Palestine Polytechnic University

College of Engineering

Department of Mechanical Engineering

Graduation Project

Design and Implementation of the Automatic 2D Painter

Submitted to the College of Engineering
In partial fulfillment of the requirements for the Bachelor's degree
In Mechatronics engineering and computer engineering

Students:

Mohamed Al-Sweiti Mahde Sawalha Saeb Al-Sweiti

Supervisor:
Dr. Yousef Al-Sweiti

Hebron, July 26, 2021

Palestine Polytechnic University

College of Engineering

Department of Mechanical Engineering

Hebron – Palestine

Graduation Project

Design and Implementation of the Automatic 2D Painter

Submitted to the College of Engineering
In partial fulfillment of the requirements for the Bachelor's degree
In Mechatronics engineering and computer engineering

Project team:

Mohamed Al-Sweiti Mahde Sawalha Saeb Al-Sweiti

Supervisor's signature:

…….

Head of the department's signature:

……

شكر وتقدير

بعد أن من الله علينا بفضله وكرمه وجوده جل في علاه بإنهاء هذا المشروع فإننا نتقدم بجزيل الشكر والعرفان لكل من ساعدنا وساهم معنا في إنجاز هذا العمل، ونخص بالذكر مشرفنا الدكتور يوسف السويطي لما قدمه لنا من توجيه وإرشاد.

كما ونتقدم بالشكر لدائرة الهندسة الميكانيكية بجامعة بوليتكنك فلسطين على جهودها الحثيثة في مساعدتنا بتوفير المواد الخام وطباعة القطع البلاستيكية المستخدمة في مشروعنا، ونخص بالذكر المهندس جاسر المحتسب الذي لم يألل جهداً في مدِّ يَدِ العونِ لنا وتسليمنا القطع في الوقت المناسب دون تأخير، وكما نتقدم بالشكر للمهندس وائل التكروري بتقديمه لنا مختبر الحاسوب لإجراء التجارب وبتزويدنا بما لديه من خبرات لإتمام هذا العمل.

وأخيراً نتقدم بالشكر لأهالينا الأعزاء؛ فبدون دعمهم المتواصل طوال مسيرتنا التعليمية لما تمكنّا من إنجاز ما وصلنا إليه اليوم.

Abstract

This project aims to design and implement a low-cost, simple, easy to use positioning mechanism for a 2D automatic painter machine. The main idea is to use an alternative mechanism to the traditional Cartesian positioning mechanisms to control the position in the X and Y direction using two stationary actuators (the actuator frame is fixed to a stationary frame) and one timing belt. This mechanism is called the H-frame. The main advantage of this mechanism is its light weight and low number of mechanical parts, which saves power and makes the positioning faster and more practical than many of the other positioning system. A simplified 4^{th} order dynamic model is derived, which showed a good representation of how the rotational motion of the motor is transferred to the linear motion of the end-effector. Four different state-feedback tracking controllers are designed; they all show a fast response with a reduced error and achieve the desired specifications with a low torque input requirement. An image processing algorithm is presented to turn the image of the work pattern (i.e. window guard) into a graph, which is converted into geometrical paths that the system will track accordingly. This image processing algorithm makes the system user-friendly and easy to use, so the users do not need a lot of knowledge in engineering or CAD/CAM software to control to use the system. Experiments were performed on the machine. These experiments showed high repeatability and accuracy, as well as the ability to paint patterns and shapes with high complexity.

الملخص

يهدف هذا المشروع إلى تصميم وتنفيذ نظام حركة ثنائي البعد فريد من نوعه، رخيص الثمن وسهل الاستخدام. يعتبر هذا النظام بديلاً عن آليات الحركة الخطية التقليدية المستخدمة في أنظمة الـCNC. تعمل آلية الحركة في هذا النظام على التحكم بموقع أداة الطلاء عن طريق نقل الحركة بين محركين خطويين مثبتين وثماني بكرات بواسطة حزام يربط النظام ككل، بحيث يكون اتجاه حركة الأداة يعتمد على اتجاه دوران المحركين الخطويين في آن واحد. يشبه هذا النظام حرف H؛ لذلك سُمّيَ نظام الحركة هذا باسم H-Frame. من مزايا هذا النظام خفّة وزنه وقلة قطعه الميكانيكية؛ مما يجعل من هذا النظام سريع الأداء، رخيصَ الثمن، وموفراً للطاقة، مقارنةً بأنظمة الـCNC التقليدية. ويمكن استخدامه لتطبيقات أخرى كآلات الليزر، والطابعات ثلاثية الأبعاد، وآلات اللحام وغيرها.

قام فريق العمل بتصميم خوارزمية تقوم على استخراج المسار المناسب من الصورة التي يتم التقاطها للقطعة المراد رشها بالطلاء؛ حيث أن الخوارزمية لا تحتاج معلومات مسبقة عن قطعة العمل؛ لذا فهذا النظام سهل الاستخدام، ولا يحتاج إلى مختصين للتعامل معه. قام فريق العمل بإجراء تحليل فيزيائي ورياضي للنظام، وتصميم نظام تحكم، كما وقام بإجراء اختبارات أظهرت مدى دقته وفعاليته؛ حيث أظهرت نتائج هذه الاختبارات إمكانية طلاء قطعٍ بتعقيدٍ عالٍ؛ فالخوارزمية لا تتأثر بتعقيد شكل القطع.

Table of Contents

Table of Figures

Chapter 1
INTRODUCTION

1.1 Overview:

The system considered in this work is a planner positioning mechanism via an image processing algorithm. The system aims to position a cart or so called an end-effector arbitrarily in a horizontal plane within its working area. The end-effector is equipped with a tool (i.e. a spray painter head), which will spray the work piece (here it is a steel window guard) in horizontal plane XY plane. This system is called the H-frame positioning system. The motion of the end-effector is controlled using two actuators (servomotors or stepper motors) attached to a stationary frame and one timing belt. This system will be used as a CNC painting mechanism. Instead of drawing the path of the work piece using programs like AutoCAD and G-code generators. An image processing algorithm is used to generate the painting path the end-effector will move in the XY plane based on a picture of the work piece taken by a camera connected to the microcontroller, making the system easier to use and saving time to paint the piece. The accuracy and speed issues in addition to the input torque level are solved by designing alternative controllers as expressed in dynamic modeling chapter.

1.2 Literature Review:

Survey of literature review was performed related to previous work on control of flexible transmission elements. Most XY-positioning systems described were stacked ball screw and lead screw systems. However, a few publications were found describing similar parallel XY-positioning systems as the described H-Frame system, which works with stationary motors and one timing belt.

A publication was found[1]. The author described the motion of a ball screw system, which addressed an issue of having finite stiffness, leading to a difference between the position of the end-effector and the motor position. The little friction can cause the system to be back-driven at certain lead angles. A torsional

displacement feedback controller was designed to improve the tracking in this work. Another source [2] showed a system consisting of two stacked belt drive axis was studied. The dependence on the belt showed an uncertainty in the end-effector transition. Fortunately, an observer-based sliding control strategy was designed to address this issue. A source about the H-Frame mechanism [3] addressed the issue of the uncertainty of the position of the end-effector at high speeds, due to the frictional forces and the flexibility of the elastic timing belt. An adaptive gain control technique was designed for this issue. It showed good repeatability and the accuracy of the positioning was improved. Another source [4] developed a detailed 8^{th} order dynamic model of the H-Frame, which showed accurate predictions of the dynamics of the developed H-Frame. The model solved the issues of the nonlinearities due to the frictional forces. However, we found the dynamic modeling complex, but it gave us a clear imagination of the direction we went for our basic dynamic modeling of the system. For a similar mechanism, a source [5] addressed the positioning errors of a Core XY mechanism and proposed a reliable full degree of freedom dynamic modeling with simulation to detect the positioning errors accurately.

These publications and studies gave a clear motivation and a strong interest to study this unique mechanism and gave the working team on this project better understanding of its dynamics characteristics. We will use some of the information we observed from these publications to analyze and design the system.

1.3 Motivation and scope of the study:

The H-Bot mechanism has special advantages over the nonparallel traditional XY-positioning systems. The parallel configuration with stationary motors is different from most commonly used positioning mechanisms. The traditional systems are made of two independent linear axes. One of these two axes including its driving motor and mechanical pieces, is then mounted perpendicular onto the other axis. The overlapping of the motion of these two axes gives the position in x and y. However, this leads to comparably big mass to be moved, which causes the motor to exert more torque, therefore consume more power. On the other hand, in the H-frame, the motors are stationary, so their masses are not taken into consideration and the mechanical parts that transmit the power to move the end-effector are not heavy, which means there is less mass to be moved, which results in less inertia

and therefore in the ability to accelerate quicker; this makes the positioning faster with comparable motor power. For instance, most CNC machines use lead screw or ball screw mechanisms, which are relatively heavy compared to a light timing belt in the H-Frame.

The H-frame is low profile, compared to the traditional Cartesian systems, which have to arrange their axes in order to move one through the other, whereas the H-frame can be designed to be entirely in one plane. All that makes the H-Frame ideal for applications that have restrictions in space in the Z direction. Another advantage is that the H-frame is a low cost alternative to the traditional XY positioning systems, since less precision parts are needed to transmit the rotational motion of the motors into the linear motion of the end-effector. Keep in mind, the pulleys and the timing belts are cheap machine parts compared to the parts of the traditional positioning systems.

We find the idea of utilizing the image processing exciting, because it is a unique idea that allows the user to paint or plot complex patterns, instead of using the CAD/CAM software to draw those patterns, generate the G-codes and transfer the path to the system. This makes the system time-saving and user-friendly.

1.4 Recognition of the Need:

- Capability to paint pieces with complex shapes or patterns.
- The dimensions of the prototype are 400×400 mm.
- The prototype should be capable of painting pieces with dimensions up to 200×300 mm or more.
- Repeatability: The system should be capable of repeating the patterns in case we have more than one piece with the same pattern.
- The system should repeat the path once to make sure every part of the piece is painted.
- The system should have a suitable dynamic response with a settling time of less than 1 second and a low overshoot under 10%.
- The system should consume the least amount of energy possible; should be lightweight, which makes it use less torque than the traditional CNC machine.

1.5 Approach:

1.5.1 The General Working Principle of the System (Steps):

A. The user takes a picture of the working piece (e.g Window guard, pattern, etc.) using a camera connected to the controller. (In this project; we used an HD camera of an android phone).
B. The picture goes through an image processing algorithm and is turned into a graph (vertex and edges). Then this graph will be turned into a path the painter will move accordingly. This image processing algorithm will be explained later in Chapter 4.
C. The microcontroller processes the code of the path, and then sends the information to the driver that controls the speed and position of the motors (to simplify the control algorithm, two stepper motors will be used in the prototype to do this task)
D. The motors move the end-effector (the painter head) based on their direction of rotation. The system automatically commands the painter to start painting the piece according to the path generated in step B. The process should never leave any part of the working piece unpainted.
E. After the face of the piece is painted, the system moves the painter back to the starting position.
F. The piece is flipped.
G. After the piece is flipped, the system paints the back of the piece using the same algorithm.

1.5.2 The general Flowchart of the system:

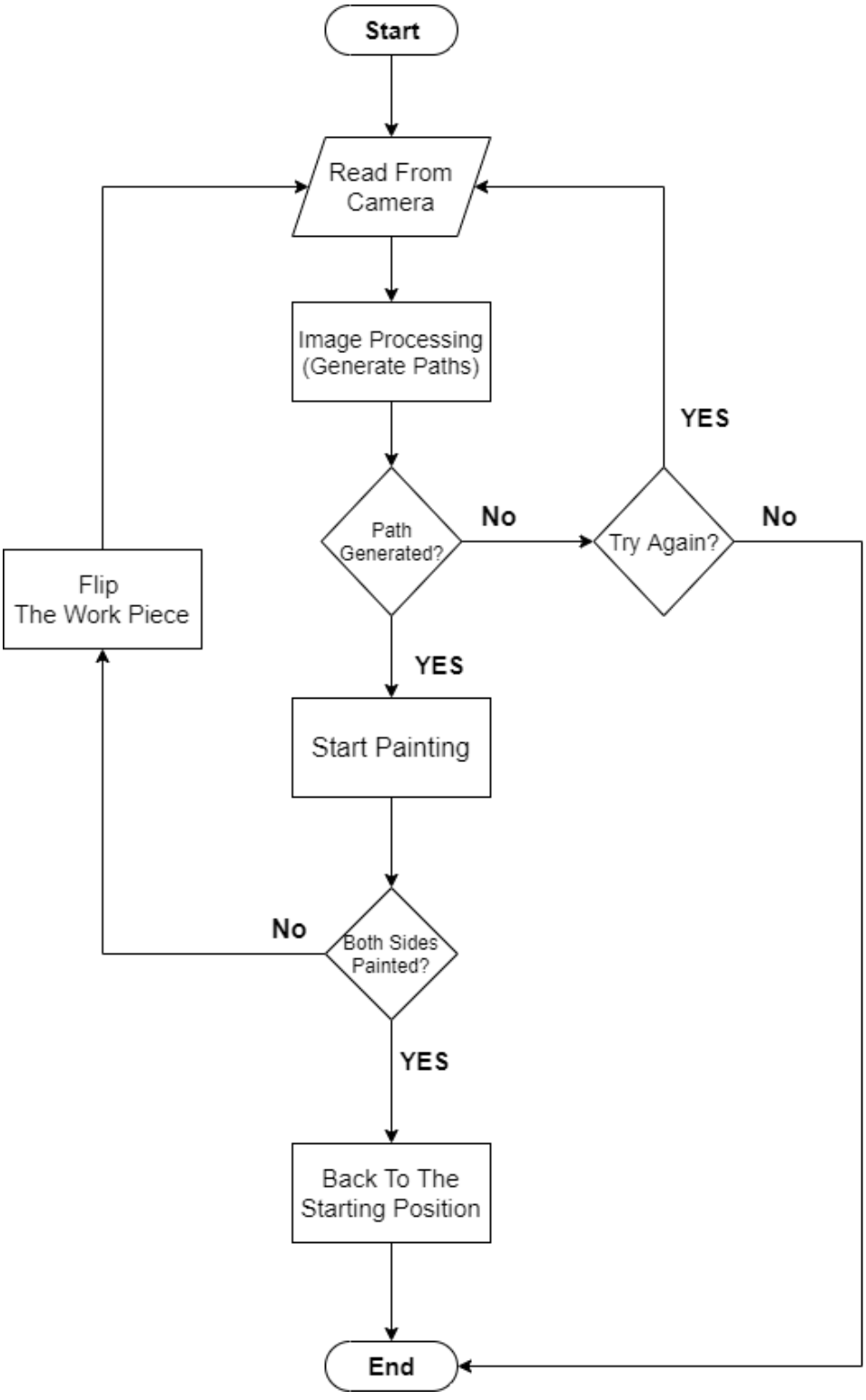


Figure 1-1 General flowchart of the system

1.6 Related systems:

Here we will explain similar systems that gave us the inspiration for the design of our project.

1.6.1 The Core XY 3D Printer:

Figure 1-2: Core XY 3D Printer

Our project is based on this system. The print head is moved using two stationary motors and only two timing belts. Depending on the direction in which the motor rotates, the print head can move in different directions. If only a single motor is activated, the print head moves diagonally. If the print head is to move in either X or Y direction only, the motors must move in opposite directions or together.

1.6.2 The CNC 2D Plotter:

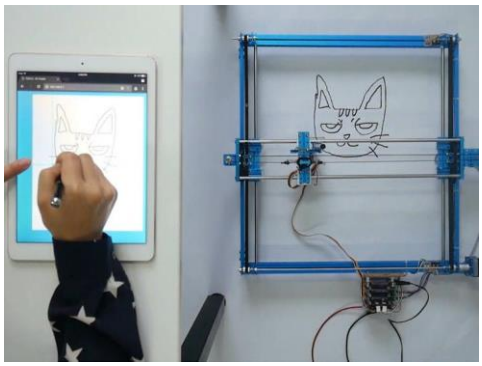


Figure 1-3: The 2D Plotter

The initial design of our project was based on this traditional Cartesian system. The X-axis moves with the plotting head and the Y-axis moves with the plotter bed. That means that the X-axis is moved by the Y-axis motor carrying its motor.

1.7 Budget:

In this section, the costs of the project are determined. This contains the parts we purchased so far.

