Gesture Designated & Voice Controlled Robotic Hand Using Myograph Sensors

Bhavy Shah	15012041060
Mohit Bhalala	15012041077
Abhinav Lakhani	15012041078
Raghav Pathak	15012041080

FINAL YEAR PROJECT REPORT

(NOV-DEC 2018)

Gesture Designated & Voice Controlled Robotic Hand Using Myograph Sensors

DEPT. OF MECHATRONICS ENGINEERING

U. V. PATEL COLLEGE OF ENGINEERING

GANPAT UNIVERSITY

GUIDED BY:

Prof. V. D, Patel
Dept. of Mechatronics Engg.

SUBMITTED BY:

Bhavy Shah	15012041060
Mohit Bhalala	15012041077
Abhinav Lakhani	15012041078
Raghav Pathak	15012041080

ACKNOWLEDGEMENT

I am very thankful to Prof. J. P. Patel, Head of Department, Mechatronics Engineering, U.V Patel College Engineering, Ganpat University for giving me the opportunity to undertake my project "Gesture Designated & Voice Controlled Robotic Hand Using Myograph Sensors".

I express my sincere gratitude to Prof. V. D. Patel, Dept. of Mechatronics Engineering, U.V Patel College Engineering, Ganpat University for his guidance, continuous encouragement and supervision throughout the course of present work.

Special thanks goes to Prof. J. K. Prajapati and Prof. D. K. Soni for providing knowledge. We are also obliged to all the faculty of Department of Mechatronics Engineering.

I am extremely thankful to U.V Patel College of Engineering, Ganpat University for providing us the theoretical knowledge, infrastructural facilities and productive environment to work, without which this work would not have been possible.

ABSTRACT

Each day in present, someone is trying to build something; a machine, which can replicate humans as nearly as possible. This replication is what we call humanoid. Here, we present the making of a necessary part of this humanoid, its hands. The hand will be functional from shoulder up to finger-tips. The fingers and palm are 3-D printed and do not use tendon for movement. Gesture is primarily used to feed data & make it learn. While major concern of voice controlling is to give orders & obtain task performance. Myographic sensors are used for fulfilling gesture recognition purpose. The educational object of this project was to learn the skills to solve an engineering design problem in a cooperative environment.

This report presents a general background of the project along with a description of the overall group dynamics. The report is organized into two main sections: the design and manufacture of the arm, which was the major task for us, and the controls system of the arm which is based on Machine Learning and Artificial Intelligence.

The Design section contains description of The Base, Forearm and Wrist, while the control system contains acquiring of data from EMG Sensor, Machine Vision and Voice Command to provide data processing and efficient output.

TABLE OF CONTENTS

LIST OF FIGURES

DRAWING SHEET

LITERATURE REVIEW

1. 'Manipulator": A machine that has functions similar to human upper limbs and moves object spatially. "Robotics”: A reprogrammable with minimum of four degrees of freedom designed to both manipulate and transport parts, tools or specialized equipments through variable programmable motion to perform various task. - [I] R.K.Mittal, I.J.Nagrath, Robotics and Control, Tata McGraw Hill,2007

2. The concept of dexterity and its dimension have been difficult to define through the years and various research communities have definitions of dexterity. In this paper, we present the concept of dexterity and its dimensions have been difficult to define through the years, and various research communities have a different definitions of dexterity. In this paper, we present a novel approach to understand people's perception of dexterity and its different dimensions in daily tasks. The approach entails a video-based survey where people score different tasks along pre-specified dimensions of dexterity. The results show how the dimensions contribute to an overall dexterity score as well as the importance of each dimension. Index Terms-Dexterity, Hand Function, Daily tasks. [5]- Ravi Balasubramanian, Brian Dellon, Sujata Pradhan, Spencer Gibbs, and Yoky

INTRODUCTION

There are certain tasks which humans cannot perform and situations where humans need assistance. Also, there are several long term or repetitive operations which humans are incapable to do or exhibit it inconsistently.

The core motivation is to build something that can have multifunctionality in accordance with the requirement basis. Robotic hand, here is a child, which if made to learn shooting, fighting, etc. will make it useful in defense sector. It can be used to assist humanity in its daily life like cooking, cleaning, etc. It can be used as a helping hand to perform rescue operations in conditions and places where humans cannot reach. If it is made to learn and identify differences in materials, functions like sample gathering, etc. it can be used to explore any corner of space under any conditions.

Mentioned above and many other countless objectives can be fulfilled with the only condition that this robotic hand system is attached to a functional mechanical body.

The concept of tendon-less movement of fingers has been originally developed by Festo in making its 3 fingered gripper mechanism. There are certain projects available online showing use of myograph sensors in making gesture controlled robotic hand. However, this use is constrained only up to fingers and wrist movement. Also, there are gesture controlled robotic hands developed using other sensors like accelerometer, gyroscope, neural sensors, etc.

There are many research initiatives carried out globally which make robotic hand. These robotic hand systems even use complex components like pneumatic drives, electromagnetic drives, etc. Various robotic systems using artificial intelligence & machine learning are under research globally.

The robotic hand is easily attachable with various bodies which are differently used for various day-to-day tasks. It is more suitable to use such mechanical body at places which are hazardous for human to work at. It is aimed to be providing more dexterity compared to many other similar models available globally due to provision of its own brain using AI & ML as well as 7 degree of freedom. Gesture designated learning enables to replicate and copy human actions and style of performing tasks. It is also polymath as it can be reprogrammed according to various required scenarios independently by different operators. It can be made to perform tasks by giving simple audio commands.

