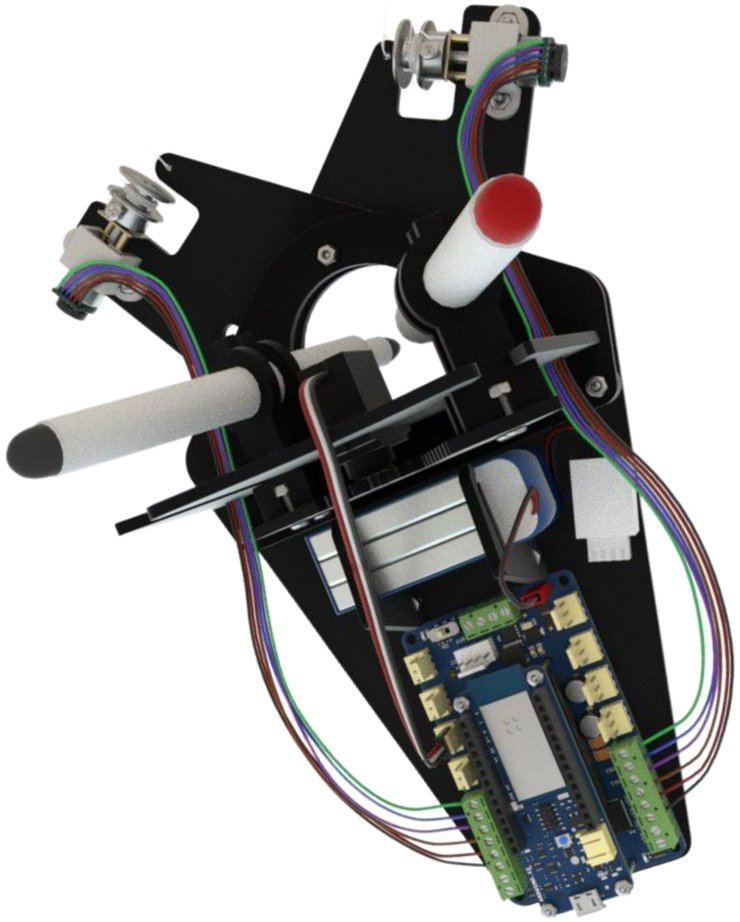
CONTROL OF STRING DRIVEN 2D CARTESIAN PLANAR ROBOT



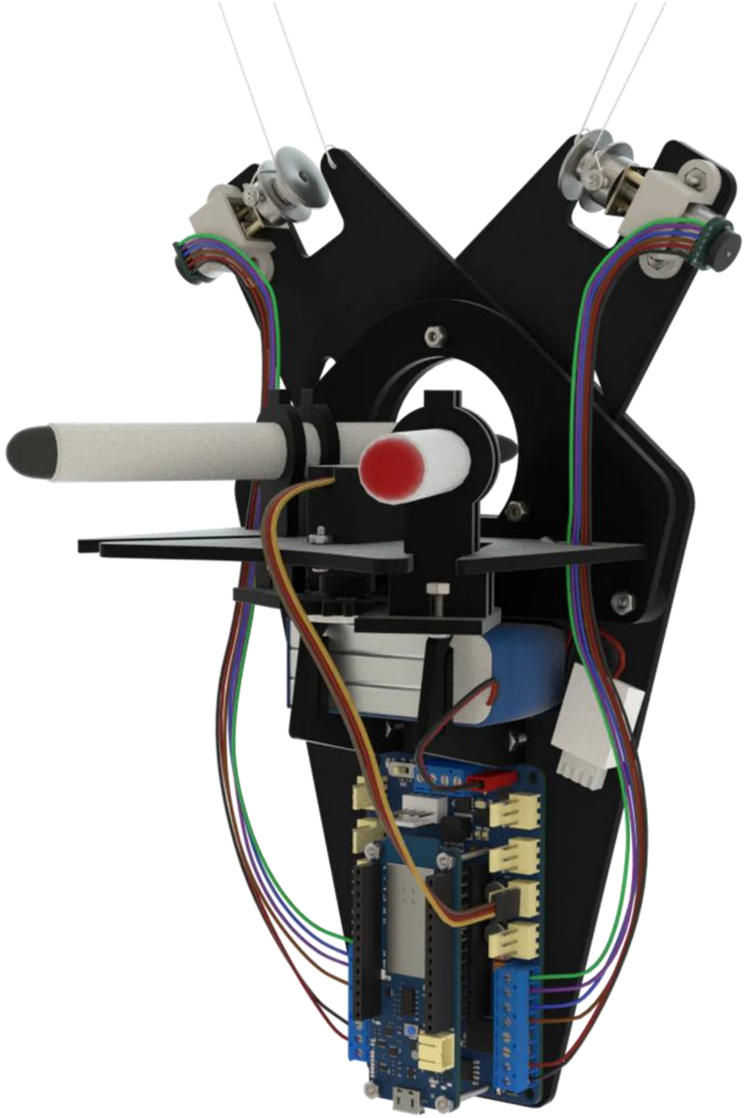
Anuj Satish Zore
Guide: Prof. Abhra Roy Chowdhury

CONTENT

INTRODUCTION

This project covers drawing robot's assembly, electronic hardware integration and MATLAB programming.

Usability: This robot is capable of drawing 2D text / figures in the XY plane with provision of choosing between red and black marker.



LEARNINGS

During this project we learned

- **MATLAB Programming**
- **N20 Polulu DC Motor control**
- **SG-90 Servo Motor Duty-Cycle Control**
- **Understanding Coordinate Control**
- **Basic Image Processing**

Reference : https://aek.arduino.cc/chapter/drawing-robot

TUTORIALS

- Components Test - SG90 Servo Motor
  - N20 Polulu Motor (150 RPM)
  - MKR Motor Carrier
  - Pulley Mechanism
- Whiteboard Position Limits Test
- Wifi Configuration and Control
- Exercise 1 : Controlling Drawing Robot With GUI
- Exercise 2 : Drawing "CPDM" in Open Loop
- Exercise 3 : Algorithm/Flowchart for image processing and drawing
- Exercise 4 : Drawing Preprocessed Image
- Exercise 5 : Drawing Live Image

ASSEMBLY VIDEOS AND COMPONENTS LIST

Components List

- SG90 Tower Servo Motor
- N20 Polulu Geared DC Motor With Encoder
- MKR1000
- MKR Motor Carrier
- Jumper Wires
- Others

Reference : https://aek.arduino.cc/chapter/drawing-robot

ASSEMBLED MODEL DIMENSIONS

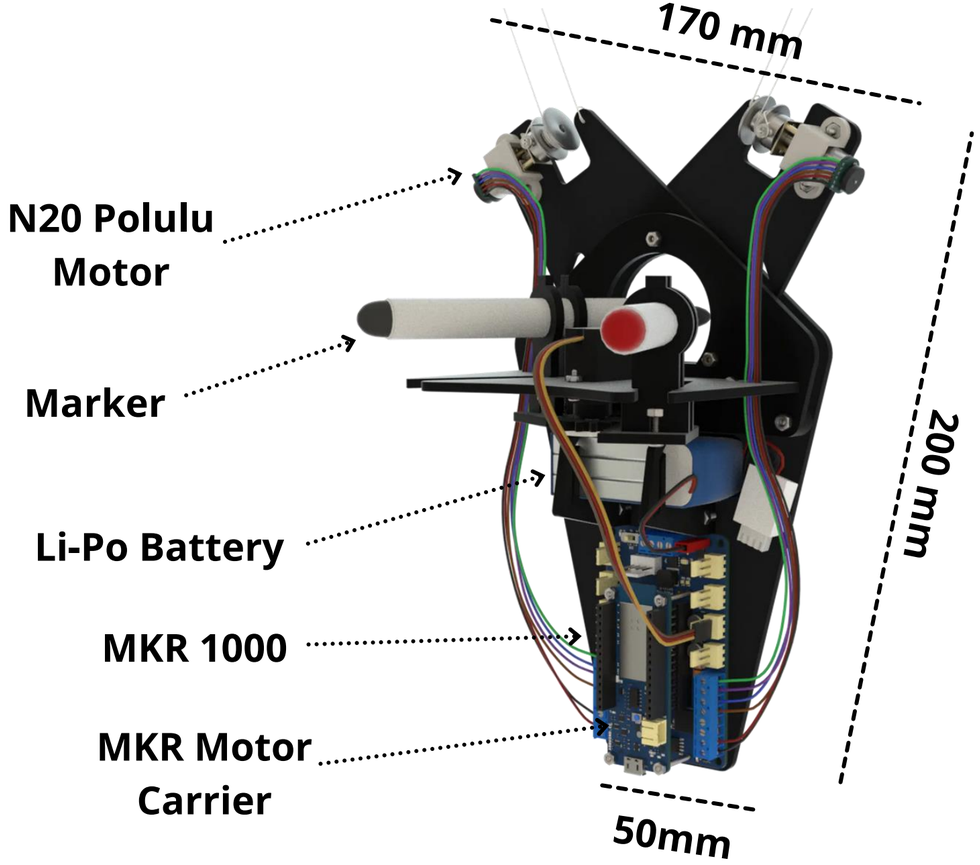


Fig 0 : Assembled Model labelled diagram

Reference : https://aek.arduino.cc/chapter/drawing-robot

COMPONENTS & SPECIFICATION

Fig 1 SG90 Tower Servo Motor

Model: SG90 Servo Motor

Control System: PPM

Working Frequence: 1520µs / 50hz

(RX) Required Pulse: 3.3 ~ 5 Volt Peak to Peak Square Wave

Operating Voltage: 4.8 ~ 6 V DC Volts

Dimensions: 23 x 11.5 x 24mm

Weight: 9 grams

Fig 2 N20 Polulu Geared DC Motor With Encoder

Encoder Test Wire Length: 140mm.

Wheel Dia.: 43 mm.

Rated Voltage: DC 3V.

Output Speed: 150RPM.

Output Shaft Size: 3 x 10mm.

Output Shaft type: D type.

Shipment weight : 0.016 kg

Shipment Dimensions : 6 × 4 × 3 cm

Fig 3 MKR1000

Microcontroller : SAMD21 Cortex-M0+ 32bit low power ARM MCU

Board Power Supply (USB/VIN): 5V

Circuit Operating Voltage: 3.3V

Digital I/O Pins: 8

PWM Pins 12 : (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - or 18 -, A4 -or

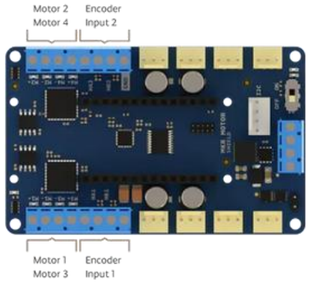


Fig 4 MKR Motor Carrier

Microcontroller : ATSAMD11 (Arm Cortex-M0+ processor)

Max current (MC33926) : 5 Amps Peak

Max current (DRV8871) : 3 Amps peak ; 6.5 to 11.1V

On board voltage regulator : 5V

COMPONENT TESTS

This section will verify if all the components are connected and responding properly to our system.

Component Test :

- **Tower Pro SG90**
- **MKR1000**
- **Motor Carrier**

Setting Up Arduino

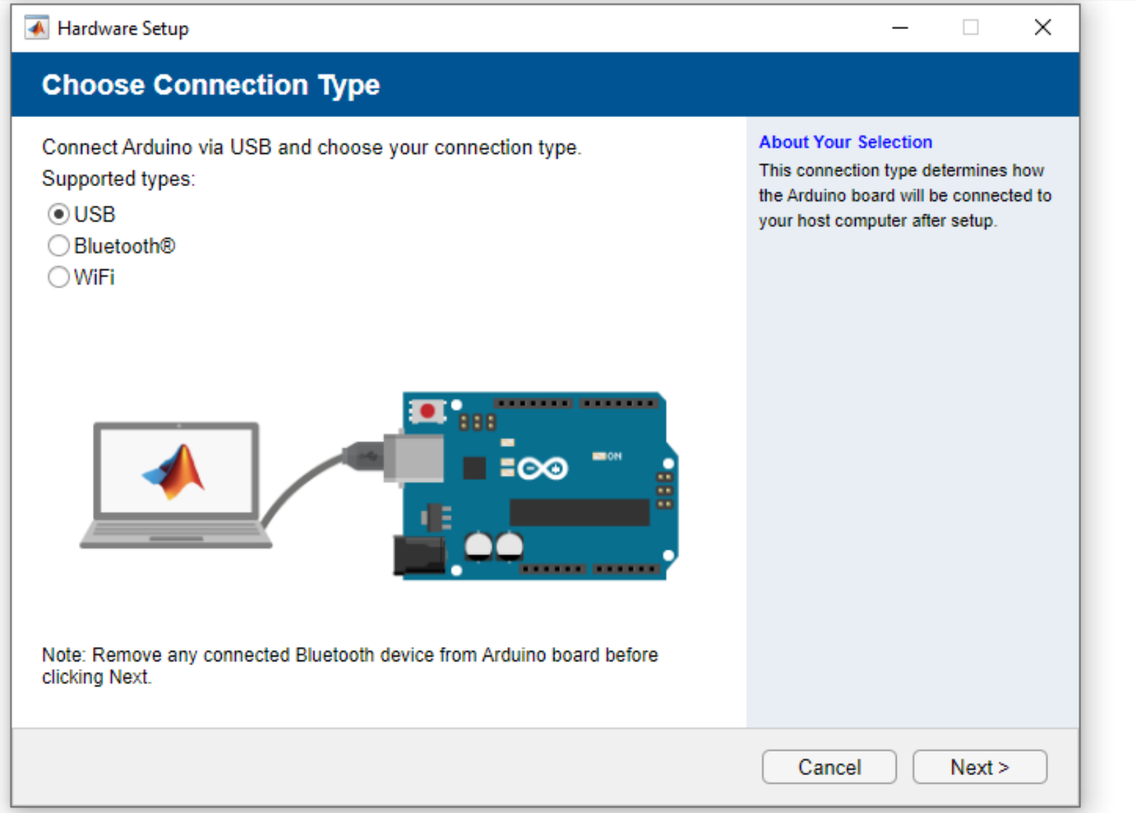


Fig 6 Arduino Setup Preview

- **Connect the MKR1000 to computer and run the arduinosetup command at the MATLAB command prompt.**
- **This will launch the Arduino hardware setup interface as shown in left figure.**

Tower Pro SG90 Test

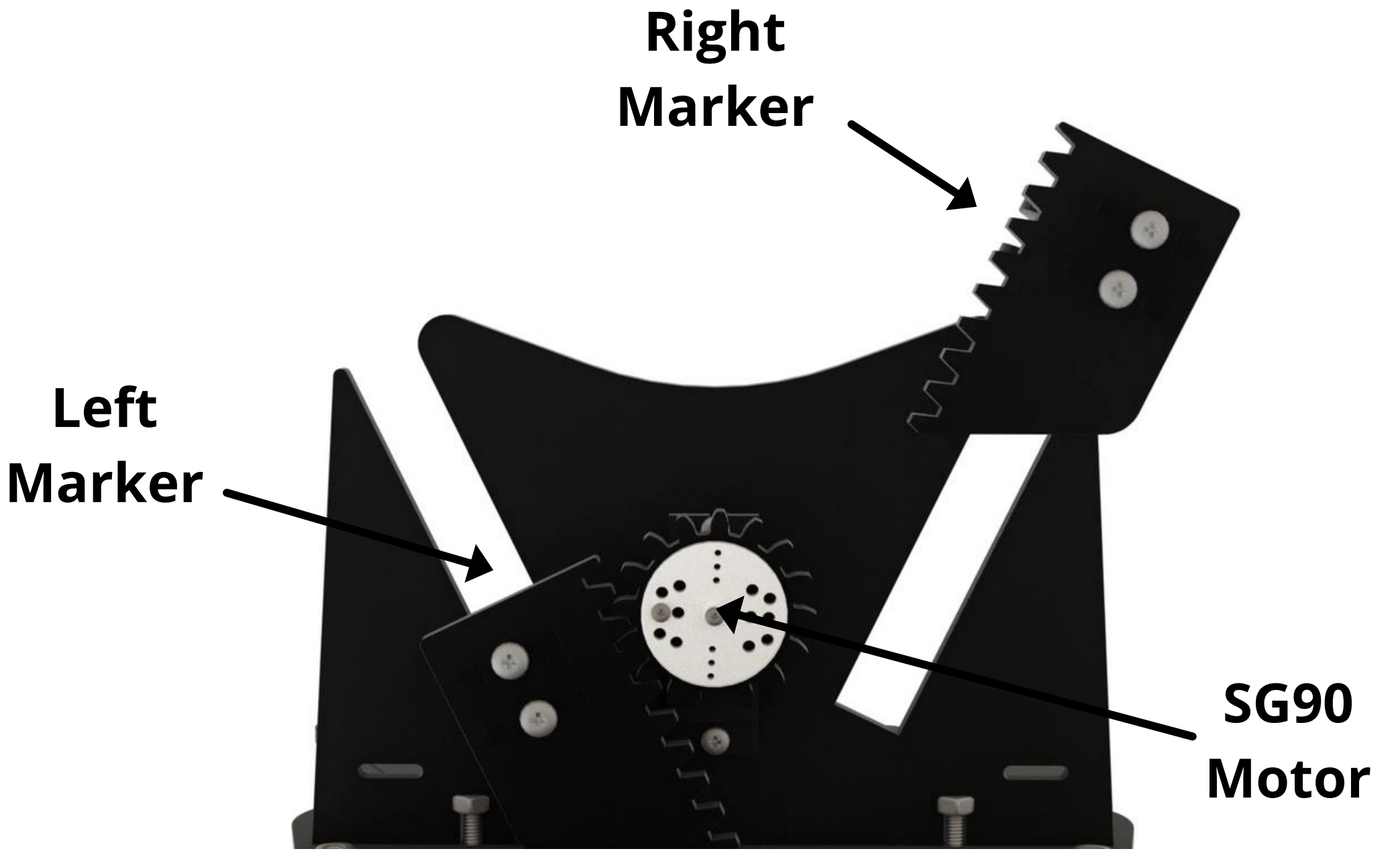


Fig 7 Marker Adjustment Mechanism

The drawing robot has two different color markers that can be raised and lowered by means of a servo motor.

MKR 1000 & MKR Carrier

Fig 8 MKR 1000

- Check for power LED as shown in the figure.
- In case of any error with USB detection, open the device manager and reinstall the USB driver for our microcontroller.

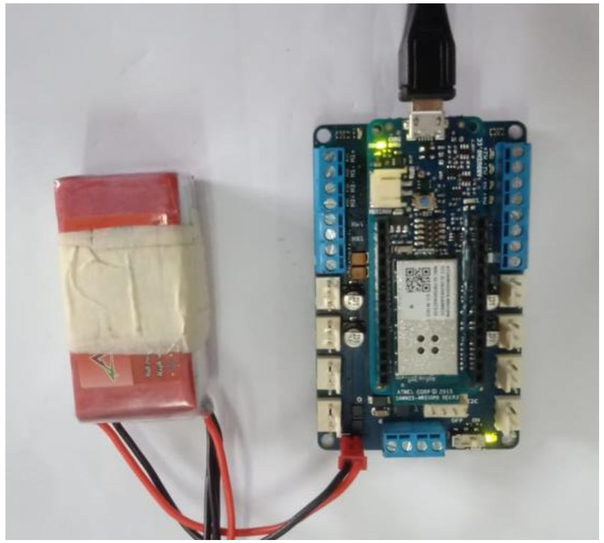


Fig 9 Motor Carrier

Connect the battery terminal properly and switch on the switch.

