PROJECT SUBMISSIONS

Program: **Mechatronics Engineering**

Course Code: *MCT – 333s*

Course Name: *Mechatronic System Design*

Examination Committee

DR. Mohamed Ibrahim
DR. Mohamed Omar
Eng. Hossam Moataz
Eng. Mohamed Osama

Ain Shams University
Faculty of Engineering
Spring Semester – 2022

Student Personal Information for Group Work

Student Names:	Student Codes:
Abdelrahman Mahmoud Ehab Leithy	1809161
Ahmed Yasser Ahmed Abdel Salam	1805632
Ali Hany Ahmed Ali	1803896
Anas Ahmed Talaat Khaled	1806766
El Sayed Ayman El Sayed Ali	1804765
Mahmoud Ayman Abdelaziz EL Said	1802936

Contribution Sheet:

Names	ID	Contribution
Abdelrahman Mahmoud Ehab Leithy	1809161	Matlab , Wiring of Design
Ahmed Yasser Ahmed Abdel Salam	1805632	Control
Ali Hany Ahmed Ali	1803896	Control
Anas Ahmed Talaat Khaled	1806766	Design , Wiring
El Sayed Ayman El Sayed Ali	1804765	Control , Wiring
Mahmoud Ayman Abdelaziz EL Said	1802936	Design , Matlab

1. Overview

The construction of the machine is based on DIY CNC Laser Engraver machine where the goal was to make the simplest CNC machine with minimum parts possible. It uses 2 NEMA 23 stepper motors for the X, Y and a servo motor for the Z axis motion, The brain of this CNC Pen plotter machine is an Arduino UNO board in combination with a CNC shield and two TB6600 stepper drivers

We have to provide co-ordinates number of any text, shape, and image in order to take a print, of course, we will take help of a software to convert our text, shape or image in co-ordinate form that we will see later in this post. We generally used to call that that co-ordinate number containing the file is "Gcode" (general code), so we have to save our files in the "Gcode" format.

2. Scope of the project

Design of a mechatronic system and applying the concept of concurrent engineering and integration between the different design domains using the V-Shaped model for Design of Mechatronic Systems

- Mechanical System (Robot Body & Mechanism)
- Electrical design (Wiring & PCB Design)
- Control System

