"I hereby declare that I have read through this report entitle "Leap Motion – Controlled robotic arm" and found that it has comply the partial fulfillment for awarding the degree of Bachelor of Electrical Engineering (Mechatronic Engineering)"

Signature	:	....
Supervisor's Name	:	.... LOI WEI SEN ....
Date	:	....

LEAP MOTION – CONTROLLED ROBOTIC ARM

LEE JUN WEI

A report submitted in partial fulfillment of the requirements for the Degree of Mechatronic Engineering

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2014

I declare that this report entitle “Leap Motion – Controlled robotic arm” is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	....
Name	:	.... LEE JUN WEI ....
Date	:	....

To my beloved Mother and Father

Thank you for your incessant support and encouragement. Your sacrifices and loves have helped me to achieve this accomplishment.

Dear Lecturers and Supervisors

Thank you for your continuous support, knowledge and guidance.

Dear Friends

Thank you for all the information, guidance, support and encouragement.

ACKNOWLEDGEMENT

First and foremost, I would like to take this chance to thank my beloved supervisor Mr. Loi Wei Sen for giving the support mentally and physically, by sharing his expertise, knowledge and experience with me. He has always provides me with guidance on how my project should work and gives me new idea to be implemented, as well as solutions for some difficult problems. Never forget the good advices I got from Mr. Lim Wee Teck, Ms. Nur Maisarah binti Mohd Sobran and Ms. Arfah binti Ahmad to do a better report. Nevertheless, as my first panel, Dr. Ahmad Zaki has guided me about my hardware as Dr. Zaki is major in robotics field.

I am highly indebted to pioneers, Tham Yu Jiang and Ali Mutasim who have provided me with the valuable guidance and knowledge which very friendly to share although distance is not allow to meet, and I appreciated it with all of my heart.

Last but not least, a million thanks to all those time, concern and efforts that were given to me during the whole process of completing this project and report. I am thankful to everyone who always inspires me directly and indirectly during my Final Year Project.

ABSTRACT

The robotic arm structure is designed for a purpose, to pick and place an object, controlled by a gesture control mechanism called Leap Motion controller. This is to expose the use of the new way as a controller to control a device or hardware. Furthermore, this is applicable for those tasks that unreachable by human. The robotic arm structure of pick and place is controlled by Arduino as microcontroller to control the angles and displacements of the servo motor precisely. The position and orientation of the fingers and hands send to the Arduino through command prompt. Next, the programming is done by JavaScript language to communicate between Arduino and the Leap Motion controller. Specifically, a detailed 3D drawing is drawn by using SolidWorks and the dimensions for each part drawn needed for the fabrication. The robotic arm hardware then assembled all together. After the platform is done, kinematic and inverse kinematic equation and calculations are programed into JavaScript language for the robotic arm mechanism. The robotic arm followed the hand gesture with the fingers. The position and orientation and the finger are be the same as the robotic arm. Robotic arm then integrate with Leap Motion for pick and place purpose. The movement and accuracy are improved from the experiments and some calculations after the data collected from the experiment. Another test is given to the randomly selected participants to control the robotic arm with a task given. This is to record the participants understanding on the first try on controlling the robotic arm with their hand gestures. Few experiments are carried out to increase the accuracy analyze the precision while improve the overall pick and place performance.

ABSTRAK

Struktur lengan robot direka untuk tujuan, untuk memilih dan meletakkan objek, yang dikawal oleh satu mekanisme kawalan isyarat dipanggil pengawal Leap Motion. Ini adalah untuk memberi pendedahan kepada penggunaan cara baru sebagai pengawal untuk mengawal alat atau perkakasan. Tambahan pula, ini adalah terpakai bagi mereka tugas-tugas yang tidak dapat dicapai oleh manusia. Struktur lengan robot pick dan tempat dikawal oleh Arduino sebagai pengawal mikro untuk mengawal sudut dan anjakan motor servo itu dengan tepat. Kedudukan dan orientasi jari dan tangan hantar ke Arduino melalui arahan segera. Seterusnya, pengaturcaraan dilakukan dengan bahasa JavaScript untuk berkomunikasi antara Arduino dan pengawal Leap Motion. Secara khusus, lukisan 3D terperinci dilukis dengan menggunakan ToT dan dimensi untuk setiap bahagian disediakan diperlukan untuk fabrikasi. Perkakasan lengan robot kemudian dipasang semua bersama-sama. Selepas platform itu dilakukan, persamaan kinematik kinematik dan songsang dan pengiraan diprogramkan ke dalam bahasa JavaScript mekanisme lengan robot ini. Cabang robot diikuti isyarat tangan dengan jari. Kedudukan dan orientasi dan jari adalah sama dengan lengan robot ini. Lengan robot kemudian mengintegrasikan dengan Leap Motion untuk tujuan memilih dan tempat. Pergerakan dan ketepatan dipertingkatkan dari eksperimen dan beberapa pengiraan selepas data yang diperoleh daripada eksperimen. Ujian lagi diberikan kepada peserta yang dipilih secara rawak untuk mengawal lengan robot ini dengan tugas yang diberikan. Ini adalah untuk merekodkan peserta memahami pada percubaan pertama pada mengawal lengan robot dengan isyarat tangan mereka. Beberapa eksperimen yang dijalankan untuk meningkatkan ketepatan menganalisis ketepatan manakala meningkatkan memilih dan tempat prestasi keseluruhan.

TABLE OF CONTENTS

CHAPTER **TITLE** **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

x	-	coordinate point of end effector of x-axis
y	-	coordinate point of end effector of y-axis
z	-	coordinate point of end effector of z-axis
θ_i	-	the angle from X_{i-1} to X_i measured about Z_i

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Numerous robots have been developed and used in factories, plants and hazardous environments. They supported human workers and significantly reducing the risk of accidents. In the future, it is expected that robots expanded their work space not only to produce and hazardous environments, but also for the home and office environments to support their daily activities [1]. By combining the advantages of human perception and recognition skills with consistent and accuracy robots, a human-robot collaboration system can enhance target identification rate and reduce the complexity of robotic systems [2, 3].

In many industries, the machines are controlled by complicated control panel, many buttons, joysticks or touchscreen panel. For example of the ABB teach pendant, KUKA teach pendent or even Samsung FARA teach pendent, complicated buttons that confused the operator such as in Figure 1.1 , Figure 1.2, and Figure 1.3 respectively. This Leap Motion controller is a new future to our way of controlling things and the upcoming field that this controller is applicable is in computer technology. ASUS and HP laptops have been applied this controller to their product. In the news, the ASUS will be start shipping their laptop internationally with built in the Leap motion controller. While in October, Leap took a baby step with the HP envy 17 Leap Motion SE. It is the first laptop with embedded Leap Motion controller, and it used a new module that is 70 percent thinner than the on inside Leap's standalone sensor.

Figure 1.1: ABB Teach Pendant

Figure 1.2: KUKA Teach Pendant

Figure 1.3: FARA Teach Pendant

However, there are one device that has movement, voice and gesture recognition that hit the market last year 2010, called Kinect. It is released by Microsoft and is applicable for Windows and Xbox for gaming purposes. However, the price for a piece of Kinect is RM 465 internet price. In contrast, the Leap Motion controller is just cost RM 250 including shipment fee. Moreover, it is easy to use by just using the hand gesture rather than Kinect sensed the body movement to control. Thus, it is cheap and very convenient for the developers to explore. In conclusion, there is a huge potential for developers to expand this future to other audience who are excited to discover the potential of this Leap Motion controller created by the developers. As saying goes, “get your application into millions of hands or even more fingers”.

This project is to design hardware that able to control by Leap Motion controller. The main purpose is to expose the new technology that able to replace the conventional controller. When it comes to a controller for a remote car or a computer game, think of controller with buttons, joysticks or even go further to touchscreen panel. But what if a controller does not has those features, it using human movement instinct for human to control the remote car and the computer games. The conventional controller needed human to observe and get used to it before they master the controller. The natural way our hands move determined the output of a device rather than buttons or joysticks. By using the Leap Motion Controller as the controller and with a wave of a hand or lift of a finger, it replaced all those clicks and taps and drags and drops. The Leap Motion Controller sensed our hands and fingers and also followed every move it make. Our hands and fingers are allowed to move in the wide open space between the controller and the hardware.

The hardware for this project is a Leap Motion controlled robotic arm. The uses of robotic arm can be expanded to greater fields such as for the disabilities, in robotic industries, chemical laboratories, and accomplished tasks where human unable to reach. All this fields needed accuracy and precision to do the tasks, and Leap Motion has very fine system, with the servo actuated robotic arm able to perform such tasks given. Controlling a robotic arm using joysticks or buttons might be difficult, but using the natural and understandable movements and gestures, task is easily performed.

Objective of the Leap Motion is used because it is a new technology of controller able to bring the future in the whole new level of controlling. This will further expand well acceptance usage in robotic fields, despite its small in size, precise and it is a revolutionary way to control an object.

1.2 Problem Statement

In the industries, or in the laboratory which needed to handle stuff with care used robotic arm. The most likely issue that operator of the controller found out difficulties in using those control panels, manual book might be as thick as a textbook. Conventional controller also led to performing task slowly as the confusing buttons and joystick which is not synchronize with the way of the operator could imagine. Moreover, this kind of conventional controller need more time to adapt to the use of the buttons and joysticks to able to do task smoothly.

Several approaches have been developed for sensing hand movements and corresponding by controlling robotic hand. Glove-based techniques well known means to recognize hand gestures. Mechanical utilizes glove is attached sensor that directly measure hand and joint angles of the arm and the spatial position. Although gestural interfaces based glove give more accuracy, if limits freedom because it forces users to use the patch cumbersome devices [4].

1.3 Objective

The main objectives of the project are embarked as below:

1. To develop Leap Motion gesture controlled robotic arm hardware.
2. To construct an algorithm using JavaScript to control the robotic arm.
3. To integrate a Leap Motion controlled robotic arm structure for pick and place purpose.
4. To analyze the accuracy and precision of the robotic arm and to improve its accuracy and pick and place performance.

1.4 Scope

The Leap Motion controller is a new technology device that is launched in the mid-year of 2013. To use this device, exploration is needed and learned how to connect to computer. It able to act as a mouse, but the only different is using hand gesture to function like a mouse. An Arduino Uno Rev3 is used as a microcontroller to the robotic arm. Thus the servos are directly connected to the Arduino board. Next, to connect the Leap Motion and the Arduino, both are connected to the same host computer. Appropriate way to connect both Leap Motion controller and Arduino is determined to send data from Leap Motion controller to Arduino. This project covered on the programming in JavaScript language to control the robotic arm by hand gestures sensed by the Leap motion controller. To edit the program, Visual Studio is good software to use. Moreover, this project also constructed the structure of the robotic arm. But before that, a 3D drawing of the robotic arm is drawn by using SolidWorks. The average length of a human arm, which is measured from shoulder to third finger, is approximately 65cm long. Thus the length of robotic arm for this project is shorter than 65cm that is from the base to the gripper of the robotic arm. This project is planned to have a 6 degree of freedoms but the derivation of the formulas take months to get the answer. Thus this robotic arm is limited to maximum 3 degree of freedoms. Another 2 degree of freedom which is for the flexion, extension, supination and pronation wrist movement is included to the robotic arm. This 2 degree of freedoms is independent, thus, no inverse kinematics formulas derivation are needed. Consequently, total of 5 degree of freedoms of robotic arm is built to perform a simple pick and place task. Furthermore, as usual a normal RC servomotor has its error, thus reduce the accuracy.

Therefore, few experiments are carried out to increase the accuracy analyze the precision while improve the overall pick and place performance.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Figure 1.1 above showed an ABB robotic arm controller. The joystick is the controller the movement of the ABB robotic arm. It is a conventional controller in the market now days. While the Figure 2.2 shows a black glove which is another controller to control a device or a machine. The robot controlling is done by Fusion of Hand Positioning and Arm Gestures using data glove [4]. It is necessary to wear the glove, thus limits it freedom although the precision is high. The weakness is that proper light and camera angle are required for capturing hand gesture correctly. An Electromyographic (EMG) signals are a famous way of collect the data from human arm motion by the movement of muscle in the most research [5]. A technique is needed to handle the light source and viewing angle to capture efficiently hand gesture [4]. But despite having all these available assisting tools, improvement for the aids assisting controller are still much needed.

Figure 2.1: Back glove as controller

2.2 Dark Glove with Colored Cues

The name of the robot is CRS A-460 with three-fingered gripper but is a manipulator graphic model simulator. The real time hand gesture able to guide the robotic arm in gripping gestures in the simulator. This robotic arm has six degree of freedom and it is operated with electric DC servo motors. Furthermore, to control the robotic with hand gestures is to wear a dark glove marked with colored clues on the dark glove. For the tracking the colored cues on the dark glove to capture images in sequence in real time, a single camera is used. The limitation is limited the operator’s freedom on need to wear a glove [6].

2.3 Simple Video Camera

3D MATLAB Kinematic model of a PUMA 762 is used for the research. It has six degree of freedom and is operated by electric DC servo motors. The modules on interpretation of hand gestures are in four steps. First are a real time hand gesture formation monitors and gesture capture and then hand feature extraction. Then, pattern matching for gesture capture and last is the command determination corresponding to shown gestures and performing action by robotic system. As computer vision, simple video camera is used to track gestures presentation that has 3 speeds of frames per second. Therefore, real-time technique is used to track the hand when the hand is in the range of vision of the camera. To identify hand, required continuous captured 3 frames to compare the framework to search within which there is movement. 10 different gestures are used for 10 different movements for the robotic arm to gesture commands changed robot specific language to operate the robotic arm. The accuracy is 90% in the right light and low light performance declines. The restriction is placed in the chamber which requires adequate light and camera angle [4].

2.4 A Camera

Camera also has another alternative method to control a robotic arm where consist of four-axis servo motor handmade used for the operation. There are two control modules robotic arm. The first is to compare all the pre-stored data in the database Template Matching Algorithm. Next is to find where the fingertips are and count the number of the same, use the Signature Signal. Both methods have short in the calculation, making it suitable for continuous frame capture sequence. To control the robotic arm gripper, local features used to provide hand gesture while global features hand position is 3D dimension. As for this case, the application of the robot hand control the robotic arm is followed the same path of the human hand. One limitation is that the shoulder position must be configured to the system manually before starting the system.

The experiment is tested under the same conditions light source light is on the left and in the corner at the hands of their fans. This type of source positions always because shadows on the object observed [7].

2.5 Kinect

Virtual robotic arm is used and the task given is to pick a virtual ball and place into a virtual basket. A Kinect sensor is used to track the hand position. The module on controlling the virtual robotic arm is by programming-by-demonstration. It ran VizArtist software with a custom made OSC plugin for accepting Open Sound Control messages and controlling the virtual set. The Kinect sensor tracked the hand position of the operator and delivered the related OSC messages in real-time. Moreover, the implementation of the virtual robotic arm is by spatial mapping to directly control the joint configuration of the robotic arm, based on the user’s left hand position. The limitation for this system is that it involved the operator’s whole body to control [8].

2.6 Infrared Proximity Array

Infrared proximity sensor (IPA) capable of capturing 3D depth information in real time and realized the machine control. In this paper, IPA is used to replace the mouse cursor control in PC. IPA can directly measure the distance on the map and get more consistent results in various lighting conditions using infrared light. Therefore, is capable of measuring the intensity of the reflection in infrared light. There are two modes of application of this sensor applied, which is the control mode fingertips and the palm control mode. To detect the fingertips, the algorithm searches the window process with the shortest distance. If object is not detected in the working range, implying no significant movement and cursor command is fixed. Moreover, the first stage of palm input mode determination is required whether or not the user's hand is suitably in front of the sensor. The sensor checks each distance value for all windows so you can measure the distance of the hand and almost guess the angle

Experiments to measure throughput and completion time are carried out. Throughput is the famous performance index to measure the usability of input devices such as a mouse and ball track. The goal of the experiment is designed to measure throughput and completion time of advanced interfaces and compare its performance with other input devices. Eight students participated in this experiment are in a certain age and no sensory difficulties or problems at hand. The participants sat in front of her and showed advanced 3D sensor with index finger to the input motion. Button targets appear in a random location on the screen and participants control the cursor to move to the target button and managed to click the target at the fastest possible time. 20 random buttons appears on the screen after each successful attempt. Processing of lower than normal mouse, and the completion time is also much larger than mice. This difference is due to the low profitability and the errors in movement transitions between movements and clicks [9].

2.7 Data Glove

In this review, a data glove (called KHU -11 data glove sensors consists of three three-axis accelerometer, a controller and Bluetooth once). The data glove is able to implement a digital hand model for 3D tracking and recognition of hand movement. It used kinematic chain theory utilizing ellipsoids and joints to create the 3D digital hand model.

The index finger, middle finger, ring and little fingers are controlled by an accelerometer fixed on the middle finger of representing the whole hand motion using only three accelerometers. With the use of gloves with sensors connected, there are many advantages to it. First is the motion sensors are not affected by the surroundings. Second is the glove is attached to the hand, so the more coverage and the third are the signals are available wirelessly. However, it is necessary to limit the operator to use the glove to acquire data [10].

2.8 Comparisons

In summary, the similarity is they are using hand gesture and finger position to complete a task virtually and technically. They using camera sensor as a visual aid to sense the hands and fingers and capturing the data of the hands and fingers of the operator. For the information, Kinect sensor also contained camera sensor inside. By wearing black glove or not wearing it, some researchers managed to detect the hands and fingers using the proper software to encrypt the images of the hands and fingers. It can say that by wearing black glove with color and magnetic chips on it, the precision will be increase dramatically to track the fingers and hands. If by using the camera sensor, it can increase the precision and accuracy by using 2 camera sensors as mention from the paper 1 and 3. Yet, the drawback of using camera sensor is that the light intensity is important. Improper light arrangement leads to decrease in performance.

Infrared Proximity Array is a new way of approached to capture hand motion. It not a camera as other journal reviews, but it is better performance in capturing the 3D shape of the hand. Reflection of infrared light is better than light where light performance can be disturbed by other light source. Data glove is another way of not implement camera, but rather a glove with the sensors attached to it. The coverage is considered wide and not in the vision range of the camera which is limited and small. As mentioned, hand detection and recognition by wearing glove has more precision data than imaging processing.

CHAPTER 3

METHODOLOGY

3.1 Introduction

Leap Motion brand new technology just launch in the mid-year of 2013. A research on the use of Leap Motion controller through internet is best way. Before planning for the hardware, software must be built as the first step. Inverse kinematics formulas are derived before building the hardware. Then the formulas translated to JavaScript language so that the computer able to process it and provided the answer according to the calculation of the formulas. Next the dimension of the hardware is planned well and fabricated for the assembly. After that, the related dimension of the robotic arm is included to the formulas.

Controlling a device using human natural movement such as hand gesture, eye movement and human hand muscle is an expanding research field. Using muscle sensors to sense human muscle pulse needed very deep research into it. Moreover, sensing the reflection of eye muscles needed biology knowledge to do so. Consequently, hand gesture is a suitable topic to do research on. However, Kinect or Leap Motion controller? Leap Motion is not just cheaper than Kinect, it also easy to use as it sensed motion of hands and fingers rather than the motion of whole body. In addition, Leap Motion is much smaller than Kinect. Leap Motion is an open source device, where developers have the chance to expand the usage of Leap Motion.

3.2 Experiment Section

As the Leap Motion controller is giving the position of *x*, *y* and *z* and orientation for each fingers when the fingers is on top and in the range of the Leap motion controller. The position of the hand is acted as the tool at the end effector of the robotic arm. Well, the tool is a gripper. Thus the inverse kinematics able to find the required joint angles to place the tool as the tool is the same position and orientation as the hand.

3.2.1 Inverse Kinematics Formulation

In order to obtain the equation for solving the angle of each joint after getting the coordination position, the formulation for the inverse kinematics must derived first according to the designed robotic arm prototype. The limitation range for Leap Motion controller is recorded. While for this experiment, 2 Degree of Freedom is tested to decrease the difficulties of the experiment.

3.2.2 Program the Derived Equations in JavaScript Language using Visual Studio

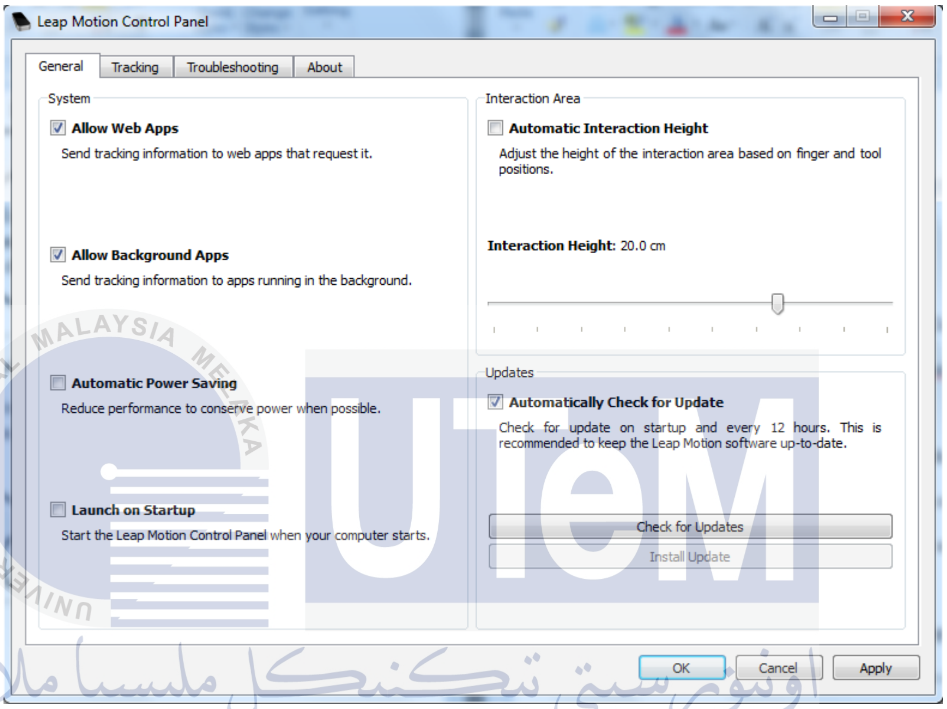
The formulas converted to JavaScript language form for computation.