<u>Software requirements:</u>

- Raspbian (OS)
- Python 3
- MATLAB & Simulink
- Solidworks
- Proteus
- Microsoft Azure (CP)
- ROS (If required)

Hardware requirements:

- Arduino
- EMG(Electromyography) sensor & electrodes
- Accelerometer
- Microphone
- Raspberry Pi 3B+
- Tactile/Flex sensor
- Stepper motors
- Servo motors
- Li-po batteries and connecting wires
- Tilt pan motor brackets
- 3-D printed fingers and palm
- Aluminium frame/sections
- Motor mountings
- Support pole
- Workspace base
- Clamps

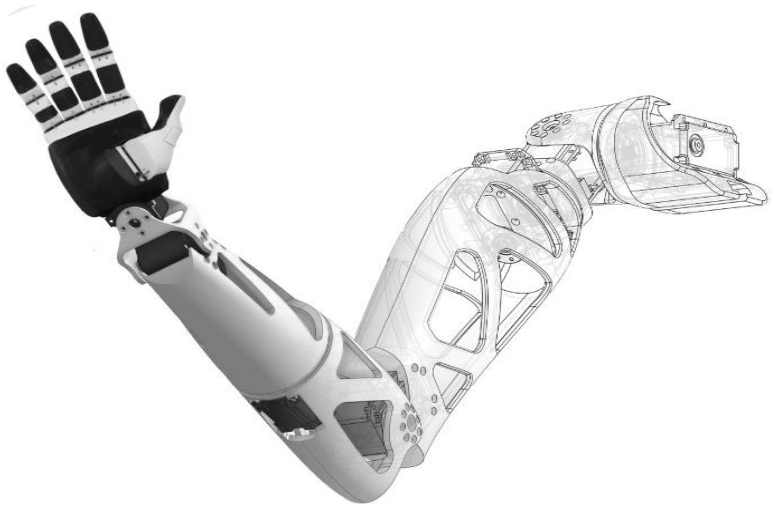


Fig. 2.1: A Robotic Hand

MECHANICAL DESIGN

3.1 OVERVIEW

After considering and observing the various robotic hands and manipulators that are currently existing globally, we have made our own modifications and designed this hand according to the best possible outcome which we considered to be more beneficial for implementation in practical applications in various ways like Industrial, Household, Hospitality, Defense etc.

The robotic hand consists of 3 parts; Upper Arm, Fore Arm and Wrist. The wrist consists of tendonless grippers inspired by Festo Adaptive Grippers which gives major advantage of absence of links and metal strings for controlling the fingers. The forearm, upper arm and wrist are sized proportionally accordance to human hand.

The design is made to achieve and replicate the actions of human hand precisely. This provides us with 7 Degrees of Freedom for movement in robotic hand to perform a specific task; which also increases its dexterity.

The currently available models consist of either of EMG, Voice Command or Machine Vision. Our design consists of all three parameters and their integral implementation.

3.2 OBJECTIVES

At the initial meeting in June 2018, our team met to discuss the project and how the design of the robotic arm would be achieved. After reviewing the previous works on the project, we decided on some design objectives and constraints that would make sure that our robotic arm would be an improvement upon the previous designs.

The design needed to be cheaper and lighter than that of other variants available in market. Other improvements that were to be implemented were to increase the dexterity of the arm and to have a fully functional control system. By the end of the meeting, we decided to stand upon following parameters for the new robotic manipulator:

- 7 Degrees of freedom
- Maximum weight of entire assembly less than 5 kgs.
- Fully-functional control system with reprogrammable memory.
- Lift a 0.5 kg mass.
- Maximum cost of ₹ 50,000 including the controls.

3.3 TERMINOLOGY

Early in the project, it was realized that describing the different motions, parts, and joints of the arm was difficult. Therefore, members of both teams decided to create a standard terminology in order to make communication easier and less confusing.

The basis of the terminology was the human arm. The main parts of the arm were chosen as the base, upper arm, forearm, and palm. The joints were defined as they would be on the human arm, shoulder, elbow, and wrist. The shoulder is the joint between the base and the upper arm. The elbow is the joint between the upper arm and forearm. Finally, the wrist is between the forearm and the palm. Since these three joints account for 7 degrees of freedom, the motion of each of the joints were added to the terminology.

The human hand has got 7 degree of freedom. As this is the replica made for a human hand, it also has 7 degree of freedom. The wrist has got 3 degree of freedom; the elbow has got 2 degree of freedom as well as the shoulder has got 2 degree of freedom.

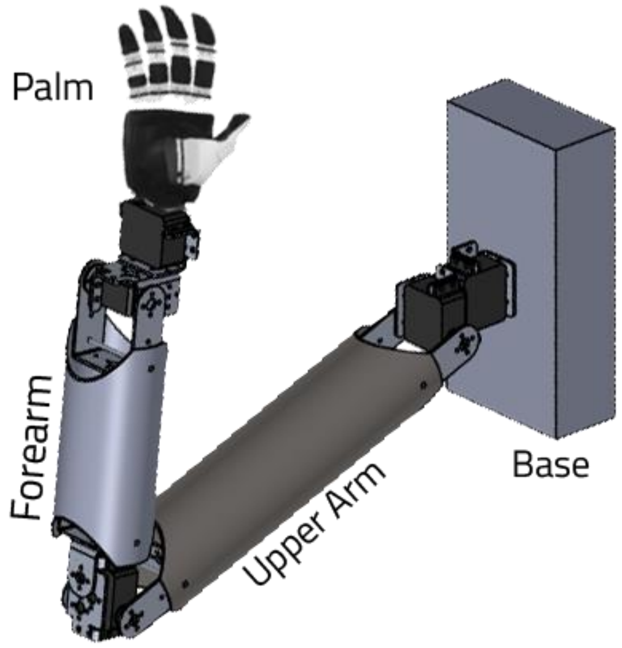


Fig. 3.1: Terminology of Robotic Hand

3.4 COMPONENTS

The motor selection was based on torque calculations of the hand. The effective load was considered as 0.5 kg and self-weight of fingers was considered for knuckle motors. The load and palm weight were considered for wrist motors and similarly, forearm weight for elbow motors and upper arm weight for shoulder motors. The load distribution was done according to UDL and point load to obtain the torque requirements.

The obtained values are:

- Finger Torque = 0.5 kgcm
- Wrist Torque = 10.5 kgcm
- Elbow torque = 38 kgcm
- Shoulder Torque = 90 kgcm

The motor selection was based on several factors like availability of motors, cost parameters, clearances in load values, dimensions of product etc. The basic options available Servo Motors, Stepper Motors, Micro Servo and Micro Stepper Motors.