#	Elements	Quantity	Cost (₪)
1.	GT2 Timing Belts (5 meters)	1	100
1.	GT2 Timing Pulleys.	8	100
2.	Raspberry Pi 4 Kit	1	250
3.	Linear Rails (40 cm)	3	480
4.	Aluminum Extrusions (40 cm)	2 meters	120
5.	3D Printed Parts	6 (bought)	500
6.	Stepper Motor Drivers	2	80
7.	Electronics (Breadboards, Wires, etc)	--	80
		Total	1760

We have provided the stepper motors from an old 3D printer.

1.8 Schedule:

1.8.1 The Tasks to be Done:

Task 1: *Selecting The Idea:*
To determine the idea of the project, the motivation and what to be done.

Task 2: *Collecting The Data:*
To gather the required data for the project, such as open-source designs, codes and information from research papers, books and past graduation projects.

Task 3: *Selecting The Mechanical And Electrical Components:*
To choose the suitable parts for the prototype we will design and begin to provide them.

Task 4: *Mechanical Simulation:*
We will use 3D designing software (e.g. SolidWorks, CATIA) to simulate the assembly of our project prototype.

Task 5: *Dynamic and Mathematical Modeling, System Analysis and Simulation:*
At this stage, the mathematical models that describe the motion of the system should be determined. Once the dynamic modeling is done, we will begin to simulate it on simulation software such as MATLAB, Simulink, etc.

Task 6: *Documentation*:
To document and write down all the steps, modeling and design procedure on this project report.

Task 7: *Assembly and Installation:*
The initial prototype of the system should be built at this stage. It should be done by the end of the first semester.

Task 8: *Testing and Calibration:*
To test the initial modules and designs of the system to make sure we are on the right track of the designing procedure and figure out our errors.

Task 9: *Preparing The First Presentation:*
To prepare the presentation for the discussion of the graduation project introduction course.

Task 10: *Building and Testing the Codes:*
The code of the microcontroller (the Python code) should be finished and tested at this stage.

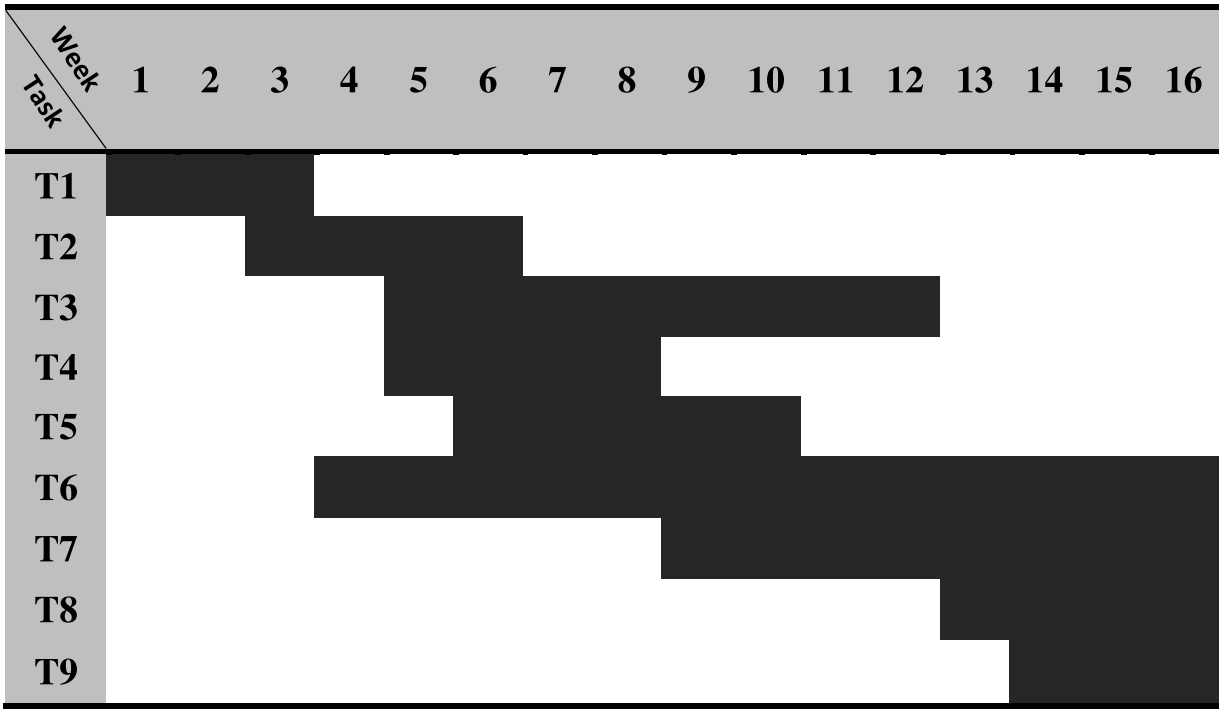
Task 11: *Final implementation and Validation:*
Once the testing and coding is done, the final prototype should be done at this stage.

Task 12: *Writing the Final Report Of the Project:*
The final version of the project report that contains the necessary additions and modifications should be done in the last couple of weeks of the 2nd semester.

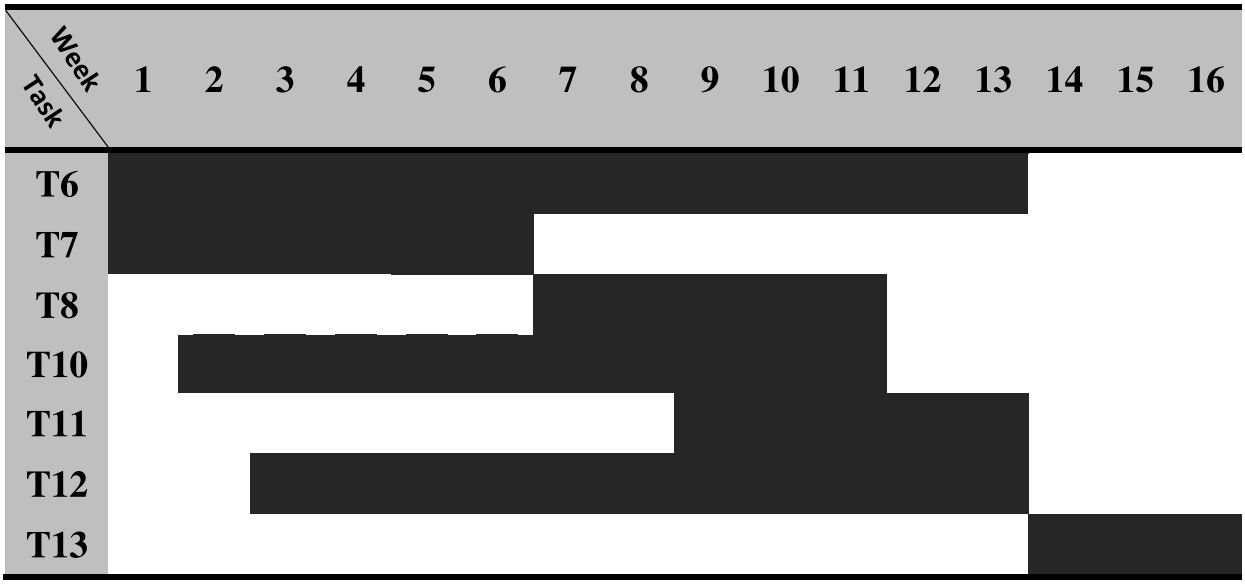
Task 13: *Preparing the Final Presentation:*
To prepare the final presentation of the project and get ready for the final discussion of the graduation project.

1.8.2 Semester plans:

First Semester:



Second Semester:



Chapter 2 :
Conceptual Design

2.1 Introduction:

In this chapter we will explain the specifications of our selection of the parts for the prototype of the project and the designs of the components. We will also show a number of drawings and initial models for the system.

2.2 The Mechanical Frame:

For the mechanical frame of this system, we have 3 options:

A. The H-Bot System: This is the frame we decided for our project. The painter head moves in two directions (X & Y). The motion is controlled by stationary 2 stepper motors. The whole system is connected with only one timing belt. The motion of the painter head is controlled by the direction of the rotation of each of the two stepper motors. Figure 2-1 shows the mechanism:

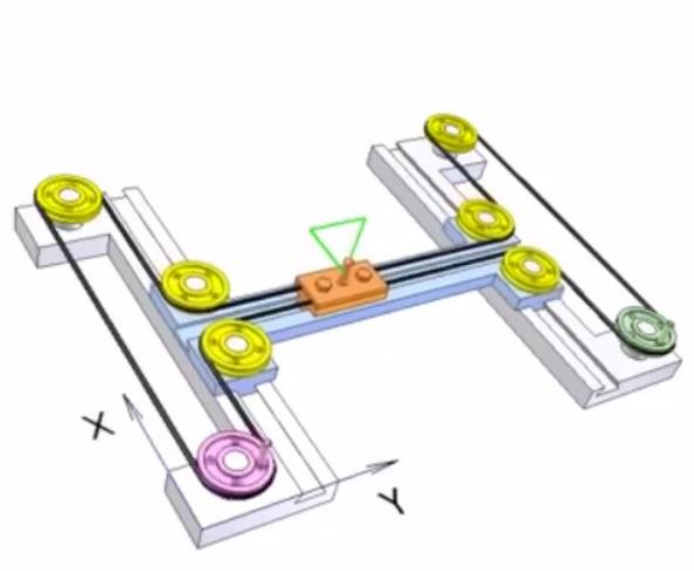


Figure 2-1: H-Bot Mechanism

There are four scenarios which describe its movement. It is important to know that in these scenarios both motors are rotating at **the same speed.**

- **The first scenario** is translation in the negative Y axis which can be generated by rotating T1 counter-clockwise and T2 clockwise.

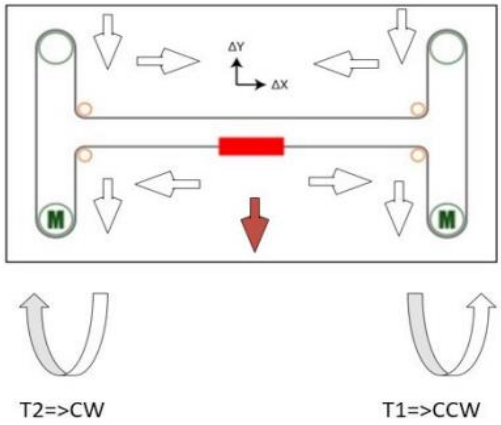


Figure 2-2: H-Bot -Y translation

- **The second scenario** is translation in the positive X-axis which can be generated by rotating both T1 and T2 counter-clockwise.

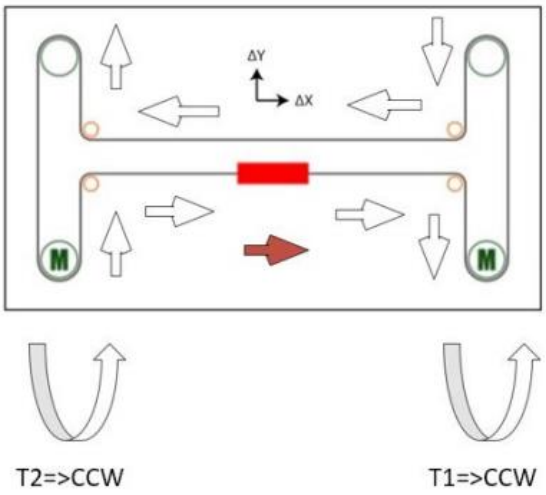


Figure 2-3: H-Bot +X translation

- **The third one** is translation in the negative X axis which can be generated by rotating both T1 and T2 clockwise.

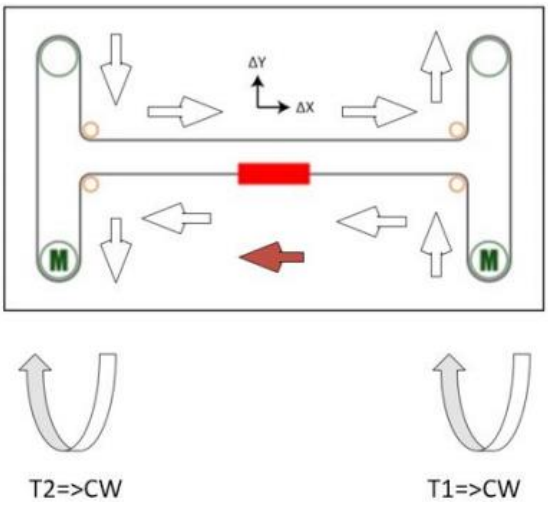


Figure 2-4: H-bot -X translation

- **The forth scenario** is translation in the positive Y axis which can be generated by rotating T1 clock wise and T2 counter-clock wise. For combined movement in both X and Y axes.

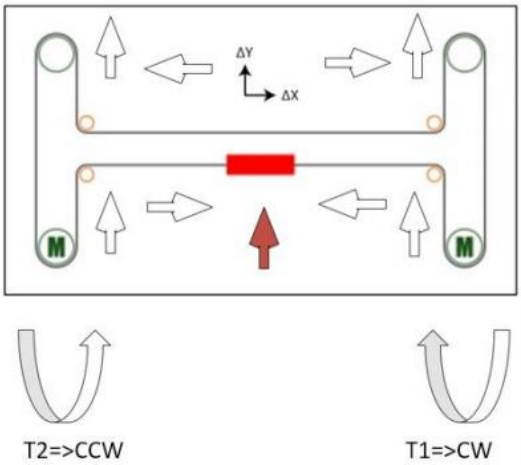


Figure 2-5: H-Bot +Y translation

More to mention, if only one of the motors rotates, this causes the end-effector to move diagonally.

This table describes the kinematics and the motion of the H-Bot system:

T_1	T_2	f_x	f_y	Movement
CCW	CW	0	-	-Y
CCW	CCW	+	0	+X
CW	CW	-	0	-X
CW	CCW	0	+	+Y
CCW	0	+	-	+X & -Y
CW	0	-	+	-X & +Y
0	CCW	+	+	+X & +Y
0	CW	-	-	-X & -Y

Figure 2-6: Table of H-Bot kinematics

B. The Mid-T Model: [6] An alternative model. The system has 2 motors that control the X and Y axes with only one timing belt. The model looks like a cross. The painter head is moved in two directions according to the rotation of each of the two steppers motors, see the figures below:

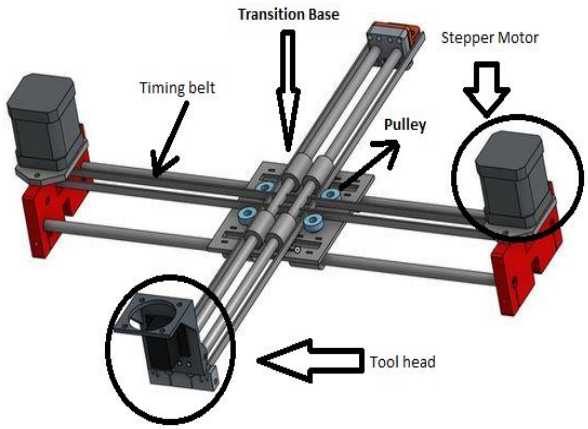


Figure 2-7: The Mid-T Model

There are four scenarios which describe the movement of this system movement:

- **The first scenario** is the translation in the positive Y-axis direction. The motors rotate in opposite directions. The left motor rotates clockwise. The right rotates counter-clockwise.
- **The second scenario** is the translation in the positive X-axis. Both motors rotate in counter-clockwise direction.
- **The third** is the translation in the negative Y-axis. The left motor rotates counter-clockwise, while the right rotates clockwise.
- **The fourth** is the translation in the negative X-axis. Both motors rotate clockwise.

The kinematics of the system is shown in the figure and the table below:

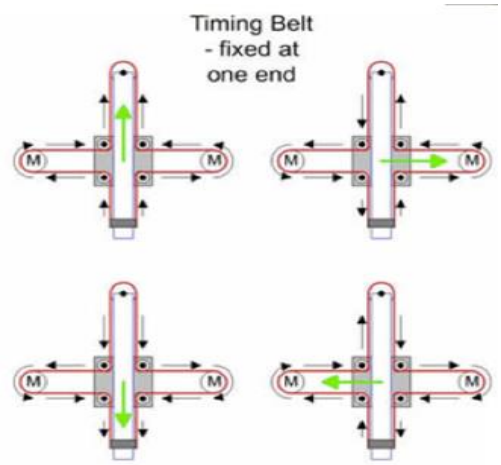


Figure 2-8: Mid-T Kinematics

M (L)	M (R)	F_X	F_Y	Movement
CW	CCW	0	+	+Y
CCW	CCW	+	0	+X
CCW	CW	0	-	-Y
CW	CW	-	0	-X

Figure 2-9: Table of Mid-T Kinematics

C. The Core XY positioning system [5]: This system is very similar to the H-Frame. It has two stationary motors and two timing belts. Each timing belt is connected to one of the two motors. However, the requirement of two belts on separate planes is a disadvantage of the system, making the system more complex to model. However, this system is becoming popular in different applications, most commonly, the 3D printers. The reason is that an accuracy in positioning in x and y directions can be achieved if modeled and configured well. The dynamic modeling of the system may follow the same techniques as the H-Frame; however there may be more mathematics to be done. Due to its similarity to the H-framed, we can use this as an alternative solution.