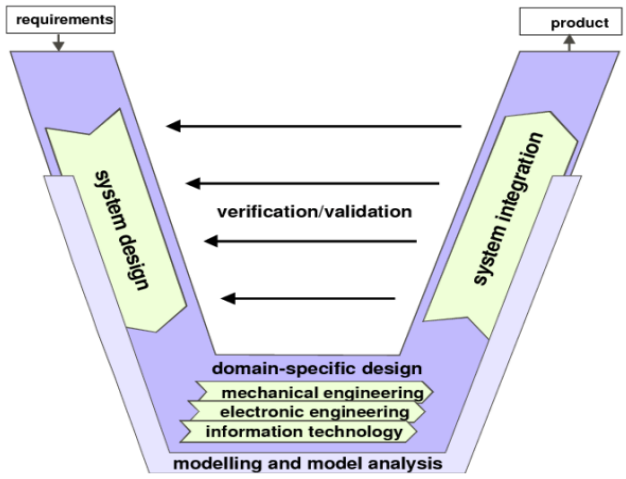

Table of Contents

Table of Figures

3. Mechanical System Design

a. Solidworks

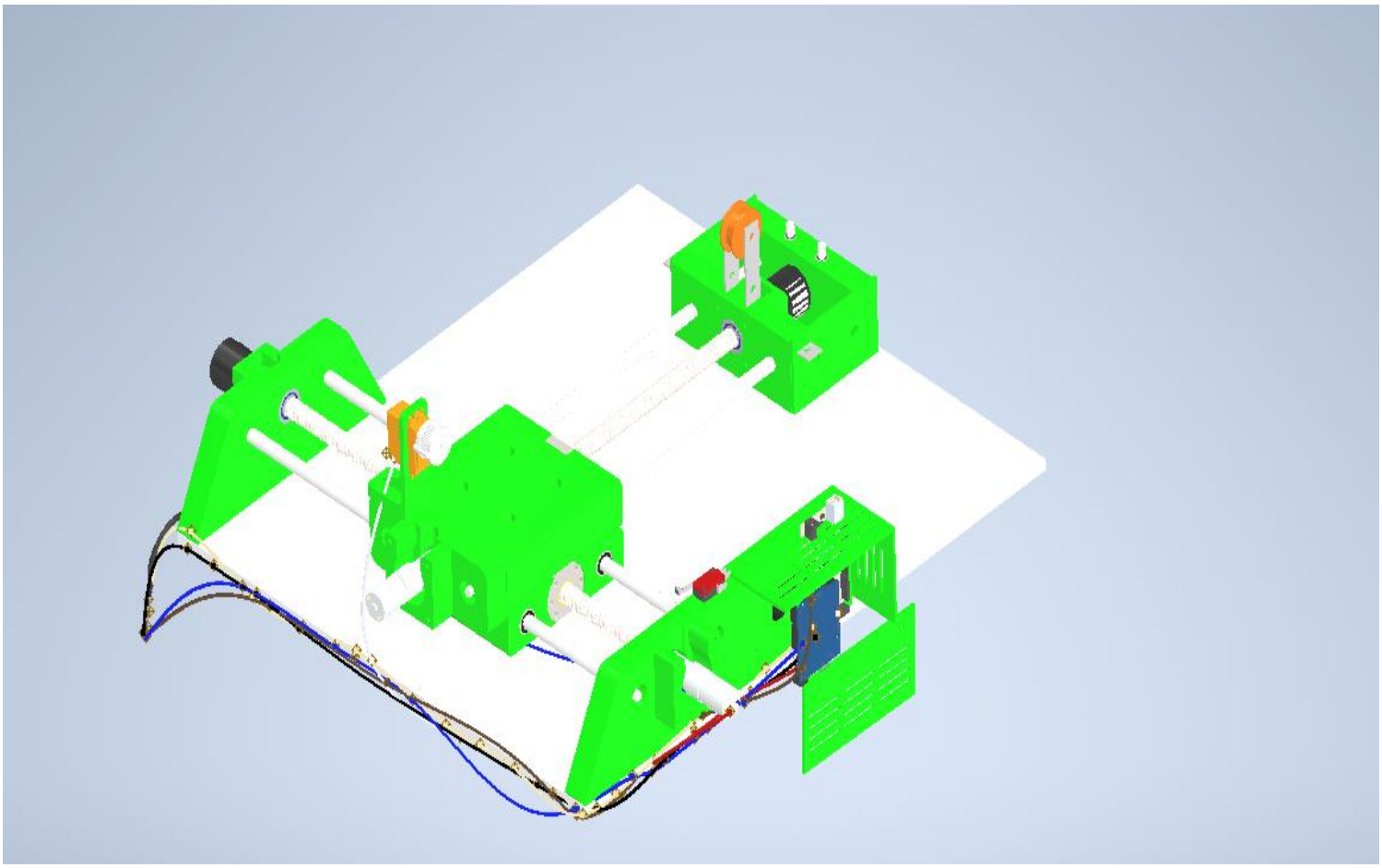

Figure 1 Mechanical Design

b. Matlab for Open Loop

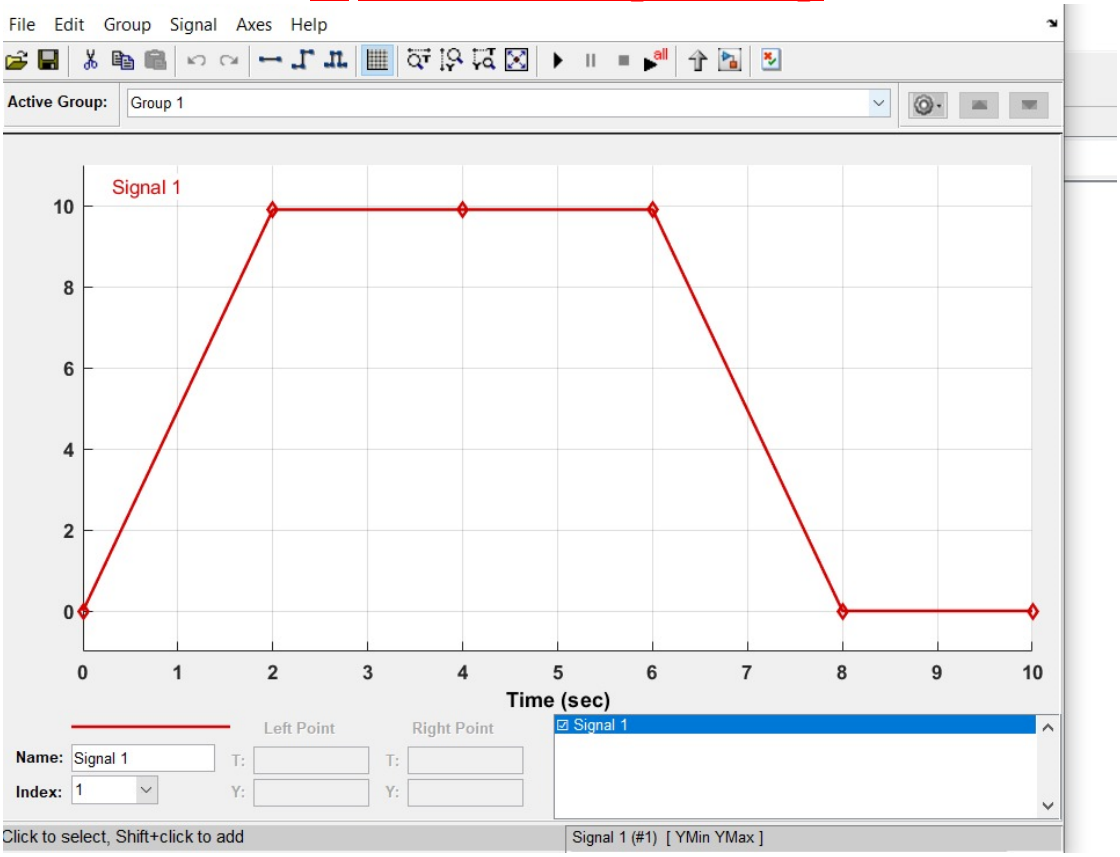