3.2.3 Make Connection between the Leap Motion Controller and the Robotic Arm

The extracted data of the position and orientation of the hand and fingers is send from Leap Motion controller to the host computer. While host, there will be node.js which is acted as a library to enable the Leap Motion controller to communicate with the Arduino Uno board. To begin the process of pick and place using hand gesture, it needs to be executed using host's command prompt and from there, it is able to check error of the program. If there are errors, the program in the Visual Studio has to be edited until the command prompt able to execute it and the link between the host computer and the Arduino Uno board is connected.

The procedure to make connection from Leap Motion controller to robotic arm is as follow:

1. Leap Motion controller and the Arduino Uno board are plugged to the host computer
2. “Allow Web Apps” or also known as Websocket is checked in the Leap Motion controller settings as shown in Figure 3.1.


Leap Motion Control Panel

General | Tracking | Troubleshooting | About

System

☑ Allow Web Apps
Send tracking information to web apps that request it.

☑ Allow Background Apps
Send tracking information to apps running in the background.

☐ Automatic Power Saving
Reduce performance to conserve power when possible.

☐ Launch on Startup
Start the Leap Motion Control Panel when your computer starts.

Interaction Area

☐ Automatic Interaction Height
Adjust the height of the interaction area based on finger and tool positions.

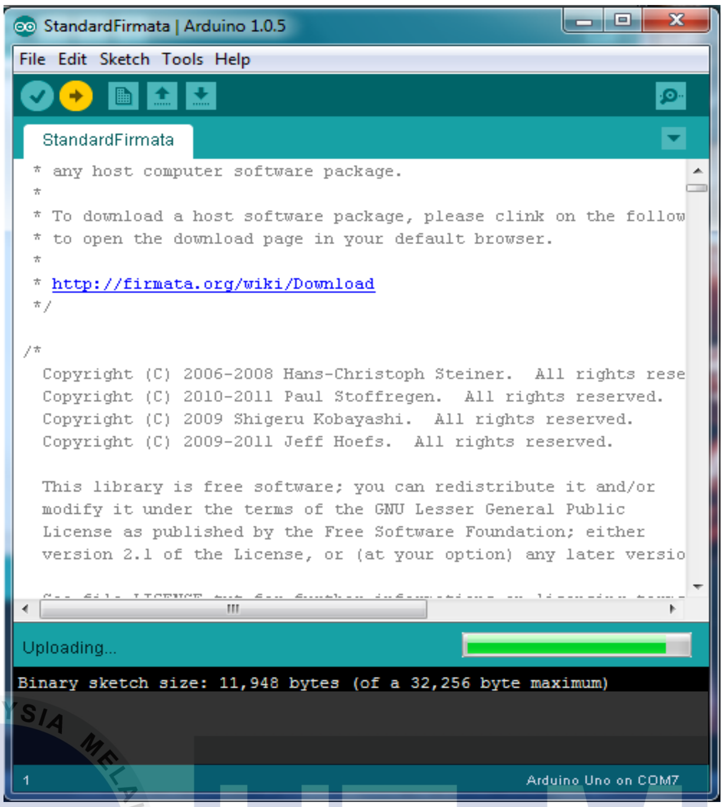
Interaction Height: 20.0 cm

Updates

☑ Automatically Check for Update
Check for update on startup and every 12 hours. This is recommended to keep the Leap Motion software up-to-date.

Figure 3.1: Leap Motion controller’s settings

3. In the Arduino software, the Standard Firmata is opened from the Example of the Arduino menu bar.
4. Standard Firmata is uploaded into the Arduino Uno board using the Arduino software as shown in Figure 3.2.

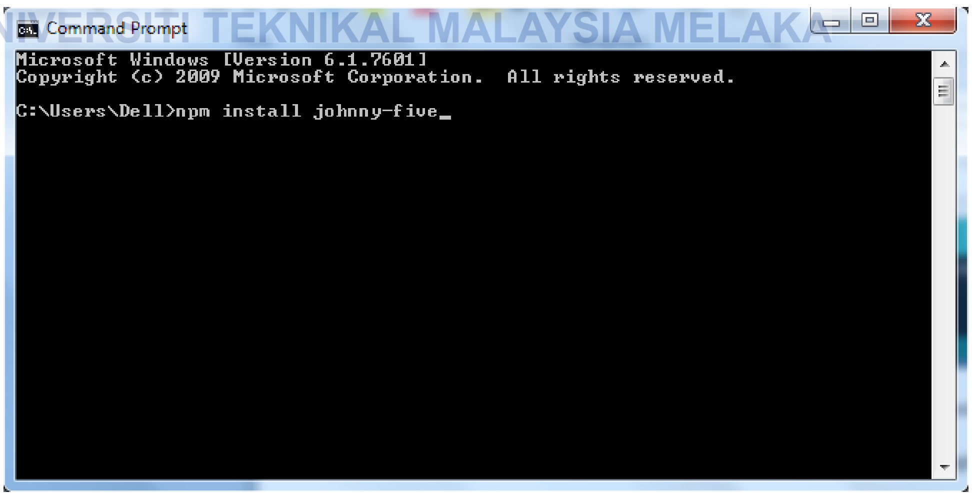


```
 * any host computer software package.
 *
 * To download a host software package, please clink on the follow
 * to open the download page in your default browser.
 *
 * http://firmata.org/wiki/Download
 */

/*
  Copyright (C) 2006-2008 Hans-Christoph Steiner.  All rights rese
  Copyright (C) 2010-2011 Paul Stoffregen.  All rights reserved.
  Copyright (C) 2009 Shigeru Kobayashi.  All rights reserved.
  Copyright (C) 2009-2011 Jeff Hoefs.  All rights reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later versio
```

```
Uploading...
Binary sketch size: 11,948 bytes (of a 32,256 byte maximum)
```

Figure 3.2: Arduino software

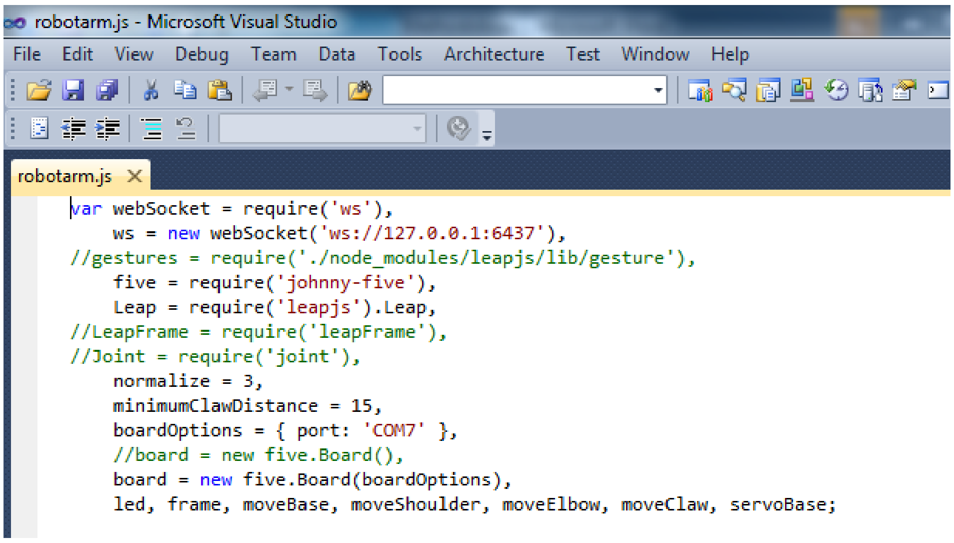
5. Johnny-five and leapjs are installed into a node module file located in the host’s user file using command prompt by typing for an example ‘npm install johnny-five’ and Enter button is hit as shown in Figure 3.3.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Dell>npm install johnny-five_
```

Figure 3.3: Command prompt to install file

6. Johnny-five and leapjs are used in the programming program as shown in Figure 3.4.

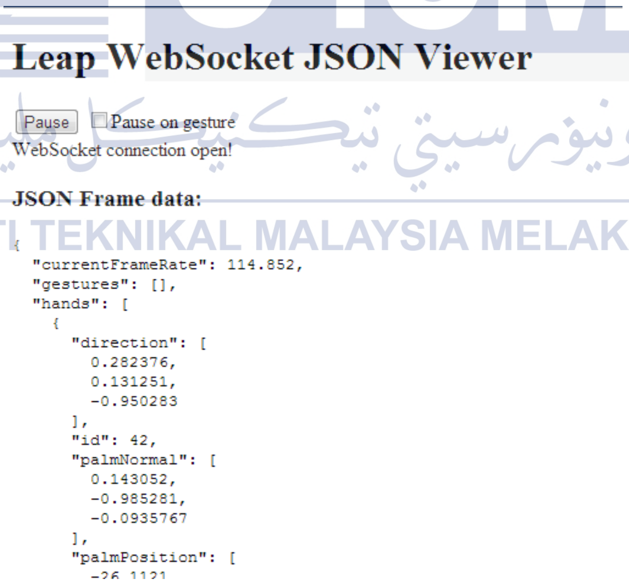

robotarm.js - Microsoft Visual Studio
File Edit View Debug Team Data Tools Architecture Test Window Help
robotarm.js

```
var webSocket = require('ws'),
    ws = new webSocket('ws://127.0.0.1:6437'),
//gestures = require('./node_modules/leapjs/lib/gesture'),
    five = require('johnny-five'),
    Leap = require('leapjs').Leap,
//LeapFrame = require('leapFrame'),
//Joint = require('joint'),
    normalize = 3,
    minimumClawDistance = 15,
    boardOptions = { port: 'COM7' },
    //board = new five.Board(),
    board = new five.Board(boardOptions),
    led, frame, moveBase, moveShoulder, moveElbow, moveClaw, servoBase;
```

Figure 3.4: Visual Studio for programming

7. Port number for the Arduino Uno board is assigned in the programming program to connect it with the host computer. The programming is saved as .js file type in the host's user file.

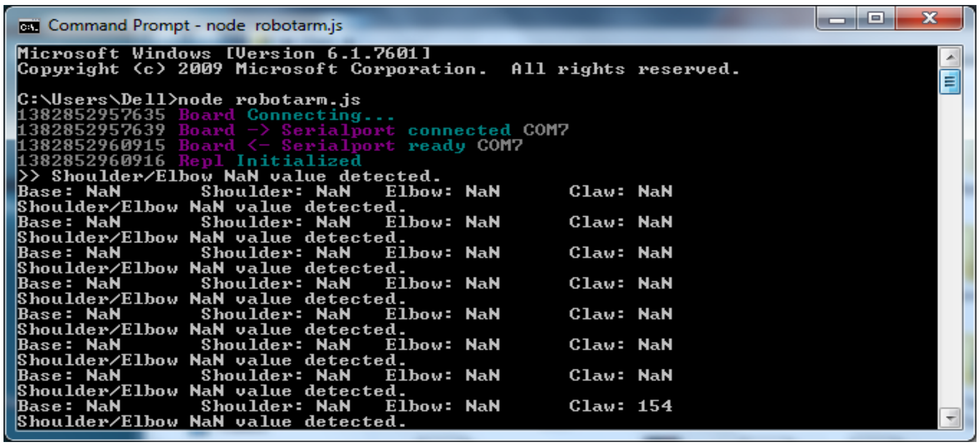
8. The data of the hands and fingers that the Leap Motion controller captured is observed from the JSON Viewer as shown in Figure 3.5.



```
{
  "currentFrameRate": 114.852,
  "gestures": [],
  "hands": [
    {
      "direction": [
        0.282376,
        0.131251,
        -0.950283
      ],
      "id": 42,
      "palmNormal": [
        0.143052,
        -0.985281,
        -0.0935767
      ],
      "palmPosition": [
        -26.1121
```

Figure 3.5: JSON Viewer

9. The programming is executed using command prompt by typing for an example 'node robotarm.js' and Enter button is hit as shown in Figure 3.6.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Dell>node robotarm.js
1382852957635 Board Connecting...
1382852957639 Board -> Serialport connected COM7
1382852960915 Board <- Serialport ready COM7
1382852960916 Repl Initialized
>> Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: NaN
Shoulder/Elbow NaN value detected.
Base: NaN        Shoulder: NaN   Elbow: NaN      Claw: 154
Shoulder/Elbow NaN value detected.
```

Figure 3.6: Program execution

3.2.4 Test the Sample Robotic Arm with Real Hand Gesture

From this test, the hand gesture is determined whether the formulas for the forward and inverse kinematics of the robotic arm are correct or not. Fortunately, a sample of a 4 Degree of Freedom robotic arm used to test the program and the inverse formula equations. If the robotic arm is not synchronizing with the hand and fingers gesture, the program need to be edited again. The formulas and the program of formula must be correct. Figure 3.7 and Figure 3.8 show the Arduino Uno Rev3 board microcontroller and the sample of robotic arm respectively.

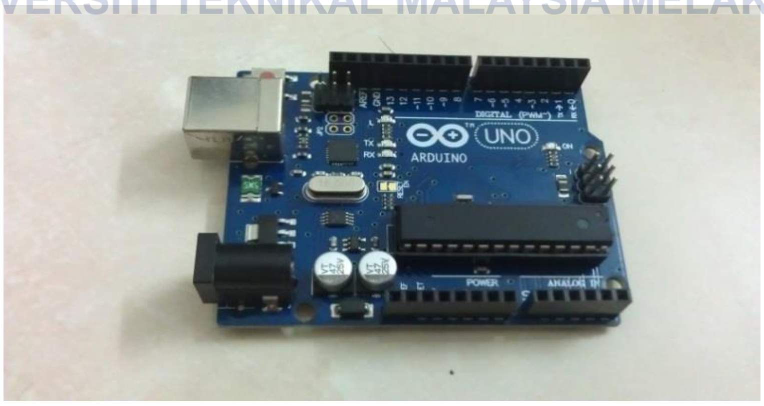


Figure 3.7: Arduino Uno Rev3 board microcontroller

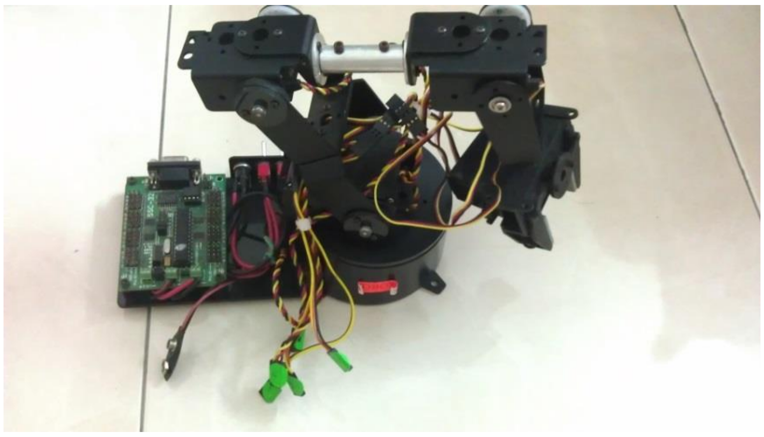


Figure 3.8: Sample of robotic arm from the Robotic Laboratory

3.3 End User Verification

For example the end user is like operator of the robotic arm who operated the robotic arm. They are the key people who can give commends and critics to improve the prototype of robotic arm.

3.3.1 Robotic Arm Prototype

The sample robotic arm needed to test, but that is belonged to the university property, it cannot be a personal property. Moreover, the robotic arm is unable to perform proper pick and place task by using hand gestures. Thus a modification is applied to the robotic arm not just in programming wise, but also in hardware. However, modification cannot be done to the sample robotic arm, as a result a newly design robotic arm is designed to meet the criteria of pick and place task controlled by hand gestures.

3.3.1.1 Robotic Arm Design

The robotic arm is designed using the SolidWorks. The dimension for every part of the structure is decided carefully. The dimensions of the servo motors are took into drawing to have the exact design output of the robotic arm. Type of hole and type of screws needed for assemble also included.

3.3.1.2 Type of Material used for the Structure and Components for the Robotic Arm

The robotic arm's structure is done by 3D printing machine. The material used by 3D printer is filament which the same material as the Lego toys. The thickness of the structure will be thicker to increase the strength of the structure. Servomotors are used to move the robotic arm. Each part of the robotic arm is used differently as the weightage for each joint is different than the others. For the base of the robotic arm is unavailable. Moreover the design of the robotic arm needed to be long, thus a middle to extend the length of the robotic arm is needed. Thus a base and a middle are needed to design by using SolidWorks and fabricate out using Rapid Prototyping Machine.

3.3.1.3 Assemble the Robotic Arm

After the structures and the servomotors are ready, all part can be assembled together. The wiring for the servomotors to the Arduino Uno board is done. Screws are needed for tightening the structure and the servo motors.

3.3.2 Record Randomly Picked Participants' Understandability When Controlling the Robotic Arm and also Test the Reliability of the Robotic Arm

As the hardware of the robotic arm is completed, participants are randomly pick at an exhibition to test the understandability when controlling the robotic arm and also test the reliability of the robotic arm. To test, a pick and place task is given to the participants and before that, a demonstration on controlling the robotic arm using the Leap Motion controller is shown. The task is just picking a block of toy and place into a box. After completed the task, they are given a survey form to answer few simple questions. It is an electronic survey where the participants submitted the survey through Google. Maximum 2 trials will be given to allow the participants to perform the simple pick and place task given. If the participant completed the task, thus it considered a check mark, if else cross mark is counted. The sample of the survey form is shown in the Appendix A. The flow chart of the flow of the project and robotic arm prototype is shown in Appendix B.

3.4 Experiment

The accuracy and precision of the robotic arm performance is very important. Many industries such as food packaging industry, the processes are repeated and the robot need perform for a very long period of time. Thus the robot must make sure the target to pack the food into a box must be constant. If it misses the target, even a single millimeter, it will disturb the whole process. Although this project is not focus on industrial type robotic arm, however, a few experiments needed to set up to check the accuracy and precision of the robotic arm.

3.4.1 Experiment 1: Error of the Servomotor

This experiment is to analyse the error given by the servomotors. The type of servomotors used in each joint of the robotic arm is a RC servomotor but not encoded servomotor. Thus it has no feedback to the controller. At default, the servomotor gives the angle of rotation not as programmed. For a robotic arm, accuracy is very crucial. Accuracy is the difference between a measurement reading and the true value of that measurement. Thus there is error that must be eliminated. Error is the amount of deviation from a standard or specification. Errors should be eliminated in the measuring process. The end effector of the robotic arm is unable to reach to the accurate position with a less accuracy servomotor.

To do this experiment, only the robotic arm’s waist, shoulder and elbow servomotor are important to the end effector. As the waist is tested, the result is the same to shoulder and elbow. However, the wrist rotation, gripper and wrist linear servomotor are not taken into consideration as these parts’ of accuracy are not important. These degrees of freedom are randomly adjusted base on the user because the sensitivity can be tuned. Moreover, at these degrees of freedom data for wrist rotation and wrist linear given by JSON Viewer are in the number from -1 to 1 as shown in Figure 3.10. However, for the gripper, the distance between 2 claws is also not important in accuracy as is depended from the user’s fingers. User able to adjust the distance of the gripper’s claws by observed the gripper’s claws and altered by the user’s finger.

The waist of the robotic arm used C40R servomotor while elbow and shoulder used C55R servomotor. Moreover, wrist rotation and the gripper used C36R servomotor while wrist linear used C40R. All type of the servomotor has almost the same error.

From the Leap Motion controller, the data given is considered accurate as default with a proper light condition. No changes of settings or modification are done on the controller. All the data given from the JSON Viewer are taken to the account without calibration or modification. The procedure to carry out this experiment is as followed:

1. The servomotor is stickled with a protractor scaled paper, and a pointer as shown in Figure 3.9.
2. Angles from 0˚ to 180˚ with 10˚ intervals readings are programmed to feed into servomotor.
3. The servomotor is set to 0˚as programmed.
4. The programmed is executed from the command prompt for 10˚.
5. The data is recorded.
6. The servomotor is set back to 0˚again as programmed.
7. The programmed is executed from the command prompt for 20˚ and so on as programmed.
8. The data is recorded.
9. Steps 6 to 8 are repeated for 8 times to get the average data of the rotation angle of the servomotor.
10. The error is calculated by comparing the average data and the programmed value.
11. The standard deviation is calculated.
12. Steps 6 to 11 are repeated for 8 times to get the average data of the rotation angle of the servomotor after the calibrated value is added into calculation.

