-0.000000969) * (8.5cm)$$

$$X, tip = -0.000008236 \text{ cm} \tag{4.19}$$

Combination or Summation for all x value for each joint in robot arm

$$X =3.249200013 \text{ cm} + 0 \text{ cm} + 10.5 \text{ cm} + (-0.000008236 \text{ cm})$$

$$X = 13.74919178 \text{ cm} \tag{4.20}$$

Finally, this calculation gets back the coordinate point location for robot arm reach

Figure 4.15: Result Calculation for Forward Kinematics Algorithm

4.4 Analysis Data Input Accelerometer Using Matlab

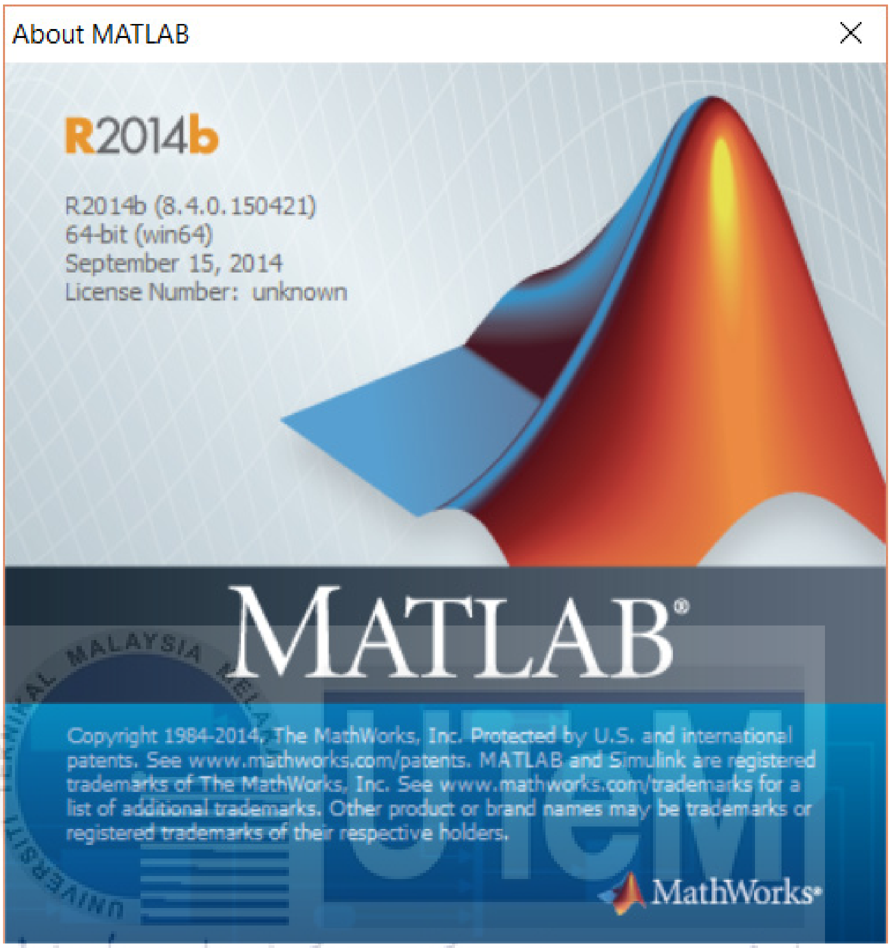


Figure 4.16: Matlab R2014b 8.4.0.150421 Version Support.

4.4.1 Arduino and Matlab Setup

Before accelerometer data analysis, should be follow a few steps to setup in between Arduino hardware and your personal computer to setup serial communication through Matlab. To connect the Arduino broad to your personal computer via USB, program the Arduino microcontroller using the Arduino IDE, and identify the COM port, enter it into Matlab program and save and rename the file such as “setupSerial”. After created the file and run the function file “setupSerial.m”. The following below is the example for Arduino IDE program and Matlab “setupSerial.m” setup.

Arduino IDE program setup

```
// These constant won't change. They're used to give names
to the pins used:
const int analogPinA0 = A0; // Analog Input attached to: x;
const int analogPinA1 = A1; // Analog Input attached to: y;
const int analogPinA2 = A2; // Analog Input attached to: z;

//initialize varialles
int mode = -1;
unsigned int sensorValue = 0;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin (9600);
  // check serial communication- acknowledgement routine.
  Serial.println ('a'); // sending a charachter to pc
  char a = 'b';
  while (a != 'a')
  {
    //wait for a specific charachter form the PC
    a = Serial.read();
  }
}

void loop() {
  if (Serial.available () > 0) // check is any data has been
sent by the PC
  {
    mode = Serial.read(); // check if there is a request for
accelerometer values
    if (mode == 'R') // used to set different modes for
various operations. R is used to read accelerometer values
    {
      //read analog values of pinA0(x) A1(y) A2(z) and send
then to the PC
      sensorValue = analogRead (analogPinA0);
      Serial.println (sensorValue);
      sensorValue = analogRead (analogPinA1);
      Serial.println (sensorValue);
      sensorValue = analogRead (analogPinA2);
      Serial.println (sensorValue);
    }
    // wait 20ms before the next loop for the analog to
digital converter to settle after the last reading
    delay (20);
  }
}
```

Figure 4.17: Sample for Arduino IDE Program Setup

Matlab program "setupSerial.m"

```
function [s,flag] = setupSerial(comPort)
% Initialize the serial port communication between Arduino
and Mathlab
% We ensure that arduino is also communicating with Matlab
at this
% time. A predefined code on the arduino acknowledge this.
% if setup is complete than the value of setup is returned
as 1 else 0
flag = 1;

s = serial(comPort);
set(s,'DataBits',8);
set(s,'StopBit',1);
set(s,'BaudRate',9600);
set(s,'Parity','none');
fopen(s);
a='b';
while (a ~='a')
    a = fread (s, 1, 'uchar');
end
if (a == 'a')
    disp ('serial read');
end
fprintf (s, '%c', 'a');
mbox = msgbox ('Serial Communication Setup.');
uiwait(mbox);
fscanf (s, '%u');
end
```

Figure 4.18: Sample Matlab Program "setupSerial.m"

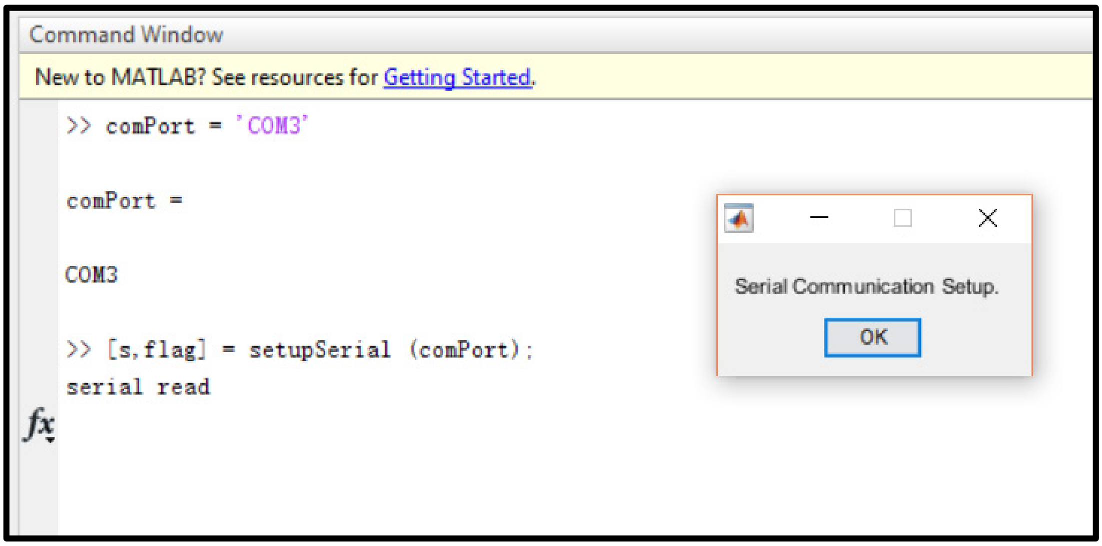


Figure 4.19: Matlab Command Window.

4.4.2 Connecting and Calibrating an Accelerometer Sensor

Calibration as an important part to getting the most accurate data analysis form accelerometer sensor. The calibration is to correlate the sensor reading and also standard in order to check and confirm the sensor accuracy. Form this calibration, we can adjust and control the experimental results you need or to allow comparison with other data. Before the calibration data, we should be program "readAcc.m" is to read or pass data information form Arduino processor to Matlab program.

Matlab program "readAcc.m"

```
function [gx, gy, gz] = readAcc(out, calCo)

fprintf(out.s,'R');

%read value from accelerometer
readings(1)= fscanf(out.s,'%u');
readings(2)= fscanf(out.s,'%u');
readings(3)= fscanf(out.s,'%u');

%determine what offset and again values to use
offset = calCo.offset;
gain = calCo.g;
accel = (readings-offset)./gain;

%map analog input to axes
gx = accel(1);
gy = accel(2);
gz = accel(3);
end
```

Figure 4.20: Sample Matlab Program "readAcc.m"

Matlab program "calibrate.m"

```
function calCo = calibrate(s)
out.s=s;
calCo.offset=0;
calCo.g=1;
%read the raw accelerometer output at three different
orientations
%gz = 1, gx = gy = 0 orientation
mbox = msgbox ('Lay accelerometer on a flat surface.' ,
'Calibration'); uiwait(mbox);
[gx_z, gy_z, gz_z] = readAcc(out,calCo);
%gx = 1, gy = gz = 0 orientation
mbox = msgbox ('Stand accelerometer on edge so that X arrow
points up.','Calibration'); uiwait(mbox);
[gx_x, gy_x, gz_x] = readAcc(out,calCo);
%gy = 1, gx = gz = 0 orientation
mbox = msgbox ('Stand accelerometer on edge so that Y arrow
points up.','Calibration'); uiwait(mbox);
[gx_y, gy_y, gz_y] = readAcc(out,calCo);

%calculate offsets for each axis
offsetX = (gx_z + gx_y)/2;
offsetY = (gy_x + gy_z)/2;
offsetZ = (gz_x + gz_y)/2;

%calculate scaling factors
gainX = gx_x - offsetX;
gainY = gy_y - offsetY;
gainZ = gz_z - offsetZ;

calCo.offset = [offsetX offsetY offsetZ];
calCo.g = [gainX gainY gainZ];
mbox = msgbox ('Sensor calibration complete');
uiwait(mbox);
end
```

Figure 4.21: Sample Matlab Program "calibrate.m"

Matlab program "closeSerial.m"

```
clc
clear all
if ~isempty(instrfind)
    fclose(instrfind);
    delete(instrfind);
end
close all
clc
disp('Serial Port Closed')
```

Figure 4.22: Sample Matlab Program "closeSerial.m"

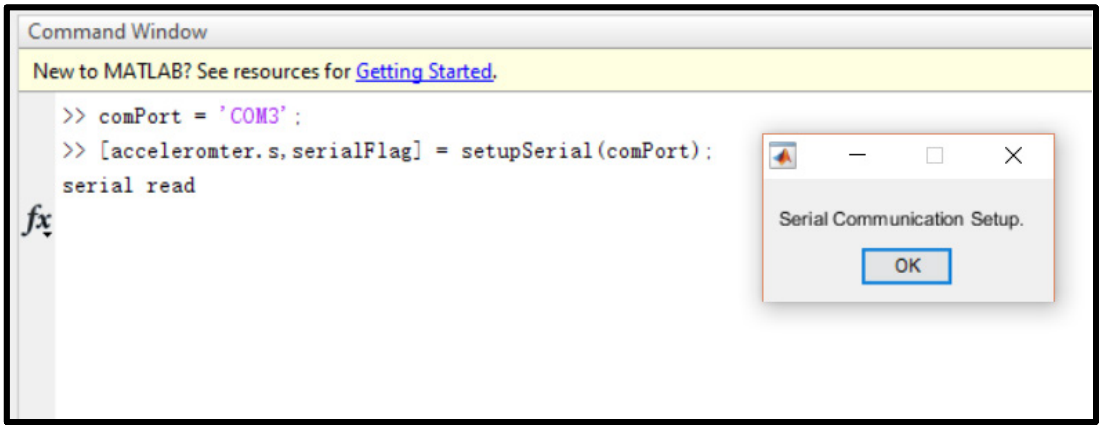


Figure 4.23: Matlab Command Window Serial Communication Setup “click ok”.

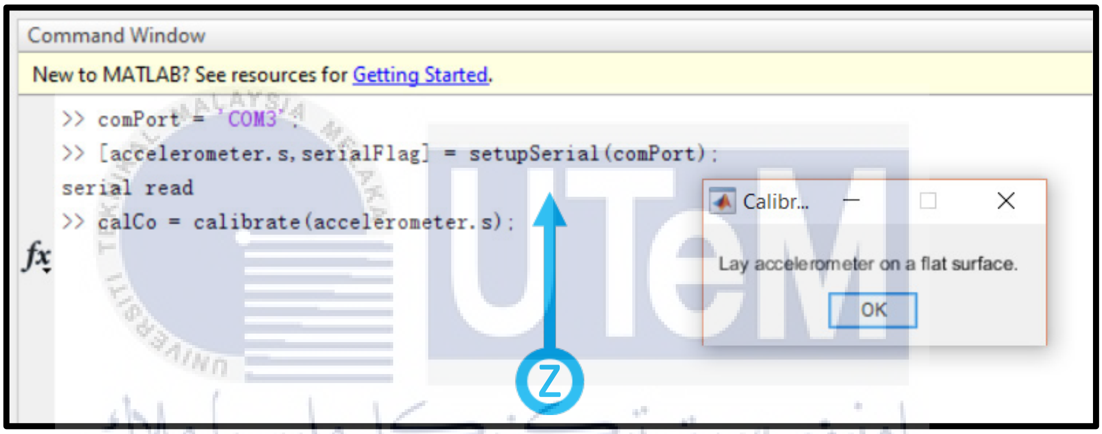


Figure 4.24: Matlab Command Window Lay accelerometer on a Flat surface “click ok”.

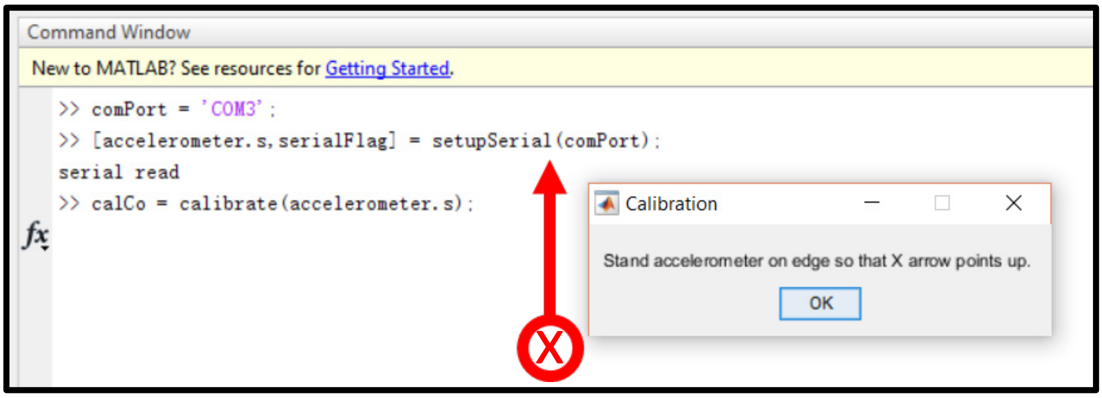


Figure 4.25: Matlab Command Window Stand accelerometer on edge so that X arrow point up “click ok”.

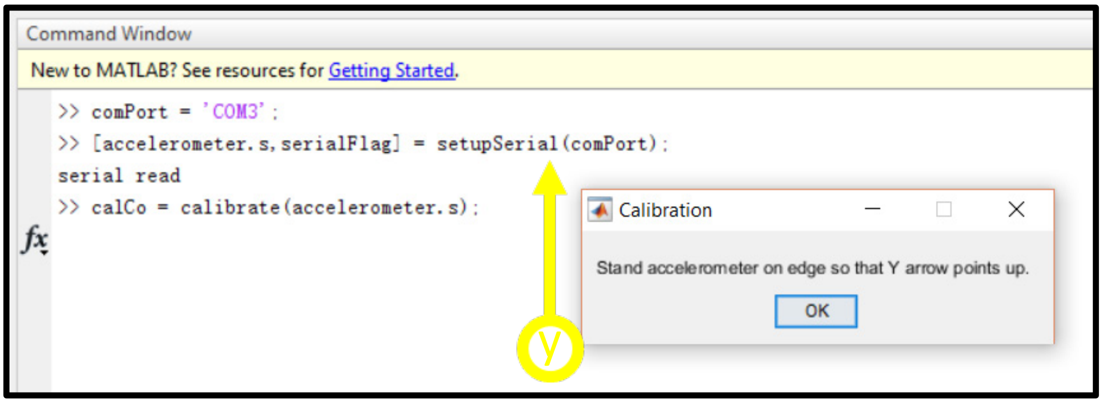


Figure 4.26: Matlab Command Window Stand accelerometer on edge so that Y arrow point up "click ok".

Figure 4.27: Matlab Command Window Sensor calibration complete "click ok".

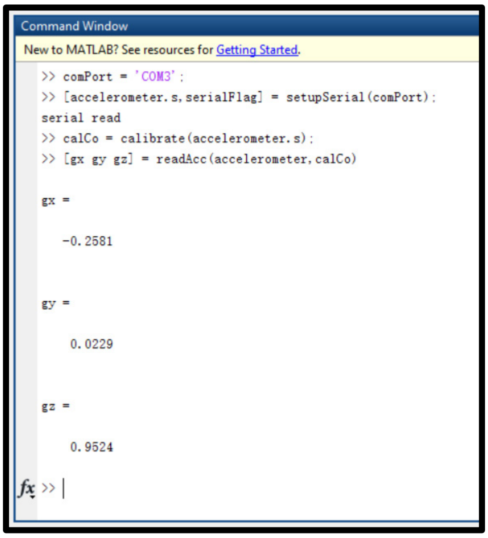


Figure 4.28: Matlab Command Window Showing Accelerometer Data Calibration.

4.4.3 Vector and Magnitude Visualization of Accelerometer Data

Some instruction for Matlab should be follow before undergo the vector and magnitude visualization of accelerometer data. Setup serial communication over the COM Port, calibrate the accelerometer by orienting it in 3 directions, read accelerometer data form the Arduino broad, display X, Y, Z vectors in real-time graph and repeat read accelerometer data and display X, Y, Z vector steps. After finish close the serial port.