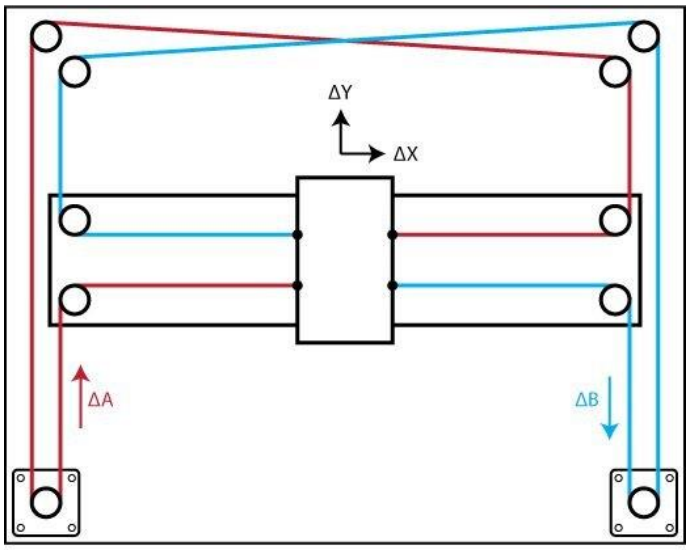


Figure 2-10: The Core XY mechanism

The general kinematics of the system is the same as those of the H-Bot frame. The kinematics can be described as follows:

- Both motors rotate clockwise, the end-effector moves left (-X).
- Both motors rotate counter-clockwise, the end-effector moves right (+X).
- The left motor rotates counter-clockwise and the right rotates clockwise, the end-effector moves forwards (+Y).
- The left motor rotates clockwise and the right moves counter-clockwise, the end-effector moves backwards (-Y).
- Only one motor rotates, the end-effector moves diagonally.

The kinematics of the Core-XY is shown in the figure and the table below:

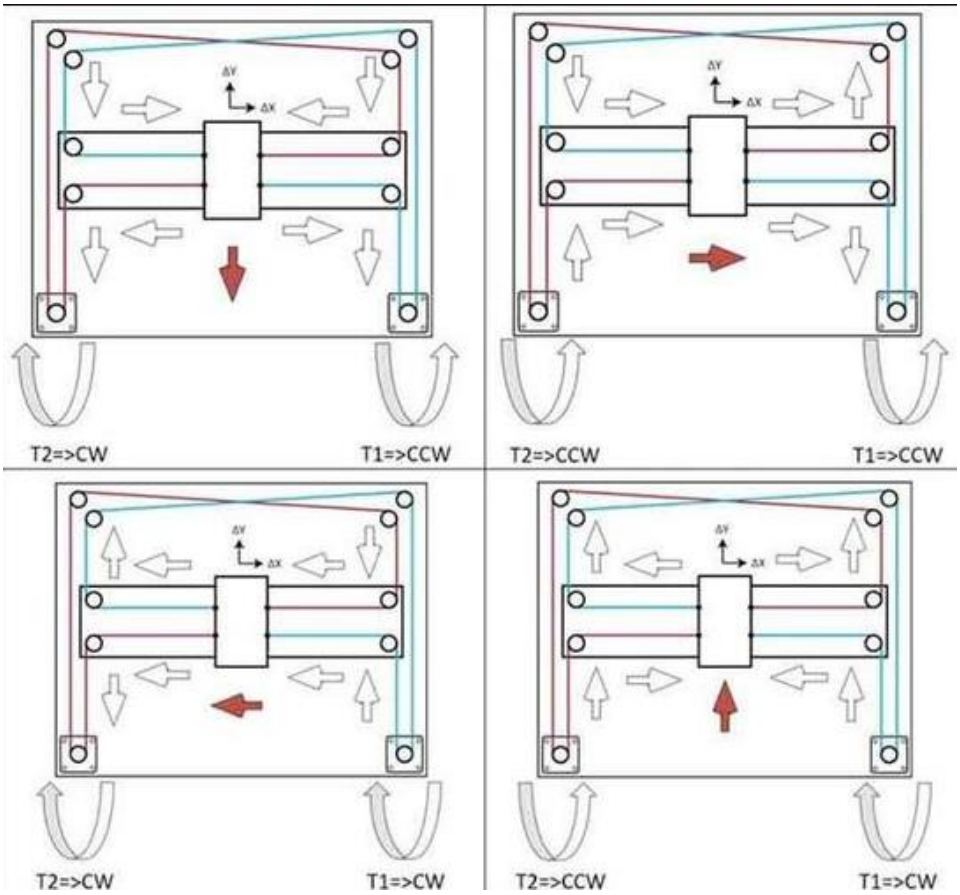


Figure 2-11: Core XY Kinematics

T_1	T_2	f_x	f_y	Movement
CCW	**CW**	0	-	-Y
CCW	**CCW**	+	0	+X
CW	**CW**	-	0	-X
CW	**CCW**	0	+	+Y
CCW	**0**	+	-	+X & -Y
CW	**0**	-	+	-X & +Y
0	**CCW**	+	+	+X & +Y
0	**CW**	-	-	-X & -Y

Figure 2-12: Core XY kinematics table

2.3 Static Analysis and Mechanical Design:

We used the SolidWorks software to simulate the mechanical frame of the system and determine its mechanical characteristics, i.e. stress, strain and displacement (deformation).

This figure 2-13 shows the 3D mechanical design:

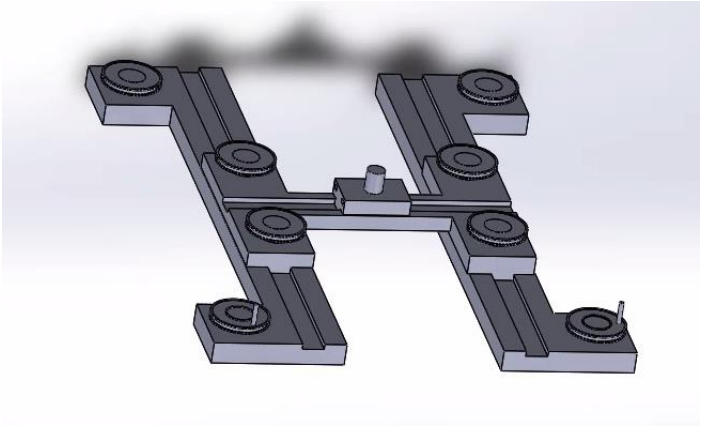


Figure 2-13: The Mechanical Frame on SolidWorks

The material we selected for the frame is steel; thus, the calculations are based on this material. We selected a force of 100 N on the bridge parts as the maximum force this frame can handle. The yield strength of the frame material is 710 MPa, whereas the maximum stress is about 6.741 MPa, as shown in the figure 2-14:

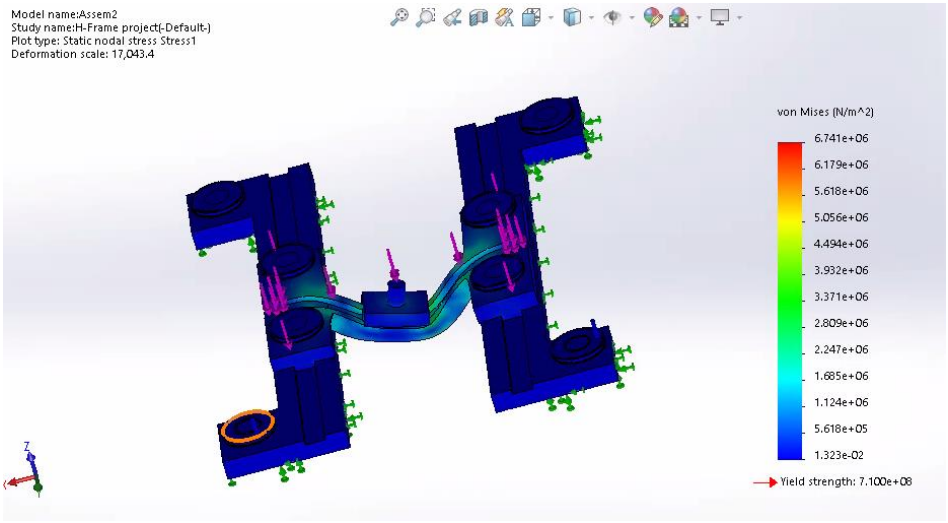


Figure 2-14: Stress Analysis of The Frame

The maximum strain of the frame with the applied force is 14.93×10^{-6}, as shown in the figure

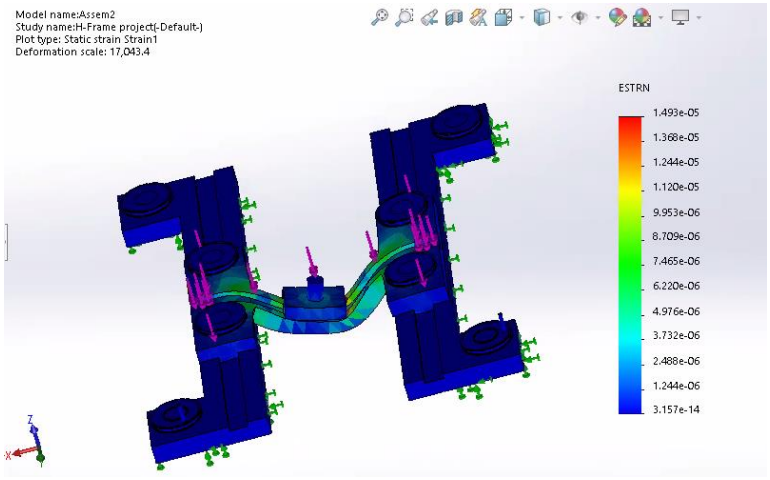


Figure 2-15: The Strain Analysis of the Frame

The maximum deflection (displacement) caused by the 100 N force is approximately 1.475×10^{-3} mm as shown in the figure 2-16:

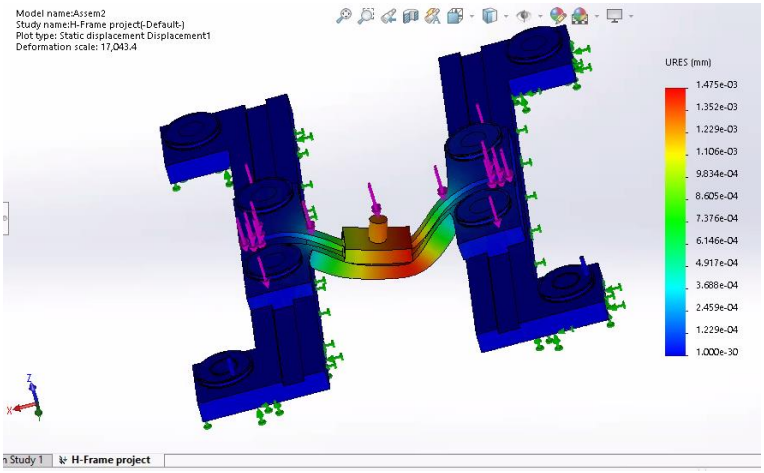


Figure 2-16: Maximum Deflection

The results showed that the frame is not affected by the 100N force, which means that the system should be able handle large forces without any deformation, as desired.

2.4 The Hardware Configuration:

2.4.1 Hardware:

1. **Microcontroller**: We will use Raspberry Pi controller for this project, because it can perform complex image processing operations and features an open source operating system, making it easier to control and program.
2. **Stepper motors:** For the prototype**,** Nema 17 Stepper motors will be used to control the motion of the painter head (nozzle) on the (X-Y) Frame and the flipping mechanism. But for full scale realization it is recommended to use servomotors with feedback loop in the controller to achieve disturbance rejection and reduce system sensitivity to plant variations
3. **Driver circuit for stepper motors:** For stepper motor control.
4. **Sensors:** We will use a camera (only takes pictures).
5. **Paining System:** For our prototype, we may use a solenoid based pointing system instead of the nozzle system, since our main focus in the project is going to be the positioning mechanism.
6. **X and Y Frame (H-Model)**

2.4.2 Software:

1. **Python:** The Python language has a large community and thousands of libraries and packages, so it will be easier for us to build the best codes and software possible.
2. **MATLAB:** To determine the dynamic models, design and simulate the control system.
3. **3D Design Software:** We will use SolidWorks to design and simulate the mechanical structure of the system.
4. **Simulation software for the electrical system:** E.g. Proteus.

2.4.3 Conceptual Block Diagram of the Hardware System:

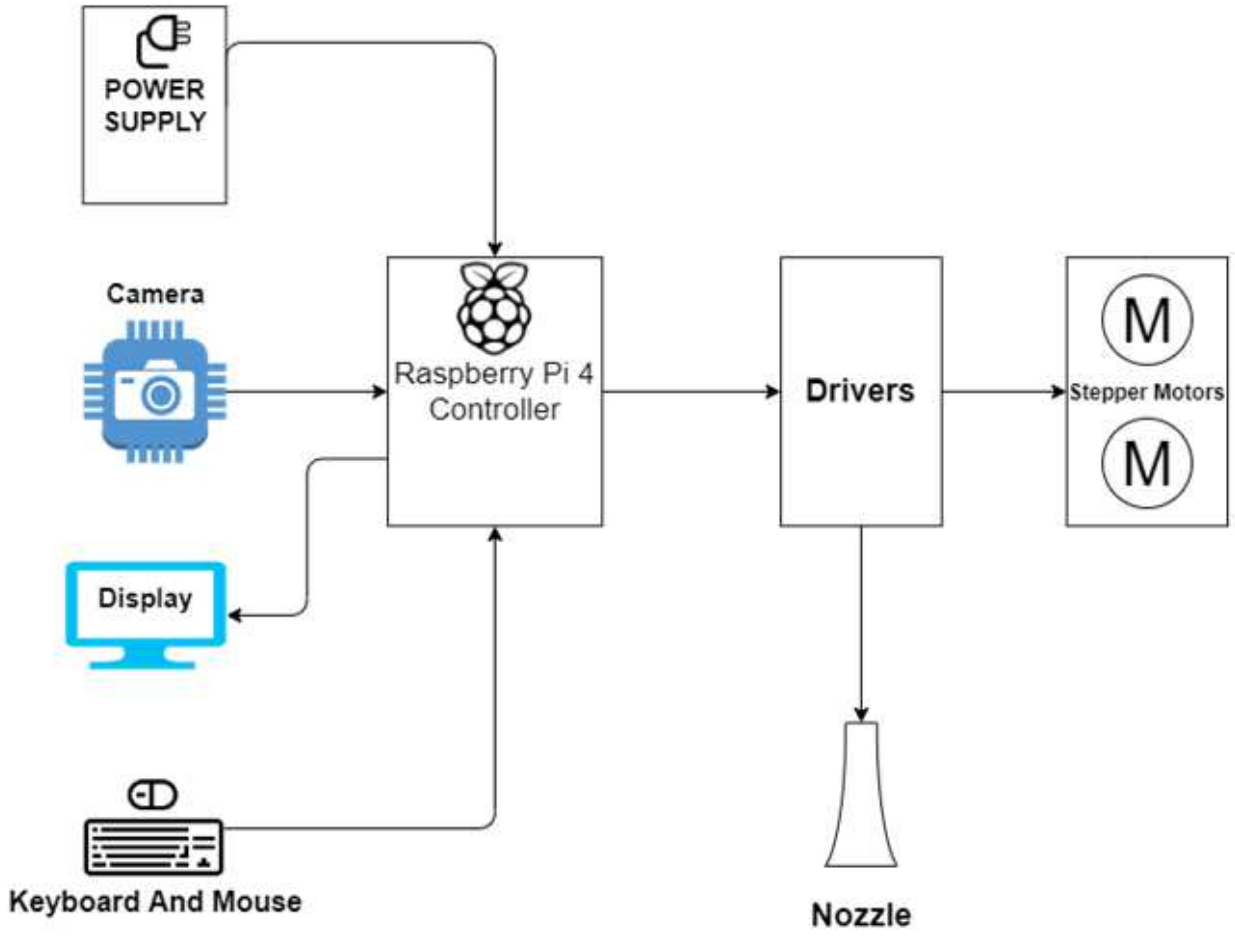


Figure 2-17: The conceptual diagram of the system

Chapter 3 :
Mathematical Modeling and Control

The kinematics and the dynamics of the system will be determined in this chapter. It will be divided into three sections; general kinematics, dynamic analysis and the control system.