Exercises

1] Controlling Drawing Robot With GUI

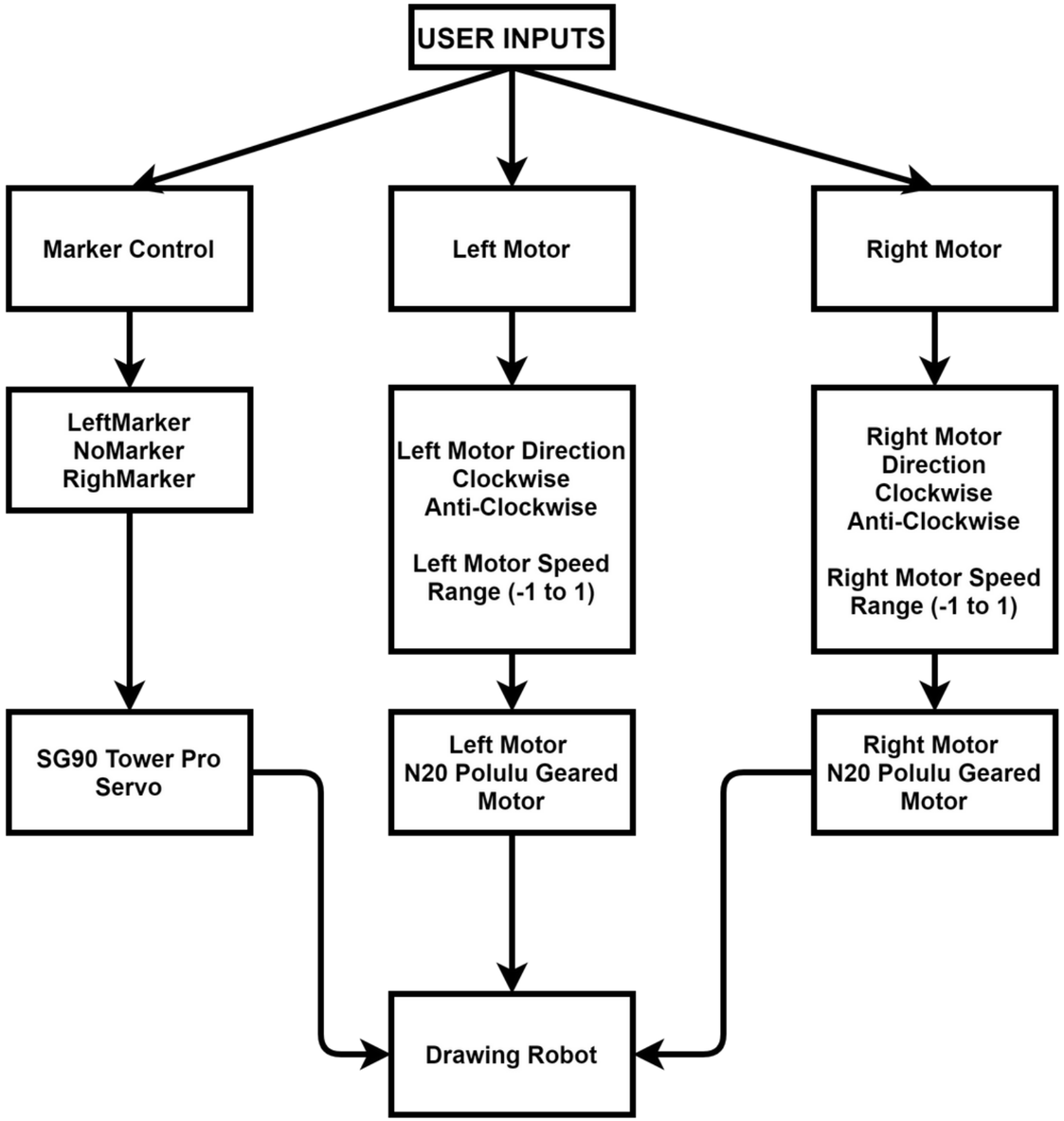
2] Drawing "CPDM" in Open

3] Algorithm / Flowchart for Image Processing & Drawing Loop

4] Drawing Preprocessed Image (Closed Loop)

5] Drawing Live Image (Closed Loop)

EXERCISE 1
CONTROLLING ROBOT WITH GUI



FlowChart

GUI CONTROLLED ROBOT FLOW CHART

In GUI control window drawing robot has three actuators to control i.e. Marker, Left Motor, Right Motor.

1] three marker positions are predefined in accordance with the appropriate duty cycle.

2] In both Left and right Motor Control, depending on the direction of the motor input values ranges from 0 (Min Speed) to 1(Max Speed).

3] The resultant motion can be observed on Drawing Robot

GUI CONTROL MODE

Marker Selection Box

Left Motor Speed Control

Right Motor Speed Control

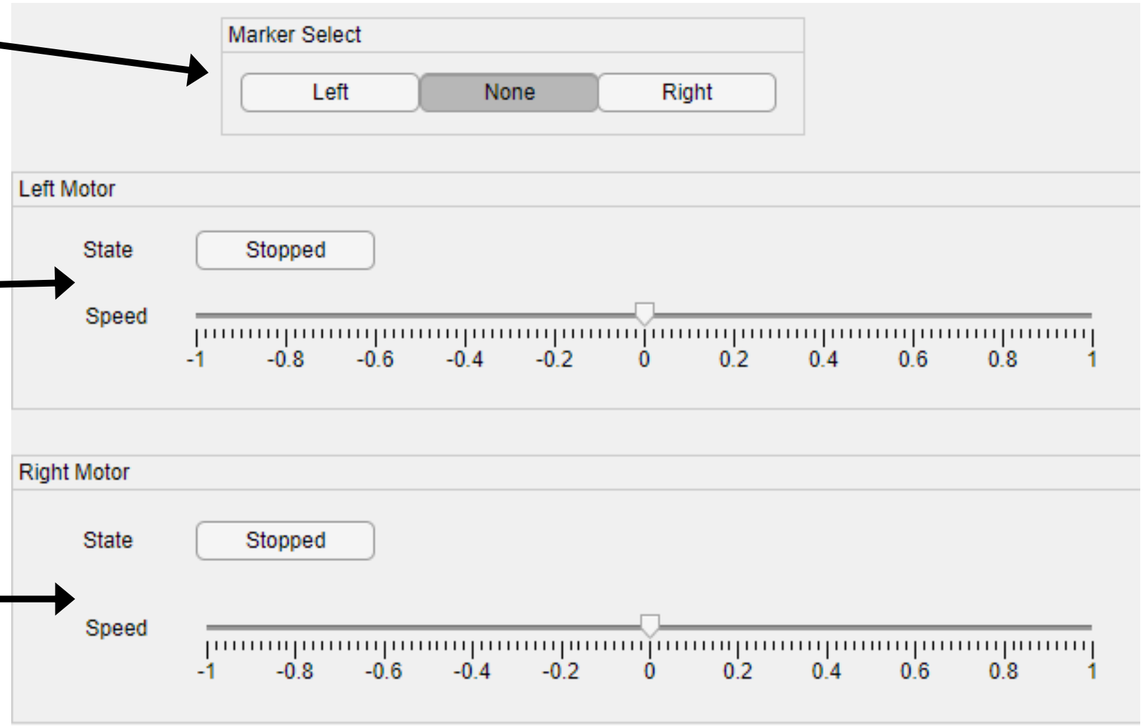
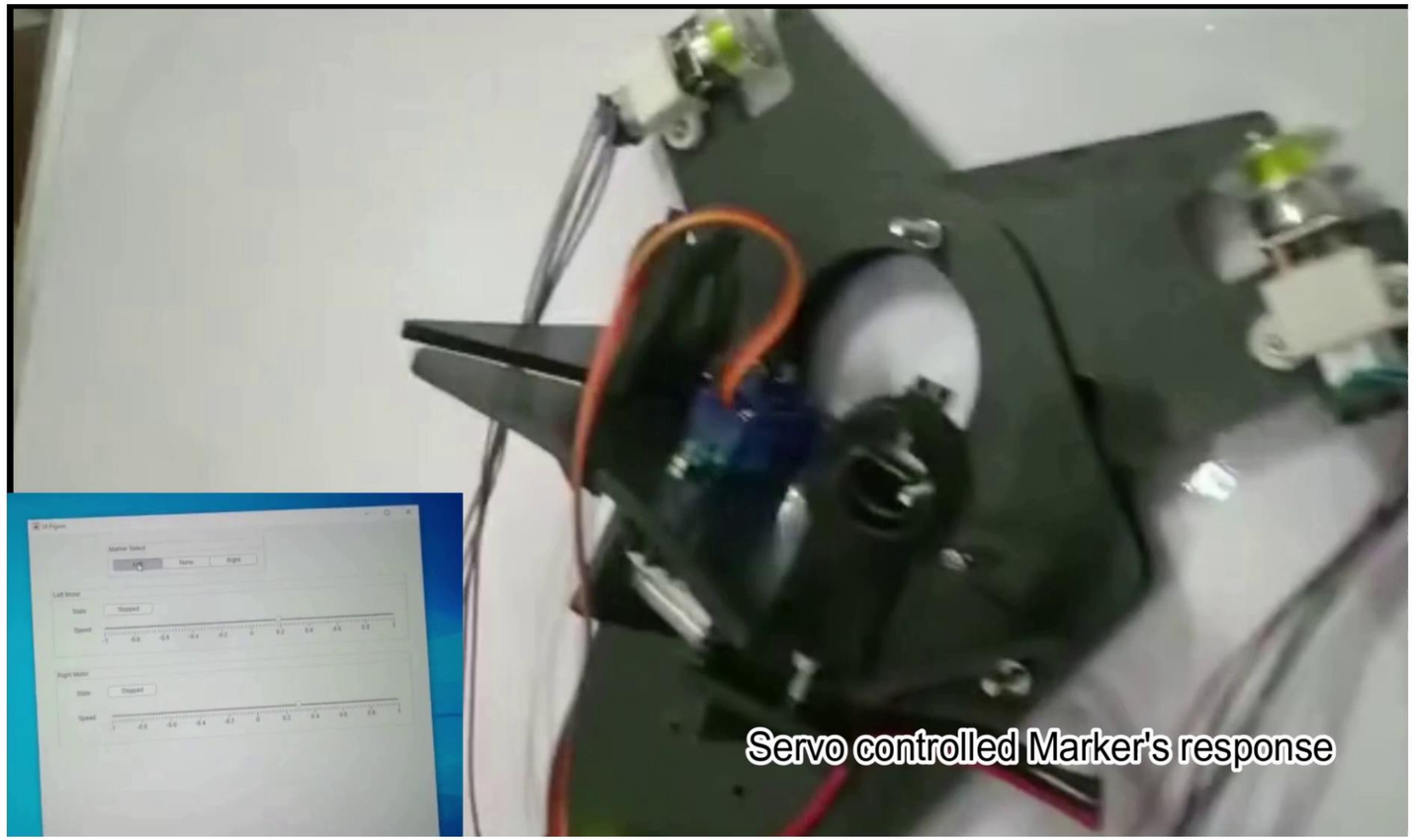


Fig 10 Preview of GUI Demonstration

- Both mL and mR are moved in the XY plane.
- change in duty cycle = change the marker positions = change in drawing color.
- The linear scale ----> velocity commands Stop function ----> start or stop the motor.

GUI CONTROLLED DRAWING ROBOT



Video 01 Video Preview of GUI Demonstration

EXERCISE 2
Drawing "CPDM" in Open Loop

OPEN LOOP DRAWING'S PROCEDURE

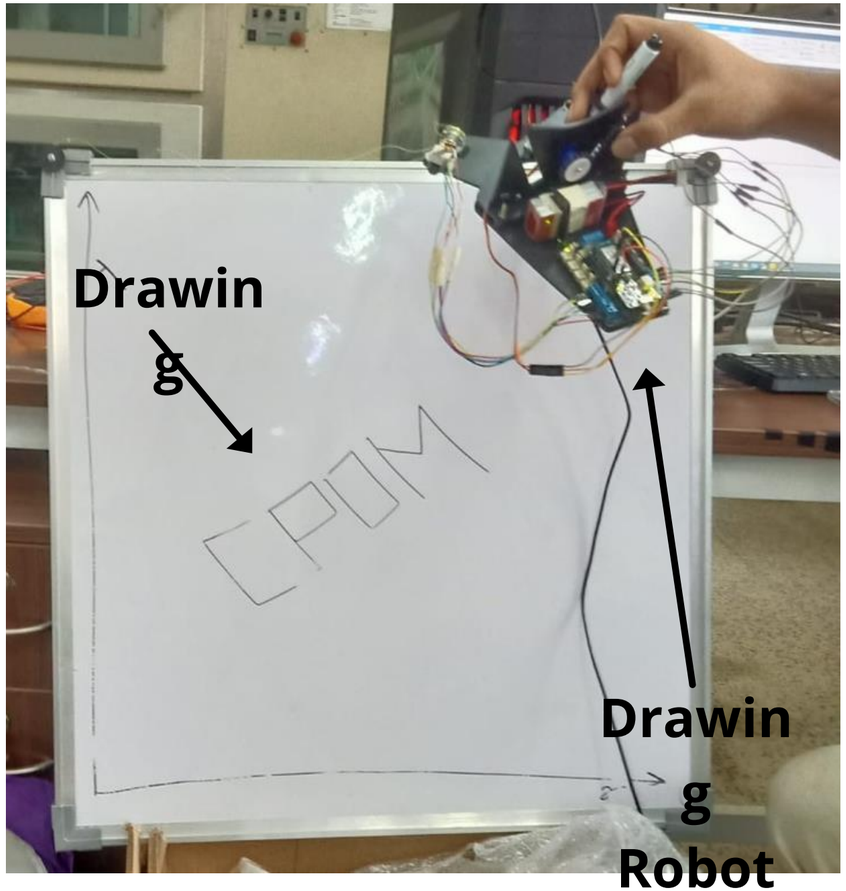


Fig 11 : Drawing of "CPDM"

Steps for drawing "CPDM"

1] Calculate travel time & motor direction* for each letters.

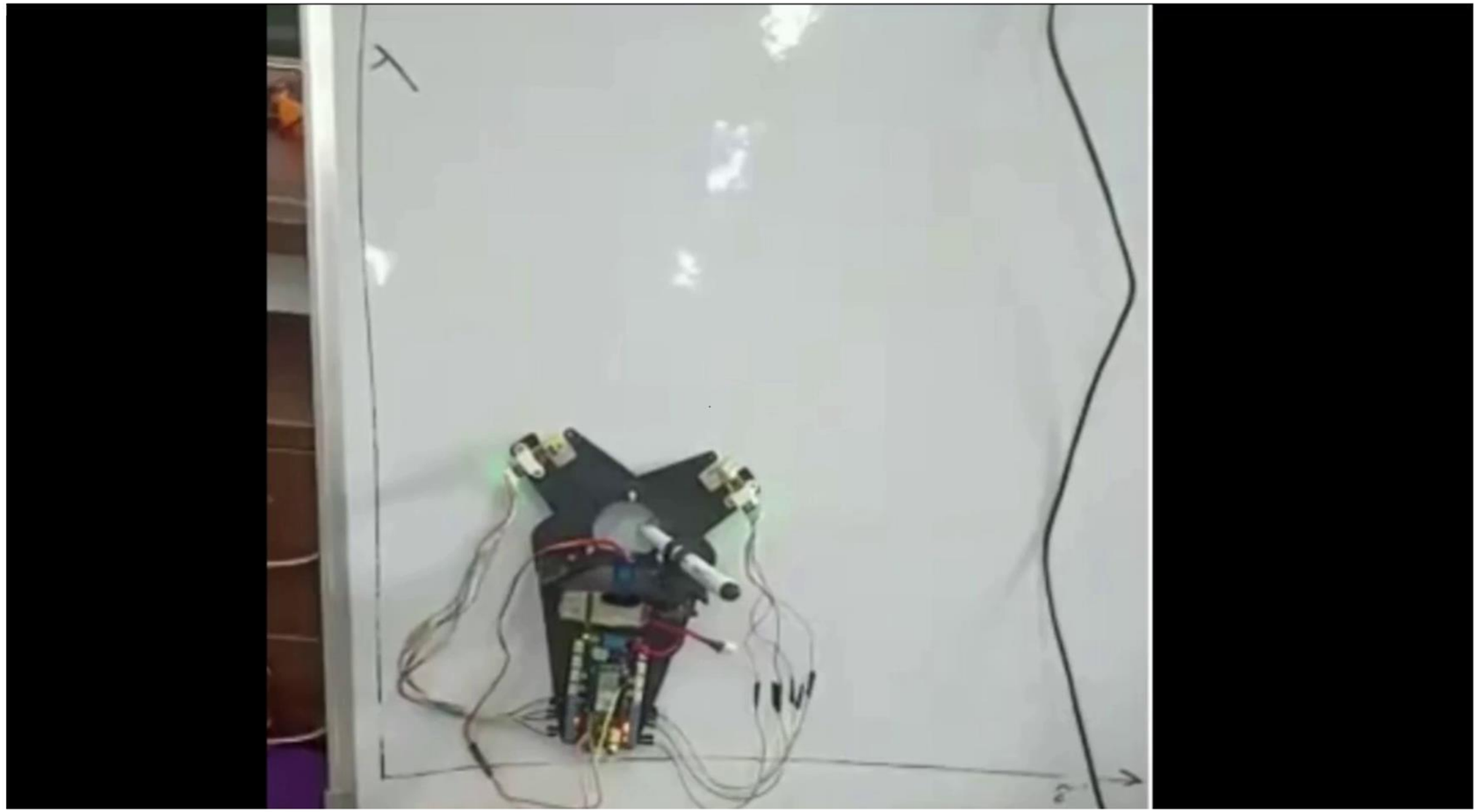
2] Adjust the variable (Servo's Duty cycle, delay) and convert commands to single scripts

3] Trial and error (For minor corrections)

4] Run the final script

For more info about motor direction refer Slide 36

OUTPUT



Video 2 : Demonstration Video of drawing "CPDM"

IDENTIFY POSITION LIMITS

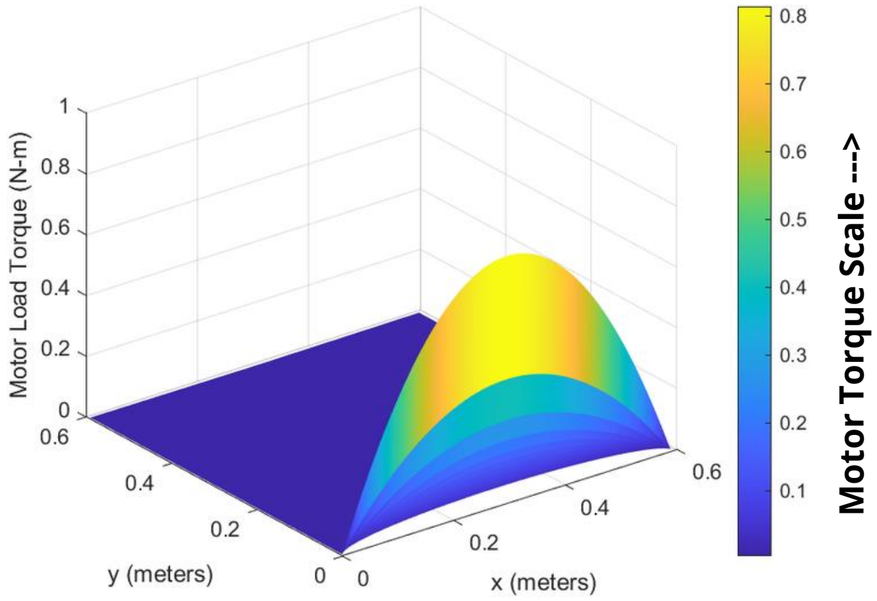


Fig 16 : Motor Load Torque vs xy coordinates

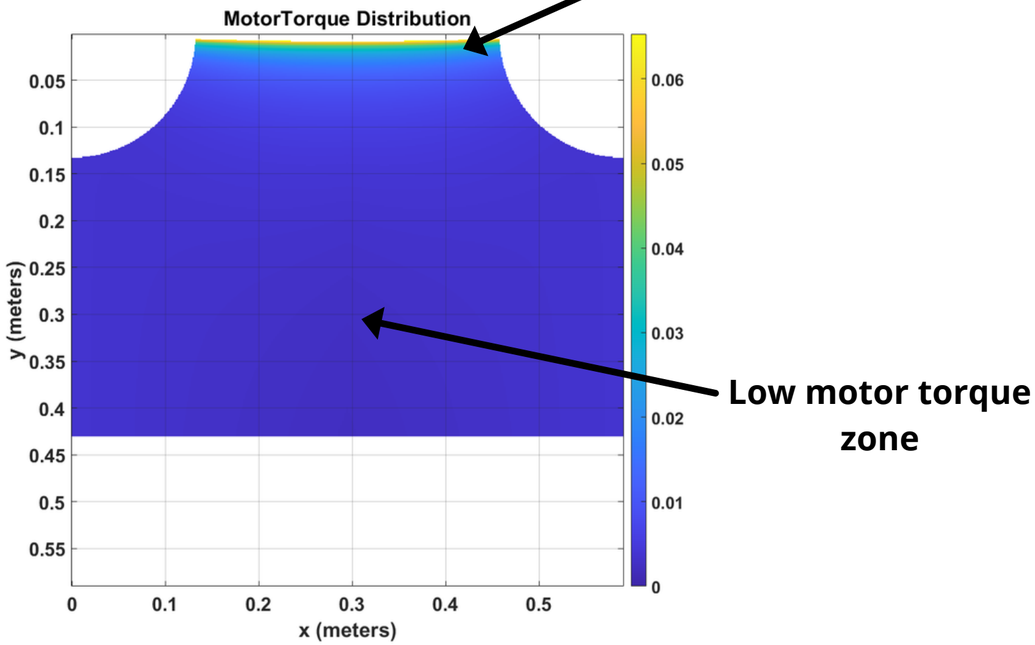


Fig 16 : Available WorkSpace with Torque Constraints

Here, blue region indicates comparatively low motor torque distribution thus favorable for drawing robot

Here, Motor specifications i.e. rated voltage, load-free RPM and Stall Torque are refereed : 6v, 150RPM and 2.4 Kg-cm respectively.