As per above requirements and design considerations, we decided to utilize Micro-Stepper Motors for fingers, Servo Motors for wrist and elbow, and Stepper Motors for shoulder

3.4.1 MICRO STEPPER MOTOR

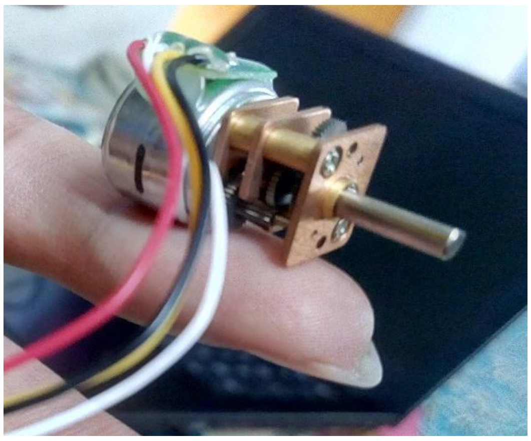


Fig. 3.2: Micro Stepper Motor

SPECIFICATION

- Stepper Motor:
- Coil resistance: 30Ω±10%
- NO. Of Phases: 2 PHASES
- Step Angle: 18° / Step
- MAX.Starting Frequency: 900PPS min
- MAX.Slewing Frequency: 1200 PPS min
- Pull IN Torque: 6gf-cm min
- Lead Out Torque: 10gf-cm max
- Stepper Motor with Gear Box:
- Step Angle: 18/100° / Step
- MAX.Starting Frequency: 900PPS min
- MAX.Slewing Frequency: 1200 PPS min
- Pull IN Torque: 0.5Kgf-cm min
- Lead Out Torque: 0.8Kgf-cm max
- Cable Length: 15cm
- Weight: 14g

3.4.2 SERVO MOTOR

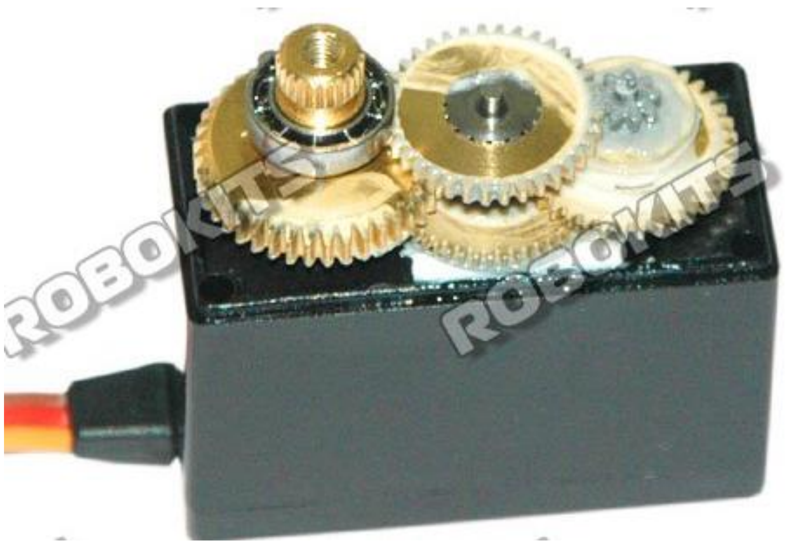


Fig 3.3: Servo Motor

SPECIFICATION

- Required Pulse: 3-5 Volt Peak to Peak Square Wave
- Operating Voltage: 4.8-6.0 Volts
- Operating Temperature Range: -10 to +60 Degree C
- Operating Speed (4.8V): 0.18sec/60 degrees at no load
- Operating Speed (6.0V): 0.14sec/60 degrees at no load
- Stall Torque (4.8V): 14kg/cm
- Stall Torque (6.0V): 16kg/cm
- 360 Modifiable: Yes
- Potentiometer Drive: Indirect Drive
- Bearing Type: Double Ball Bearing
- Gear Type: All Metal Gears
- Connector Wire Length: 12"
- Dimensions: 1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
- Weight: 48gm

CONTROL SECTION

4.1 OVERVIEW

The electrodes of the myograph sensor are placed on the pre-decided positions of the forearm, upper arm and shoulder of human hand respectively. These electrodes sense the muscle movement & activities and convert them in the form of electric signals in waveform of the motion performed. This obtained signals will be send to the Arduino which is further connected to the Robotic hand. Thus, on the basis of the signal obtained, the robotic hand will perform the motion and replicate the human hand. This process is the learning procedure or the gesture designation procedure for the robotic hand.

The motion performed by the robotic hand will be saved textually as well as an audio name will be linked with it. It will be programmed such that when the audio is delivered using microphone, it will work as input for the hand and the hand will perform the motion that will be linked with that specific audio file. The storing of this audio files with its linked motion, the object files, other program sources will be saved in external memory which will be installed in the Raspberry Pi to carry out the process and procedure required in getting the whole output. This whole procedure is the part of project delivered as the Voice controlling of the robotic hand.

Here, the issue that comes in action is that if for sample purpose, “pick and place” operation is taken into consideration, the method to hold different sized and shaped objects will be different. This is where the AI & ML part comes into action. The robotic hand will be provided with its own intelligence of using Machine Vision, it will identify the size and the shape of object. Now, multiple pick & place methods will be linked with a single audio file. Using its AI, it will decide by itself that which of all the linked methods is most suitable for the operation to be performed. Thus, it will carry out the task. This process is the portion that makes the robotic hand smart.