3.1 General Kinematics of the H-Bot Mechanism:

The relationship between the angles of the two motors and the x-y position of the end-effector will be described in this section. The coordinate and the axis of the system are described in this figure:

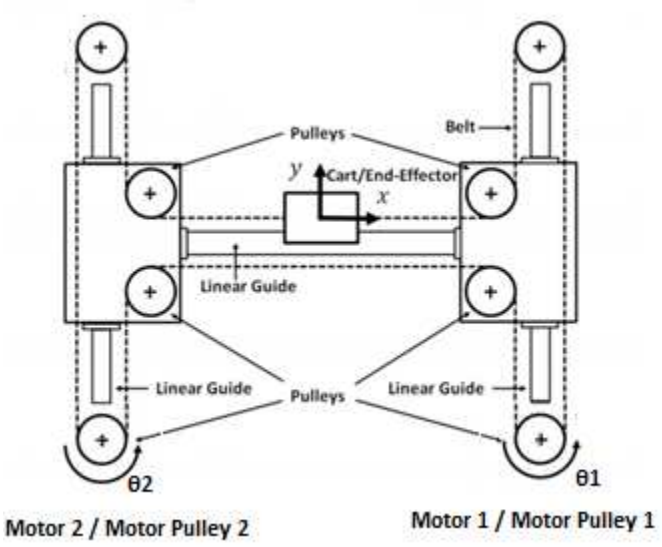


Figure 3-1: Coordinate and the axis of the H-Bot

Based on the configuration of the system, as described in chapter 2, turning only one motor causes the end-effector to move in a diagonal ($\pm 45^{o}$ angle) direction. Turning the two motors in a positive direction (counter-clockwise) causes the end-effector to move along the positive x-axis, while rotating motor 2 in a positive (CCW) direction and motor 1 in a negative direction (CW) causes the end-effector to move along the positive Y-axis. Therefore, by the varying the amount of rotation of each motor, a motion in any x-y direction can be generated. [3]

The relationship between the angles and the x-y position of the end-effector can be described with the kinematics, as shown on equation (1):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{r}{2}\begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad (1)$$

Where x and y are the position of the end-effector and θ_1 and θ_2 are the rotation angles of the motors 1 and 2, respectively, r is the radius of the pulley.

From equation (1), the position of the end-effector in each of the x and y directions are described in the equations (2) and (3):

$$x = -\frac{r}{2}(\theta_1 + \theta_2) \quad (2)$$

$$y = \frac{r}{2}(\theta_1 - \theta_2) \quad (3)$$

By taking the derivative with respect to time for each of equations [2] and [3], we get the velocity of the motion of the end-effector, as shown in equations [4] and [5]:

$$\dot{x} = -\frac{r}{2}(\dot{\theta}_1 + \dot{\theta}_2) \quad (4)$$

$$\dot{y} = \frac{r}{2}(\dot{\theta}_1 - \dot{\theta}_2) \quad (5)$$

3.2 The Dynamic Analysis of the System:

To find the required torque needed to move the end-effector for each of the two motors, we will use the Lagrangian mechanics. Firstly, equations of motion can be determined by the Euler-Lagrange equation:

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} + \frac{\partial R}{\partial \dot{\theta}} \quad (6)$$

Where $\boldsymbol{\tau}$ is the generalized force (torque), R is the frictional energy dissipation and L is the Lagrangian.

The Lagrangian is the difference between the kinetic energy and the potential energy, as shown in equation [7]:

$$L = T - U \tag{7}$$

Where T is the kinetic energy and U is the potential energy. The system is not affected by any potential energy; hence the potential energy term U is eliminated, therefore:

$$L = T \tag{7'}$$

Now that we are done defining the general equations for the system, let us define the total kinetic energy of the system:

$$\sum T = T_{pulley} + T_{\substack{motor \\ Shaft}} + T_{cart} + T_{Bridge} \tag{8}$$

To define the kinetic energy for each element:

$$T_{pulley} = \frac{1}{2} J_{p1} \dot{\theta}_1^2 + \frac{1}{2} J_{p2} \dot{\theta}_2^2 \tag{9}$$

$$T_{\substack{motor \\ shaft}} = \frac{1}{2} J_{m1} \dot{\theta}_1^2 + \frac{1}{2} J_{m2} \dot{\theta}_2^2 \tag{10}$$

$$T_{cart} = \frac{1}{2} M_C \, (\dot{x}^2 + \dot{y}^2) \tag{11}$$

$$T_{Bridge} = \frac{1}{2} M_B \, (\dot{y}^2) \tag{12}$$

Where J_p is the mass moment of inertia of the pulley and J_m is the mass moment of inertia of the motor shaft. Since J_{p1} and J_{p2} are equal:

$$T_{pulley} = \frac{1}{2} J_p (\dot{\theta}_1^2 + \dot{\theta}_2^2) \tag{9'}$$

Also, since J_{m1} and J_{m2} are equal:

$$T_{\substack{motor \\ shaft}} = \frac{1}{2} J_m (\dot{\theta}_1^2 + \dot{\theta}_2^2) \tag{10'}$$

By substituting equations (4) and (5) in (11) and (12):

$$T_{cart} = \frac{1}{4} M_C \, r^2 (\dot{\theta}_1^2 + \dot{\theta}_2^2) \tag{11'}$$

$$T_{Bridge} = \frac{1}{8} M_B \, r^2 (\dot{\theta}_1^2 - 2\dot{\theta}_1 \dot{\theta}_2 + \dot{\theta}_2^2) \tag{12'}$$

Substituting the K.E. equations (9'),(10'),(11') and (12') in equation (8) gives the total kinetic energy:

$$\sum T = \frac{1}{2}J_p\left(\dot{\theta}_1^2 + \dot{\theta}_2^2\right) + \frac{1}{2}J_m\left(\dot{\theta}_1^2 + \dot{\theta}_2^2\right) + \frac{1}{4}M_C\, r^2\left(\dot{\theta}_1^2 + \dot{\theta}_2^2\right) + \frac{1}{8}M_B\, r^2(\dot{\theta}_1^2 - 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \quad (8)$$

Simplifying equation (8):

$$\sum T = \frac{1}{2}\left(J_p + J_m + \frac{1}{2}M_C\, r^2\right)\left(\dot{\theta}_1^2 + \dot{\theta}_2^2\right) + \frac{1}{8}M_B\, r^2(\dot{\theta}_1^2 - 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) \quad (8')$$

To find the simplified equation of the frictional energy dissipation:

$$\sum R = \frac{1}{2}(C_p + \frac{1}{2}C_c r^2)(\dot{\theta_1^2} + \dot{\theta_2^2}) + \frac{1}{8}C_B r^2(\dot{\theta}_1^2 - 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)$$

Where C_p, C_c, C_B are the viscous friction parameters of the motors, the cart and the bridge. Now, the equations of motion for each motor are derived from the Euler-Lagrange equation (6):

$$\tau_1 = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_1}\right) - \frac{\partial T}{\partial \theta_1} + \frac{\partial R}{\partial \dot{\theta}_1} \quad (6.1.)$$

$$\tau_2 = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_2}\right) - \frac{\partial T}{\partial \theta_2} + \frac{\partial R}{\partial \dot{\theta}_2} \quad (6.2.)$$

Starting from the term $\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_\iota}\right)$:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_1}\right) = \left(J_p + J_m + \frac{1}{2}M_C\, r^2\right)\left(\ddot{\theta}_1\right) + \frac{1}{4}M_B\, r^2\left(\ddot{\theta}_1 - \ddot{\theta}_2\right) \quad (6.1.1.)$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_2}\right) = \left(J_p + J_m + \frac{1}{2}M_C\, r^2\right)\left(\ddot{\theta}_2\right) + \frac{1}{4}M_B\, r^2\left(\ddot{\theta}_2 - \ddot{\theta}_1\right) \quad (6.2.2.)$$

In the equations of motion of the system, we do not take the stiffness of the system into consideration, hence the term$\left(\frac{\partial T}{\partial \theta_i}\right)$ is eliminated. Next, the term$\left(\frac{\partial R}{\partial \dot{\theta}_1}\right)$:

$$\frac{\partial R}{\partial \dot{\theta}_1} = (C_p + \frac{1}{2}C_c r^2)(\dot{\theta}_1) + \frac{1}{4}C_B r^2(\dot{\theta}_1 - \dot{\theta}_2) \quad (6.1.2.)$$

$$\frac{\partial R}{\partial \dot{\theta}_2} = (C_p + \frac{1}{2}C_c^2)(\dot{\theta}_2) + \frac{1}{4}C_B r^2(\dot{\theta}_2 - \dot{\theta}_1) \quad (6.2.2.)$$

Putting the terms together gives us the equations of motion (6.1.') and (6.2.'):

$$\tau_1 = (J_p + J_m + \frac{1}{2}M_C\, r^2)(\ddot{\theta}_1) + \frac{1}{4}M_B\, r^2(\ddot{\theta}_1 - \ddot{\theta}_2) + (C_p + \frac{1}{2}C_c)(\dot{\theta}_1) + \frac{1}{4}C_B r^2(\dot{\theta}_1 - \dot{\theta}_2) \quad (6.1.')$$

$$\tau_2 = (J_p + J_m + \frac{1}{2}M_C\, r^2)(\ddot{\theta}_2) + \frac{1}{4}M_B\, r^2(\ddot{\theta}_2 - \ddot{\theta}_1) + (C_p + \frac{1}{2}C_c)(\dot{\theta}_2) + \frac{1}{4}C_B r^2(\dot{\theta}_2 - \dot{\theta}_1) \quad (6.2.')$$

3.3 The State-Space Model and the Control System:

3.3.1 The State-Space Model:

The previous set of equations can be transferred into state-space form as shown in equations (13)-(18):

$$\begin{aligned} &x_1 = \theta_1 \qquad x_2 = \dot{x}_1 \qquad x_3 = \theta_2 \qquad x_4 = \dot{x}_3 \\ &x_2 = \dot{\theta}_1 \qquad\qquad x_4 = \dot{\theta}_2 \end{aligned} \quad (13)$$

$$y_1 = \theta_1 \qquad y_2 = \theta_2$$

Substituting the variables into the two equations (6.1.') and (6.1.'):

$$\tau_1 = (J_p + J_m + \frac{1}{2}M_C\, r^2)(\dot{x_2}) + \frac{1}{4}M_B\, r^2(\dot{x_2} - \dot{x_4}) + (C_p + \frac{1}{2}C_c)(x_2) + \frac{1}{4}C_B r^2(x_2 - x_4) \quad (6.1.')$$

$$\tau_2 = (J_p + J_m + \frac{1}{2}M_C\, r^2)(\dot{x_4}) + \frac{1}{4}M_B\, r^2(\dot{x_4} - \dot{x_2}) + (C_p + \frac{1}{2}C_c)(x_4) + \frac{1}{4}C_B r^2(x_4 - x_2) \quad (6.2.')$$

Note that the each of the two equations of motion has two derivative variables. Therefore, the derivative terms can be put together in one matrix ($\vec{E}$) and the state terms can be put together in another matrix ($\vec{B}$):

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a_1 + a_2 & 0 & -a_2 \\ 0 & 0 & 1 & 0 \\ 0 & -a_2 & 0 & a_1 + a_2 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & b_1 + b_2 & 0 & -b_2 \\ 0 & 0 & 0 & 1 \\ 0 & -b_2 & 0 & b_1 + b_2 \end{bmatrix} \quad (13)$$

Where:

$$a_1 = (J_p + J_m + \frac{1}{2} M_C\, r^2) \qquad a_2 = (\frac{1}{4} M_B\, r^2)$$

$$b_1 = (C_p + \frac{1}{2} C_c r^2) \qquad b_2 = \frac{1}{4} C_B r^2$$

The values we selected for these parameters are based on values we got from one of the sources we found on this type of mechanism [4].

For the input (torque) matrix **G**:

$$\boldsymbol{G} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{14}$$

The output matrix:

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The feedthrough matrix: (15)

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

To find the state equations:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}.\boldsymbol{x} + \boldsymbol{B}.\boldsymbol{u} \tag{16}$$

$$\boldsymbol{y} = \boldsymbol{C}.\boldsymbol{x} + \boldsymbol{D} \tag{17}$$

Where:

$$\boldsymbol{A} = \boldsymbol{E}^{-1}\boldsymbol{Q} \qquad \boldsymbol{B} = \boldsymbol{E}^{-1}\boldsymbol{G}$$

3.3.2 The Control System:

We used the MATLAB software to design 4 different types of controllers; an analog tracker, an analog robust tracker, a discrete-time tracker and a discrete time robust tracker, based on the mathematical model we determined in section 3.3.1. here the desired response specifications to achieve the closed loop eigenvalues are $\omega_n \approx 10$ rad/s and $\zeta \geq 0.80$.

Simulink Model for the analog tracking system:

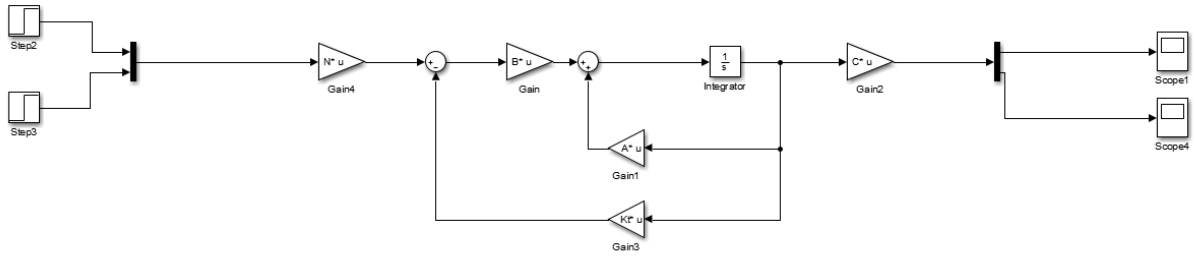


Figure 3-2: Simulink model of the analog tracking controller

The figures below show the response of the system to a step inputs for the two actuators:

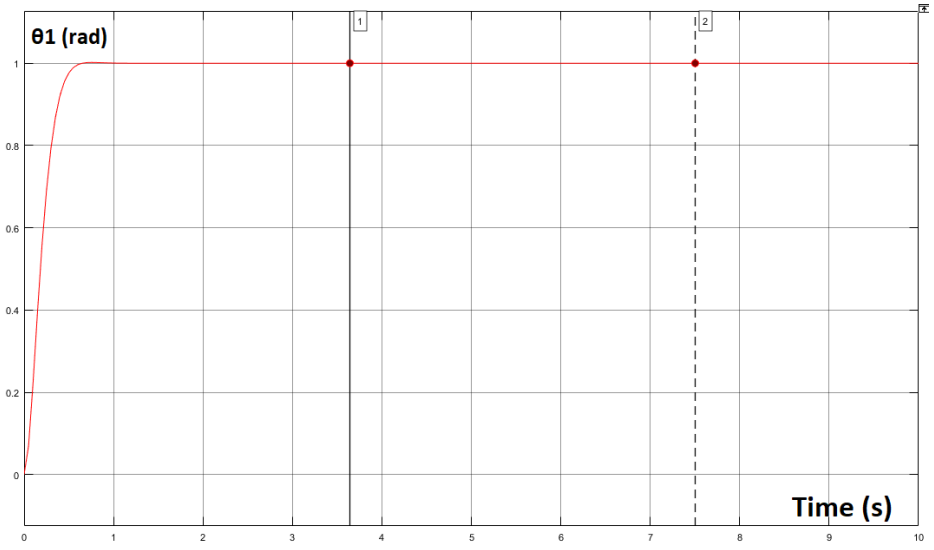


Figure 3-3: Step response of θ1 with analog tracking

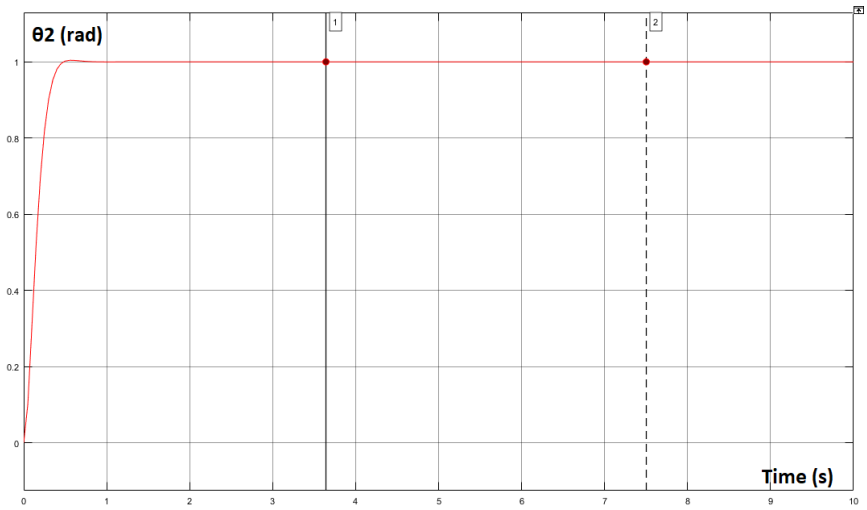


Figure 3-4: Step response of θ2 with analog tracking

The settling time of the system is a little less than 1 second and the overshoot is about 0.5% for each of θ_1 and θ_2.