To calculate theoretical stall torque for our battery : *Tstall = Vbattery / R*

where Tstall - is motor stall torque and *R = Voltage rated / Stall torque of motor.*

OUTPUT - Positional Error

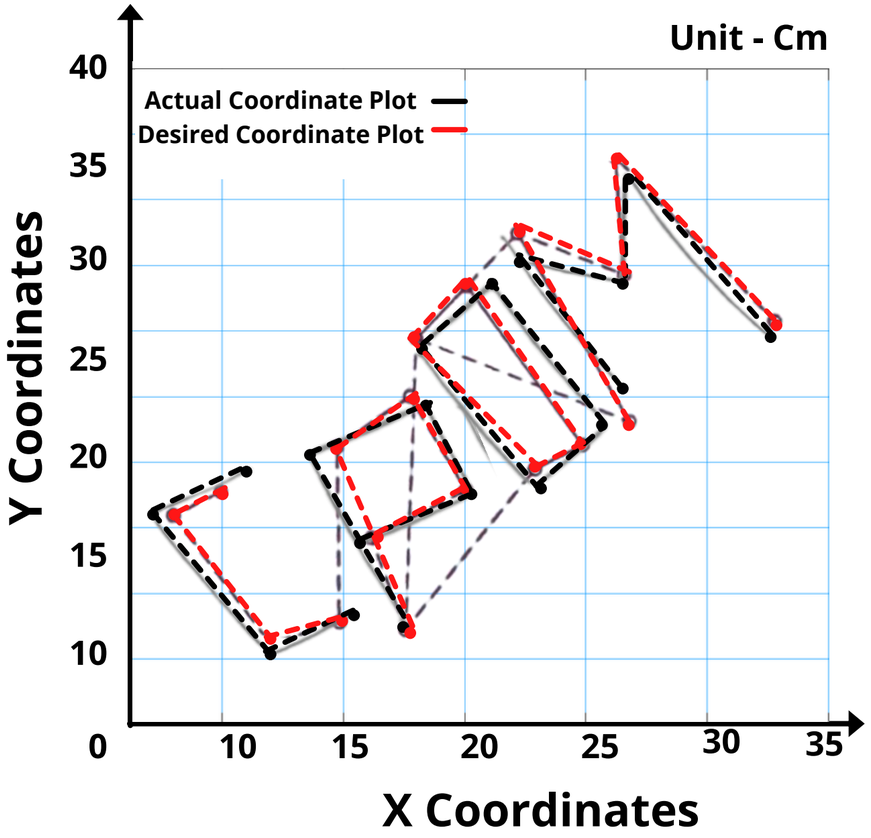


Fig 12 : Positional error graph plot of 'CPDM'

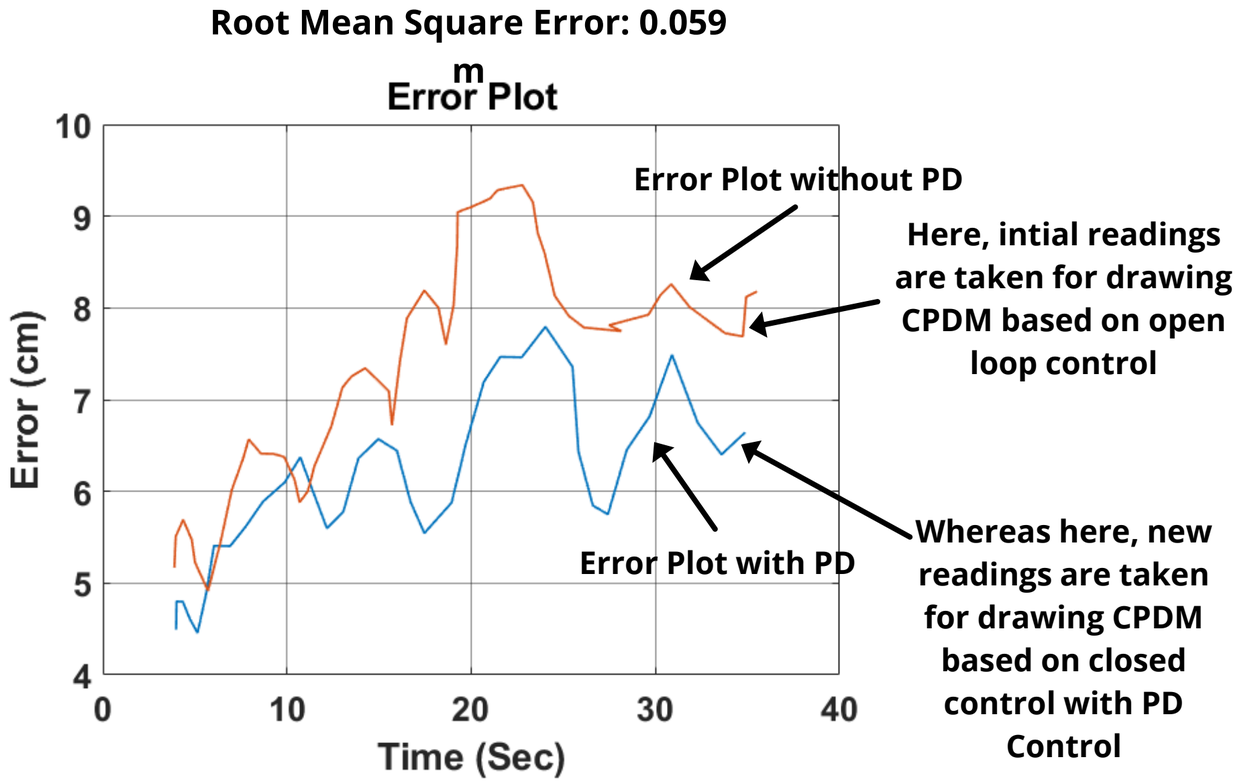


Fig A] Error Vs Time Plot

OUTPUT - Velocity vs Time

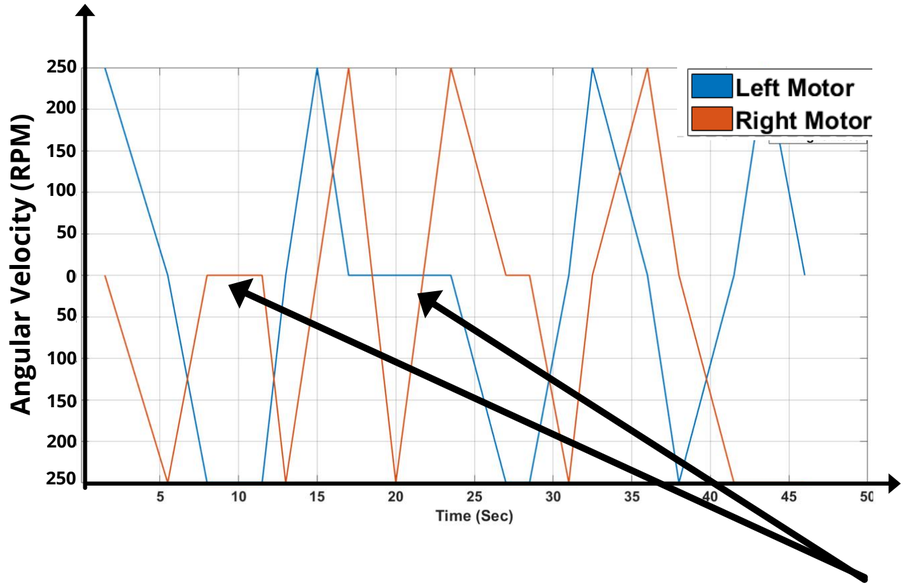


Fig 13 : Angular Velocity vs Time Plot

We can observe step change in agular velocity of both motors with some constant angular velocity in between

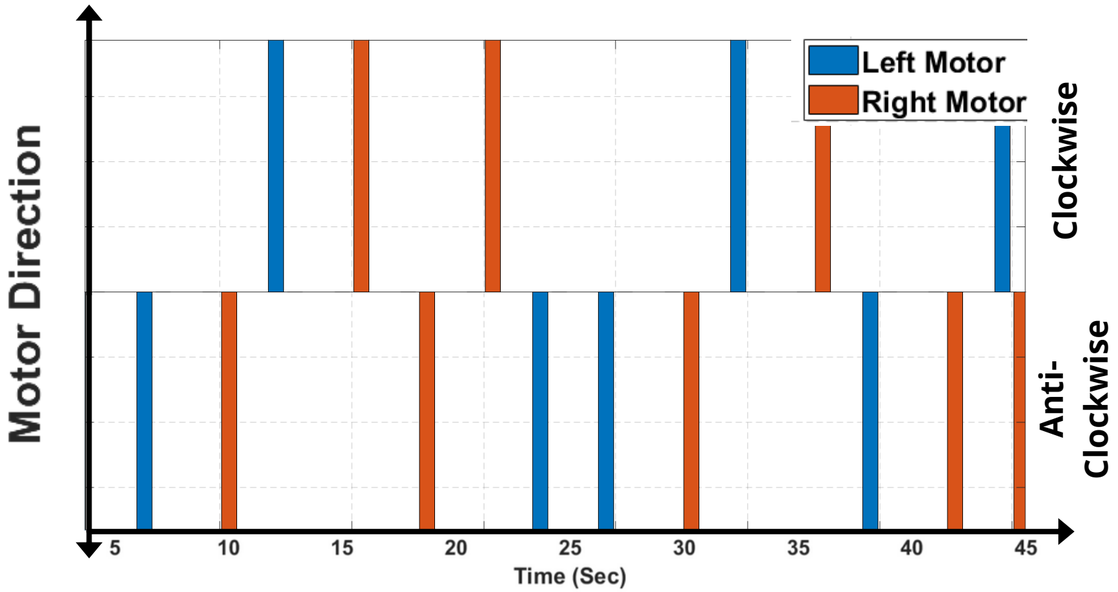


Fig 14 : Motor Direction vs Time Plot

UNDERSTANDING MOTOR TORQUE

Relation between motor load (torque)(T), supply voltage (V) and rotational speed (w)

T = k(V - w*k)/R

where ***k = motor constant (0.3820)***

R = resistance in motor windings (21.0542 ohms)

V = 3.7 V

Stall Torque = 0.2177 Nm

Tmax = 0.3*Stall = 0.0201 Nm {Should not exceed 30%}

OUTPUT - Motor Torque

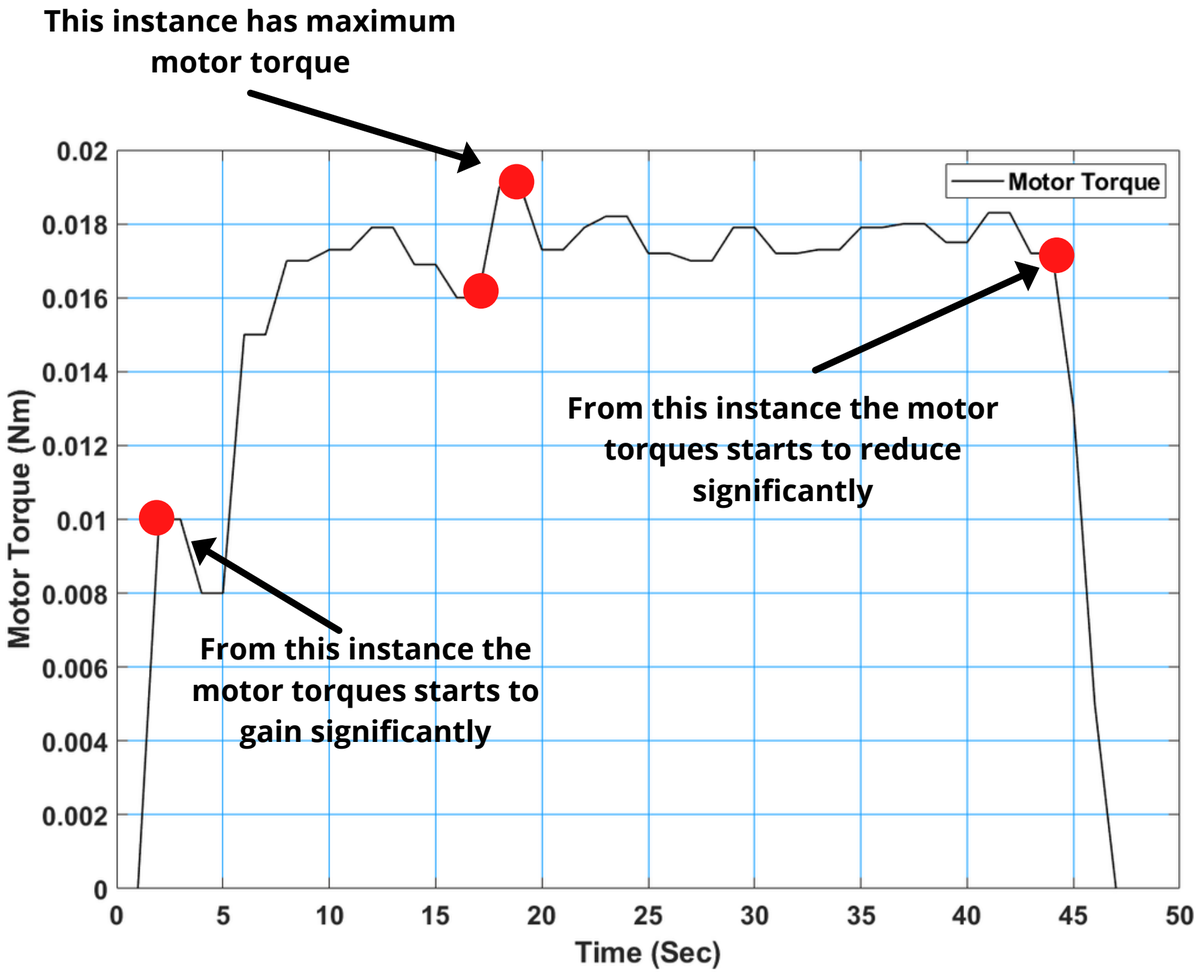


Fig 15: Motor Torque Vs Time plot

It can observe gradual fluctuation in motor torque as our robot normally works in vertical plane and motor torque depends on the position of robot at that instant. Hence we can observe some resemblance in position of the robot and motor torque .

FUNCTION FOR MOTION (COORDINATES SYSTEM)

moveToCoordinate **is a closed-loop control based custom function for moving drawing robot from its specified home position to the required position.**

The function can be divided into the following sections:

1] Current string length to initial coordinates

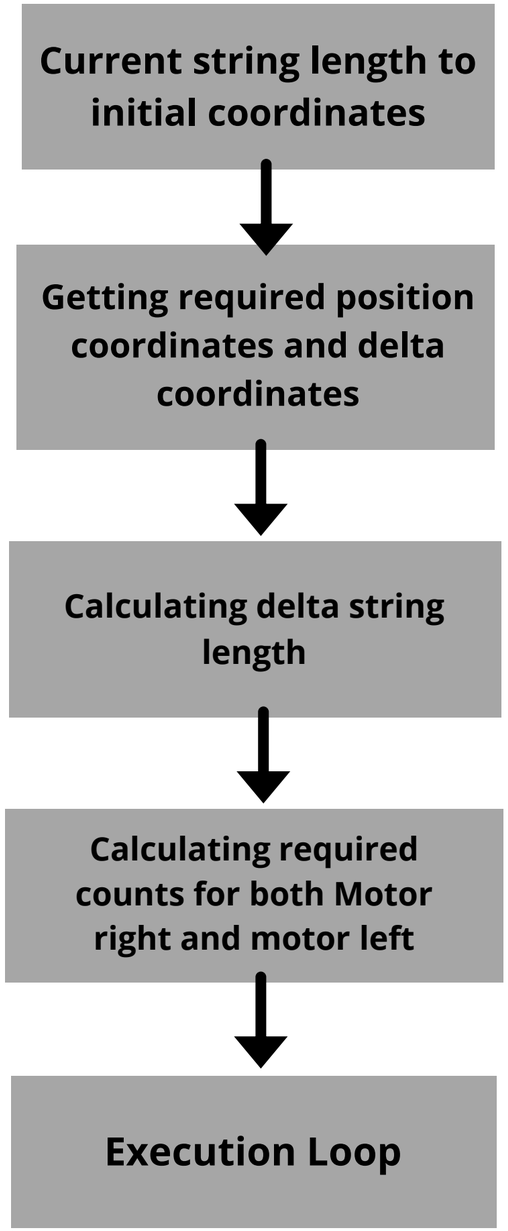
2] Getting required position coordinates and delta coordinates

3] Calculating delta string length

4] Calculating required counts for both Motor right and motor left

5] If loop for execution

FLOWCHART


Current string length to initial coordinates
Getting required position coordinates and delta coordinates
Calculating delta string length
Calculating required counts for both Motor right and motor left
Execution Loop

MOVE FUNCTION IMPLEMENTATION

Coordinates

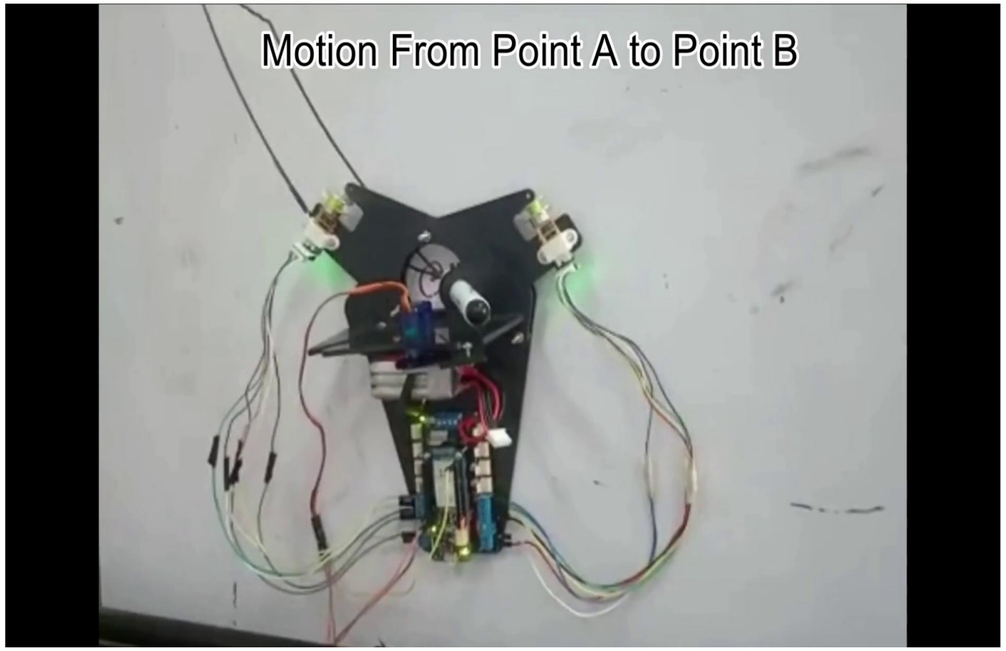
A

(0.2950,0.3954)