Figure 2 Matlab Open loop

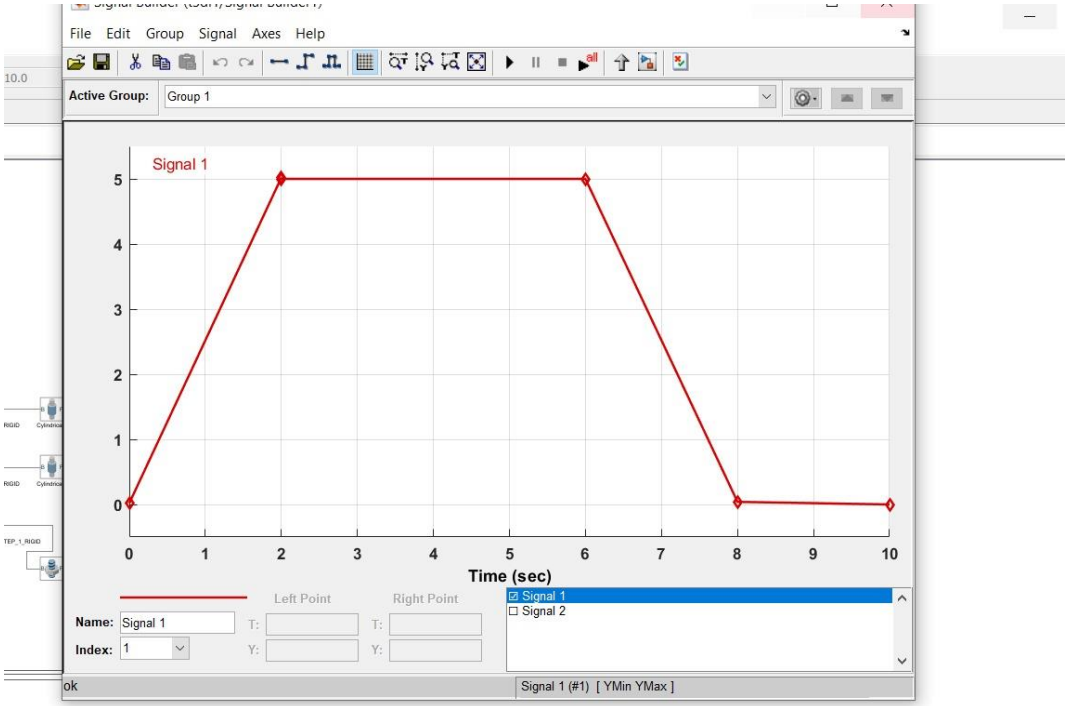

Figure 3 Matlab Open loop

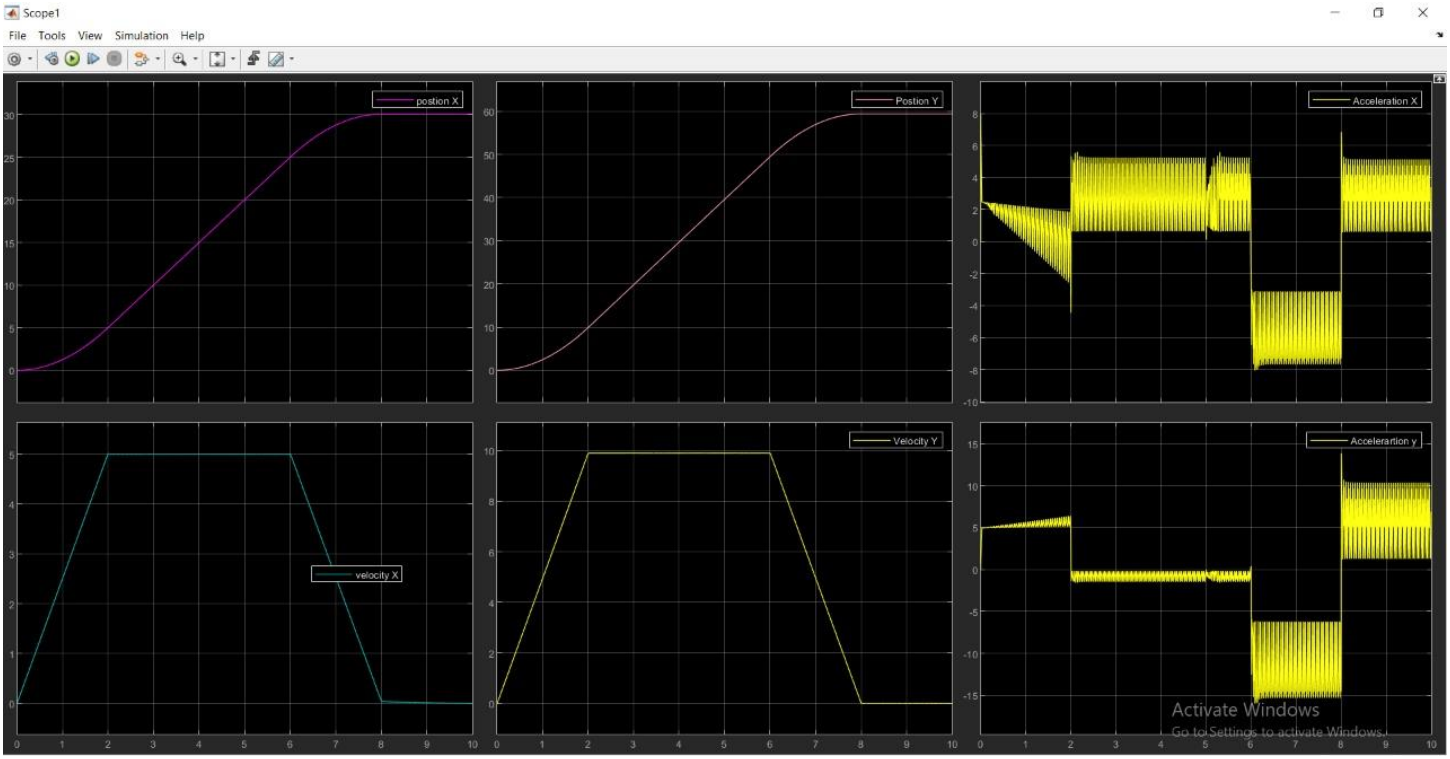

Figure 4 Matlab Open loop

➢ Link for Simulation:

https://drive.google.com/drive/folders/1lTEJYbmv6he-pZj3L7K9BmndIGL0Orxb?usp=sharing

