Low Cost Robotic Manupilandum to deliver Personalised Therapeutic Exercise for the Upper Limb of Stroke Patients

By

Hamza bin Sohail	**201201988BSMME11112F**
Ali Khalid Qureshi	**201204162BSMME11112F**
Abdullah Ghazi	**201200103BSMME11112F**
Ahmed Bilal	**201200968BSMME11112F**

Supervised By

Dr Nabeel Anwar

School of Mechanical and Manufacturing Engineering,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan

June, 2016

Low Cost Robotic Manupilandum to deliver Personalised Therapeutic Exercise for the Upper Limb of Stroke Patients

By

Hamza bin Sohail	**201201988BSMME11112F**
Ali Khalid Qureshi	**201204162BSMME11112F**
Abdullah Ghazi	**201200103BSMME11112F**
Ahmed Bilal	**201200968BSMME11112F**

Supervised By

Dr Nabeel Anwar

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelors of Engineering in Mechanical Engineering

School of Mechanical and Manufacturing Engineering,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan

June, 2016

National University of Sciences & Technology

FINAL YEAR PROJECT REPORT

We hereby recommend that the dissertation prepared under our supervision by: Hamza bin Sohail (01988), Ali Khalid Qureshi (04162), Abdullah Ghazi (00103) & Ahmed Bilal (00968) Titled: Low Cost Robotic Manupilandum to deliver Personalised Therapeutic Exercise for the Upper Limb of Stroke Patients be accepted in partial fulfillment of the requirements for the award of Bachelors of Engineering in Mechanical Engineering degree with (____ grade)

English and format checked by Ms Aamna Hassan, Signature: _____

Guidance Committee Members

1. Name:_____ Signature: _____

2. Name:_____ Signature: _____

3. Name:_____ Signature: _____

Supervisor's Name:_____ Signature: _____

Date: _____

_____ _____

Head of Department Date

COUNTERSIGNED

Date: _____ _____

Dean/Principal

Declaration

I/We certify that this research work titled "*Low Cost Robotic Manupilandum to deliver Personalised Therapeutic Exercise for the Upper Limb of Stroke Patients"* is my/our own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student
Hamza bin Sohail
201201988BSMME11112F

Signature of Student
Ali Khalid Qureshi
201204162BSMME11112F

Signature of Student
Abdullah Ghazi
201200103BSMME11112F

Signature of Student
Ahmed Bilal
201200968BSMME11112F

Copyright Statement

Dedicated to ME-04/B

Acknowledgments

First of all, we would thank ALLAH Almighty, who gave us knowledge and dedication to be able to complete this research.

We would also like to thank our Supervisor Dr. Nabeel Anwar, who, through constant guidance and direction, helped us in completion our project.

Furthermore, we would like to thank Engr. M. Naweed Hassan, for assisting us in manufacture of our project through Manufacturing Resource Centre (MRC).

Lastly, we would like to thank the members of the Humans Systems Lab (HSL) for support and patience during our brief period, for developing the project, with them.

Abstract

The aim of the project was to design and manufacture a low cost robotic manupilandum for personalised therapeutic exercise of upper limbs of stroke patients. The project scope also included the development of software for assisting the doctors in therapy.

Keywords: manupilandum, stroke, rehabilitation, therapy

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

Traditional methods of rehabilitation in practice are expensive, costly and are un-interactive. Only a handful of therapists are available to cater approximately 450,000 new stroke patients every year. This leads to higher demand and low availability, which leads to extremely high costs. The project solves this problem because now the therapist can address much more patients in the same amount of time. In addtion, our product is manufactured locally, and therefore readily available and affordable by majority of rehabilitation centers.

1.2 Aims and Objectives

The aims of the project were to:

- Design a low cost model
- Manufacture the prototype
- Ensure the end effector runs in three different paths with minimum backlash
- Develop a simple and easy to use software for therapists
- Integrate the software with the prototype

1.3 Thesis Structure

The brief description of the contents of the remaining chapters in thesis is described below.

Chapter 2 Literature Review: This chapter includes the literature review of the project. Various papers were consulted in order to develop a sense of the project and about the existing models already in the market.

Chapter 3 Project Management: This chapter shows the timeline of the project using smart art and creating a work based structure.

Chapter 4 Design: This chapter outlines the design of the entire project including the concept designs and the material selection.

Chapter 5 Analysis: This chapter will focus on various analysis conducted on the project

Chapter 6 Manufacturing: This chapter will help in understanding the manufacturing process and the problems faced during it.

Chapter 7 Motor Controller: This chapter will give us insight on the development of motor controller

Chapter 8 Software Development: This chapter will give us complete overview on the software developed and the platform used for developing it.

Chapter 9 Results and Future Work: This chapter will give a brief insight on what possibly could be the next step in the project.

Chapter 2

Literature Review

The literature review can be divided into three sub-chapters:

- Market Requirement
- Design Requirements
- Existing solutions

2.1 Market Requirements:

- Stroke is an important cause of disability under large proportion of individuals who survive stroke are chronically disabled, a king stroke a leading cause of serious, long term disability [1]
- Rehabilitation begins with acute hospitalization together with a systematic rehabilitation therapy, unfortunately it a poorly understood concept in Pakistan and there is a need to improve the stroke rehabilitation services in the country
- According to Pakistan Stroke Society, there are average of 350,000 new stroke patients every year in Pakistan [2]
- Due to lack of access to technology, the conventional therapeutic methods are used. Conventional methods not only overload the rehabilitation centers but also limit the personal care concept which bring the need of introducing more advanced therapeutic methods

2.2 Design Requirements:

- Mechanical Robustness
- Minimum Vibration of the end effector
- Low inertia
- Low friction
- 2 DOF for planar motion
- No backlash
- Sizeable range of motion

- Max force in any direction at the end effector= 200N
- Planar elliptical workspace with major axis of 800mm, transversal with respect to the subject and the minor axis of 400mm
- Nominal special resolution at the center of work space= 0.5mm [3]
- Continuous exert able force=50N
- Force torque ratio= 2N/nm [4]
- Dry Friction less than 0.1N in the center of workspace
- Stiffness = 25N/mm when vision obscured (haptic robotics; Bringing technology to multimedia)

2.3 Existing Solutions:

There are some robots which try to mimic the traditional therapeutic process. From mechanical design point of view, they can be divided into 2 major categories

1. Exoskeletons

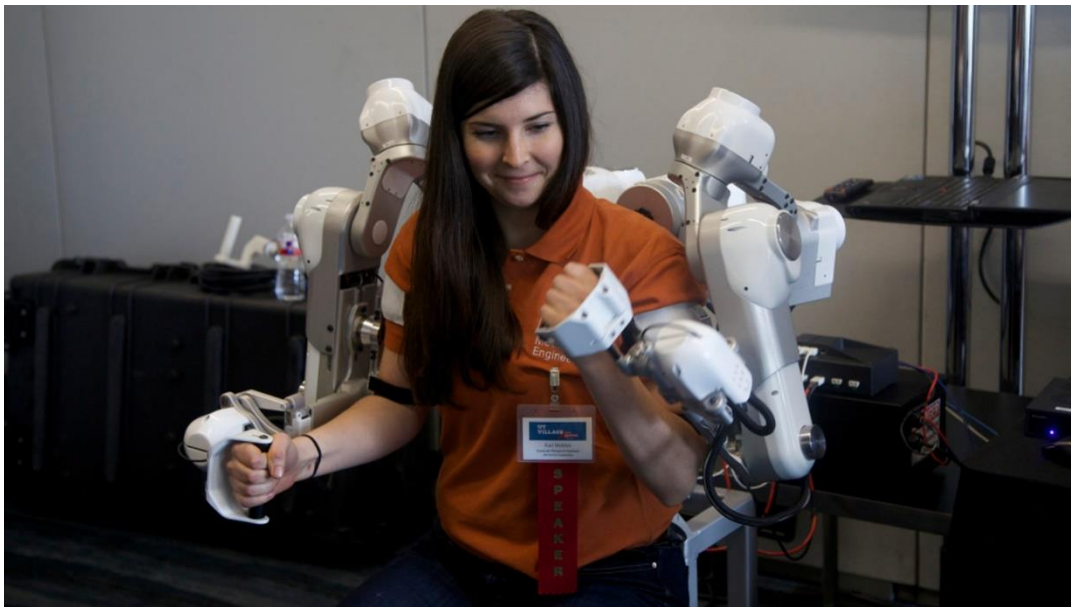


Figure 2.01: Exoskeleton

These are standalone robotic arms that are wearable and are used in assisting the therapy of stroke patients.

2. Manipulandum

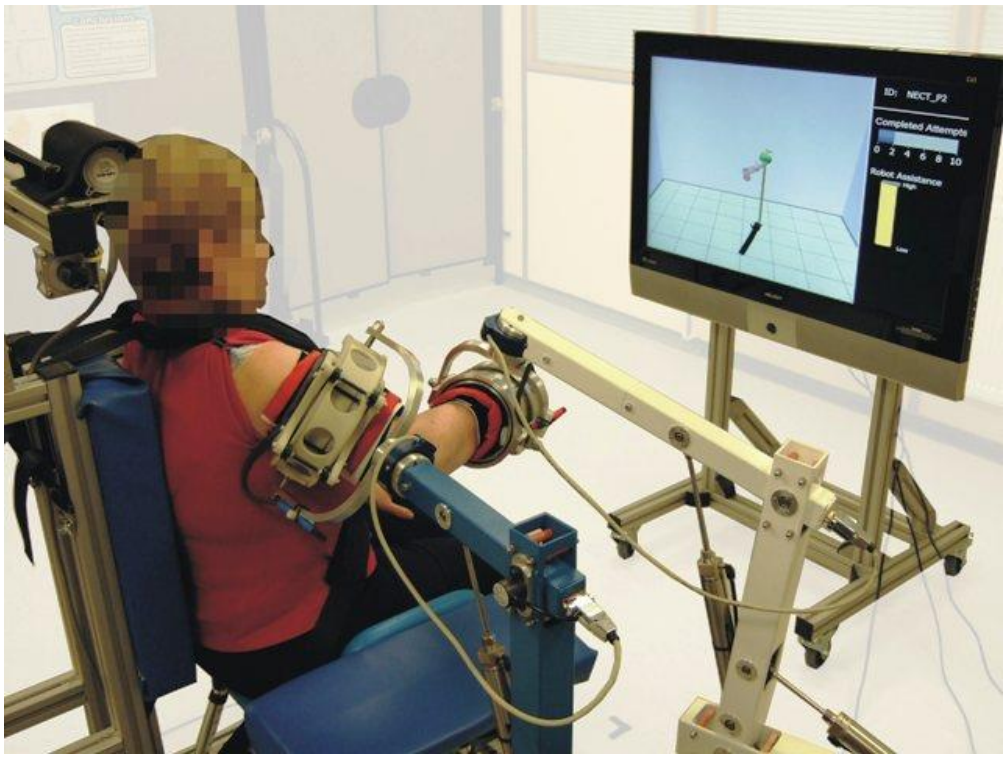


Figure 2.02: Manupilandum Type Stroke Therapy

In this type of robotic arm, there are a series of links that are attached to a centrally controlled unit.