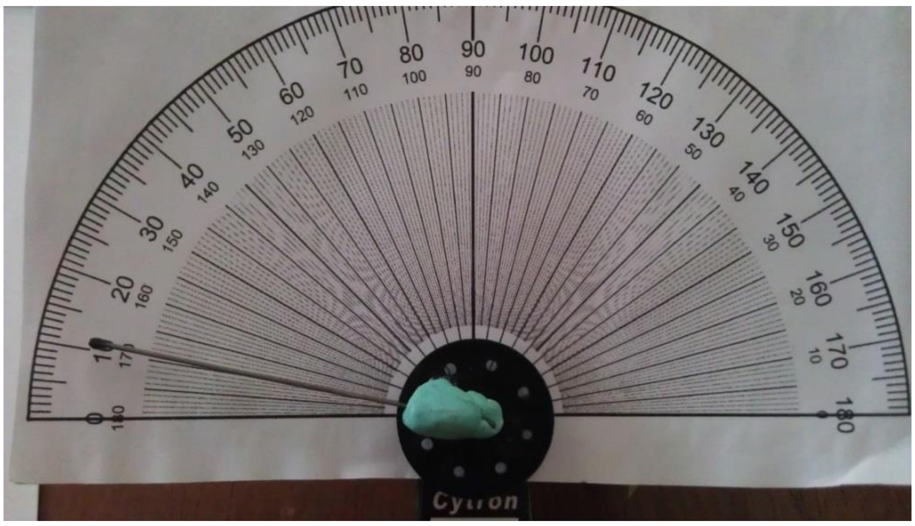
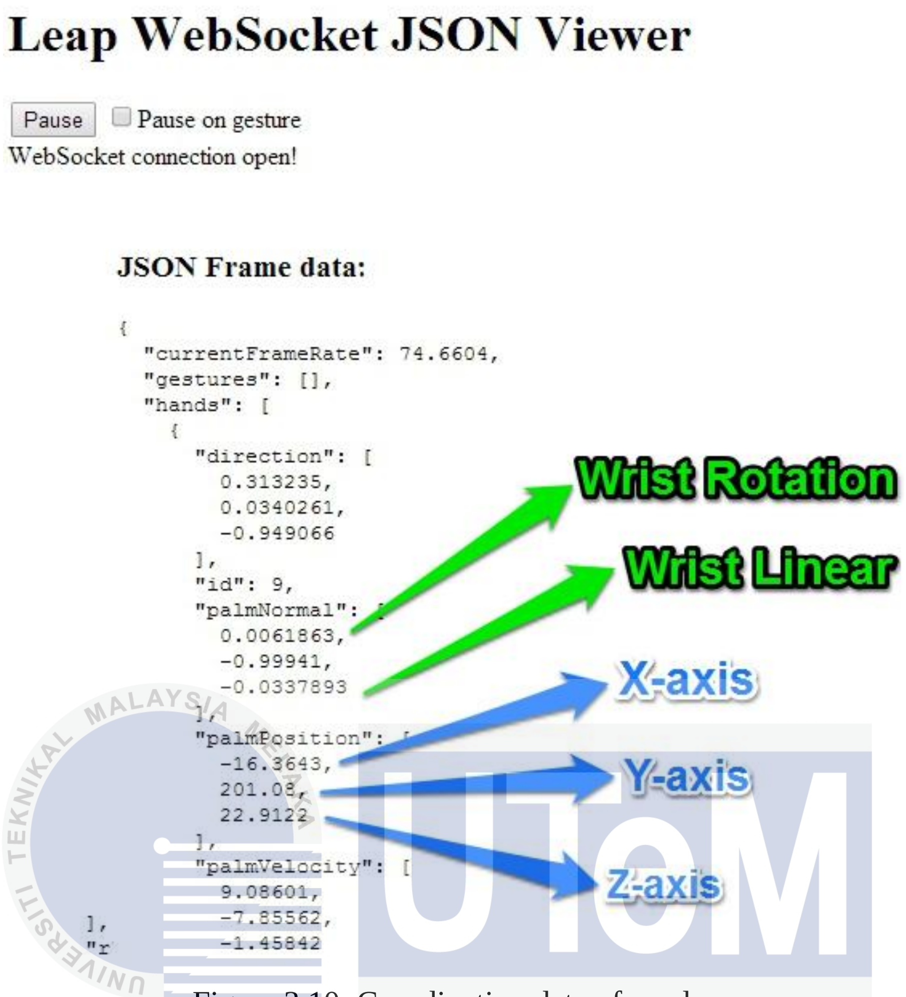


Figure 3.9: Servomotor with a protractor scaled paper



Leap WebSocket JSON Viewer

Pause Pause on gesture
WebSocket connection open!

JSON Frame data:

```
{
  "currentFrameRate": 74.6604,
  "gestures": [],
  "hands": [
    {
      "direction": [
        0.313235,
        0.0340261,
        -0.949066
      ],
      "id": 9,
      "palmNormal": [
        0.0061863,
        -0.99941,
        -0.0337893
      ],
      "palmPosition": [
        -16.3643,
        201.08,
        22.9122
      ],
      "palmVelocity": [
        9.08601,
        -7.85562,
        -1.45842
  ],
  "r
```

Figure 3.10: Coordination data of a palm

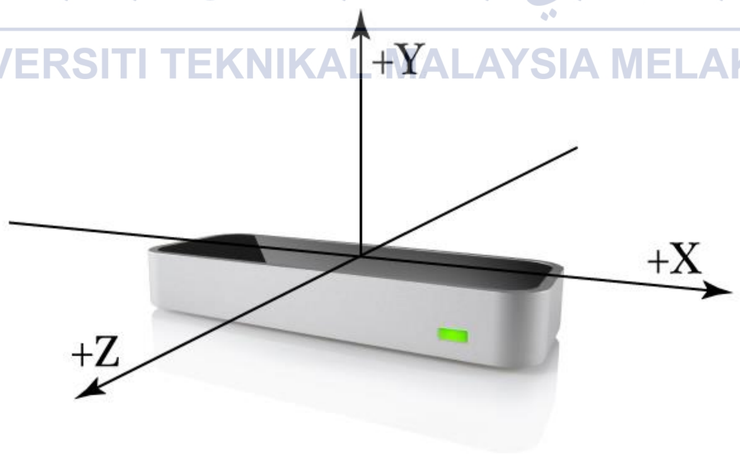


Figure 3.11: The Cartesian coordinate of the Leap Motion controller

The Figure 3.11 shows Cartesian coordinate system of *x*, *y* and *z* axes of the Leap Motion controller [14]. The origin is midpoint at the top of the Leap Motion controller.

$$\begin{aligned} Error &= True\ Value - Approximate\ Value \\ &= Programmed\ Angles - Average\ Servomotor\ Readings \end{aligned} \tag{3.1}$$

$$Relative\ Error = \left|\frac{Error}{True\ Error}\right| \tag{3.2}$$

$$True\ per\ cent\ relative\ error = Relative\ Error \times 100\% \tag{3.3}$$

$$Standard\ deviation, s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}} \tag{3.4}$$

3.4.2 Experiment 2: Accuracy's Reliability on the Waist Experiment

After the calibrated value is added into account of the programming calculation of the servomotor, there is a decreased of error of the rotation angle of the servomotor, and this eventually increased the accuracy of the servomotor. To test the accuracy's reliability, another analysis is tested with dummy hand as an example of hand and moved on the Leap Motion controller. The Leap Motion controller also can sense the dummy hand as a real hand. Dummy hand is used to avoid hand sore and tired also easy to get measurements. Thus the coordination of the palm position's data from the JSON Viewer is fed to the programming for calculation as refer to Figure 3.10. This is to use the real time coordination of the palm position's data from the JSON Viewer and check the error of the rotation angle of the servomotor with and without the calibrated value that done previously. The procedure to carry out this experiment is as followed:

1. The servomotor is stickled with a protractor scaled paper, and a pointer as shown in Figure 3.9.
2. The dummy hand is place on a box with a height of 185mm form the surface of Leap Motion controller according to the data given by JSON Viewer.
3. The servomotor is set to 90˚as starting point as programmed.
4. The programmed is executed from the command prompt.
5. The dummy hand is placed at the coordinate of x-axis is 200mm, y-axis is 185mm and z-axis is 0mm.
6. The servomotor rotated and the data is recorded.

7. Steps 5 to 6 are repeated for 3 times to get the average data of the rotation angle of the servomotor.
8. Steps 5 to 7 are repeated for 5 times to calculate the average value.
9. Steps 5 to 8 are repeated for x-axis varied to 150mm, 100mm, 50mm, 0mm, -50mm, -100mm, -150mm and lastly -200mm.
10. The error is calculated by comparing the average data and the programmed value.
11. The standard deviation is calculated.
12. Steps 5 to 11 are repeated after the calibrated value is added into calculation.

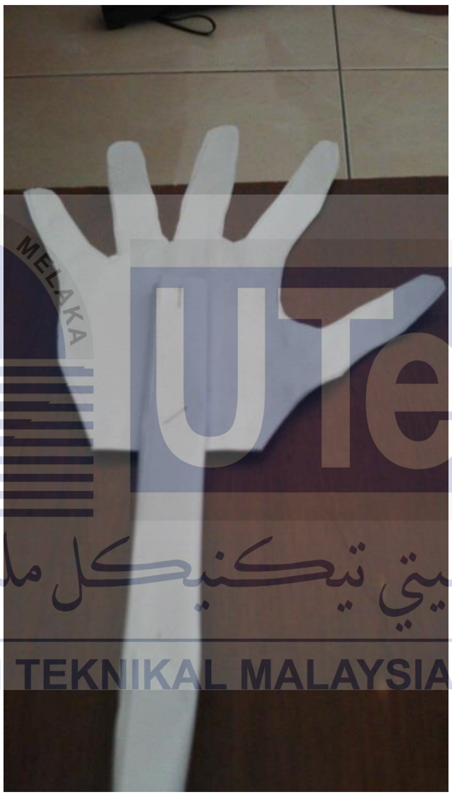


Figure 3.12: Dummy hand made of layer of papers

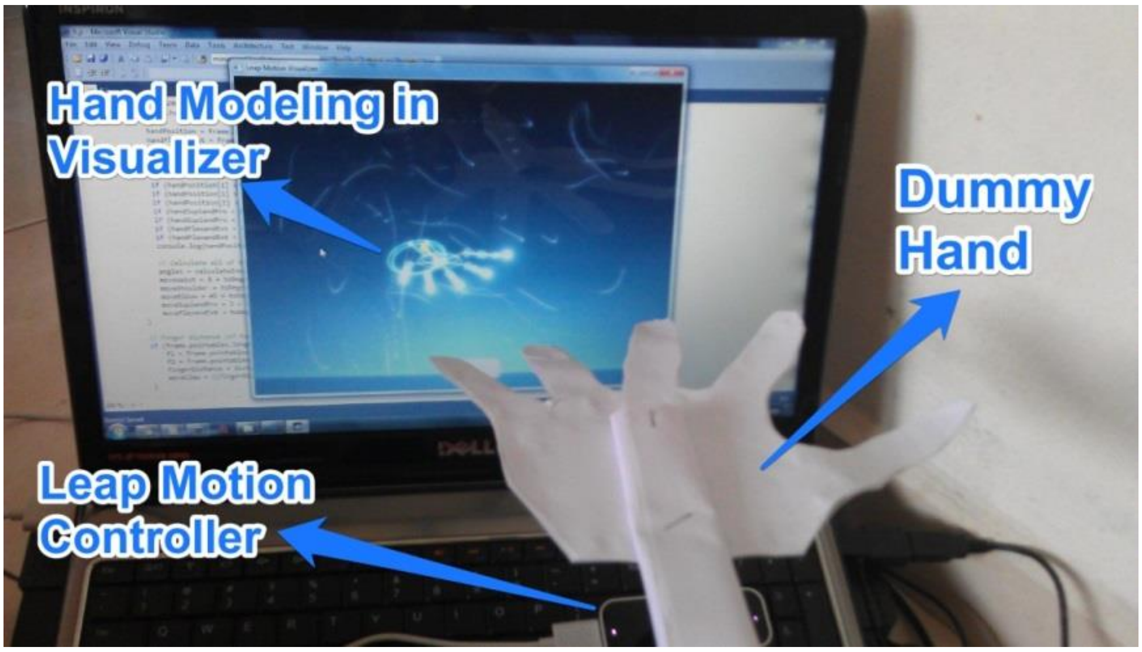


Figure 3.13: Dummy hand on Leap Motion Visualizer

3.4.3 Experiment 3: Precision of the Robotic Arm by Repeatability Test

The coordinates for the target is given to the robotic arm to reach a point from its initial position as a task. If the given task is repeated where the robotic arm needed to reach the same target every cycle, does the robotic arm will misalign with the target given? This experiment is performed to analyze the precision of the robotic arm performance by using statistical method by giving a coordinate which on Quadrant 1. The coordinate and the mark "X" is mapped on the graph paper as shown in Figure 3.14 and Figure 3.15 respectively. However, the joints at wrist rotate, wrist linear and gripper must be disabled. Means these joints must set at fix value to prevent data updates to these joints. Thus, wrist rotate and wrist linear are set to 90° and gripper set to zero. The procedure to carry out this experiment is as follows:

1. The *x*-coordinate, *y*-coordinate and *z*-coordinate are set in the programming as 100, 200 and 0 respectively.
2. The programmed is executed from the command prompt and the robotic arm is in initial position.
3. Hand is put on the Leap Motion controller to initiate the programming.
4. The gripper's tip touched on the graph paper and the center point is recorded as reference.

5. Hand is removed from the Leap Motion controller in order the robotic arm to back to its initial position.
6. Hand is placed on the Leap Motion controller again to initiate the programming.
7. The gripper's tip touched on the graph paper and the "X" is marked on the graph paper and the *x*-coordinate and *z*-coordinate data are recorded.
8. Steps 6 to 7 then step 5 are repeated 15 times to calculate the mean and standard deviation.

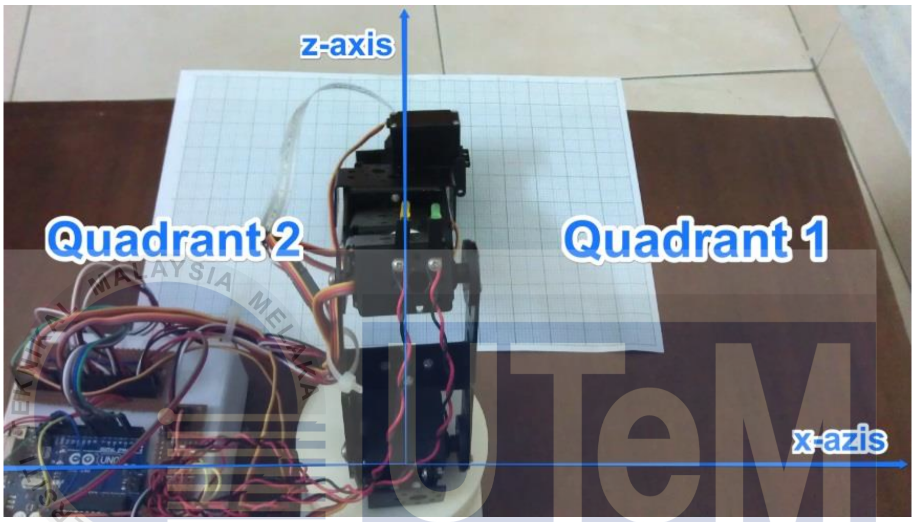


Figure 3.14: Coordinate target is at Quadrant 1

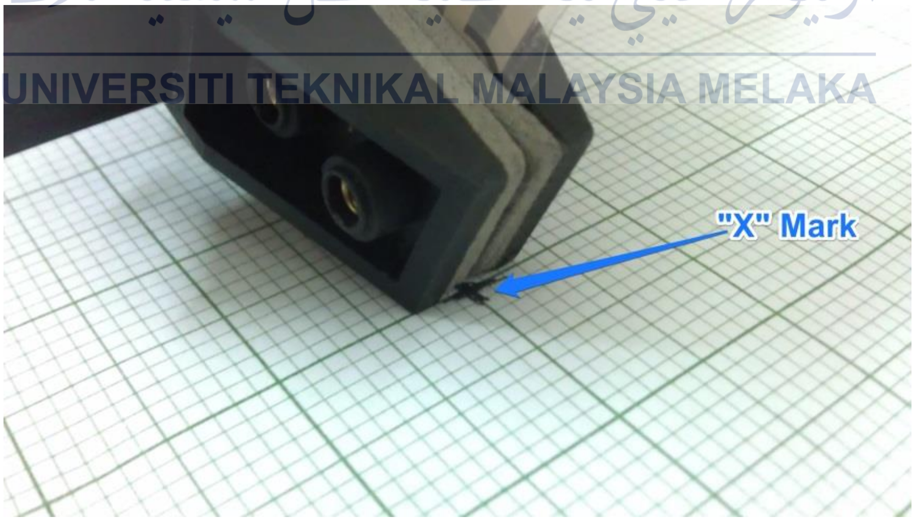


Figure 3.15: "X" Mark on the graph paper (as the center point)

3.4.4 Validity of the Experiments

As for the Experiment 1 and 2 for finding the accuracy of the servomotor, the method and calculation to find out the average, percent relative error and standard deviation in mathematically method. The equations and the calculation steps can be applied to this Experiment 1 to Experiment 3.

While for Experiment 3, this experiment is conducted during Mechatronic System Engineering Laboratory 3. It also conducted the experiment on repetitive pick and place application from the part feeder to the conveyer using a RHINO robot. It also needed to program the coordinates of the path that the RHINO robot need to take to the target position by using RoboTalk™. The RHINO robot also can used the Teach Pendent to move each corresponding of motor until the robot move to the target position requires. However, this project does not have Teach Pendent, thus using the programming method to enter the coordinate of the target position as mentioned. Thus, the repetitive procedure of Experiment 3 can be referred to this RHINO robot experiment as guides and finding the precision of the robotic arm is possible. Moreover, RHINO robot experiment on measuring accuracy of the RHINO robot also conducted during the laboratory. Thus the procedure of this experiment can be referred for Experiment 1 to Experiment 3.

3.4.5 Reliability of the Experiments

The experiment to test the error of the servomotor and the experiment to test the accuracy of the waist of the robotic arm have common influential factors that interferes the experiments. The first factor is the systematic error. This error usually occurs when there is wrong with the measuring instrument of data handling system or the instrument is wrongly used. Such error must be avoided by avoided parallax error when taking data from the protractor. The needle and the eye observant must be in the right position and the needle must parallel to the scale of the protractor. Moreover, from the methodology, many steps can be caused by the human errors which may cause the data to differ. As the systematic error reduced, the accuracy of the measurements increased.

Another factor is the random error. This error usually occurs in electronic noise in the circuit of the servomotor or the controller. This error can be avoided by taking measurement repeatedly and calculate the average data taken.

However, during the conduction of the experiment to test the accuracy of the waist of the robotic arm, it used the Leap Motion controller. It has control variable that must be avoided which is the operating environment. The Leap Motion controller performance is light dependent. Bright light sources or reflective sources will impact performance. Thus, less bright or reflective environment is a suitable condition. For best performance, there few things that need to be concerned which is listed below [15]:

1. Maintain clear area of view between the Leap Motion controller and the hands and fingers.
2. Avoid loose sleeves, jewellery and non-transparent objects near the device.
3. Avoid wear dark glove, or dark or transparent instruments to use the controller.

The Leap Motion device has a LED indicator such as in Figure 3.16. When the LED status is green colour means is powered on and the light source is in suitable condition. If the LED status is not green, such as yellow or red, means either the light source is not in good condition, something wrong to the device or software.

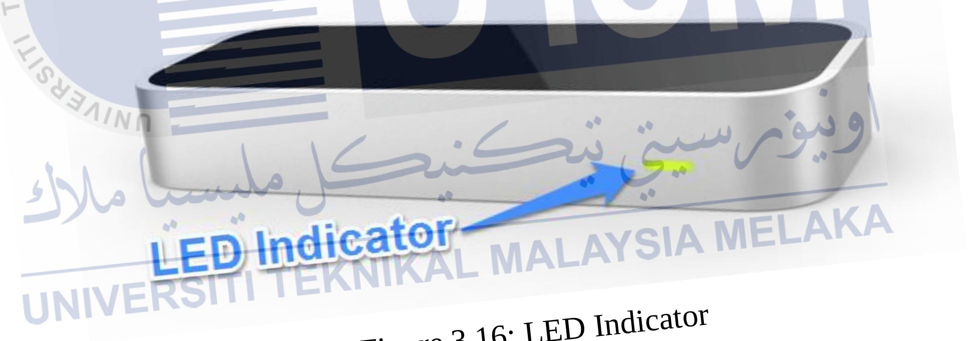


Figure 3.16: LED Indicator

3.5 Cost and Budget

Cost and budget planning is important for a project. A good planning on cost and budget can help save money in order to produce low cost product and able to spend extra on more important sides. Table 3.1 below shows the cost and budget for the material.

Table 3.1: Cost and budget of material

Material	**Company**	**Cost (RM)**	**Quantity**	**Budget (RM)**
Leap Motion controller	Leap Motion, Inc	250	1	250
Servomotor (C55R)	Cytron Technologies	88	1	88
Servomotor (C40R)	Cytron Technologies	49	2	50
Servomotor (C36R)	Cytron Technologies	39	3	40
Arduino Uno Rev3	Cytron Technologies	72	1	70
U-bracket	Cytron Technologies	15	3	15
Servo Bracket	Cytron Technologies	16	4	15
Gripper	Robotic Shop Malim	50	1	50
Power Adapter	Central Electronic Store	40	1	40
Others	-	-	-	-
	Total	619	Total	618

3.6 Gantt Chart

Gantt chart is done and used to timeline the project of design robotic arm for pick and place purpose controlled by Leap Motion controller. Gantt chart is planned for time management for the project to avoid over budget and avoid any slack which causes failure project and. The Gantt chart is shown in Appendix C.

CHAPTER 4

RESULT AND DISCUSSION

4.1 SolidWorks Drawing

Figure 4.1 below shows the drawing of the robotic arm structure using SolidWorks. This drawing included the servomotors, the servo bracket, U bracket, microcontroller's deck, gripper for the purpose of pick and place and other parts that needed to fabricate. The dimensions of each part are as almost accurate as the real parts. The base, base cover and the middle design are shown in Appendix D, Appendix E and Appendix F respectively.

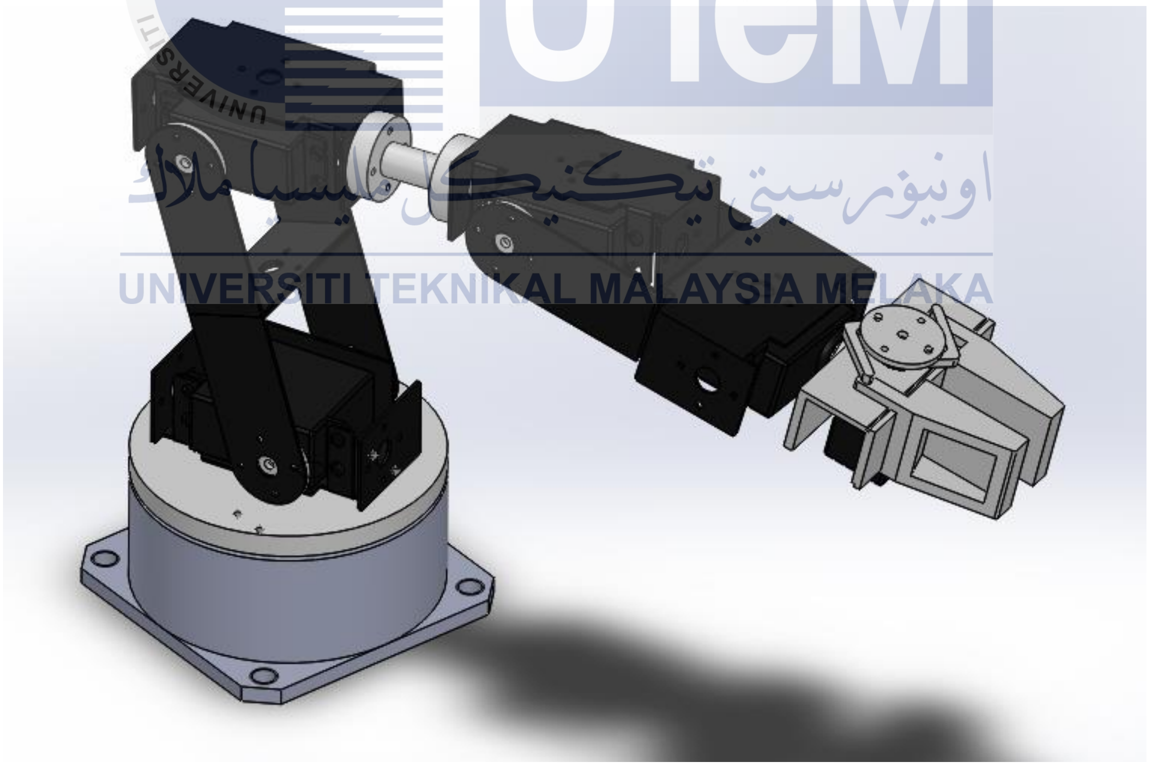


Figure 4.1: Robotic arm for pick and place drawing

4.2 Degree of Freedom of the Robotic Arm

The designed robotic arm has five degree of freedom which consisted of waist, shoulder, elbow, wrist and fingers. However, the wrist part has two degree of freedom. It able to do human's wrist movements which are flexion, extension, pronation and supination. Flexion and extension are a linear movement at the wrist while pronation and supination are the rotation movement at the wrist.