c. Matlab for Closed Loop

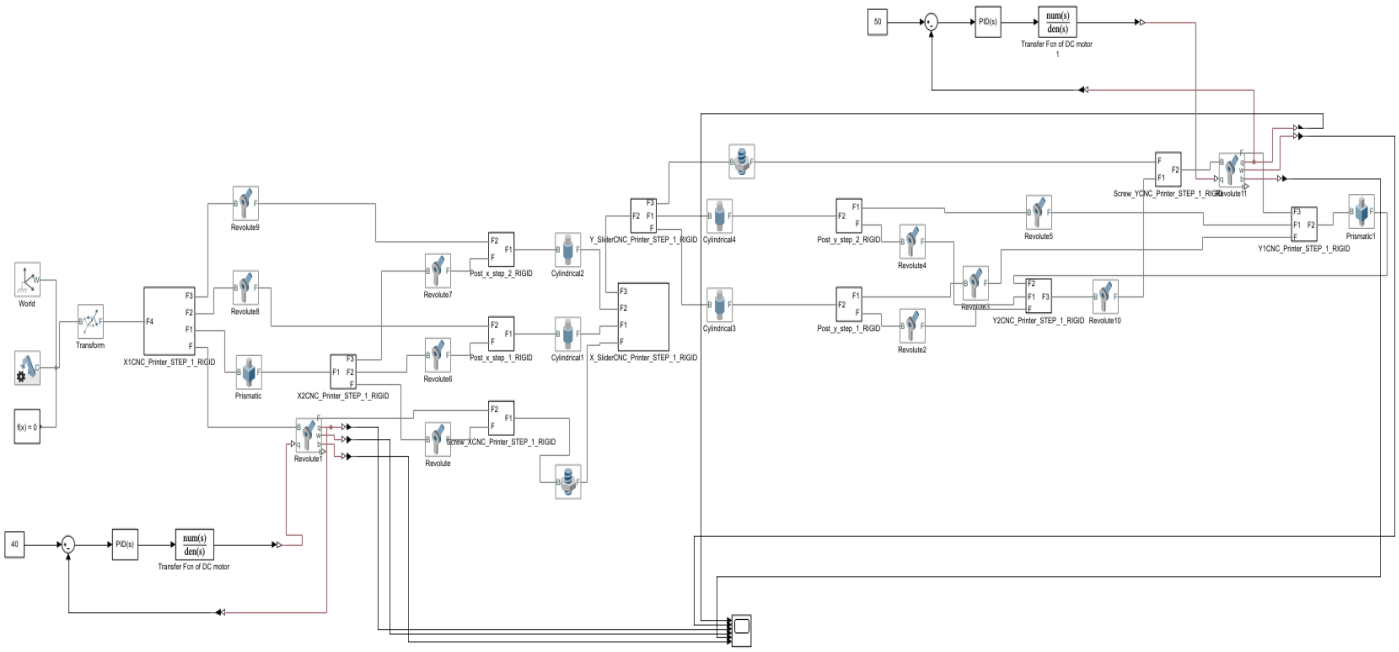

Figure 5 Matlab Closed loop

Set point

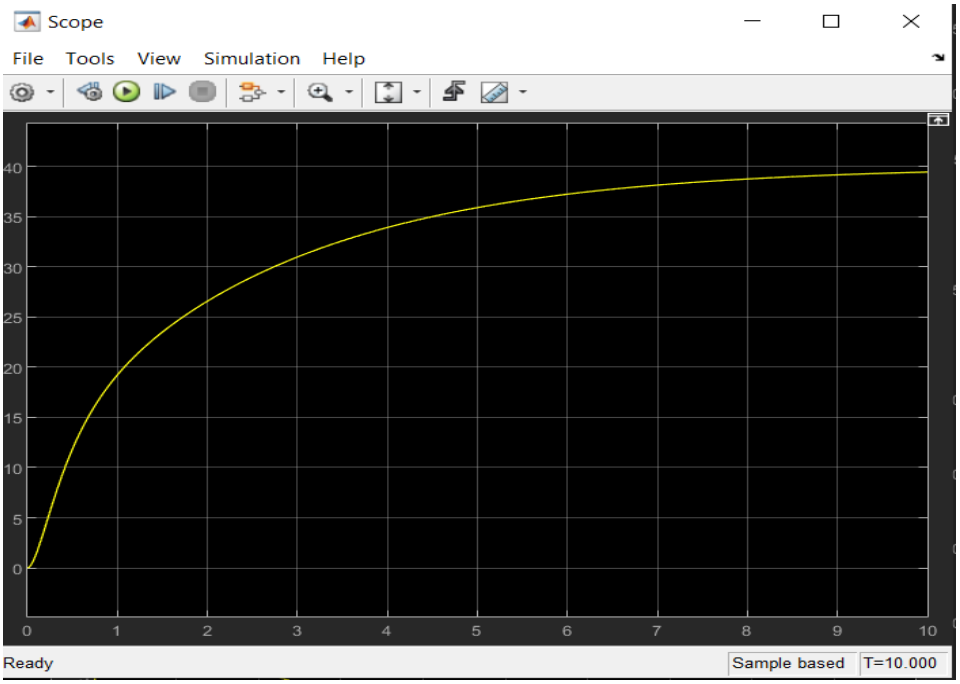

Figure 6 Matlab Closed loop

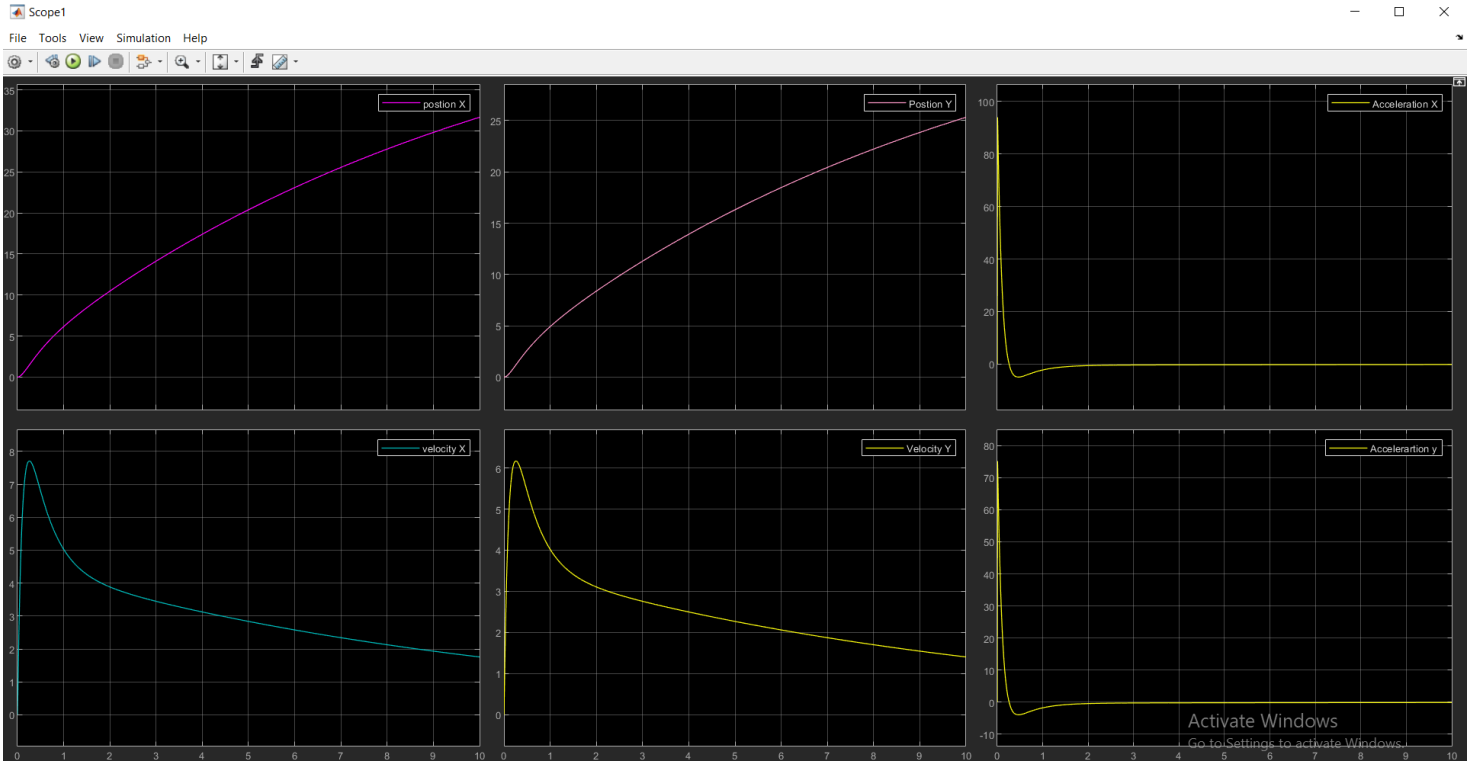

Figure 7 Matlab Closed loop

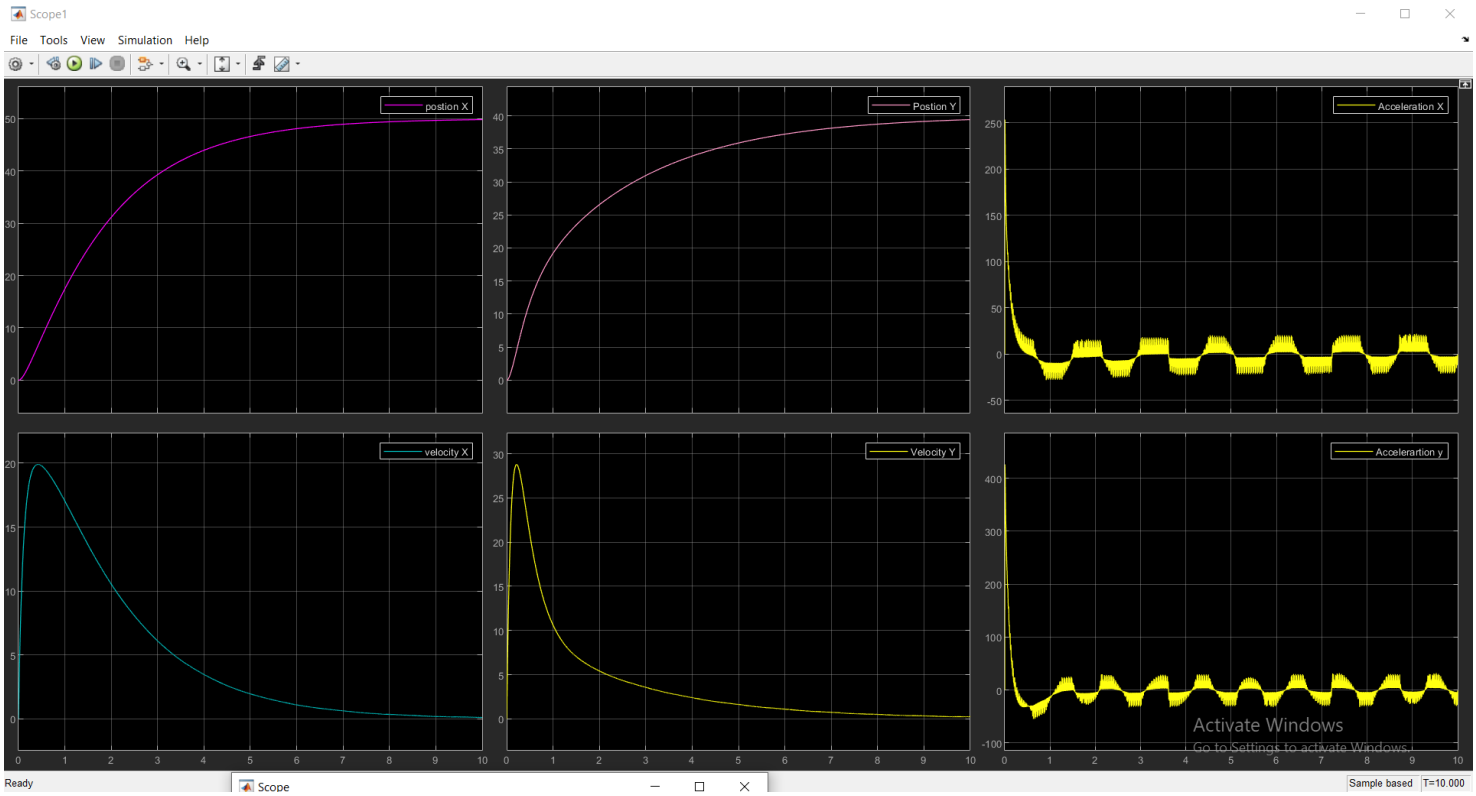

Figure 8 Matlab Closed loop

Simulation

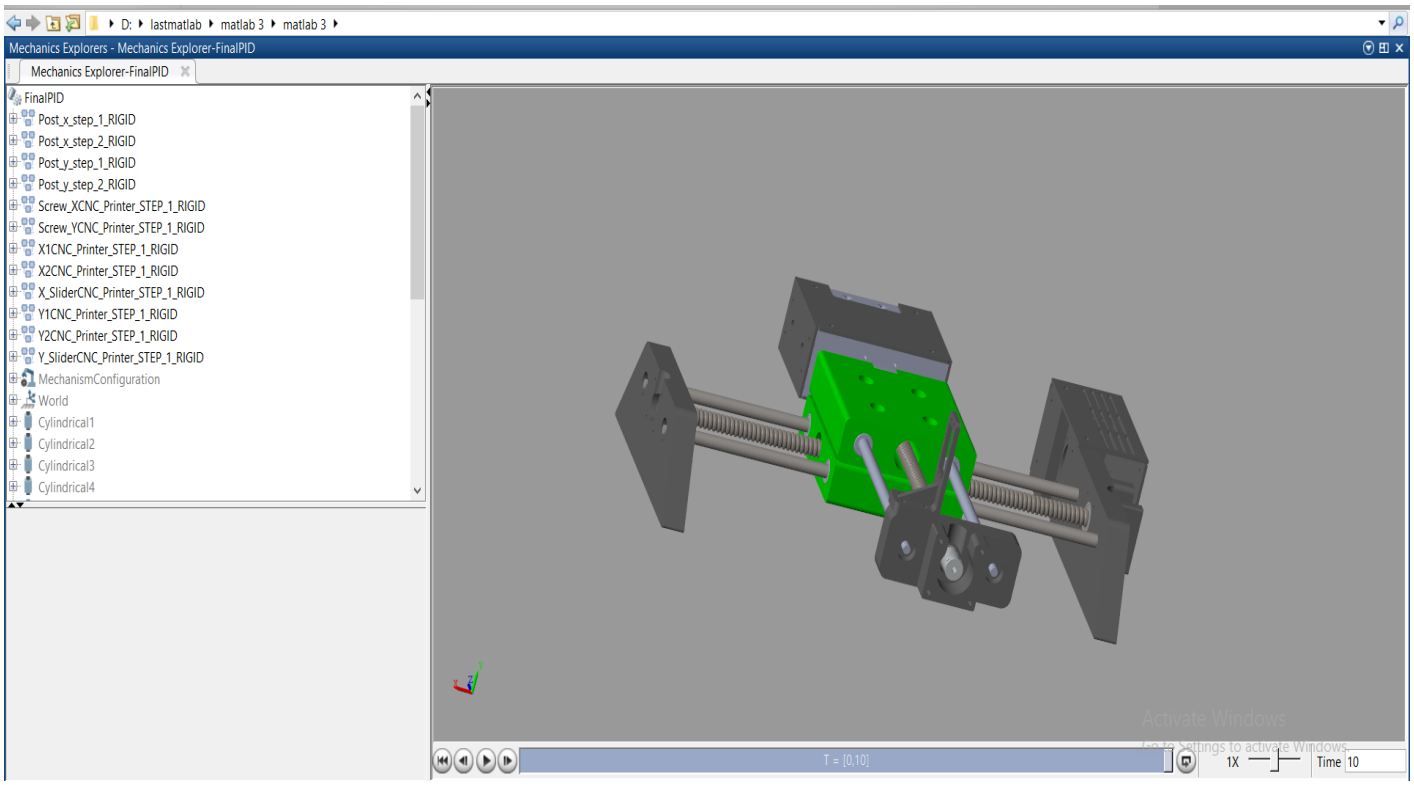

Figure 9 simulation Closed loop

➢ Link for Simulation:

https://drive.google.com/drive/folders/1lTEJYbmv6he-pZj3L7K9BmndIGL0Orxb?usp=sharing

4. Electrical System Design

a. PCB

For closed Loop

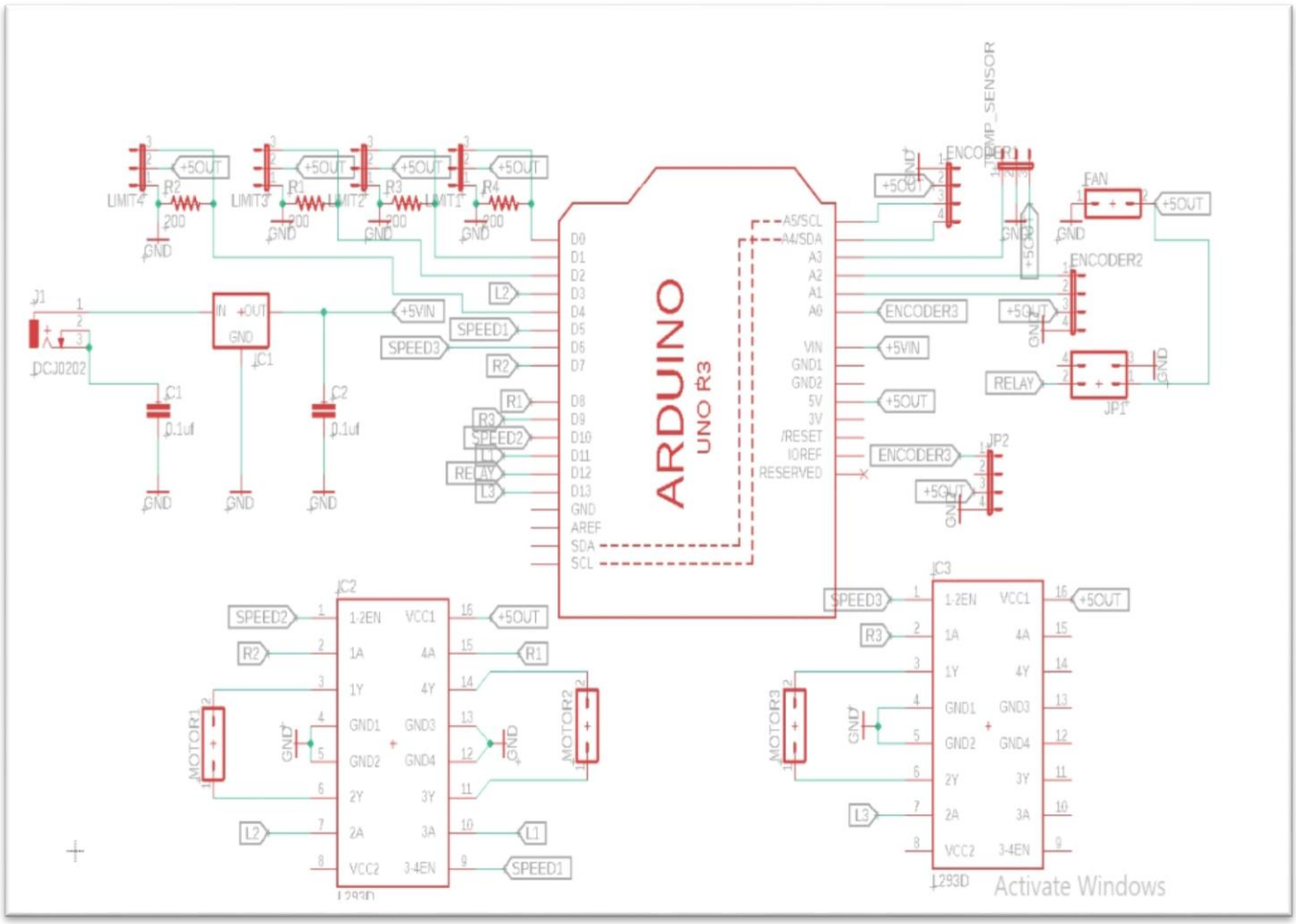

Figure 10 PCB Closed loop

5. Control System

a. Introduction:

The CNC stands for Computer Numerical Control. CNC operates on digitized data, a computer. CAM program is used to control, automate, and monitor the movements of a machine. The CNC controller works together with a series of motors and drive components to move and control the machine axes. Open-source software (GRBL Plotter) is used for executing the G-code for machining applications.

b. Software Implementation

The flow chart of application execution. At start, power supply and computer are turned on. After that all motors are initialized to its zero position. These zero positions are given through software. The circuit board is ready to accept instructions from computer. These instructions are in the form of G-codes. It will wait still instructions to be received. After instructions are received, it starts to decode it into its own language that is in the voltage and current form. As per instruction, when task is completed, it is the end of the flow of execution.