Chapter 3

Project Management

3.1 Project Definition:

- **Objective:**
  - To design and manufacture robotic arm for stroke therapy by May 2016, while not exceeding the maximum cost of PKR 100,000.
- **Deliverables:**
  - A Manipulandum that would provide assistive/ resistive forces in real time to assist subject's arm movement in a predefined path in 2 axis
  - Software with user friendly interface to select the exercises
  - A complete business plan to market the manipulandum apparatus
- **Milestones:**
  - Concept design –1st November
  - Final CAD model-18th January
  - Manufacturing and Assembly of Manipulandum-25th January
  - Control System and Software-15th march
  - Integrate and test- 1st April
  - Final Inspection-1st May
- **Technical Requirements:**
  - 2 DOF for planar motion
  - No backlash
  - Max force in any direction at the end effector= 200N
  - Planar elliptical workspace with major axis of 800mm, transversal with respect to the subject and the minor axis of 400mm
  - Nominal special resolution at the center of work space= 0.5mm
  - Continuous exert able force=50N
  - Force torque ratio= 2N/nm
  - Dry Friction less than 0.1N in the center of workspace

3.2 Establishing Priorities

A Priority matrix was created in order to identify the priorities of the project. A Priority Matrix assists in prioritizing between 3 major factors: Time, Cost and Performance.

	Time	Cost	Performance
Constraint	90		
Enhance			80
Accept		70	

Table 3.01 Priority Matrix

3.3 Creating Work Based Structure (WBS):

A WBS is a diagram showing all the work that is needed to be done in the order that it has to be done. (PTO)

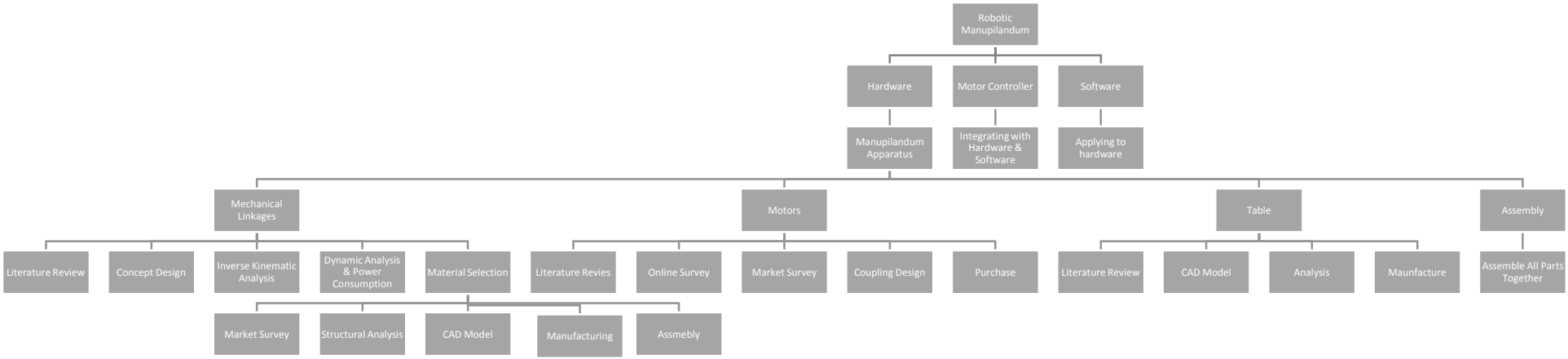


Figure 3.01 Work Based Structure (WBS)

3.4 Coding WBS on Information System

By coding WBS on an information system, a Gantt Chart can be obtained that can give a complete timeline of the project. (PTO)

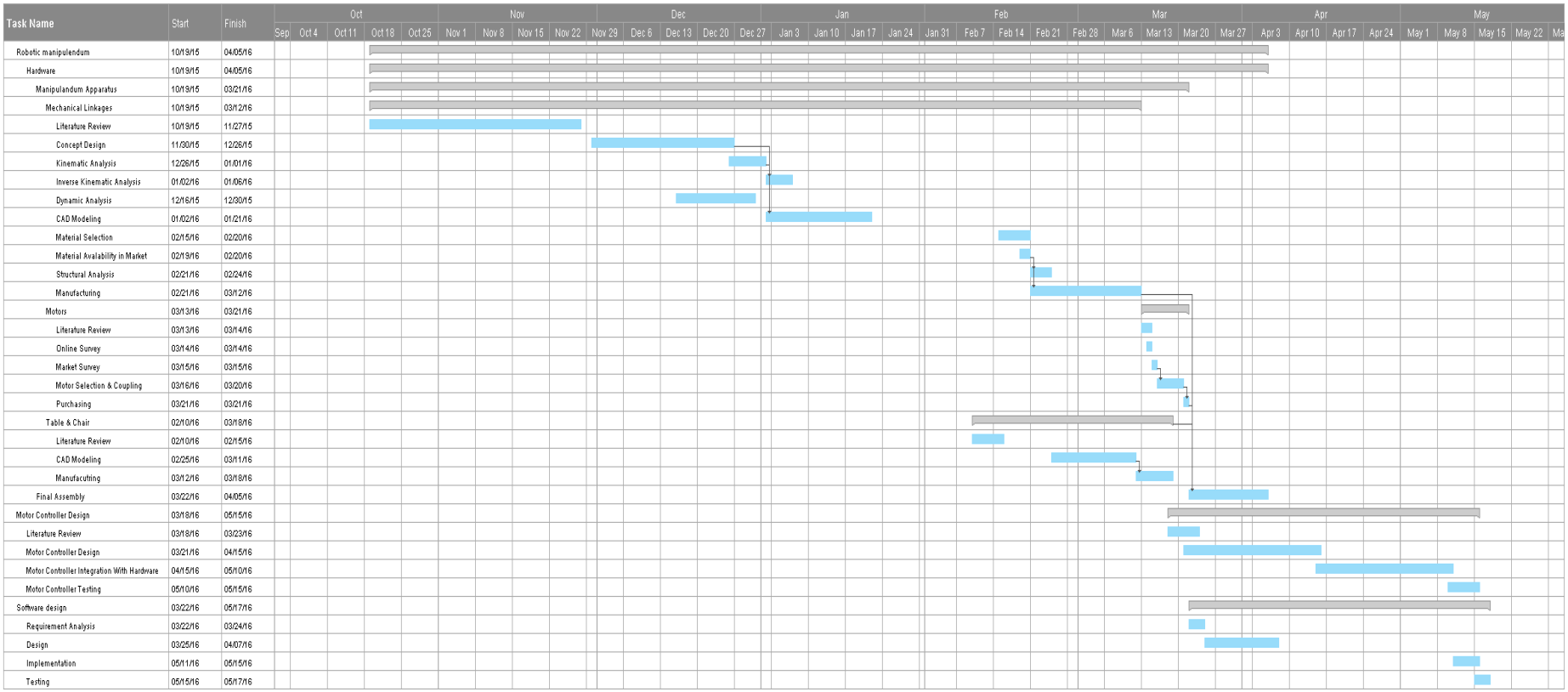


Figure 3.02 Gantt Chart

Chapter 4

Design

4.1 Concept Design:

Three Concept Designs were discussed and based on the advantages and disadvantages of each.

1. Four bar linkage:

   Design and manufacture four bar links with motors fixed at the stationary end.

   a) Advantages:
      a. Heavy motors could be used reducing the cost.
      b. No singularity
      c. Simple and easy manufacturing
      d. Low structural space
   b) Disadvantages:
      a. Low end effector torque
      b. Difficult to develop motor controller for
2. 3-D Printer Style:
   a) Advantages:
      a. High end effector torque
      b. Easy motor controller
   b) Disadvantages:
      a. Very high structural space
      b. Complex to design
      c. Complex to manufacture
      d. High cost of rotational screws
3. Two bar linkage:
   a) Advantages:
      a. Very low structural space
      b. Simple to design
   b) Disadvantages:
      a. Secondary motor has to be light, increasing cost

4.2 Final Design:

The factors affecting the final design were:

- Lowest workspace possible
- No availability of low weight, high torque motors in Pakistan
- Simple design that is easy to reproduce

Based on these factors, the final design chosen was four bar linkage.

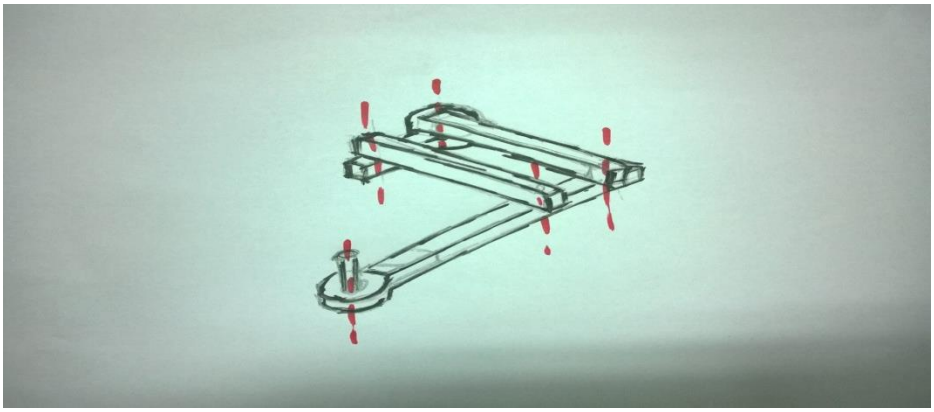


Figure 4.01 Four Bar Linkage Concept Design

4.3 Optimization of links:

The process started with iteratively selecting different arm lengths to create a manipulandum which fulfills the requirement of work envelope. With some basic lengths in mind a mathematical model was built and Kinematic analysis was performed on Matlab. After successive testing, arm link lengths were optimized follows:

- L1: 85 mm
- L2: 1280 mm
- L3: 1280 mm
- L4: 2365 mm

4.4 Selection of Material:

The 3 most commonly available materials for our application were:

- **Mild Steel:**
    - Moderate Strength
    - Moderate Weight
    - Cheap

    - Easy to Machine
    - Gauges: 14,16,18,20
    - Sizes: 0.5” x 0.5” to 5” x 5”
- **Stainless Steel:**
    - High Strength
    - Corrosion Resistant
    - Moderate Weight
    - Expensive
    - Difficult to Machine
    - Gauges: 20,18
    - Sizes: 0.5” x 0.5”
- **Aluminum:**
    - Low Weight
    - Low Strength
    - Expensive
    - Easy to Machine
    - Sizes available on demand

Stainless steel was rejected due to very high cost of material and machining. Structural analysis was performed. For similar cross-sections of aluminum and mild steel, Aluminum was unable to meet one of the desired parameters i.e. a nominal special resolution of 0.5 mm. Thus mild steel was finalized to be used for manufacturing of link arms.

Stress and strain analysis was performed in solid works for each cross-section and gauge. After which gauge 14 and 1.5*1.5 cross-section was found to be fit for our application.