The structure of the gripper showed the number of fingers, means the position of the two fingers determined the position of the clamps of the gripper. Figure 4.2 shows the human representation parts on the robotic arm while Figure 4.3 shows the movement of each part of the robotic arm.

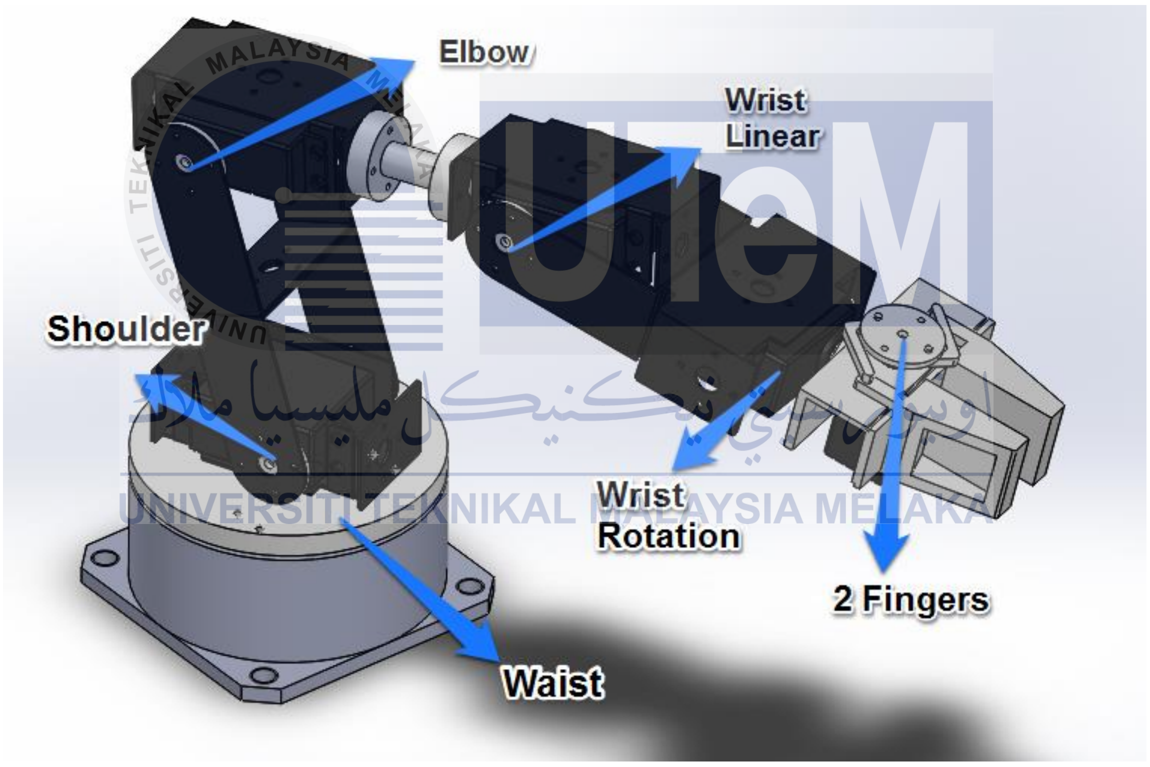


Figure 4.2: Human representation parts on the robotic arm

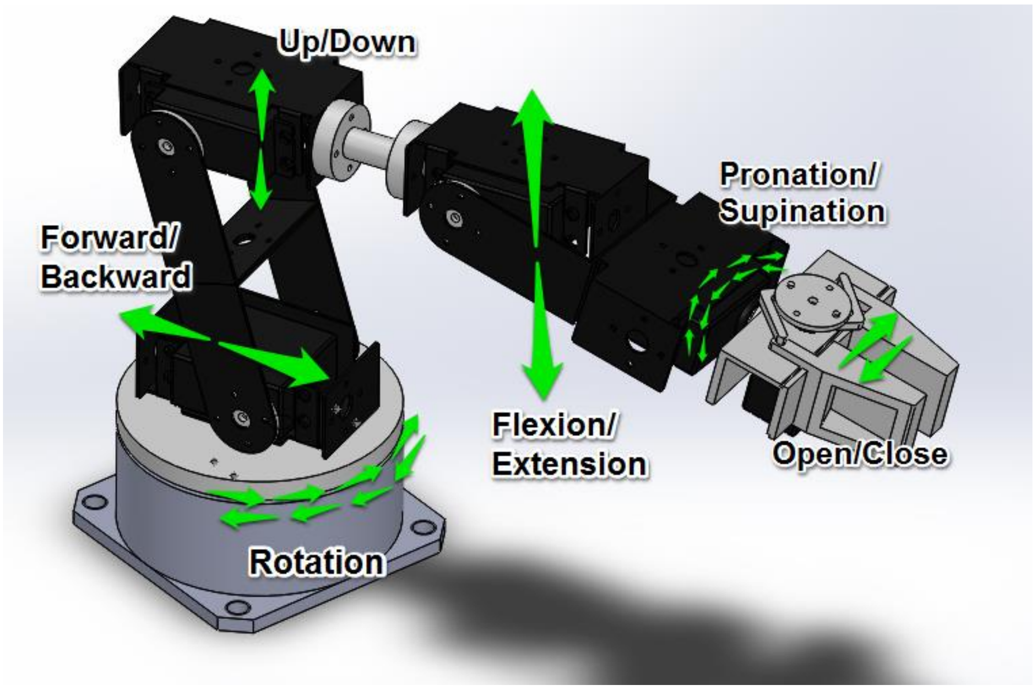


Figure 4.3: Movement of each part of the robotic arm

4.2.1 Affixing Frames on Robotic Arm Link

The forward kinematics calculation is started with affixing the frames at the robotic arm links as shown below. There is just two degree of freedom to be calculated in the forward and inverse kinematics because this two degree of freedom is related to each other while the other parts are independent to shoulder and elbow of the robotic arm. The gripper, or as the end effector or also a tool is not count as a degree of freedom. Figure 4.4 shows the two degree of freedom of the shoulder and the elbow [11].

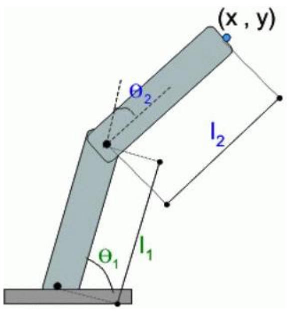


Figure 4.4: Steer end-effector to (x, y) target position

4.3 Inverse Kinematics

Inverse kinematics compute the set of joints angle by given the desired or required position and orientation of the robotic arm structure by using algebraic approach. To test the motion of the robotic arm using the algorithm created, a sample robotic arm has borrowed from the Robotic Laboratory.

4.3.1 Algebraic Equations

The Leap Motion controller fed the position of the hands and fingers in the *x*, *y* and *z*-axis, thus the angle of the shoulder and elbow is calculated as below [11]:-

$$x = l_1\cos(\theta_1) + l_2\cos(\theta_1 + \theta_2) \tag{4.1}$$

$$y = l_1\sin(\theta_1) + l_2\sin(\theta_1 + \theta_2) \tag{4.2}$$

$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1l_2\cos(\theta_2) \tag{4.3}$$

$$\cos(\theta_2) = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \tag{4.4}$$

$$x = l_1\cos(\theta_1) + (l_2(\cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2)) \tag{4.5}$$

$$x = \cos(\theta_1)(l_1 + l_2\cos(\theta_2)) - \sin(\theta_1)(l_2\sin(\theta_2)) \tag{4.6}$$

$$y = \cos(\theta_1)(l_2\sin(\theta_2)) + \sin(\theta_1)(l_1 + l_2\cos(\theta_2)) \tag{4.7}$$

$$\cos(\theta_1) = \frac{x + \sin(\theta_1)\, l_2\sin(\theta_2)}{l_1 + l^2\cos(\theta_2)} \tag{4.8}$$

$$\sin(\theta_1) = \frac{(l_1 + l_2\cos(\theta_1))y - l_2\sin(\theta_2)\,x}{l_1^2 + l_2^2 + 2l_1l_2\cos(\theta_2)} \tag{4.9}$$

4.4 JavaScript Programming Program

The program for the controlling the robotic arm is in the Appendix G [12]. The meaning of the data that taken from the JSON Viewer is as shown below [13]:-

1. `frame.hands[0].palmPosition[0]` = 1 of the hand's palm position at *x*-axis.
2. `frame.hands[0].palmPosition[1]` = 1 of the hand's palm position at *y*-axis.
3. `frame.hands[0].palmPosition[2]` = 1 of the hand's palm position at *z*-axis.
4. `frame.hands[0].palmNormal[0]` = 1 of the hand's palm normal to the in *x*-axis.
5. `frame.hands[0].palmNormal[2]` = 1 of the hand's palm normal to the in *z*-axis.
6. `frame.pointables[0]` = the first pointable object with the specified ID associated with this hand.
7. `frame.pointables[1]` = the second pointable object with the specified ID associated with this hand.
8. `frame.pointables[0].tipPosition[0]` = the first pointable object's tip position at the x-axis.
9. `frame.pointables[0].tipPosition[1]` = the first pointable object's tip position at the x-axis.
10. `frame.pointables[0].tipPosition[2]` = the first pointable object's tip position at the x-axis.
11. `frame.hands.length` = the number of fingers.

4.4.1 Restrict Certain Inputs to Prevent Physical Damage

Human arm has its limitation of movement at the each joint so do the robotic arm. Each servomotor can only rotate from 0° to 180° only. However, if the servomotor turns over than 180° will damage to the servomotor. Thus, restriction to certain input is set to prevent physical damage.

The input data from the Leap Motion controller is taken from JSON Viewer.

```
if (handPosition[0] > 250) handPosition[0] = 250;
if (handPosition[0] < -250) handPosition[0] = -250;
if (handPosition[1] < 120) handPosition[1] = 120;
if (handPosition[1] > 415) handPosition[1] = 415;
if (handPosition[2] > 200) handPosition[2] = 200;
if (handSupiandPro > 0.833) handSupiandPro = 0.833;
if (handSupiandPro < -0.833) handSupiandPro = -0.833;
if (handFlexandExt > 0.8) handFlexandExt = 0.8;
if (handFlexandExt < -0.6) handFlexandExt = -0.6;
```

4.4.2 Mathematical Functions

Many mathematical formulas are needed in this calculation, but JavaScript only understand its own language. Thus the formulas must be written in JavaScript language.

```
function distance(x1, y1, z1, x2, y2, z2) {
    return Math.sqrt(square(x2 - x1) + square(y2 - y1) + square(z2 - z1));
}

function square(x) {
    return x * x;
}

function toDegrees(r) {
    return r * 57.2957795;
}
```

4.4.3 Inverse Kinematics Equation

According to the Equation [4.4] and Equation [4.9], the formulas are written in JavaScript language for the computer to able to execute it. The l_1 and l_2 are both means θ_1 and θ_2 respectively. Figure 4.5 and Figure 4.6 show the movement of the shoulder and the elbow of the robotic arm when the hand is moving forward and backward respectively.

```
var t1 = Math.acos((square(z) + square(y) - square(l1) - square(l2)) / (2 * l1 *
l2))
      + Math.acos((square(z) + square(y) - square(l1) - square(l2)) / (2 * l1 * l2))
      * 0.0688;

var t2 = Math.asin(((l1 + l2 * Math.cos(t1)) * y - l2 * Math.sin(t1) * z) / (square(l1)
      + square(l2) + 2 * l1 * l2 * Math.cos(t1)))
       + Math.asin(((l1 + l2 * Math.cos(t1)) * y - l2 * Math.sin(t1) * z) / (square(l1)
      + square(l2) + 2 * l1 * l2 * Math.cos(t1))) * 0.0688;

angles = calculateInverseKinematics(handPosition[0], handPosition[1],
handPosition[2]);

          theta1: t1,

          theta2: t2,

      moveShoulder = toDegrees(angles.theta1);
```

```
moveElbow = 45 + toDegrees(angles.theta2);
```

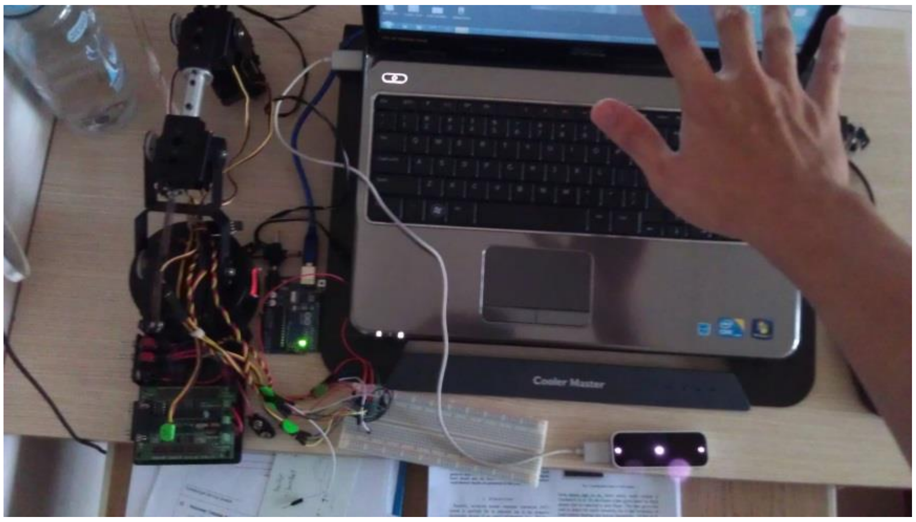


Figure 4.5: Hand move forward

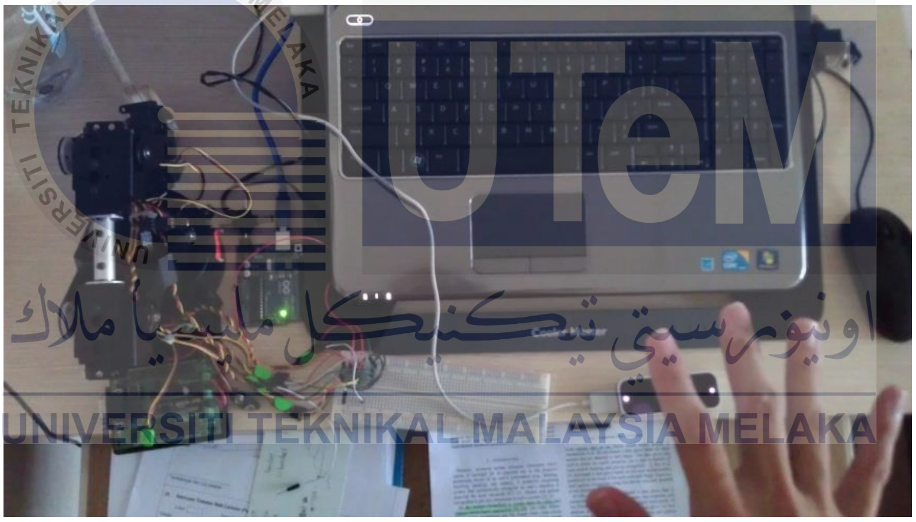


Figure 4.6: Hand move backward

4.4.4 Finger Distance (of Two Fingers only) Controls the End Effector

To calculate a distance between to point in a 3D Cartesian plane, the formula is shown as below which used ‘fingerDistance’. The ‘moveClaw’ is to determine the gripper to open and close with the same distance as the human fingers. The equation in the program for ‘moveClaw’ is for the sensitivity for the gripping. Figure 4.8 and Figure 4.9 show the gripper open fully and close fully respectively and Figure 4.10 shows the gripper open half according to the distance of the two fingers.

```
if (frame.pointables.length > 1) {
            f1 = frame.pointables[0];
            f2 = frame.pointables[1];
            fingerDistance = distance(f1.tipPosition[0], f1.tipPosition[1],
f1.tipPosition[2], f2.tipPosition[0], f2.tipPosition[1], f2.tipPosition[2]);
            moveClaw = ((fingerDistance / 1.2) - minClawDistance) * 4;
        }
```

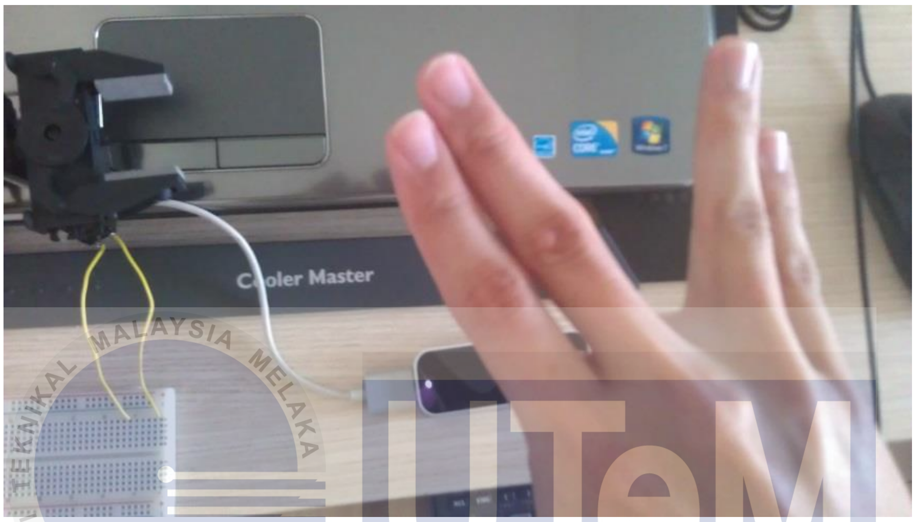


Figure 4.7: Gripper open full

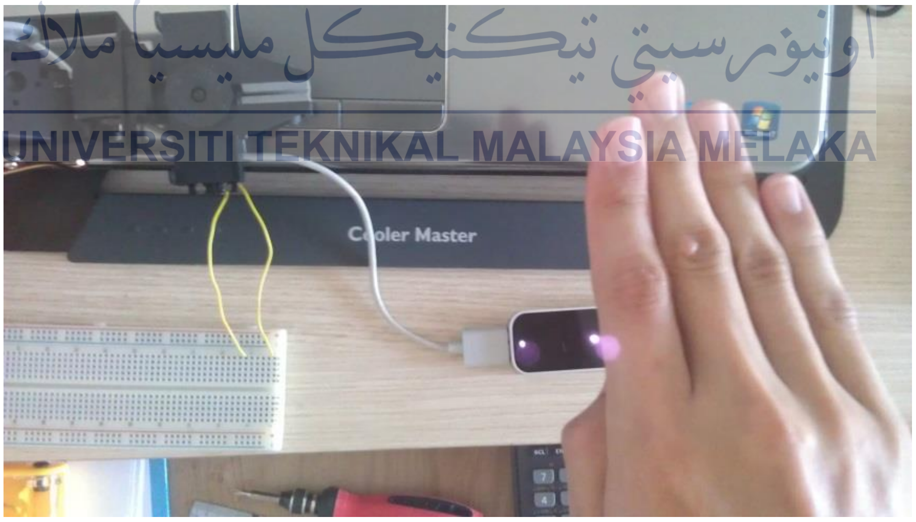


Figure 4.8: Gripper close full

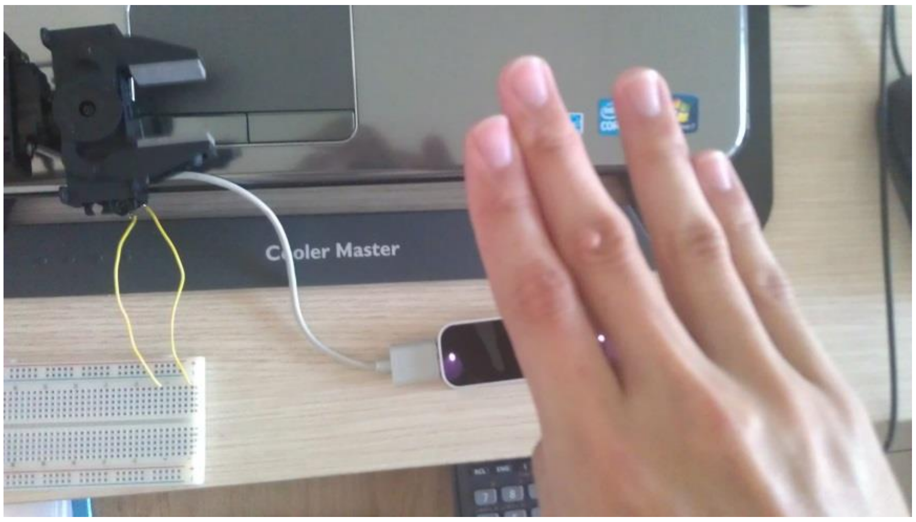


Figure 4.9: Gripper open small

4.4.5 Waist Calculation

The hand is not move in the circular way but move lineally to the x-axis, and the movement to need be nonlinear due to the circular nature of the Waist. Thus the calculation is as below. Figure 4.10 and Figure 4.11 show the waist movement of the robotic arm to move right and left position respectively.

```
moveWaist = toDegrees(angles.theta5);
var t5 = Math.acos(x / 250) + Math.acos(x / 250) * 0.0688;
theta5: t5,
```