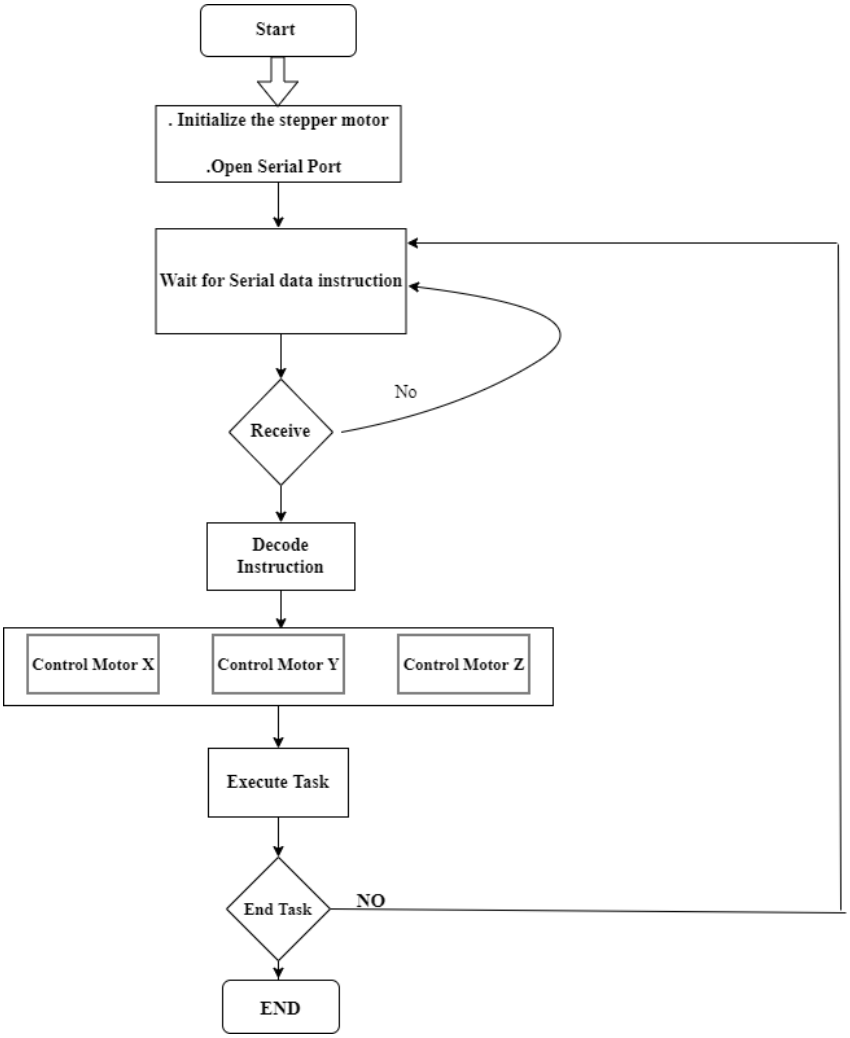

Figure 11 Software Implementation

❖The P.I.D Controller

The goal of PID controller is to adjust the control value at the OUPUT by continuously evaluating the :

ERROR e(t) = (SP (set point) – PV (PROCESS VARIABLE)) between a SETPOINT (SP) and the PROCESS VARIABLE (PV) being controlled and applies a correction based on proportional, integral, and derivative terms, to achieve the stability and rapid response in the system.

$$\text{Output} = K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$$

Where : $e = \text{Setpoint} - \text{Input}$

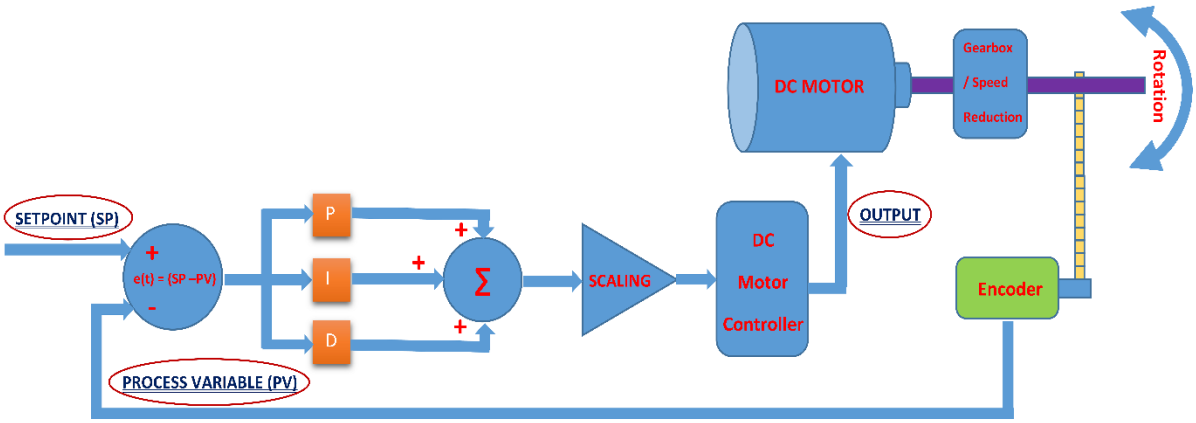

Figure 12 PID Closed loop

- In my project, **Arduino Mega 2560** is used just like a **DC servo controller**. It performs **P.I.D control** for the X and Y axis DC motors.

- Motor will be driven by speed or position but with this PID controller, the setpoint are (**step**) + (**direction**) signals from **Arduino Uno** which has **GRBL firmware** pre-installed.
- The PID control signals are as follows:

SETPOINT - SP: They are (**(step) + (direction) signals (axis-in x** + **((step) + (direction) signals in y-axis)** signals that are sent from Arduino Uno R3 with a CNC Shield to Arduino Mega 2560, and the Arduino Uno has GRBL firmware pre-installed.

PROCESS VARIABLE - PV: The measured feedback value from quadrature magnetic encoders to Arduino Mega 2560.

OUTPUT: The PWM signals from Arduino L293D Motor Shield (controlled by Arduino Mega 2560) to printer DC motors.

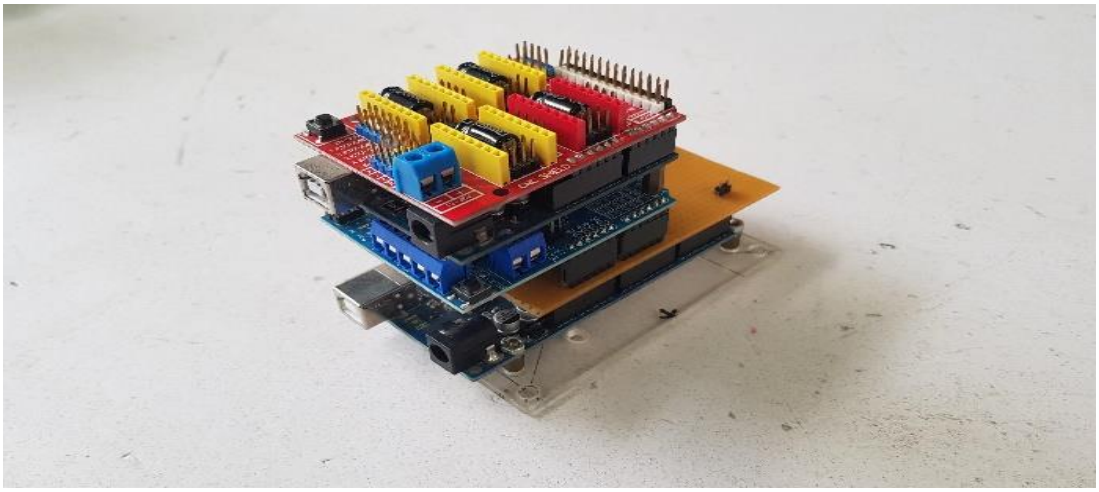

Figure 13 CNC Shield

- The **STEP** and **DIR** signals are sent from **GRBL** CNC Shield of (**Arduino Uno to Arduino Mega 2560)**. Arduino Mega 2560 receive these **STEP** and **DIR** signals of each axis X/Y, then combine them to create the **SETPOINT** values for each PID controller.