Matlab program "vector.m"

```
%===========================================================================
%================Accelerometer Vector Skeleton Code=======================
%===========================================================================
%This skeleton script does the following:
%1. Specifies the COM port that the Arduino board is connected to
%2. Initializes the Serial Port - setupSerial()
%3. Runs a calibration routine if needed - calibration()
%4. Opens a new figure and customizes it by adding start/stop and close
%   serial buttons
%   - A different stop call
%5. Runs a loop that continually reads the accelerometer values readAcc()
%   The accelerometer data is placed in the variable [gx gy gz].
%6. Displays the vectors using line() command

%% 1. Specifies the COM port that the Arduino board is connected to
%comPoert = 'COM3';
% This can be found out using the device manager (Windows) on a mac type Is
% /dev /tty* in terminal and identify the device that is listed as usbmodem
% Example for a MAC comPort = '/dev/tty.usbmodemfa131';

comPort = 'COM3';

%% 2. Initialoze the Serial Port - setupSerial()

% connect MATLAB to the Accelerometer

if (~exist('serialFlag','var'))
    [accelerometer.s,serialFlag] = setupSerial(comPort);
end

%% 3. Run a calibration routine if needed - calibrate()

% if the accelerometer is not calibrated, calibrate now

if (~exist('calCo','var'))
    calCo = calibrate(accelerometer.s);
end

%% 4. Open a new figure - add start/stop and close serial buttons

% initialize the figure that we will plot in if it does not exist

if (~exist('h','var')|| ~ishandle(h))
    h = figure(1);
    ax = axes('box','on');
end
```


Figure 4.29: Sample Matlab Program "vector.m"

```
% add a start/stop and close serial button inside the figure
% Keep in mind the 'stop_call_wk3' function that this button calls
% everytime it is pressed

if (~exist('button','var'))
    button = uicontrol('Style','pushbutton','String','Stop',...
        'pos', [0 0 50 25], 'parent',h,...
        'Callback','stop_call_vector','UserData',1);
end

% Keep in mind the 'close_call' function that button calls everytime it is
% pressed

if (~exist('button2','var'))
    button2 = uicontrol('Style','pushbutton','String','Close Serial
Port',...
        'pos',[250 0 150 25],'parent',h,...
        'Callback','closeSerial','UserData',1);
end

%% 5. Runs a loop that continually reads the acceleromter values
% The accelerometer data is placed in the variables [gx gy gz].

while (get(button, 'UserData'))

    %read accelerometer output
    [gx, gy, gz] = readAcc(accelerometer, calCo);

    cla; %clear everything from the current axis

    %Resultance Acceleration
     grdataT = (sqrt(gx^2 + gy^2 + gz^2));

     %plot X accelerometer vector
    line([0 0 gx],[0 0 0],[0 0 0], 'Color', 'r', 'LineWidth', 2, 'Marker',
'o');
    %plot Y accelerometer vector
    line([0 0 0],[0 0 gy],[0 0 0], 'Color', 'y', 'LineWidth', 2, 'Marker',
'o');
    %plot Z accelerometer vector
    line([0 0 0],[0 0 0],[0 0 gz], 'Color', 'b', 'LineWidth', 2, 'Marker',
'o');

    %limit plot to +/- 1.25 g in all directions and make axis square
    limits = 2.5;
    axis([-limits limits -limits limits -limits limits]);
    axis square;
    grid on
    view(3)
    xlabel('X-axis acceleration')
    ylabel('Y-axis acceleration')
    zlabel('Z-axis acceleration')

    %calculate the angle of the resultant acceleration vector and print
    theta = atand(gy/gx);
    title(['Accelerometer tilt angle: ' num2str(theta,'%.OF')]);

    %force MATLAB to redraw the figure
    drawnow;

end
```

Figure 4.30: Sample Matlab Continue Program “vector.m”

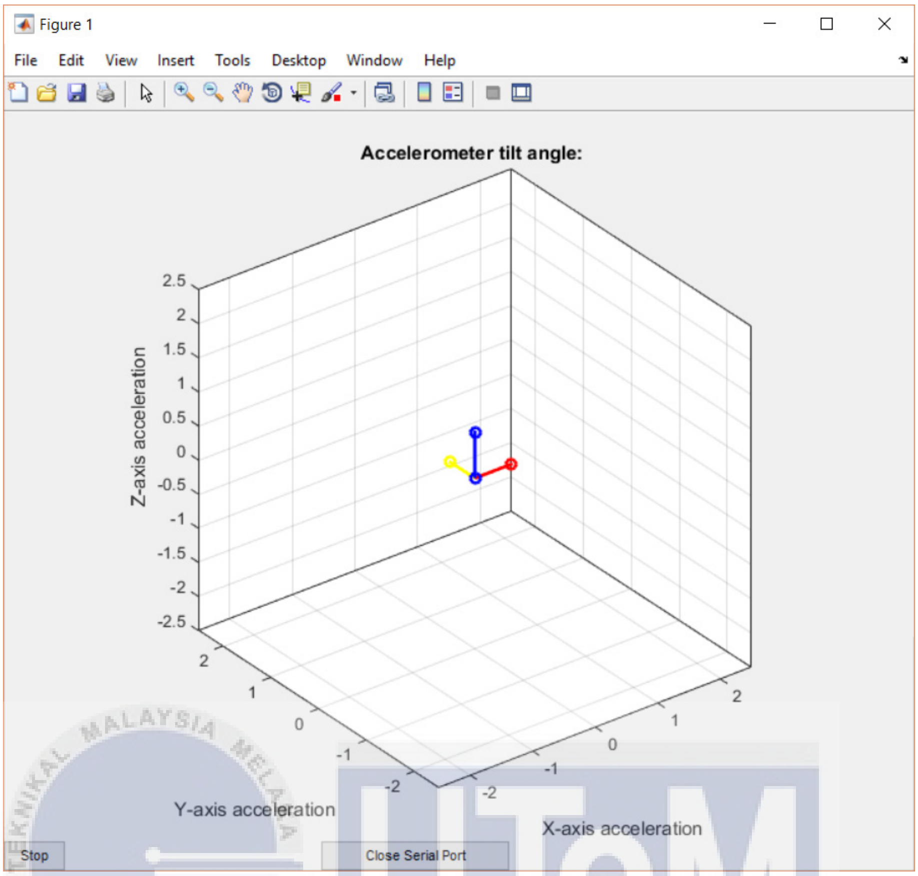


Figure 4.31: Vector Reading in 3 Colour Axis Acceleration Form Sensor.

Matlab program "magnitude.m"

```
%========================================================================
%================Accelerometer Magnitude Skeleton Code===================
%========================================================================
%This skeleton script does the following:
%1. Specifies the COM port that the Arduino board is connected to
%2. Initializes the Serial Port - setupSerial()
%3. Runs a calibration routine if needed - calibration()
%4. Opens a new figure and customizes it by adding start/stop and close
%   serial buttons
%   - A different stop call
%5. Runs a loop that continually reads the accelerometer values readAcc()
%   The accelerometer data is placed in the variable [gx gy gz].
%6. Displays the vectors using line() command

%% 1. Specifies the COM port that the Arduino board is connected to
%comPoert = 'COM3';
% This can be found out using the device manager (Windows) on a mac type Is
% /dev /tty* in terminal and identify the device that is listed as usbmodem
% Example for a MAC comPort = '/dev/tty.usbmodemfa131';

comPort = 'COM3';

%% 2. Initialoze the Serial Port - setupSerial()

% connect MATLAB to the Accelerometer

if (~exist('serialFlag','var'))
    [accelerometer.s,serialFlag] = setupSerial(comPort);
end

%% 3. Run a calibration routine if needed - calibrate()

% if the accelerometer is not calibrated, calibrate now

if (~exist('calCo','var'))
    calCo = calibrate(accelerometer.s);
end

%% 4. Open a new figure - add start/stop and close serial buttons

% initialize the figure that we will plot in if it does not exist

if (~exist('h','var')|| ~ishandle(h))
    h = figure(1);
    ax = axes('box','on');
end

% add a start/stop and close serial button inside the figure
% Keep in mind the 'stop_call_wk3' function that this button calls
% everytime it is pressed

if (~exist('button','var'))
    button = uicontrol('Style','pushbutton','String','Stop',...
        'pos', [0 0 50 25], 'parent',h,...
        'Callback','stop_call_vector','UserData',1);
end

% Keep in mind the 'close_call' function that button calls everytime it is
% pressed

if (~exist('button2','var'))
    button2 = uicontrol('Style','pushbutton','String','Close Serial
Port',...
        'pos',[250 0 150 25],'parent',h,...
        'Callback','closeSerial','UserData',1);
end
```

Figure 4.32: Sample Matlab Program "magnitube.m"

```
%% 5. Initializing the Rolling Plot

% The rolling plot works like the last code. Set it up in the same manner.

buf_len = 100;
index = 1:buf_len;

% create variables for the X axis
gxdata = zeros(buf_len,1);
% create variables for the Y axis
gydata = zeros(buf_len,1);
% create variables for the Z axis
gzdata = zeros(buf_len,1);


%% 6. Data Collection and Plotting

% While the figure window is open.
while(get(button,'UserData'))

    %Get the new values from the accelerometer
    [gx, gy, gz] = readAcc(accelerometer, calCo);

    % Update the rolling plot. Append the new reading to the end of the
    % rolling plot data. Drop the first value
    gxdata = [gxdata(2:end);gx];
    gydata = [gydata(2:end);gy];
    gzdata = [gzdata(2:end);gz];

    % Plot for 3 magnitude
    %plot(index, gxdata, 'r');
    %plot(index, gydata, 'y');
    %plot(index, gzdata, 'b');
    plot(index, gxdata, 'r', index, gydata, 'y', index, gzdata, 'b');
    axis([1 buf_len -3.5 3.5]);
    xlabel('time');
    ylabel('Magnitude of 3 axis acceleration');

    drawnow;

end
```


Figure 4.33: Sample Matlab Continue Program “magnitube.m”

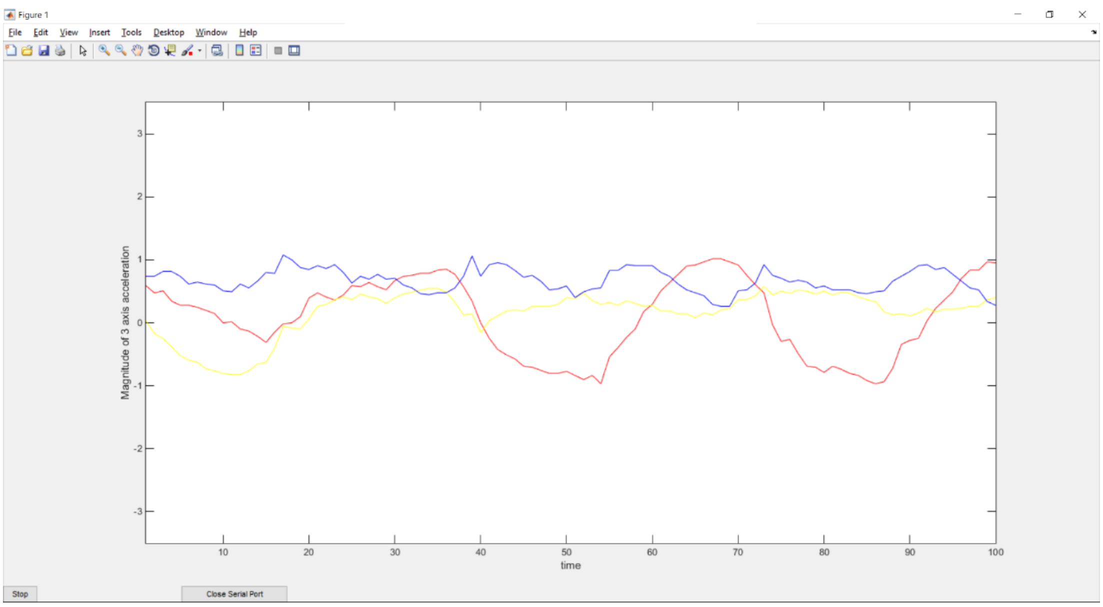


Figure 4.34: Magnitude Reading in 3 Colour Axis Acceleration Form Sensor.

4.4.4 Filtering Noise Out of Accelerometer Data

Filter noise is most of technical person is need to filtering out and get the perfect signal or data to controlling some technical work. Noise is a non-specific data form each signal it can be high and low, that will be an error for controlling any type for robotic system. So, in this case the statistical results in research the Low-pass Filter is more accurate and also more correspond with the other type of filter. The Exponential Moving Average (EMA) is discrete 1st- order feedback low pass filter. At any time, output of the filter is a weighted sum of new sensor value and the old filter output. Filter coefficient controls how much filtering is applied: -

$$\alpha \in [0,1] \tag{4.21}$$

$$gX_{filt} = (1- \alpha)*gX_{filt\text{-}old} + \alpha*gX_{raw} \tag{4.22}$$

and

$$\alpha = 1 \tag{4.23}$$

$$gX_{filt} = gX_{raw} \tag{4.24}$$

Matlab program "vectorFilter.m"

```
%% 5. Runs a loop that continually reads the acceleromter values
% The accelerometer data is placed in the variables [gx gy gz].

%initialize filter variable & coefficient
gxFilt = 0;
gyFilt = 0;
gzFilt = 0;
alpha = 0.5;

while (get(button, 'UserData'))

    %read accelerometer output
    [gx, gy, gz] = readAcc(accelerometer, calCo);

    %low pass filter accelerometer output
    gxFilt = (1 - alpha)*gxFilt + alpha*gx;
    gyFilt = (1 - alpha)*gyFilt + alpha*gy;
    gzFilt = (1 - alpha)*gzFilt + alpha*gz;


    %force MATLAB to redraw the figure
    drawnow;

end
```

Figure 4.35: Sample Matlab Program "vectorFilter.m"

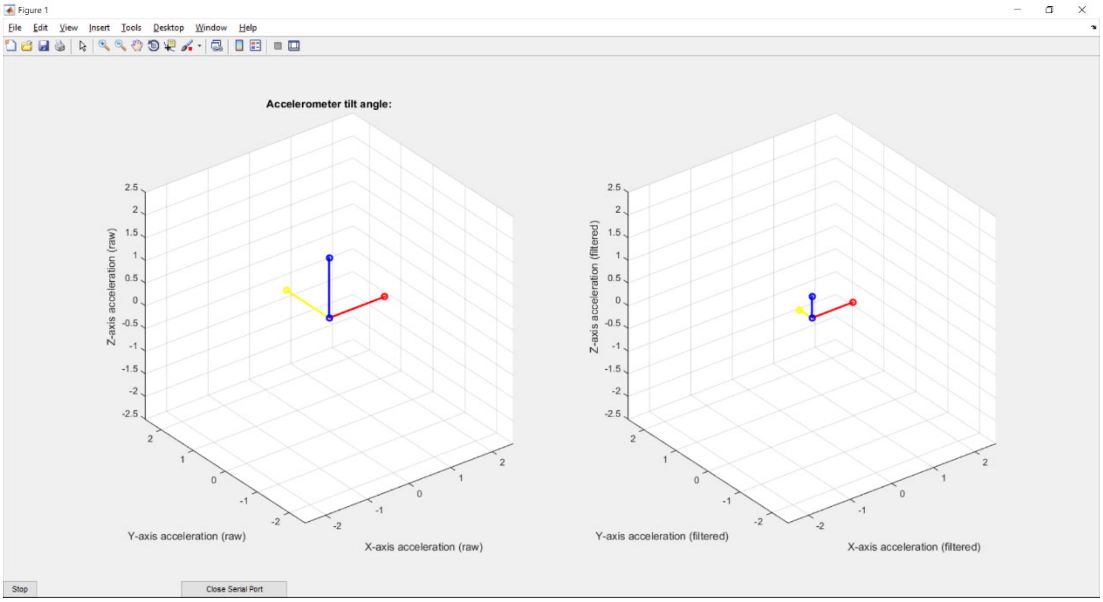


Figure 4.36: Comparison Vector Filter Reading in 3 Colour Axis Acceleration.

Matlab program “magnitudeFilter.m”

```
%% 5. Runs a loop that continually reads the accelerometer values
% The accelerometer data is placed in the variables [gx gy gz].

%initialize filter variable & coefficient
gxFilt = 0;
gyFilt = 0;
gzFilt = 0;
alpha = 0.5;

while (get(button, 'UserData'))

% Update the rolling plot. Append the new reading to the end of the
    % rolling plot data. Drop the first value
    gxdata = [gxdata(2:end);gx];
    gydata = [gydata(2:end);gy];
    gzdata = [gzdata(2:end);gz];

    % Filter the data and update the rolling plot
    gxdataFilt = [gxdataFilt(2:end); ...
        mean(gxdata(buf_len : -1 : buf_len-taps+1))];
    gydataFilt = [gydataFilt(2:end); ...
        mean(gydata(buf_len : -1 : buf_len-taps+1))];
    gzdataFilt = [gzdataFilt(2:end); ...
        mean(gzdata(buf_len : -1 : buf_len-taps+1))];

    %force MATLAB to redraw the figure
    drawnow;

end
```

Figure 4.37: Sample Matlab Program “magnitubeFilter.m”

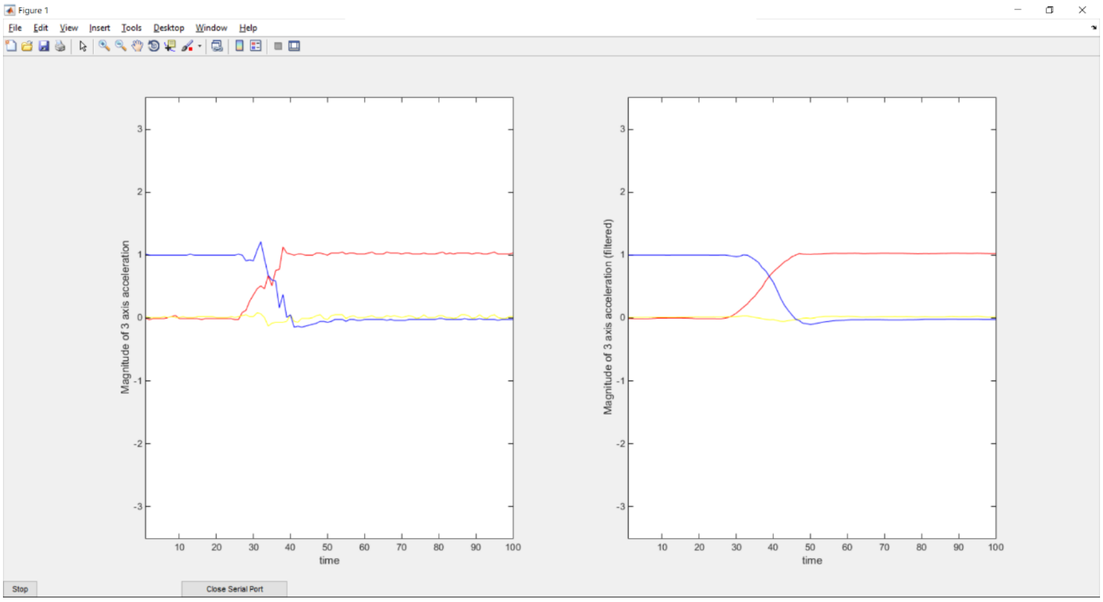


Figure 4.38: Comparison Magnitude Filter Reading in 3 Colour Axis Acceleration.

4.4.5 Threshold Crossing

This threshold crossing is a count the number of times the accelerometer signal crossed the threshold.

Figure 4.39: Ideal for Threshold.

Case 1 – Signal crosses the threshold for the first time.

i. Increment counter variable.

```
stepCount = stepCount + 1;
```
(4.25)

ii. Note that the signal is above the threshold.

```
isHighFlag = 1;
```
(4.26)

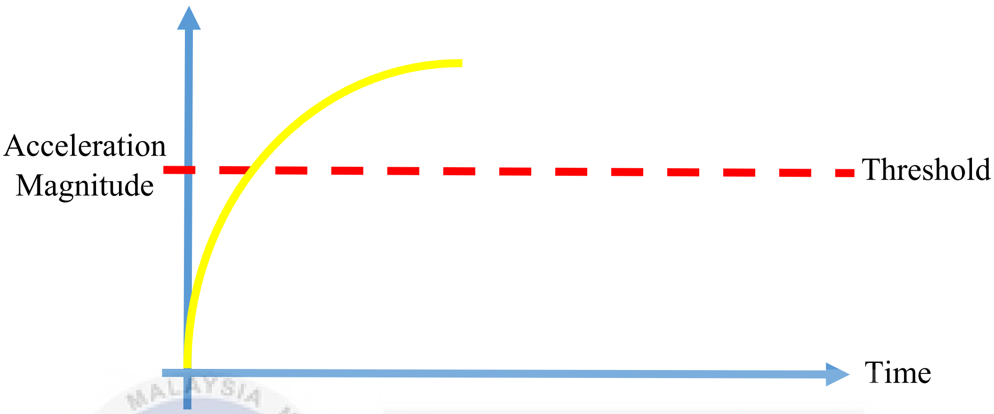


Figure 4.40: Case 1 for Threshold Crossing.

Case 2 – Signal remaining above the threshold.

i. Do nothing.

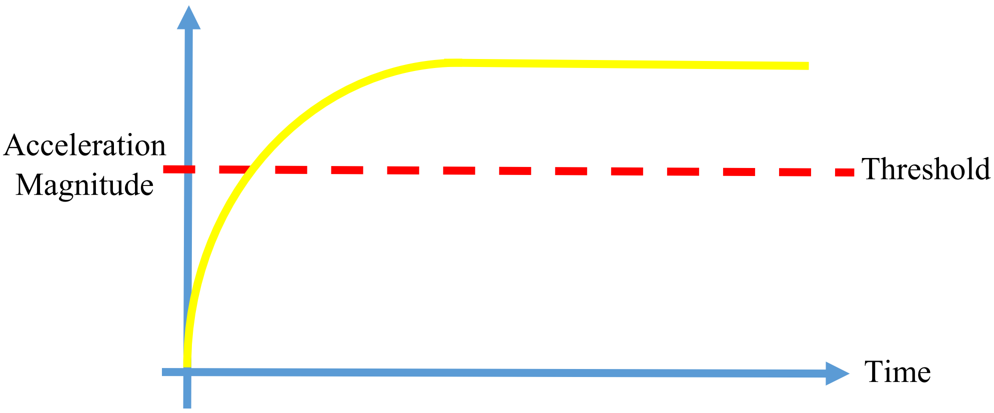


Figure 4.41: Case 2 for Threshold Crossing.

Case 3 – Signal falls below threshold.

i. Note that signal is no longer above the threshold.