B

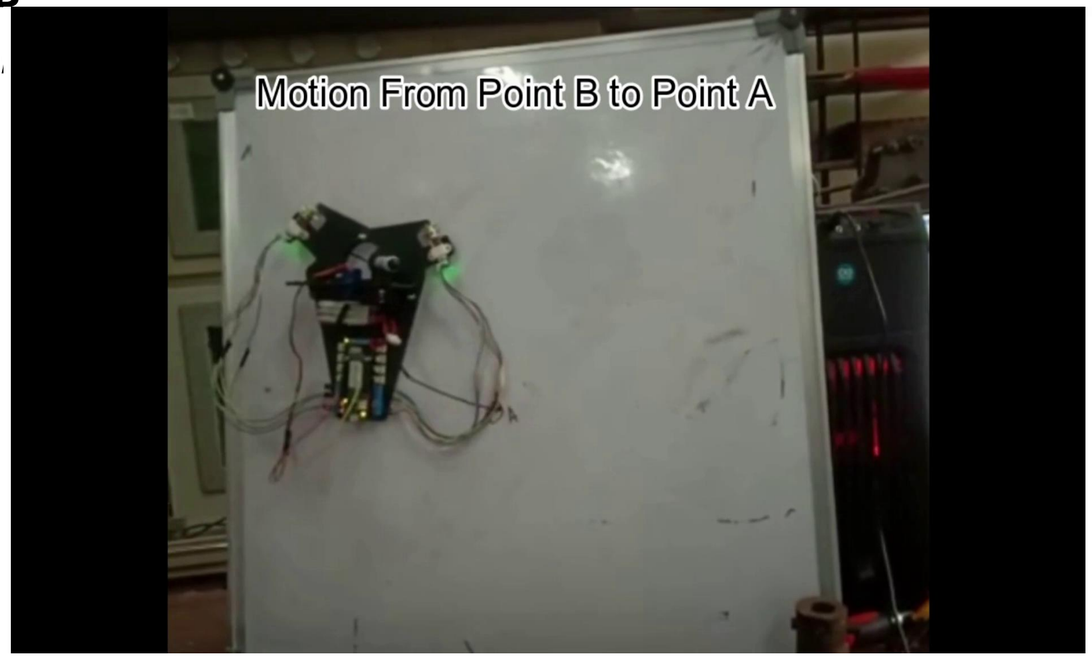
(0.1094,

A to B



Video 4 : Preview - Motion A to B

B to A



Video 5 : Preview - Motion B to A

CREATING GRID OF ALL POSSIBLE BOARD POSITION

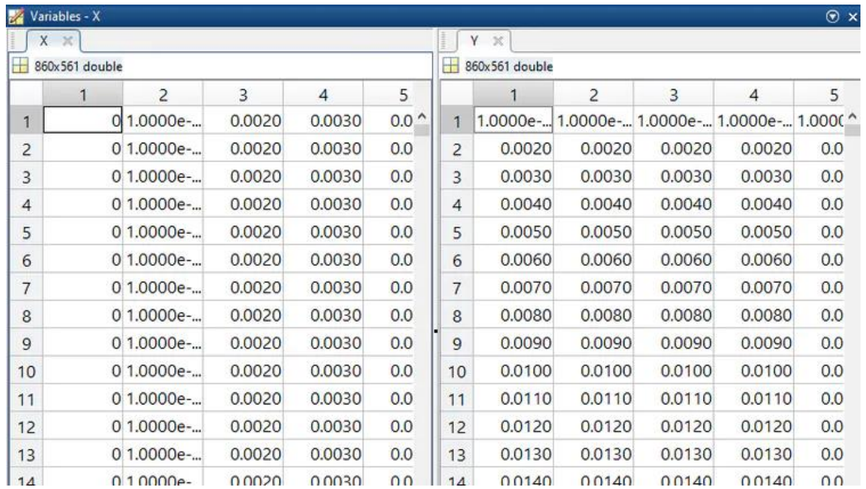


Fig 18 Position Grid

Removing / Editing Dimension Limits

Value of xLim and yLim can be edited as shown in figure 25

xLim = x axial limit
yLim = y axial limits

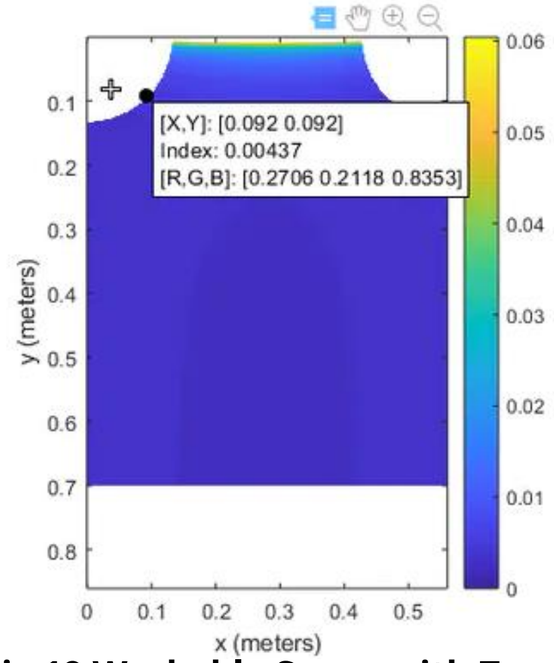


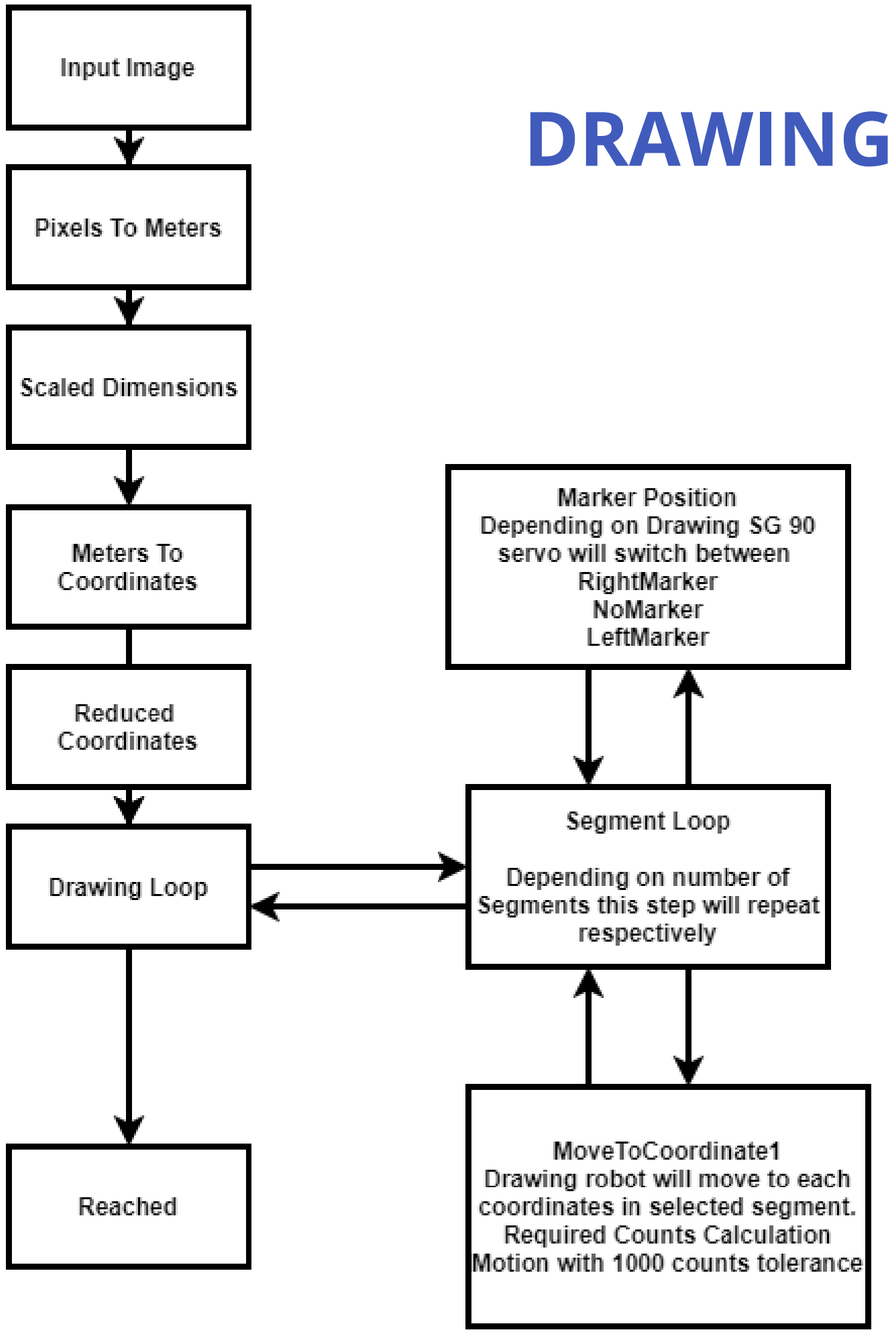
Fig 19 Workable Space with Torque Constraints

In Fig 23, For ensuring safety of DC motors (N20) its recommended not exceed 30% of Battery Stall torque

All possible board position are calculated employing torque limit and whiteboard dimensions

EXERCISE 3
Algorithm / Flowchart for Image Processing & Drawing

DRAWING ROBOT'S FLOWCHART



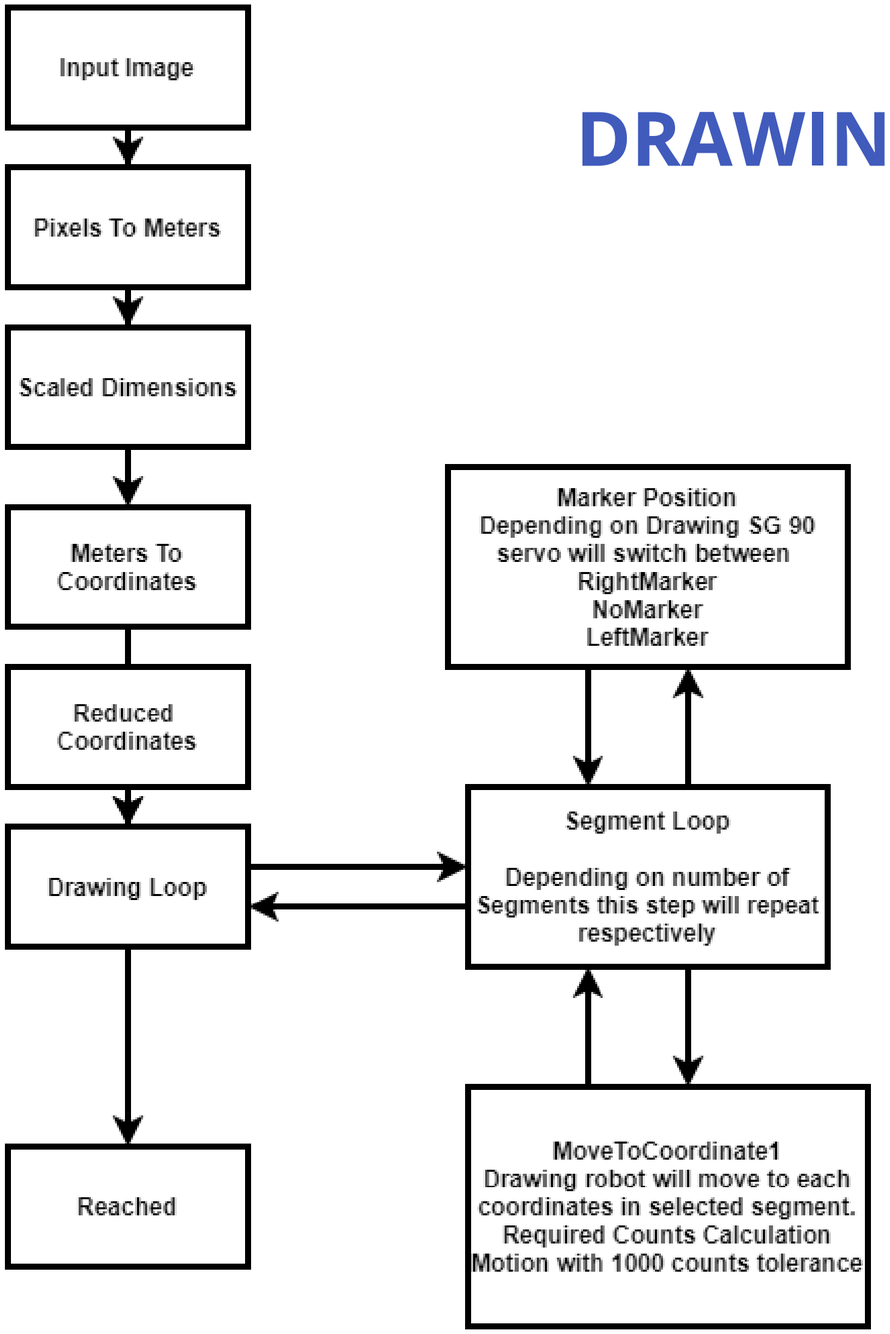
Step 1] Image (Pixels) ------> RGB Data

Step 2] The pixels are converted to metric form (meters)

Step 3] These dimensions are scaled in accordance to the whiteboard workable space restrictions.

Step 4] Scaled metric data ------> Coordinates.

DRAWING ROBOT'S FLOWCHART



Step 5] Coordinates reduction.

Step 6] Drawing loop process the reduced coordinates with segment loop, Marker position and MoveToCoordinate1.

Step 6.1] Commands to Marker position and moveToCoordinate1 block.

Step 7] Completion of drawing loop & coordinates ---> Task Completed

EXERCISE 4
Drawing Preprocessed Image (Closed Loop)

DRAWING PRE-PROCESSED IMAGES

Reading Images

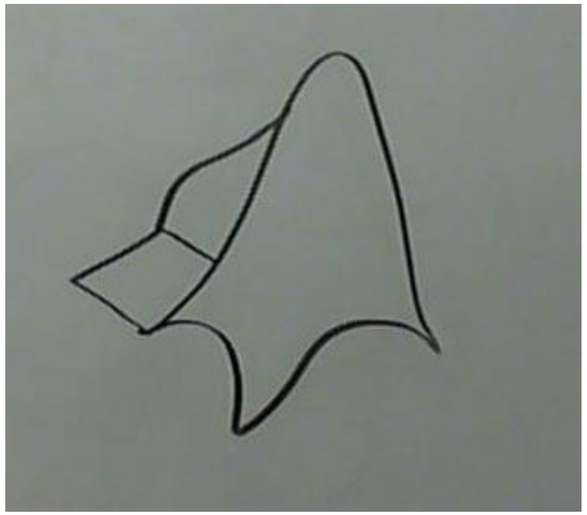


Fig 20 Original Image - MATLAB Logo

Each one of the components (R, G, B) representing one of the basic colors gets 8 bits to store its value in digital form.

Load and Plot Coordinates for an Existing Image

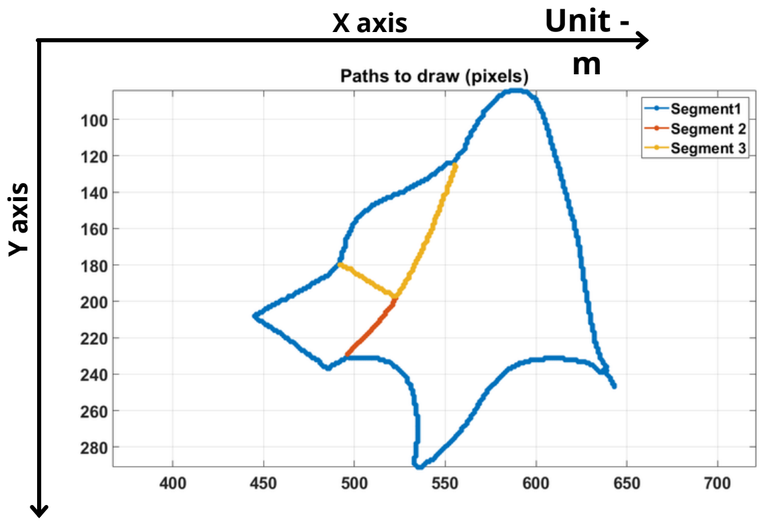


Fig 21 Coordinates Plotting

The MATALAB pre-processed coordinates and pixels values are converted in term of distance(Meters) with different segments

Scaling To Physical Units

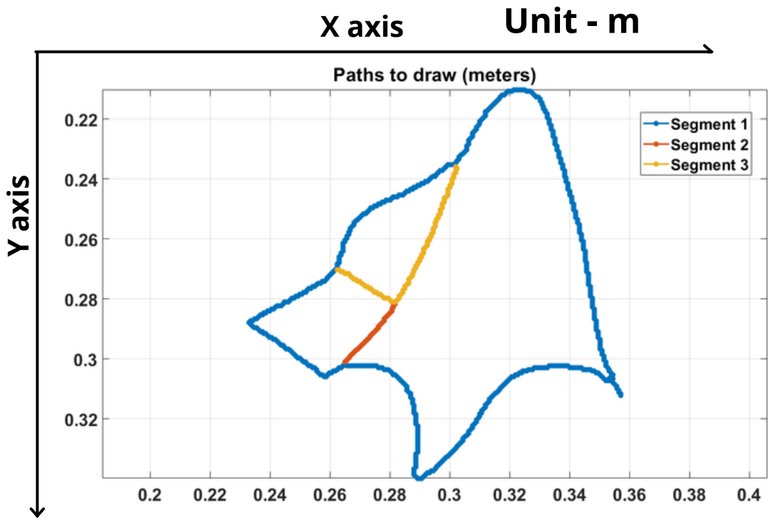


Fig 22 Scaled Coordinates

This converted data is scaled down with respective to the dimenstionality constraints.

DRAWING PRE-PROCESSED IMAGES

Reduce Segment

The reduceSegment function reduces the number of points in a segment by removing points within a specified radius. In other words, it filters out points, what reduces the amount of subsegments to draw.

function drawImageFromPix Steps :

1] Define whiteboard limits

2] Convert pixel coordinates to physical distances and then to encoder counts

3] Reduce size of each segment

4] Loop for processing each coordinates

Index

- **Function transformPixelsToMeters converts pixels coordinates to physical distances.**
- **function reduceSegment for reducing the coordinates density**
- **Function xyToCounts to convert coordinates to counts.**

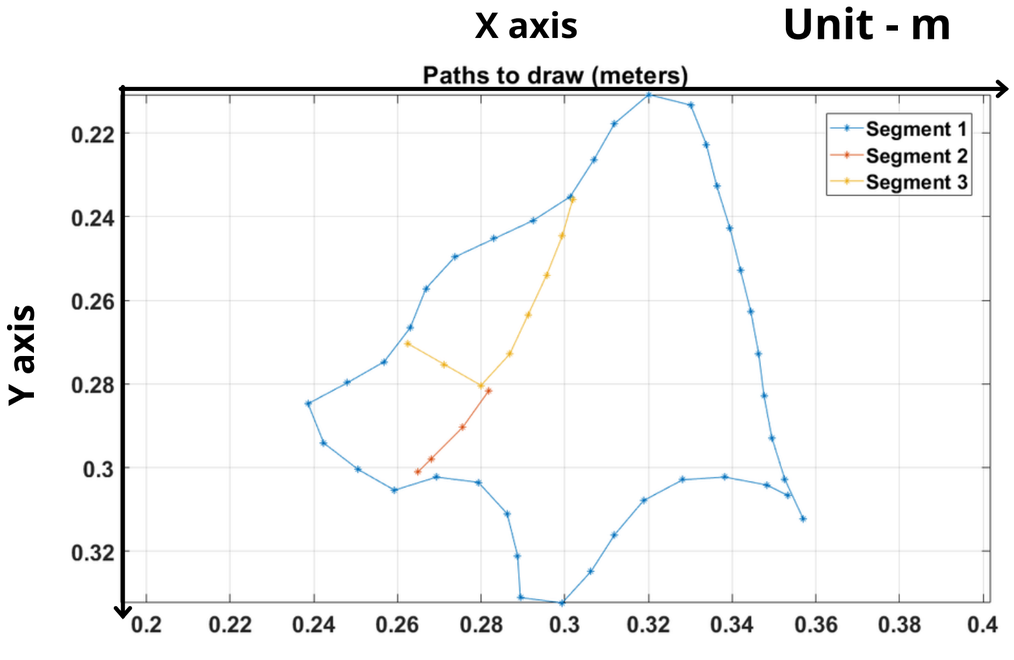
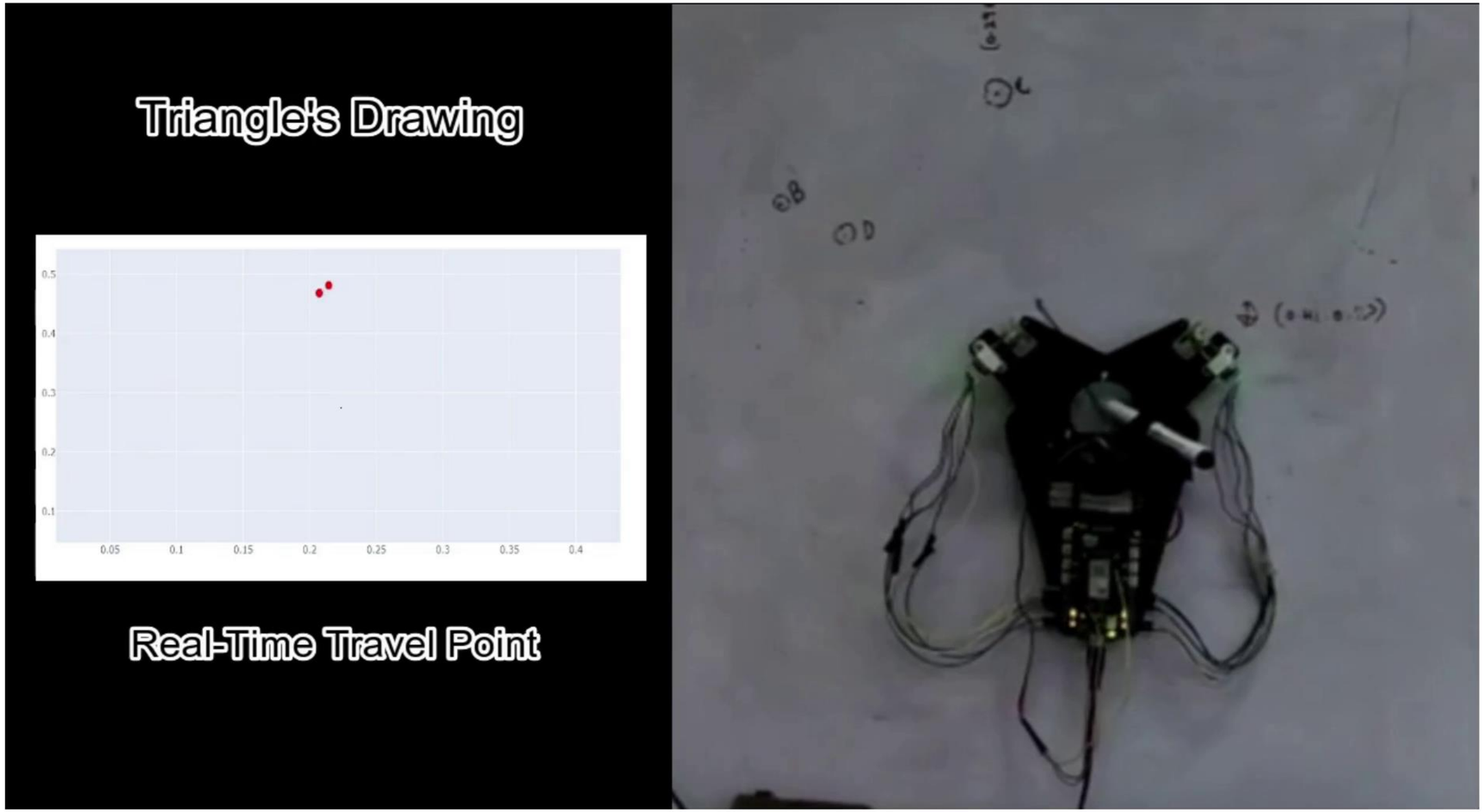