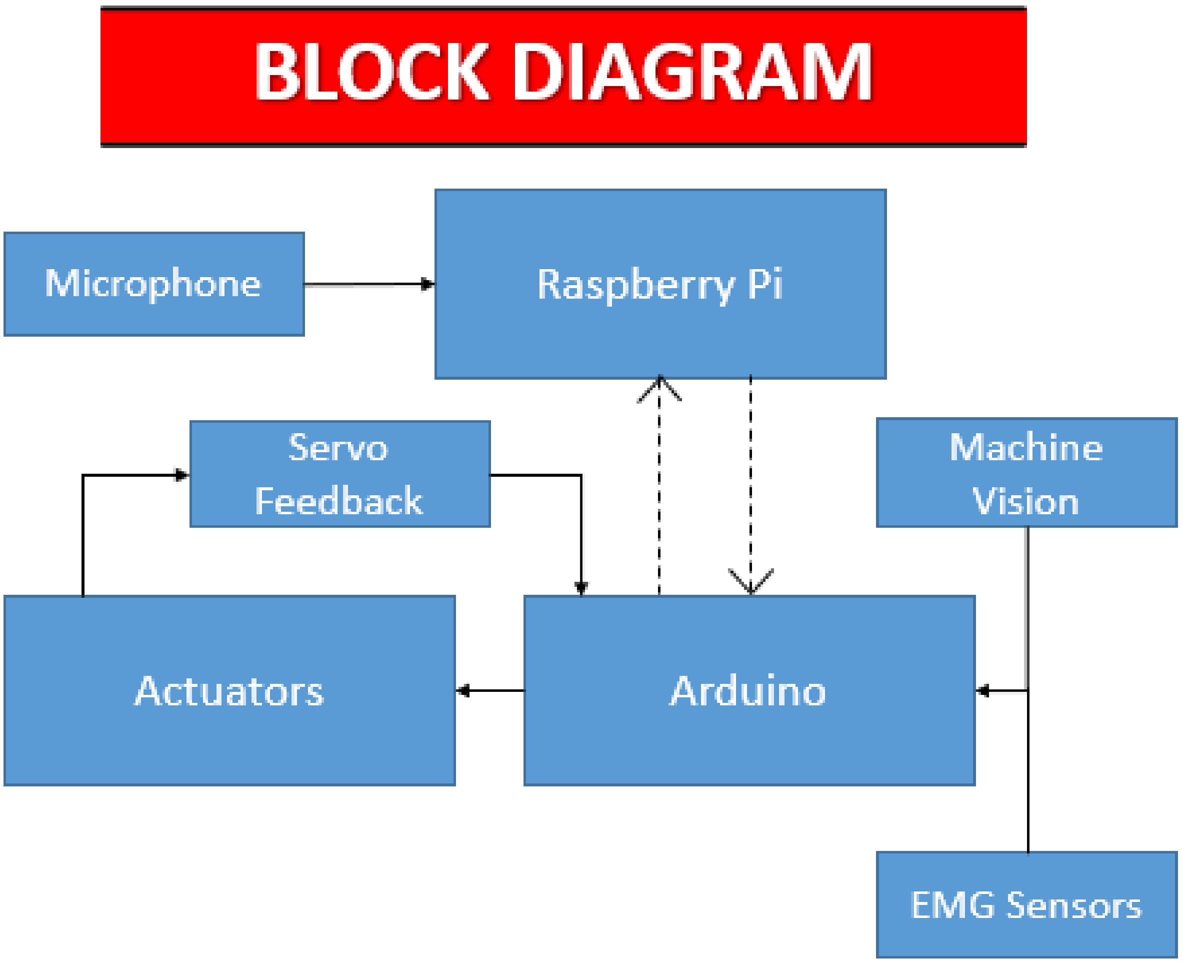


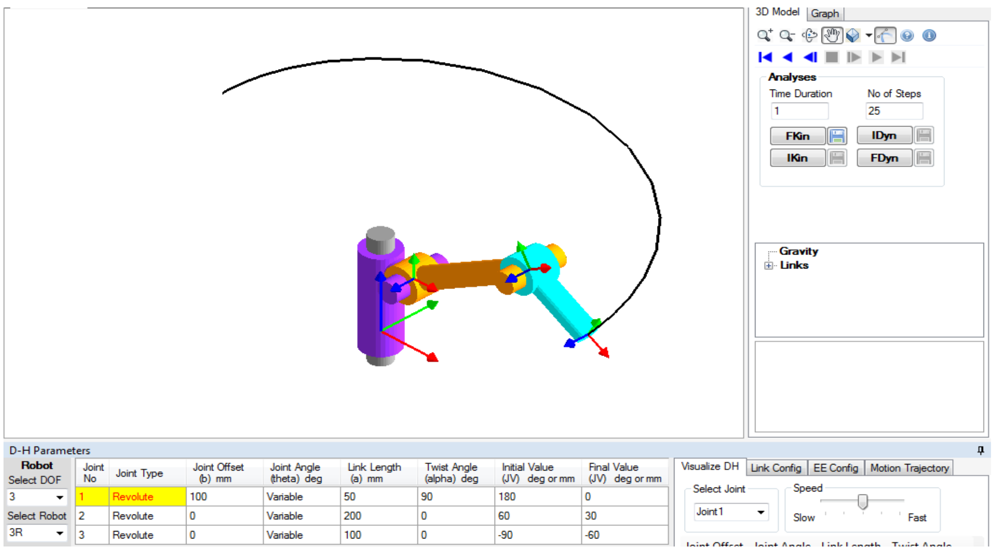
Fig. 4.1: Control System Block Diagram

Also, the flex sensor is used in the fingers to read the contact made between the surfaces of the object and the fingers. When the suitable reading is obtained, the current flow in the finger motor stops, avoiding exertion of unnecessary pressure on the object.

Using Machine Vision, the position of the object will be obtained. To reach that position, the forward kinematics & inverse kinematics will be implemented on the joints of the robotic hand and it will reach at the calculated position where the task needs to be performed.

4.2 ROBOT KINEMATICS

For determining the position and orientation of end effector, we will use the forward kinematics to get transformation matrix of each joints which will help us visualize robot's position and orientation in 3D space. Similarly, the path travelled by the hand will be determined by the position and orientation of object in 3D space. The co-ordinates will act as input for inverse kinematic equations to obtain the angular movement of each joint actuators. The solutions to forward and inverse kinematics have been implemented in the programming.



Time Duration: 1
No of Steps: 25

D-H Parameters

Robot
Select DOF: 3
Select Robot: 3R

Joint No	Joint Type	Joint Offset (b) mm	Joint Angle (theta) deg	Link Length (a) mm	Twist Angle (alpha) deg	Initial Value (JV) deg or mm	Final Value (JV) deg or mm
1	Revolute	100	Variable	50	90	180	0
2	Revolute	0	Variable	200	0	60	30
3	Revolute	0	Variable	100	0	-90	-60