Now, the input for each of θ_1 and θ_2 is as follows:

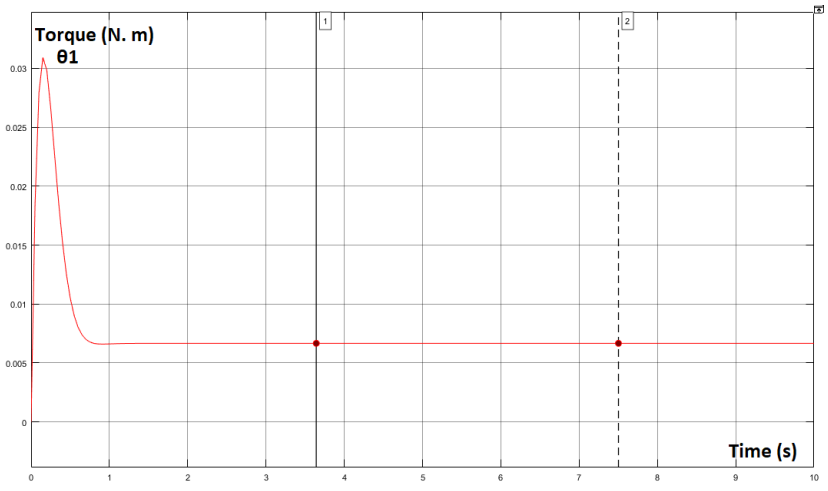


Figure 3-5: Input for θ1 with analog tracking [N.m]

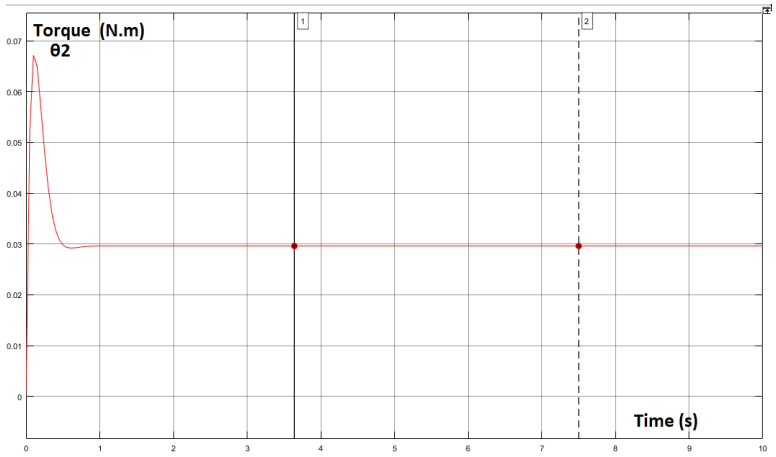


Figure 3-6: Input of θ2 with analog tracking [N.m]

The Simulink model for the analog robust tracker is as follows:

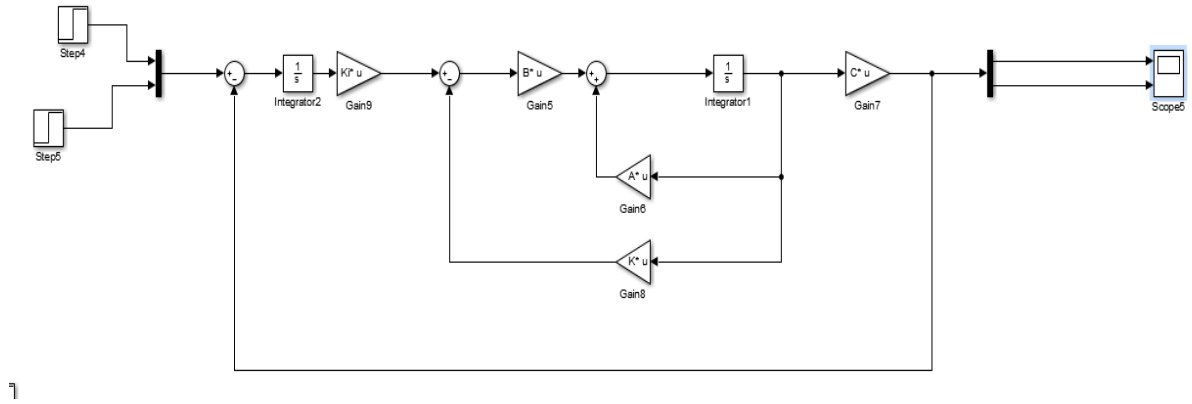


Figure 3-7: Simulink model of the system with the robust tracking controller

For a step input, the settling time was about 1second and the overshoot was about 0.5% for each of θ_1 and θ_2.

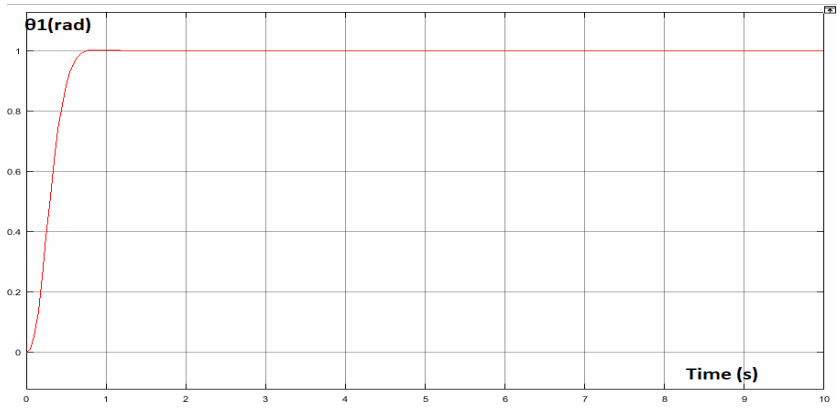


Figure 3-8: Step response of θ1 with analog robust tracking

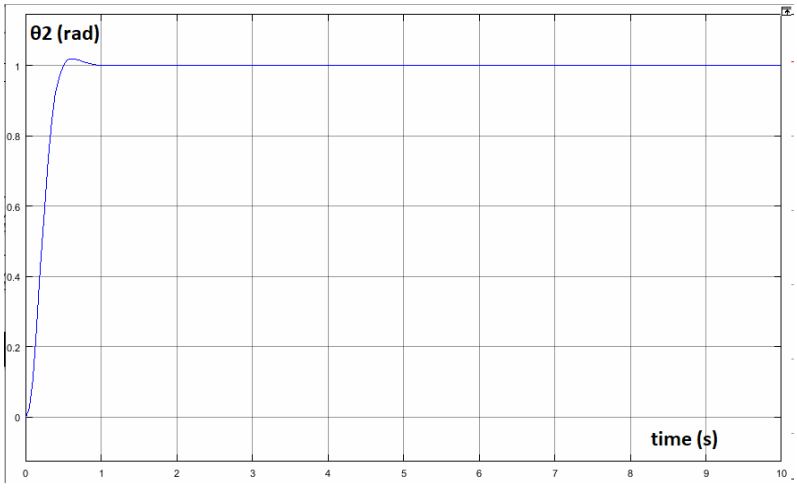


Figure 3-9: Step response of θ2 with analog robust tracking

For the input of the system with the robust tracking controller:

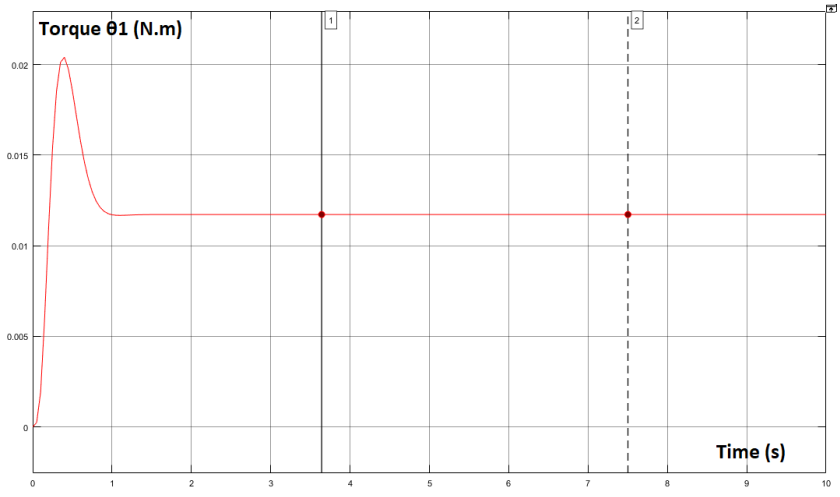


Figure 3-10: input of θ1 with analog robust tracking [N.m]

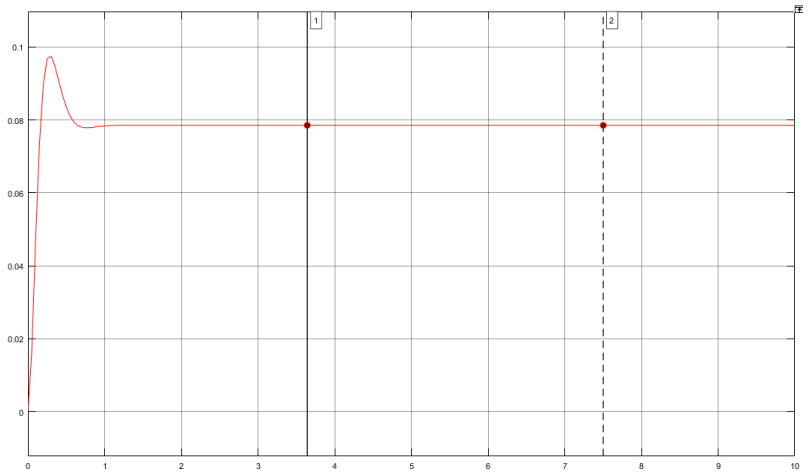


Figure 3-11: Input of θ2 with analog robust tracking [N.m]

Here is the Simulink model of the **discrete-time tracking** controller:

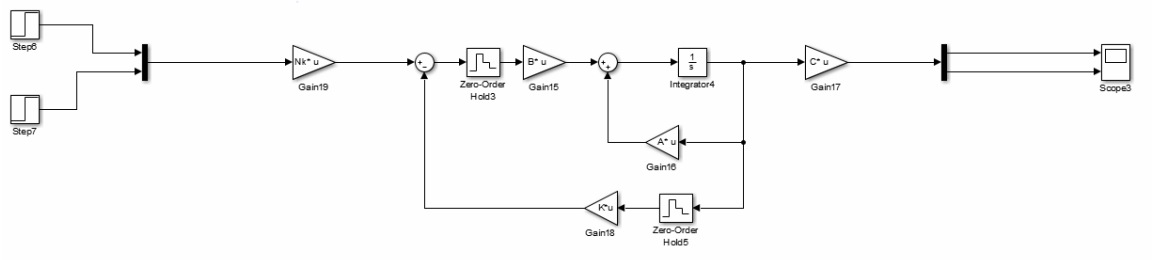


Figure 3-12: Simulink model of the system with discrete-time tracking

The step input for each of θ_1 and θ_2 gave a response of about 3% overshoot and the settling time was about 1 second.

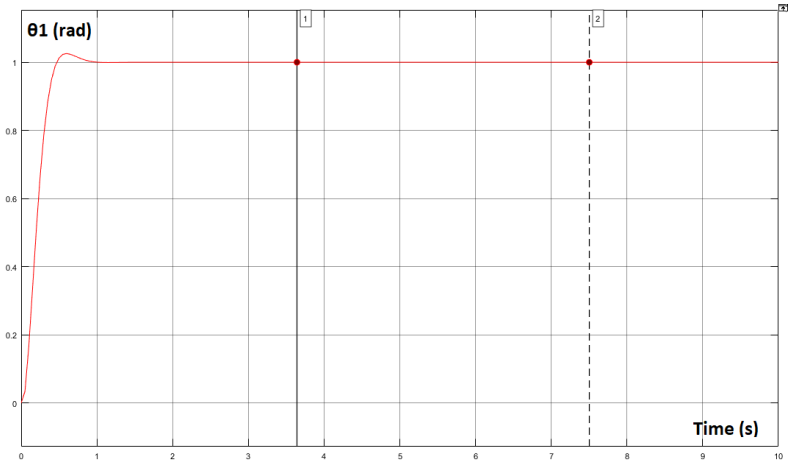


Figure 3-13: Step response of θ1 with discrete-time tracking

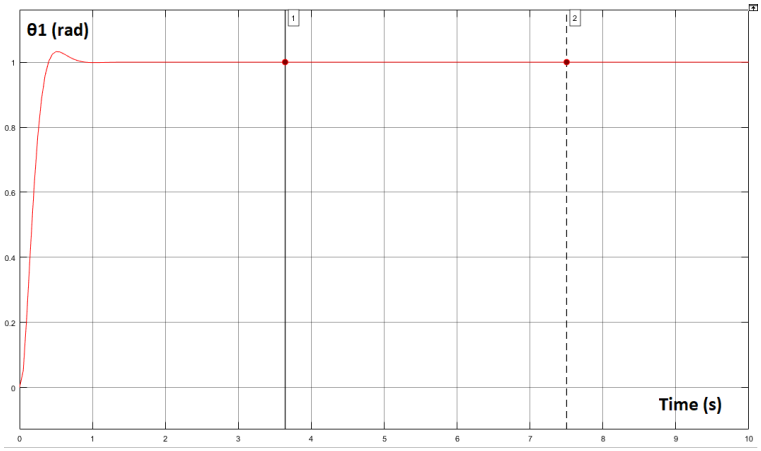


Figure 3-14: Step response of θ2 with discrete-time tracking

For the input of each of θ_1 and θ_2:

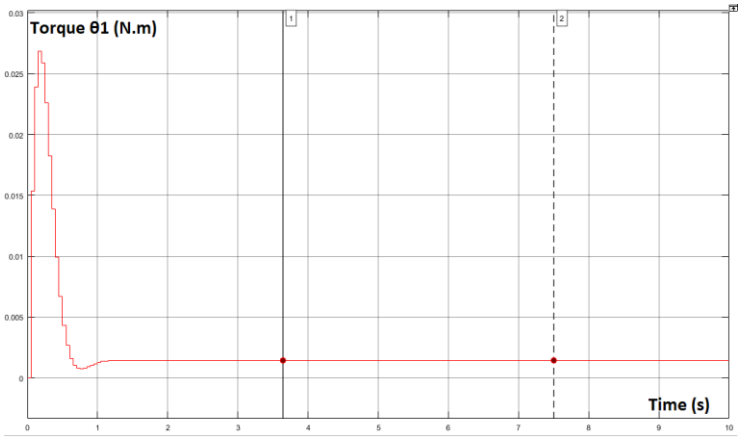


Figure 3-15: Input of θ1 with discrete-time tracking [N.m]

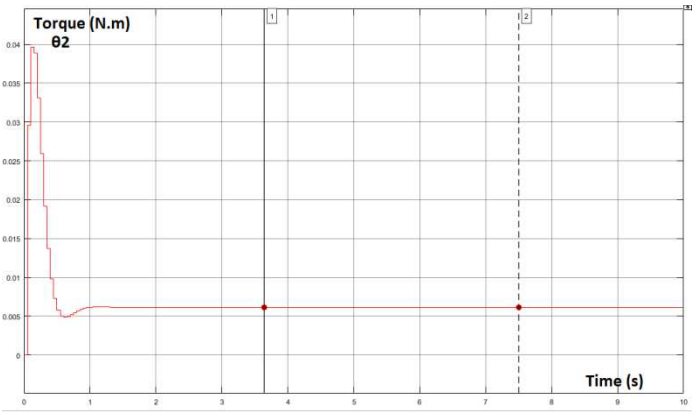


Figure 3-16: Input of θ2 with discrete-time tracking [N.m]

And finally, the discrete-time robust tracking controller:

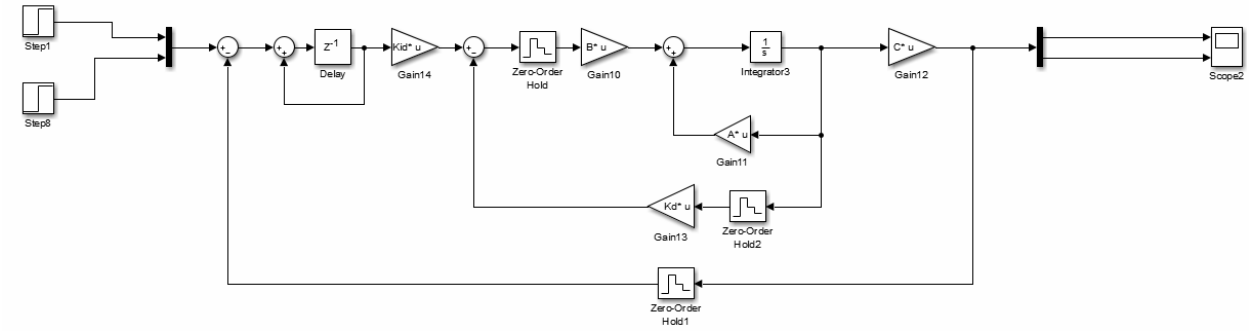


Figure 3-17: Simulink model of the system with discrete-time robust tracking

For a step input, the settling time was about 1second for θ_1 and 1.3 seconds for θ_2 and the overshoot was about 4% for θ_1 and 0.5% for θ_2.