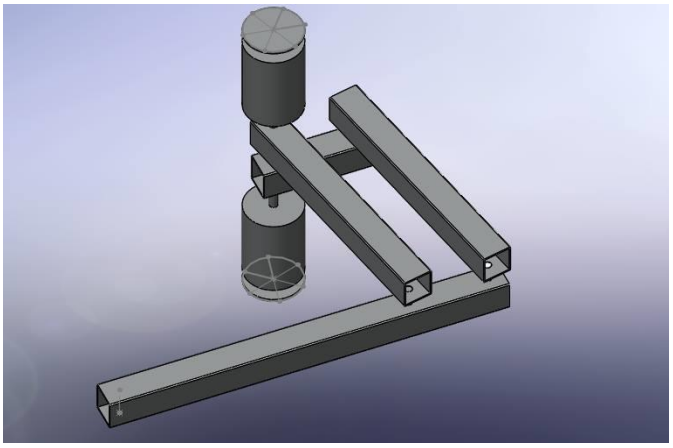


Figure 4.02 CAD Model

Chapter 5

Analysis

5.1 Kinematic Analysis:

Kinematic Analysis was carried out to find out the envelope required.

$^{0}T_{1}$= [cos(t1), -sin(t1), 0, l1*cos(t1)]

[sin(t1), cos(t1), 0, l1*sin(t1)]

[0, 0, 1, 0]

[0, 0, 0, 1]

$^{1}T_{2}$= [cos(t2), -sin(t2), 0, l2*cos(t2)]

[sin(t2), cos(t2), 0, l2*sin(t2)]

[0, 0, 1, 0]

[0, 0, 0, 1]

Transformation from zeroth to last link is

$^{0}T_{2}$= [cos(t1 + t2), -sin(t1 + t2), 0, l2

[sin(t1 + t2), cos(t1 + t2), 0, l2

[0, 0, 1,

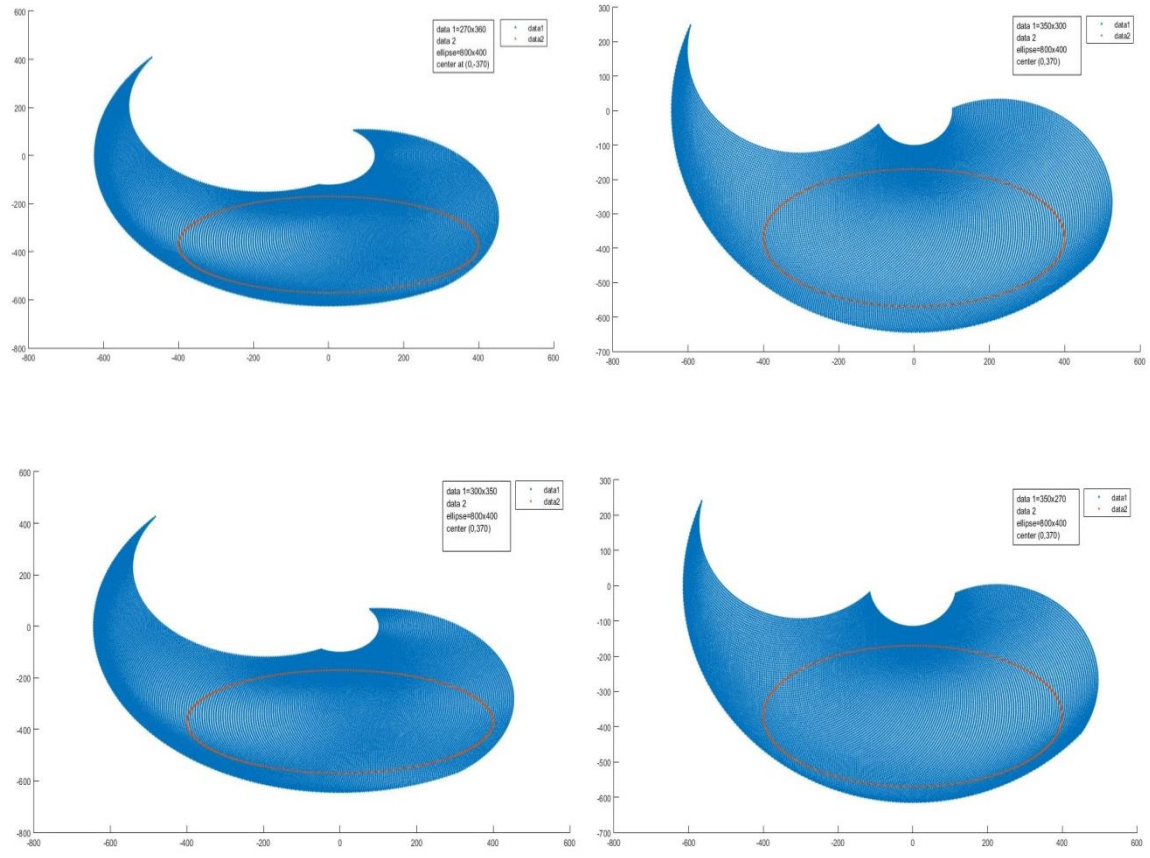
[0, 0, 0,

Matlab Code:

```
l1=350;
l2=270;
x=[];
y=[];
for t1 = 150:1:310
    for t2 = 16:1:164
        x = [x;l1*cos(t1/180 *pi) + l2*cos(t1/180 *pi)*cos(t2/180 *pi) - l2*sin(t1/180 *pi)*sin(t2/180 *pi)];
        y = [y;l1*sin(t1/180 *pi) + l2*cos(t1/180 *pi)*sin(t2/180 *pi) + l2*cos(t2/180 *pi)*sin(t1/180 *pi)];
    end
end
hold on
plot(x,y,'.');

r1=400;
r2=200;
x=[];
y=[];
for t=0:1:360;
 x=[x;r1*cos(t/180 *pi)];
 y=[y;r2*sin(t/180 *pi)-370];
end
hold on;
plot(x,y,'.');
disp(x);
disp(y);
```

Envelope Simulation:



Figures 5.01, 5.02, 5.03 & 5.04 Envelopes Simulations

Final Envelope Analysis:

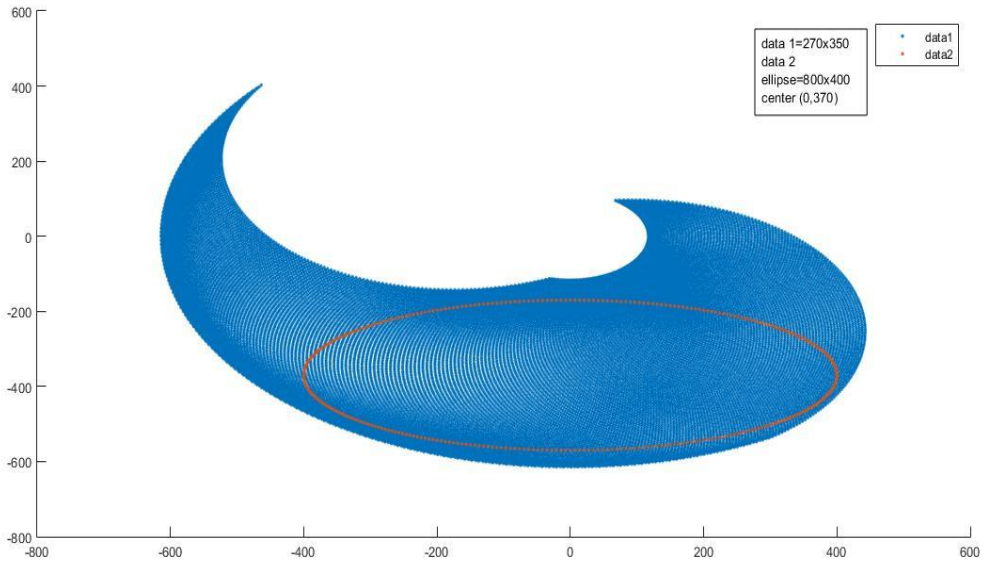


Figure 5.05 Chosen Envelope 800x400 mm

5.2 Inverse Kinematic Analysis:

The inverse analysis was done to find out the position of the links at all angles. The angles would then be used in the motor controller to define paths.

x	y	c2	s2	k1	k2	theta1	theta1 degrees	theta2	theta2 degrees	theta1 degrees	theta2 degrees			theeta 3 degrees
400	-370	0.485597	0.874183	584.7059	314.7059	-0.74646	-42.76882539	1.063751	60.94842552	-42.76882539	60.94842552			18.1796
399.9391	-366.51	0.472122	0.881533	587.352	317.352	-0.74181	-42.50255687	1.0791	61.82787798	-42.50255687	61.82787798			19.32532
399.7563	-363.02	0.458275	0.88881	589.9717	319.9717	-0.73727	-42.24269905	1.094743	62.72413695	-42.24269905	62.72413695			20.48144
399.4518	-359.533	0.444062	0.895996	592.5587	322.5587	-0.73285	-41.9892871	1.11067	63.63668425	-41.9892871	63.63668425	l1	270	21.6474
399.0256	-356.049	0.429486	0.903073	595.1064	325.1064	-0.72854	-41.74238537	1.126873	64.56504608	-41.74238537	64.56504608	l2	360	22.82266
398.4779	-352.569	0.414555	0.910024	597.6088	327.6088	-0.72435	-41.50205135	1.143343	65.50871257	-41.50205135	65.50871257			24.00666
397.8088	-349.094	0.399273	0.916832	600.0596	330.0596	-0.72027	-41.26834113	1.160073	66.46726179	-41.26834113	66.46726179			25.19892
397.0185	-345.626	0.383648	0.92348	602.4526	332.4526	-0.71631	-41.0413357	1.177053	67.44019374	-41.0413357	67.44019374			26.39886
396.1072	-342.165	0.367686	0.92995	604.7821	334.7821	-0.71246	-40.8211164	1.194277	68.42704897	-40.8211164	68.42704897			27.60593
395.0753	-338.713	0.351394	0.936228	607.0419	337.0419	-0.70874	-40.60774217	1.211736	69.427378	-40.60774217	69.427378			28.81964
393.9231	-335.27	0.334782	0.942296	609.2264	339.2264	-0.70514	-40.4013128	1.229422	70.44070889	-40.4013128	70.44070889			30.0394
392.6509	-331.838	0.317857	0.948139	611.33	341.33	-0.70166	-40.20191219	1.247328	71.46665031	-40.20191219	71.46665031			31.26474
391.259	-328.418	0.300626	0.953742	613.3471	343.3471	-0.6983	-40.00965914	1.265447	72.50476422	-40.00965914	72.50476422			32.49511
389.748	-325.01	0.283101	0.95909	615.2724	345.2724	-0.69507	-39.82463529	1.28377	73.55461835	-39.82463529	73.55461835			33.72998
388.1183	-321.616	0.26529	0.964169	617.1007	347.1007	-0.69197	-39.6469641	1.302291	74.61580457	-39.6469641	74.61580457			34.96884
386.3703	-318.236	0.247203	0.968964	618.8269	348.8269	-0.689	-39.47678006	1.321004	75.6879309	-39.47678006	75.6879309			36.21115
384.5047	-314.873	0.228851	0.973462	620.4462	350.4462	-0.68616	-39.31418954	1.3399	76.77058911	-39.31418954	76.77058911			37.4564
382.5219	-311.526	0.210243	0.977649	621.9537	351.9537	-0.68346	-39.15935908	1.358973	77.86339896	-39.15935908	77.86339896			38.70404
380.4226	-308.197	0.191391	0.981514	623.345	353.345	-0.6809	-39.01240987	1.378217	78.96600094	-39.01240987	78.96600094			39.95359
378.2074	-304.886	0.172307	0.985043	624.6156	354.6156	-0.67847	-38.87351917	1.397625	80.07799792	-38.87351917	80.07799792			41.20448
375.877	-301.596	0.153002	0.988226	625.7613	355.7613	-0.67619	-38.74284316	1.417191	81.19903973	-38.74284316	81.19903973			42.4562
373.4322	-298.326	0.133489	0.99105	626.7781	356.7781	-0.67406	-38.62055353	1.436908	82.32874861	-38.62055353	82.32874861			43.7082
370.8735	-295.079	0.113779	0.993506	627.6622	357.6622	-0.67207	-38.50686857	1.456771	83.46680881	-38.50686857	83.46680881			44.95994
368.2019	-291.854	0.093885	0.995583	628.4099	358.4099	-0.67024	-38.40196587	1.476773	84.61284123	-38.40196587	84.61284123			46.21088
365.4182	-288.653	0.073821	0.997271	629.0177	359.0177	-0.66857	-38.30606336	1.496908	85.76650548	-38.30606336	85.76650548			47.46044
362.5231	-285.476	0.053599	0.998563	629.4825	359.4825	-0.66705	-38.21939398	1.517171	86.92751044	-38.21939398	86.92751044			48.70812
359.5176	-282.326	0.033234	0.999448	629.8011	359.8011	-0.66571	-38.14221491	1.537556	88.09546	-38.14221491	88.09546			49.95325
356.4026	-279.202	0.012739	0.999919	629.9708	359.9708	-0.66453	-38.07476077	1.558057	89.2700738	-38.07476077	89.2700738			51.19531