Fig 23 Scaled Coordinates - Reduced

DRAWING PRE-PROCESSED IMAGES



Video 6 Implementation - MATLAB Logo

OUTPUT - Positional Error

Implementation - MATLAB Logo

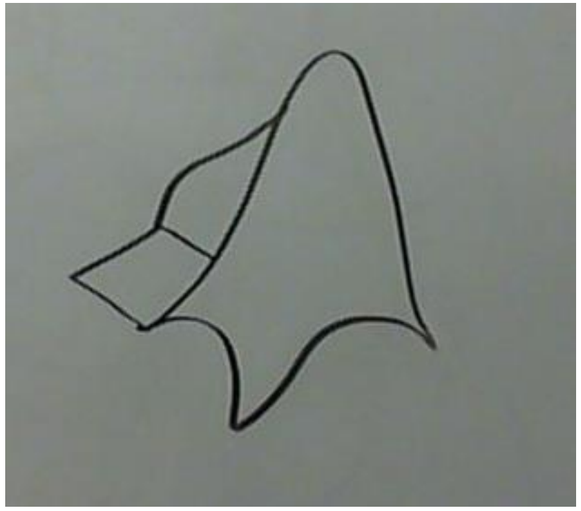


Fig 24 Original Image

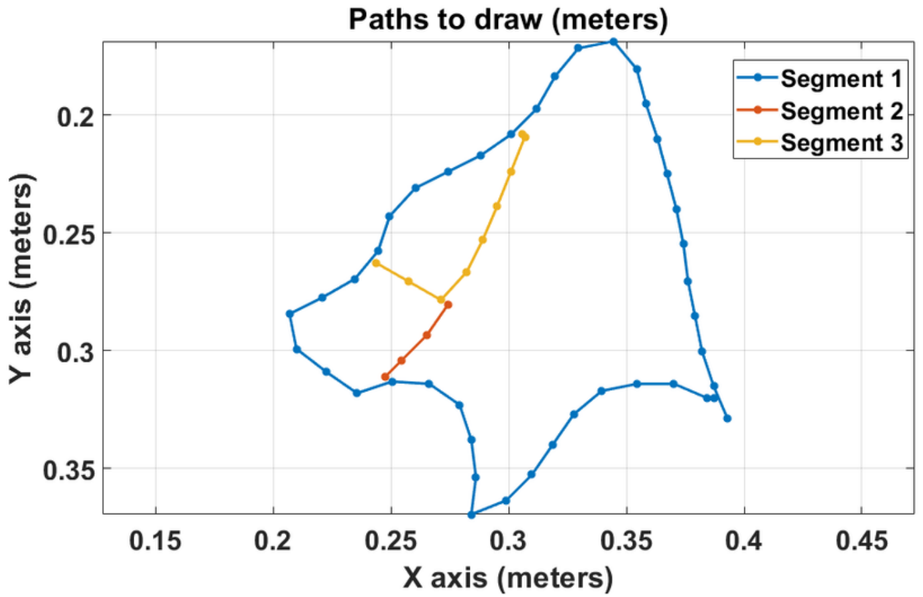


Fig 25 Coordinates Plot

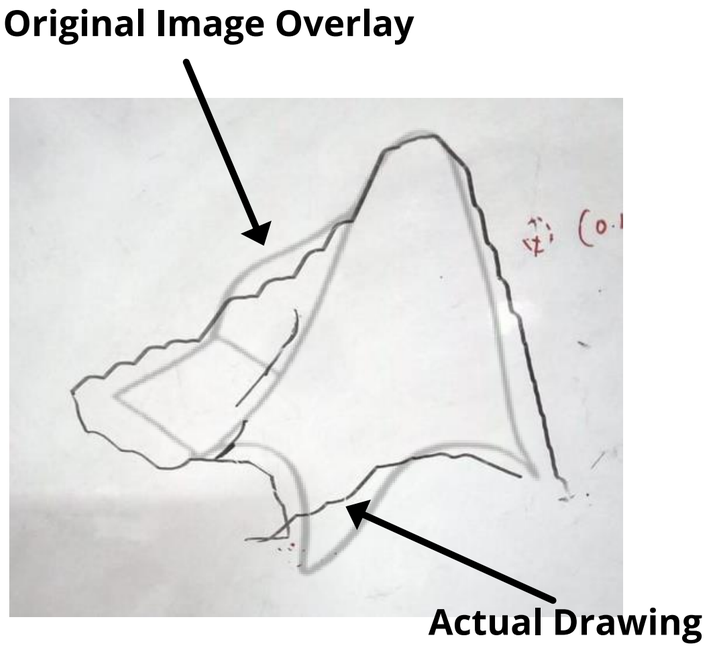
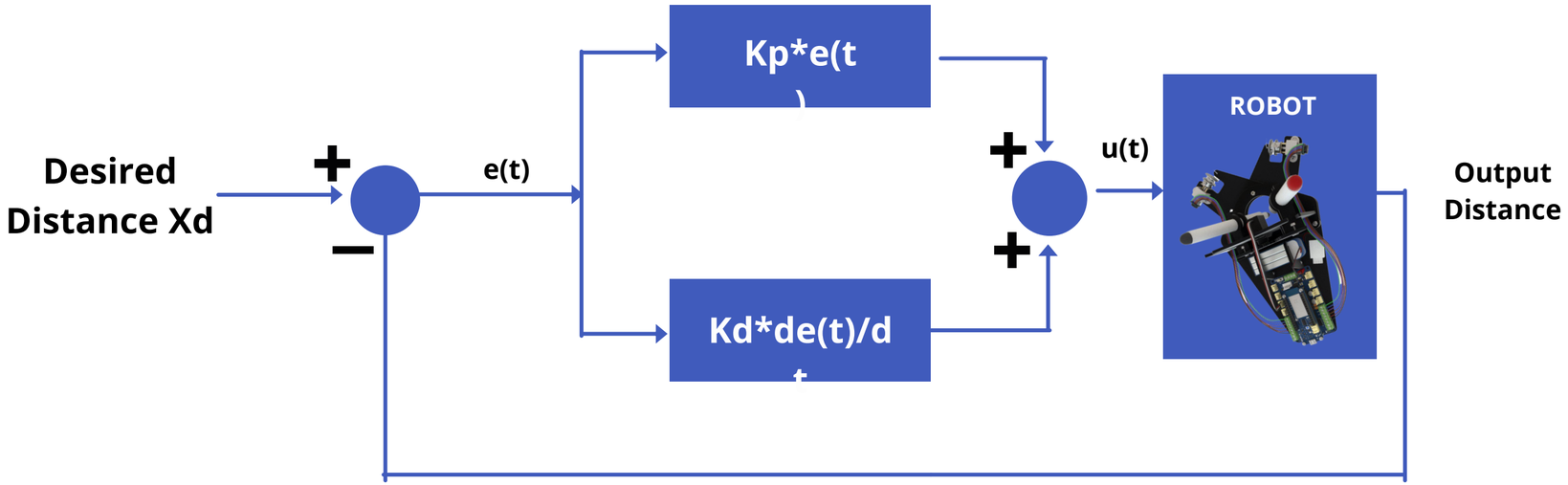


Fig 26 Output Drawing

IMPLEMENTATION OF PD CONTROLLER

PD Controller Implementation



Feedback Signal
(Encoder's Counts)

PD controller Equation:

$u(t) = Kp*e(t) + Kd*de(t)/dt$

Distance travelled = Number of Counts/49300

Here, u(t) = controller output

e(t) = error value

Kp = Proportional Gain (150)

Kd = Derivative Gain (10)

de = change in error value

dt = change in time

OUTPUT - Positional Error

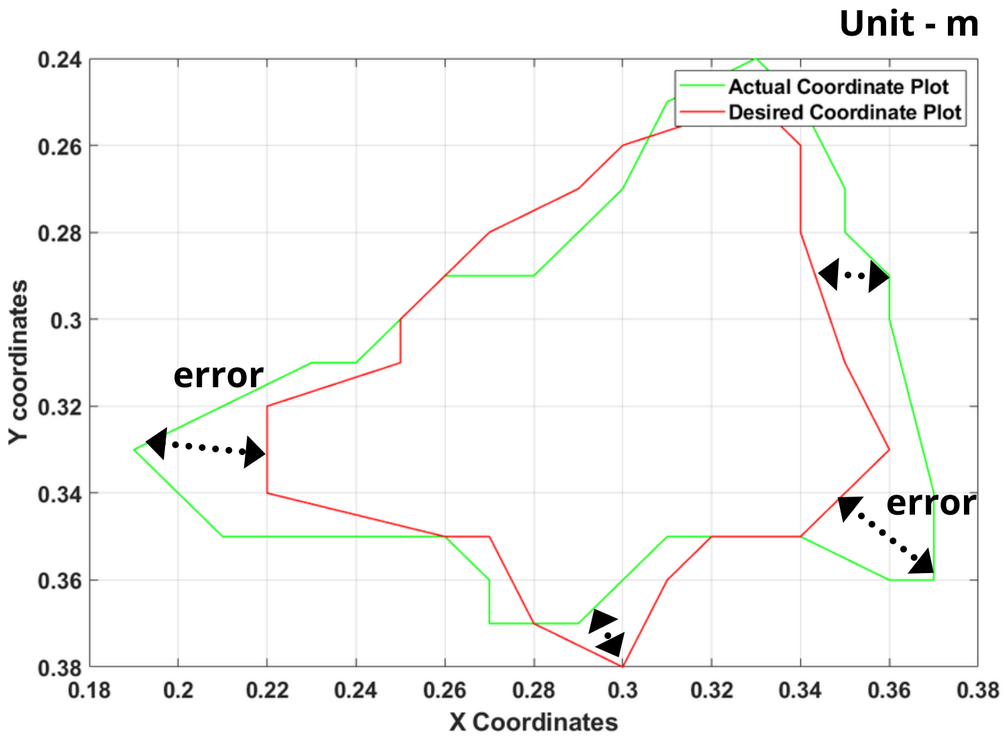


Fig 27 Actual vs Desired Coordinate Plot

Here, Drawing starts with error of more than 12 cm and gradually it decreases and this continues to fluctuates in between

Root Mean Square Error: 6.25 cm

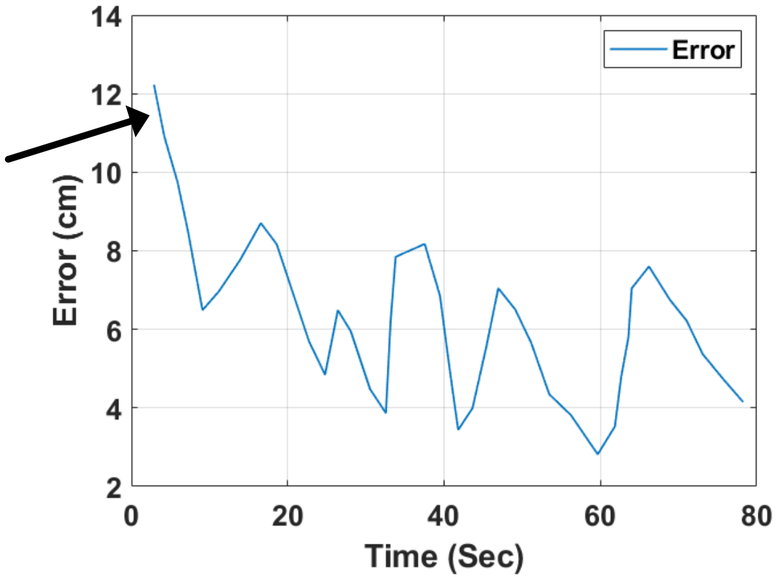
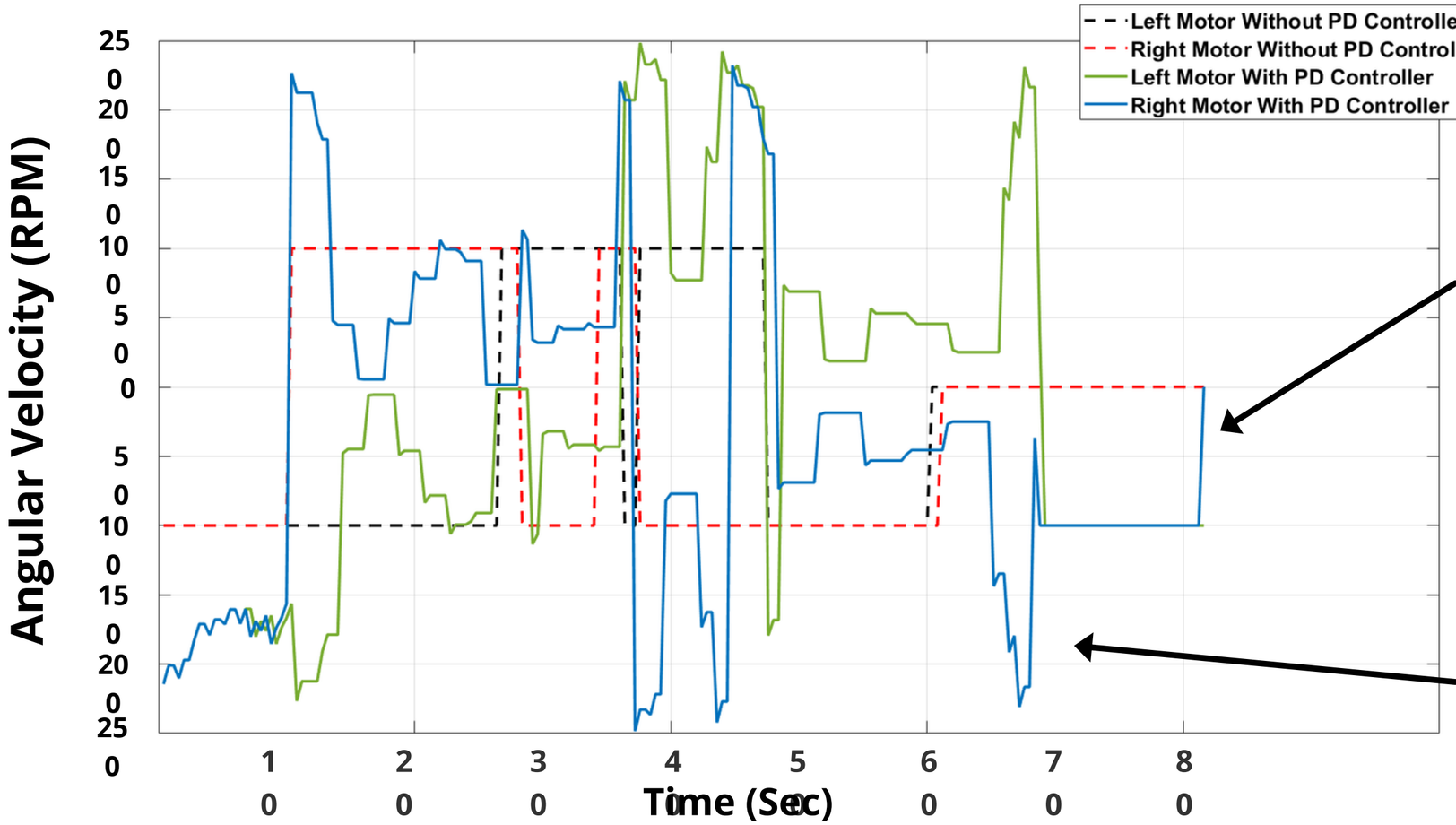


Fig B] Error Vs Time Plot

OUTPUT - Velocity vs Time

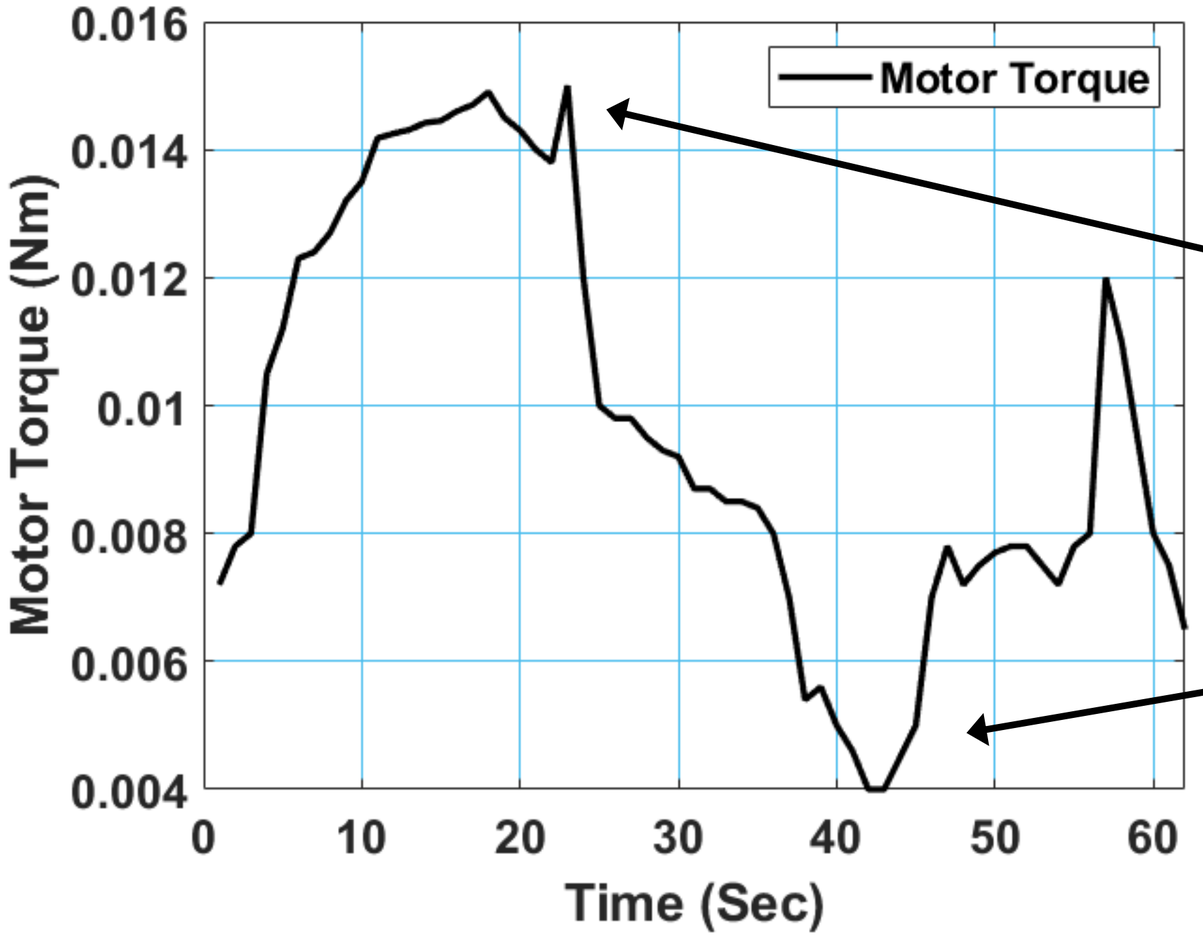


Here, reading for angular velocity without PD controller hits the threshold causing excessive energy loss.