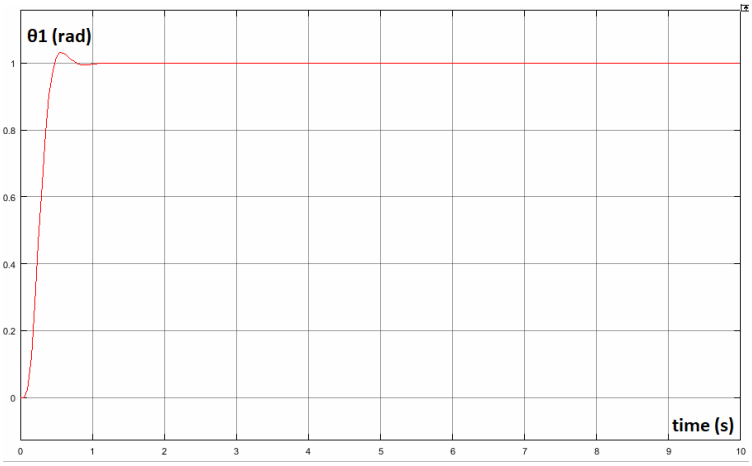


Figure 3-18: Step response of θ1 with discrete-time robust tracking

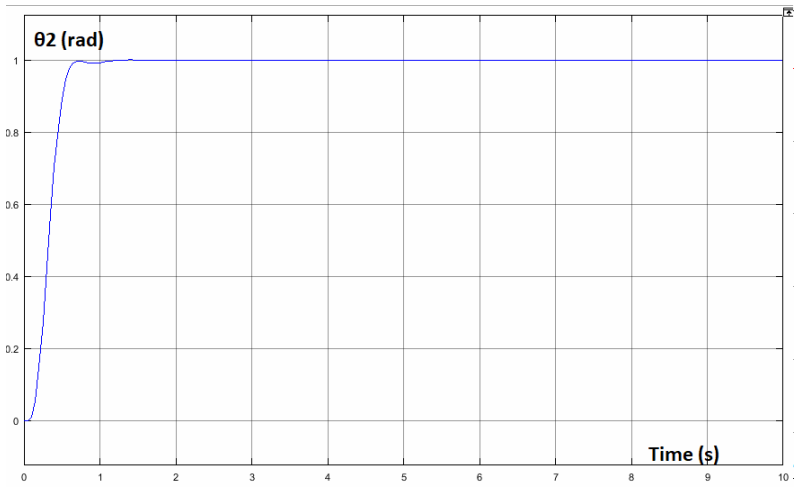


Figure 3-19: Step response of θ2 with discrete-time robust tracking

For the input:

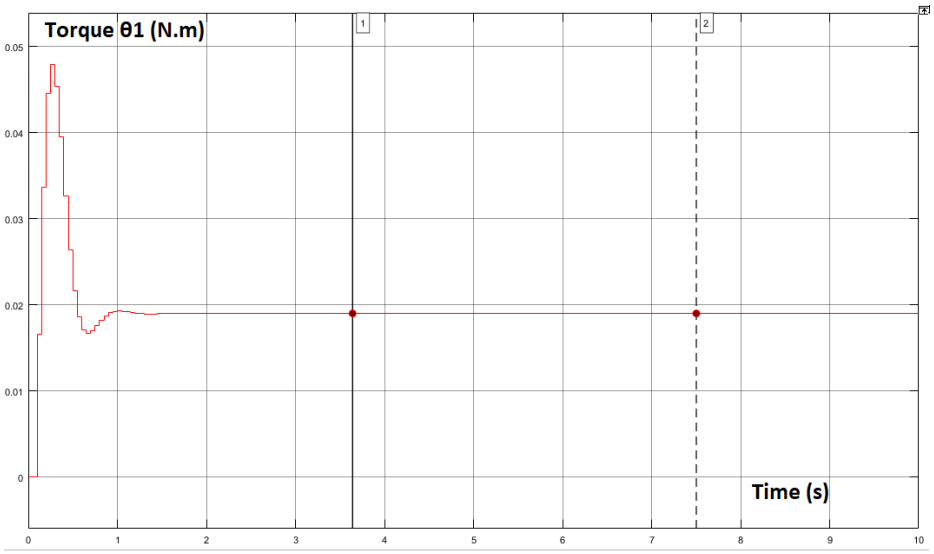


Figure 3-20: Input of θ1 with discrete-time robust tracking [N.m]

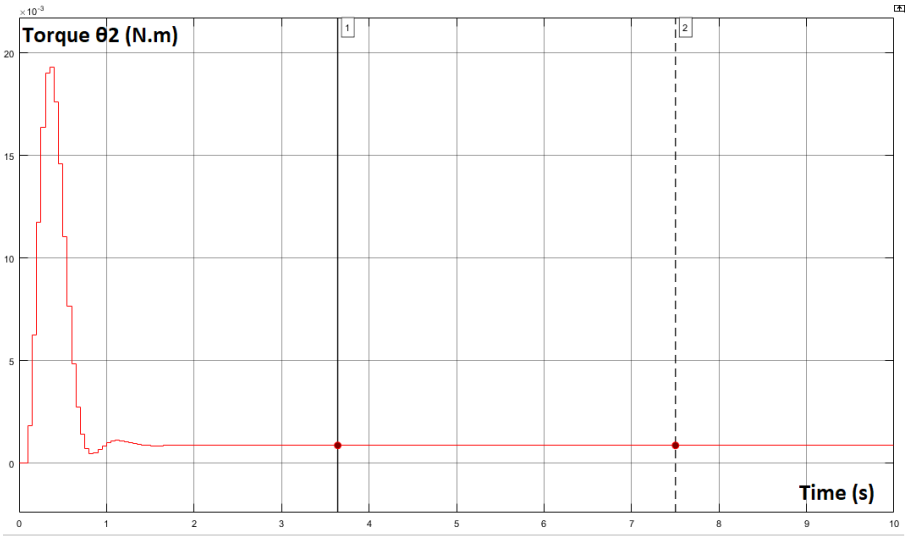


Figure 3-21: Input of θ2 with discrete-time robust tracking [N.m]

As we saw in the responses shown in the figures above, the controllers managed to improve the steady state response of this system with low settling time and overshoots.

Now we have to select the right controller which requires the least input (torque) possible, to sum it up, let us take a look at the following table:

	Required Torque (N. m)	
Controller Type	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$
Tracking	0.030	0.067
Robust Tracking	0.020	0.093
Discrete-Time Tracking	0.027	0.040
Discrete-Time Robust Tracking	0.047	0.019

Note that the discrete-time robust tracking controller shows the least torque required to move each of θ_1 and θ_2, however, because the system utilizes stepper motors, it has no feedback. Thus, the controller selected for the system shall be the Discrete-Time Tracker.

Chapter 4
Path Extraction Algorithm from an Image

In this chapter, we will explain how the path finder algorithm works. The algorithm is based on several processes used in image processing and algorithms related to graph theory. We will discuss it with an illustration.

4.1 What is Image Processing?

Image processing is a method of performing some operations on an image, to obtain an enhanced image or to extract some useful information from it. It is a type of signal processing in which the input is an image and the output may be an image or characteristics / features associated with that image. [7]

4.2 Digital Image:

A digital image a[m,n] described in a 2D discrete space is derived from an analog image a(x,y) in a 2D continuous space through a sampling process that is frequently referred to as **digitization.** [8]

To create a digital image, we need to convert the continuous sensed data into digital form. This process includes 2 processes:

- Sampling: Digitizing the co-ordinate value is called sampling.
- Quantization: Digitizing the amplitude value is called quantization

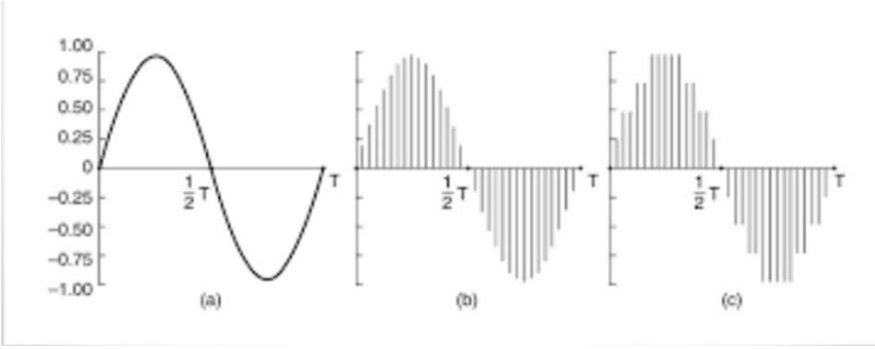


Figure 4-1: Sampling

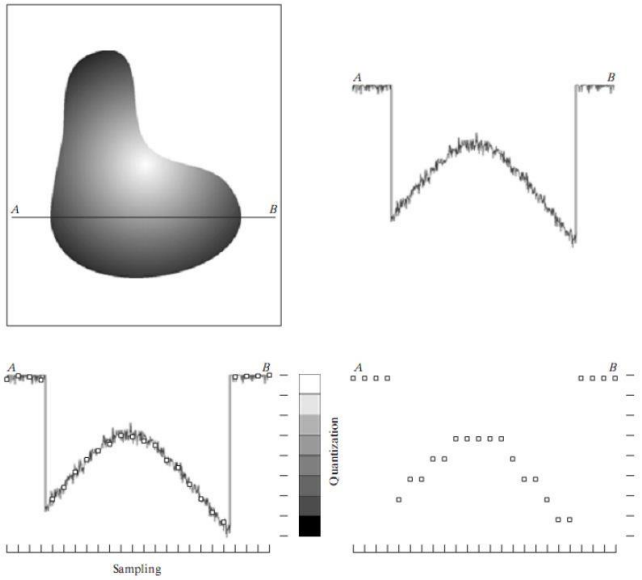


Figure 4-2: Sampling and Quantization

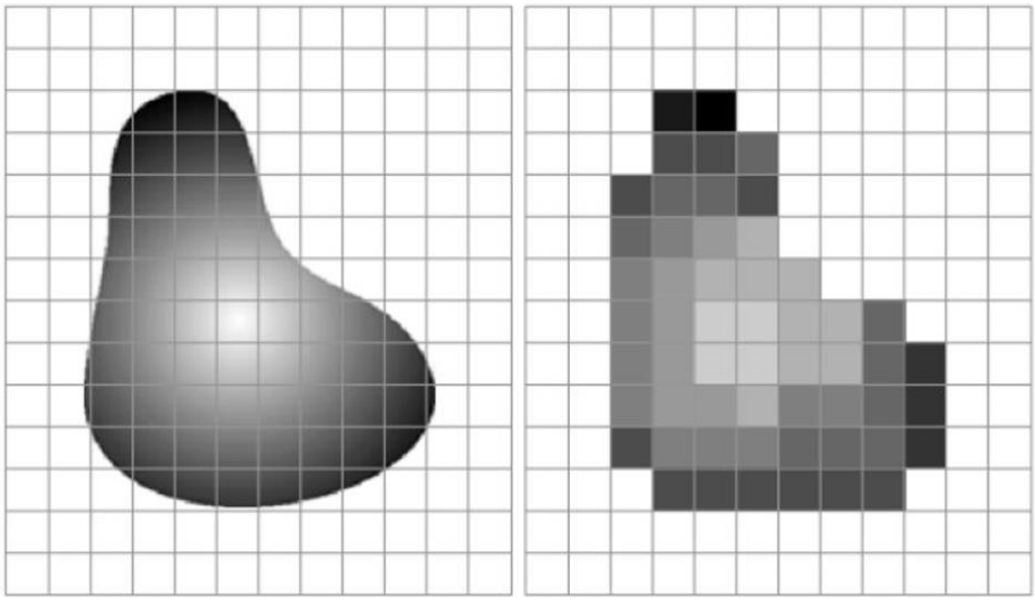


Figure 4-3: Sampling

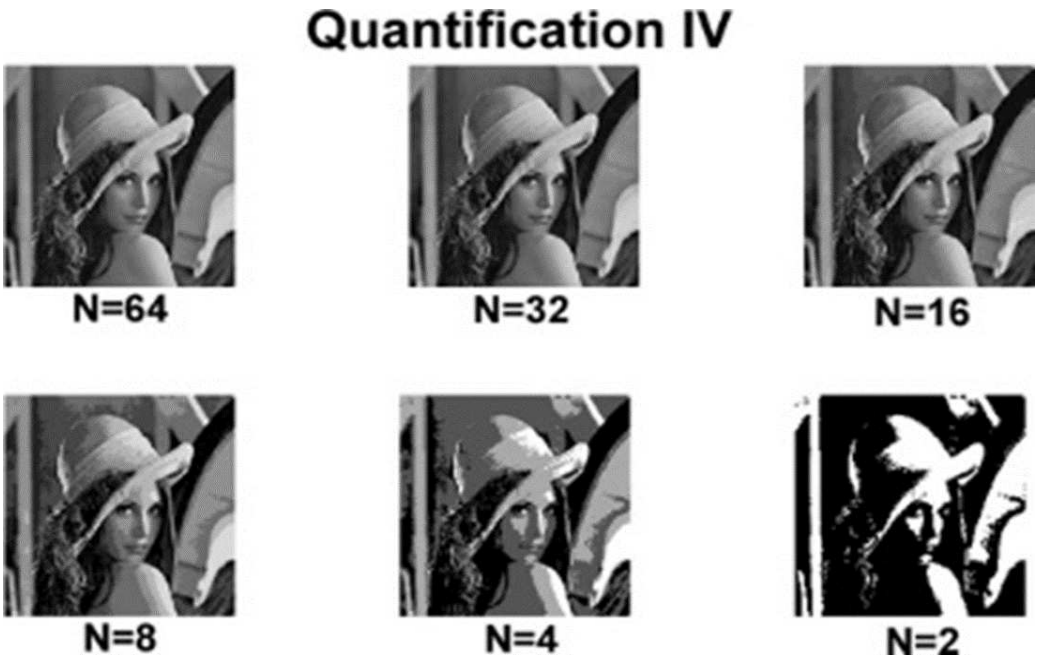


Figure 4-4: Quantification

4.3 Grayscale image:

A grey scale image is what people normally call a black and white image, but the name emphasizes that such an image will also include manyshades of grey. In a (8-bit) greyscale image each picture element has an assigned intensity thatranges from 0 to 255. [7]