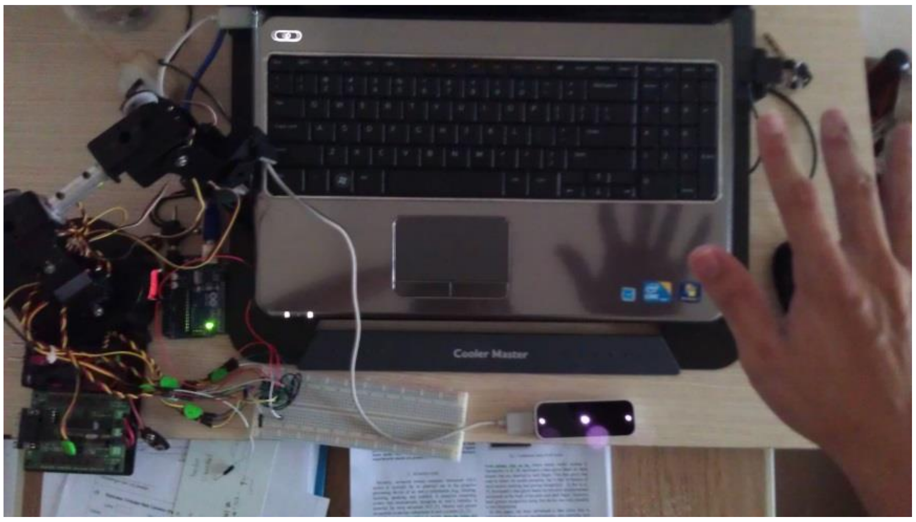


Figure 4.10: Hand move right

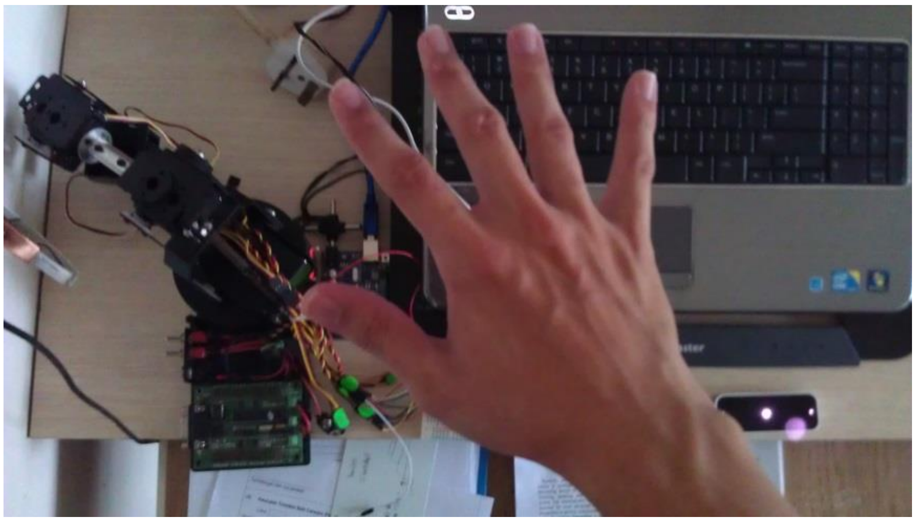


Figure 4.11: Hand move left

4.4.6 Wrist Calculation

In the JSON Viewer, the value of the ‘palmNormal’ is between -1 and 1 and value 0 means the palm is facing to the Leap Motion controller. This value never reach -1 or 1, thus an adjustment is added to the program to increase the sensitivity of the joints. Figure 4.12 to Figure 4.15 show the wrist movements of the robotic arm related to the program below.

```
moveSupiandPro = 20 + toDegrees(angles.theta3);
moveFlexandExt = 10 + toDegrees(angles.theta4);


var t3 = Math.acos((handSupiandPro) * 1.2);
    if (handFlexandExt > 0) {
        var t4 = Math.acos((handFlexandExt) * 1.3);                 // increase the
input data from JSON
    }
    else if (handFlexandExt < 0) {
        var t4 = Math.acos((handFlexandExt) * 1.45);                // increase the
input data from JSON
    }


theta3: t3,
theta4: t4,
```

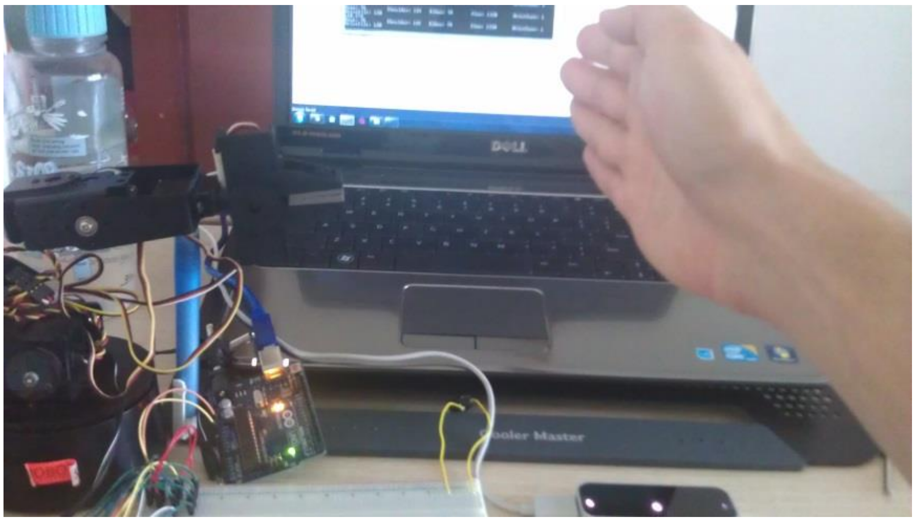


Figure 4.12: Wrist supination 90˚

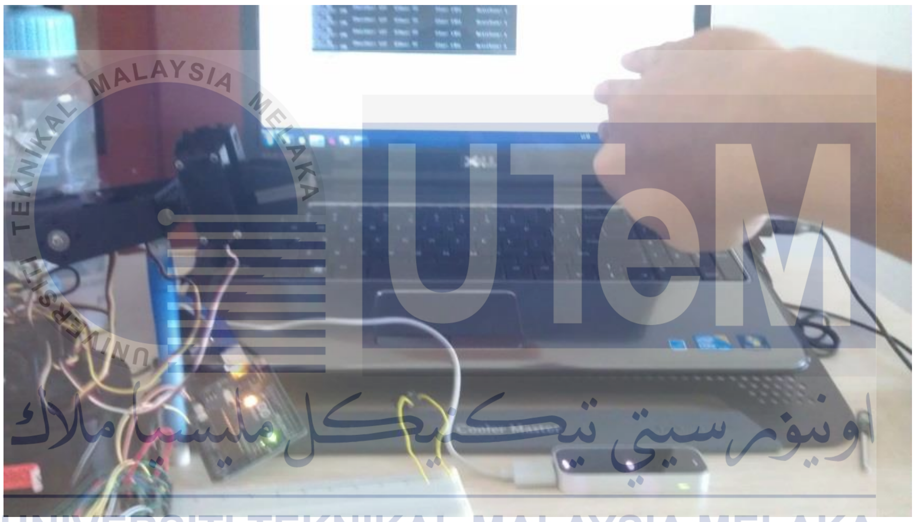


Figure 4.13: Wrist pronation 90˚

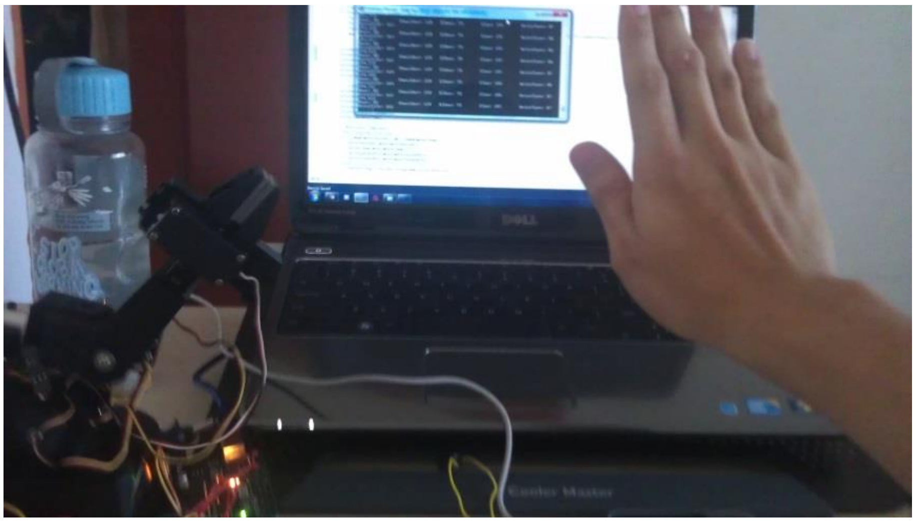


Figure 4.14: Wrist extension 90˚

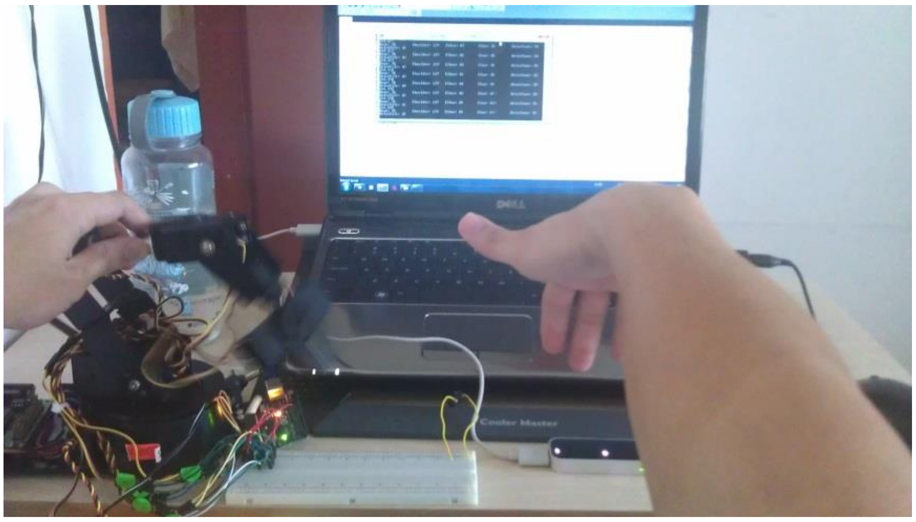


Figure 4.15: Wrist flexion 90˚

4.4.7 Initial Position Settings

For every robotic arm in the market such as for KUKA and ABB, it has initial position as an origin position every time when it started to function. Below is the program for the servomotors to start at the initial position assigned to each of the servomotors when hand and fingers are detected. Figure 4.16 shows the initial position of the robotic arm when hand is place on the Leap Motion controller.

```
servoWaist.center();
 servoShoulder.center();
 servoElbow.move(45);
 servoSupiandPro.move(90);
 servoFlexandExt.move(90);
 servoClaw.move(100);
```

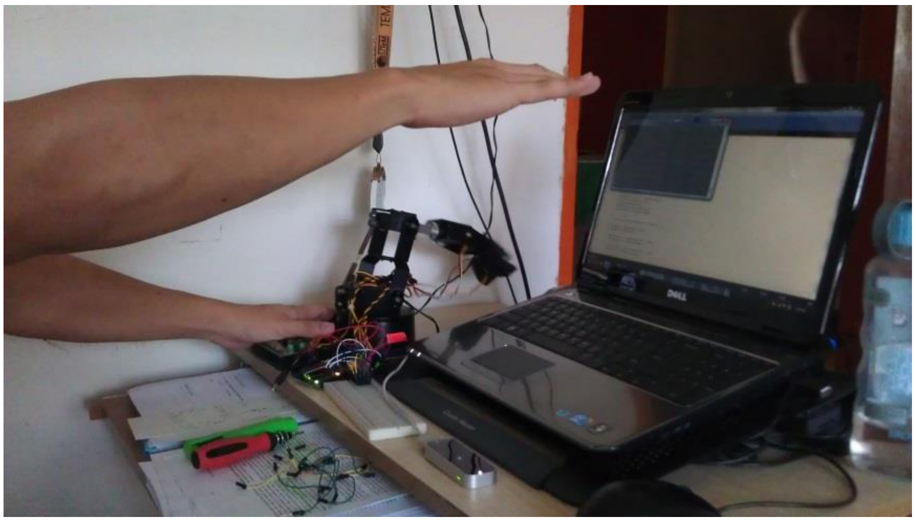


Figure 4.16: Shoulder and elbow at initial position

4.5 Arduino Uno Wire Connections

Figure 4.2 shows the wire connection to the robotic arm. Each servo's signal pin is connected to the respective ports that are set from the programming. The power of 5V and ground are also connected to the servomotors. Therefore, a 7-12V of power supply is enough for the board and the servomotors, but more than that will burn the board.

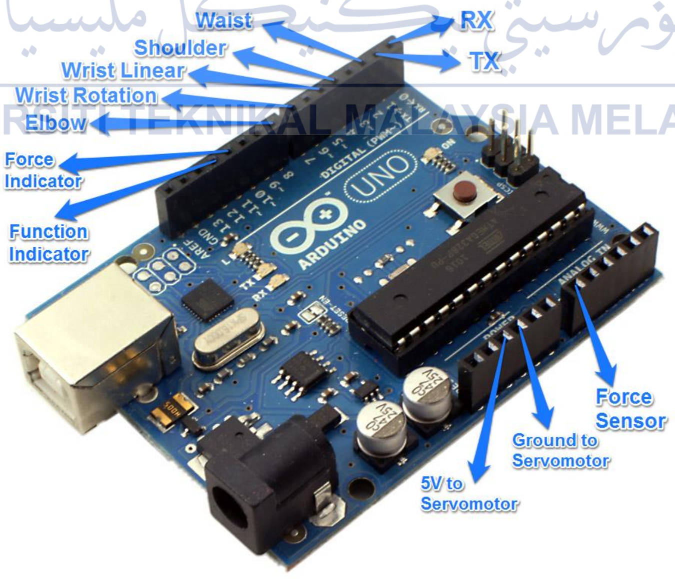


Figure 4.17: Wire connection on Arduino Uno

4.6 Result and Discussion from the Survey

Total of 24 respondents have done the survey right after they have experienced in controlling the robotic arm using their hand and finger gestures. From the total of 24 respondents, 21 of them claimed that it is difficult to control a machine using remote control. As expected, 22 out of 24 respondents have never control a machine using their hand and fingers gestures. This means that this way of controlling a machine is still a new in the market. Before they hand on to control the robotic arm, they given a live demonstration for one time to perform pick and place task using hand and fingers gestures, after that they are their own. Surprisingly, 19 out of 24 respondents are able to complete the pick and place task given after 2 trials. As the result, these respondents understand the mechanism of controlling the robotic arm with using their own hand and fingers gestures. Moreover, after the respondents completed the task, 17 out of 24 respondents found out that using own hand and fingers gestures are easier to control a machine.

From the survey, most of the respondents have problem on controlling the robotic arm with the hand and fingers gestures because it is tough or maybe it is their first time attempted. Meanwhile, some respondents gave suggestions on the way that the robotic arm and the controlling should be improved in order to improve the performance and the accuracy of the robotic arm. The summary of the result is at Appendix H.

As good news, all respondents agreed that this system is able to commercialize in the market for laboratory use. Consequently, a further research on this system is important. For instance, improving the accuracy and performance able to perform task as precise as own hand and fingers gestures. In addition, adding user interface such as adding a stop button for the user to stop if there is an emergency during performing task. Furthermore, the weight of the robotic arm also considered, thus when robotic arm performing a task, it has error at the angles. Consequently, a closed loop circuit can introduce to provide the same output from the data of the hand and fingers gestures. The robotic arm able to sync the human hand and fingers gestures almost every time. However, the servomotors are unstable, all of the servomotors have the shaking at the output shaft. Consequently this caused the whole robotic arm not stable and also shaking. Thus this eventually pulls down the accuracy of the robotic arm.

4.7 Experiment

4.7.1 Experiment 1: Error of the Servomotor

Table 4.1 recorded the average of servomotor angles and is compared with the programmed angles to find out the error of the servomotor and calculated the calibrated value to correct the error and increased the accuracy of the servomotor.

Table 4.1: Average servomotor angles compared to programmed angles

Programmed Angles, degree/°	Servomotor Readings								Average, degree/°	Standard Deviation
	1	2	3	4	5	6	7	8		
0	0	0	0	0	0	0	0	0	0.00	0.00
10	8	8	8	8	8	8	8	8	8.00	0.0000
20	19	19	18	19	20	19	19	19	19.00	0.5345
30	28	29	29	28	29	28	29	29	28.63	0.5175
40	38	38	38	38	38	38	38	38	38.00	0.0000
50	46	46	47	45	45	46	46	46	45.88	0.6409
60	55	55	55	54	55	55	55	55	54.88	0.3534
70	64	64	64	64	65	64	64	63	64.00	0.5345
80	74	74	73	73	73	73	74	74	73.50	0.5345
90	82	82	81	82	82	81	81	82	81.63	0.5175
100	91	91	92	90	90	91	91	91	90.88	0.6409
110	102	102	101	102	101	102	102	101	101.63	0.5175
120	112	112	112	111	112	112	111	112	111.75	0.4629
130	122	123	122	122	123	123	122	123	122.50	0.5345
140	132	132	133	133	132	132	133	133	132.50	0.5345
150	142	142	142	141	142	142	141	142	141.75	0.4629
160	154	153	154	153	154	153	154	153	153.50	0.5345
170	164	164	164	165	164	164	163	164	164.00	0.5345
180	170	170	170	171	170	169	170	170	170.00	0.5345

Table 4.2: Percentage of relative error of the servomotor from 0˚to 180˚

Programmed Angles, degree/˚	Servomotor Average Readings, degree/˚	Servomotor Error Angles, degree/˚	Relative Error, Angles, degree/˚	True per cent relative error, Percentage/%
0	0.00	0.00	0.0000	0.00
10	8.00	2.00	0.2000	20.00
20	19.00	1.00	0.0500	5.00
30	28.63	1.38	0.0458	4.58
40	38.00	2.00	0.0500	5.00
50	45.88	4.13	0.0825	8.25
60	54.88	5.13	0.0854	8.54
70	64.00	6.00	0.0857	8.57
80	73.50	6.50	0.0813	8.13
90	81.63	8.38	0.0931	9.31
100	90.88	9.13	0.0913	9.13
110	101.63	8.38	0.0761	7.61
120	111.75	8.25	0.0688	6.88
130	122.50	7.50	0.0577	5.77
140	132.50	7.50	0.0536	5.36
150	141.75	8.25	0.055	5.5
160	153.50	6.50	0.0406	4.06
170	164.00	6.00	0.0353	3.53
180	170.00	10.00	0.0556	5.56

$$Sum\ of\ Relative\ Error$$

$$= 20 + 5 + 4.83 + 5 + 8.25 + 8.54 + 8.57 + 8.13 + 9.31 + 9.13 + 7.61 + 6.88$$

$$+5.77 + 5.36 + 5.5 + 4.06 + 3.53 + 5.56 = 127.97$$

$$Average\ of\ relative\ error\ in\ percentage$$

$$= \left(\frac{Sum\ of\ Relative\ Error}{Number\ of\ Data}\right) \times 100\%$$

$$= \left(\frac{127.97}{19}\right) \times 100\%$$

$$= 6.88\%$$

$$Average\ of\ standard\ deviation$$

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$$

$$= 0.44$$

The percentage of error of the servomotor is 6.88%. The errors are varied and a calibrated value is needed to lower the percentage of error. This means that the servomotor lack an average of 6.88% away from the actual value. Thus the calculation needs to add another averaged of 0.0688 of programmed value to reach the actual value. Since the standard deviation of the servomotor is 0.44 which is very low, thus is almost precise.

Calculate the calibrated value and total programmed value:

$$Calibrated\ Value\ = Programmed\ Angles \times\ 0.0688$$

$$Total\ Programmed\ Value = Programmed\ Angles + Programmed\ Angles \times\ 0.0688$$

4.7.1.1 Error of the Servomotor after Calibrated Value Added

After the calibrated value is calculated and added to the programming calculation equations, the experiment is repeated as the previous to check the error given by the servomotor. The data collected is recorded into Table 4.3 as shown below. Then the percentage of error is calculated again to prove there is a higher accuracy after added the calibrated value.