Here, reading for angular velocity with PD controller doesn't hits the threshold and acquire required angular velocity

Fig 28: Angular Velocity with and without PD controller on Left Motor and right motor respectively.

OUTPUT - Motor Torque

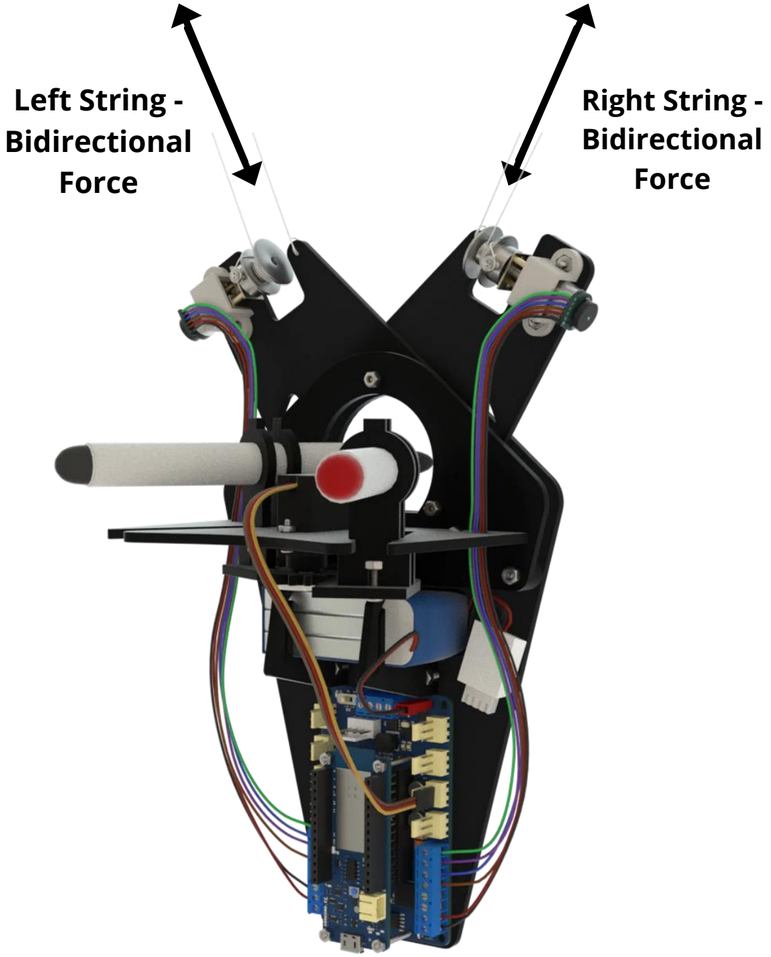


Here, during drawing robot reaches the point with highest motor torque value

Here, A dip in motor torque can be observed as this region drawing occurs at lower most region of drawing board

Fig 29: Motor Torque Vs Time

RELATION BETWEEN COUNTS AND DRAWING ROBOT'S MOTION DIRECTION



Direction - Counts - Force Vector Relation

Counts Left - positive
Motor Direction - Anticlockwise (Negative)

Counts Right - Negative
Motor Direction - AntiCockwise (Negative)

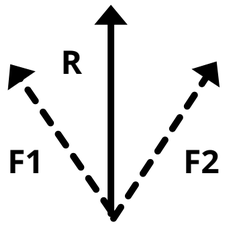
Counts Left - Negative
Motor Direction - Clockwise (Positive)

Counts Right - Positive
Motor Direction - Clockwise (Positive)

RELATION BETWEEN COUNTS AND DRAWING ROBOT'S MOTION DIRECTION

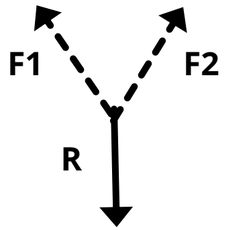
In drawing robot the F1 (Motor Left) and F2 (Motor Right) depends on the motor's input duty cycle and 'R' is resultant vector

Case 1 : Up



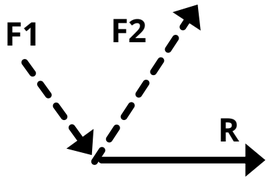
mL.Speed = mR.Speed = Negative

Case 2 : Down



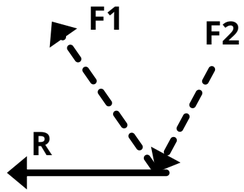
mL.Speed = mR.Speed = Positive

Case 3 : Right



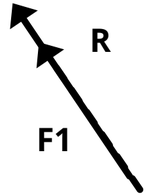
mL.Speed < mR.Speed
mL.Speed = Positive
mL.Speed = Negative

Case 4 : Left



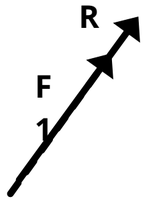
mL.Speed > mR.Speed
mL.Speed = Negative
mL.Speed = Positive

Case 5 : Top-Left



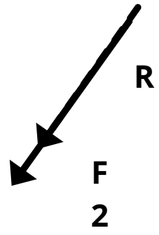
mL.Speed = Negative

Case 6 : Top-Right



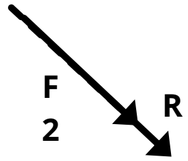
mR.Speed = Negative

Case 7 : Bottom-Left



mL.Speed = Negative

Case 8 : Bottom-Right



mL.Speed = Negative

EXERCISE 5
Drawing Live Image (Rectangle & Triangle)

PREPROCESSING

Plot Function - Coordinate Set Plotting, Pixels To Distance & Scaling

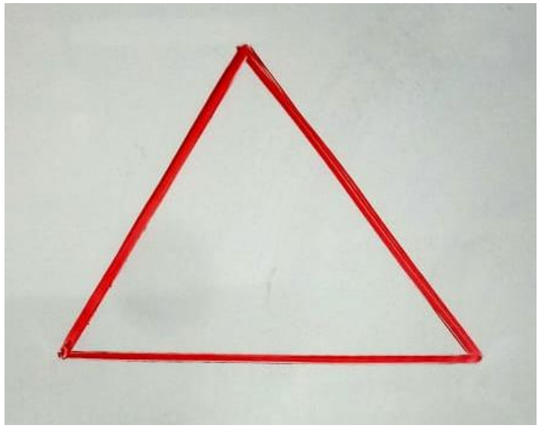


Fig 32 : Original Image

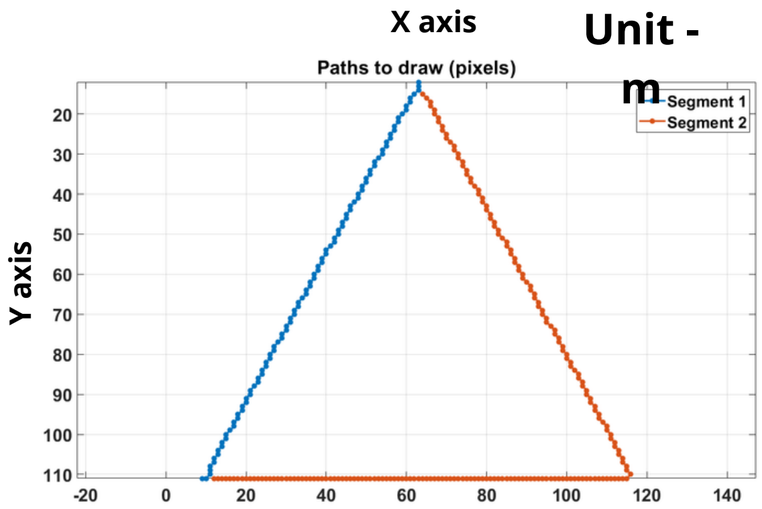


Fig 33 : PixelsToMeter Output

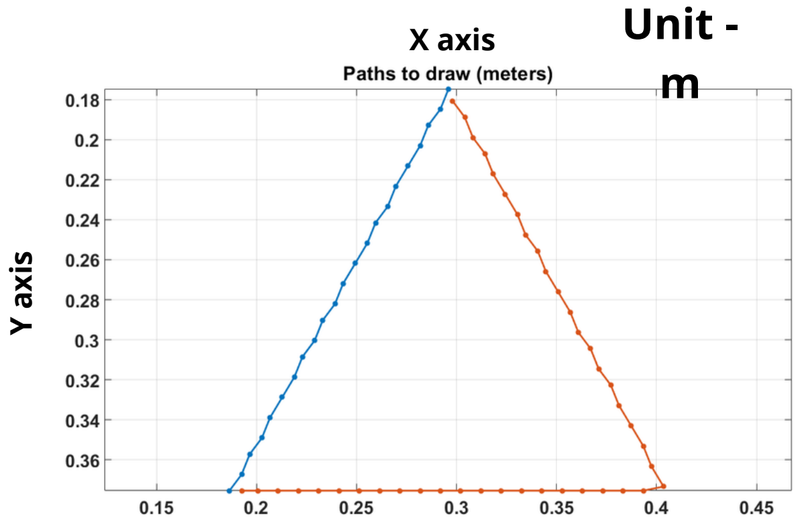


Fig 34 : Scaled Output

Process the image so that it contains only the thin traces of lines. Use image processing functions to convert the image to grayscale, convert it to binary black and white, remove isolated pixels, and thin objects to lines.

Gray scale coverstion : img2 = rgb2gray(img);
Morphologiocal operations : img4 = isomorph((img),'clean');

img = Input Image

img2, img4 = Output Image

PREPROCESSING

Coordinate Reduction and Implementation

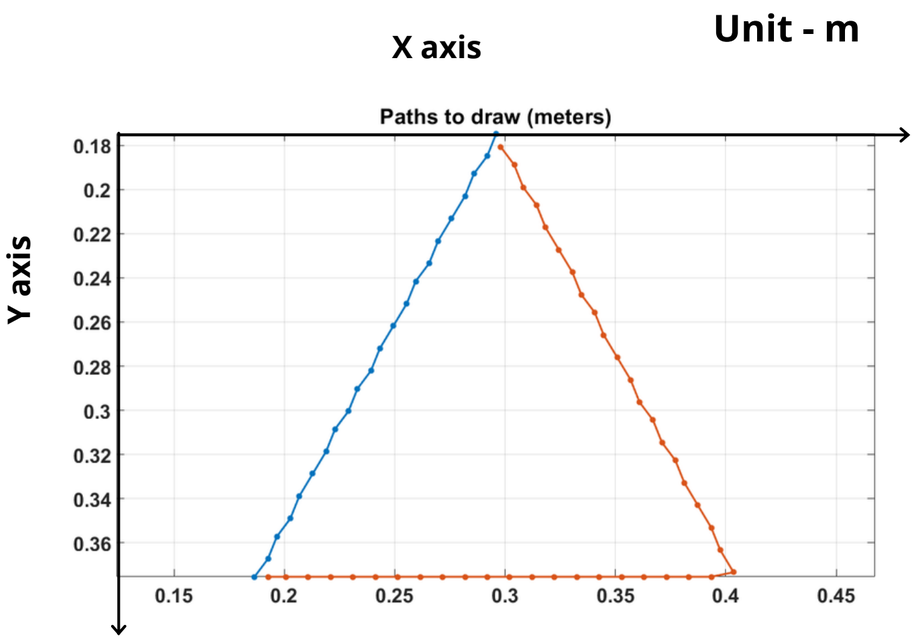
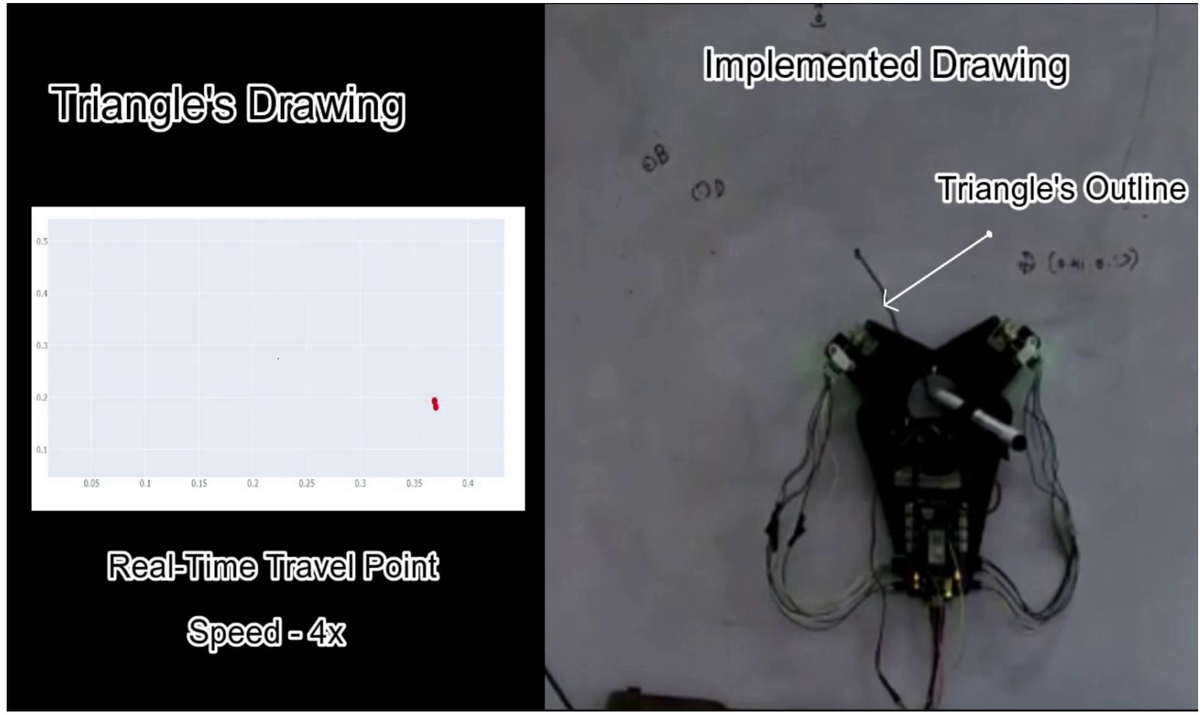


Fig 35 : Reduced Output Image with resultant coordinate Output.



Video 7: Implementation of triangle

In drawing process first the image is divided into distinct segments represented with different colors. More the segments higher is the complexity in drawing, thus affecting drawing quality of resemblence as shown in Fig 35

PD CONTROLLER IMPLEMENTATION

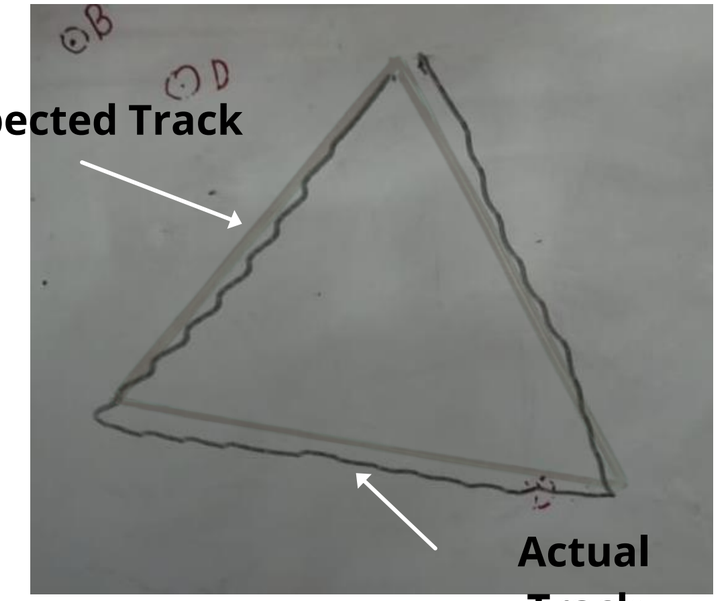


Fig 36: Drawing Without PD controller implementation and min coordinate distance = 0.02m

Where Kp = 150 , Kd = 10

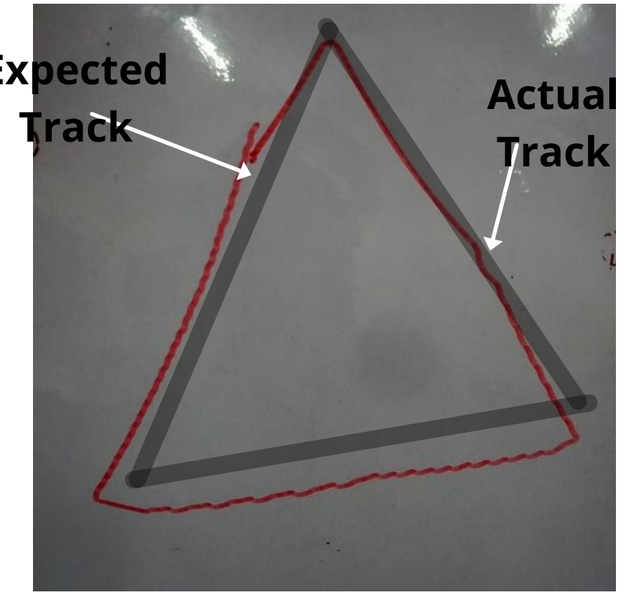


Fig 37 : Drawing With PD controller implementation and min coordinate distance = 0.008m

The PD controller regulates the value of motor constant (motor constant $\propto$ motor's angular velocity) where encoder's counts are processed for angular velocity at that instant and error (difference in desired and actual)