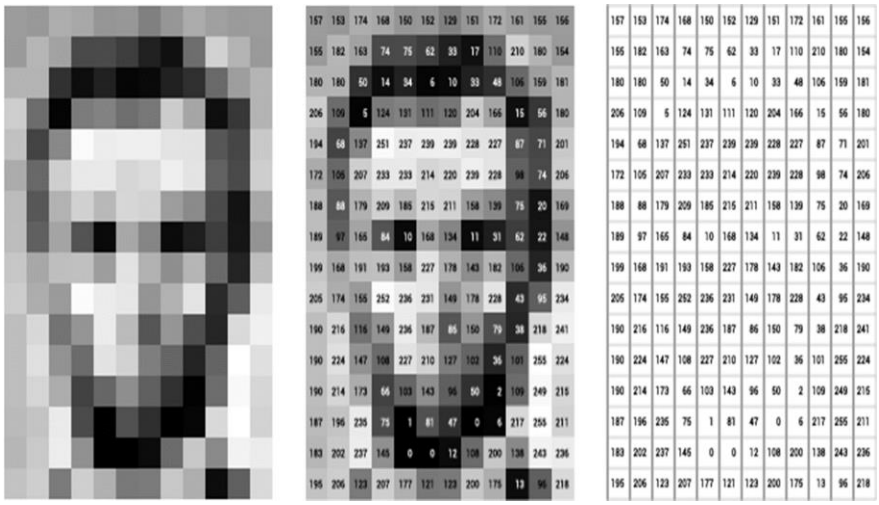


Figure 4-5: Greyscale Imaging

4.4 Colored images:

In colored images, each pixel can be represented by a vector of three numbers (each ranging from 0 to 255) for the three primary color channels: red, green, and blue. These three red, green, and blue (RGB) values are used together to decide the color of that pixel. [7]

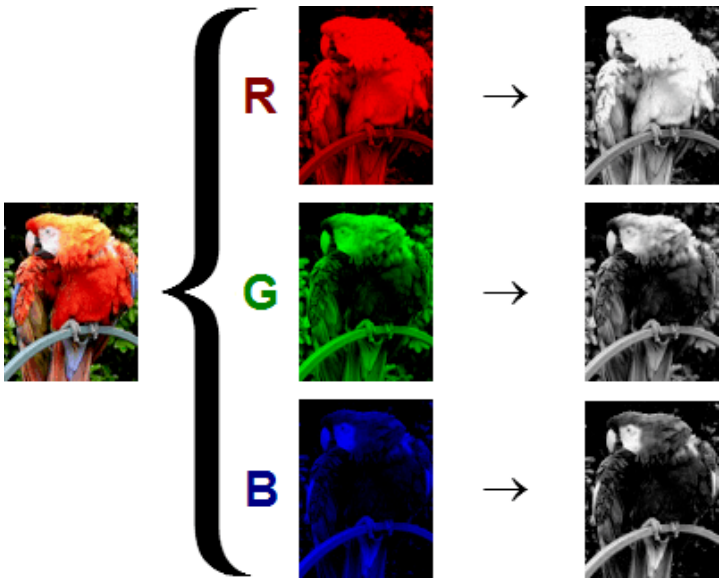


Figure 4-6: Color channeling

4.5 Segmentation:

In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects).

The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. [7]

4.6 Thresholding:

Segmentation involves separating an image into regions corresponding to objects. The simplest property that pixels in a region can share is intensity. So, a natural way to segment such regions is through thresholding, the separation of light and dark regions Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero and all pixels about that threshold to one.

If g(x, y) is a thresholded version of f(x, y) at some global threshold T , g(x, y) = 1 if f(x, y) ≥ T 0 otherwise.

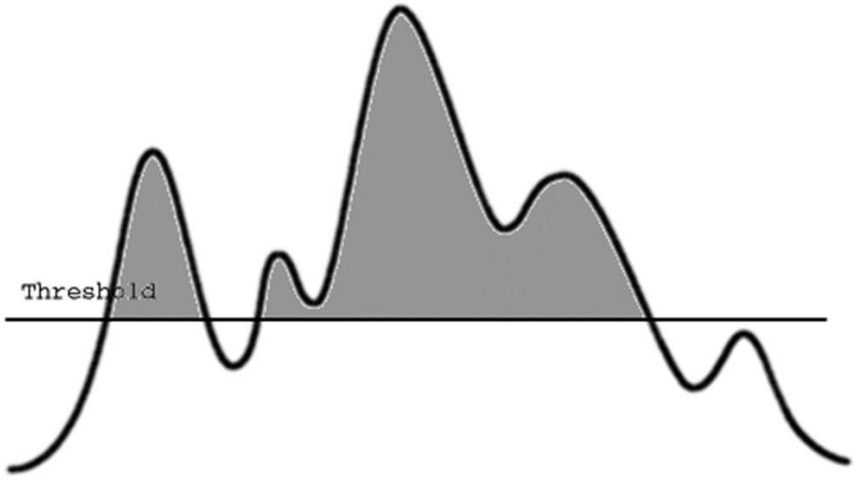


Figure 4-7: Signal Thresholding

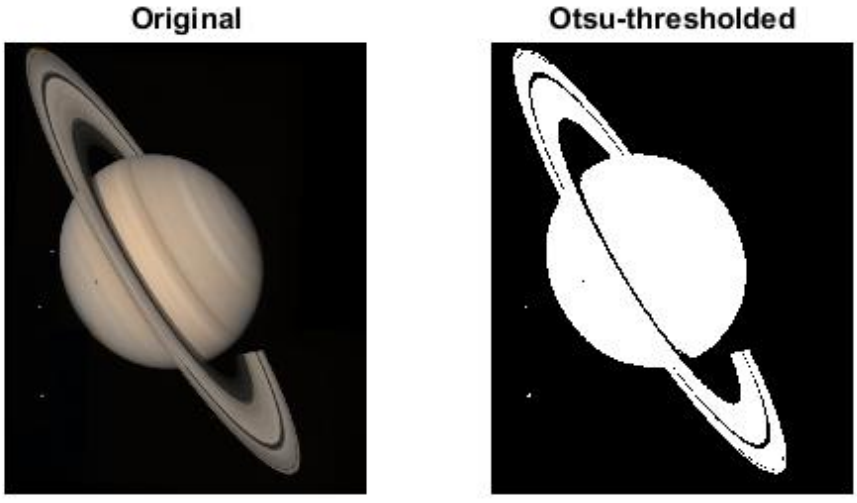


Figure 4-8: Image thresholding

4.7 Skeletonization: [9]

Skeletonization is the result of the thinning process, which peeling the contour of the object until reaches mostmedial one pixel width

The skeleton of an image can be expressed as the set of centers of discs that touch the boundaries of the object at more than one point

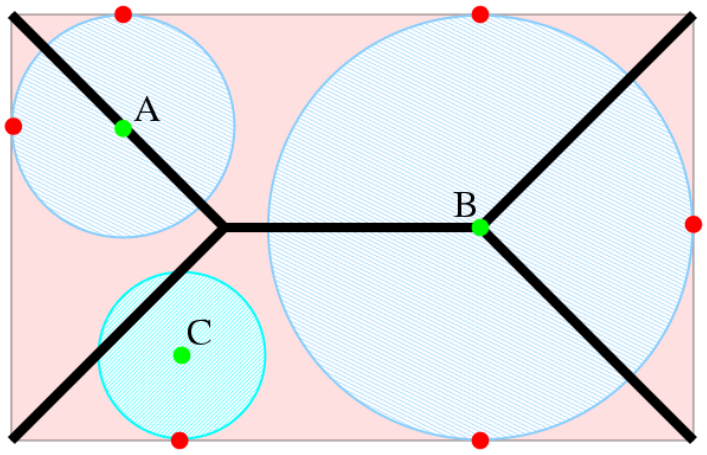


Figure 4-9: Skeletonization

The skeleton of the rectangle is black. Both circles A and B belong to the skeleton, because they have more than one tangent with the boundaries of the rectangle. Circle C does not meet the conditions.

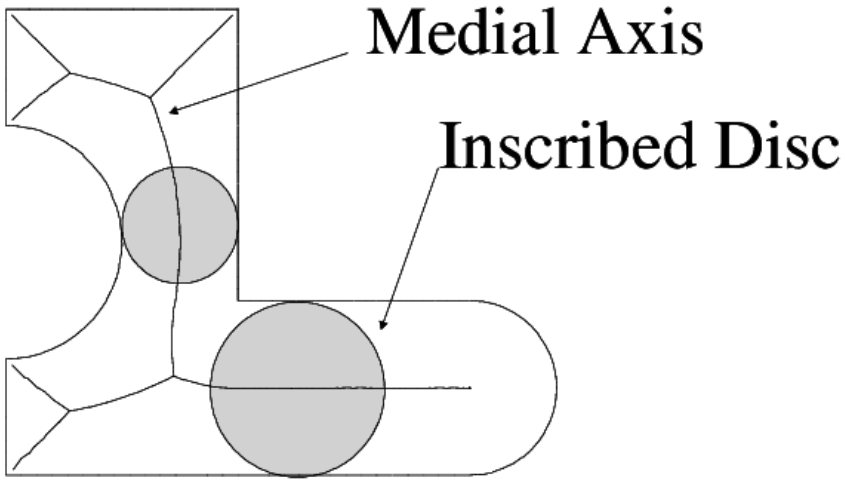


Figure 4-10: Medial axis

There are many types of skeletonization algorithm, all of which produce slightly different results. However, the general effects are all similar, as are the uses to which the skeletons are put

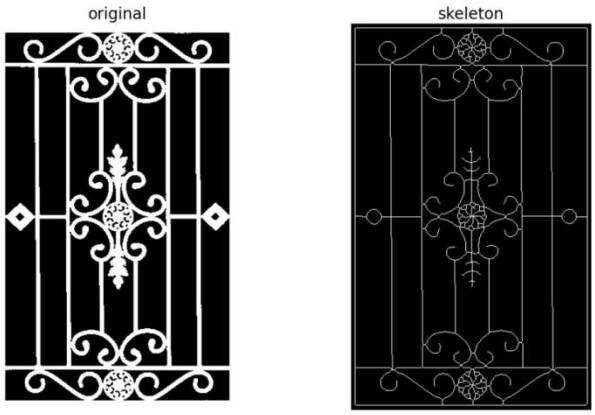


Figure 4-11: Skeletonized window guard

4.8 Graph Theory [10]:

In mathematics, and more specifically in graph theory, a graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related". The objects correspond to mathematical abstractions called vertices (also called nodes or points) and each of the related pairs of vertices is called an edge (also called link or line).

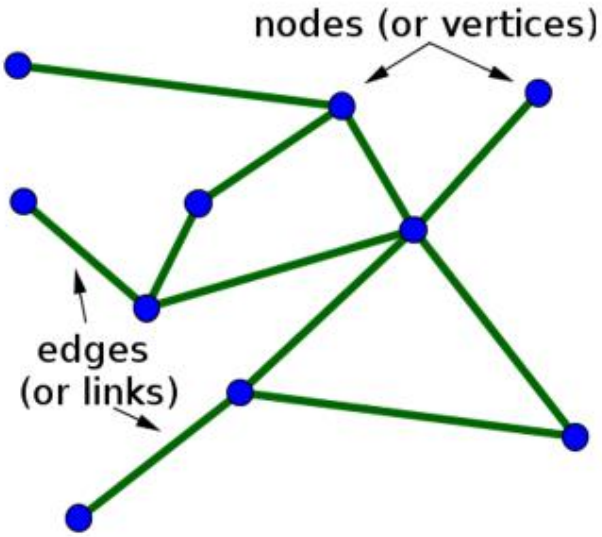


Figure 4-12: Graph (Nodes and edges)

Degree of Vertex of a Graph: It is the number of vertices adjacent to a vertex V.

The number inside each node or vertices represent the degree of vertices.

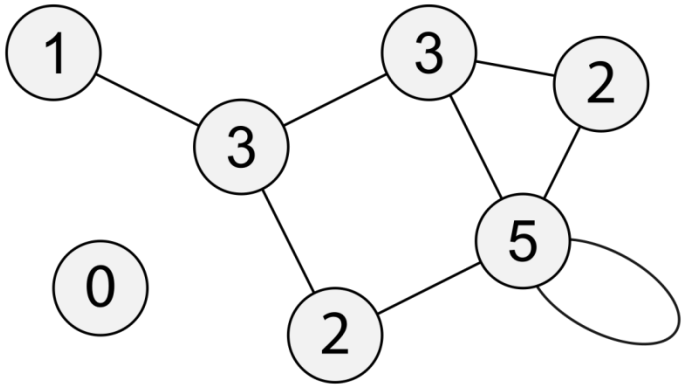
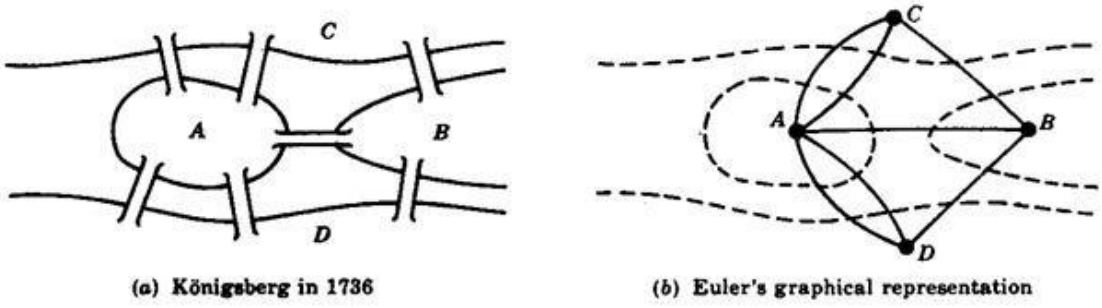


Figure 4-13: Degree of vertex of a graph

Being a postman, you would like to know the best route to distribute your letters without visiting a street twice? This problem of finding a cycle that visits every edge of a graph only once is called the Eulerian cycle problem. It is named after the mathematician Leonhard Euler, who solved the famous Seven Bridges of Königsberg problem in 1736



(a) Königsberg in 1736 (b) Euler's graphical representation

Figure 4-14: Euler representation

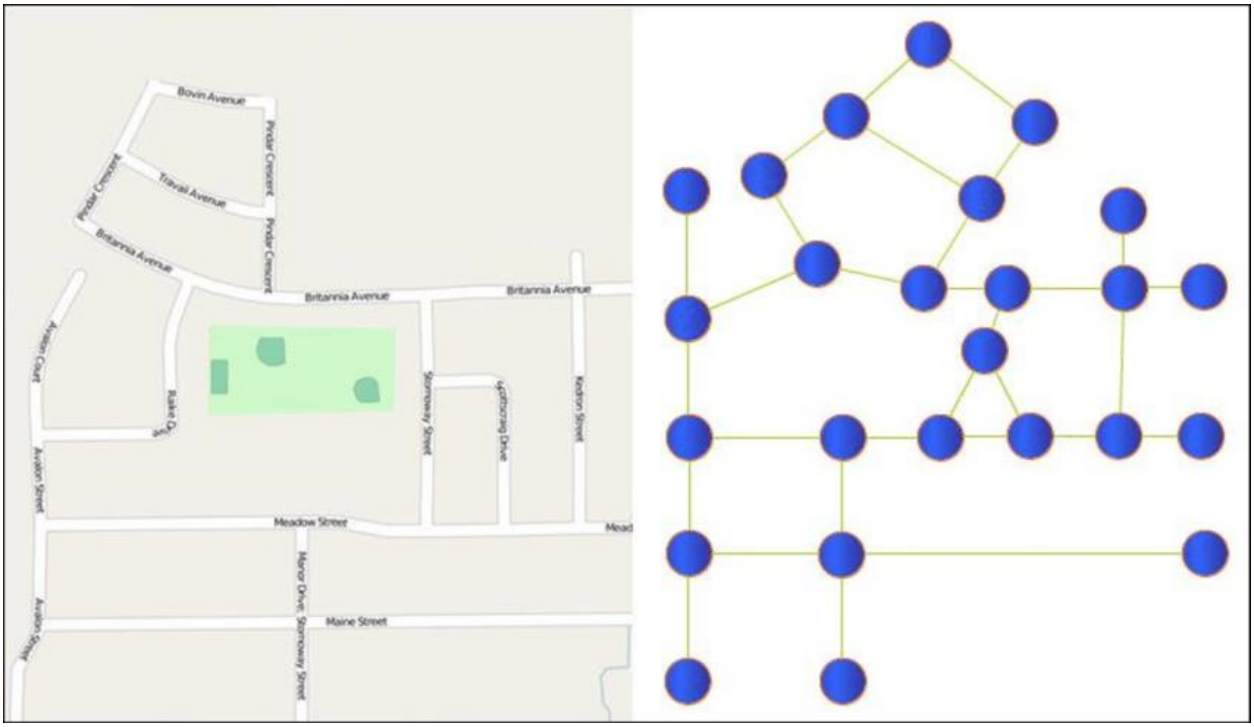


Figure 4-15: Representing roads map into a graph

4.9 Eulerian path [11]:

In graph theory, an Eulerian trail (or Eulerian path) is a trail in a finite graph that visits every edge exactly once (allowing for revisiting vertices). Similarly, an Eulerian circuit or Eulerian cycle is an Eulerian trail that starts and ends on the same vertex. For the existence of Eulerian trails it is necessary that zero or two vertices have an odd degree. If there are no vertices of odd degree, all Eulerian trails are circuits. If there are exactly two vertices of odd degree, all Eulerian trails start at one of them and end at the other. A graph that has an Eulerian trail but not an Eulerian circuit is called semi-Eulerian