Table 4.3: Average servomotor angles compared to programmed angles

Programmed Angles, degree/°	Servomotor Readings								Average, degree/°	Standard Deviation
	1	2	3	4	5	6	7	8		
0	0	0	0	0	0	0	0	0	0.00	0.0000
10	11	11	11	10	11	10	11	11	10.75	0.4629
20	19	19	20	19	19	20	19	19	19.25	0.0000
30	30	30	30	30	30	30	30	30	30.00	0.5175
40	39	39	40	40	39	40	39	39	39.38	0.5345
50	49	49	49	50	50	49	50	50	49.50	0.0000
60	60	60	60	60	60	60	60	60	60.00	0.4629
70	69	69	69	68	69	69	68	69	68.75	0.3536
80	79	79	79	78	79	79	79	79	78.88	0.4629
90	89	89	89	89	88	88	89	89	88.75	0.4629
100	99	98	99	98	99	99	99	99	98.75	0.5345
110	109	109	108	108	108	108	109	109	108.50	0.0000
120	120	120	120	120	120	120	120	120	120.00	0.0000
130	130	130	130	130	130	130	130	130	130.00	0.5175
140	140	141	141	140	141	140	141	141	140.63	0.4629
150	153	154	153	154	153	153	153	153	153.25	0.5175
160	156	156	157	157	156	157	157	157	156.63	0.4629
170	166	166	167	167	166	166	166	166	166.25	0.5175
180	170	170	170	169	169	170	170	169	169.63	0.5345

Table 4.4: Percentage of relative error of the servomotor from 0˚to 180˚

Programmed Angles, degree/˚	Servomotor Average Readings, degree/˚	Servomotor Error Angles, degree/˚	Relative Error, Angles, degree/˚	True per cent relative error, Percentage/%
0	0.00	0.00	0.00	0.00
10	10.75	-0.75	0.075	7.50
20	19.25	0.75	0.0375	3.75
30	30.00	0.00	0.00	0.00
40	39.38	0.62	0.0155	1.55
50	49.50	0.5	0.01	1.00
60	60.00	0.00	0.00	0.00
70	68.75	1.25	0.017857	1.79
80	78.88	1.12	0.014	1.40
90	88.75	1.25	0.013889	1.39
100	98.75	1.25	0.0125	1.25
110	108.50	1.50	0.013636	1.36
120	120.00	0.00	0.00	0.00
130	130.00	0.00	0.00	0.00
140	140.63	-0.63	0.0045	0.45
150	153.25	-3.25	0.02167	2.17
160	156.63	3.37	0.021063	2.11
170	166.25	3.75	0.022059	2.21
180	169.63	10.37	0.057611	5.76

$Sum\ of\ Relative\ Error$

$= 7.5 + 3.75 + 0 + 1.55 + 1 + 1.79 + 1.4 + 1.39 + 1.25 + 1.36 +$

$0.45 + 2.17 + 2.11 + 2.21 + 5.76 = 33.69$

$Average\ of\ relative\ error\ in\ percentage$

$$= \left(\frac{Sum\ of\ Relative\ Error}{Number\ of\ Data}\right) \times 100\%$$

$$= \left(\frac{33.69}{19}\right) \times 100\%$$

$$= 1.77\%$$

$Average\ of\ standard\ deviation$

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$$

$$= 0.36$$

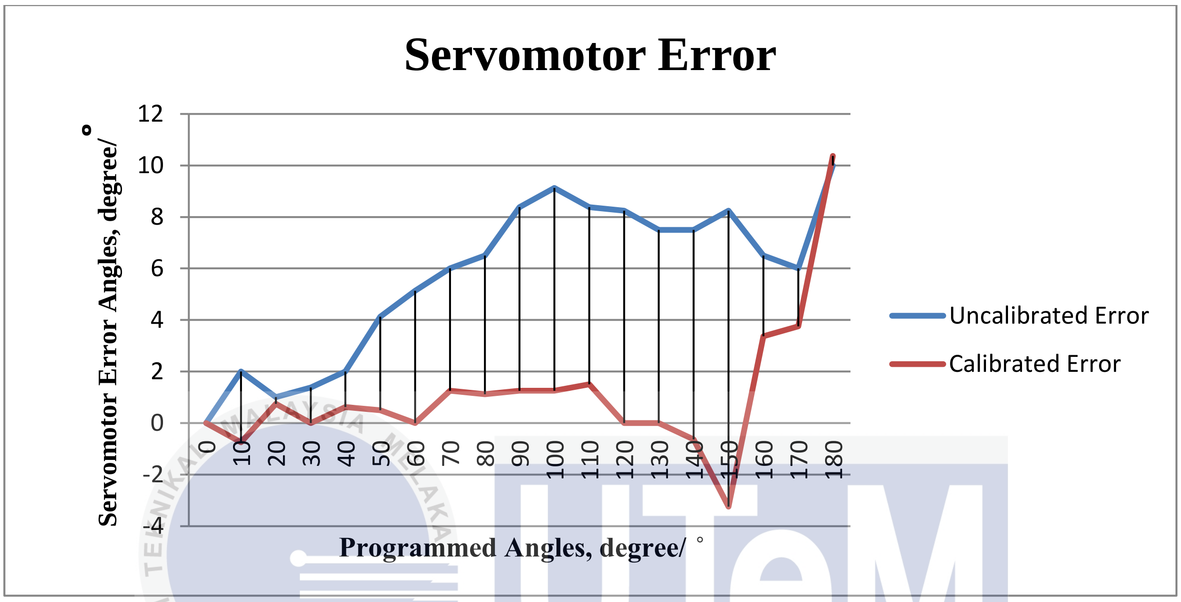


Figure 4.18: Servomotor error angles from 0˚to 180˚

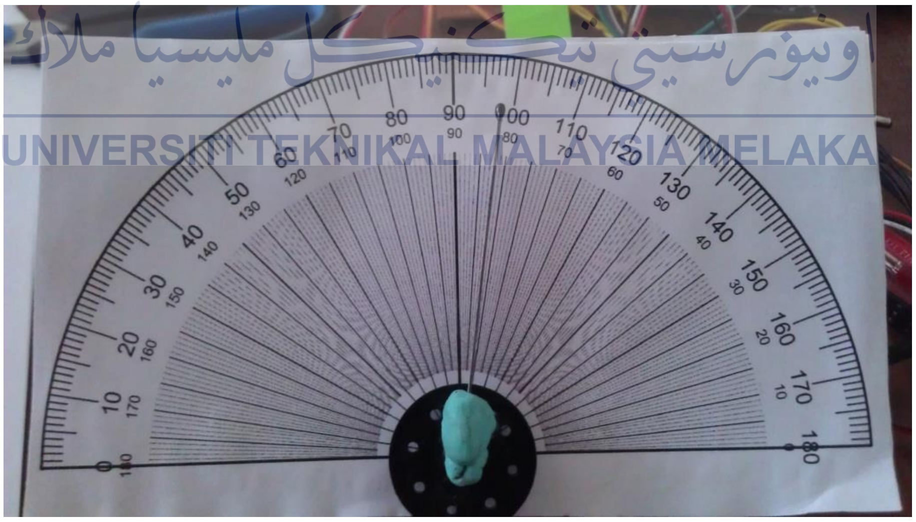


Figure 4.19: Servomotor at 90˚before include calibrated value

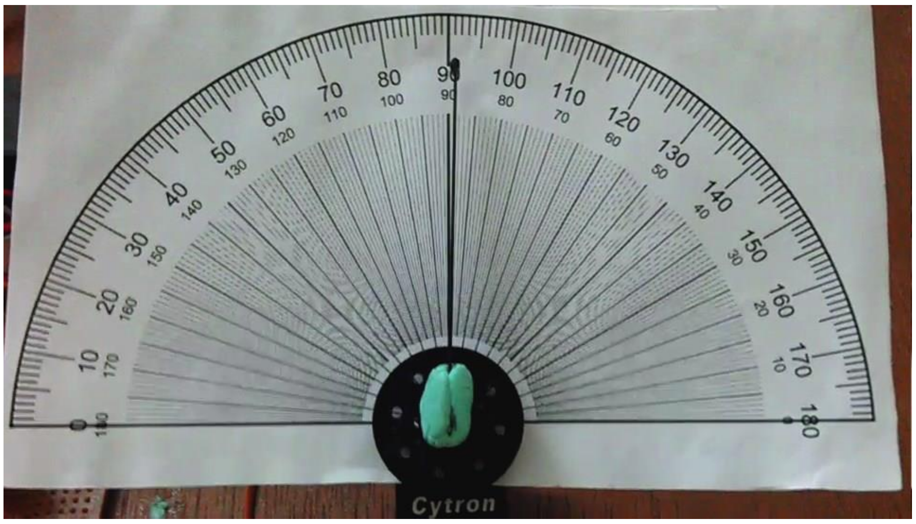


Figure 4.20: Servomotor at 90˚after included calibrated value

Before the calibrated value is introduced to the calculation, the servomotor gives less value of angle of rotation than the value programmed as shown at the Table 4.1 above. Figure 4.19 and Figure 4.20 have shown the different in angle before and after the addition of the calibrated value. A graph is plotted as shown in Figure 4.18. From the graph, the errors of the servomotor are increasing as the angle of the rotation of the servomotor increase. This is a random error that caused from the servomotor or any circuit itself.

After the calibrated value is introduced to the calculation, the percentage of error of the servomotor decreased from 6.88% to 1.77%. The percentage of error should be lower than 1.77% because the servomotor has a problem in getting to 180˚, not even more than 170˚. From Figure 4.18, the errors of the servomotor are almost in horizontal line until angle 160˚. Thus, there is a total of 74.27% improvement in the accuracy in the servomotor. Since the standard deviation of the servomotor for this experiment is 0.36 which is very low, thus is almost precise.

4.7.2 Experiment 2: Accuracy's Reliability on the Waist Experiment

Table 4.5 recorded the average of servomotor angles when the dummy hand and is compared with the programmed angles to find out the error of the servomotor and calculated the calibrated value to correct the error and increased the accuracy of the servomotor.

Table 4.5: Waist of the robotic arm's servomotor angles before the calibrated value

Waist									
x-axis, distance/mm	200	150	100	50	0	-50	-100	-150	-200
Programmed Angles, degree/°	36.87	53.13	66.42	78.46	90	101.54	113.58	126.87	143.13
1	35	50	62	72	81	94	106	118	135
2	34	51	61	73	82	94	106	119	136
3	35	50	61	72	82	95	107	119	135
4	35	50	62	72	72	95	107	119	135
5	34	51	61	73	73	94	106	118	136
Average, degree/°	34.6	50.4	61.4	72.4	78	94.4	106.4	118.6	135.4
Standard Deviation	0.5477	0.5477	0.5477	0.5477	0.5498	0.5477	0.5477	0.5477	0.5477
Servomotor Error Angles, degree/°	2.27	2.73	5.02	6.06	12	7.14	7.18	8.27	7.73
Relative Error, Angles, degree/°	0.0616	0.0514	0.0756	0.0772	0.1333	0.0703	0.0632	0.0651	0.0540
True per cent relative error, Percentage/%	6.16	5.14	7.56	7.72	13.33	7.03	6.32	3.12	5.40

$Sum\ of\ Relative\ Error$

$= 0.06158 + 0.051383 + 0.07558 + 0.077237 + 0.133333$

$+0.070317 + 0.063215 + 0.065185 + 0.054007 = 0.6518$

$Average\ of\ relative\ error\ in\ percentage$

$$= \left(\frac{Sum\ of\ Relative\ Error}{Number\ of\ Data}\right) \times 100\%$$

$$= \left(\frac{0.6518}{9}\right) \times 100\%$$

$$= 7.24\%$$

$$Average\ of\ standard\ deviation$$

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$$

$$= 1.11$$

Thus, the percentage of error of the servomotor is 7.24%. The result is almost the same as the previous experiment which is 6.88%, without include the calibrated value. Since the standard deviation of the servomotor for this experiment is 1.11 which is very low, thus is almost precise.

Table 4.6: Waist of the robotic arm’s servomotor angles after calibrated value

Waist									
x-axis, distance/mm	200	150	100	50	0	-50	-100	-150	-200
Programmed Angles, degree/°	36.87	53.13	66.42	78.46	90	101.54	113.58	126.87	143.13
1	37	53	65	78	89	100	113	126	141
2	37	53	65	78	89	101	113	126	142
3	37	53	66	77	90	101	113	127	141
4	37	53	65	77	90	101	113	127	141
5	37	53	65	78	89	100	113	127	142
Average, degree/°	37	53	65.2	77.6	89.4	100.6	113	126.6	141.4
Standard Deviation	0.00	0.00	0.4472	0.5477	0.5477	0.5477	0.00	0.5477	0.5477
Servomotor Error Angles, degree/°	-0.13	0.13	1.22	0.86	0.6	0.94	0.58	0.27	1.73
Relative Error, Angles, degree/°	0.0035	0.0024	0.0184	0.0109	0.0067	0.0093	0.0051	0.0021	0.0120
True per cent relative error, Percentage/%	0.35	0.245	1.84	1.10	0.67	0.93	0.51	0.21	1.21

$$Sum\ of\ Relative\ Error$$
$$= 0.00353+0.002447+0.018368+0.010961+0.006667+0.009257+0.005107+0.00212+0.012087 = 0.0706$$

$$Average\ of\ relative\ error\ in\ percentage$$
$$= \left(\frac{Sum\ of\ Relative\ Error}{Number\ of\ Data}\right) \times 100\%$$

$$= \left(\frac{0.0706}{9}\right) \times 100\%$$

$$= 0.78\%$$

$$Average\ of\ standard\ deviation$$

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$$

$$= 0.40$$

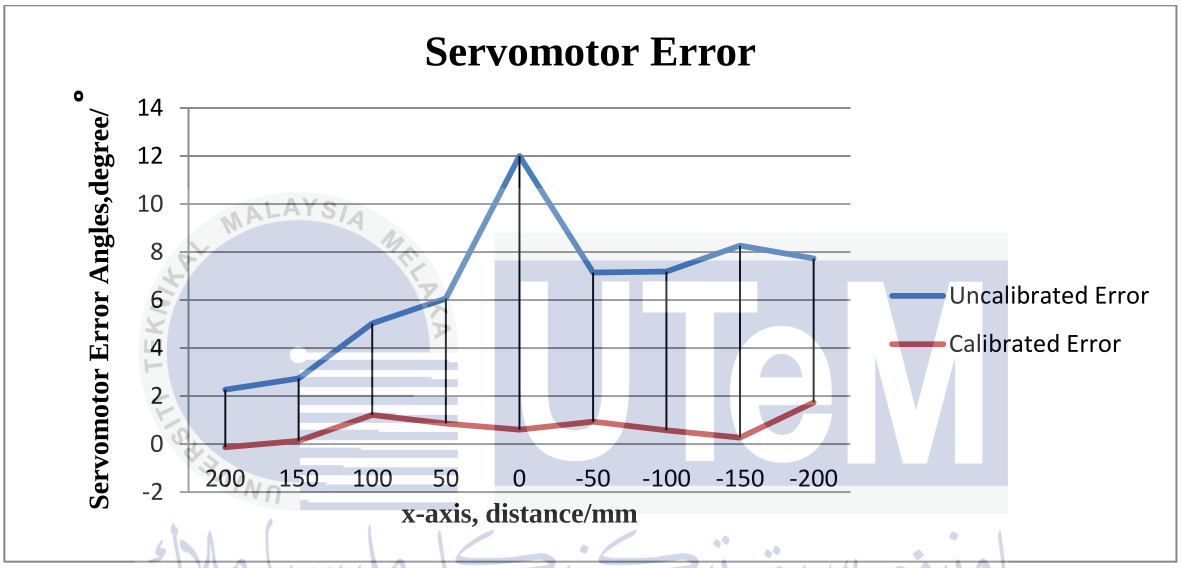


Figure 4.21: Servomotor error angles from -200mm to 200mm

The errors of the servomotor are increasing as the angle of the rotation of the servomotor increase before the calibrated value is introduced to the calculation. It has almost the same increasing line as the previous experiment graph plotted above.

The percentage of error of the servomotor decreased from 7.24% to 0.78%. Thus, there is a 90.19% improvement in the accuracy in the servomotor. The percentage of error is lower than 1.77% because in this experiment, the angle did not reach to angle more than 160˚ which bring more error to the calculation. Thus, there is a total of 90.19% improvement in the accuracy in the servomotor. From the Figure 4.21 the errors of the servomotor are almost in straight line. Thus, the accuracy is reliable for the servomotor after calibrated value is added to the calculation. Moreover, the joint movement at the waist are much more accurate. This calibrated value also added to the shoulder and elbow joints as it also using the same RC servomotor. Since the standard deviation of the servomotor for this experiment is 0.40 which is very low, thus is almost precise. As the conclusion, such accuracy of a robotic arm is acceptable for a human in using Leap Motion controller to control and of course, the higher the accuracy the better.

4.7.3 Experiment 3: Precision of Robotic Arm Repeatability Test

Table 4.7 recorded the average and standard deviation of the robotic arm for a repetitive test when the robotic arm is repeatedly moved from its initial position to the target position. Total of 15 samples are taken to calculate the average and standard deviation for precision test purpose.

Table 4.7: Average and standard deviation for the of the samples

Sample	x-coordinate	z-coordinate
1	0	2
2	2	3
3	0	2
4	2	1
5	0	0
6	1	1
7	2	1
8	0	3
9	2	2
10	2	1
11	0	2
12	2	3
13	2	3
14	1	2
15	2	2
Average, mm	1.20	1.87
Standard Deviation	0.94	0.92

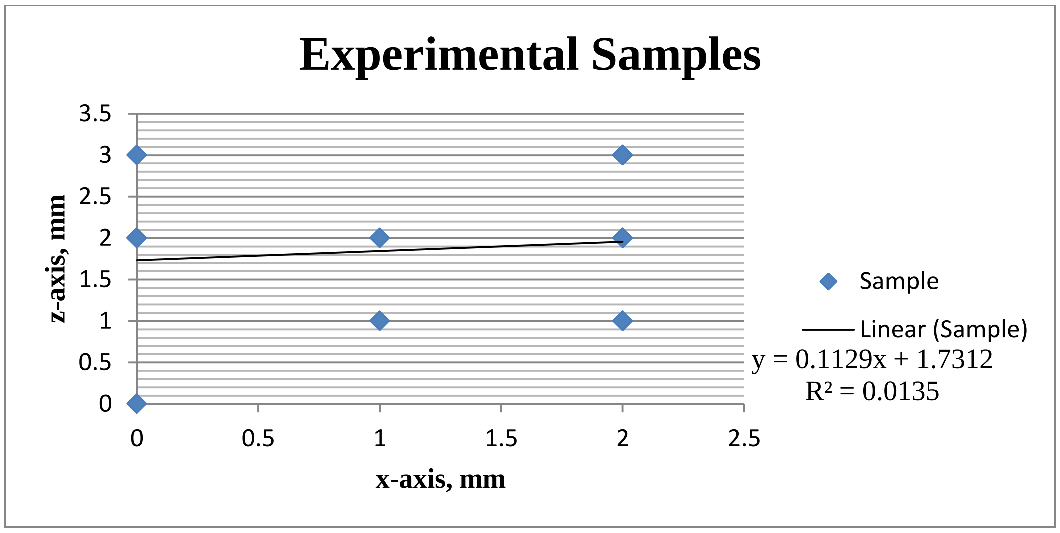


Figure 4.22: Position of the samples

By referred to the Table 4.7 above, the standard deviation of the *x*-coordinate and *z*-coordinate is 0.94 and 0.92 respectively. Since the standard deviation is low, means it is high in precision. The average error of *x*-coordinate and *z*-coordinate is 1.20mm and 1.87mm respectively, which is almost small error in precision. The gradient of the line shows 0.1129 means that in the data, the samples have higher errors in *x*-coordinate than the *z*-coordinate. As the conclusion, it may not be acceptable to perform as an industrial robot process, but for Leap Motion controlled robotic arm, is almost negligible. Human unable to detect the 2mm error of the robotic, thus it is suitable for human to use as normal pick and place application.

Throughout the experiments, the systematic error has been eliminated as much as possible. Such error like parallax error is avoided when taking data from the protractor. The needle and the eye observant must be in the right position and the needle must parallel to the scale of the protractor. As the systematic error reduced, the accuracy of the measurements increased. Nevertheless, the measurement limitation, the sensitivity of the instrument is one of the causes of having errors in measurement. On the other hand, based on the observation, the error is caused by human error also. From the methodology, many steps can be caused by the human errors. From drawing to marking, due to the limitation of the equipment provided, there are several human errors caused the data to differ from the expected results and made errors.

As the random error, the Experiment 1 to Experiment 3 can avoid this error by taking measurement repeatedly and calculate the average data taken. However, the Leap

Motion controller performance is light dependent. Bright light sources or reflective sources will impact performance. A few precautions are taken to avoid the effect on the Leap Motion performance as mentioned above. Lastly, during all experiments, the LED indicator of the Leap Motion controller must be always in green colour to avoid any unnecessary and unwanted data taken.

4.8 Completion of the Robotic Arm Prototype

Figure 4.23 shows the completion of the robotic arm according to the SolidWorks drawing. The whole robotic arm, Arduino Uno and the circuits are fixed on a big plank where the pick and place task is able to perform on it. Each parts of the project hardware is labeled in Figure 4.23.