Select Joint: Joint1
Speed

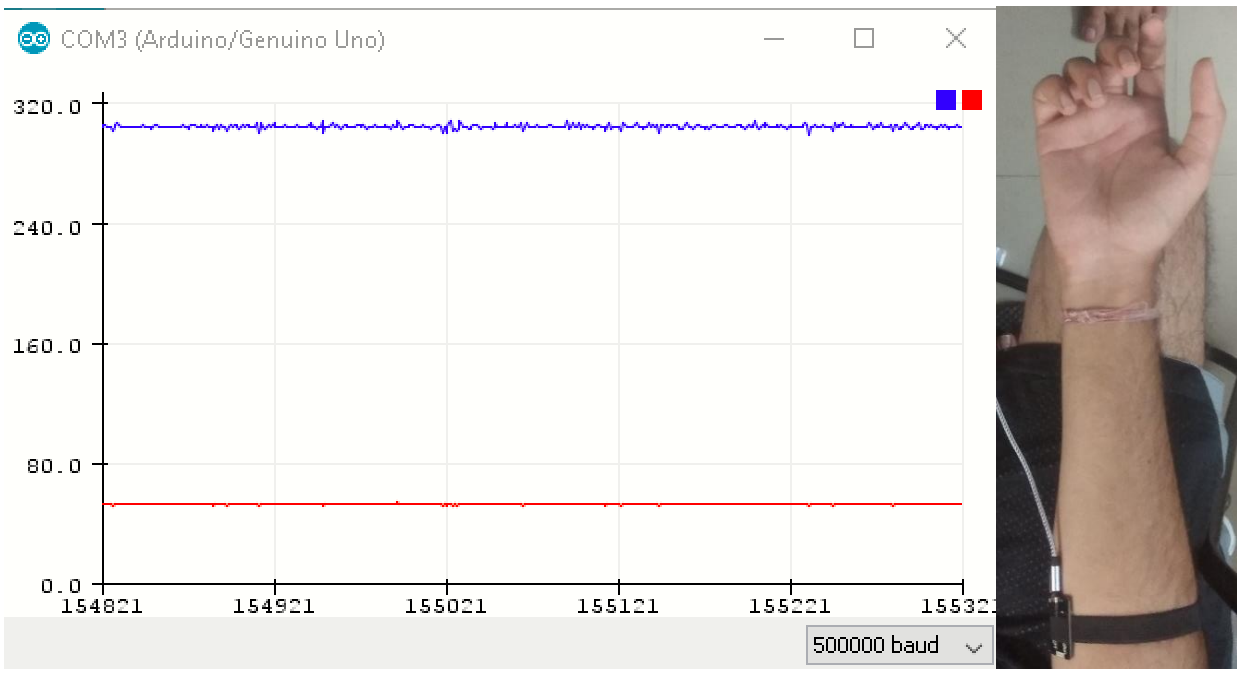
Fig: 4.2: Kinematics Simulation

4.3 ROBOT PROGRAMMING

4.3.1 EMG SIGNAL DETECTION IN ARDUINO

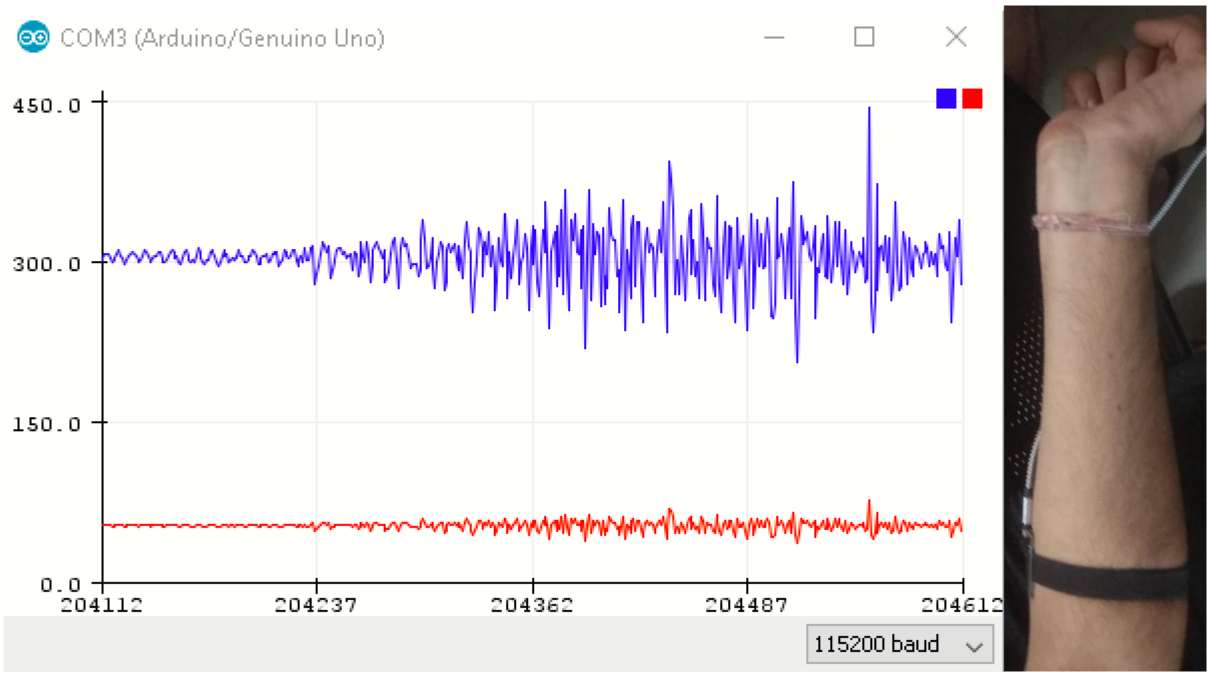
Unfiltered Program(Arduino):

```
// These constants won't change. They're used to give
names to the pins used:
const int analogInPin = A0;  // Analog input pin that
the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that
the LED is attached to
int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM
(analog out)

void setup() {
  // initialize serial communications at 115200 bps:
  Serial.begin(115200);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 179);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  //Serial.println(outputValue);
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the
analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```



500000 baud

Fig. 4.3: Unfiltered & Thresholded [no action/force]



115200 baud

Fig. 4.4: Unfiltered & Thresholded [medium force]

4.3.2 EMG SIGNAL FILTERING

filtered Program(Arduino):

```
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include "EMGFilters.h"
#define TIMING_DEBUG 1
#define SensorInputPin A0 // input pin number

EMGFilters myFilter;
// discrete filters must works with fixed sample
frequence
// our emg filter only support "SAMPLE_FREQ_500HZ" or
"SAMPLE_FREQ_1000HZ"
// other sampleRate inputs will bypass all the
EMG_FILTER
int sampleRate = SAMPLE_FREQ_1000HZ;
// For countries where power transmission is at 60 Hz,
need to change to
// "NOTCH_FREQ_60HZ"
// this emg filter only support 50Hz and 60Hz input
// other inputs will bypass all the EMG_FILTER
int humFreq = NOTCH_FREQ_50HZ;
// Calibration:
// put on the sensors, and release muscles;
// wait a few seconds, and select the max value as the
threshold;
// any value under threshold will be set to zero
static int Threshold = 0;

unsigned long timeStamp;
unsigned long timeBudget;

void setup() {
    /* add setup code here */
    myFilter.init(sampleRate, humFreq, true, true,
true);

    // open serial
    Serial.begin(500000);
    // setup for time cost measure
    // using micros()
    timeBudget = 1e6 / sampleRate;
    // micros will overflow and auto return to zero
every 70 minutes
}

void loop() {
    /* add main program code here */
    // In order to make sure the ADC sample frequence
on arduino,
    // the time cost should be measured each loop
    /*------------start here--------------------*/
    timeStamp = micros();

    int Value = analogRead(SensorInputPin);
    int mapped = 0;