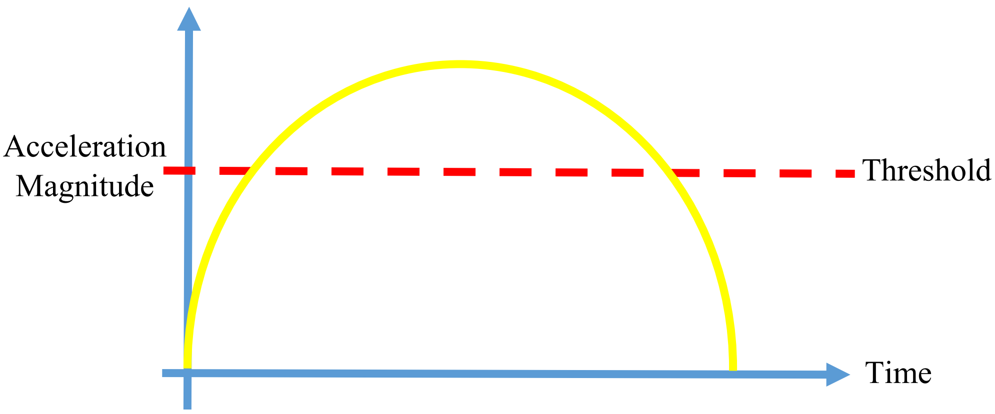
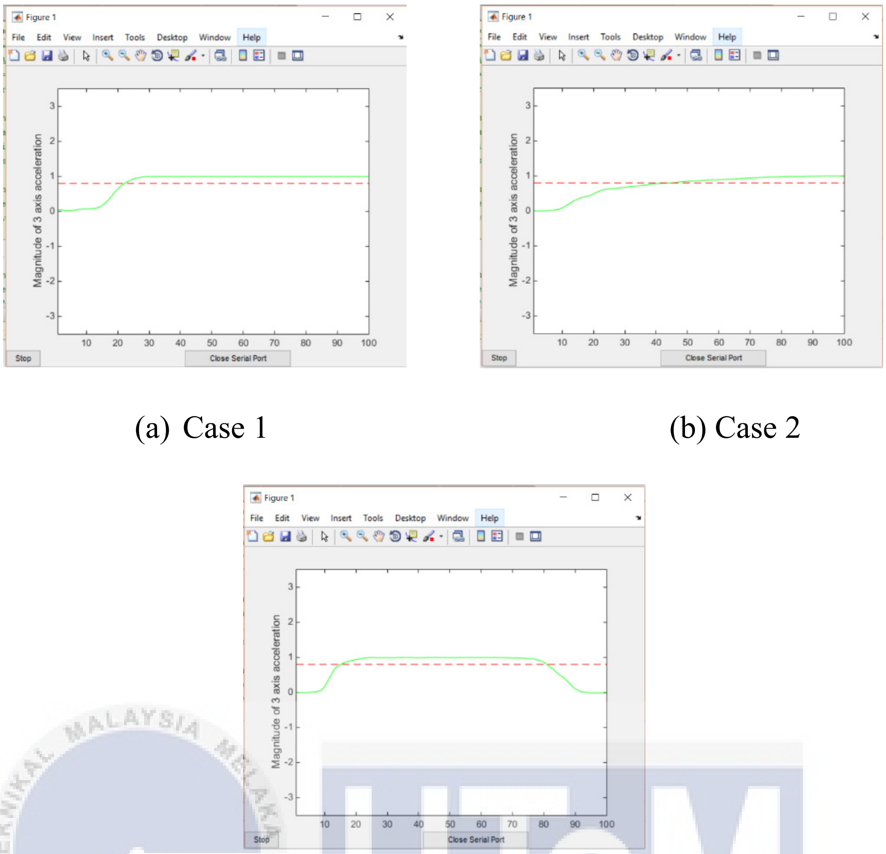


Figure 4.42: Case 3 for Threshold Crossing.

Matlab program "threshold.m"

```
%% 6. Data Collection and Plotting
%initialize filter coefficient
taps = 4;
%initialize threshold value & flag
thresholdValue = .8;
stepCount = 0;
isHighFlag = 0;
% While the figure window is open.
while(get(button,'UserData'))

    %Get the new values from the accelerometer
    [gx, gy, gz] = readAcc(accelerometer, calCo);

    % Update the rolling plot. Append the new reading to the end of the
    % rolling plot data. Drop the first value
    gxdata = [gxdata(2:end);gx];
    %gydata = [gydata(2:end);gy];
    %gzdata = [gzdata(2:end);gz];

    % Filter the data and update the rolling plot
    gxdataFilt = [gxdataFilt(2:end); ...
        mean(gxdata(buf_len : -1 : buf_len-taps+1))];
    %gydataFilt = [gydataFilt(2:end); ...
       % mean(gydata(buf_len : -1 : buf_len-taps+1))];
    %gzdataFilt = [gzdataFilt(2:end); ...
       % mean(gzdata(buf_len : -1 : buf_len-taps+1))];

    % Threshold Case 1: Signal rose above the threshold value for the 1st
    % time
    if gxdataFilt(end) > thresholdValue && isHighFlag == 0
        stepCount = stepCount + 1;
        isHighFlag = 1;

    end

    drawnow;

end
```

Figure 4.43: Sample Matlab Program "threshold.m"



(a) Case 1

(b) Case 2

(c) Case 3

Figure 4.44: Threshold Reading Form Accelerometer Sensor.

4.4.6 Acceleration, Velocity and Position Analysis

This is a robotic arm can provide fast, accurate, and repeatable movement. The feature for this robot included optional wrist rotate, a functional gripper, wrist motion, elbow, single plane shoulder and last is supported overall part of robotic arm by base rotation.

Matlab program "avd.m"

```
%% 6. Data Collection and Plotting
%initialize filter coefficient
taps = 3;
% While the figure window is open.
while(get(button,'UserData'))
    %Get the new values from the accelerometer
    [gx, gy, gz] = readAcc(accelerometer, calCo);

    % Update the rolling plot. Append the new reading to the end of the
    % rolling plot data. Drop the first value
    gxdata = [gxdata(2:end);gx];
    gydata = [gydata(2:end);gy];
    gzdata = [gzdata(2:end);gz];

    % Filter the data and update the rolling plot
    gxdataFilt = [gxdataFilt(2:end); ...
        mean(gxdata(buf_len : -1 : buf_len-taps+1))];
    gydataFilt = [gydataFilt(2:end); ...
        mean(gydata(buf_len : -1 : buf_len-taps+1))];
    gzdataFilt = [gzdataFilt(2:end); ...
        mean(gzdata(buf_len : -1 : buf_len-taps+1))];

    % Accel to Vel
    gxdataVel = [gxdataVel(2:end);gxdataFilt];
    gydataVel = [gydataVel(2:end);gydataFilt];
    gzdataVel = [gzdataVel(2:end);gzdataFilt];

    gxdataVel = cumsum(gxdataFilt);
    gydataVel = cumsum(gydataFilt);
    gzdataVel = cumsum(gzdataFilt);

    % Vel to Pos
    gxdataPos = [gxdataPos(2:end);gxdataVel];
    gydataPos = [gydataPos(2:end);gydataVel];
    gzdataPos = [gzdataPos(2:end);gzdataVel];

    gxdataPos = cumsum(gxdataVel);
    gydataPos = cumsum(gydataVel);
    gzdataPos = cumsum(gzdataVel);

     % Plot
    subplot(3,1,1)
    plot(index, gxdataFilt, 'r', index, gydataFilt, 'y',  index, gzdataFilt,
'b');
    axis([0 buf_len -1.5 1.5]);
    xlabel('time');
    ylabel('Acceleration filter [m/s^2]');

    subplot(3,1,2)
    plot(index, gxdataVel, 'r', index, gydataVel, 'y', index, gzdataVel,
'b');
    axis([0 buf_len -1.5 1.5]);
    xlabel('time');
    ylabel('Velocity [m/s]');

    subplot(3,1,3)
    plot(index, gxdataPos, 'r', index, gydataPos, 'y', index, gzdataPos,
'b');
    axis([0 buf_len -1.5 1.5]);
    xlabel('time');
    ylabel('Position [m]');

    drawnow;
end
```

Figure 4.45: Sample Matlab Program for Acceleration, Velocity, and Position "avd.m"

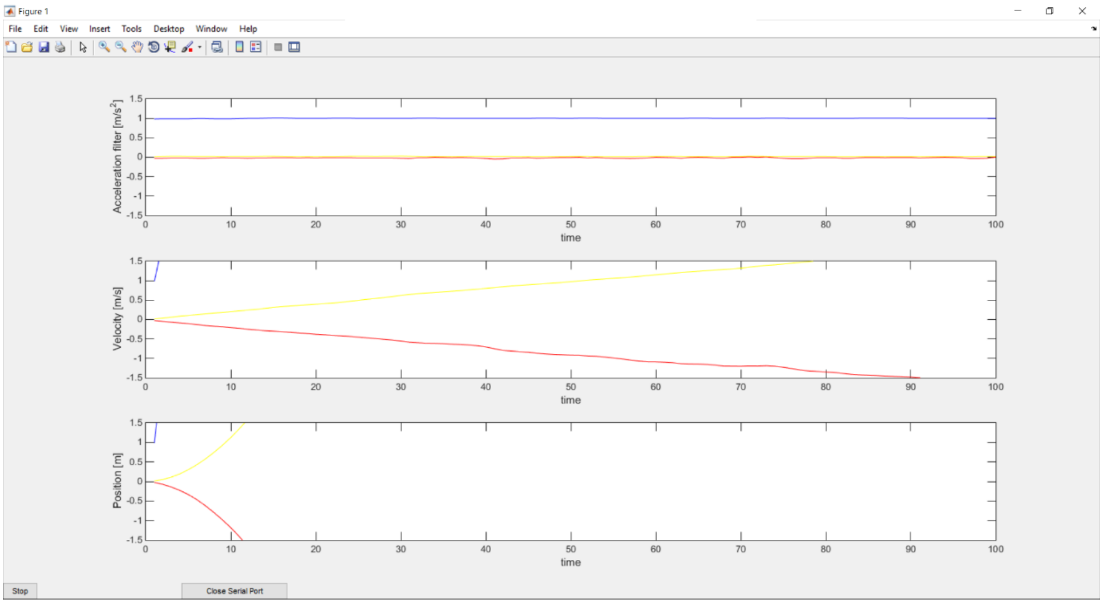


Figure 4.46: Showing 3 Axis Acceleration, Velocity and Position Form Accelerometer Sensor.

4.5 Development of Hand Gesture Glove Controller

This section is summarized the electrical design part of Hand Gesture Glove Controller development.

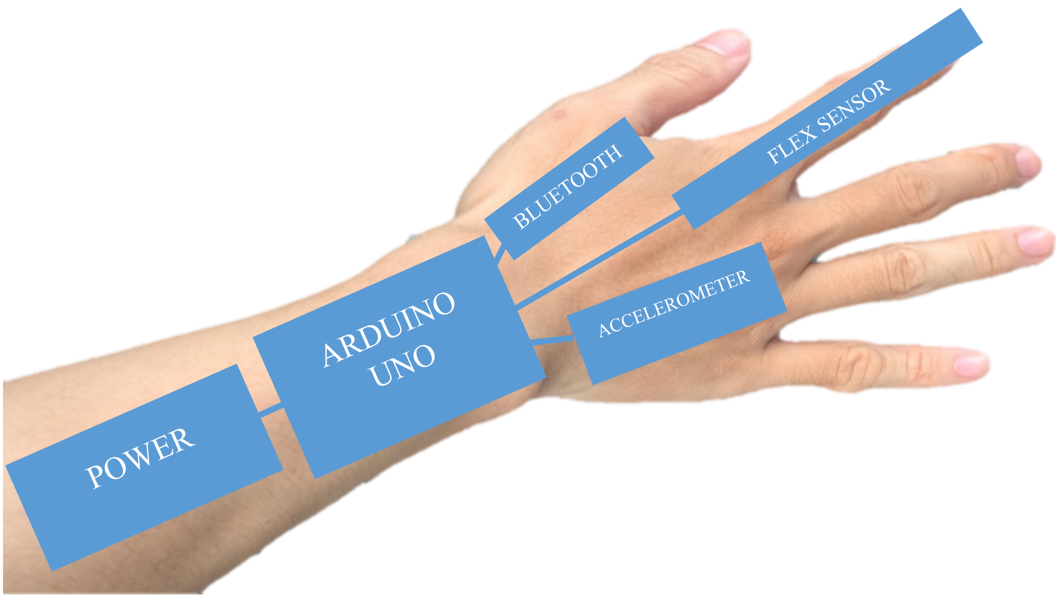


Figure 4.47: Simple Planning Design for Hand Gesture Glove Controller.

List of Electronic Design Part 1 Robotic Arm Control System			
NO	**Electronic Component**	**Type**	**Quantity**
1.	Arduino UNO Microcontroller	ATMega328	1
2.	Bluetooth Communication System	HC-05	1
3.	Wire Jumper Connector	-	Unlimited
4.	YwRobot Power Supply 5V regulator	MB102	2

Table 4.11: List of Electronic Design Part 1 Robotic Arm Control System.

List of Electronic Design Part 2 Human Gesture Glove Controller			
NO	**Electronic Component**	**Type**	**Quantity**
1.	Arduino UNO Microcontroller	ATMega328	1
2.	Bluetooth Communication System	HC-05	1
3.	Accelerometer Sensor	GY-61	1
4.	Flex Sensor	SEN-08606	1
5.	Resistor	15K / 22K	1
6.	Glove	-	1
7.	Wire Jumper Connector	-	Unlimited

Table 4.12: List of Electronic Design Part 2 Human Gesture Glove Controller.

CHAPTER 5
RESULT AND DISCUSSION

In this chapter is to discussed the result of hand gesture controller to control robotic arm, which were based on the objective of this project are obtained. The analysis for this experiment testing is to producing the best performances hand gesture controller to control robotic arm.

5.0 Hand Gesture Glove Controller

This hand gesture glove controller is produced to control robotic arm and this controller is powered by Arduino, wearable, flexible and long battery life. The function for this hand gesture glove is to capture the gesture from human producing. This hand gesture glove communicates with robotic arm using wireless interface technology there is Bluetooth communication system HC-05.

The human gesture capture by using accelerometer sensor GY-61. The accelerometer sensor will sense the human gesture produced into form of data and using Arduino UNO processor to convert the read data from accelerometer sensor into digital value of angle, value of magnitude XYZ, and some mathematic operation. This process of programming is using Arduino IDE. The last part is to control the robotic arm gripper by using flex sensor, the condition of read human gesture is similar with the flow of process accelerometer sensor.

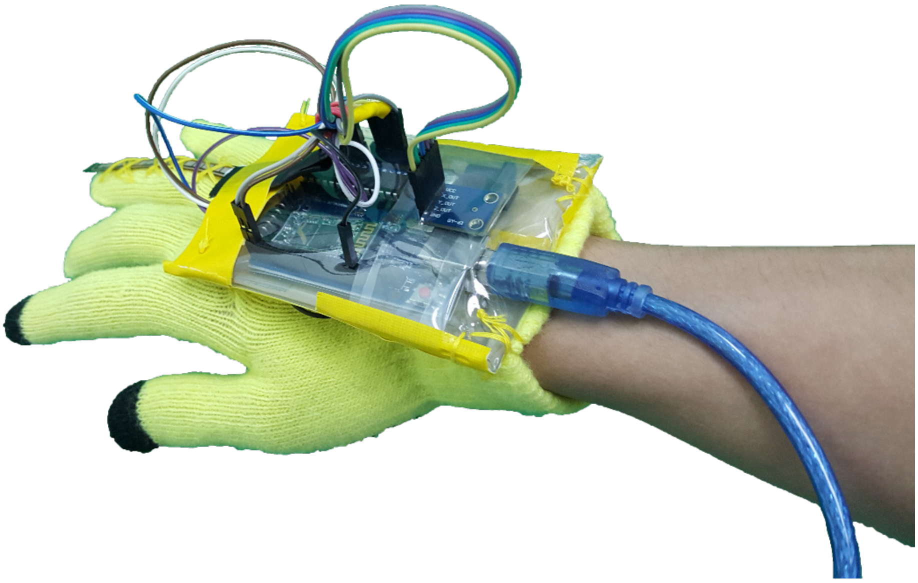


Figure 5.1: Show Hand Gesture Glove Controller.

The result for this hand gesture glove controller, is a type of data communication signal there the signal contains human gesture signal. The ideal for this data communication signal is to convert the pattern of the hand gesture produced by human. After, that the pattern convert into analogue number signal. Last, the analogue number signal is convert into Serial.write() data binary. The description for this Serial.write() is writes binary to the serial port transmission, this data is sent as a byte or series of bytes. The following Figure 5.2 is showing the sample programming to convert some form of data read to binary data and transmits or writes through the serial communication system.


```
if (Serial.read()=='R'){
Serial.write('S');
state1 = Serial.write(xRead); // Sends x
state1 = Serial.write(xRead>>8); // Sends x>>8

state2 = Serial.write(yRead); // Sends y
state2 = Serial.write(yRead>>8); // Sends y>>8

state3 = Serial.write(zRead); // Sends z
state3 = Serial.write(zRead>>8); // Sends z>>8

state4 = Serial.write(gRead); // Sends g
state4 = Serial.write(gRead>>8); // Sends g>>8
Serial.write('E');}
```

Figure 5.2: Sample Coding for Serial Communication System.

The discussion for this Figure 5.2 sample coding for serial communication system meaning that if the receiver sent back to transmitter a character ‘R’ mean that the receiver is standby or ready to receiver any data from transmitter. So, the transmitter continues to write character ‘S’ mean that the transmitter is starting to send before the actual analogue data read. After, sent the ‘S’ that will follow data xRead, yRead, and zRead. Last, to confirm the data transmission complete just program to write character ‘E’ mean the first series data is end and start and transmit again a next series data.

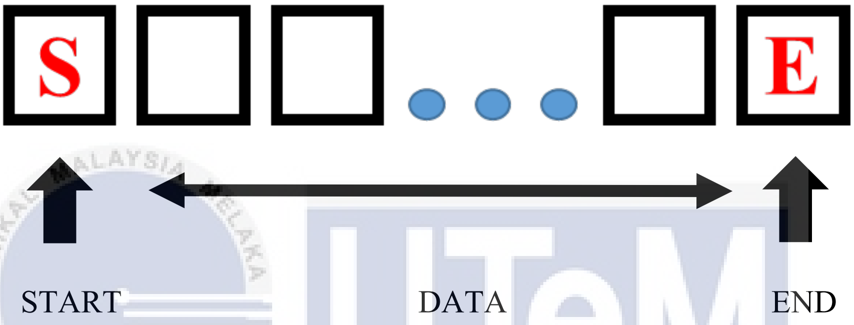


Figure 5.3: Ideal Form of Data Transmission in This Programming.

From this programming is showing in Figure 5.3 the data transmission including binary right shift. The binary right shift operator causes the bit pattern in shift-expression to be shifted to the right by the number of position specified by additive-expression. For example, binary right shift 8 times is showing figure below.

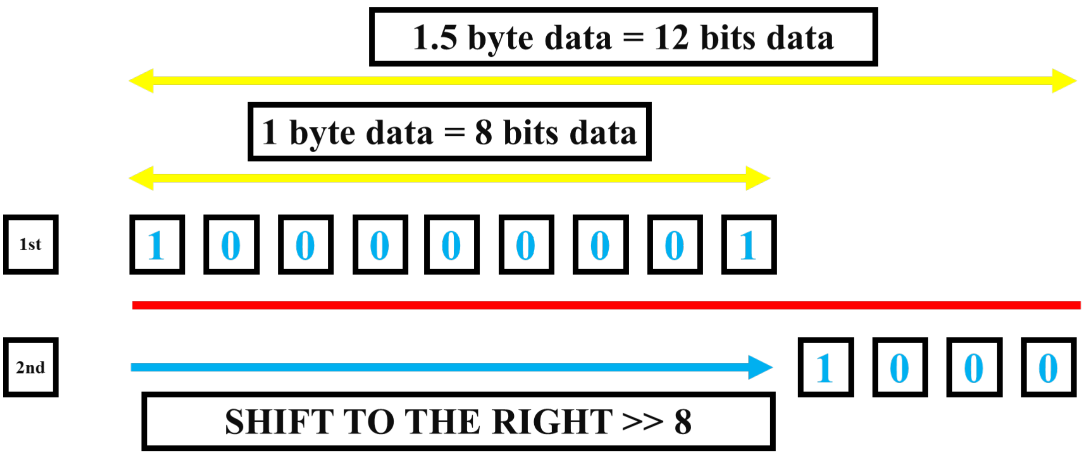


Figure 5.4: Binary Right Shift 8 Times.