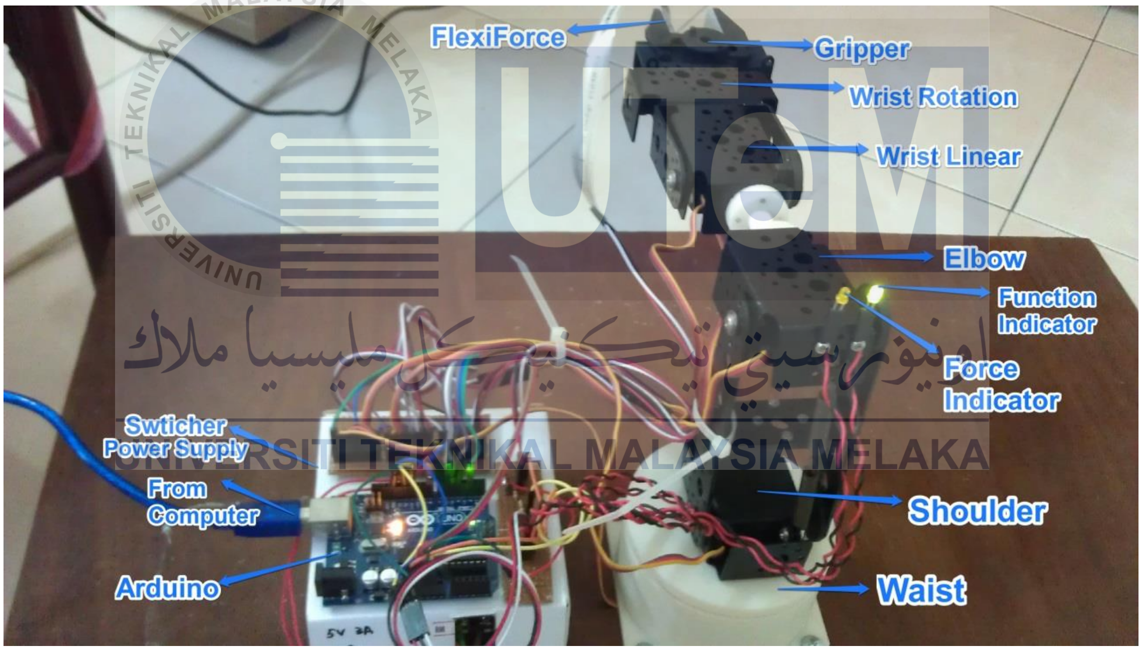


Figure 4.23: Completion of Robotic Arm Prototype

4.8.1 User interfaces

4.8.1.1 Robotic Arm System Function Indicator

A simple LED indicator is added to indicate that the robotic arm system is functioning well. The LED blinked for every 300 milliseconds. If the LED indicator stop blink, means the programming is stopped or system malfunction. The programming program is as shown below:

```
led.strobe(300);
```

4.8.1.2 Gripper Forced Indicator

Another user interfaces is added to the programming and hardware. In the gripper wise, the griper is added feature to avoid the gripper to grip to tight of an object. A FlexiForce force sensor is attached at the gripper to sense the force exerted on the object by the gripper is not too tight as referred to Figure 4.23. When the gripper's force exerted on the object at the certain pressure, LED red lighted up. This is to make sure the gripper able to grip firmly when it comes to fragile object that needed to be gripped as shown at programming program as below:

```
sensor = new five.Sensor({
      pin: "A0", freq: 250
  });

  board.repl.inject({
      sensor: sensor
  });

  sensor.scale([0, 100]).on("data", function () {
      if (this.value > 0.6) {
          led1.on();
      }
      else {
          led1.off();
      }
```

4.8.1.3 Robotic Arm Initial Position

From the lesson learned during the survey taken from the participants, the robotic arm has an initial position. This initial position is when there is no hand sensed by the Leap Motion controller and during this occasion, the robotic arm is in 90˚. Once hand is above the middle of the Leap Motion controller, the robotic arm started to move according to the hand motion and once the hand is remove from the Leap Motion controller, the robotic arm back to the initial position.

```
//Initial position of the robotic arm when no hand is detected by the controller
       else if (frame.hands.length == 0) {

           moveWaist = 90 + 90*0.0688;
           moveShoulder = 90 + 90*0.0688
           moveElbow = 90 + 90*0.0688;
           moveSupiandPro = 90 + 90*0.0688;
           moveFlexandExt = 90 + 90*0.0688;
```

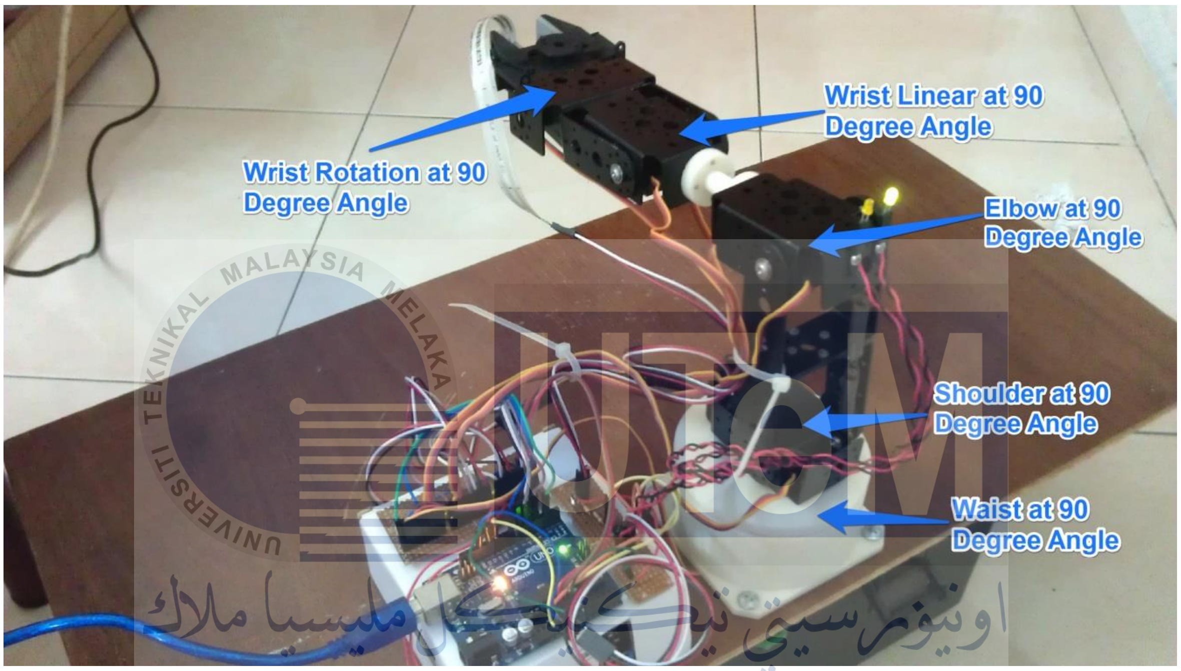

Wrist Linear at 90 Degree Angle
Wrist Rotation at 90 Degree Angle
Elbow at 90 Degree Angle
Shoulder at 90 Degree Angle
Waist at 90 Degree Angle

Figure 4.24: Initial position of the Robotic Arm

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

As conclusions, the robotic arm which controlled by Leap Motion controller able to perform pick and place task by just using hand and fingers gestures. This robotic arm structure is designed easily by using the SolidWorks software in 3D dimensions. However to control the robotic arm, programming program in JavaScript language is edited and created by using Visual Studio. Moreover, to able to get the data from the every millisecond that the Leap Motion Controller captured, a deep studied about the Leap Motion controller and JSON Viewer is very important initial step to make. In addition, the robotic arm is controlled by the microcontroller called Arduino Uno Rev which has the Standard Firmata protocol to communicate with the host computer. The lecturer that owns the robotic arm in the Robotic Laboratory is kind enough to lend The Robotic arm in the robotic laboratory is used on the research on Leap Motion controller. The robotic arm has 5 degree of freedom, which has the most basic degree of freedom for a human arm. Besides, the robotic arm gripper as the tool, thus is very suitable for the pick and place task. Next the application on the forward and inverse kinematics are derived and applied to the program for the execution. Many trial and errors and also program editing are done to meet the right motion of the robotic arm from the human hand and fingers gestures.

Next, the accuracy and performance of the servomotor are improved to perform task as precise as own hand and fingers gestures. After conducted a few experiment and analysis, the servomotor at each joint of the robotic arm has decreased dramatically. Actually, the robotic arm’s joint should use a motor with encoder. Unfortunately, the budget for construct a better robotic arm is insufficient because the price is expensive. Moreover, a few user interfaces are added for the operator. There are two LED indicators added. One is to ensure the robotic arm system is functioning well and the other is to prevent when the gripper gripped on an object too tight. The gripper is modified as it able to grip firmly when it comes to fragile object that needed to be gripped.

5.2 Recommendations and Suggestions

As recommendation, the limitation of the Leap Motion is it only detects hands and fingers in 2D plane. Sometimes the pick and place task required more than just 2D plane such as an overlapping of two fingers on the Leap Motion controller caused the Leap Motion controller to detect only one finger. However, this is the limitation of Final Year Project, thus further research on Leap Motion controller has reach its limitation. Nevertheless, this project can be improved and developed into a telerobotics system which the robotic arm can controlled from a distance, using wireless connections which most possibly using Wi-Fi or radio frequency as the medium to transfer data and Xbee as the hardware.

Last but not least, as this prototype is made of RC servomotors which without encoder, thus no feedback is available. The feedback is important as it can use feedback controller such as proportional-integral-derivative controller or (PID controller) and fuzzy logic controller to not just increased the accuracy of the prototype, but also improve overall performance. Thus, this Leap Motion controlled robotic arm is able to compete with existing robotic arm in industrial field.

REFERENCES

[1] O. Fukuda *et al*., "An EMG Controlled Robotic Manipulator Using Neural Networks," *IEEE International Workshop on Robot and Human Communication*, pp 442-448, 1997.

[2] P.T. Kidd, "Design of human-centered robotic systems," *IEEE International Conference on Robotics and Automation*, France, 1992, pp. 225-241.

[3] R. Parasuraman *et al*., "A model for types and levels of human interaction with automation," *IEEE trans. Syst., Man, Cybern. C*, vol. 297, no. 3, pp. 286-297, May 2000.

[4] J. L. Raheja *et al*., " Real-Time Robotic Hand Control using Hand Gestures," *Second International Conference on Machine Learning and Computing*, April 2010, pp. 12-15.

[5] O. Fukuda *et al*., "A Human-Assisting Manipulator Teleoperated by EMG Signals and Arm Motions," *IEEE Transactions on Robotics and Automation*, vol. 19, NO. 2, April 2003.

[6] F. Lathuiliere and J. Y. Herve, "Visual Hand Posture Tracking in a Gripper Guiding Application," in *International Conference on Robotic & Automation*, San Francisco, CA, 2010.

[7] I. Baran and Assoc. Prof. Dr. Mehmet, "Development of a Robotic-Arm Controller by using Hand Gesture Recognition," *2012 International Symposium on Innocations in Intelligent System and Applications*, 2012.

[8] D. Marinos *et al*., "Prototyping Natural Interactions in Virtual Studio Environments by Demonstration – Combining Spatial Mapping with Gesture Following," *IEEE Symposium in 3D User Interfaces 2013*, March 2013.

[9] D. Ryu *et al*., "T-less: a Novel Touchless Human-Machine Interface based on Infrared Proximity Sensing," *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5220-5225, October 2010.

[10] J. H. Kim *et al*., "3-D Hand Motion Tracking and Gesture Recognition Using a Data Glove," *IEEE International Symposium on Industrial Electronics (ISIE 2009)*, pp. 1013-1018, July 2009.

[11] L. E. Kavraki. (2013, October 31). *Protein Inverse Kinematics and the Loop Closure Problem*. [Online]. Available: http://cnx.org/content/m11613/latest/

[12] Y. J. Tham. (2013, September 25). *How to Build a Robotic Arm that Tracks Your Hand Movements*. [Online]. Available : http://yujiangtham.com/2013/07/22/how-to-build-a-robotic-arm-that-tracks-your-hand-movements/

[13] Leap Motion Developer 2013. (2013, September 20). *Hand Class Reference* [Online]. Available at: https://developer.leapmotion.com/documentation/Languages/Java/API/classcom_1_1leapmotion_1_1leap_1_1_hand.html [accessed 20 September 2013]

[14] Leap Motion Developer 2013 (2013, September 25). *API Overview* [Online]. Available : https://developer.leapmotion.com/documentation/javascript/devguide/Leap_Overview.html?highlight=controller%20cartesian [accessed 20 September 2013]

[15] Leap Motion Developer 2013 (2014, May 20). *Leap Motion Support* [Online]. Available : https://support.leapmotion.com/entries/43733443-Operating-Environment [accessed 20 May 2014]

APPENDICES

APPENDIX A – Survey

Leap motion – Controlled Robotic Arm

1. Do you attempt any activity in controlling a machine using remote control?

☐ Yes

☐ No

2. If yes, do you find it a bit difficult to control sometimes?

☐ Yes

☐ No

3. Do you heard before about Leap Motion?

☐ Yes

☐ No

4. Is this the first time attempt to control a machine using you hand and fingers gestures?

☐ Yes

☐ No

5. Are you able to complete the pick and place task given after 2 trials?

☐ Yes

☐ No

6. How do you feel about this Leap Motion to control the robotic arm?

☐ Complicated

☐ Normal

☐ Interesting

☐ Easy

☐ Tough

☐ Same as a remote control

7. Which way is easier to control a machine? Using a remote control or own hand and fingers?

☐ Remote Control

☐ Hand and Fingers

8. Do you have any problem in controlling this robotic arm using Leap Motion?

☐ Yes

☐ No

If yes please state the problem(s).

____
____
______

9. Do you have any suggestion in improving this robotic arm?

☐ Yes

☐ No

If yes please state your suggestion(s).

____
____
______

10. Do you think this system is able to commercialize in the market for laboratory use?

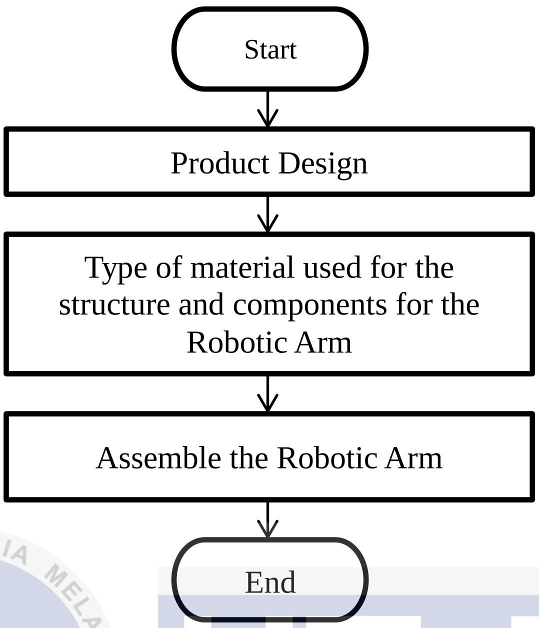
☐ Yes

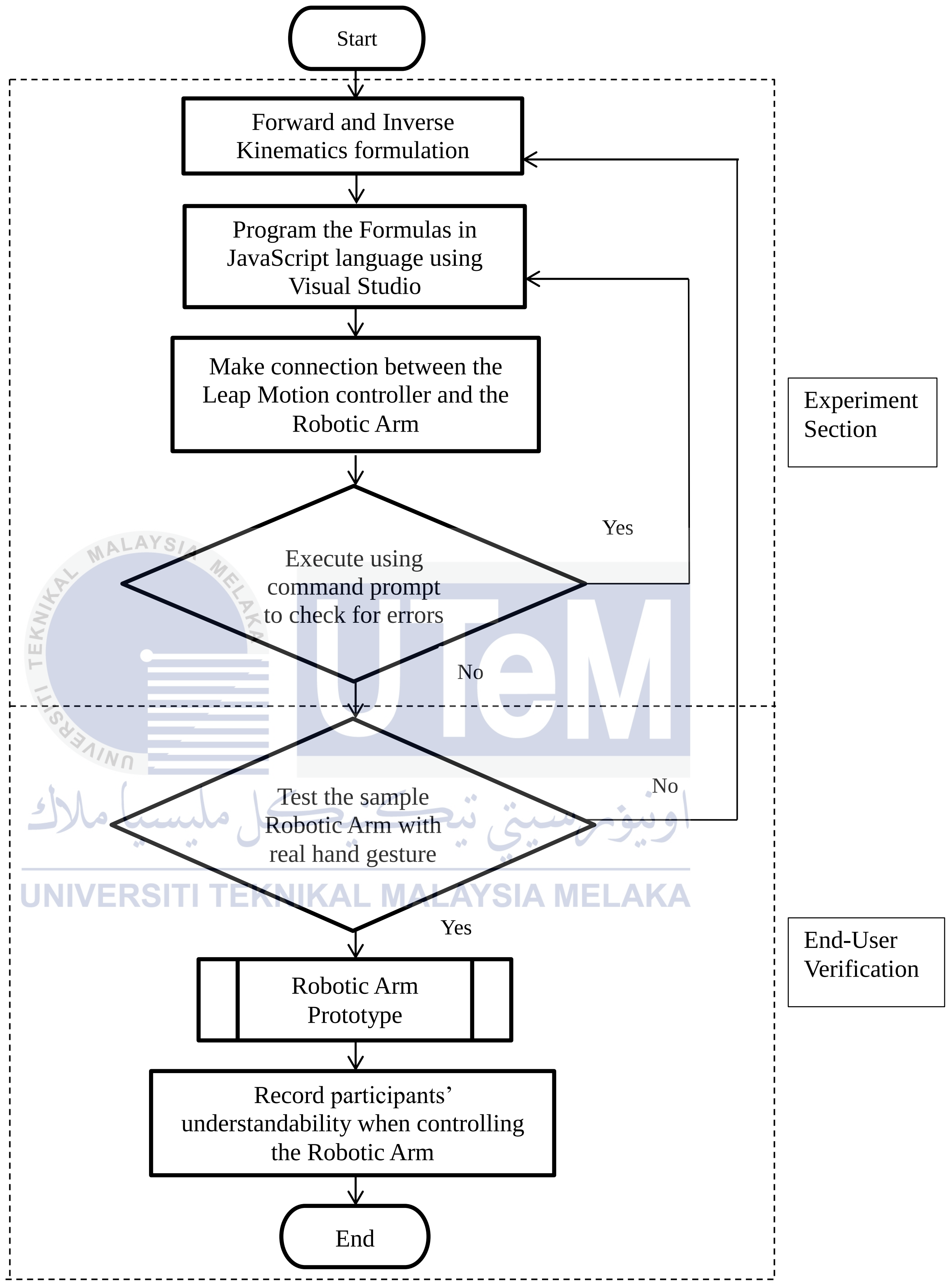
☐ No

Available in https://docs.google.com/forms/d/1_zC1T_Ev8PxDAMnB9ANwh3AsH9Dlb73BSkkj7wQuIkI/viewform

APPENDIX B –Project and Robotic Arm Prototype Flow Chart

Robotic Arm Prototype

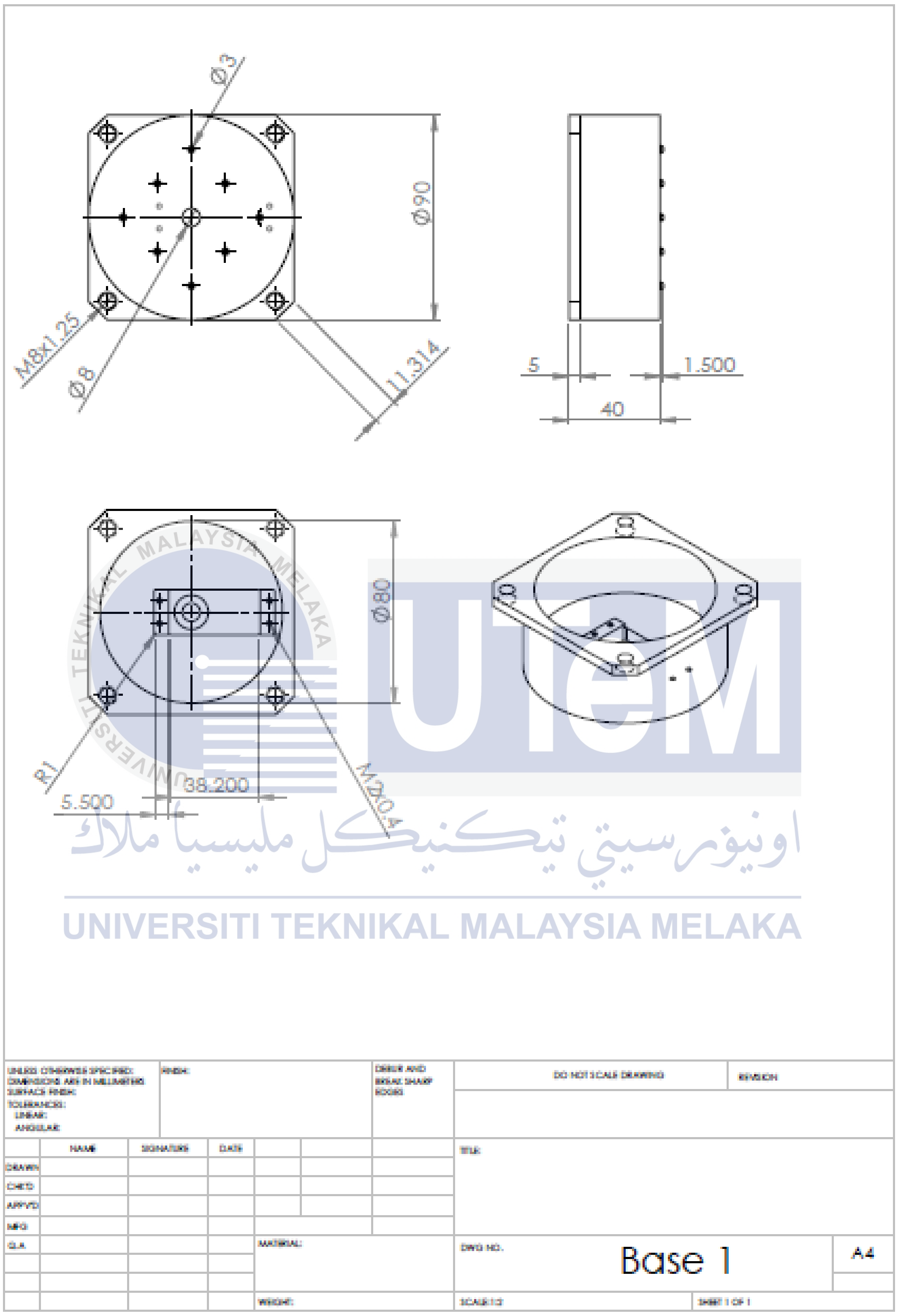

Start
Product Design
Type of material used for the structure and components for the Robotic Arm
Assemble the Robotic Arm
End


Start
Forward and Inverse Kinematics formulation
Program the Formulas in JavaScript language using Visual Studio
Make connection between the Leap Motion controller and the Robotic Arm
Execute using command prompt to check for errors
Yes
No
Test the sample Robotic Arm with real hand gesture
No
Yes
Robotic Arm Prototype
Record participants' understandability when controlling the Robotic Arm
End
Experiment Section
End-User Verification