Table 5.01 Inverse Kinematic Analysis

5.3 Structural Analysis:

Manual:

Manual structural analysis was done to find out the maximum yield strength of the linkage when a force of 200 N was applied at the end effector.

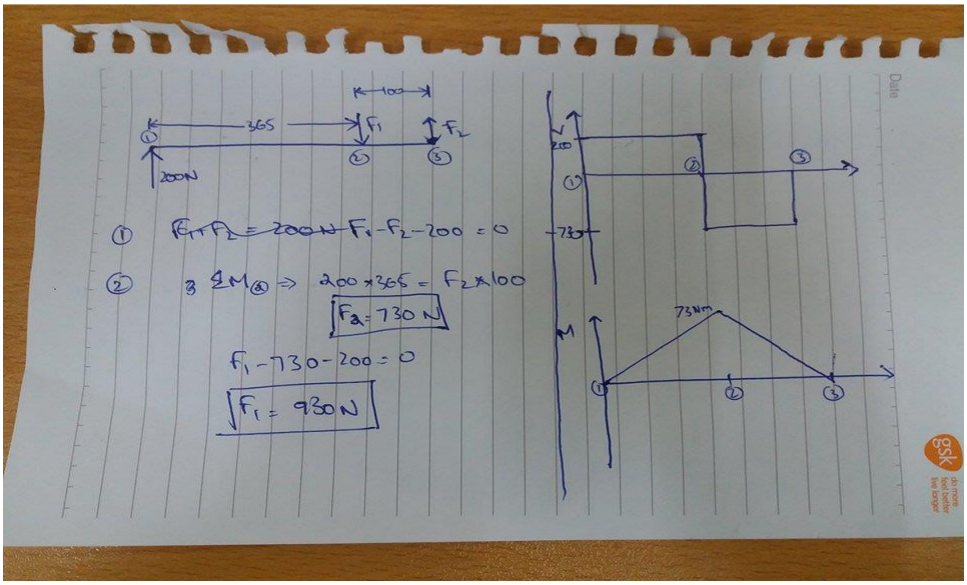


Figure 5.06 Stress Analysis

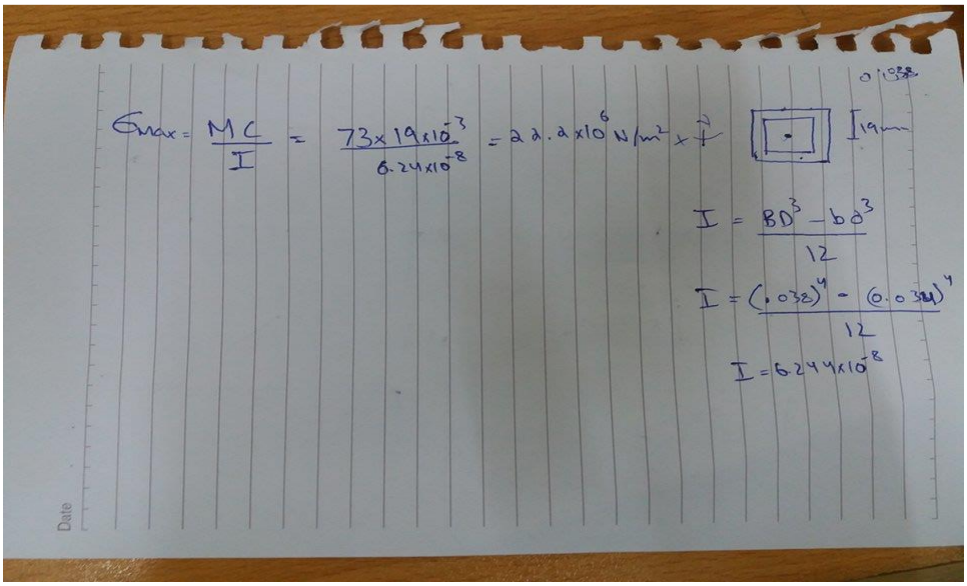


Figure 5.07 Stress Analysis

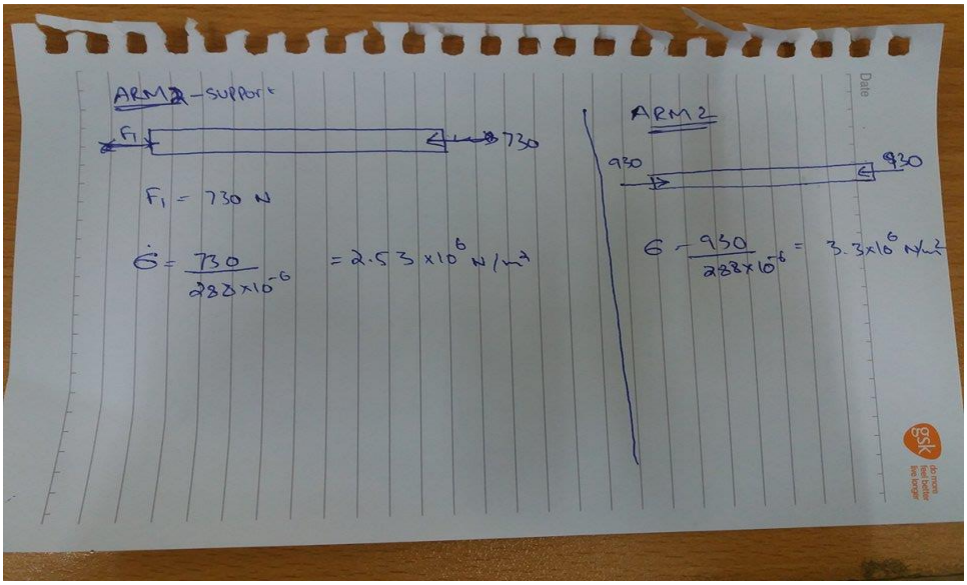


Figure 5.08 Stress Analysis

Solidworks:

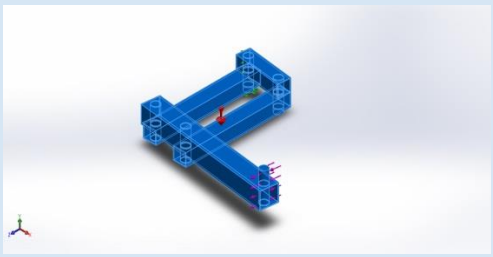
Model Reference	Properties	
	Name:	Cast Carbon Steel
	Model type:	Linear Elastic Isotropic
	Default failure criterion:	Max von Mises Stress
	Yield strength:	2.48168e+008 N/m^2
	Tensile strength:	4.82549e+008 N/m^2
	Elastic modulus:	2e+011 N/m^2
	Poisson's ratio:	0.32
	Mass density:	7800 kg/m^3
	Shear modulus:	7.6e+010 N/m^2
	Thermal expansion coefficient:	1.2e-005 /Kelvin

Table 5.02 Material Properties

Name	Type	Min	Max
Stress	VON: von Mises Stress	63.6964 N/m^2 Node: 18900	6.96628e+006 N/m^2 Node: 1788

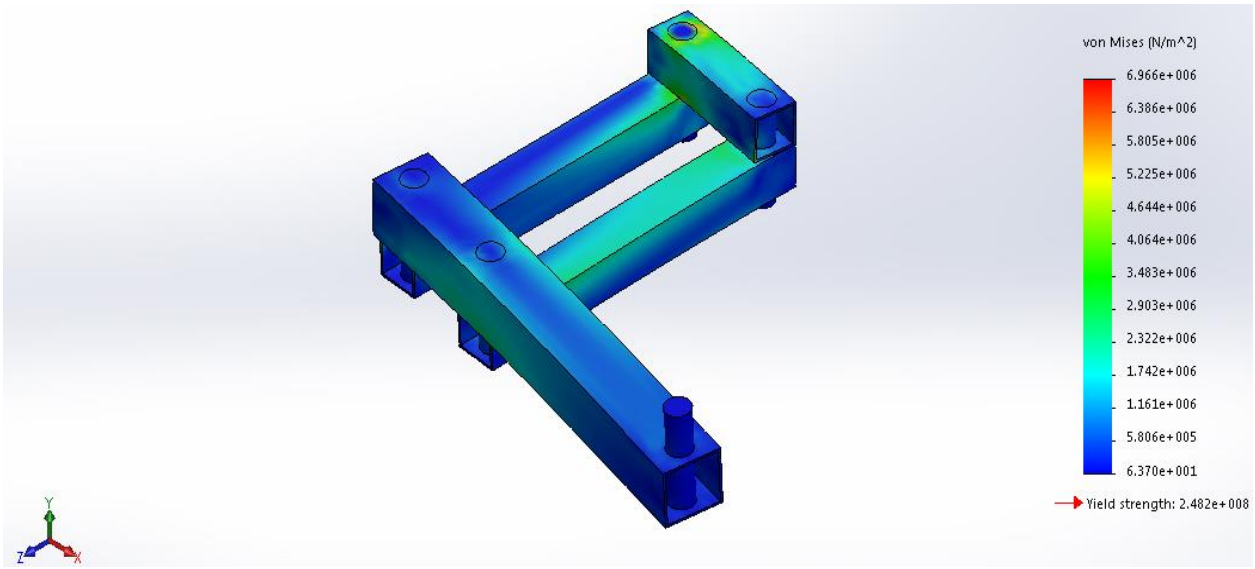


Figure 5.09 Von Mises Stress

Name	Type	Min	Max
Displacement	URES: Resultant Displacement	0 mm Node: 37	0.0667442 mm Node: 21277