OUTPUT - Positional Error

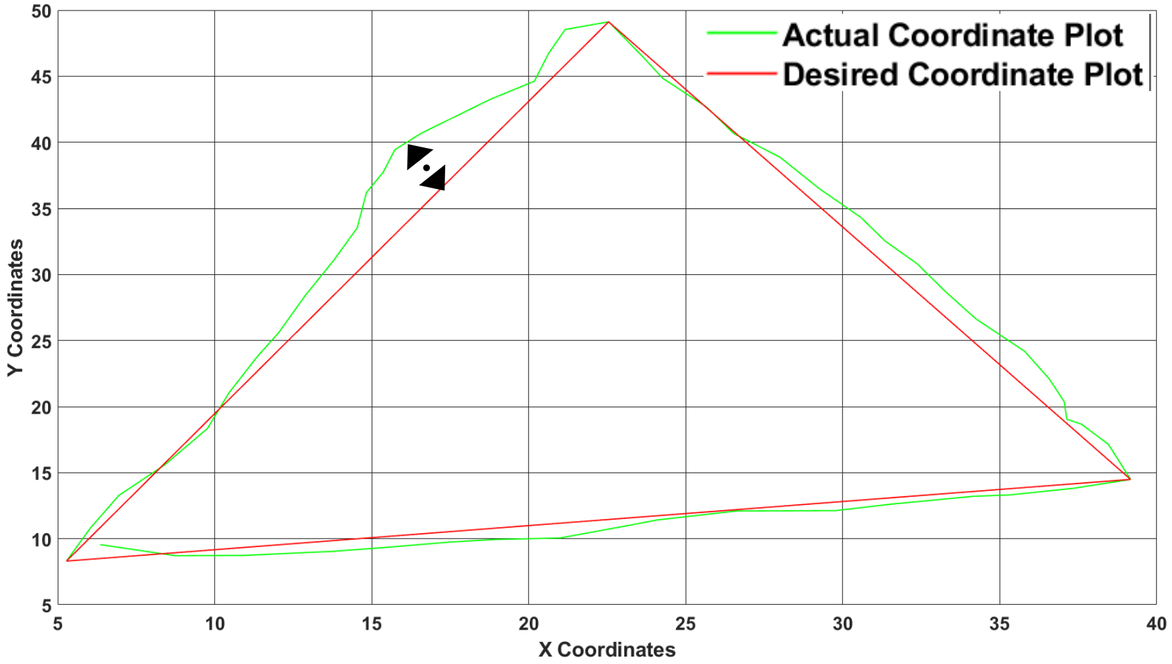


Fig 38: Expected Track vs Actual Track

Root Mean Square Error: 4.35 cm

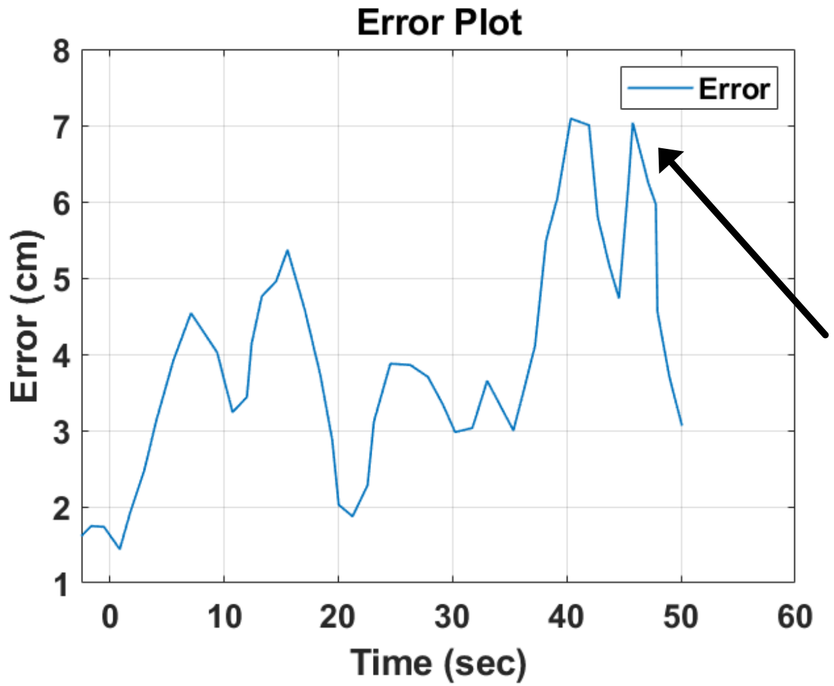
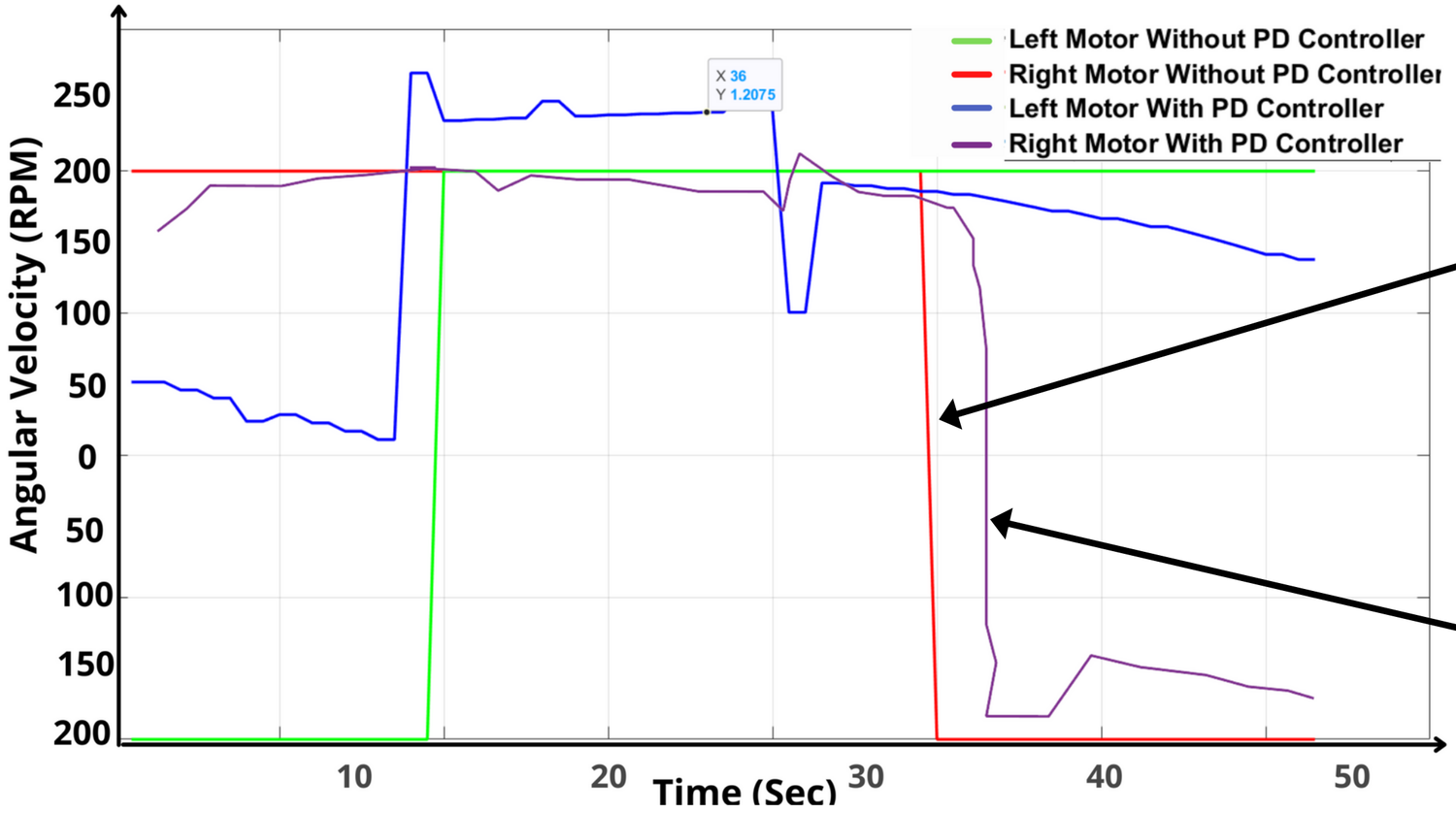


Fig C] Error Vs Time Plot

Here, Drawing ends with error of more than 6cm and this continues to fluctuates in between

OUTPUT - Velocity vs Time



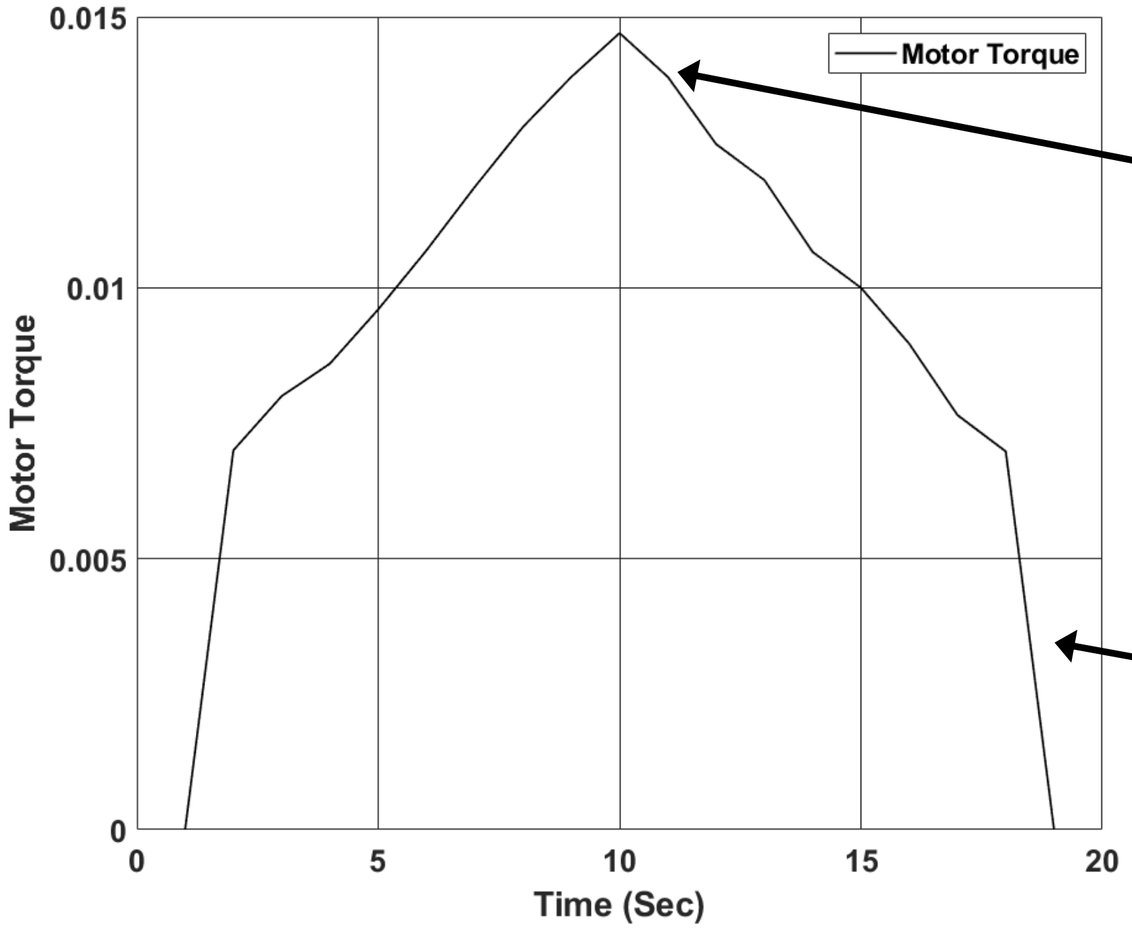
Here, reading for angular velocity without PD controller hits the threshold causing excessive energy loss.

Here, reading for angular velocity with PD controller doesn't hits the threshold and acquire required angular velocity

Fig 41 : Angular Velocity Graph plot For Right Motor, Left Motor, Left Motor with Controller

The PD controller regulates the value of motor constant (motor constant $\propto$ motor's angular velocity) where encoder's counts are processed for angular velocity at that instant and error (difference in desired and actual)

OUTPUT - Motor Torque Error



Here, during drawing robot reaches the point with highest motor torque value

Here, A dip in motor torque can be observed as this region drawing occurs at lower most region of drawing board

Fig 42: Motor Torque Vs Time

Preprocessing

Pixels To Distance; Scaling

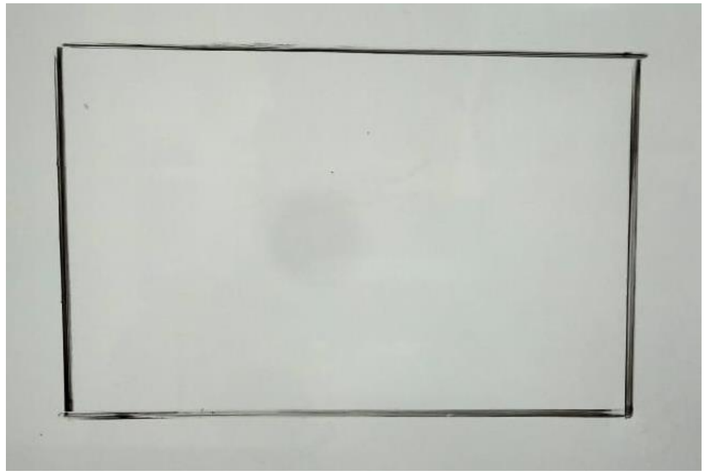


Fig 43 : Original Image

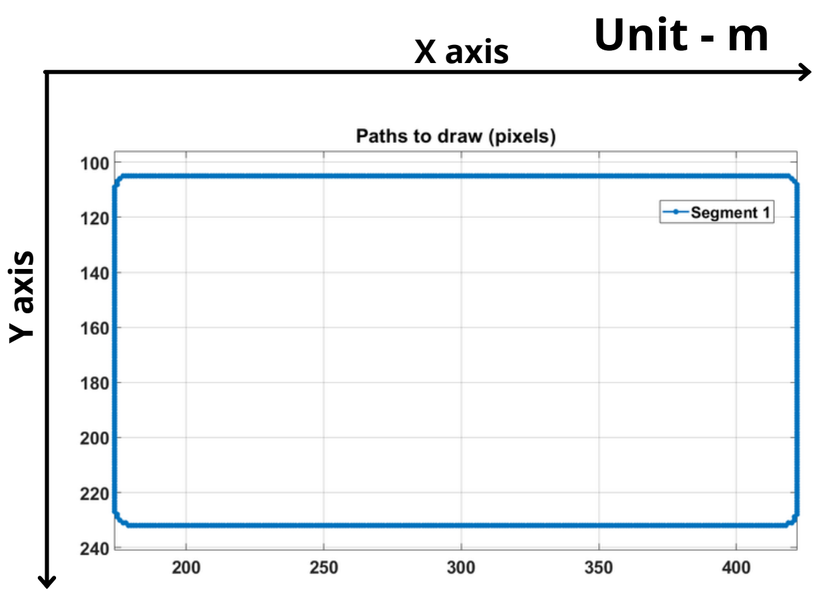


Fig 44 : PixelsToMeter Output

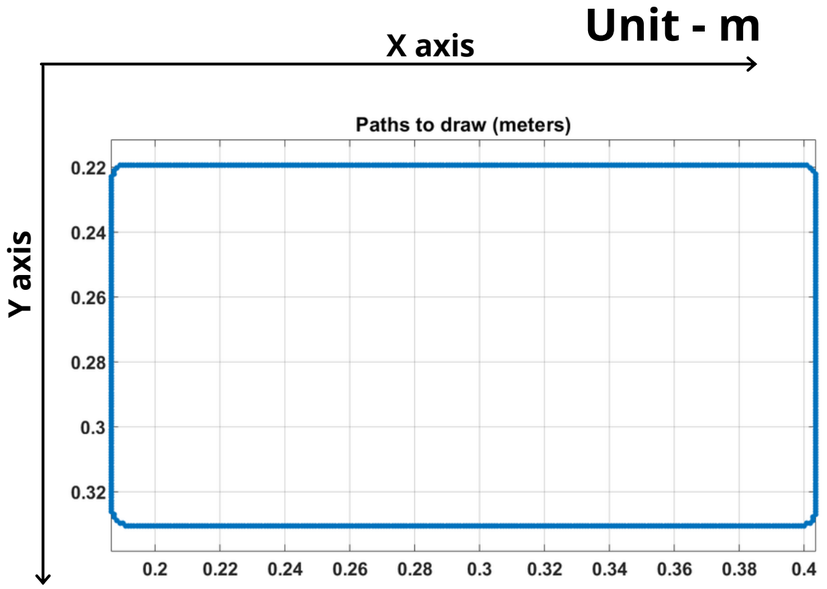


Fig 45 : Scaled Meter Output

PREPROCESSING

Scaling and Coordinate Reduction

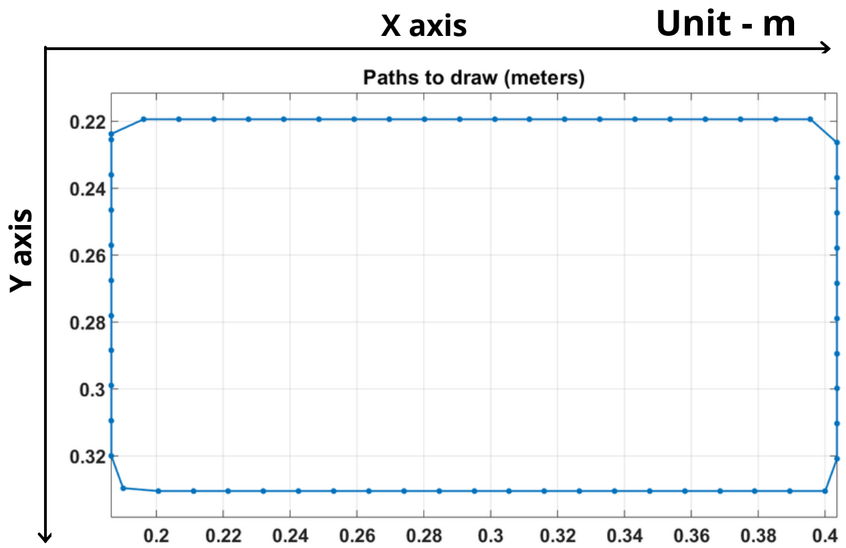
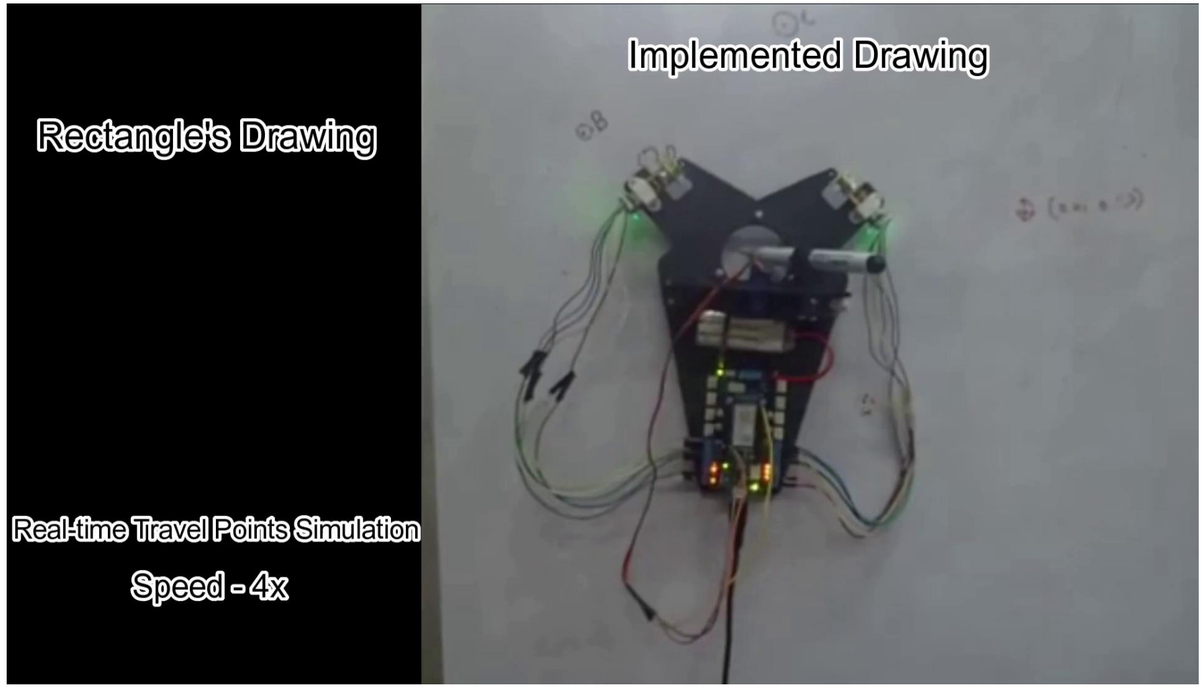


Fig 46 Reduced Output Image with resultant coordinate Output.