$$double\ SETPOINT\_X\ =\ 0;$$

$$double\ SETPOINT\_Y\ =\ 0;$$

- **INPUT_X/ INPUT_Y**: They are feedback signals which are read from magnetic encoders of X/ Y DC motors.

$$double\ INPUT\_X;$$

$$double\ INPUT\_Y;$$

- **OUTPUT_X/ OUTPUT_Y**: They are PWM output signals which control the X/Y motors.

$$double\ OLD\_INPUT\_X\ =\ 0;$$

$$double\ OLD\_INPUT\_Y\ =\ 0;$$

- **K_P/ K_I/ K_D**: They're tuning parameters. These affect how the PID will change the output.

1. **For motor X**:

$$double\ KP_X\ =\ 20.0;$$

$$double\ KI\_X\ =\ 0.03;$$

$$double\ KD\_X\ =\ 0.01;$$

2. **For motor Y**:

$$double\ KP\_Y\ =\ 9.0;$$

$$double\ KI\_Y\ =\ 0.02;$$

$$double\ KD\_Y\ =\ 0.01;$$

❖ Closed Loop:

Power supply on if (G-code generate) →

No → all three motors will stop (three DC motor integrated with magnetic encoder in XYZ direction)

Yes → all motors will move depends on G-code

three DC motors will move, but if the point to which the pen will move is less than the point at which it was coordinated, the motor will rotate CCW and if the point to which the pen will move is greater than the point at which it was coordinated, the motor will rotate CW. And in the end, I will know when the execution procedure has been completed at the end of the pulses at the encoder.

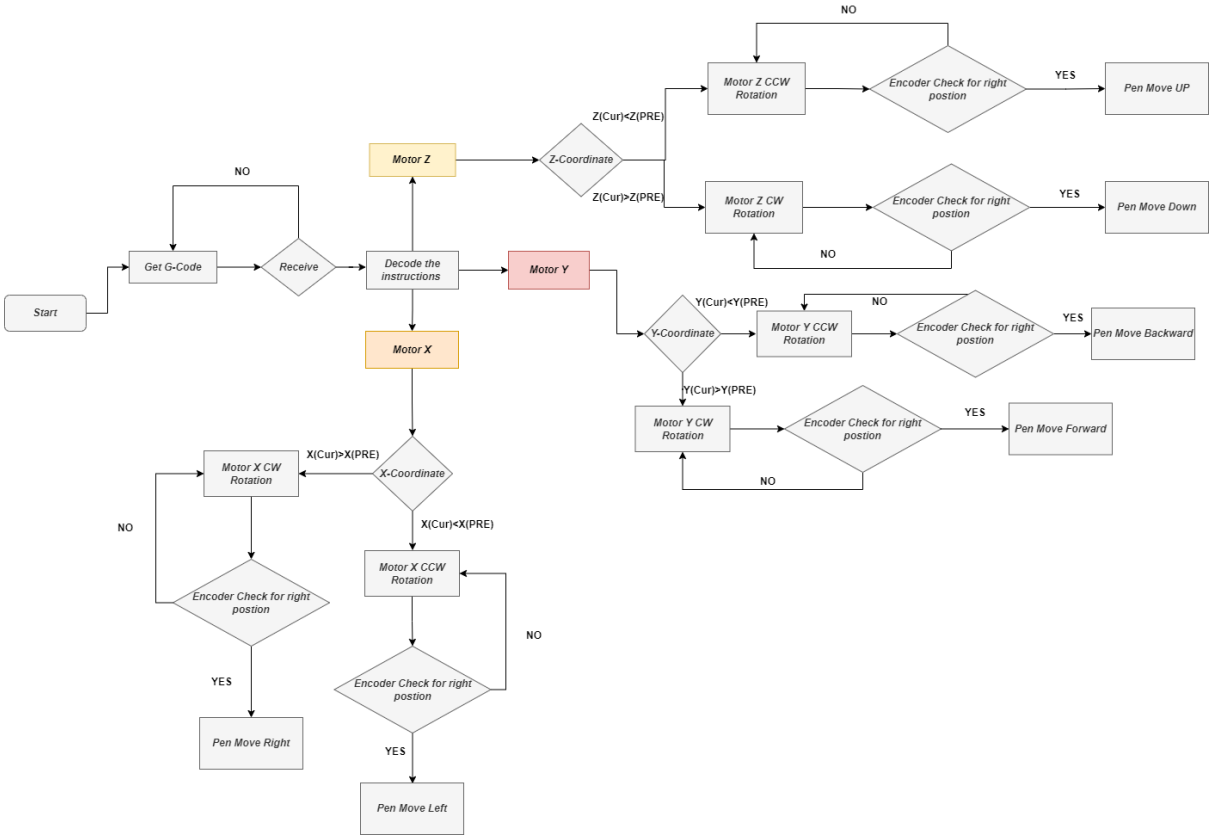

Figure 14 Flow-chart closed loop

❖ Project Code:

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
// Timer2 library
#include "FlexiTimer2.h"

// PID library
#include <PID_v1.h>

// AFMotor library
#include <AFMotor.h>

// Quadrature Encoder Library
#include "Encoder.h"

// Create the motor driver instances
AF_DCMotor motorX(1, MOTOR12_8KHZ);
AF_DCMotor motorY(2, MOTOR12_8KHZ);

// Set up pins for the quadrature encoders - Arduino MEGA2560 has 6 interrupt pins.
#define EncoderX_ChannelA   18  // Interrupt 5
#define EncoderX_ChannelB   22
#define EncoderY_ChannelA   20  // Interrupt 3
#define EncoderY_ChannelB   24

// Set up STEP & DIRECTION pins for X and Y axis
#define STEP_XPIN           19  // Interrupt 4
#define STEP_YPIN           21  // Interrupt 2
#define DIR_XPIN            23
#define DIR_YPIN            25

// Turn on/ off debugging for X/Y motor
#define DEBUG_X             0
```

Figure 15 Code Closed loop

```
#define DEBUG_Y             0

// For calculating the actual movements
#define STEPSPERMM_X      24.5    // STEP/mm is used in the GRBL for DC motor X axis.
#define DEADBW_X          6   // Deadband width = 4.5 --> Acceptable error for positioning in mm: 0.18mm.

#define STEPSPERMM_Y      192.0   // STEP/mm is used in the GRBL for DC motor Y axis.
#define DEADBW_Y          19.2    // Deadband width = 19.2 --> Acceptable error for positioning in mm: 0.1mm.