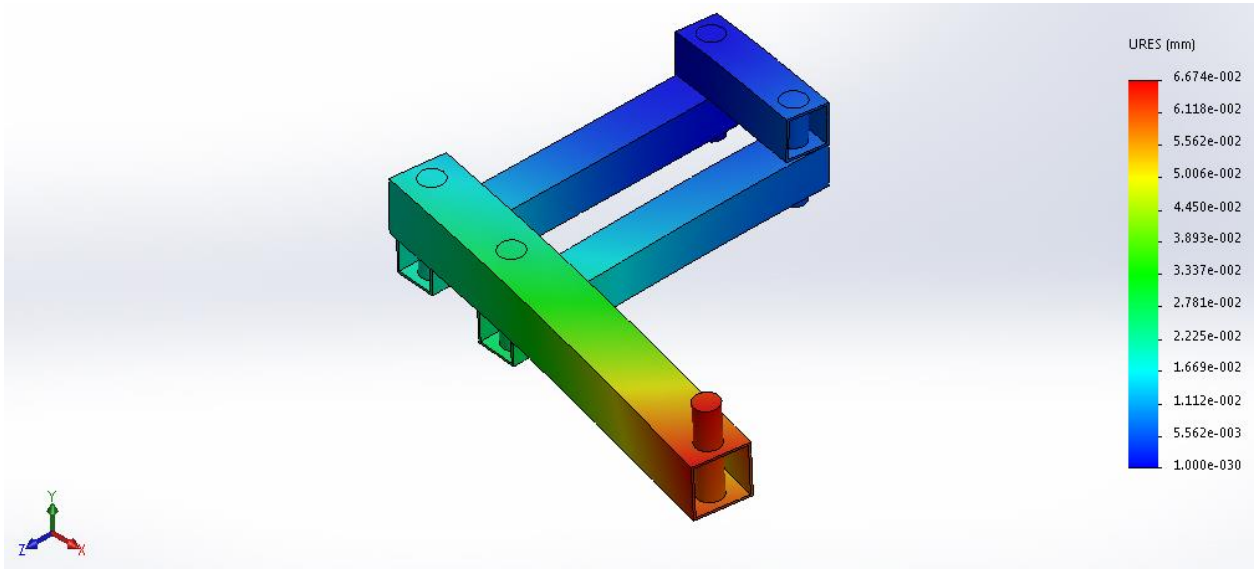


Figure 5.10 Displacement Analysis

Name	Type	Min	Max
Strain	ESTRN: Equivalent Strain	2.80526e-010 Element: 9291	3.02193e-005 Element: 975

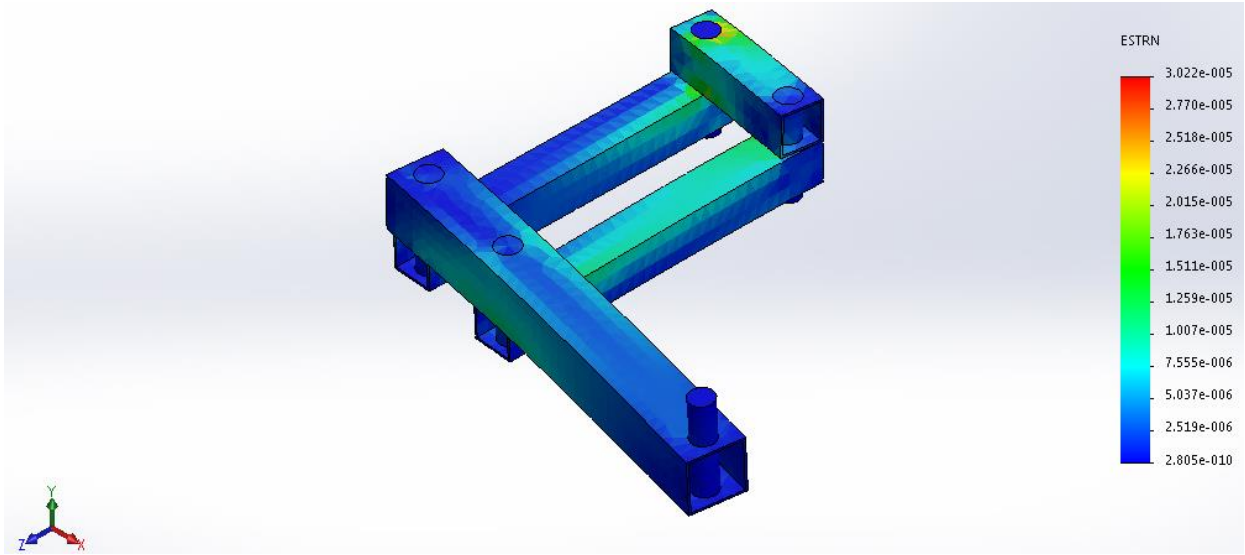


Figure 5.11 Strain Analysis

Chapter 6
Manufacturing

6.1 Parts Manufactured:

- 2 x Support Arm
- 1 x Main Arm
- 1 x Secondary Arm
- 6 x Bearing Housings
- 3 x Pins with Collars
- 2 x Motor Couplings
- 2 x Cater Balls
- 1 x End Effector

The parts were manufactured and assembled in-house at Manufacturing Resource Center (MRC)-SMME.

6.2 Machinery Involved:

- Lathe
- Vertical Drill
- Grinder
- Vertical Drill

6.3 Problems Faced:

During the manufacturing several problems arose that were tackled accordingly.

- Initially a wooden end effector was manufactured. However, since it was not made of hard wood, fitting the bearing was not possible. The wooden design was replaced with an aluminum one to overcome this.
- Flanges were created to press fit the bearing into. The flanges could not be welded to the links as it would damage bearings. Rivets were used instead to fasten the flanges to the links.
- The links sagged and created points of high moment at bearings due to high overall weight. This was overcome by drilling holes in links underneath those points and attach caster balls to them.

Chapter 7
Motor Controller

7.1 Components:

- 2 x L298N
- 1 x Arduino Mega
- 1 x 12V Supply
- 1 x 12V DC Fan
- 1 x Capacitor
- 2 x 12V Stepper Motors