Vid 8 Implementation Video

In drawing process first the image is divided into distinct segments represented with different colors. More the segments higher is the complexity in drawing, thus affecting drawing quality of resemblence

RESULTS

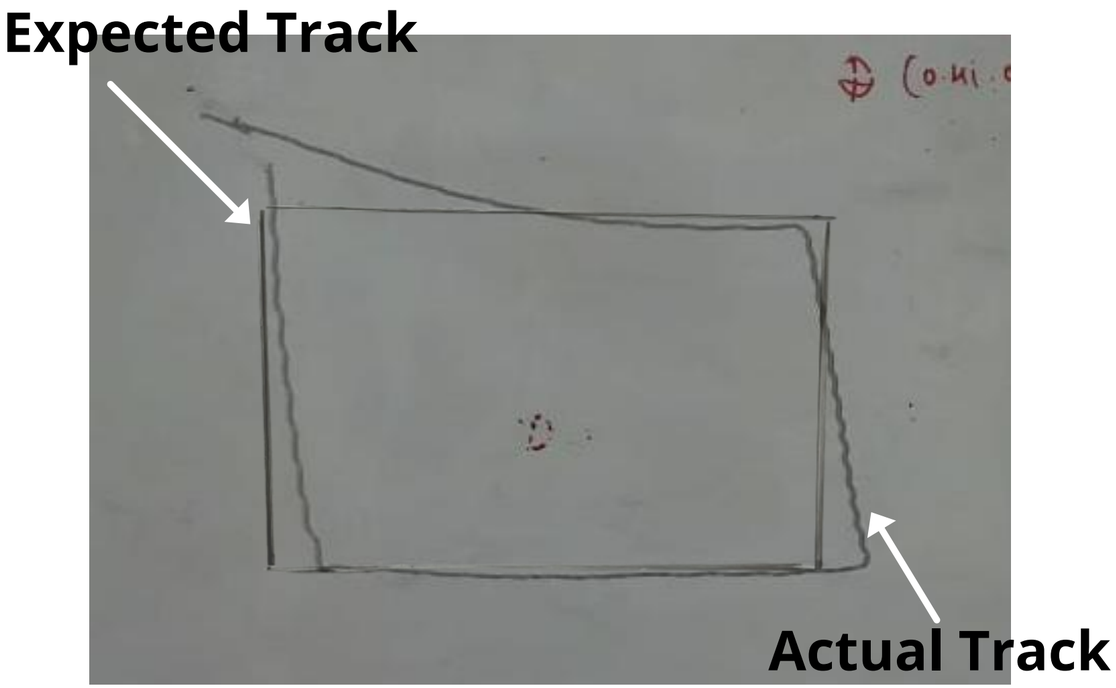


Fig 49 : Output Drawing without PD Controller

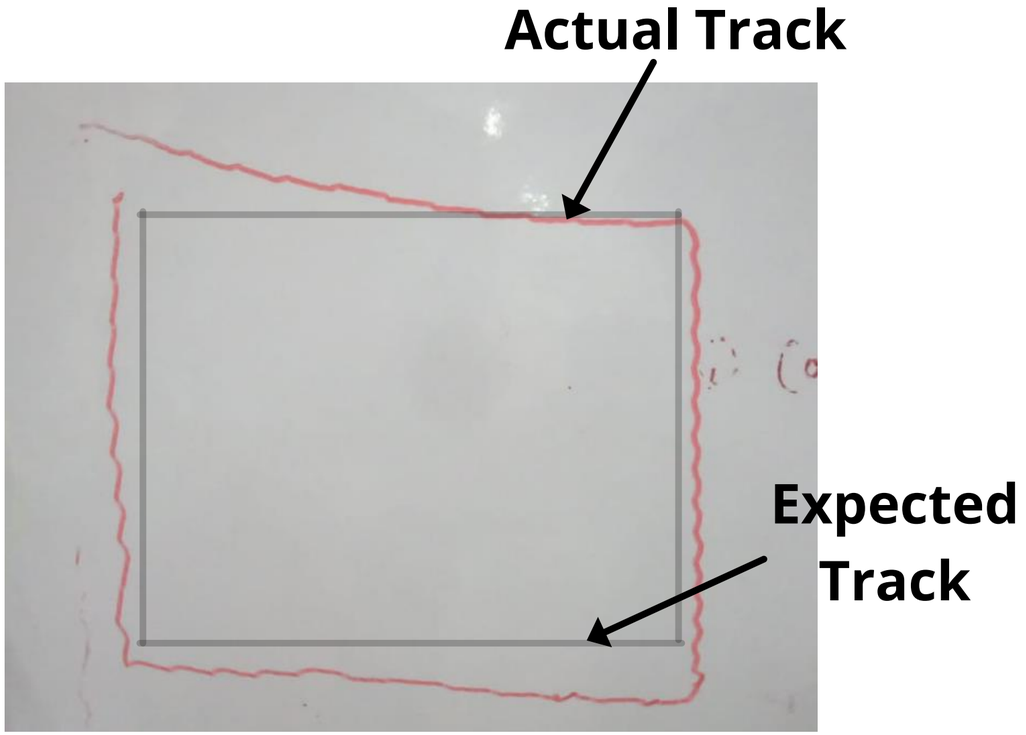


Fig 50: Output Drawing with PID Controller

OUTPUT - Positional Error

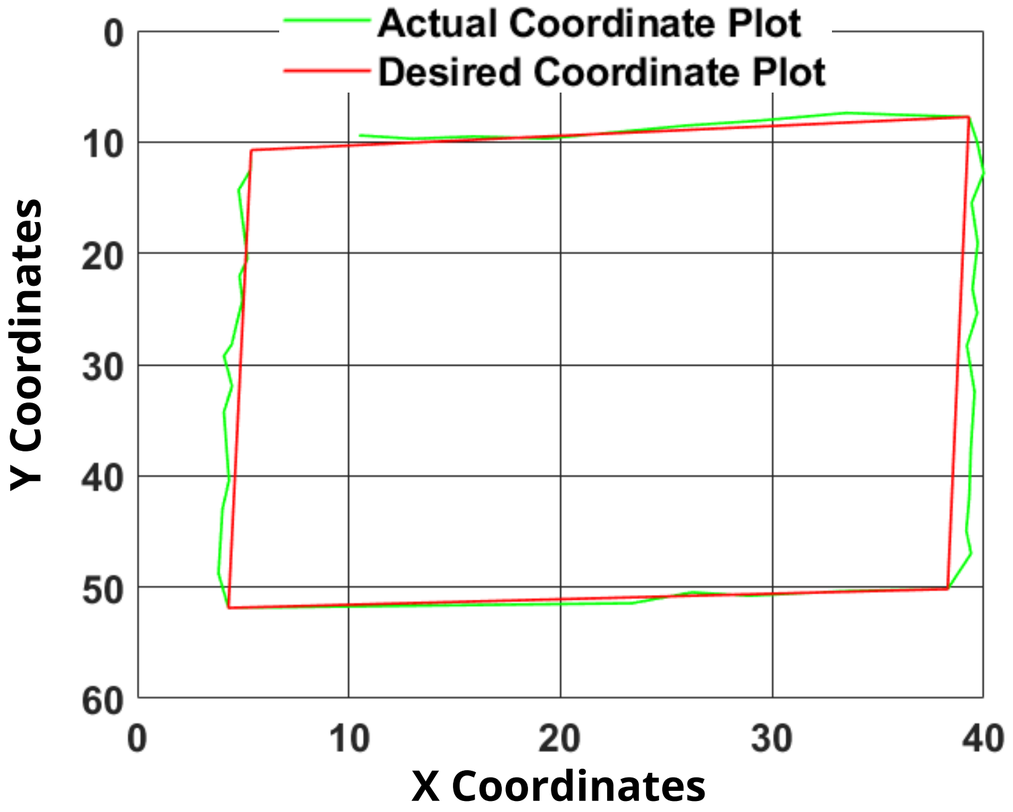


Fig 51 :Actual Track vs Expected Track

Root Mean Square Error: 7.5 cm

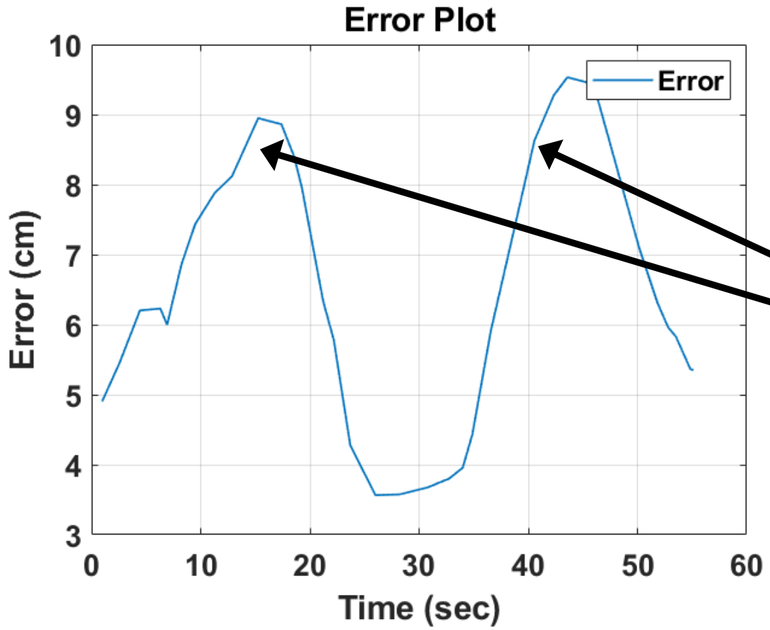
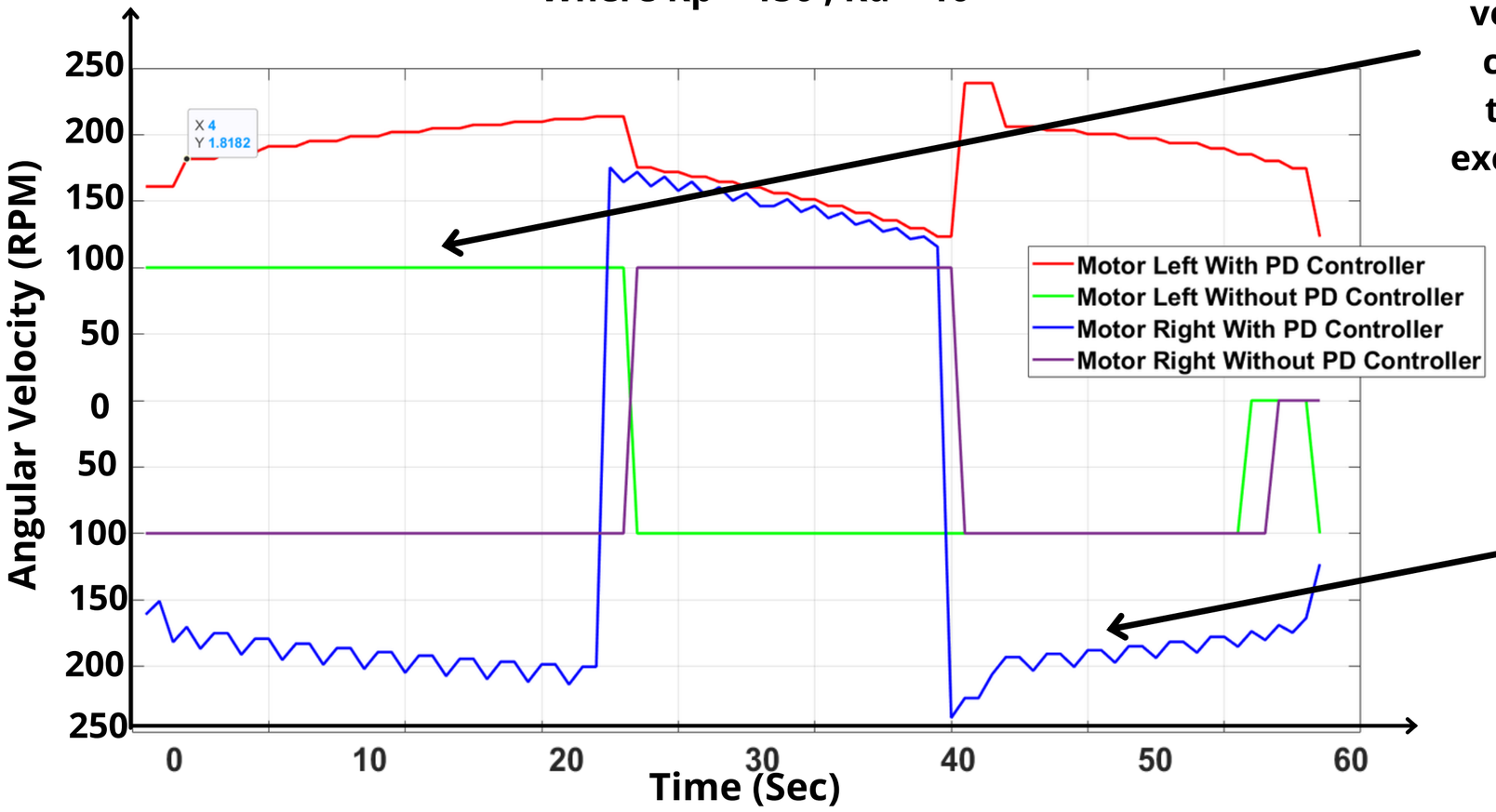


Fig D] Error Vs Time Plot

Here two error peaks in graph as error in drawing upper side and lower side of rectangle consist more fluctation but the changes appears to be gradual

OUTPUT - Velocity Vs Time

Where Kp = 150 , Kd = 10

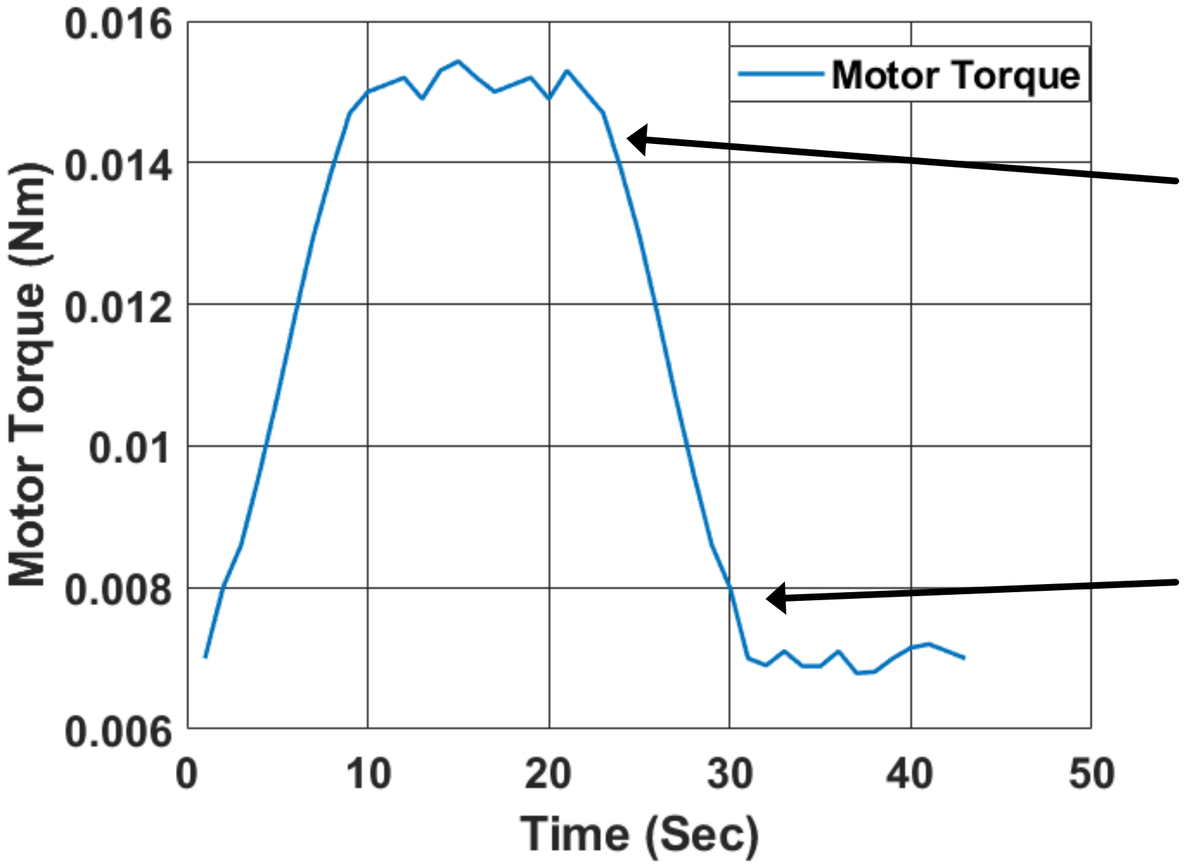


Here, reading for angular velocity without PD controller hits the threshold causing excessive energy loss.

Here, reading for angular velocity with PD controller doesn't hits the threshold and acquire required angular velocity

Fig 52: Angular Velocity Comparison with and without PD on Left Motor and Right Motor

OUTPUT - Torque Vs Time



Here, during drawing robot reaches the point with highest motor torque value

Here, A dip in motor torque can be observed as this region drawing occurs at lower most region of drawing board

Fig 53 : Motor Torque Vs Time

Application & Future Advancement

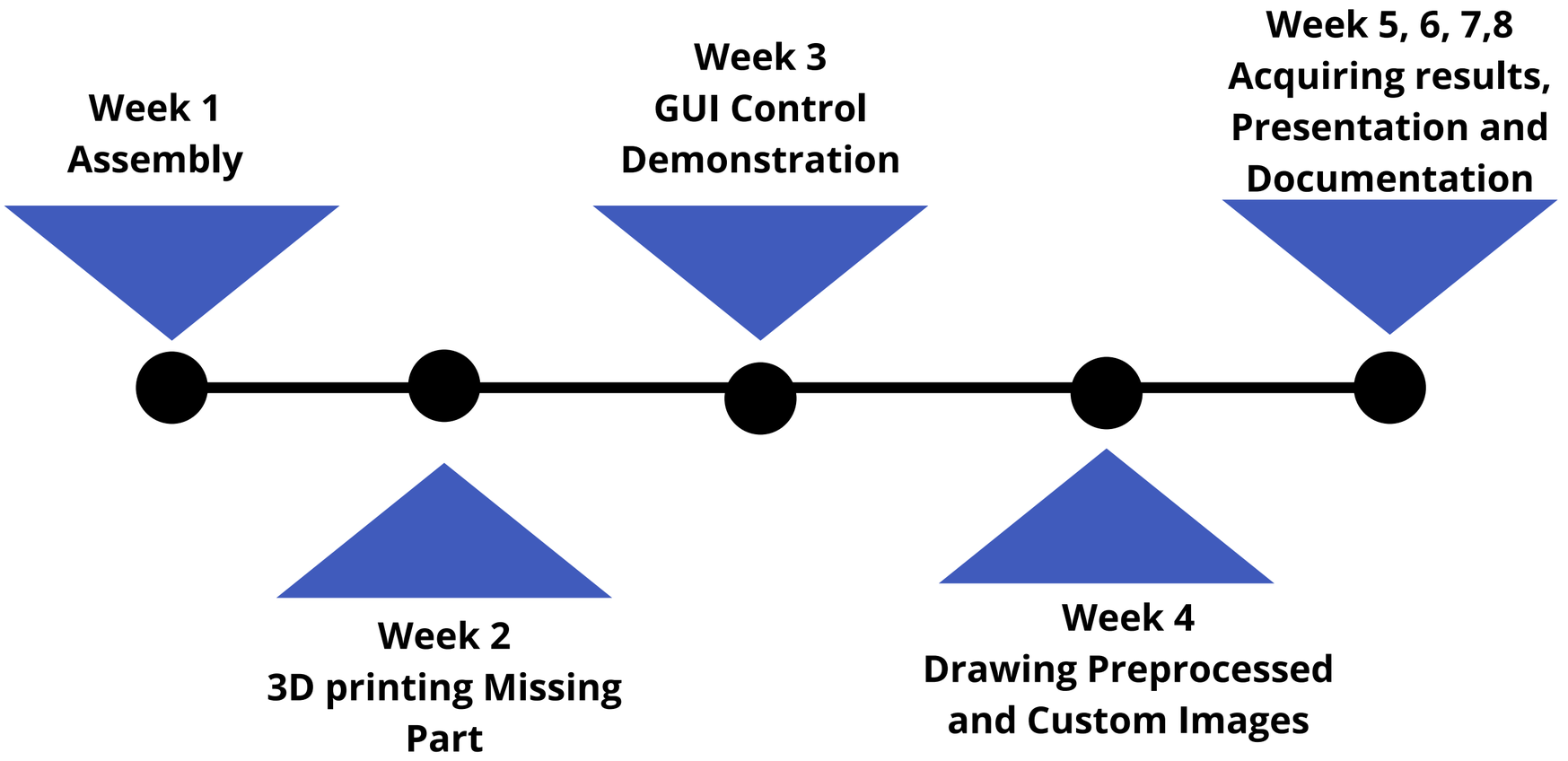
Training : This robot helps educators / students to understand the integration and scaling of coordinates with the basic image processing.

Future Advancements : More optimized alogrithm can be implemented for smooth and more recognizable drawings. Switching to stepper motor can drastically improve the movement precision.

Remarks

As we know, for x-y plane motion our drawing robot relys on string and pulley based mechanism. This allows the robot to move in any possible direction with limited constraints thus ensuring flexibility and effective utilization of workspace.
In some case beacuse of the less contraints, string tends to induce error in coordinate plotting and draws curved segments. Thus causing limitation for drawing robot to draw complex images and structure.

Timelin
e


Week 1
Assembly
Week 2
3D printing Missing
Part
Week 3
GUI Control
Demonstration
Week 4
Drawing Preprocessed
and Custom Images
Week 5, 6, 7,8
Acquiring results,
Presentation and
Documentation

Thank You!