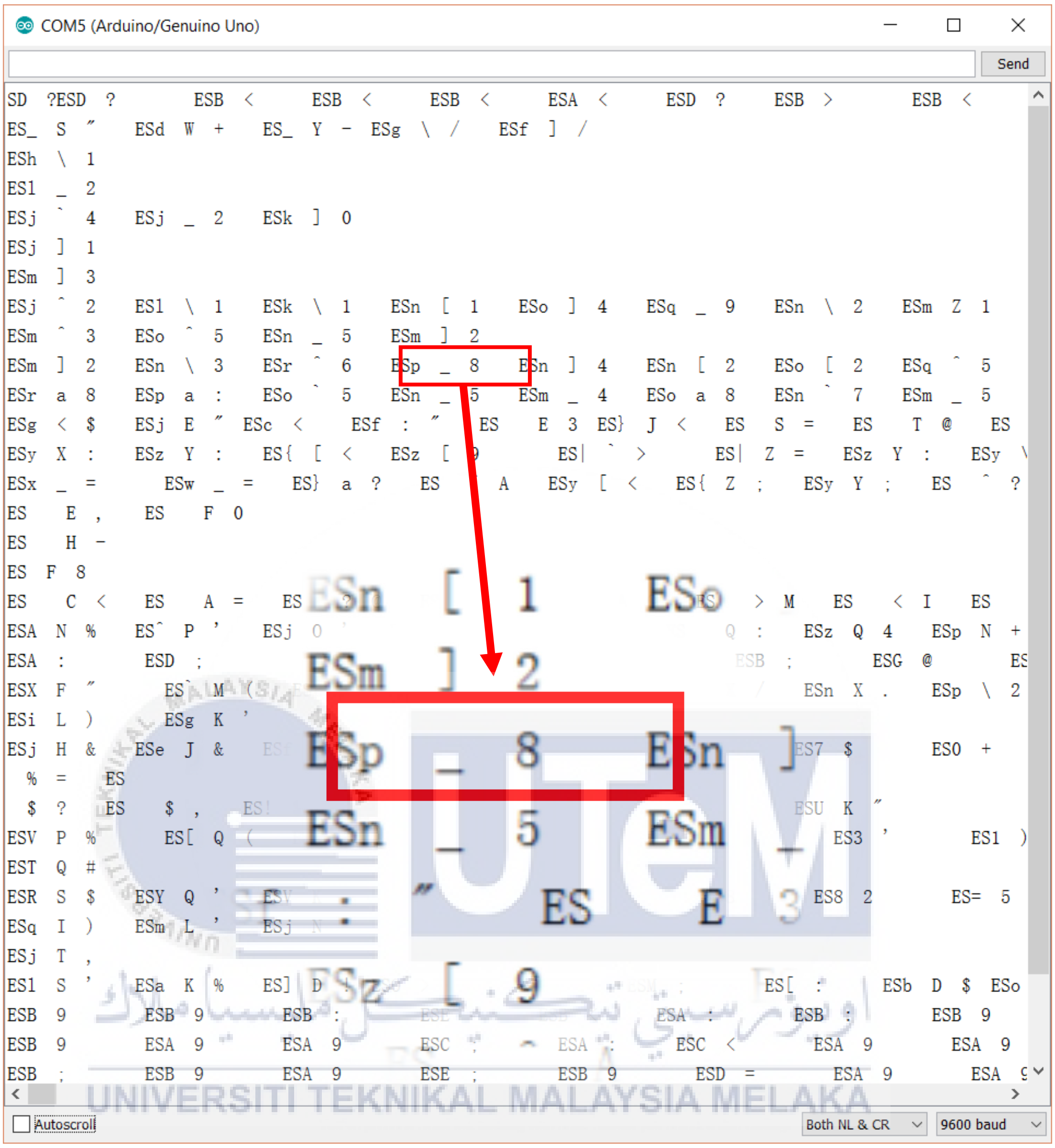


Figure 5.5: Result for Data Accelerometer Completely Sent Via Bluetooth Communication system.

5.1 Controller for Robotic Arm

The Controller for Robotic Arm is to control and set the movement of the robotic arm. This controller is including one set of Arduino UNO, power supply 5V, and Bluetooth communication system. The Arduino for this controller is to program and control the robotic arm. The Bluetooth communication system HC-05 is to become the set of receiver for this control system. The power supply 5V is to supply voltage and current to active the servo motor robotic arm. The main ideal is to receives the accelerometer data from the transmitter and converting the form of data to another form of data through the mathematical operation. The figure below showing controller for the robotic arm and the form of data transmitter and data receives.

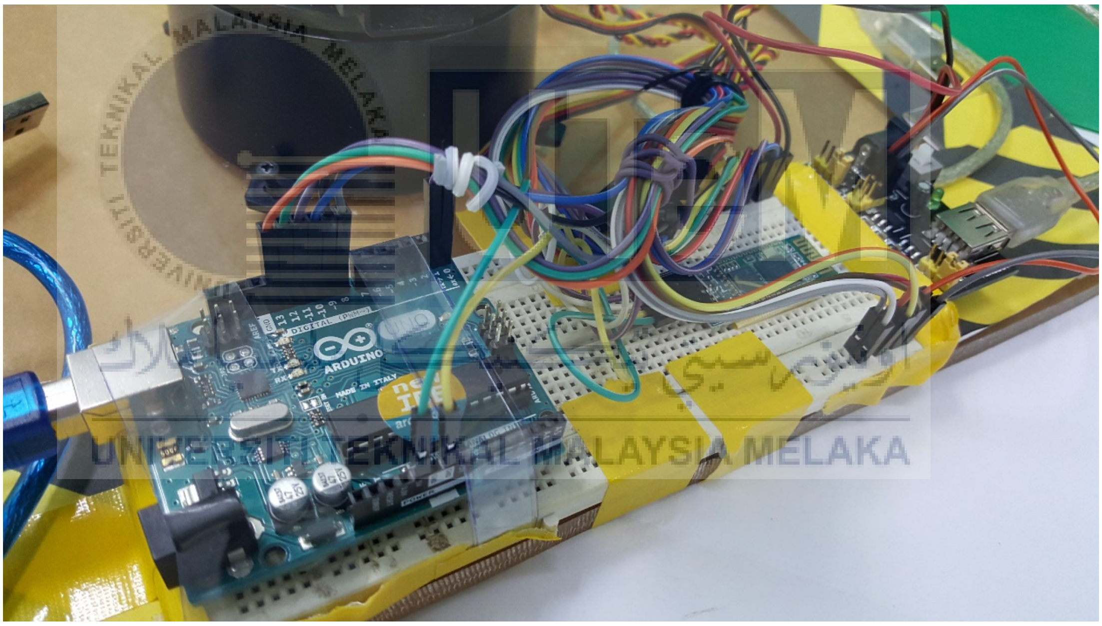


Figure 5.6: Controller for The Robotic Arm.

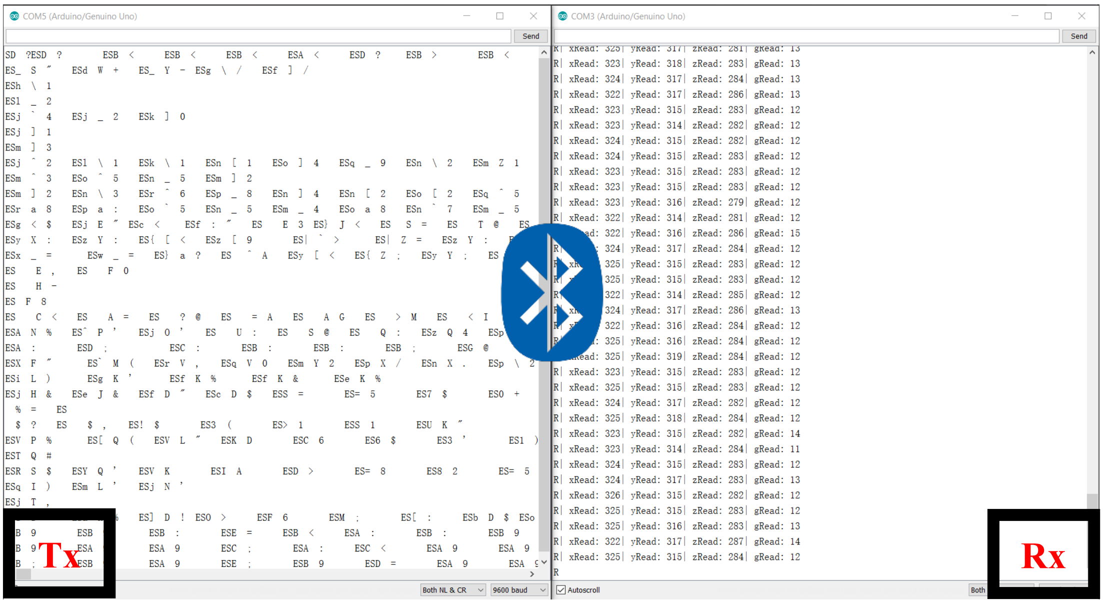


Figure 5.7: Result for data Transmitter and Receivers.

```
 delay(500);
 serialFlush();
 Serial.write('R');
 delay(250);
if(Serial.available() ==10){ // Checks whether data is coming from
the serial port
    if (Serial.read()== 'S'){
    state1 = Serial.read();
    state1 = state1+Serial.read()*256;
    state2 = Serial.read();
    state2 = state2+Serial.read()*256;
    state3 = Serial.read();
    state3 = state3+Serial.read()*256;
    state4 = Serial.read();
    state4 = state4+Serial.read()*256;
    Serial.read();
    }}
```

Figure 5.8: Coding for Receive Data.

The discussion for this Figure 5.8, serialFlush() is the function to clear all data in the receive serial buffer and waits for the transmission of outgoing serial data to complete. After this the receiver sending character ‘R’ mean the receiver is ready to receive any data from transmitter. If the transmitter sending the data, then the receiver is checking with Serial.available() is enough 10 data coming from the serial port. If the data sent not enough 10 data, the program is not continuing to next step. The next step is to make both transmitter and receiver like a pairing password or checking the correct data sending to receiver. If the transmitter sent the character ‘S’ and the receiver should be receiver the data character ‘S’. If any data from character ‘S’ the program would not be continuing receive the data from transmitter or stop the process.

For the next step is to receiver actual reading accelerometer data from transmitter, as we know Serial.write() is only sent as a byte. So in this case may be the transmitter need to two times and the receiver also need to read 2 times from the transmitter. When the receiver Serial.read() receive the first data form accelerometer XYZ and flex sensor, and store in integer state1, state2, state3 and state4. But in this case the data store in integer state is not a complete data, because the data reading from accelerometer is 12 bits for each axis meaning that the data transmit much be separated in 2 times sending. Continue the receiver part also need to receive 2 times form the transmitter, so this second times receive will store and update data from integer state1, state2, state3 and state4. Then, the 2 times receive data need to combine and form a complete one data for the process, this will program summing operation to help both data to form a complete data receive.

For example, the data reading from accelerometer is contain 12 bits data and equal to 1.5 bytes. As this discuss before Serial.write() is only sent a time only contain 1 byte only and in this case is left 0.5 bytes is equal to 4 bits data cannot send through the receiver. So, before the left 0.5 bytes data pass to receiver from the transmitter need to program for the left data 0.5 bytes is shifted 8 bits form the first data 8 bits sending to receiver. After that when the receiver receives the first and second sent data will be summing both together and multiply with 2^8 (256 bytes). This is an actual data for each axis from the accelerometer sensor. Last part, the Serial.read() is an ending read form transmitter, if the transmitter is finish sending the data need from receiver then the transmitter will send the character ‘E’ meaning that the full set complete serial data is sent.

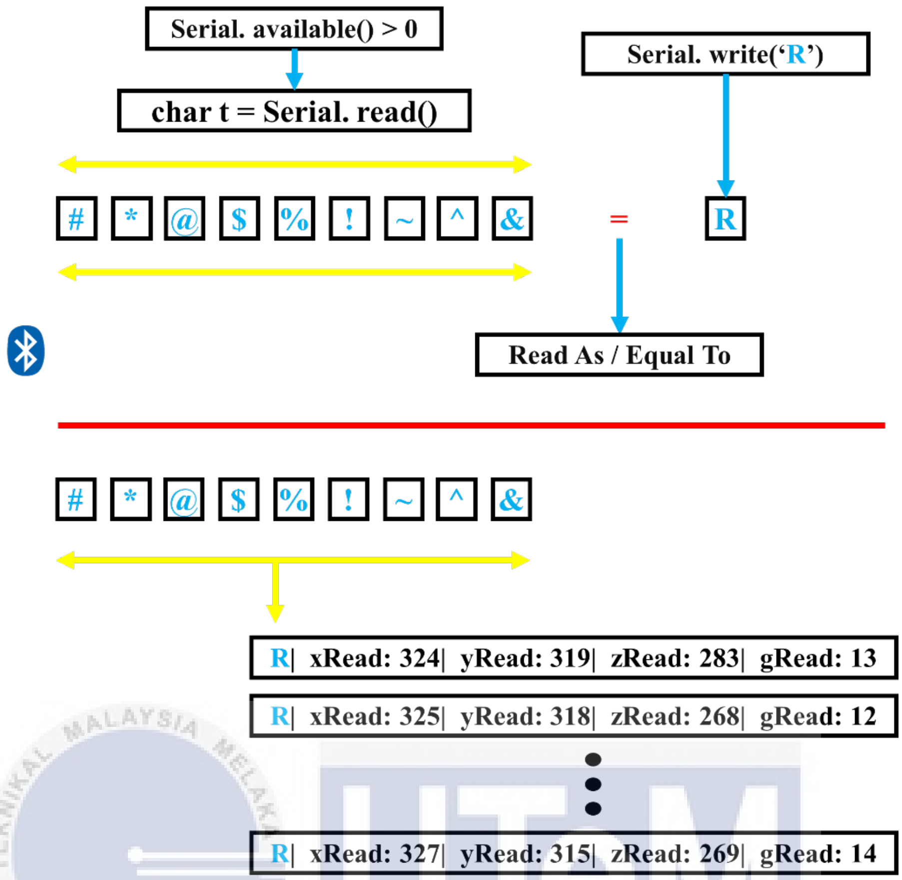


Figure 5.9: The Ideal for Clearing Serial Buffer.

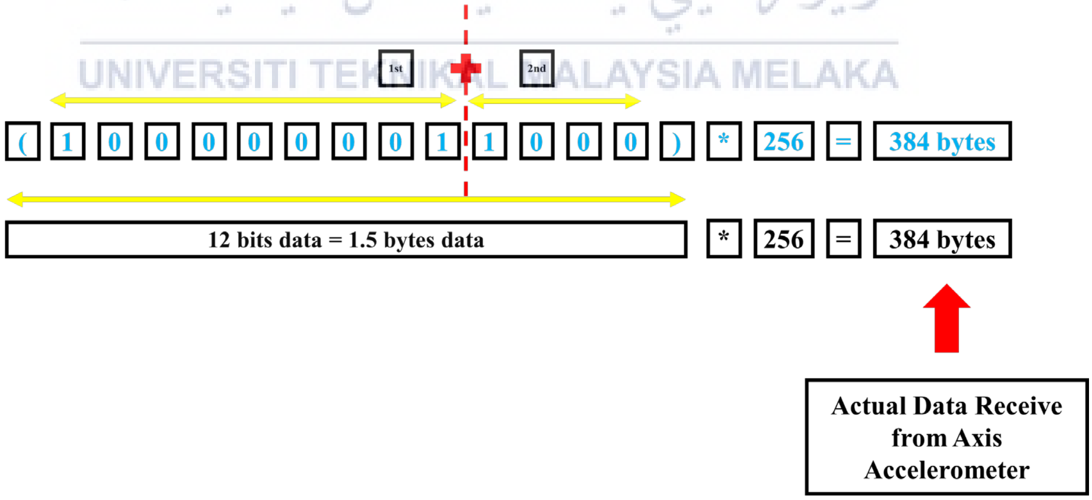


Figure 5.10: The Ideal for Transmit and Receive Data 1.5 bytes Via Bluetooth Communication system.

```
R| xRead: 323| yRead: 317| zRead: 266| gRead: 12
R| xRead: 323| yRead: 317| zRead: 289| gRead: 12
R| xRead: 325| yRead: 318| zRead: 290| gRead: 12
R| xRead: 324| yRead: 319| zRead: 283| gRead: 13
R| xRead: 326| yRead: 318| zRead: 267| gRead: 12
R| xRead: 323| yRead: 318| zRead: 266| gRead: 14
R| xRead: 325| yRead: 317| zRead: 267| gRead: 12
R| xRead: 323| yRead: 319| zRead: 268| gRead: 12
R| xRead: 325| yRead: 318| zRead: 267| gRead: 12
R| xRead: 324| yRead: 319| zRead: 268| gRead: 12
R| xRead: 323| yRead: 319| zRead: 269| gRead: 13
R| xRead: 321| yRead: 318| zRead: 268| gRead: 13
R| xRead: 323| yRead: 317| zRead: 267| gRead: 12
R| xRead: 325| yRead: 318| zRead: 267| gRead: 12
R| xRead: 332| yRead: 320| zRead: 266| gRead: 12
R| xRead: 325| yRead: 317| zRead: 267| gRead: 12
R| xRead: 323| yRead: 317| zRead: 268| gRead: 12
R| xRead: 323| yRead: 315| zRead: 267| gRead: 12
R| xRead: 326| yRead: 317| zRead: 269| gRead: 12
R| xRead: 324| yRead: 315| zRead: 267| gRead: 14
R| xRead: 325| yRead: 316| zRead: 267| gRead: 13
R| xRead: 327| yRead: 317| zRead: 267| gRead: 13
R| xRead: 325| yRead: 316| zRead: 267| gRead: 14
R| xRead: 326| yRead: 319| zRead: 267| gRead: 14
```

Figure 5.11: Result for Data Completely Receive from Transmitter.

5.2 World Frame, Base Frame and Gripper Frame

The description of a frame is the information needed to completely specify the whereabouts of the manipulator hand in Figure 5.12 is a position and an orientation. The point on the body whose position we describe could be chosen arbitrarily, however. For convenience, the point whose position we will describe is chosen as the origin of the body-attached frame. The situation of a position and an orientation pair arises so often in robotics that we define an entity called a frame, which is a set of four vectors going position and orientation information.

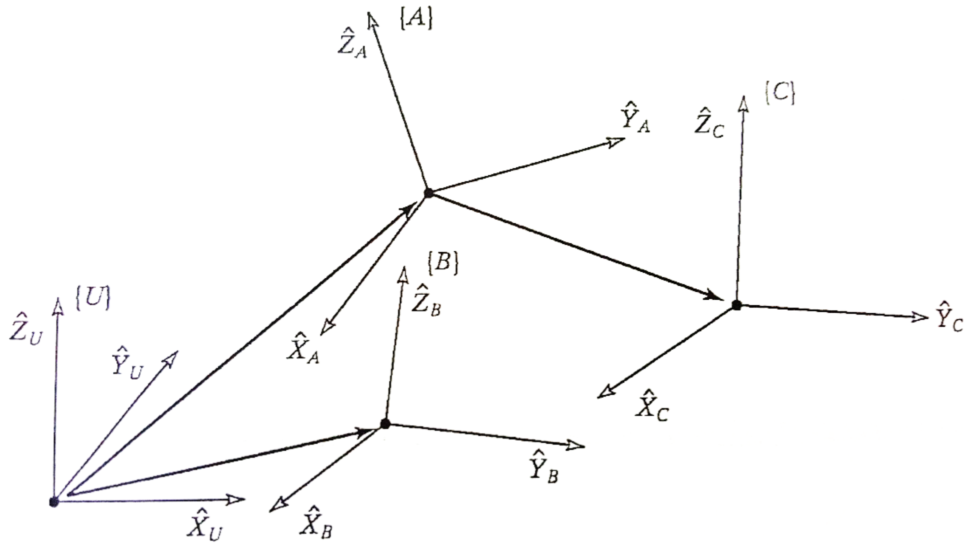


Figure 5.12: Example of Frames.

From understanding about the description of a frame, so this type of the concept convert in to the robotic arm is to determine the action and motion movement when human gesture controller control the robot arm. Before that, the robotic arm need to decide our self, where is the world frame, base frame, shoulder frame, elbow frame, wrist frame and gripper frame location needed to move the robotic arm.

What is world frame? World frame is frame where the starting location point. In Figure 5.13 showing the robotic arm with the world frame and Figure 5.14 showing the robotic arm with base, shoulder, elbow and wrist frame. To determine the freedom pattern of robotic arm action and motion movement.

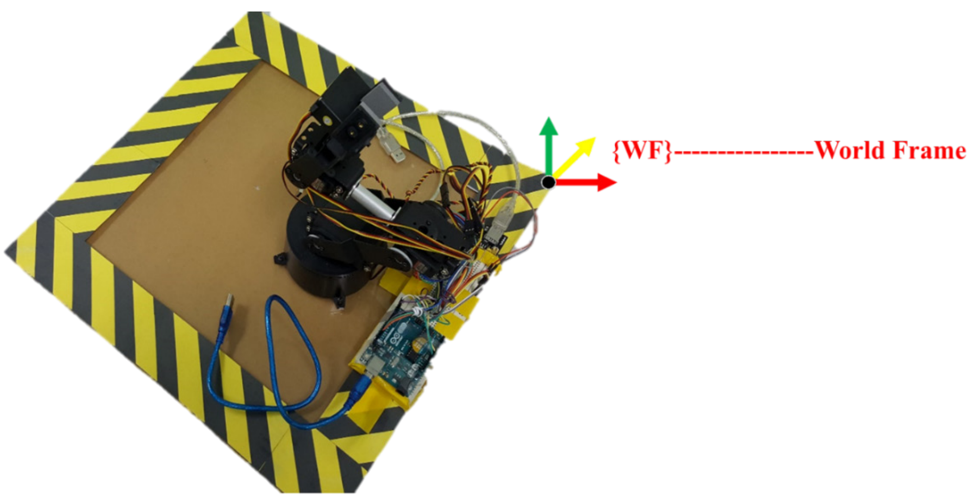


Figure 5.13: Robotic Arm with World Frame {WF}.