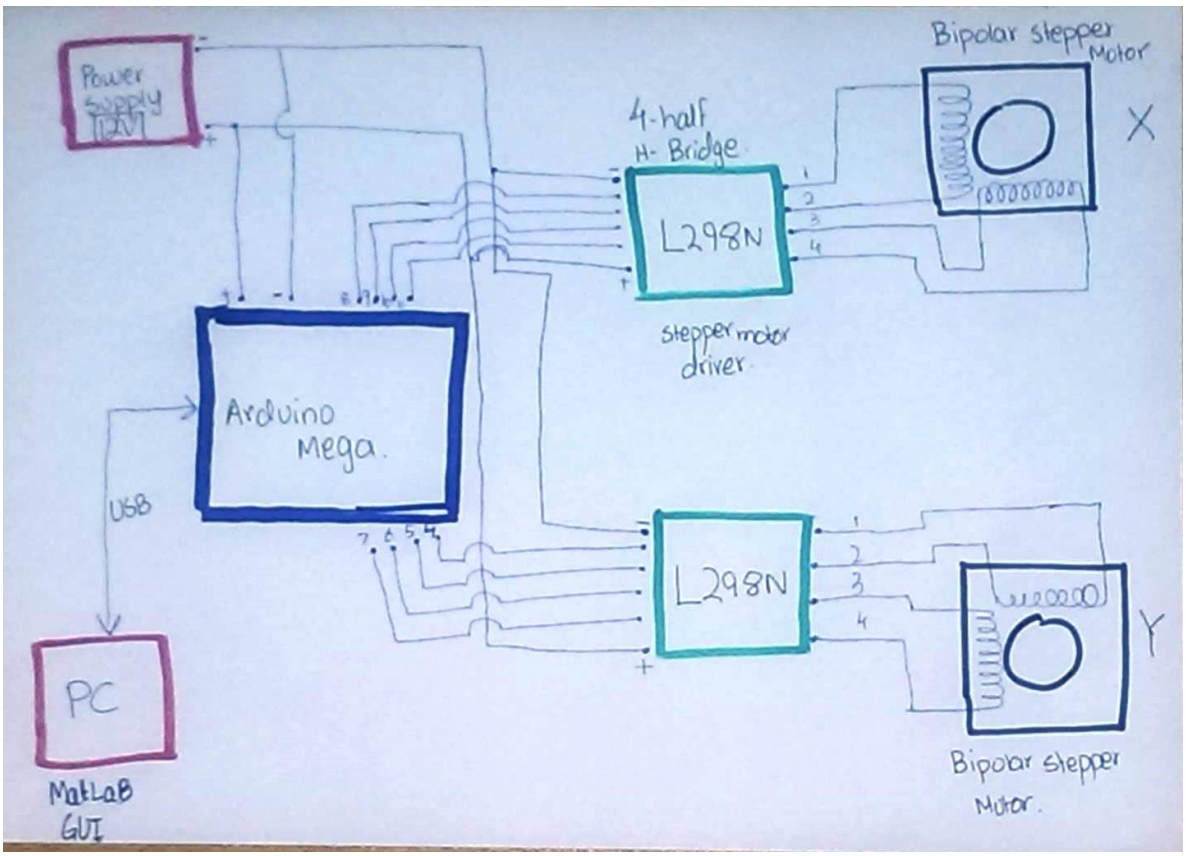


Figure 7.01 Controller Diagram

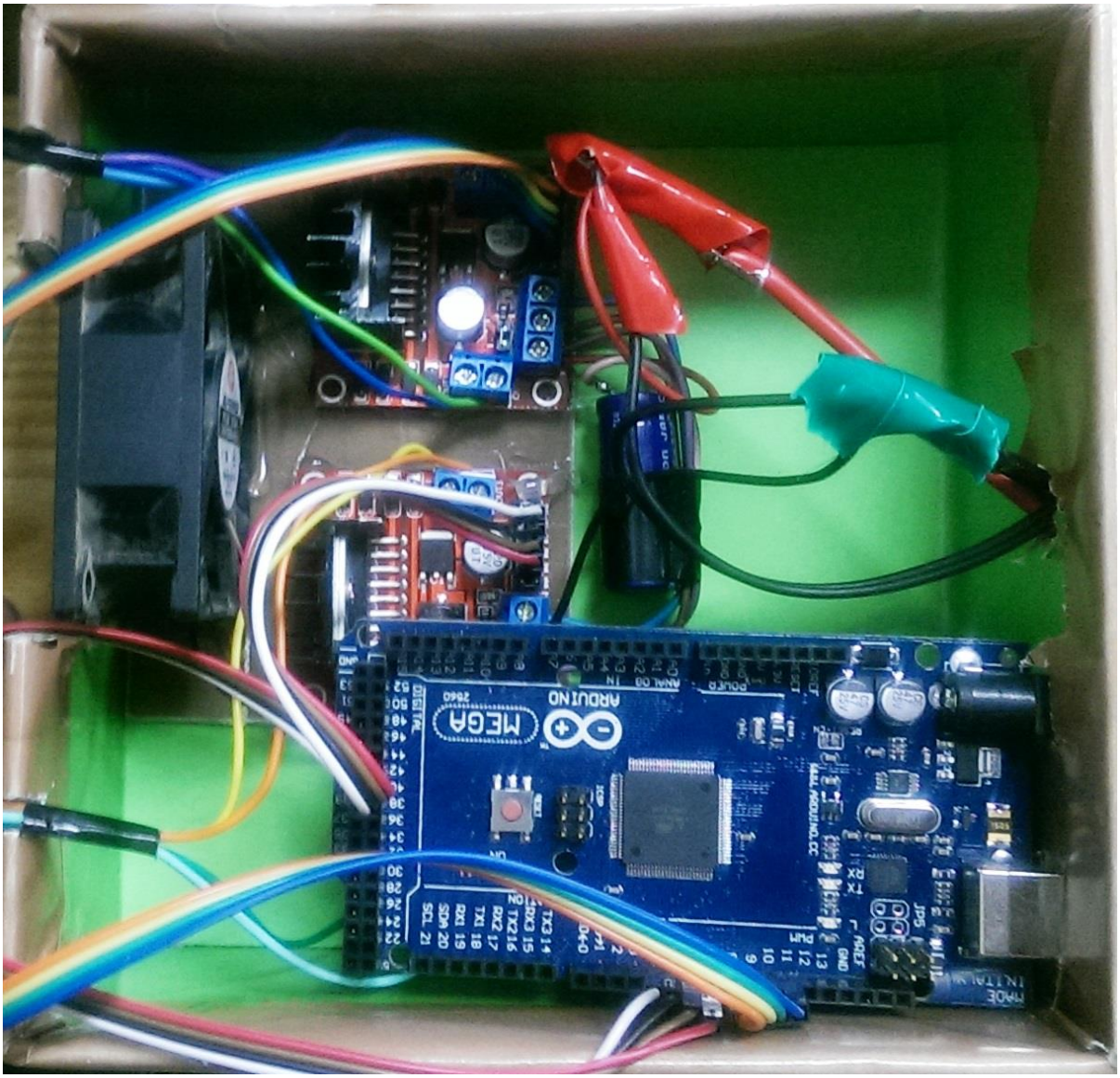


Figure 7.02 Controller

Chapter 8
Software Development

8.1 Graphical User Interface:

Interactive software has been developed in Matlab GUI for therapist to select exercises and other parameters

Window 1:

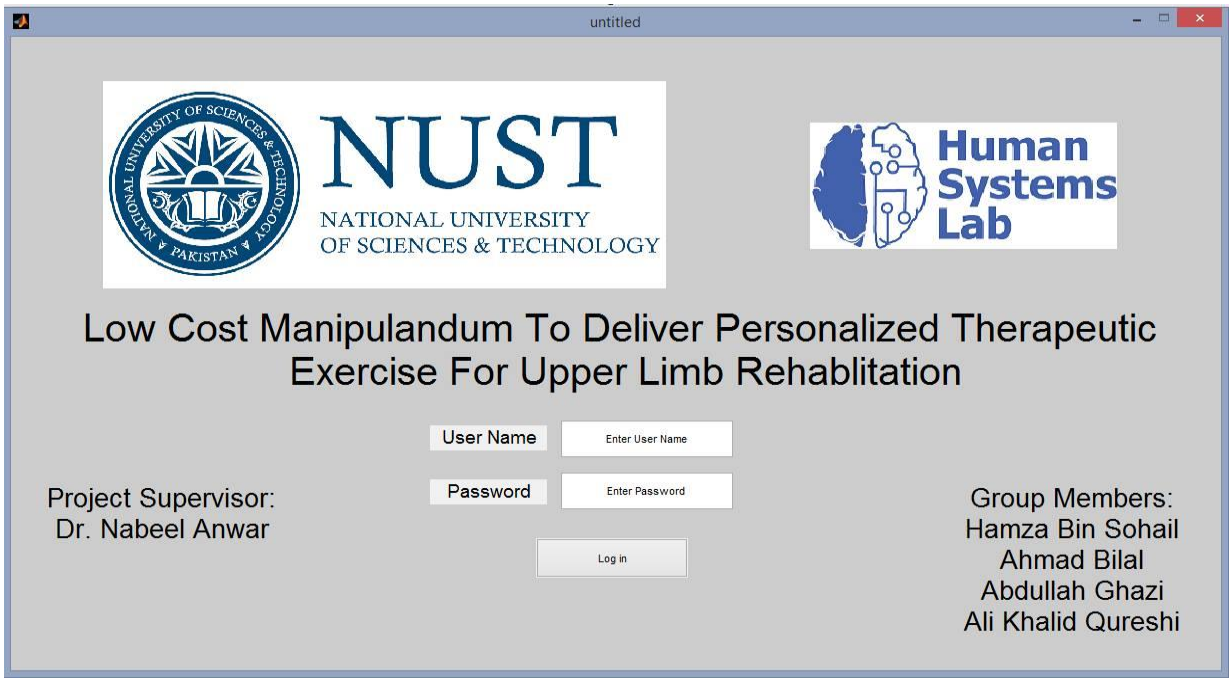


Figure 8.01 Opening Window

Code:

```
 function varargout = untitled(varargin)
% UNTITLED MATLAB code for untitled.fig
% UNTITLED, by itself, creates a new UNTITLED or raises the existing
% singleton*.
%
% H = UNTITLED returns the handle to a new UNTITLED or the handle to
% the existing singleton*.
%
% UNTITLED('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in UNTITLED.M with the given input arguments.
%
% UNTITLED('Property','Value',...) creates a new UNTITLED or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before untitled_OpeningFcn gets called. An
```

```
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to untitled_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help untitled
% Last Modified by GUIDE v2.5 12-May-2016 11:12:45
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @untitled_OpeningFcn, ...
'gui_OutputFcn', @untitled_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to untitled (see VARARGIN)
% Choose default command line output for untitled
handles.output = hObject;
axes(handles.axes1);
imshow('nustlogo.png');
axes(handles.axes3);
imshow('hsl_logo_fulltitle.png');
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes untitled wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = untitled_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
```

```
varargout{1} = handles.output;
function edit1_Callback(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
% str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
% str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
ID = get(handles.edit1,'string');
PW = get(handles.edit3,'string');
if strcmp(ID,'Dr Nabeel Anwar') && strcmp(PW,'smme')
secondwindow;
close untitled;
else
errordlg('Invalid ID or Password');
set(handles.edit1,'string','')
set(handles.edit3,'string','')
end
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject handle to figure1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

Window 2:

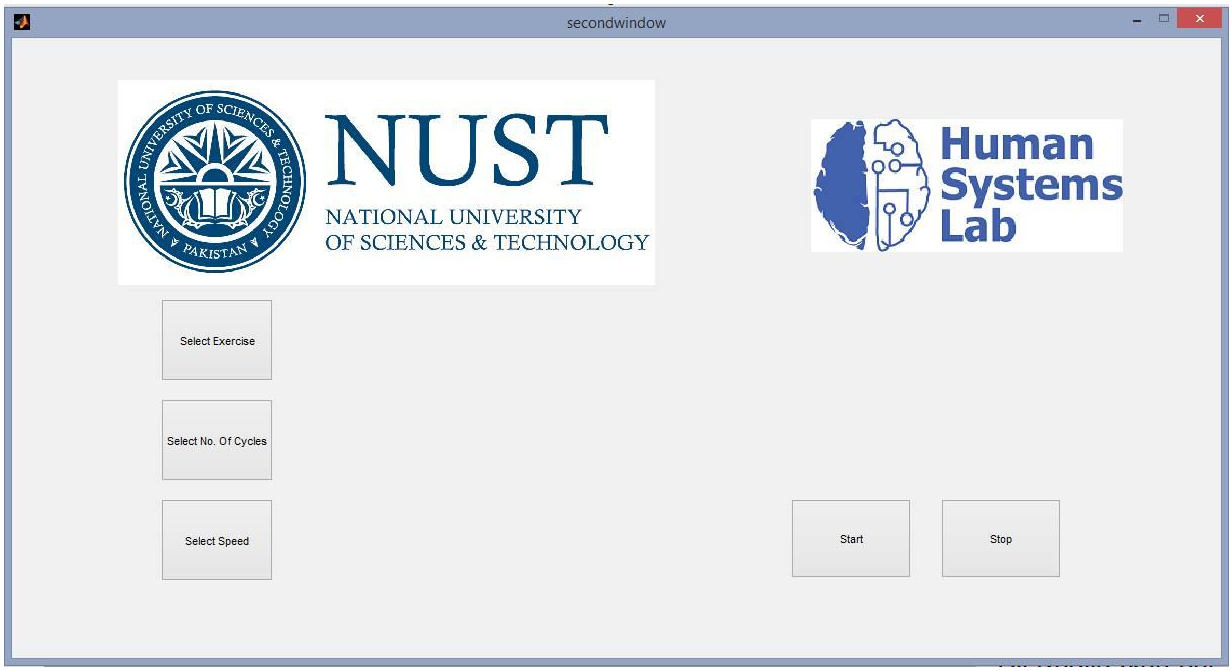


Figure 8.02 Menu Window

Code:

```
 function varargout = secondwindow(varargin)
% SECONDWINDOW MATLAB code for secondwindow.fig
% SECONDWINDOW, by itself, creates a new SECONDWINDOW or raises the existing
% singleton*.
%
% H = SECONDWINDOW returns the handle to a new SECONDWINDOW or the handle to
% the existing singleton*.
%
% SECONDWINDOW('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in SECONDWINDOW.M with the given input arguments.
%
% SECONDWINDOW('Property','Value',...) creates a new SECONDWINDOW or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before secondwindow_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to secondwindow_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help secondwindow
% Last Modified by GUIDE v2.5 06-May-2016 02:33:27
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @secondwindow_OpeningFcn, ...
'gui_OutputFcn', @secondwindow_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before secondwindow is made visible.
function secondwindow_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to secondwindow (see VARARGIN)
% Choose default command line output for secondwindow
handles.output = hObject;
axes(handles.axes1);
imshow('nustlogo.png');
axes(handles.axes2);
imshow('hsl_logo_fulltitle.png');
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes secondwindow wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = secondwindow_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
global exercise;
exercise
selectexercise;
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles
selectcycles;
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
selectspeed;
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
close secondwindow;
```

Window 3:

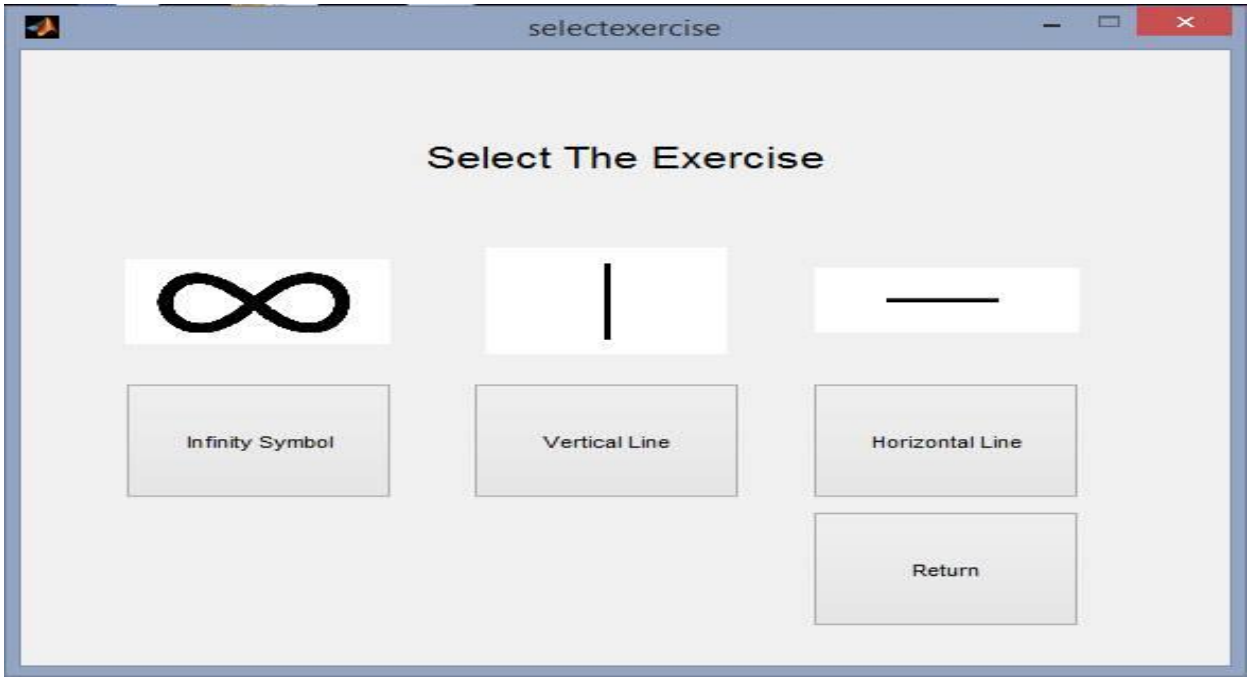


Figure 8.03 Exercise Window

Code:

```
 function [exercise] = selectexercise(varargin)
% SELECTEXERCISE MATLAB code for selectexercise.fig
% SELECTEXERCISE, by itself, creates a new SELECTEXERCISE or raises the existing
% singleton*.
%
% H = SELECTEXERCISE returns the handle to a new SELECTEXERCISE or the handle to
% the existing singleton*.
%
% SELECTEXERCISE('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in SELECTEXERCISE.M with the given input arguments.
%
% SELECTEXERCISE('Property','Value',...) creates a new SELECTEXERCISE or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before selectexercise_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to selectexercise_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help selectexercise
% Last Modified by GUIDE v2.5 06-May-2016 02:39:25
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @selectexercise_OpeningFcn, ...
'gui_OutputFcn', @selectexercise_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[exercise{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before selectexercise is made visible.
function selectexercise_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to selectexercise (see VARARGIN)
% Choose default command line output for selectexercise
handles.output = hObject;
```

```
axes(handles.axes1);
imshow('infinity_symbol.png');
axes(handles.axes3);
imshow('horizontal line.png');
axes(handles.axes2);
imshow('vertical line.png');
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes selectexercise wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = selectexercise_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global exercise;
exercise = 1;
close selectexercise;
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global exercise;
exercise = 2;
close selectexercise;
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global exercise;
exercise = 3;
close selectexercise;
```

Window 4:

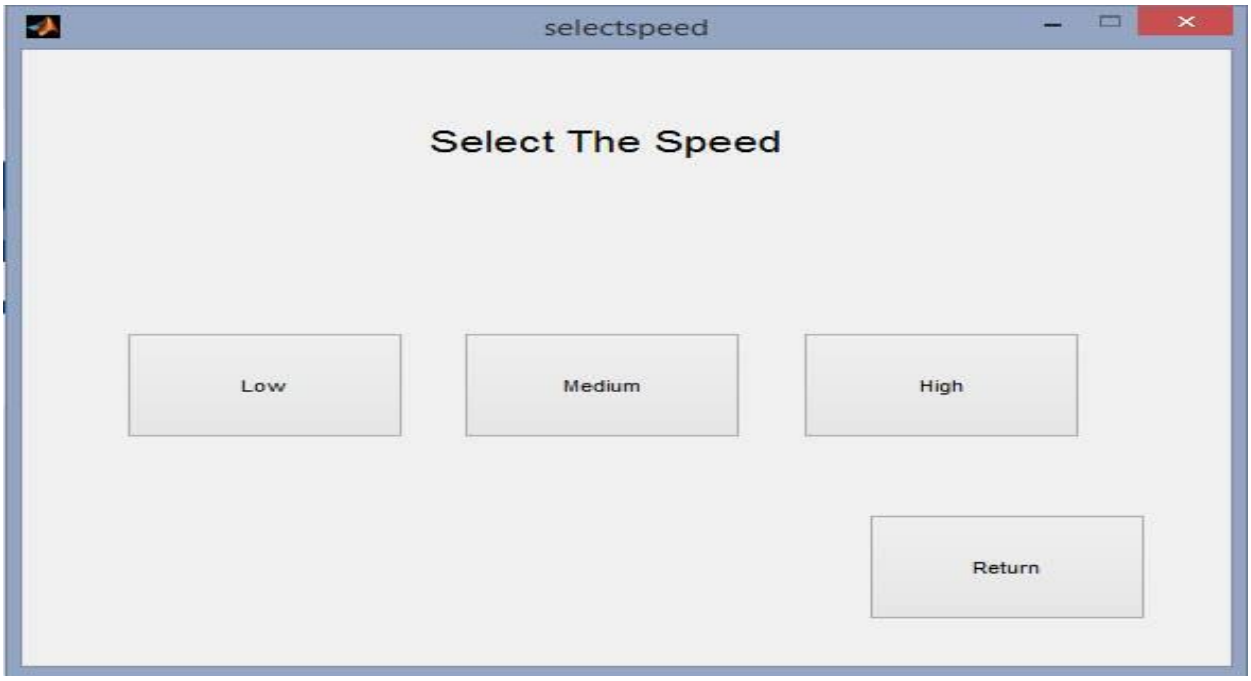


Figure 8.04 Speed Selection Window

Code:

```
 function varargout = selectspeed(varargin)
% SELECTSPEED MATLAB code for selectspeed.fig
% SELECTSPEED, by itself, creates a new SELECTSPEED or raises the existing
% singleton*.
%
% H = SELECTSPEED returns the handle to a new SELECTSPEED or the handle to
% the existing singleton*.
%
% SELECTSPEED('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in SELECTSPEED.M with the given input arguments.
%
% SELECTSPEED('Property','Value',...) creates a new SELECTSPEED or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before selectspeed_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to selectspeed_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help selectspeed
% Last Modified by GUIDE v2.5 06-May-2016 03:01:37
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @selectspeed_OpeningFcn, ...
'gui_OutputFcn', @selectspeed_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before selectspeed is made visible.
function selectspeed_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to selectspeed (see VARARGIN)
% Choose default command line output for selectspeed
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes selectspeed wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = selectspeed_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
close selectspeed;
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton4 (see GCBO)
```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

Window 5:

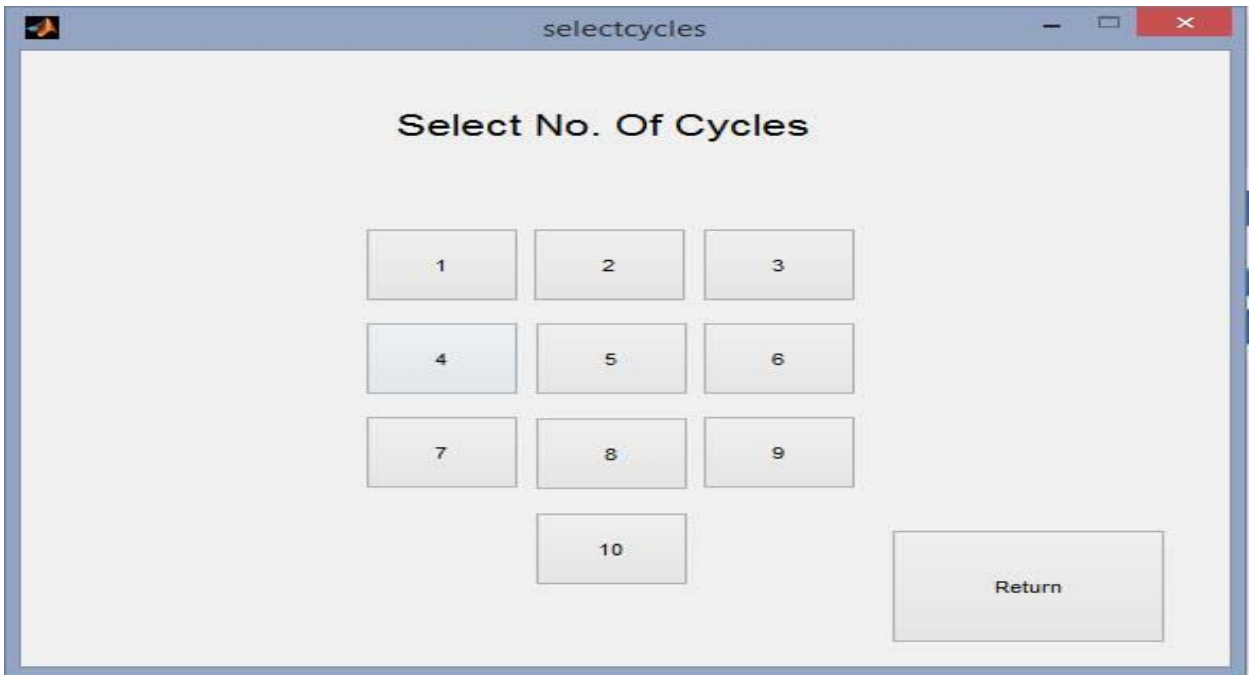


Figure 8.05 Cycle Select Window

Code:

function varargout = selectcycles(varargin)
% SELECTCYCLES MATLAB code for selectcycles.fig
% SELECTCYCLES, by itself, creates a new SELECTCYCLES or raises the existing
% singleton*.
%
% H = SELECTCYCLES returns the handle to a new SELECTCYCLES or the handle to
% the existing singleton*.
%
% SELECTCYCLES('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in SELECTCYCLES.M with the given input arguments.
%
% SELECTCYCLES('Property','Value',...) creates a new SELECTCYCLES or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before selectcycles_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to selectcycles_OpeningFcn via varargin.
%

```
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help selectcycles
% Last Modified by GUIDE v2.5 06-May-2016 02:53:17
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @selectcycles_OpeningFcn, ...
'gui_OutputFcn', @selectcycles_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before selectcycles is made visible.
function selectcycles_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to selectcycles (see VARARGIN)
% Choose default command line output for selectcycles
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes selectcycles wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = selectcycles_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
```

```
cycles = 1;
close selectcycles;
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 2;
close selectcycles;
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 4;
close selectcycles;
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 3;
close selectcycles;
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 5;
close selectcycles;
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton8 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 10;
close selectcycles;
% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 6;
close selectcycles;
```

```
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton10 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 7;
close selectcycles;
% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton11 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 9;
close selectcycles;
% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton12 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global cycles;
cycles = 8;
close selectcycles;
% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton13 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
close selectcycles;
```

8.2 Arduino Code:

Arduino was used to link the software with the hardware.

Code:

```
 int wire1=8;
int wire2=9;
int wire3=10;
int wire4=11;
int wire5=4;
int wire6=5;
int wire7=6;
int wire8=7;
int exercise;
int cycles;
```

const int inf_set2time_motor1[348] = {4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 3, 3, 3, 2, 2, 2, 1, 1,
1, 0, 0, 0, -1, -1, -1, -1, -2, -2, -2, -2, -2, -3, -3, -3, -3, -3, -3, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4,
-4, -4, -4, -4, -4, -4, -3, -3, -3, -3, -3, -3, -3, -2, -2, -2, -2, -2, -2, -2, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-2, -2, -2, -2, -2, -2, -2, -2, -3, -3, -3, -3, -4, -4, -4, -4, -4, -4, -4, -5, -5, -5, -5, -5, -5, -5, -5, -5, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -3, -3, -3, -3, -3, -3, -3,
-3, -3, -3, -3, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 0 };
const int inf_set2time_motor2[348] = {-2, -1, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -4,
-4, -4, -4, -3, -3, -3, -3, -3, -3, -3, -3, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1,
-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -2, -2, -2, -2,
-2, -2, -2, -2, -2, -2, -2, -2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0 };
const int vline_set2time_motor1[106] = {-2, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -4, -4, -4, -4, -4, -4, -4, -4, -4, -1, 5, 5, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 1, -4, -4, -4, -4, -4, -4, -4, -3, -3,
-3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3 };
const int vline_set2time_motor2[106] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 2, 5, 1, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -3, -3, -3, 0, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1};
const int hline_set2time_motor1[240] = {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2,
-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
1, 1, 1, 1, 1, 1, 1};
const int hline_set2time_motor2[240] = {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2,
-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2,
-2, -2, -2, -2, -2, -2, -2, -2, -2, -2};
void runMotor1CW(int starting_point_id, int num_steps);
void runMotor1CCW(int starting_point_id, int num_steps);
void runMotor2CW(int starting_point_id, int num_steps);
void runMotor2CCW(int starting_point_id, int num_steps);
void draw_inf(int ncycles);
void draw_vline(int ncycles);
void draw_hline(int ncycles);
void setup() {
Serial.begin(9600);
pinMode(wire1,OUTPUT); // put your setup code here, to run once:
pinMode(wire2,OUTPUT);
pinMode(wire3,OUTPUT);
pinMode(wire4,OUTPUT);
```

```
pinMode(wire5,OUTPUT);
pinMode(wire6,OUTPUT);
pinMode(wire7,OUTPUT);
pinMode(wire8,OUTPUT);
}
char stepper_motor_states_cw[][4] = {
{1, 0, 0, 0 },
{0, 1, 0, 0 },
{0, 0, 1, 0 },
{0, 0, 0, 1 }
};
char stepper_motor_states_ccw[][4] = {
{0, 0, 0, 1 },
{0, 0, 1, 0 },
{0, 1, 0, 0 },
{1, 0, 0, 0 }
};
void loop() {
// put your main code here, to run repeatedly:
while(Serial.available()==0)
{
exercise=Serial.read();
cycles=Serial.read();
}
if (exercise == 1)
{
draw_inf(cycles);
}
else if (exercise == 2)
{
draw_vline(cycles);
}
else if (exercise == 3)
{
draw_hline(cycles);
}
else
{
Serial.flush();
}
Serial.flush();
}
// DRAWING FUNCTIONS
// Infinity Sign drawing function
// exercise 1
void draw_inf(int ncycles)
{
int current_state_1 = 0;
int current_state_2 = 0;
for(int count = 0; count < ncycles; count++)
```

```
{
for(int i = 0; i < 348; i++)
{
if (inf_set2time_motor1[i] < 0)
{
runMotor1CCW(current_state_1, -1 * inf_set2time_motor1[i]);
current_state_1 = (current_state_1 + (-1 * inf_set2time_motor1[i])) % 4 ;
}
else if (inf_set2time_motor1[i] == 0)
{
digitalWrite(wire1,LOW);
digitalWrite(wire2,LOW);
digitalWrite(wire3,LOW);
digitalWrite(wire4,LOW);
delay(20);
}
else
{
runMotor1CW(current_state_1, inf_set2time_motor1[i]);
current_state_1 = (current_state_1 + inf_set2time_motor1[i]) % 4;
}
int k=i;
if (inf_set2time_motor2[k] < 0)
{
runMotor2CCW(current_state_2, -1 * inf_set2time_motor2[k]);
current_state_2 = (current_state_2 + (-1 * inf_set2time_motor2[k])) % 4 ;
}
else if (inf_set2time_motor2[k] == 0)
{
digitalWrite(wire1,LOW);
digitalWrite(wire2,LOW);
digitalWrite(wire3,LOW);
digitalWrite(wire4,LOW);
delay(20);
}
else
{
runMotor2CW(current_state_2, inf_set2time_motor2[k]);
current_state_2 = (current_state_2 + inf_set2time_motor2[k]) % 4;
}
}
}
}
// Vertical Line drawing function
// exercise 2
void draw_vline(int ncycles)
{
int current_state_1 = 0;
int current_state_2 = 0;
for(int count = 0; count < ncycles; count++)
```

```
{
for(int i = 0; i < 106; i++)
{
if (vline_set2time_motor1[i] < 0)
{
runMotor1CCW(current_state_1, -1 * vline_set2time_motor1[i]); current_state_1 =
(current_state_1 + (-1 * vline_set2time_motor1[i])) % 4 ;
}
else if (vline_set2time_motor1[i] == 0)
{
digitalWrite(wire1,LOW);
digitalWrite(wire2,LOW);
digitalWrite(wire3,LOW);
digitalWrite(wire4,LOW);
delay(20);
}
else
{
runMotor1CW(current_state_1, vline_set2time_motor1[i]);
current_state_1 = (current_state_1 + vline_set2time_motor1[i]) % 4;
}
int k=i;
if (vline_set2time_motor2[k] < 0)
{
runMotor2CCW(current_state_2, -1 * vline_set2time_motor2[k]);
current_state_2 = (current_state_2 + (-1 * vline_set2time_motor2[k])) % 4 ;
}
else if (vline_set2time_motor2[k] == 0)
{
digitalWrite(wire1,LOW);
digitalWrite(wire2,LOW);
digitalWrite(wire3,LOW);
digitalWrite(wire4,LOW);
delay(20);
}
else
{
runMotor2CW(current_state_2, vline_set2time_motor2[k]);
current_state_2 = (current_state_2 + vline_set2time_motor2[k]) % 4;
}
}
}
}
// Horizontal Line drawing function
// exercise 3
void draw_hline(int ncycles)
{
int current_state_1 = 0;
int current_state_2 = 0;
for(int count = 0; count < ncycles; count++)
```

```
{
for(int i = 0; i < 240; i++)
{
if (hline_set2time_motor1[i] < 0)
{
runMotor1CCW(current_state_1, -1 * hline_set2time_motor1[i]);
current_state_1 = (current_state_1 + (-1 * hline_set2time_motor1[i])) % 4 ;
}
else if (hline_set2time_motor1[i] == 0)
{
digitalWrite(wire1,LOW);
digitalWrite(wire2,LOW);
digitalWrite(wire3,LOW);
digitalWrite(wire4,LOW);
delay(20);
}
else
{
runMotor1CW(current_state_1, hline_set2time_motor1[i]);
current_state_1 = (current_state_1 + hline_set2time_motor1[i]) % 4;
}
int k=i;
if (hline_set2time_motor2[k] < 0)
{
runMotor2CCW(current_state_2, -1 * hline_set2time_motor2[k]);
current_state_2 = (current_state_2 + (-1 * hline_set2time_motor2[k])) % 4 ;
}
else if (hline_set2time_motor2[k] == 0)
{
digitalWrite(wire1,LOW);
digitalWrite(wire2,LOW);
digitalWrite(wire3,LOW);
digitalWrite(wire4,LOW);
delay(20);
}
else
{
runMotor2CW(current_state_2, hline_set2time_motor2[k]);
current_state_2 = (current_state_2 + hline_set2time_motor2[k]) % 4;
}
}
}
}
// MOTOR HARDWARE DRIVE FUNCTIONS
void runMotor1CW(int starting_point_id, int num_steps)
{
int j;
for(j = 0; j < num_steps; j++)
{
digitalWrite(wire1, stepper_motor_states_cw[(starting_point_id + j) % 4][0]);
```

```
digitalWrite(wire2, stepper_motor_states_cw[(starting_point_id + j) % 4][1]);
digitalWrite(wire3, stepper_motor_states_cw[(starting_point_id + j) % 4][2]);
digitalWrite(wire4, stepper_motor_states_cw[(starting_point_id + j) % 4][3]);
delayMicroseconds(100000 / num_steps);
}
}
void runMotor1CCW(int starting_point_id, int num_steps)
{
int j;
for(j = 0; j < num_steps; j++)
{
digitalWrite(wire1, stepper_motor_states_ccw[(starting_point_id + j + 1) % 4][0]);
digitalWrite(wire2, stepper_motor_states_ccw[(starting_point_id + j + 1) % 4][1]);
digitalWrite(wire3, stepper_motor_states_ccw[(starting_point_id + j + 1) % 4][2]);
digitalWrite(wire4, stepper_motor_states_ccw[(starting_point_id + j + 1) % 4][3]);
delayMicroseconds(100000 / num_steps);
}
}
void runMotor2CW(int starting_point_id_2, int num_steps_2)
{
int z;
for(z = 0; z < num_steps_2; z++)
{
digitalWrite(wire5, stepper_motor_states_cw[(starting_point_id_2 + z) % 4][0]);
digitalWrite(wire6, stepper_motor_states_cw[(starting_point_id_2 + z) % 4][1]);
digitalWrite(wire7, stepper_motor_states_cw[(starting_point_id_2 + z) % 4][2]);
digitalWrite(wire8, stepper_motor_states_cw[(starting_point_id_2 + z) % 4][3]);
delayMicroseconds(100000 / num_steps_2);
}
}
void runMotor2CCW(int starting_point_id_2, int num_steps_2)
{
int z;
for(z = 0; z < num_steps_2; z++)
{
digitalWrite(wire5, stepper_motor_states_ccw[(starting_point_id_2 + z + 1) % 4][0]);
digitalWrite(wire6, stepper_motor_states_ccw[(starting_point_id_2 + z + 1) % 4][1]);
digitalWrite(wire7, stepper_motor_states_ccw[(starting_point_id_2 + z + 1) % 4][2]);
digitalWrite(wire8, stepper_motor_states_ccw[(starting_point_id_2 + z + 1) % 4][3]);
delayMicroseconds(100000 / num_steps_2);
}
}
```

Chapter 9
Results and future work

The manupilandum was programmed to move in three different patterns in three different speeds as per the scope of the work. The torque provided by the motors however was not enough to be resistive torque. Resistive torque can be provided by importing high torque motors. The project can be further be improved by implementing a closed loop feedback system to provide real time location of the end effector. Furthermore, the friction can be reduced by using magnetic support instead of caster balls at points of high moment.

References

[1] (CJ, Stroke Rehabilitation: A Function-Based Approach, 2007)

[2] (http://www.pakstroke.com/, 2013)

[3] (A Journal of Cerberal Circulation, 1999)

[4] (A Journal of Cerebral Circulation, 2001)