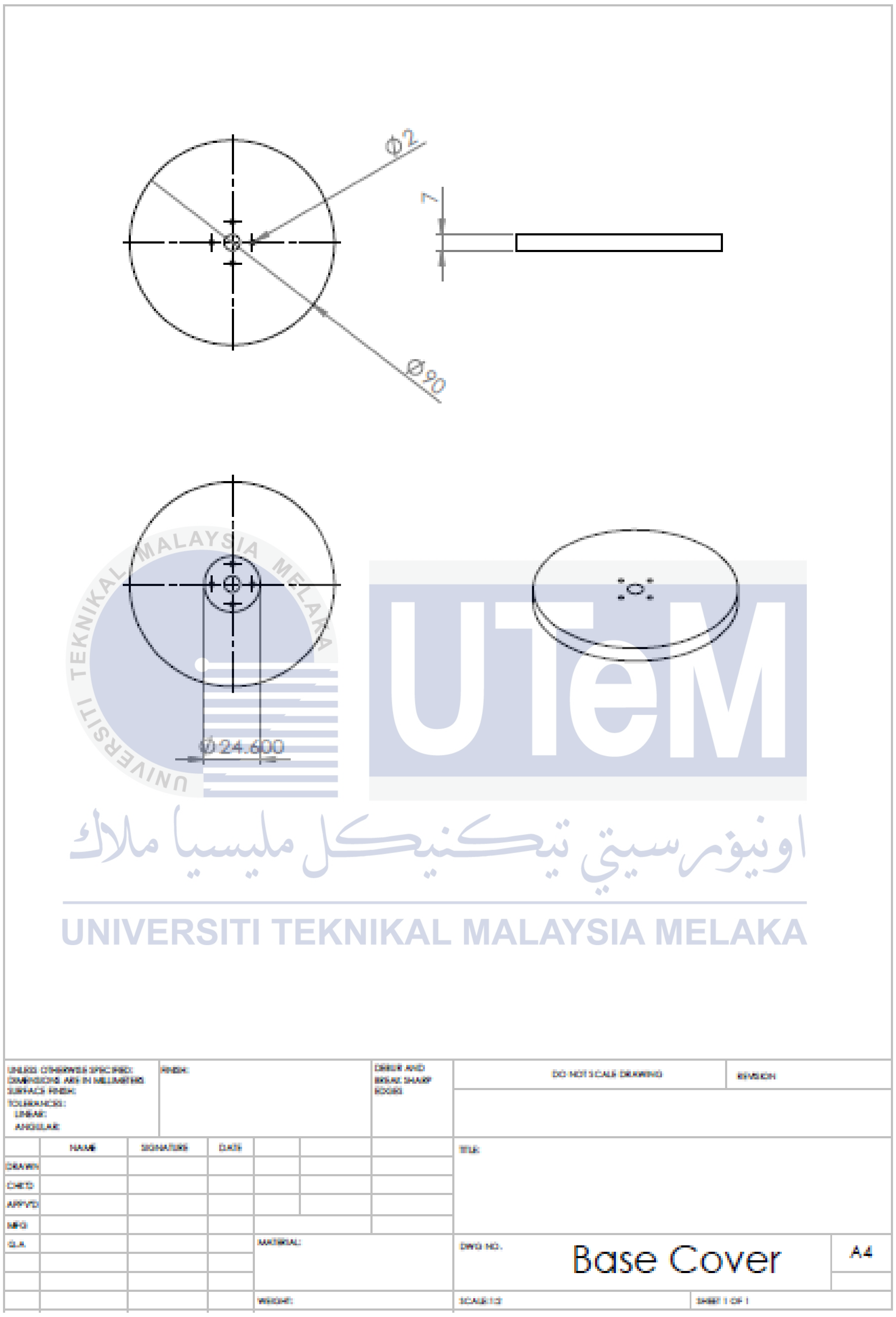
APPENDIX C – Gantt Chart

	Month									
Task	9	10	11	12	1	2	3	4	5	6
Journal finding	🟦									
Papers and journals review	🟦	🟦								
Software development	🟦	🟦								
Introduction of proposed project		🟦								
Forward and inverse kinematics calculation		🟦								
JavaScript programming		🟦	🟦							
Testing, analysis, trial and error		🟦	🟦							
SolidWorks drawing			🟦							
Early Result			🟦							
FYP 1 report writing			🟦							
Progress report submission to supervisor			🟦							
Submission report to panel 1 and panel 2			🟦							
Presentation on FYP 1			🟦	🟦						
Submission corrected report to supervisor				🟦						
Purchase parts and materials			🟦	🟦						
Robotic arm assembly				🟦	🟦					
Experiment done on robot					🟦	🟦	🟦			
Analyze the results							🟦	🟦		
FYP 2 report writing								🟦	🟦	
FYP 2 draft report submission to supervisor									🟦	🟦
Presentation FYP 2										🟦
FYP 2 report correction										🟦
Submission full report										🟦

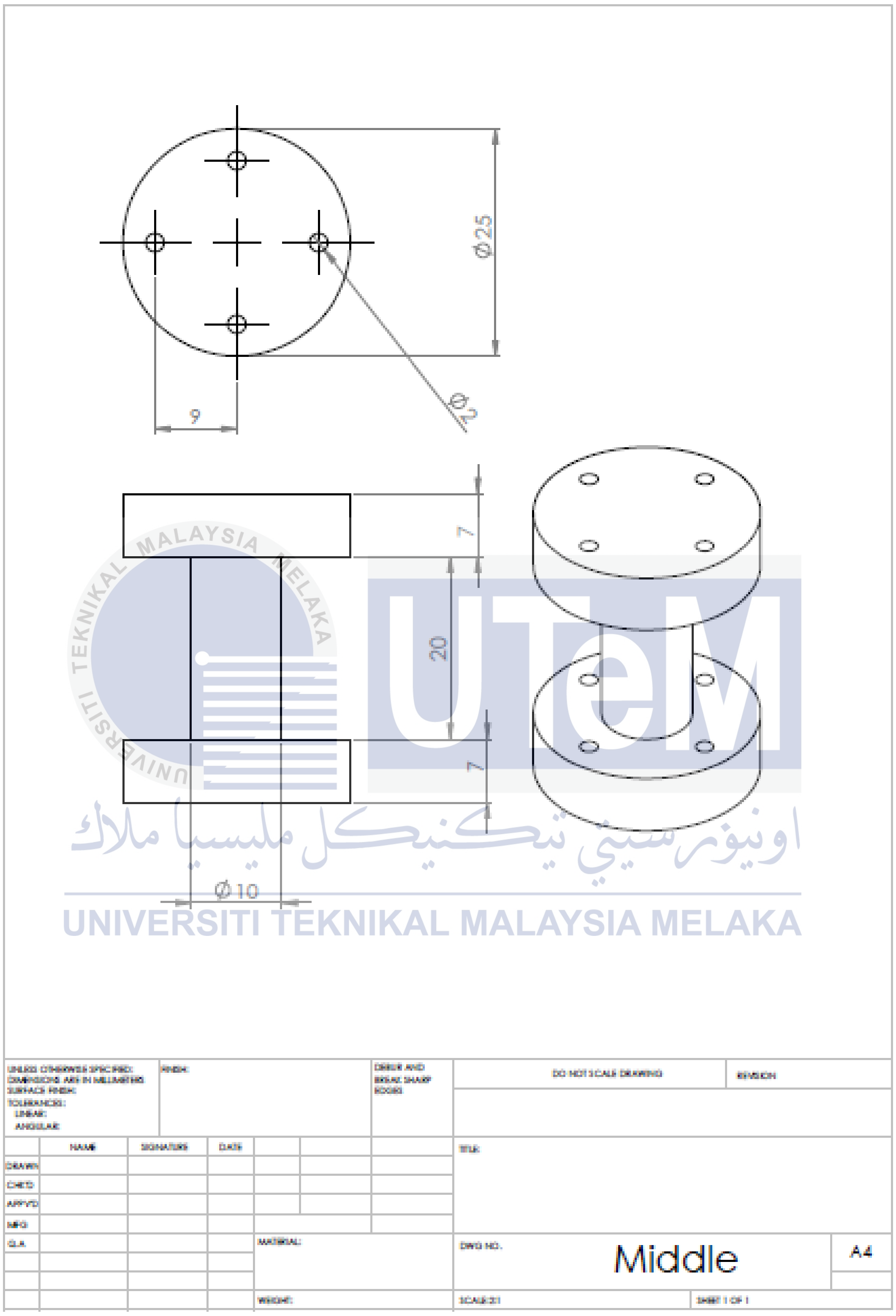
APPENDIX D – SolidWorks Base Drawing



APPENDIX E – SolidWorks Base Cover Drawing



APPENDIX F – SolidWorks Middle Cover Drawing



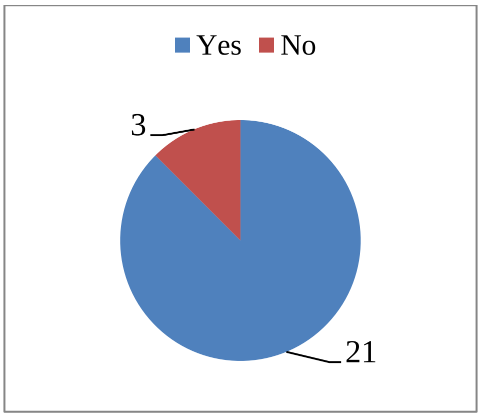
APPENDIX G – Program controlling robotic arm in JavaScript language

```
//Title          :   LEAP MOTION CONTROLLED ROBOTIC ARM
//Create by      :   LEE JUN WEI
//Matrix Number  :   B011010079
//Contact HP.no  :   010-2478439
//Course         :   Bachelor of Mechatronic Engineering (BEKM)
//University     :   UNIVERSITI TEKNIKAL MALAYSIA MELAKA
//Faculty        :   FAKULTI KEJURUTERAAN ELEKTRIK
//Supervised by  :   VINCENT LOI WEI SEN
//Judged by      :   DR ZAKI
//Last Edited    :   15/5/2014


var webSocket = require('ws'),
    ws = new webSocket('ws://127.0.0.1:6437'),
//gestures = require('./node_modules/leapjs/lib/gesture'),
//LeapFrame= require('leapFrame'),
//Joint = require('joint'),
//board = new five.Board(),
    five = require('johnny-five'),
    Leap = require('leapjs'),
    frame, moveWaist, moveShoulder, moveElbow, moveSupiandPro, moveFlexandExt,
moveClaw,
    servoWaist, servoShoulder, servoElbow, servoSupiandPro, servoFlexandExt,
servoClaw,

    minClawDistance = 15,
    boardOptions = { port: 'COM7' },
    board = new five.Board(boardOptions);

board.on('ready', function () {
    led = new five.Led(12);
    led1 = new five.Led(11);

    // "strobe" the led in 300ms on-off phases
    led.strobe(300);

    //Force sensor at pin A0
    sensor = new five.Sensor({
        pin: "A0", freq: 250
    });

    board.repl.inject({
        sensor: sensor
    });

    sensor.scale([0, 100]).on("data", function () {
        if (this.value > 0.6) {
            led1.on();
        }
        else {
            led1.off();
        }
    });
    ws.on('message', function (data, flags) {
        frame = JSON.parse(data);       //Getting data from the JSON
        if (frame.hands.length >0) {
            handPosition = frame.hands[0].palmPosition;
            handFlexandExt = frame.hands[0].palmNormal[2];      //Flexion and 
Extension of the hand
            handSupiandPro = frame.hands[0].palmNormal[0];      //Supination and 
Pronation of the hand

            // Restrict certain inputs to prevent physical damage
            // These values can be changed depending on
            if (handPosition[0] > 250) handPosition[0] = 250;
            if (handPosition[0] < -250) handPosition[0] = -250;
            if (handPosition[1] < 120) handPosition[1] = 120;
            if (handPosition[1] > 415) handPosition[1] = 415;
            if (handPosition[2] > 200) handPosition[2] = 200;
            if (handSupiandPro > 0.833) handSupiandPro = 0.833;
            if (handSupiandPro < -0.833) handSupiandPro = -0.833;
            if (handFlexandExt > 0.8) handFlexandExt = 0.8;
            if (handFlexandExt < -0.6) handFlexandExt = -0.6;
            console.log(handPosition[1]);

            // Calculate all of the movement angles
            angles = calculateInverseKinematics(handPosition[0], handPosition[1], 
handPosition[2]);
            moveWaist = toDegrees(angles.theta5);
            moveShoulder = toDegrees(angles.theta1);
            moveElbow = 45 + toDegrees(angles.theta2);
            moveSupiandPro = 20 + toDegrees(angles.theta3);
            moveFlexandExt = 10 + toDegrees(angles.theta4);
        }

        // Finger distance (of two fingers only) controls the end effector
        if (frame.pointables.length > 1) {
            f1 = frame.pointables[0];
            f2 = frame.pointables[1];
            fingerDistance = distance(f1.tipPosition[0], f1.tipPosition[1], 
f1.tipPosition[2], f2.tipPosition[0], f2.tipPosition[1], f2.tipPosition[2]);
            moveClaw = ((fingerDistance / 1.2) - minClawDistance) * 4;
        }

        //Initial position of the robotic arm when no hand is detected by the 
controller
        else if (frame.hands.length == 0) {

            moveWaist = 90 + 90 * 0.0688;
            moveShoulder = 90 + 90 * 0.0688;
            moveElbow = 90 + 90 * 0.0688;
            moveSupiandPro = 105 + 90 * 0.0688;
            moveFlexandExt = 90 + 90 * 0.0688; // need adjustment
        }
    })
});

//define pins on Arduino Uno
board.on('ready', function () {
    servoWaist = new five.Servo(2);
    servoElbow = new five.Servo(8);
    servoShoulder = new five.Servo(4);
    servoSupiandPro = new five.Servo(5);
    servoFlexandExt = new five.Servo(6);
    servoClaw = new five.Servo(7);

    // Initial positions of the robot arm
    servoWaist.center();
    servoShoulder.center();
    servoElbow.move(45);
    servoSupiandPro.move(90);
    servoFlexandExt.move(90);
    servoClaw.move(100);

    // Move each component
    this.loop(40, function () {
        if (!isNaN(moveShoulder) && !isNaN(moveElbow)) {
            servoShoulder.move(moveShoulder);
            servoElbow.move(moveElbow);
            servoWaist.move(moveWaist);
            servoSupiandPro.move(moveSupiandPro);
            servoFlexandExt.move(moveFlexandExt);
        }
        //limit the angle and distance of the joint
        else {
            console.log("Shoulder/Elbow NaN value detected.");
        }
        if (moveWaist >= 0 && moveWaist <= 180) {
            servoWaist.move(moveWaist);
        }
        if (moveSupiandPro >= 0 && moveSupiandPro <= 180) {
            servoSupiandPro.move(moveSupiandPro);
        }
        if (servoFlexandExt >= 0 && servoFlexandExt <= 180) {
            servoFlexandExt.move(moveFlexandExt);
        }
        if (moveClaw >= 0 && moveClaw <= 250) {
            servoClaw.move(moveClaw);
        }

        console.log("Waist: " + Math.floor(moveWaist) + "\tShoulder: " +
Math.floor(moveShoulder) + "\tElbow: " + Math.floor(moveElbow) + "\tClaw: "
        + Math.floor(moveClaw) + "\tWristTurn: " + Math.floor(moveSupiandPro) +
"\tWristTilt: " + Math.floor(moveFlexandExt));
    });
});

function calculateInverseKinematics(x, y, z) {
    // Adjust the input values
    x = x;
    y = y / 1.6;
    z = -z / 1.3;

    // Adjust the values to mesh with your desired input range
    var l1 = 150;
    var l2 = 150;

    // Inverse kinematics equations
    var t1 = Math.acos((square(z) + square(y) - square(l1) - square(l2)) / (2 * l1
* l2))
            + Math.acos((square(z) + square(y) - square(l1) - square(l2)) / (2 * l1
* l2)) * 0.0688;
```

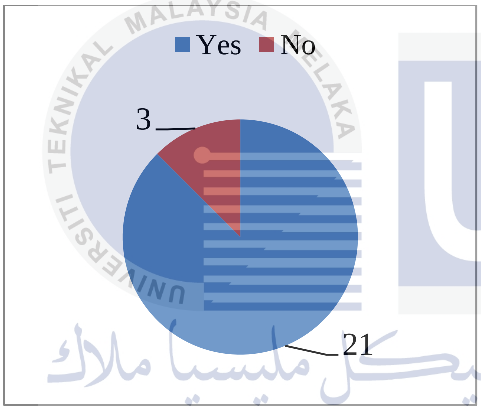
```
    var t2 = Math.asin(((l1 + l2 * Math.cos(t1)) * y - l2 * Math.sin(t1) * z) /
(square(l1) + square(l2) + 2 * l1 * l2 * Math.cos(t1)))
             + Math.asin(((l1 + l2 * Math.cos(t1)) * y - l2 * Math.sin(t1) * z) /
(square(l1) + square(l2) + 2 * l1 * l2 * Math.cos(t1))) * 0.0688;

    var t3 = Math.acos((handSupiandPro) * 1.2);
    if (handFlexandExt > 0) {
        var t4 = Math.acos((handFlexandExt) * 1.3);                // increase the
input data from JSON
    }
    else if (handFlexandExt < 0) {
        var t4 = Math.acos((handFlexandExt) * 1.45);               // increase the
input data from JSON
    }
    var t5 = Math.acos(x / 250) + Math.acos(x / 250) * 0.0688;

    //console.log("[DEBUG] THETA1: " + toDegrees(t1) + "\tTHETA2: " + toDegrees(t2));
    return {
        theta1: t1,
        theta2: t2,
        theta3: t3,
        theta4: t4,
        theta5: t5
    }
}

/*
* Utility Functions
*/

function distance(x1, y1, z1, x2, y2, z2) {
    return Math.sqrt(square(x2 - x1) + square(y2 - y1) + square(z2 - z1));
}

function square(x) {
    return x * x;
}

function toDegrees(r) {
    return r * 57.2957795;
}
```

APPENDIX H – Summary of the Survey

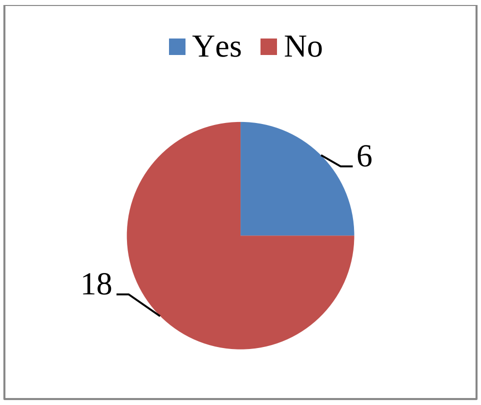
1. Do you attempt any activity in controlling a machine using remote control?



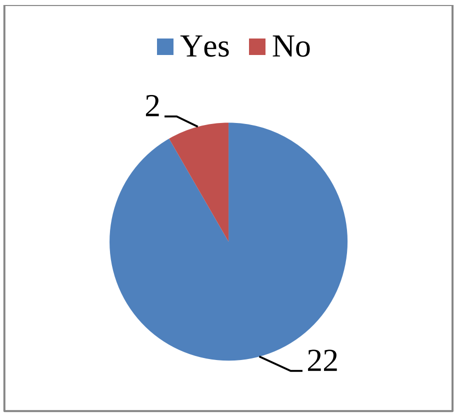
2. If yes, do you find it a bit difficult to control sometimes?



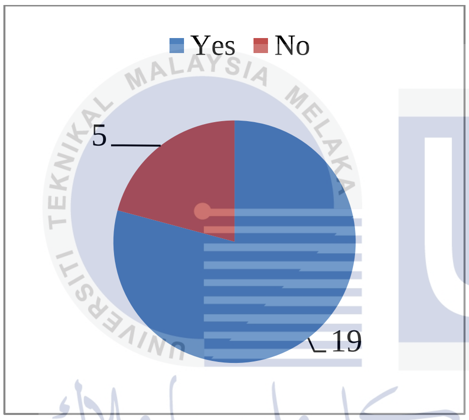
3. Do you heard before about Leap Motion?



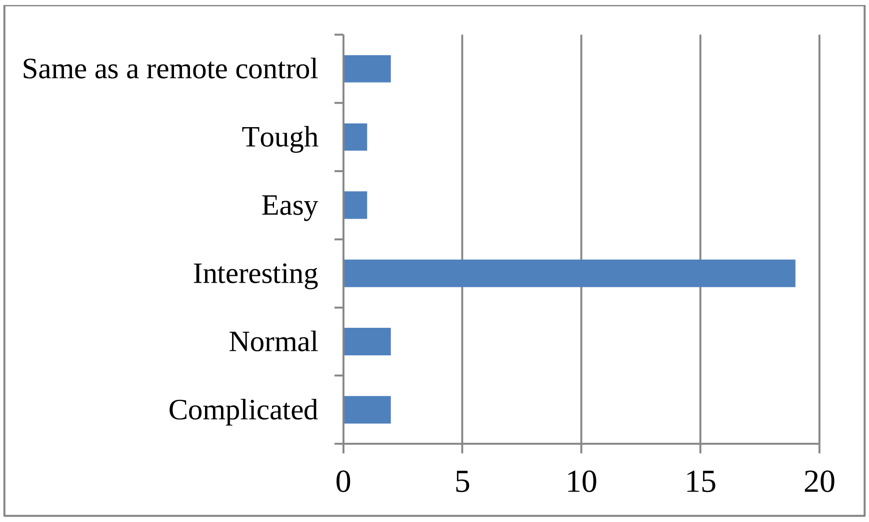
4. Is this the first time attempt to control a machine using your hand and fingers gestures?



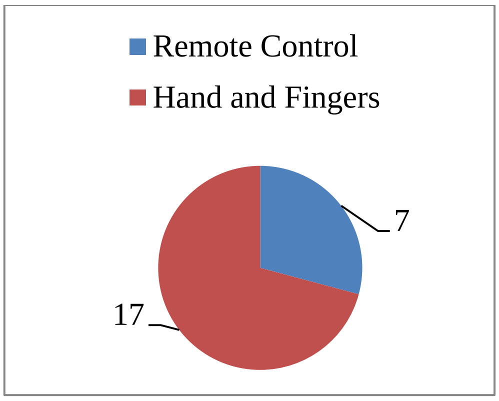
5. Are you able to complete the pick and place task given after 2 trials?



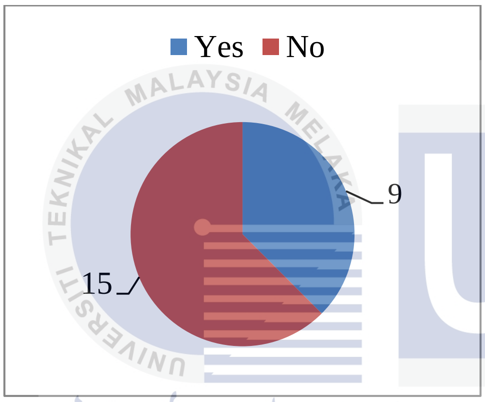
6. How do you feel about this Leap Motion to control the robotic arm?



7. Which way is easier to control a machine? Using a remote control or own hand and fingers?



8. Do you have any problem in controlling this robotic arm using Leap Motion?



If yes please state the problem.

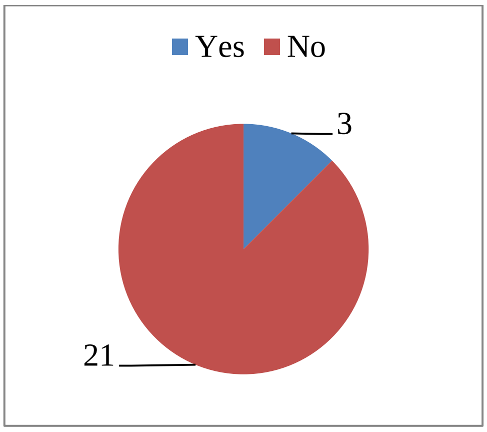
a. We need to compensate the robotic arm's motion.

b. Maybe is the first time, so it is quite hard to control.

c. A bit hard to control.

d. Lost control when lost focusing thing.

9. Do you have any suggestion in improving this robotic arm?



If yes please state the suggestion.

a. Improve accuracy.

b. Filter the finger movement data for more stable and smooth robot movement.

c. Increase the sensitivity of the sensor.

10. Do you think this system is able to commercialize in the market for laboratory use?