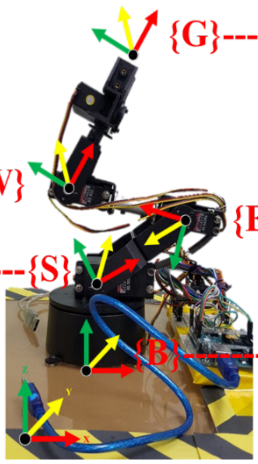


Figure 5.14: Robotic Arm with Base, Shoulder, Elbow and Wrist Frame.

After finish determine each frame on this robotic arm. Which can be continue to find out how far the length in between world frame to object frame. The purpose is to determine how far the robotic arm can pick and place from the object set point. For the figure below is showing 3 type of object set point. Meaning that the robotic arm much be undergo the three pattern process to pick and place the object. For example, figure showing at below.

In this session also showing the discussion to discuss the comparison about the calculation and measurement. To proving the similarity value of calculation with the measurement of motion robotic arm movement and also calculate how far the translation or how many rotation movements to pick and place from the world and object set point in Table 5.1.

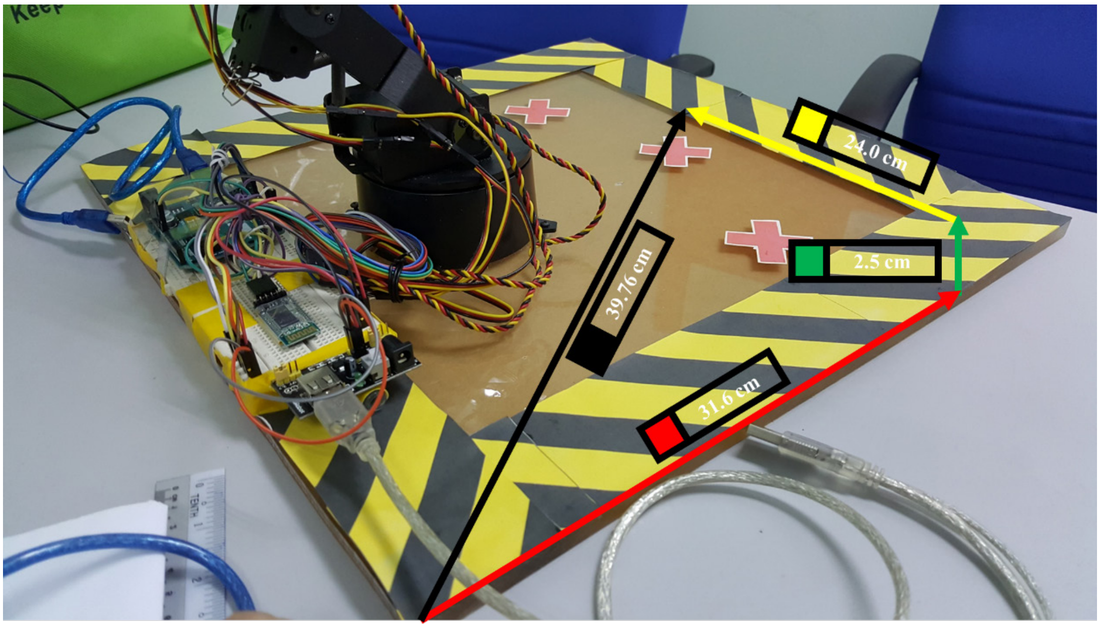


Figure 5.15: Showing the Direction 3 Axis Length for First Object $T1^{O}_{WF}$.

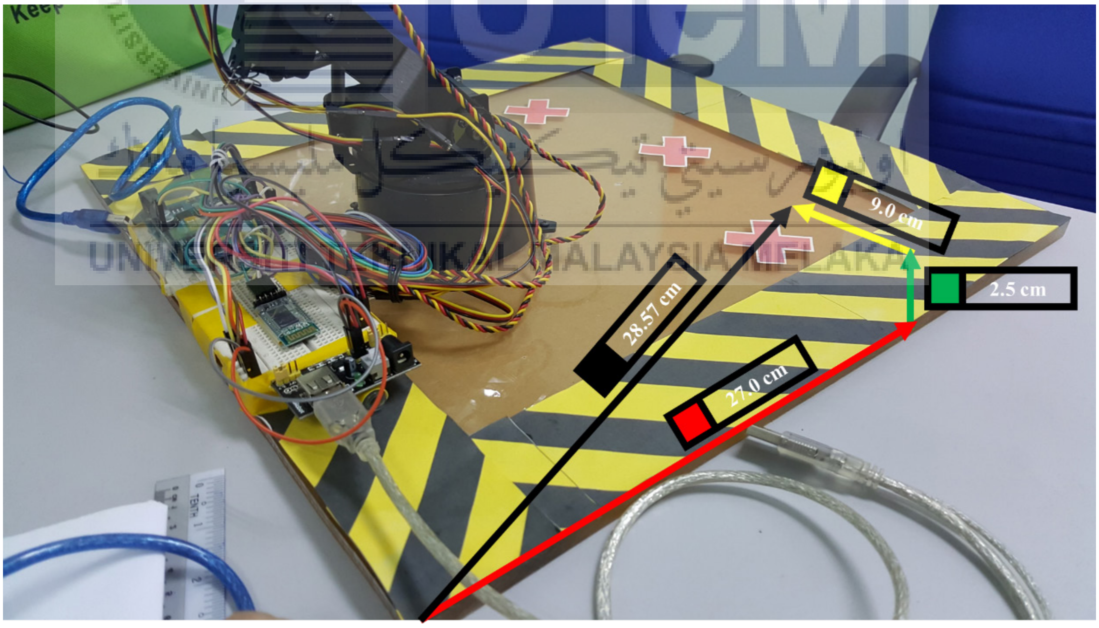


Figure 5.16: Showing the Direction 3 Axis Length for Second Object $T2^{O}_{WF}$.

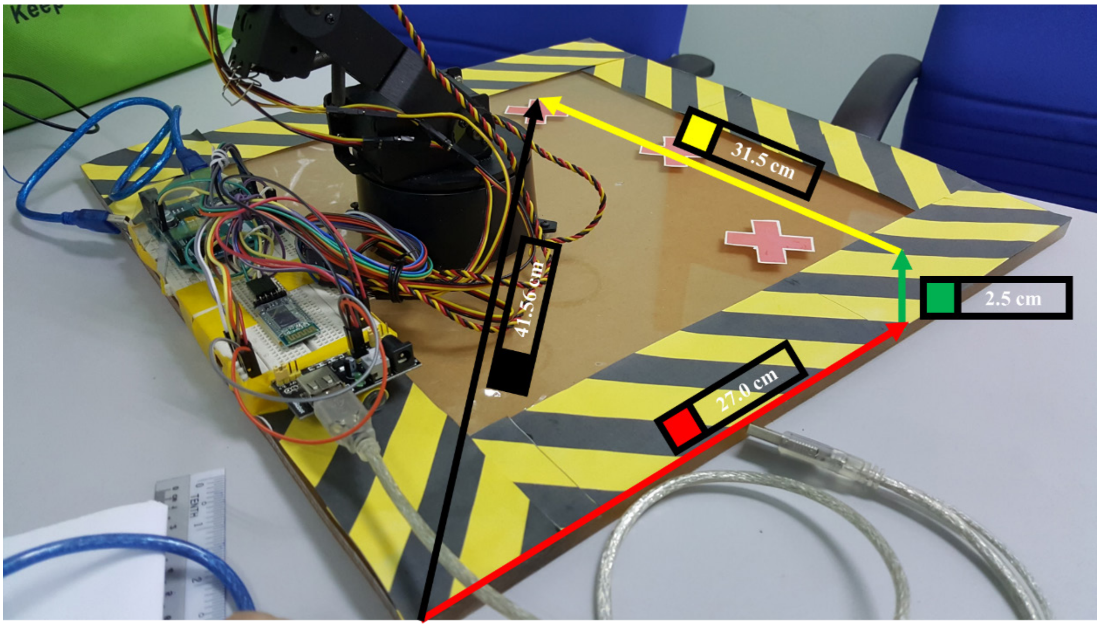


Figure 5.17: Showing the Direction 3 Axis Length for Third Object $T3^{O}_{WF}$.

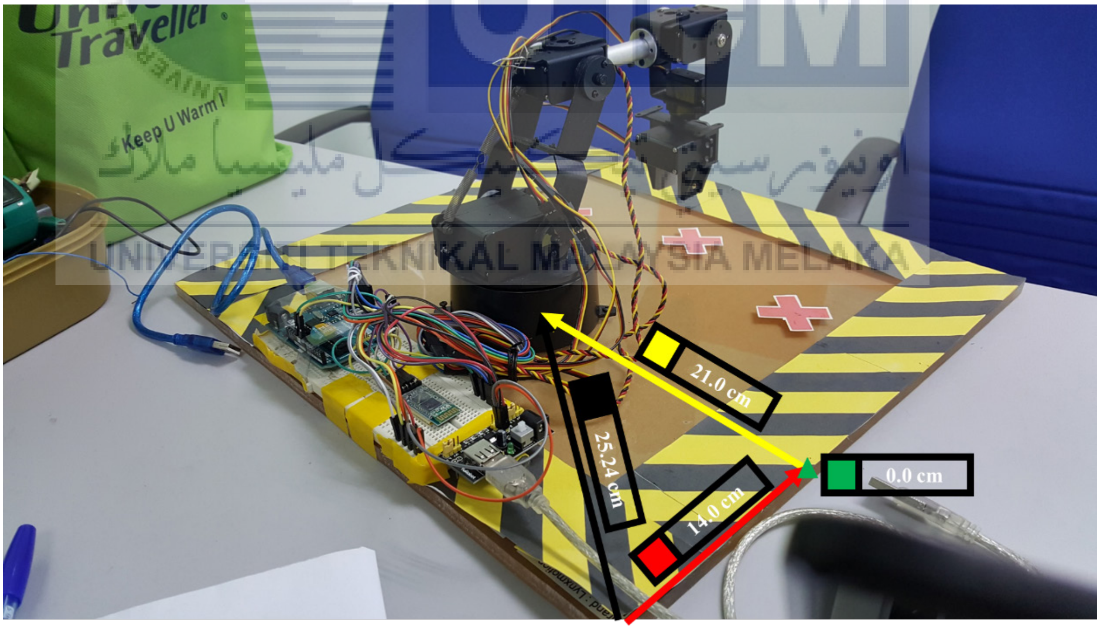


Figure 5.18: Showing the Direction 3 Axis Length from World to Based Frame T^{B}_{WF}.

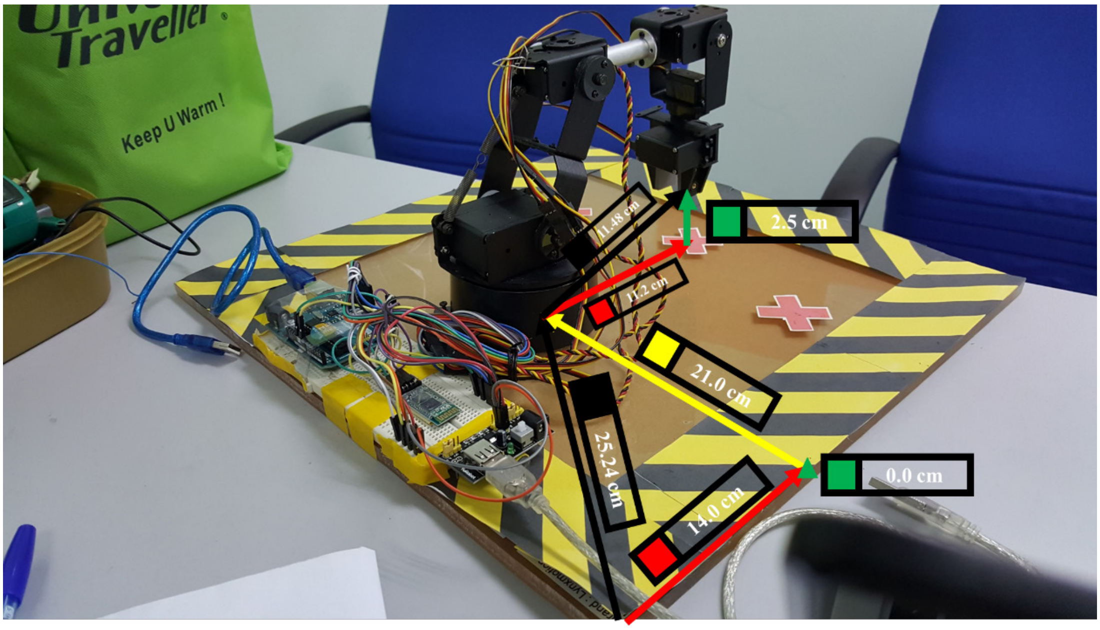


Figure 5.19: Showing the Direction 3 Axis Length for T_{WF}^{B} and $T1_{B}^{O}$.

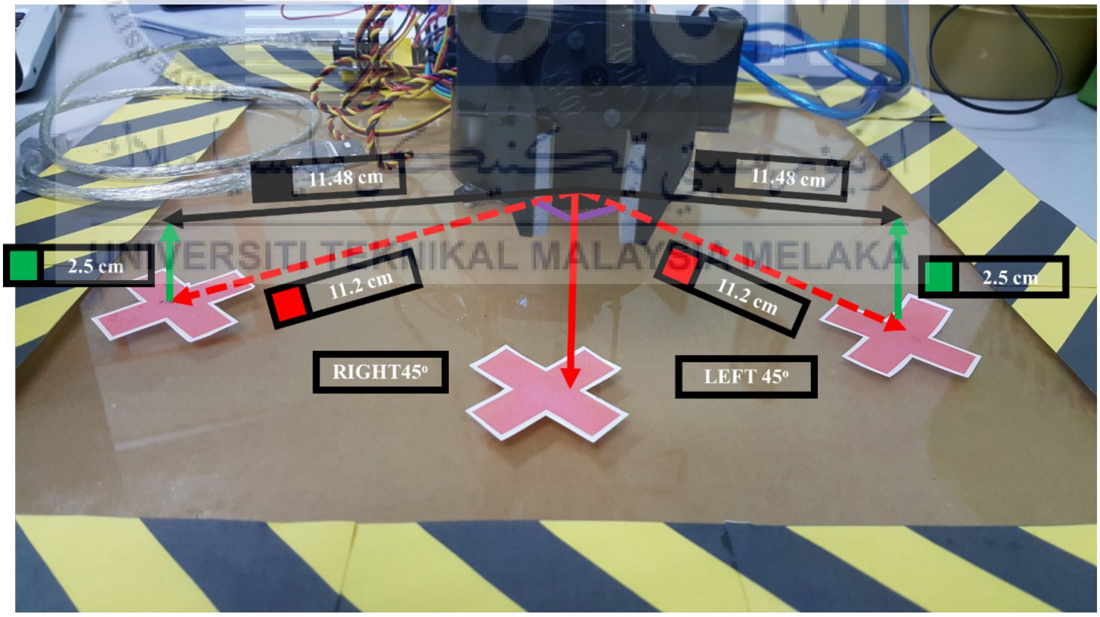


Figure 5.20: Showing the Direction 3 Axis Length for Second $T2_{B}^{O}$ and Third Object Pick $T3_{B}^{O}$.

Discussion Simple Calculation and Measurement		
Label	Measure	Calculate
$T1^{O}_{WF}$	$\begin{matrix} 1 & 0 & 0 & 31.6 \\ 0 & 1 & 0 & 24.0 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{matrix}$	>> TWB = [1,0,0,14.0; 0,1,0,21.0; 0,0,1,0; 0,0,0,1] TWB = 1 0 0 14 0 1 0 21 0 0 1 0 0 0 0 1 >> T1BO = [1,0,0,11.2; 0,1,0,0; 0,0,1,2.5; 0,0,0,1] T1BO = 1.0000 0 0 11.2000 0 1.0000 0 0 0 0 1.0000 2.5000 0 0 0 1.0000 >> T1WO = TWB * T1BO T1WO = 1.0000 0 0 25.2000 0 1.0000 0 21.0000 0 0 1.0000 2.5000 0 0 0 1.0000
$T2^{O}_{WF}$	$\begin{matrix} 1 & 0 & 0 & 27.0 \\ 0 & 1 & 0 & 9.0 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{matrix}$	>> TWB = [1,0,0,14.0; 0,1,0,21.0; 0,0,1,0; 0,0,0,1] TWB = 1 0 0 14 0 1 0 21 0 0 1 0 0 0 0 1 >> T2BO = [0.5253,-0.8509,0,11.2; 0.8509,0.5253,0,0; 0,0,1,2.5; 0,0,0,1] T2BO = 0.5253 -0.8509 0 11.2000 0.8509 0.5253 0 0 0 0 1.0000 2.5000 0 0 0 1.0000 >> T2WO = TWB * T2BO T2WO = 0.5253 -0.8509 0 25.2000 0.8509 0.5253 0 21.0000 0 0 1.0000 2.5000 0 0 0 1.0000

<table>
<tr>
<td>$T3_{WF}^{O}$</td>
<td>$\begin{matrix} 1 & 0 & 0 & 27.0 \\ 0 & 1 & 0 & 31.5 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{matrix}$</td>
<td><pre>
>> TWB = [1,0,0,14.0; 0,1,0,21.0; 0,0,1,0; 0,0,0,1]

TWB =

     1   0   0   14
     0   1   0   21
     0   0   1   0
     0   0   0   1

>> T3BO = [0.5253,0.8509,0,11.2; -0.8509,0.5253,0,0; 0,0,1,2.5; 0,0,0,1]

T3BO =

    0.5253   0.8509   0   11.2000
   -0.8509   0.5253   0   0
         0   0   1.0000   2.5000
         0   0   0   1.0000

>> T3WO = TWB * T3BO

T3WO =

    0.5253   0.8509   0   25.2000
   -0.8509   0.5253   0   21.0000
         0   0   1.0000   2.5000
         0   0   0   1.0000
</pre></td>
</tr>
<tr>
<td>T_{wF}^{B}</td>
<td colspan="2">$\begin{matrix} 1 & 0 & 0 & 14.0 \\ 0 & 1 & 0 & 21.0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$</td>
</tr>
<tr>
<td>$T1_{B}^{O}$</td>
<td colspan="2">$\begin{matrix} 1 & 0 & 0 & 11.2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{matrix}$</td>
</tr>
</table>

$T2_B^O$	$\begin{matrix} 0.5253 & -0.8509 & 0 & 11.2 \\ 0.8509 & 0.5253 & 0 & 0 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{matrix}$
$T3_B^O$	$\begin{matrix} 0.5253 & 0.8509 & 0 & 11.2 \\ -0.8509 & 0.5253 & 0 & 0 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{matrix}$

Table 5.1: Simple Calculation and Measurement for Each Frame Robotic Arm.

Finally, the discussion to compare between calculation and measurement of the robotic arm movement also not so much similar, because this comparison between calculation and measurement having some error problem to control or interrupt our data comparison between calculation and measurement. The type of error in measurement system there is systematic error and random errors. Systematic error including instrumental errors, environmental errors, observational errors and theoretical errors. For the random errors also including gross errors, blunders and measurement error. To avoid this type of error to control or interrupt our comparison between calculation and measurement, so trying difference method solution to solve out the calculation there is manipulator kinematics.

The introduction of kinematics is the science of motion that treats the subject without regard to the forces that causes it. In this session consider position and orientation of the manipulator linkages in statics situations. Link description including revolute, cylindrical, screw, prismatic, planar and spherical.

The summary of the link parameters in terms of the link frames. If the link frames have been attached to the links according to our convention, the following definitions of the link parameters are valid:

a_i = the distance from Z_i to Z_{i+1} measured along X_i;

α_i = the angle from Z_i to Z_{i+1} measured along X_i;

d_i = the distance from X_{i-1} to X_i measurement along Z_i;

θ_i = the angle from X_{i-1} to X_i measured about Z_i.

The summary of link frame attachment procedure. First, identify the joint axes and imagine infinite lines along them. Second, identify the common perpendicular between them, or point of intersection. Third, assign the Z_i axis pointing along the *i* th joint axis. Fourth, assign the X_i axis pointing along the common perpendicular, or if the axes intersect, assign X_i to be normal to the plane containing the two axes. Fifth, assign the Y_i axis to complete a right-hand coordinate system. Sixth, assign {0} to match {1} when the first joint variable is zero. The equation showing below is derivation of link transformations.

$$ {}^{i-1}_{\;\;i}T = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i\, C\alpha_{i-1} & C\theta_i\, C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\ S\theta_i\, S\alpha_{i-1} & C\theta_i\, S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1} $$

From Figure 5.11, and convert it in to DH-table meaning that is link parameter of the robotic arm. This table is showing below:

i	***α_{i-1}***	***a_{i-1}***	***d_i***	***θ_i***
1	0	0	0	θ1
2	+90°	0	0	θ2
3	0	L1	0	θ3
4	0	L2	0	θ4
5	-90°	0	0	θ5

Table 5.2: DH-Table (Link Parameter Robotic Arm).