    // filter processing
    int DataAfterFilter = myFilter.update(Value);

    int envlope = sq(DataAfterFilter);
    // any value under threshold will be set to zero
    envlope = (envlope > Threshold) ? envlope : 0;

    timeStamp = micros() - timeStamp;
    if (TIMING_DEBUG) {
        //Serial.print("Read Data: ");
Serial.println(Value);
        //Serial.print("Filtered Data:");
Serial.println(DataAfterFilter);

        Serial.print("Squared Data: ");
        //Serial.println(envlope);
        mapped = map(envlope, 0, 16000, 0, 255);
        Serial.println(mapped);

        //Serial.print("Filters cost time: ");
Serial.println(timeStamp);
        // the filter cost average around 520 us
    }

    /*------------end here---------------------*/
    // if less than timeBudget, then you still have
(timeBudget - timeStamp) to
```

```
    // do your work
    delayMicroseconds(500);
    // if more than timeBudget, the sample rate need to
reduce to
    // SAMPLE_FREQ_500HZ
}
```

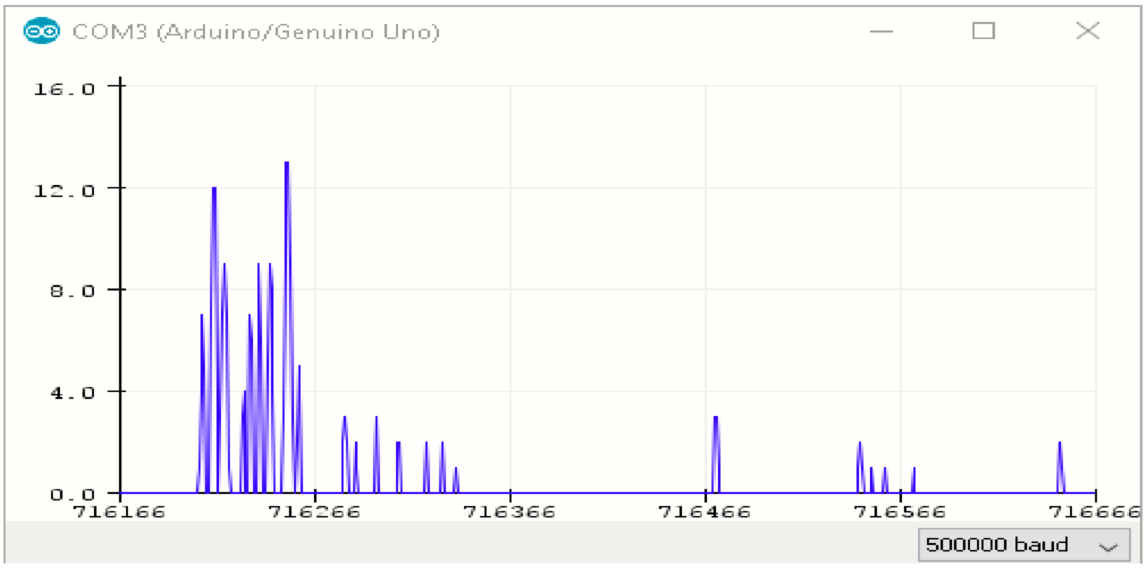


Fig. 4.5: Filtered force 10

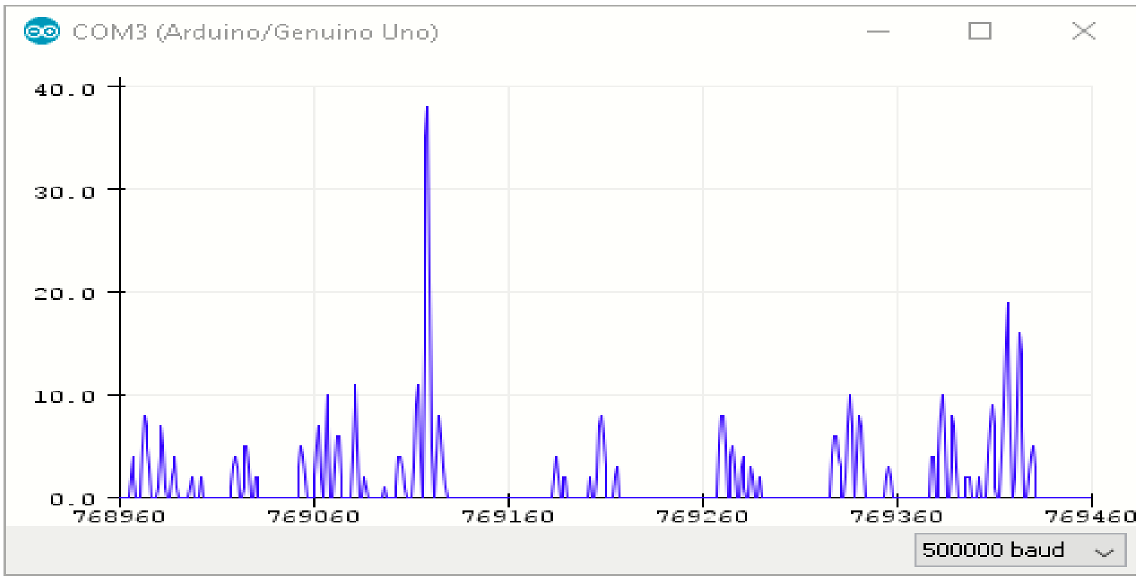


Fig. 4.6: Filtered force 50

4.3.3 ARDUINO RASPBERRY PI COMMUNICATION

pyFirmata is a Python interface for the Firmata protocol. It runs on Python 2.x and 3.x.

Firmata is a generic protocol for communicating with microcontrollers from software on a host computer. It is intended to work with any host computer software package. Right now there is a matching object in a number of languages. It is easy to add objects for other software to use this protocol. Basically, this firmware establishes a protocol for talking to the Arduino from the host software. The aim is to allow people to completely control the Arduino from software on the host computer.

Basic usage:

- from pyfirmata import Arduino, util
- board = Arduino('/dev/tty.usbserial-A6008rIF')
- board.digital.write(1)

To use analog ports, it is probably handy to start an iterator thread. Otherwise the board will keep sending data to the serial, until it overflows:

```
it = util.Iterator(board)
it.start()
board.analog[0].enable_reporting()
board.analog[0].read()
0.661440304938
```

Board layout:

To use a board with a different layout than the standard Arduino or the Arduino Mega (for which there exist the shortcut classes pyfirmata.Arduino and pyfirmata.ArduinoMega), instantiate the Board class with a dictionary as the layout argument. This is the layout for the Mega for example:

```
mega = {
         'digital' : tuple(x for x in range(54)),
         'analog' : tuple(x for x in range(16)),
         'pwm' : tuple(x for x in range(2,14)),
         'use_ports' : True,
         'disabled' : (0, 1, 14, 15) # Rx, Tx, Crystal
         }
```

4.3.4 MATLAB FORWARD KINEMATICS PROGRAM

```
%% Helper script to visualize forward kinematics of a
the robot
% ynx_fk function

close all
pause on;  % Set this to on if want to watch the
animation
GraphingTimeDelay = 0.05; % The length of time that
Matlab should pause between positions when graphing, if
at all, in seconds.
totalTimeSteps = 100; % Number of steps in the
animation

% Generate the starting and final joint angles, and
gripper distance
% Set values manually, or randomise as shown below
q01 = 0;
q02 = 0;
q03 = 0;
```

```
q04 = 0;
q05 = 0;
q06 = 0;
q07 = 0;

q11 = 0;
q12 = 0;
q13 = 0;
q14 = 0;
q15 = 0;
q16 = 0;
q17 = 0;

% Use below code if want to randomise
% q01 = rand(1) * 2 * pi;
% q02 = rand(1) * 2 * pi;
% q03 = rand(1) * 2 * pi;
% q04 = rand(1) * 2 * pi;
% q05 = rand(1) * 2 * pi;
% q06 = rand(1) * 2 * pi;
% q07 = rand(1) * 2 * pi;
% g0 = rand(1) * 2;
%
% q11 = rand(1) * 2 * pi;
% q12 = rand(1) * 2 * pi;
% q13 = rand(1) * 2 * pi;
% q14 = rand(1) * 2 * pi;
% q15 = rand(1) * 2 * pi;
% q17 = rand(1) * 2 * pi;
% q16 = rand(1) * 2 * pi;
% g1 = 0;

% Setup plot
figure
scale_f = 100;
axis vis3d
axis(scale_f*[-1/2 1/2 -1/2 1/2 -1/4 1/2])
grid on
view(70,10)
xlabel('X (in.)')
ylabel('Y (in.)')
zlabel('Z (in.)')

% Plot robot initially
hold on
```

```
hrobot = plot3([0 0 10], [0 0 0], [0 6 6],'k.-
','linewidth',1,'markersize',10);

% Animate the vector
pause(GraphingTimeDelay);
for i = 1:totalTimeSteps

    t = i/totalTimeSteps;
    [pos, R] = RPR_fk(q01*(1-t) + q11*(t), q02*(1-t) +
q12*(t),...
        q03*(1-t) + q13*(t),q04*(1-t) + q14*(t),q05*(1-
t) + q15*(t),...
        q06*(1-t) + q16*(t), q07*(1-t) + q17*(t));
%test - 1
%     pos =[0         0         0;
%          0         0    3.0000;
%     0.0000   -0.0000    8.7500;
%     7.3750   -0.0000    8.7500;
%     7.3750   -0.0000    8.7500;
%    10.3750   -0.0000    8.7500;
%    10.3750   -0.0000    7.7500;
%    10.3750   -0.0000    9.7500;
%    11.5000   -0.0000    7.7500;
%    11.5000   -0.0000    9.7500];
%test - 2
%    pos = lynx_fk(pi,pi/2,pi/2,-pi/2,-pi/6,2);


    set(hrobot,'xdata',[pos(1, 1) pos(2, 1) pos(3, 1)
pos(4, 1) pos(5, 1) pos(6, 1) pos(7, 1) pos(8, 1)]',...
        'ydata',[pos(1, 2) pos(2, 2) pos(3, 2) pos(4,
2) pos(5, 2) pos(6, 2) pos(7, 2) pos(8, 2)]',...
        'zdata',[pos(1, 3) pos(2, 3) pos(3, 3) pos(4,
3) pos(5, 3) pos(6, 3) pos(7, 3) pos(8, 3)]');

    pause(GraphingTimeDelay);
end
pos
```

Function File:

```
function [ pos, R ] = RPR_fk( theta1, theta2, theta3,
theta4, theta5, theta6, theta7)
%    The input to the function will be the joint
%    angles of the robot in radians, and the extension
of the prismatic joint in inches.
%    The output includes:
%    1) The position of the end effector and the
position of
%    each of the joints of the robot
%    2) The rotation matrix R_06
    xl=1;%small link length
    l1=xl;
    l2=xl;
    L3=15;
    l4=xl;
    L5=12.5;
    l6=xl;
    l7=xl;

    pos = zeros(8, 3);
    R = eye(3);
    %first rotate w.r.t Y-axis
    temp = [0 0 -1 0; 0 1 0 0; 1 0 0 0; 0 0 0 1];

    tf1 = temp*compute_dh_matrix(0,                  (-
pi/2),        l1,      theta1);
    tf2 = tf1*compute_dh_matrix(0,
pi/2,         -l2,     theta2-pi/2);
    tf3 = tf2*compute_dh_matrix(0,                   -
pi/2,          L3,     theta3-pi/2);
    tf4 = tf3*compute_dh_matrix(0,
pi/2,          l4,     theta4+pi/2);
    tf5 = tf4*compute_dh_matrix(0,                   -
pi/2,          L5,     theta5+0);
    tf6 = tf5*compute_dh_matrix(0,                   -
pi/2,          l6,     theta6-pi/2);
    tf7 = tf6*compute_dh_matrix(0,
0,           l7,     theta7+0);