// Set up Input
double INPUT_X;
double INPUT_Y;

double OLD_INPUT_X = 0;
double OLD_INPUT_Y = 0;

// Set up Actual value
double ACTUAL_X_MM;
double ACTUAL_Y_MM;

double OLD_ACTUAL_X_MM;
double OLD_ACTUAL_Y_MM;


// PID controller constants
double KP_X = 60.0;     // P for X motor
double KI_X = 0.02;     // I for X motor
double KD_X = 0.5;     // D for X motor
```

`

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
double KP_Y = 60.0;      // P for Y motor
double KI_Y = 0.02;     // I for Y motor
double KD_Y = 0.7;     // D for Y motor

// The Output variable motor speed to the motor driver
double OUTPUT_X;
double OUTPUT_Y;

double OLD_OUTPUT_X = 0;
double OLD_OUTPUT_Y = 0;

// Setpoint
double SETPOINT_X = 0;
double SETPOINT_Y = 0;

double OLD_SETPOINT_X = 0;
double OLD_SETPOINT_Y = 0;

double ERROR_X = 0;
double ERROR_Y = 0;

// PID controller
PID myPID_X(&INPUT_X, &OUTPUT_X, &SETPOINT_X, KP_X, KI_X, KD_X, DIRECT);
PID myPID_Y(&INPUT_Y, &OUTPUT_Y, &SETPOINT_Y, KP_Y, KI_Y, KD_Y, DIRECT);

// Setup optical encoders
Encoder XEncoder(EncoderX_ChannelA, EncoderX_ChannelB);
Encoder YEncoder(EncoderY_ChannelA, EncoderY_ChannelB);

void setup()
```

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
{
  // For debugging
  Serial.begin(115200);

  pinMode(STEP_XPIN, INPUT);
  pinMode(STEP_YPIN, INPUT);
  pinMode(DIR_XPIN, INPUT);
  pinMode(DIR_YPIN, INPUT);

  // The stepper simulator
  attachInterrupt(4, doXstep, RISING);  // PIN 19 (Interrupt 4) - Interrupt X step at rising edge pulses
  attachInterrupt(2, doYstep, RISING);  // PIN 21 (Interrupt 2) - Interrupt Y step at rising edge pulses

  // Outpout PWM limits
  myPID_X.SetOutputLimits(-255,255);
  myPID_Y.SetOutputLimits(-255,255);

  // Compute output every 1ms
  myPID_X.SetSampleTime(1);
  myPID_Y.SetSampleTime(1);

  // Setup PID mode
  myPID_X.SetMode(AUTOMATIC);
  myPID_Y.SetMode(AUTOMATIC);

  // Apply PID every 1ms by FlexiTimer2
  FlexiTimer2::set(1, 1.0/1000, doPID);
  FlexiTimer2::start();
}
```

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
void loop()
{
  // Read X and Y axis optical encoders
  INPUT_X = XEncoder.read();
  INPUT_Y = YEncoder.read();
//  Serial.print("Encoder X = ");
//  Serial.println(INPUT_X);

  // Calculating the error
  ERROR_X = (INPUT_X - SETPOINT_X);
  ERROR_Y = (INPUT_Y - SETPOINT_Y);
  Serial.print("Error X = ");
  Serial.println(ERROR_X);
  // For debugging
 if (DEBUG_X)
 {
  ACTUAL_X_MM = INPUT_X / STEPSPERMM_X;
 // Debugging X motor actual position in mm
 if (OLD_ACTUAL_X_MM != ACTUAL_X_MM)
 {
    Serial.print("ACTUAL X(MM): ");
    Serial.println(ACTUAL_X_MM);
    OLD_ACTUAL_X_MM = ACTUAL_X_MM;
 }
 // Debugging position X encoders
 if (OLD_INPUT_X != INPUT_X)
 {
    Serial.print("POSITION X: ");
    Serial.println(INPUT_X);
    OLD_INPUT_X = INPUT_X;
```

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
  }
  // Debugging X stepping input
  if ( SETPOINT_X != OLD_SETPOINT_X )
  {
    Serial.print("SETPOINT X: ");
    Serial.println(SETPOINT_X);
    OLD_SETPOINT_X = SETPOINT_X;
  }
  // Debugging X motor PWM output
  if ( OUTPUT_X != OLD_OUTPUT_X )
  {
    Serial.print("OUTPUT X: ");
    Serial.println(OUTPUT_X);
    OLD_OUTPUT_X = OUTPUT_X;
  }

 }
 if (DEBUG_Y)
 {
    ACTUAL_Y_MM = INPUT_Y / STEPSPERMM_Y;
 // Debugging Y motor actual position in mm
 if (OLD_ACTUAL_Y_MM != ACTUAL_Y_MM)
 {
    Serial.print("ACTUAL Y(MM): ");
    Serial.println(ACTUAL_Y_MM);
    OLD_ACTUAL_Y_MM = ACTUAL_Y_MM;
  }
  // Debugging Y stepping input
  if ( SETPOINT_Y != OLD_SETPOINT_Y )
  {
```

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
    Serial.print("SETPOINT Y: ");
    Serial.println(SETPOINT_Y);
    OLD_SETPOINT_Y = SETPOINT_Y;
  }
  // Debugging position Y encoders
  if (OLD_INPUT_Y != INPUT_Y)
  {
    Serial.print("POSITION Y: ");
    Serial.println(INPUT_Y);
    OLD_INPUT_Y = INPUT_Y;
  }
  // Debugging Y motor PWM output
  if ( OUTPUT_Y != OLD_OUTPUT_Y )
  {
    Serial.print("OUTPUT Y: ");
    Serial.println(OUTPUT_Y);
    OLD_OUTPUT_Y = OUTPUT_Y;
  }

 }
}

void doXstep()
{
  if ( digitalRead(DIR_XPIN) == HIGH ) SETPOINT_X--;
  else SETPOINT_X++;
}

void doYstep()
{
```

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
  if ( digitalRead(DIR_XPIN) == HIGH ) SETPOINT_X--;
  else SETPOINT_X++;
}

void doYstep()
{
  if ( digitalRead(DIR_YPIN) == HIGH ) SETPOINT_Y--;
  else SETPOINT_Y++;
}

void doPID()
{
  interrupts();
  myPID_X.Compute();
  myPID_Y.Compute();
  if (abs(ERROR_X) < DEADBW_X) // If the motor X is in position within the deadband width (acceptable error)
    {
      motorX.setSpeed(0); // Turn off the X motor
    }
  else
    {
      motorX.setSpeed(abs(int(OUTPUT_X)));  // X Motor is regulated by PID controller ouput
    }
  if (abs(ERROR_Y) < DEADBW_Y) // If the motor Y is in position within the deadband width (acceptable error)
    {
      motorY.setSpeed(0); // Turn off the Y motor
    }
  else
    {
      motorY.setSpeed(abs(int(OUTPUT_Y)));  // Y Motor is regulated by PID controller ouput
```

CNC-PLOTTER-FROM-DC-MOTORS-AND-OPTICAL-ENCODERS

```
      motorY.setSpeed(0); // Turn off the Y motor
    }
  else
    {
      motorY.setSpeed(abs(int(OUTPUT_Y)));  // Y Motor is regulated by PID controller ouput
    }
  int directionX;
  int directionY;

  if(OUTPUT_X > 0)
    {
      directionX = FORWARD;
    }
  if(OUTPUT_X < 0)
    {
      directionX = BACKWARD;
    }

  if(OUTPUT_Y > 0)
    {
      directionY = FORWARD;
    }
  if(OUTPUT_Y < 0)
    {
      directionY = BACKWARD;
    }

  motorX.run(directionX);
  motorY.run(directionY);
}
```

6. Videos Link:

https://drive.google.com/drive/folders/10zWVY6sjim90Kg1b--hV8KTCObmYNCnC?usp=sharing