Using (5.1), we compute each of the link transformations:

$$ {}^{1}_{0}T = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & 0 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{1}T = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{2}T = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & L1 \\ S\theta_3 & C\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{3}T = \begin{bmatrix} C\theta_4 & -S\theta_4 & 0 & L2 \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{4}T = \begin{bmatrix} C\theta_5 & -S\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S\theta_5 & -C\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.2) $$

Finally, we obtain the product of all five link transforms:

$$ {}^{5}_{0}T = {}^{1}_{0}T \; {}^{2}_{1}T \; {}^{3}_{2}T \; {}^{4}_{3}T \; {}^{5}_{4}T \qquad (5.3) $$

When the angle (θ) and length (L) given is showing below:

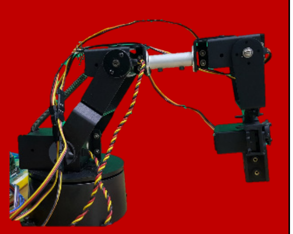
i	α_{i-1}	a_{i-1}	d_i	θ_i	
1	0	0	0	0°	
2	+90°	0	0	70°	
3	0	9.5cm	0	90°	
4	0	10.5cm	0	0°	
5	-90°	0	0	90°	

Table 5.3: DH-Table When Robotic Arm Start Pick Condition.

```
>> T01 = [1,0,0,0; 0,1,0,0; 0,0,1,0; 0,0,0,1]

T01 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> T12 = [0.342,-0.9397,0,0; 0,0,-1,0; 0.9397,0.342,0,0; 0,0,0,1]

T12 =

    0.3420   -0.9397         0         0
         0         0   -1.0000         0
    0.9397    0.3420         0         0
         0         0         0    1.0000

>> T23 = [0,-1,0,9.5; 1,0,0,0; 0,0,1,0; 0,0,0,1]

T23 =

         0   -1.0000         0    9.5000
    1.0000         0         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T34 = [1,0,0,10.5; 0,1,0,0; 0,0,1,0; 0,0,0,1]

T34 =

    1.0000         0         0   10.5000
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T45 = [0,-1,0,0; 0,0,1,0; -1,0,0,0; 0,0,0,1]

T45 =

     0    -1     0     0
     0     0     1     0
    -1     0     0     0
     0     0     0     1

>> T05 = T01*T12*T23*T34*T45

T05 =

         0    0.9397   -0.3420   -6.6178
    1.0000         0         0         0
         0   -0.3420   -0.9397   12.5181
         0         0         0    1.0000
```

Figure 5.21: Result Calculation for Robotic Arm Start Pick Condition.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	80º
2	+90º	0	0	180º
3	0	9.5cm	0	150º
4	0	10.5cm	0	135º
5	-90º	0	0	90º

Table 5.4: DH-Table When Robotic Arm Initial Condition.

```
>> T01 = [0.1736,-0.9848,0,0; 0.9848,0.1736,0,0; 0,0,1,0; 0,0,0,1]

T01 =

    0.1736   -0.9848         0         0
    0.9848    0.1736         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T12 = [-1,0,0,0; 0,0,-1,0; 0,-1,0,0; 0,0,0,1]

T12 =

    -1     0     0     0
     0     0    -1     0
     0    -1     0     0
     0     0     0     1

>> T23 = [-0.866,-0.5,0,9.5; 0.5,-0.866,0,0; 0,0,1,0; 0,0,0,1]

T23 =

   -0.8660   -0.5000         0    9.5000
    0.5000   -0.8660         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T34 = [-0.7071,-0.7071,0,10.5; 0.7071,-0.7071,0,0; 0,0,1,0; 0,0,0,1]

T34 =

   -0.7071   -0.7071         0   10.5000
    0.7071   -0.7071         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T45 = [0,-1,0,0; 0,0,1,0; -1,0,0,0; 0,0,0,1]

T45 =

     0    -1     0     0
     0     0     1     0
    -1     0     0     0
     0     0     0     1

>> T05 = T01*T12*T23*T34*T45

T05 =

   -0.9848    0.0449   -0.1677   -0.0707
    0.1736    0.2549   -0.9512   -0.4008
         0   -0.9659   -0.2588   -5.2500
         0         0         0    1.0000
```

Figure 5.22: Result Calculation for Robotic Arm Initial Condition.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	0°
2	+90°	0	0	90°
3	0	9.5cm	0	100°
4	0	10.5cm	0	0°
5	-90°	0	0	90°

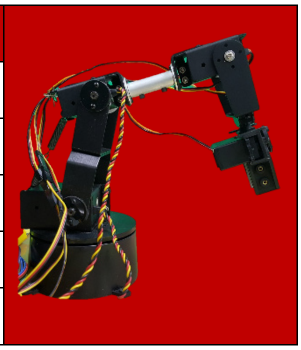


Table 5.5: DH-Table When Robotic Arm Completed Pick Up.

```
>> T01 = [1,0,0,0; 0,1,0,0; 0,0,1,0; 0,0,0,1]

T01 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> T12 = [0,-1,0,0; 0,0,-1,0; 1,0,0,0; 0,0,0,1]

T12 =

     0    -1     0     0
     0     0    -1     0
     1     0     0     0
     0     0     0     1

>> T23 = [-0.1736,-0.9848,0,9.5; 0.9848,-0.1736,0,0; 0,0,1,0; 0,0,0,1]

T23 =

   -0.1736   -0.9848         0    9.5000
    0.9848   -0.1736         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T34 = [1,0,0,10.5; 0,1,0,0; 0,0,1,0; 0,0,0,1]

T34 =

    1.0000         0         0   10.5000
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T45 = [0,-1,0,0; 0,0,1,0; -1,0,0,0; 0,0,0,1]

T45 =

     0    -1     0     0
     0     0     1     0
    -1     0     0     0
     0     0     0     1

>> T05 = T01*T12*T23*T34*T45

T05 =

         0    0.9848    0.1736  -10.3404
    1.0000         0         0         0
         0    0.1736   -0.9848    7.6772
         0         0         0    1.0000
```

Figure 5.23: Result Calculation for Robotic Arm Completed Pick Up.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	45°
2	+90°	0	0	70°
3	0	9.5cm	0	90°
4	0	10.5cm	0	0°
5	-90°	0	0	90°

Table 5.6: DH-Table When Robotic Arm Turn Right.

```
>> T01 = [0.7071,-0.7071,0,0; 0.7071,0.7071,0,0; 0,0,1,0; 0,0,0,1]

T01 =

    0.7071   -0.7071         0         0
    0.7071    0.7071         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T12 = [0.3420,-0.9397,0,0; 0,0,-1,0; 0.9397,0.3420,0,0; 0,0,0,1]

T12 =

    0.3420   -0.9397         0         0
         0         0   -1.0000         0
    0.9397    0.3420         0         0
         0         0         0    1.0000

>> T23 = [0,-1,0,9.5; 1,0,0,0; 0,0,1,0; 0,0,0,1]

T23 =

         0   -1.0000         0    9.5000
    1.0000         0         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T34 = [1,0,0,10.5; 0,1,0,0; 0,0,1,0; 0,0,0,1]

T34 =

    1.0000         0         0   10.5000
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T45 = [0,-1,0,0; 0,0,1,0; -1,0,0,0; 0,0,0,1]

T45 =

     0    -1     0     0
     0     0     1     0
    -1     0     0     0
     0     0     0     1

>> T05 = T01*T12*T23*T34*T45

T05 =

   -0.7071    0.6645   -0.2418   -4.6795
    0.7071    0.6645   -0.2418   -4.6795
         0   -0.3420   -0.9397   12.5181
         0         0         0    1.0000
```

Figure 5.24: Result Calculation for Robotic Arm Turn Right.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	135°
2	+90°	0	0	70°
3	0	9.5cm	0	90°
4	0	10.5cm	0	0°
5	-90°	0	0	90°

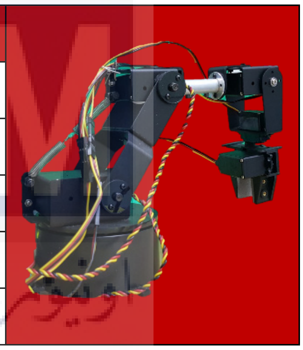


Table 5.7 DH-Table When Robotic Arm Turn Left.


```
>> T01 = [-0.7071,-0.7071,0,0; 0.7071,-0.7071,0,0; 0,0,1,0; 0,0,0,1]

T01 =

   -0.7071   -0.7071         0         0
    0.7071   -0.7071         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T12 = [0.3420,-0.9397,0,0; 0,0,-1,0; 0.9397,0.3420,0,0; 0,0,0,1]

T12 =

    0.3420   -0.9397         0         0
         0         0   -1.0000         0
    0.9397    0.3420         0         0
         0         0         0    1.0000

>> T23 = [0,-1,0,9.5; 1,0,0,0; 0,0,1,0; 0,0,0,1]

T23 =

         0   -1.0000         0    9.5000
    1.0000         0         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T34 = [1,0,0,10.5; 0,1,0,0; 0,0,1,0; 0,0,0,1]

T34 =

    1.0000         0         0   10.5000
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000

>> T45 = [0,-1,0,0; 0,0,1,0; -1,0,0,0; 0,0,0,1]

T45 =

     0    -1     0     0
     0     0     1     0
    -1     0     0     0
     0     0     0     1

>> T05 = T01*T12*T23*T34*T45

T05 =

   -0.7071   -0.6645    0.2418    4.6795
   -0.7071    0.6645   -0.2418   -4.6795
         0   -0.3420   -0.9397   12.5181
         0         0         0    1.0000
```

Figure 5.25: Result Calculation for Robotic Arm Turn Left.

5.3 Performance of Gripper and Robotic Arm Movement

This wireless interface control system transmit time taken is achieving approximately 0.1 second in between hand gesture glove controller and robotic arm. The result performance of gripper and robotic arm movement is showing in Table 5.8 below:

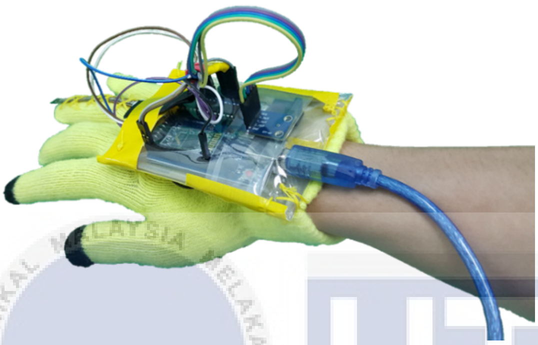
Hand Gesture Glove Controller	Time (TX/RX)	Robotic Arm
	>>>>>>> ≈ 0.1sec >>>>>>>	

Table 5.8: Showing The Result Performance in This Wireless Interface Control System

The speed performance of robotic arm controller process is achieving approximately 0.75 seconds. In this process time is separated in to 2 different process which is 0.50 seconds is for process clearing serial buffer serialFlush () and 0.25 seconds is for process read data or value signal from sensor through serial.available(), if equal to 10 data process. More detail explanation can be referring to Figure 5.9.

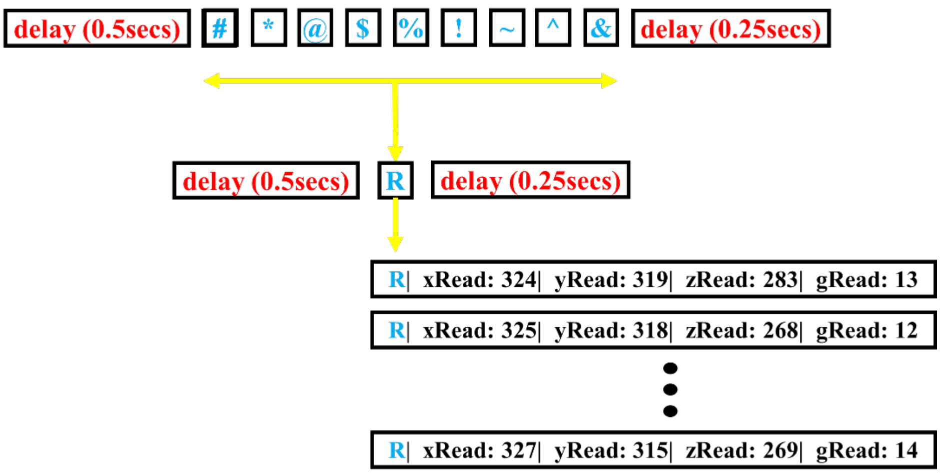


Figure 5.26: The Ideal for Time Processing Is 0.75 secs

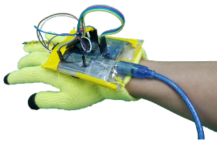
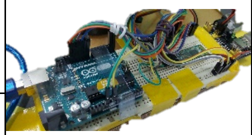
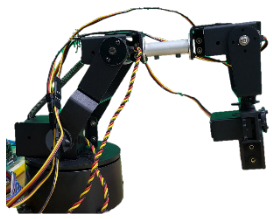
Summary Performance Hand Gesture Glove Controller Control Robotic Arm					
Hand Gesture Glove Controller	Time (Tx/Rx)	Robotic Arm Controller	Time (Process Arduino)		Robotic Arm
	>>>>> $\approx$ **0.10s** >>>>>		>>>>> >>>>> $\approx$ **0.75s** + $\approx$ **0.15s** >>>>> >>>>>		 **$\approx$ 1 second Output Process**
	Transmitting Data Speed		**Receiving Data Speed**	**Arduino Process Speed**	

Table 5.9: Summary Speed Performance Hand Gesture Glove Controller Control Robotic Arm.

Summary Pattern Movement Performance Hand Gesture Glove Controller Control Robotic Arm		
Human Hand	**Hand Glove**	**Robotic Arm**
Human Hand Up and Robotic Arm Initial Condition		
		
Human Hand Down and Robotic Arm Start To Pick		
		

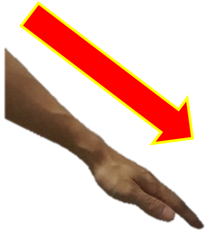
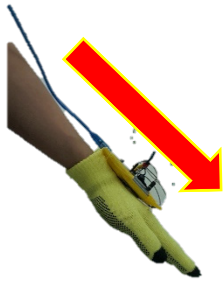
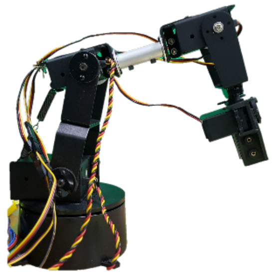
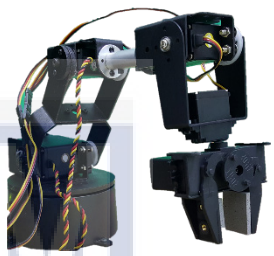
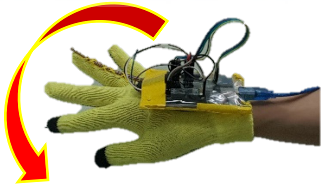
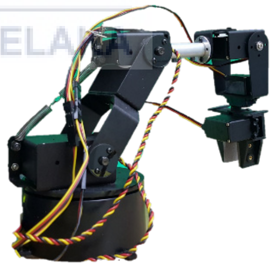
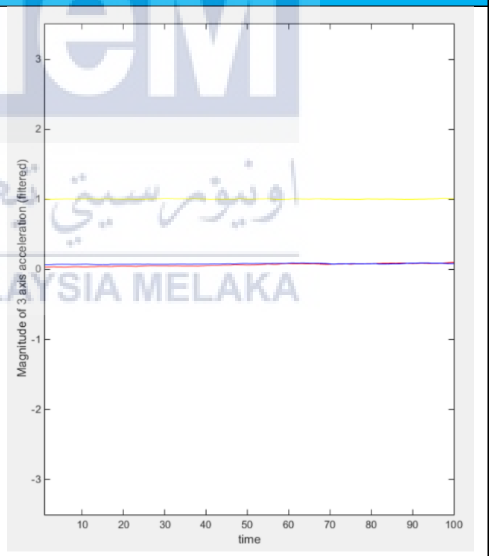
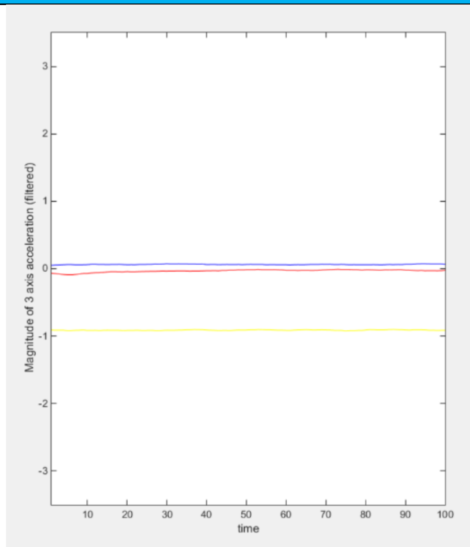
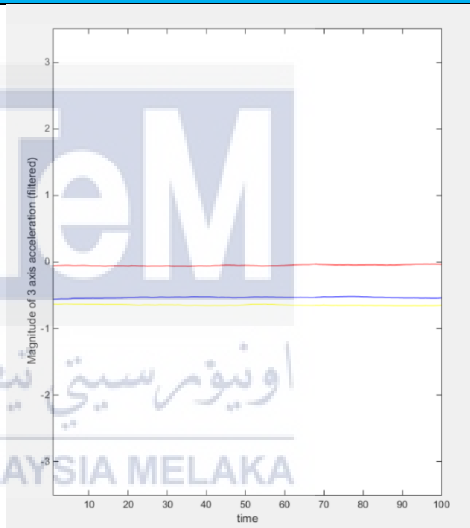
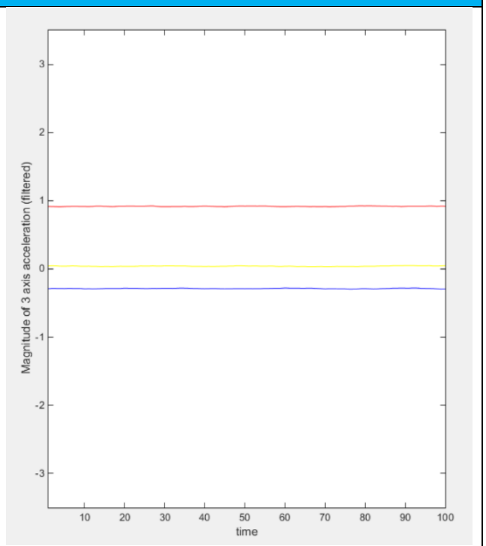
Human Hand Up 45° and Robotic Arm Completed Pick Up		
		
Human Hand Turn Right and Robotic Arm Turn Right		
		
Human Hard Turn Left and Robotic Arm Turn Left		
		

Table 5.10: Summary Pattern Movement Performance Hand Gesture Glove Controller Control Robotic Arm.

5.4 Analysis Human Hand Gesture Glove Controller

This analysis is to show or determine the human hand gesture provided by accelerometer sensor and convert the signal to show in signal waveform and display on screen computer. The 5 type of robotic arm pattern movement provided by human hand gesture glove controller including gesture hand up, hand down, hand turn right, hand turn left and hand up to 45°. The 5 difference type of gesture provided by human and the sensor of the controller also changing accordingly by the 5 difference human gesture. The Table 5.11 and 5.12 shows below is the summary analysis hand gesture glove controller control robotic arm with magnitude and vector signal.