    pos = [0        0        0;
           tf1(1,4) tf1(2,4) tf1(3,4);
           tf2(1,4) tf2(2,4) tf2(3,4);
           tf3(1,4) tf3(2,4) tf3(3,4);
           tf4(1,4) tf4(2,4) tf4(3,4);
           tf5(1,4) tf5(2,4) tf5(3,4);
           tf6(1,4) tf6(2,4) tf6(3,4);
           tf7(1,4) tf7(2,4) tf7(3,4)];
    R = tf7(1:3, 1:3);
end
%compute D-H table
function A = compute_dh_matrix(r, alpha, d, theta)
    A = [cos(theta) -sin(theta)*cos(alpha)
sin(theta)*sin(alpha) cos(theta)*r;
        sin(theta) cos(theta)*cos(alpha) -
cos(theta)*sin(alpha) sin(theta)*r;
        0          sin(alpha)                cos(alpha)
d;
        0          0                         0
1];
end
```

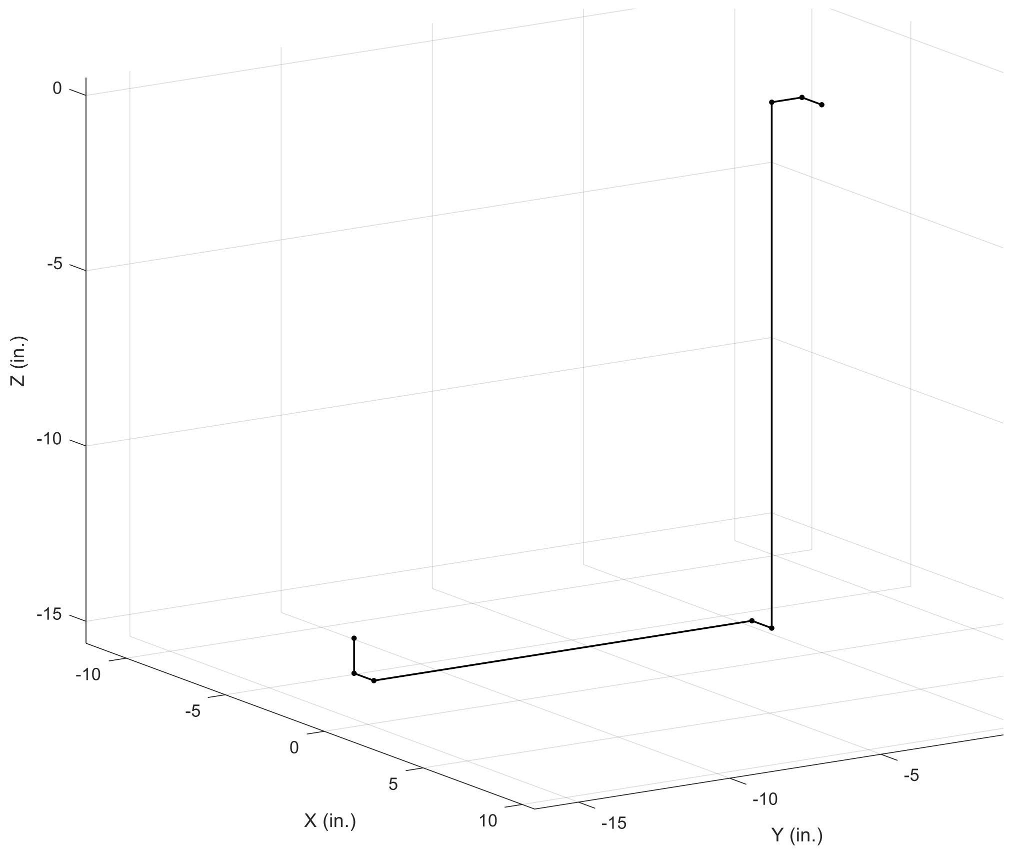


Fig. 4.7: Forward Kinematics Output

4.3.5 STEPPER MOTOR PROGRAM

Stepper Motor Control - speed control.

This program drives a unipolar or bipolar stepper motor. The motor is attached to digital pins 8 - 11 of the Arduino. A potentiometer is connected to analog input 0. The motor will rotate in a clockwise direction. The higher the potentiometer value, the faster the motor speed. Because setSpeed() sets the delay between steps, one may notice the motor is less responsive to changes in the sensor value at low speeds.

```
#include <Stepper.h>
const int stepsPerRevolution = 200;  // change this to
fit the number of steps per revolution
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);
int stepCount = 0;  // number of steps the motor has
taken
void setup() {}
void loop() {
  int sensorReading = analogRead(A0);
  // map it to a range from 0 to 100:
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  if (motorSpeed > 0) {
    myStepper.setSpeed(motorSpeed);
    // step 1/100 of a revolution:
    myStepper.step(stepsPerRevolution / 100);
  }
}
```

FUTURE SCOPE

The current version of Robotic Hand is made considering the offline model and programs are stored in External data memory of on-board computing device.

The future aspects include improved versions of Machine Vision or 3D Sensor. This will not only improve depth sensing skills but also enhance the object recognition and face recognition features to produce its effective utilization.

The voice and data processing requires sample of voice; so if new user wants to use it, his voice needs to be recorded. Hence considering the home automation project, use of Voice Assistant like Google Home, Alexa can be corresponded to use the robotic hand, which will save time of recording every unique sample of operator and provides ease of access.

Also, the data needs to be accessible at different locations around the world over a range of devices like PC, laptop, mobile etc. and so the use of Cloud Computing services such as Google Drive, Microsoft Azure etc. will play a major role in increasing accessibility and manipulation of the programs.

The current model has limitations in amount of load lifted due to economical parameters. The increase in budget of project will allow the use of high toque motors and high quality 3D printed micro structured parts increasing the performance of robotic hand.

REFERENCES

https://www.theseus.fi/bitstream/handle/10024/147196/thesis_susan.pdf?sequence=1&isAllowed=y

https://pdfs.semanticscholar.org/0137/98a83b43f0f749dc018e5269397a93f5b492.pdf

https://makezine.com/2013/02/08/teen-creates-3d-printed-brain-powered-prosthetic-arm/

https://github.com/caplingerc/Roboarm/blob/master/Roboarm.ino

https://www.hackster.io/29949/how-to-drive-a-servo-with-emg-signals-5b3715

https://backyardbrains.com/experiments/muscleSpikerShield

https://www.youtube.com/watch?v=1LjE07z5r7c&feature=youtu.be

https://www.youtube.com/watch?v=D-6GDlvAMCI&feature=youtu.be

https://player.vimeo.com/video/154571244

https://pressureprofile.com/oem-robotics

https://www.dfrobot.com/product-1661.html

https://circuitdigest.com/microcontroller-projects/controlling-arduino-with-raspberry-pi-using-pyfirmata

https://pyfirmata.readthedocs.io/en/latest/pyfirmata.html?highlight=serial

https://circuitdigest.com/microcontroller-projects/controlling-arduino-with-raspberry-pi-using-pyfirmata

https://github.com/nanpy/nanpy

http://houseofbots.com/news-detail/3409-4-a-data-science-for-good-machine-learning-project-walk-through-in-python-part-one

https://www.festo.com/cat/en_us/products_DHAS