Summary Analysis Hand Gesture Glove Controller Control Robotic Arm		
Human Hand	**Hand Glove**	**Magnitude Signal**
Human Hand Up and Robotic Arm Initial Condition		
		

Human Hand Down and Robotic Arm Start To Pick		
		
Human Hand Up 45° and Robotic Arm Completed Pick Up		
		
Human Hand Turn Right and Robotic Arm Turn Right		
		

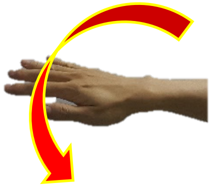
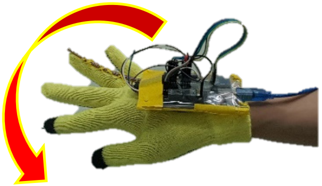
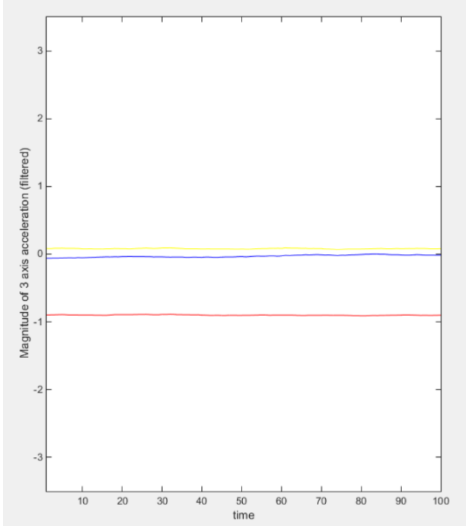
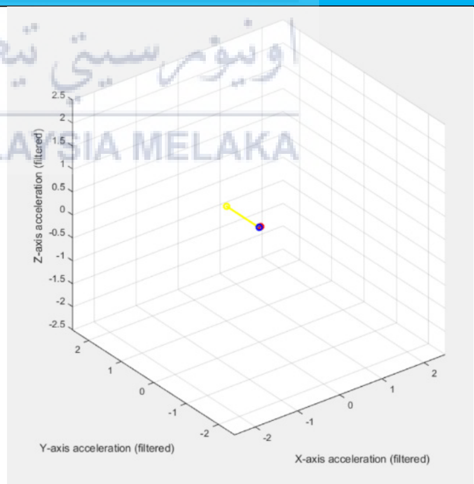
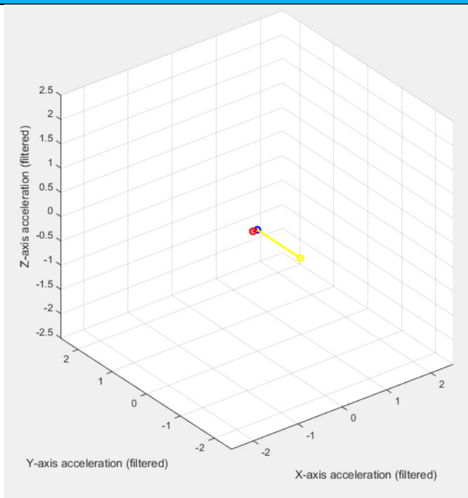
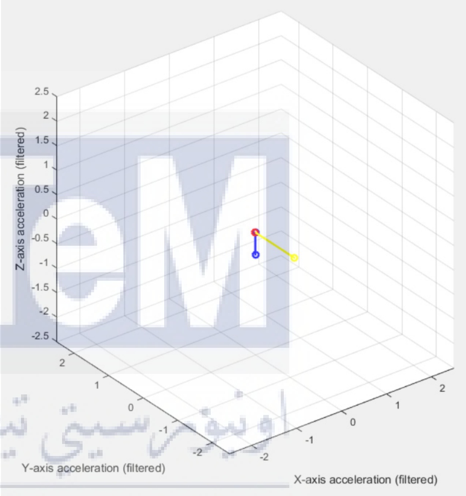
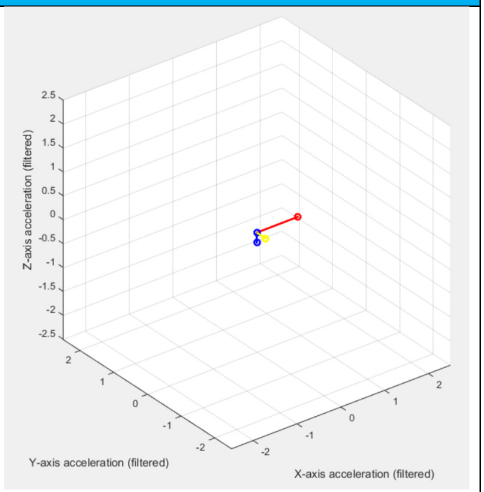
Human Hard Turn Left and Robotic Arm Turn Left		
		

Table 5.11: Summary Analysis Gesture Glove Controller with Magnitude Signal.

Summary Analysis Hand Gesture Glove Controller Control Robotic Arm		
Human Hand	**Hand Glove**	**Vector Signal**
Human Hand Up and Robotic Arm Initial Condition		
		

<table>
<tr><th colspan="3">Human Hand Down and Robotic Arm Start To Pick</th></tr>
<tr><td></td><td></td><td>
</td></tr>
<tr><th colspan="3">Human Hand Up 45° and Robotic Arm Completed Pick Up</th></tr>
<tr><td></td><td></td><td>
</td></tr>
<tr><th colspan="3">Human Hand Turn Right and Robotic Arm Turn Right</th></tr>
<tr><td></td><td></td><td>
</td></tr>
</table>

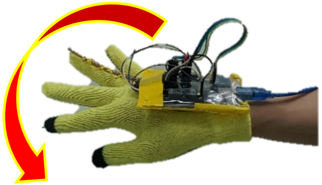
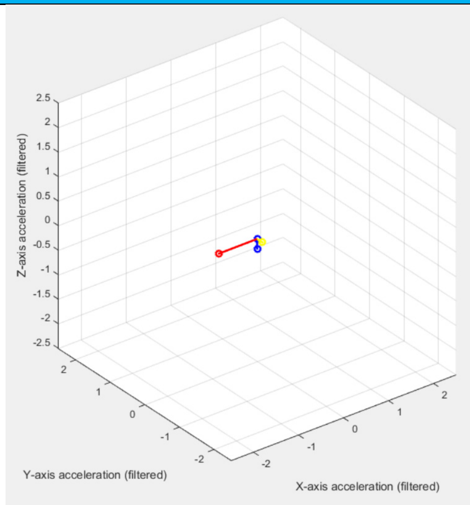
Human Hard Turn Left and Robotic Arm Turn Left		
		

Table 5.12: Summary Analysis Gesture Glove Controller with Vector Signal.

Figure 5.27: The Colour Axis Set for Shows the Difference Axis Magnitude and Vector Signal Provided by Accelerometer Sensor.

5.5 Discussion

The result and discussion in this development of hand gestures glove to control robot arm is to analysis the performance of the wireless communication control system, accelerometer sensor control system, flex sensor control system, and human gesture control system. This result and discussion is important to very project development. The result of development of hand gesture to control robot arm using wireless communication control system can provide approximately 1 second including 0.1 second transmit to receiver and 0.9 second for receiver process including receiving data and Arduino process speed. The result for accelerometer and flex sensor pairing together in this development of hand gesture glove to control robot arm. The purpose is to sharing control system. Accelerometer sensor control the robotic arm left and right movement, initial condition, start to pick action, and after pick action. Flex sensor control the robotic arm pick object action. Finally, the result for this development project completely successful using human gesture to control robotic arm.

The discussion to improve this development project to become more smooth movement by using some algorithm system. The algorithm using in this development project is determine and calculate the frame and distance in between frame to frame in this robotic arm and determine manipulator kinematics including forward and inverse kinematics. This algorithm is using to develop the project even though this robotic arm cannot control similarly like actual human gesture movement but this robotic arm can control by capture the human gesture pattern movement and the controller read the human gesture provided and last the controller will give the instruction for robotic arm to move and action. The overall discussion for this development project can be improve by using more sensor to control other movement pattern to complex other user requirement.

CHAPTER 6

CONCLUSION

This chapter including summary, conclusion, recommendation, and project potential. This summary part is explaining the summary of introduction and method development of this project. The conclusion part is to conclude and finalise the development of this project. The recommendation part is to recommending this development project can be improving by using other method to make this project better. The potential part is to elaborate the further innovation and creative to improvable or implementation on other project.

6.0 Summary

This development human gesture glove to control robot arm is using the human gesture movement capture and control by wearing a wearable glove to communicate with robotic arm and ask the robotic arm to follow the instruction set by using programming language when the signal, data and human gesture provided by human. The process to develop this human gesture glove is including some component, where is accelerometer and flex sensor, transmit and receive Bluetooth communication system, and supported by Arduino UNO processor. Accelerometer sensor is a main active component in this human gesture glove development. This accelerometer sensor using is to control the basic robotic arm movement by x-axis, y-axis and z-axis. Because this accelerometer sensor can be providing different 3-axis acceleration data when the sensor moves or rotate by angle. Flex sensor is a second active component in this glove control system. The flex sensor can be providing the linear data after the flex sensor is banding. Using this linear data provided and covert to linear angle by using formula in Arduino IDE. The Bluetooth communication system is a type of wireless system in between control device and controlled device. The advantage by using this wireless controller system is to improve overall control system more freedom to control in between the human and robotic arm compare with the wire control system.

6.1 Conclusion

This development project finally achieves the objective, that is successfully hander and apply the basic concept of hand gesture glove to control robot arm with wireless interface and overall the connection of component using is develop in one wearable human glove. Last, after finished project development, also successfully to analysis the performance control system is achieve 1 second control speed in between controller and controlled device, to replacing other devices controller in all market.

6.2 Recommendation

In order to further study, the effectiveness of this development project can be improve by using more control system to control robotic arm movement pattern and improve the distance communication system in between controller device and controlled device. Increasing the performance of control system using accelerometer sensor to control robotic arm and also pairing with other advance sensor to develop the robotic arm movement and orientation action simultaneously with the human hand gesture movement and orientation action. Last, can improve the stability robotic arm movement and orientation action.

6.3 Project Potential of Commercialization

This project potential of commercialization for development human gesture glove to control robotic arm including use for industries, become consumer products and further improvable or implementation on other product. This human gesture glove controller control system including some advantage focus for industries worker that will helping many workers to decrease the work stress and this product can bring out fun and interesting to every uses or worker. This product can become consumer products in all market. For further this product can improvable or implementation on other product according to the social technology change in this world.

REFERENCES

[1] Shamsheer Verma, "Hand Gestures Remote Controlled Robotic Arm," Advance in Electronic and Electric Engineering., vol. 3, no. 2231-1297, pp. 601-606, 2013.

[2] SwarnaPrabha Jena, Sworaj Kumar Nayak, Saroj Kumar Sahoo, Sibu Ranjan Sahoo, Saraswata Dash, Sunil Kumar Sahoo, "Accelerometer Based Gesture Controlled Robot Using Arduino," International Journal Of Engineering Sciences & Research Technology, no. 2277-9655, pp. 469-475, April 2015.

[3] A. Dekate; A. Kamal; K. S. Surekha, "Magic Glove-Wireless Hand Gesture Hardware Controller," Electronics and Communication System (ICECS), 2014 International Conference on Coimbatore, pp. 1-4, 13-14 Fed 2014.

[4] Rithesh M Nanda, Harshini H K, Praveen Kuruvadi, Ankhit B V, C Gururaj, "Wireless Gesture Controlled Systems," IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) e-ISSN: 2278-1676, p-ISSN: 2320-3331, vol. 9, no. 6 Ver. IV, pp. 32-39, Nov-Dec 2014.

[5] P. Neto, J. N. Pires and A. P. Moreira, "Accelerometer-Based Control of an Industrial Robotic Arm," Robot and Human Interactive Communication, 209. RO-MAN 2009. The 18th IEEE Internation Symposium on , pp. 1192-1197, 27-2 Sept-Oct 2009.

[6] AAKASH K. SANCHETI, "Gesture Actuated Robotic Arm," International Journal of Scientific and Research Publications, ISSN: 2250-3153, vol. 2, no. 12, December 2012.

[7] Love Aggarwal, Varnika Gaur, Puneet Verma, "Design and Implementation of a Wireless Gesture Controlled Robotics Arm with Vision," International Journal of Computer Application (0975-8887), vol. 79, pp. 39-43, 13 October 2013.

[8] O. Sidek; M. Abdul Hadi, "Wireless Gesture Recognition System using MEMS Accelerometer," International Symposium on Technology Management and Emerging Technologies (ISTMET), 2014 International Symposium on, pp. 444-447, 27-29 May 2014.

[9] Dr.Shaik Meeravali, M. Aparna, "Design and Development of a Hand-glove Controlled Wheel Chair Based on MEMS," International Journal of Engineering Trends and Technology (IJETI), ISSN: 2231-5381, vol. 4, no. 8, pp. 3706-3712, August 2013.

[10] Nghia X. Tran, Hoa Phan, Vince V. Dinh, Jeffrey Ellen, Bryan Berg, Jason Lum, Eldridge Alcantara, Mike Bruch, Marion G. Ceruti, Charles Kao, Daniel Garcia, Sunny Fugate, and LorRaine Duffy, "Wireless Data Glove for Gesture-Based Robotic Control," J.A.Jacko (Ed): Human-Computer Interaction, Part II, HCII 2009, LNCS 5611, pp. 271-280, July 2009.

[11] MUHAMMAD AHKAM KHAN, MUHAMMAD WAQAR, "Wireless Gesture Controlled Robot (FYP Report)," Education, Technology, Business, June 2013.

[12] J. H. Kim ; N. D. Thang ; T. S. Kim, "3-D Hand Motion Tracking and Gesture Recognition Using a Data Glove," IEEE International Symposium on Industrial Electronics (ISIE 2009), pp. 1013-1018, 5-8 July 2009.

[13] W. G. Hao; Y. Y. Leck; L. C. Hun, "6-DOF PC-BASED ROBOTIC ARM (PC-ROBOARM) WITH EFFICIENT TRAJECTORY PLANNING AND SPEED CONTROL," 2011 4th International Conference on Mechatronics (ICOM), pp. 1-7, 17-19 May 2011.

[14] A. I. Bhuyan; T. C. Mallick, "Gyro-Accelerometer based control of a robotic Arm using AVR Microcontroller," The 9th International Forum on Strategic Technology (IFOST), pp. 409-413, 21-23 Oct 2014.

[15] J. K. Perng ; B. Fisher ; S. Hollar ; K. S. J. Pister, "Acceleration sensing glove (ASG)," The 3th International Symposium on Wearable Computer, 1999. Digest of Papers., pp. 178-180, 18-19 Oct 1999.

[16] K.Brahmani, K.S.Roy, Mahaboob Ali, "Arm 7 Based Robotic Arm COntrol By Electronic Gesture Recognition Unit Using Mems," Internatioanal Journal of Engineering Trends and Technology (IJETT), ISSN: 2231-5381, vol. 4, no. 4, pp. 1245-1248, April 2013.

[17] YasserF. O. MOHAMMAD, Toyoaki NISHIDA, "Interactive Perception for Amplification of Intended Behavior in Complex Noisy Environments".

[18] Rini Akmeliawati, Faez S. Ba Tis, Umar J. Wani, "Design and Development of a Hand-glove Controlled Wheel Chair," 2011 4th International Conference on Mechatronics (ICOM),, 17-19 May 2011.

[19] Gaurav Gautam, Abhijeet Ashish, Anil Kumar, Avdesh, "Wirelessly Hand Glove Operated Robot," International Journal of Advanced Research in Electronics and Communaication Engineering (IJARECE), ISSN: 2278-909X, vol. 3, no. 11, pp. 1545-1547, November 2014.

[20] Harish Kumar Kaura, Vipul Honrao, Sayali Patil, Pravish Shetty, "Gesture Controlled Robot using Image Processing," International Journal of Advanced Research in Artificial Intelligence (IJARAI), vol. 2, pp. 69-77, 2013.

[21] Shruthi B.N, Shivraj, Sumathi S, "Handgesture Based Direction Control of Robocar using Arduino Microcontroller," International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, vol. 3, no. 3, pp. 33-35, July 2014.

[22] Aswath S, Chinmaya Krishna Tilak, Amal Suresh, and Ganesha Udupa, "Human Gesture Recognition for Real-Time Control of Humanoid Robot," Int'l Journal of Advances in Mechanical & Automobile Engg. (IJAMAE) ISSN 2349-1485 EISSN 2349-1493, vol. 1, no. 1, pp. 96-100, 2014.

[23] Anand Kumar Kirori Prof. Rajeshwar Lal Dua, "Review of Control Mechanism of Multi-fingered Robotic Arm and Proposal of New Design," IOSR Journal of Engineering ISSN: 2250-3021, vol. 2(5), pp. 1251-1254, May 2012.

[24] C. V. Vishal Ramaswamy, S. Angel Deborah, "A Survey of Robotic Hand-Arm Systems," International Journal of Computer Applications (0975-8887), vol. 109, pp. 26-31, 8 January 2015.

[25] Yoon Sang Kim, Byung Seok Soh, Sang-Goog Lee, "A New Wearable Input Device: SCURRY," IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, vol. 52, pp. 1490-1499, DECEMBER 2005.

[26] Aswath S, Chinmaya Krishna Tilak, Abhay Sengar, Ganesha Udupa, "Design and Development of Mobile Operated Control System for Humanoid Robot," Advances in Computing, vol. 3 (3), pp. 50-56, 2013.

[27] Bret Comnes & A. La Rosa, "Arduino PID Example Lab," Portland State University is a public, nonprofit, coeducational research university located in the southwest University District of downtown Portland, Oregon, United States., 2013.

[28] Nathan Delson, "PID Control: A brief introduction and guide, using Arduino," MAELabs UCSD Jacods School of Engineering, pp. 1-8, 26 sep 2011.

APPENDICES

Appendix A – Project Planning Gantt Chart

Appendix B – Robotic Arm

Appendix C – Circuit Connection Sensor with Arduino Microcontroller

Appendix D – Programming Coding Orientation, Movement and Gripper

Appendix E – Advance Programming Coding Human Gesture Controller Device (TX)

Appendix F – Advance Programming Coding Controlled Device Robotic Arm (RX)

APPENDIX A

PROJECT PLANNING GANTT CHART

A. PROJECT PLANNING

List down the main activity for the project proposal. State the time frame needed for each activity.

Project Activity	2016																										2017											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. Topic selection	BP Briefing	■						Mid Term Break								Study Week	Final Semester Examination		Semester Break									Mid Term Break							Study Week	Final Semester Examination		
2. Internet and library research		■	■																																			
3. Preparation of proposal			■	■																																		
4. Submission of proposal				■																																		
5. Preparation for progress report					■	■	■		■																													
6. Software simulation							■		■	■																												
7. Laboratory testing										■	■																											
8. Data analysis										■	■																											
9. Edit progress report											■	■	■																									
10. Preparation for presentation												■																										
11. Presentation project													■																									
12. Submission of progress report														■																								
13. Preparation material														■							■																	
14. Contruct circuit and hardware																					■	■	■															
15. Testing circuit																							■	■														
16. Hardware combination																								■	■	■												
17. Software& hardware combination																									■	■			■	■								
18. Result evaluation																													■	■	■	■						

APPENDIX B

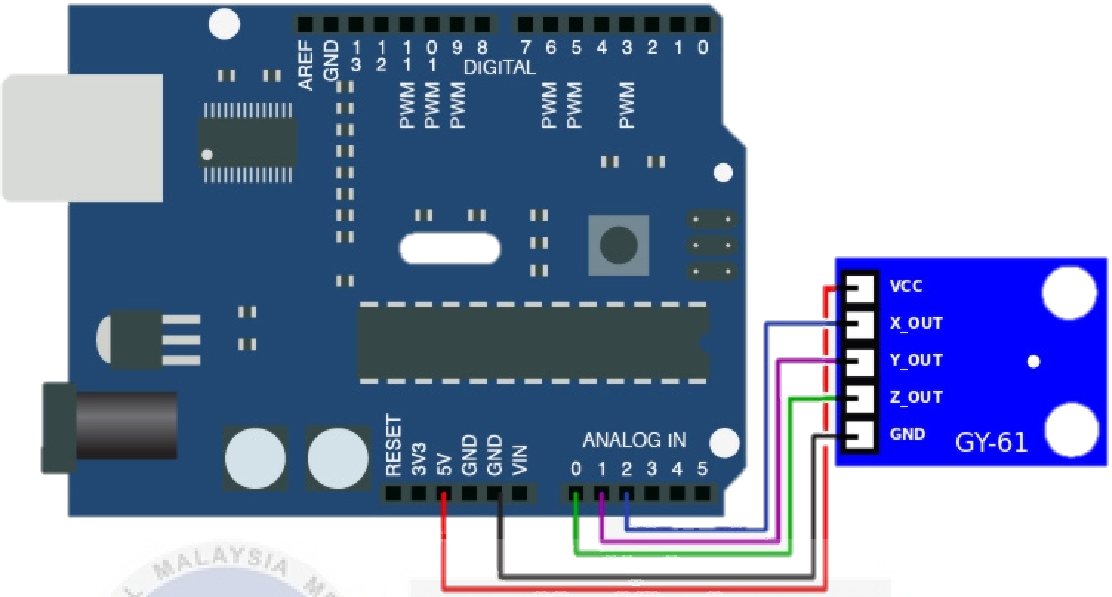
ROBOTIC ARM


Dimensions and Specs
Shoulder to elbow: 3.75”
Elbow to wrist 4.25”
Wrist to tip of gripper: 3.375”
Height (arm parked): approx. 6.00”
Height (reaching up): approx. 14.00”
Median forward reach: approx. 5.75”
Gripper opening: 1.25”
Alternate gripper opening: 1.875”
Lift weight (arm extended): approx. 113.398 grams
Weight: 56.699 grams
Range of motion per axis: 180º

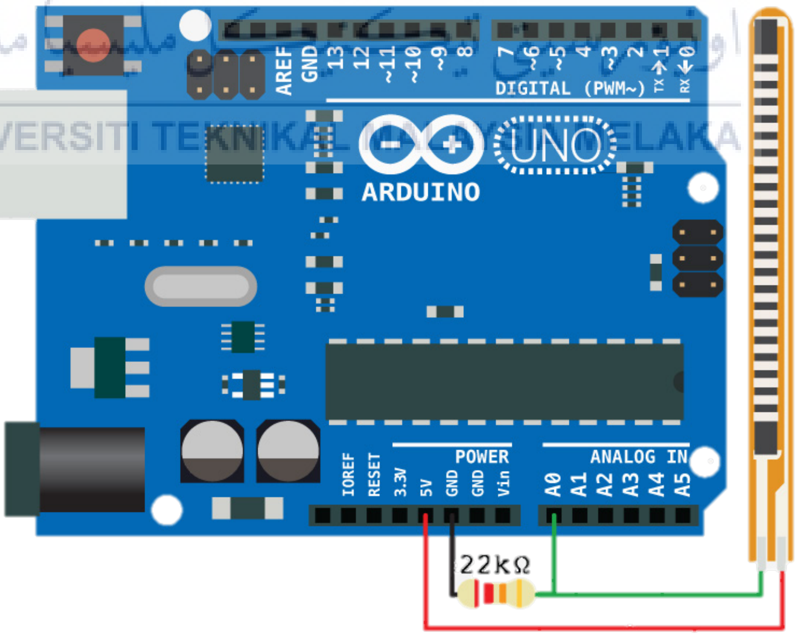
APPENDIX C

CIRCUIT CONNECTION SENSOR WITH ARDUINO MICROCONTROLLER

Circuit Connection for Accelerometer connected with Arduino

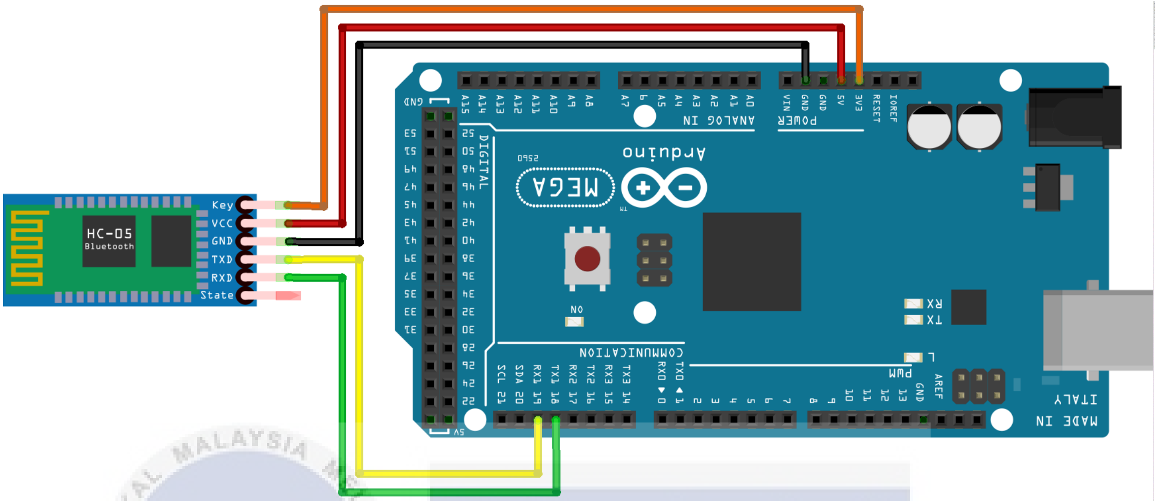


Circuit Connection for Flex Sensor connected with Arduino

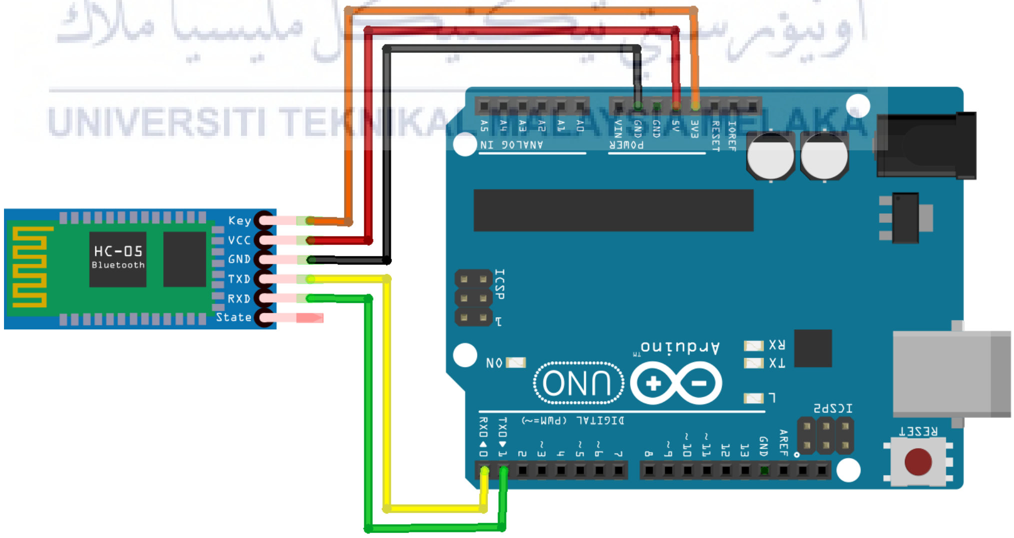


Circuit Connection for Bluetooth connected with Arduino

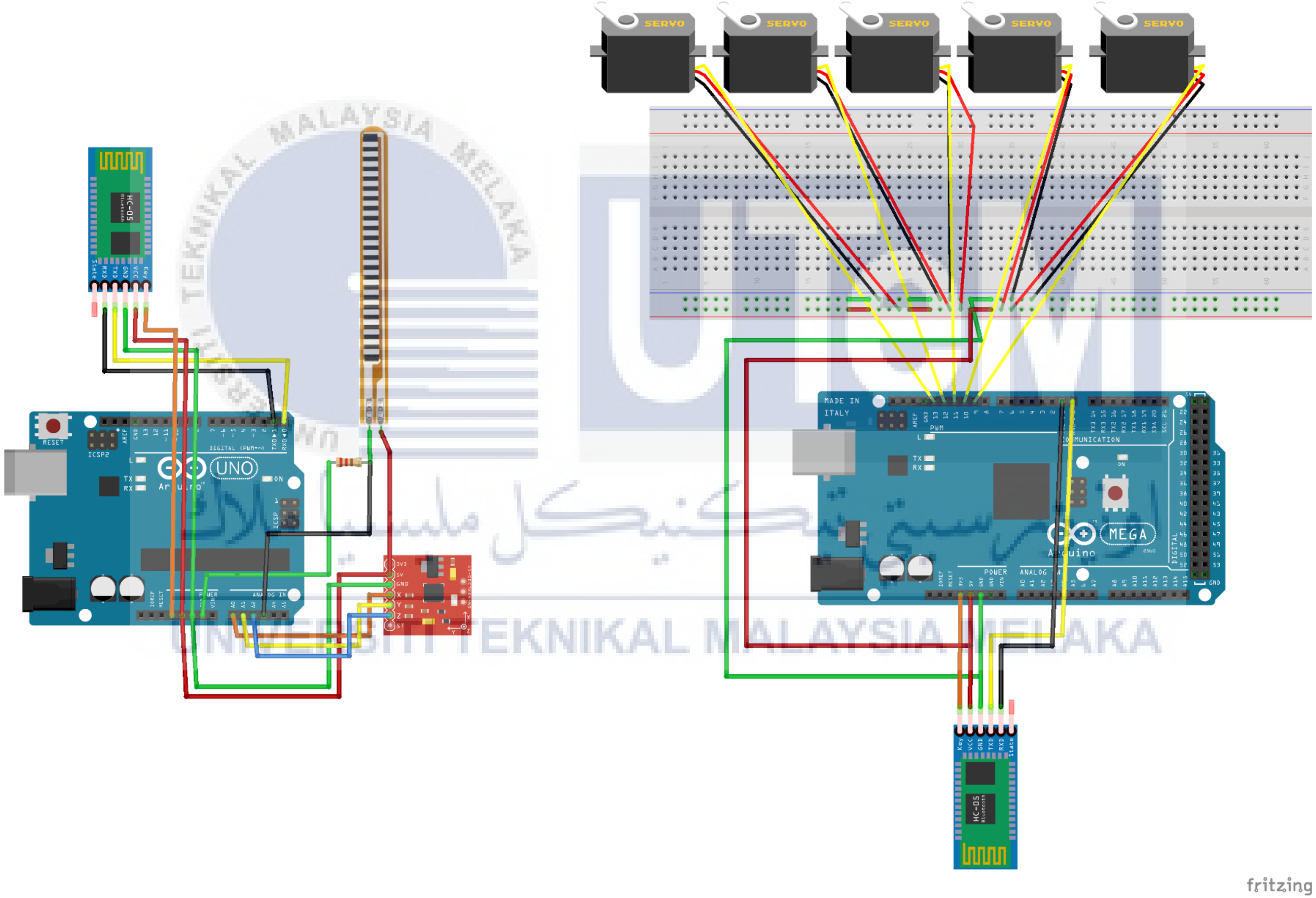
Received Channel for Control Robotic Arm



Transmitted Channel for Hand Gesture Glove Controller



Circuit Connection for Combination All Sensor with Arduino



APPENDIX D

PROGRAMMING CODING ORIENTATION, MOVEMENT AND GRIPPER

```
//servo motor
#include <Servo.h>

//Analog read pins
const int xPin = 0;
const int yPin = 1;
const int zPin = 2;
const int gPin = 3;

//The minimum and maximum values that came from
//the accelerometer while standing still
//You very well may need to change these
int minVal = 265;
int maxVal = 402;


//to hold the caculated values
double x,y,z;
double mx,my,mz;
double ix,iy,iz;

// create servo object to control servo
Servo s1,s2,s3,s4;

//variable to store the servo position
int pos1 = 0;
int pos2 = 0;
int pos3 = 0;

void setup(){
  Serial.begin(9600);
  s1.attach(13); //orientation
  s2.attach(12); //position movement
  s3.attach(11); //position movement2
  s4.attach(10); // gripper
}


void loop(){

  //read the analog values from the accelerometer
  int xRead = analogRead(xPin);
  int yRead = analogRead(yPin);
  int zRead = analogRead(zPin);
```

```
  int gRead = analogRead(gPin);

  //convert analog value to digital
  mx = (xRead*5.0/1023-1.65)/0.4785;
  my = (yRead*5.0/1023-1.65)/0.4785;
  mz = (zRead*5.0/1023-1.65)/0.4785;

  ix = xRead;
  iy = yRead;
  iz = zRead;

  // conver to 255
  ix = map (ix,0, 1023, 0, 180);
  iy = map (iy,0, 1023, 0, 180);
  iz = map (iz,0, 1023, 0, 180);

  //convert read values to degrees -90 to 90 - Needed for
atan2
  int xAng = map(xRead, minVal, maxVal, -90, 90);
  int yAng = map(yRead, minVal, maxVal, -90, 90);
  int zAng = map(zRead, minVal, maxVal, -90, 90);

  //Caculate 360deg values like so: atan2(-yAng, -zAng)
  //atan2 outputs the value of -π to π (radians)
  //We are then converting the radians to degrees
  x = RAD_TO_DEG * (atan2(-yAng, -zAng) + PI);
  y = RAD_TO_DEG * (atan2(-xAng, -zAng) + PI);
  z = RAD_TO_DEG * (atan2(-yAng, -xAng) + PI);

  // write x,y,z degree to pos1,2,3
  pos1 = x;
  pos2 = y;
  pos3 = z;

 /* //Output the caculations
  Serial.print("xRead: ");
  Serial.print(xRead);
  Serial.print(" | yRead: ");
  Serial.print(yRead);
  Serial.print(" | zRead: ");
  Serial.print(zRead);
  Serial.print(" | gRead: ");
  Serial.println(gRead);*/
```

```
  delay(100);//just here to slow down the serial output -
Easier to read

  // output to servo motor orientation
 if(pos1<180 && pos1>0)
  {
  int orien =pos1;
  s1.write(orien);
  }

  /*// output to servo motor movement
  if (yRead>278 && zRead<380)
  {
    for(pos2=180; pos2>=1; pos2-=10)
  {
  s2.write(pos2);
  }
  }
  if(yRead>367 && zRead<388 )
  {
   for(pos2 = 0; pos2<180; pos2+=10)
  {
    s2.write(pos2);
  }
  }*/

 // output to servo motor movement2
 int move = xRead;
 int a = yRead - move;
 double b = a*4;
 /* Serial.print(" | a: ");
  Serial.print(a);
  Serial.print(" | b: ");
  Serial.println(b);*/
  s3.write(b);

 // gripper
  gRead = map(gRead, 5, 21, 0, 255);
  s4.write(gRead);
  /*delay(200);*/
  }
```

APPENDIX E

ADVANCE PROGRAMMING CODING HUMAN GESTURE CONTROLLER DEVICE (TX)

```
/*
 * Human Gesture Controller Device Programming (TX)
 *                  == MASTER CODE ==
 */
int state1, state2, state3, state4 = 0;
int xRead, yRead, zRead, gRead;

void setup() {
  Serial.begin(9600); // Default communication rate of
the Bluetooth module
}

void loop() {
 if(Serial.available() > 0){ // Checks whether data is
comming from the serial port
    //state1 = Serial.read(); // Reads the data from the
serial port
    //state2 = Serial.read();
    //state3 = Serial.read();
    //state4 = Serial.read();
 }

 // Reading the Sensor
 xRead = analogRead(A0);
 yRead = analogRead(A1);
 zRead = analogRead(A2);
 gRead = analogRead(A3);

 // Reading the Sensor
   Serial.print ("| xRead: ");
   Serial.print (xRead);
   Serial.print ("| yRead: ");
   Serial.print (yRead);
   Serial.print ("| zRead: ");
```

```
 Serial.print (zRead);
 Serial.print ("| gRead: ");
 Serial.println (gRead);

if (Serial.read()=='R'){
Serial.write('S');
state1 = Serial.write(xRead); // Sends x
state1 = Serial.write(xRead>>8); // Sends x>>8

state2 = Serial.write(yRead); // Sends y
state2 = Serial.write(yRead>>8); // Sends y>>8

state3 = Serial.write(zRead); // Sends z
state3 = Serial.write(zRead>>8); // Sends z>>8

state4 = Serial.write(gRead); // Sends g
state4 = Serial.write(gRead>>8); // Sends g>>8
Serial.write('E');
delay(100);
}
}
```

APPENDIX F

ADVANCE PROGRAMMING CODING CONTROLLED DEVICE ROBOTIC ARM (RX)

```
/*
 * Controlled Device Robotic Arm Programming (RX)
 *                  == SLAVE CODE ==
 */

#include <Servo.h>

int state1, state2, state3, state4 = 0;
int xRead, yRead, zRead, gRead = 0;

//The minimum and maximum values that came from
//the accelerometer while standing still
//You very well may need to change these
int minVal = 270;
int maxVal = 400;

//to hold the caculated values
double x,y,z;
double mx,my,mz;
double ix,iy,iz;
double offsetX, offsetY, offsetZ;
double gx_z, gy_z, gz_z;
double gx_x, gy_x, gz_x;
double gx_y, gy_y, gz_y;
double gainX, gainY, gainZ;
double accelX, accelY, accelZ;
double VoltsRx,VoltsRy,VoltsRz;
double DeltaVoltsRx,DeltaVoltsRy,DeltaVoltsRz;
double Resultant;

double velocityX, velocityY, velocityZ = 0;
double positionX, positionY, positionZ = 0;
double currAccelX,currAccelY,currAccelZ;
double prevAccelX, prevAccelY, prevAccelZ = 0;
```

```
double currVelX,currVelY,currVelZ;
double prevVelX, prevVelY, prevVelZ = 0;
double Rvelocity = 0;
unsigned long prevtime = millis();
unsigned long currtime;

double velocityx, velocityy;
double pvelocityx, pvelocityy = 0;
double positionx, positiony;
double ppositionx, ppositiony = 0;

// create servo object to control servo
Servo s1,s2,s3,s4,s5,s6;

//variable to store the servo position
int pos1 = 0;
int pos2 = 0;
int pos3 = 0;
int AVGpos23 = 0;

// counter time second
int countx;
int county;
int countyAng1;
int countyAng2;
int countzAng;
int countxAng;
int countxAng2;
int gripperO;
int gripperC;

void setup() {
  Serial.begin(9600); // Default communication rate of
the Bluetooth module
```

```
  s1.attach(13); // base motor
  s2.attach(12); // shoulder motor
  s3.attach(11); // elbow motor
  s4.attach(10); // wrist motor
  s5.attach(9);  // gripper head motor
  s6.attach(8);  // gripper motor
}

void serialFlush(){
  while(Serial.available() > 0) {
    char t = Serial.read();
  }
}

void loop() {

 delay(500);
 serialFlush();
 Serial.write('R');
 delay(250);

 if(Serial.available() ==10){ // Checks whether data is
comming from the serial port
    if (Serial.read()== 'S'){
    state1 = Serial.read(); // Reads the data from the
serial port
    state1 = state1+Serial.read()*256;
    state2 = Serial.read();
    state2 = state2+Serial.read()*256;
    state3 = Serial.read();
    state3 = state3+Serial.read()*256;
    state4 = Serial.read();
    state4 = state4+Serial.read()*256;
    Serial.read();
```

```
    //convert back state to Read
    xRead = state1;
    yRead = state2;
    zRead = state3;
    gRead = state4;

   Serial.print("| xRead: ");
   Serial.print(xRead);
   Serial.print("| yRead: ");
   Serial.print(yRead);
   Serial.print("| zRead: ");
   Serial.print(zRead);
   Serial.print("| gRead: ");
   Serial.print(gRead);

  //convert read values to degrees -90 to 90 - Needed for
atan2
  int xAng = map(xRead, minVal, maxVal, -90, 90);
  int yAng = map(yRead, minVal, maxVal, -90, 90);
  int zAng = map(zRead, minVal, maxVal, -90, 90);

  Serial.print("| xAng: ");
  Serial.print(xAng);
  Serial.print("| yAng: ");
  Serial.print(yAng);
  Serial.print("| zAng: ");
  Serial.println(zAng);

if (yAng>70)
{
  countyAng2++;
}
else
```

```
 {
  countyAng2 = 0;
}
if (countyAng2>=1)
{
if (yAng>70)
{
  s1.write(87); //base
  s2.write(180);//shoulder
  s3.write(180);//elbow
  s4.write(180);//wrist
  s5.write(90);//gripper head
 Serial.println(" Robot Arm Initial Condition ");
  }
 }

if (yAng<-90)
{
countyAng1++;
}
else
 {
countyAng1 = 0;
}
if (countyAng1>=1)
{
if (yAng<-90)
{

   s2.write(50);//shoulder
   s3.write(90);//elbow
   s4.write(0);//wrist
   s5.write(90);//gripper head
   Serial.println(" Robot Arm Start to Pick ");
```

```
    }
}

  if ((zAng>-90) && (zAng<-50) && (yAng>-80)&&(yAng<-20))
{
  countzAng++;
}
else
 {
  countzAng = 0;
}
if (countzAng>=2)
{
 if ((zAng>-90) && (zAng<-50) && (yAng>-80)&&(yAng<-20))
{
  s2.write(70);//shoulder
  s3.write(100);//elbow
  s4.write(0);//wrist
  s5.write(90);//gripper head
  Serial.println (" Robot Arm Completly Pick Up ");
}
}

if ((xAng<-90)&&(xAng>-100)&&(yAng<0)&&(zAng<50))
{
  countxAng++;
}
else
 {
  countxAng = 0;
}
if (countxAng>=1){
if ((xAng<-90)&&(xAng>-100)&&(yAng<0)&&(zAng<50))
{
```

```
s1.write(135); // base
Serial.println (" Robot Arm Base Turn left ");
}
}

if ((xAng>20)&&(yAng<30)&&(zAng<0))
{
countxAng2++;
}
else
 {
countxAng2 = 0;
}
if (countxAng2>=2)
{
if ((xAng>20)&&(yAng<30)&&(zAng<0))
{
s1.write(45); // base
Serial.println (" Robot Arm Base Turn RIght ");
}
}

 /*Serial.print("| gRead: ");
 Serial.println(gRead);*/

if (gRead<16 && gRead>7)
{
gripperO++;
}
else
 {
gripperO = 0;
}
if (gripperO>=1)
```

```
{
  if (gRead<16 && gRead>10)
  {
  s6.write(90);
 Serial.println(" Robot Gripper Open ");
  }
}

  if (gRead>0 && gRead<6)
{
gripperC++;
}
else
 {
gripperC = 0;
}
if (gripperC>=1)
{
  if (gRead>0 && gRead<6)
  {
  s6.write(180);
  Serial.println(" Robot Gripper Close");
  }
}
  }
 }
}
```