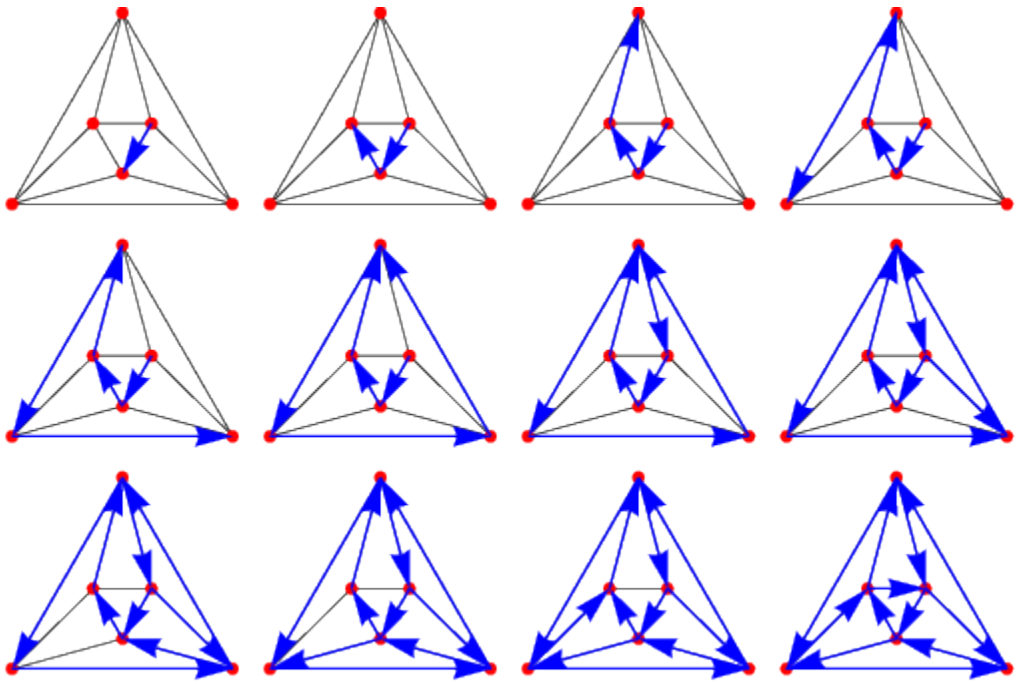


Figure 4-16: Euler circuit

When returning to the bridges of Königsberg, we conclude that the corresponding graph has four nodes with odd degree, therefore no Eulerian cycle exists. However, if the graph fulfilled all requirements, Hierholzer's Algorithm would compute an Eulerian cycle or path.

Idea of the algorithm:

The basic idea of Hierholzer's algorithm is the stepwise construction of the Eulerian cycle by connecting disjunctive circles. It starts with a random node and then follows an arbitrary unvisited edge to a neighbour. This step is repeated until one returns to the starting node. This yields a first circle in the graph. If this circle covers all nodes it is an Eulerian cycle and the algorithm is finished. Otherwise, one chooses another node among the cycles' nodes with unvisited edges and constructs another circle, called subtour. By choice of edges in the construction the new circle does not contain any edge of the first circle, both are disjunct. However, both circles must intersect in at least one node by choice of the starting node of the second circle. Therefore one can represent both circles as one new circle. To do so, one iterates the nodes of the first circle and replaces the subtour's starting node by the complete node sequence of the subtour. Thus, one inegrates additional circles into the first circle. If the extended cycle does include all edges the algorithm is finished. Otherwise, we can find another cycle to include.

In the case of an undirected, semi-Eulerian graph the algorithm starts with one of the two nodes with odd degree. In the directed case with the node with one additional outgoing edge. One of the subtours to be found will then not form a cycle; instead it will also be a path.

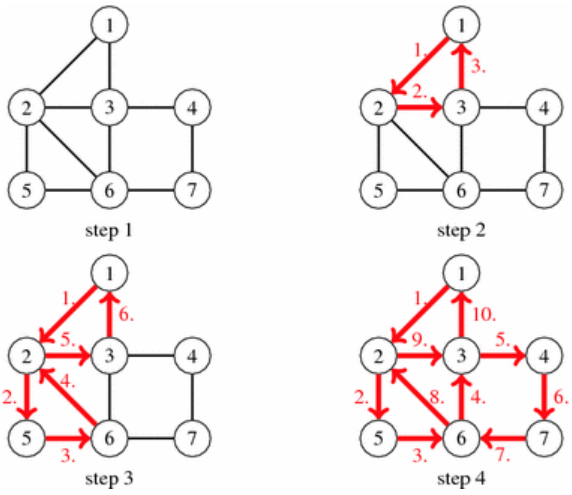


Figure 4-17: The basic idea of Hierholzer's algorithm

4.10 Eulerizing a Graph [11]:

If a graph doesn't have an Euler circuit, we can "Eulerize" it by duplicating existing edges until all degrees are even.

Eulerization is the process of adding edges to a graph to create an Euler circuit on a graph. To eulerize a graph, edges are duplicated to connect pairs of vertices with odd degree. Connecting two odd degree vertices increases the degree of each, giving them both even degree. When two odd degree vertices are not directly connected, we can duplicate all edges in a path connecting the two.

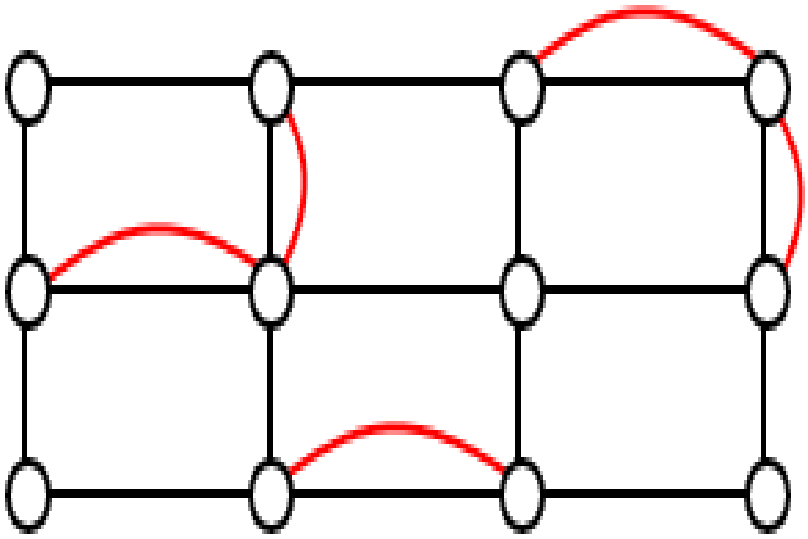


Figure 4-18: Duplicated edges

The red edges are added to the original graph to make it an Euler graph.

4.11 Time Complexity of Algorithms:

Time complexity estimates how an algorithm performs regardless kind of machine it runs on. You can get the time complexity by "counting" the number of operations performed by your code. This time complexity is defined as a function of the input size n using big-o notation. N indicates the size of the input, while o is the worst-case scenario growth rate function. [12]

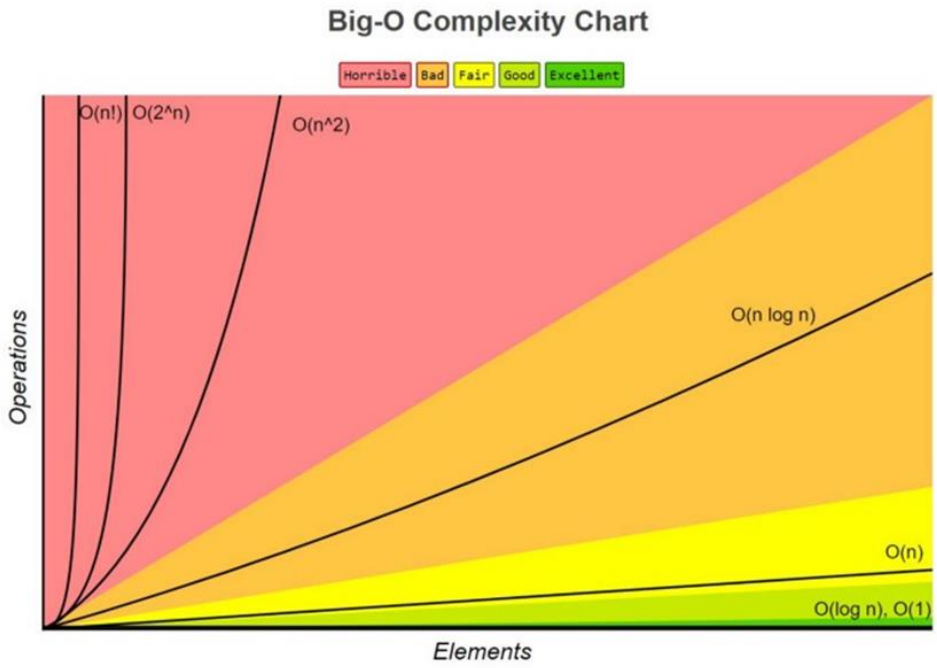


Figure 4-19: Big-O Complexity Chart

The problem of finding the optimal Eulerization is called the Chinese Postman Problem, a name given by an American in honor of the Chinese mathematician Meigu Guan who first studied the problem in 1962 while trying to find optimal delivery routes for postal carriers. This problem is important in determining efficient routes for garbage trucks, school buses, parking meter checkers, street sweepers, and more. [13]

Unfortunately, algorithms to solve this problem are fairly complex; it can be solved in polynomial time.

However if we duplicate every edge in the graph, then all vertices will have an even degree and have an Euler path. This method will not give the optimal

eulerization (the minimum added edges) , but it will eulerize a graph in linear time "fast algorithm " ”depends on the number of edges”. [13]

This method has advantages; since the end-effector crosses any point on the path twice; all parts of the work piece (i.e. The window guard or pattern) are painted equally. This means that it leaves no part of the pattern unpainted. Moreover, the painting procedure is usually not done without any path repetitions (once), but twice or more to make sure the pattern is fully painted.

We designed another version of the algorithm that makes the painter follow the path only once. This is done by adding new lines to the path as shown in the image below:

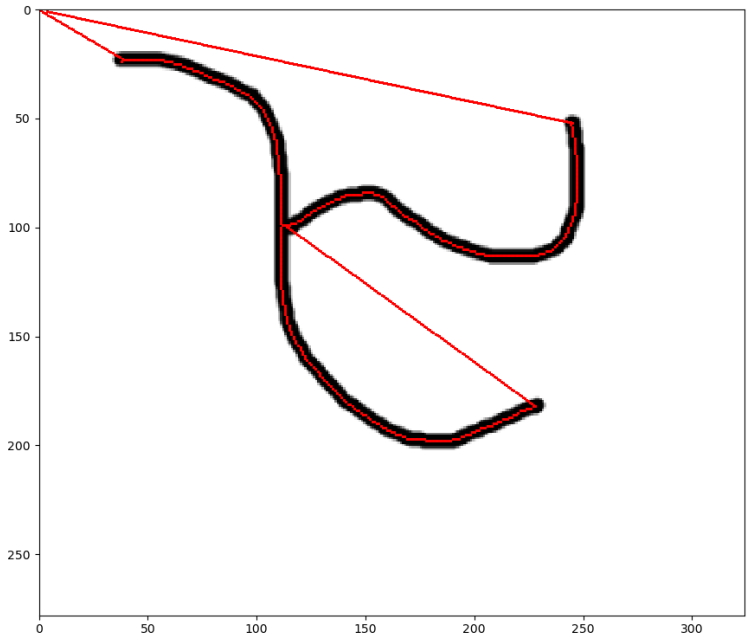


Figure 4-20: Crossing a path only once

These lines will not be painted. It is done for moving between the parts of the graph. Check out the simulation through the shortened link in the appendix.

It is good if we want to paint the object only once time, but because this method adds new cycles of painting, the old version performs faster.

4.12 How does the algorithm work to find the right path?

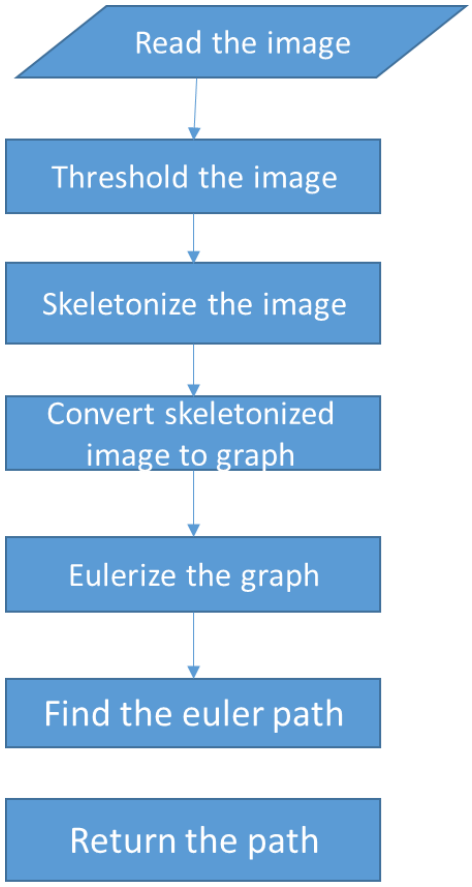


Figure 4-21: Flowchart of the algorithm

- The goal of thresholding the image is to isolate the object to be painted from the rest of the image.
- We come to the second step, which is creating Skeletonize for the image, and the goal is to extract the set of points that represent the path that the machine will take to do the painting process.
- The aim of the third step is to convert the set of points extracted from the second step to a graph in order to apply the path finding algorithm to it.
- The fourth step is to add edges to the graph so that it becomes an Euler graph.
- After these operations, we can apply the path finder algorithm.

4.13 How to convert the skeletonized image to graph[14]?

The pixels that represent the skeleton convert to nodes "vertex" make edges between pixel neighbourhood. The neighbourhood of a pixel is the collection of pixels which surround it as shown in the figure:

x−1,y−1	x−1,y	x−1,y+1
x,y−1	x,y	x,y+1
x+1,y−1	x+1,y	x+1,y+1

.

Figure 4-22: Pixel neighbourhood

If we convert every pixel in the image to graph, the graph structure will be like this:

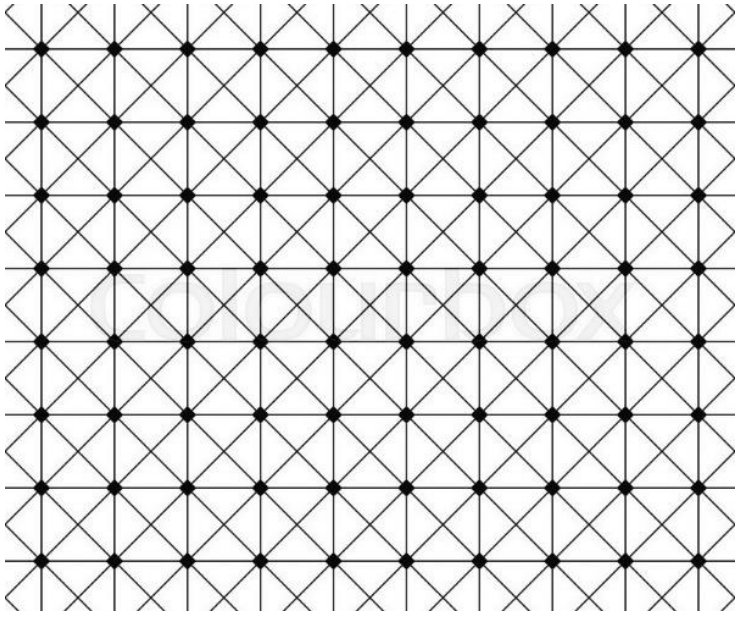


Figure 4-23: The graph structure of the image

The result of the transformation process:

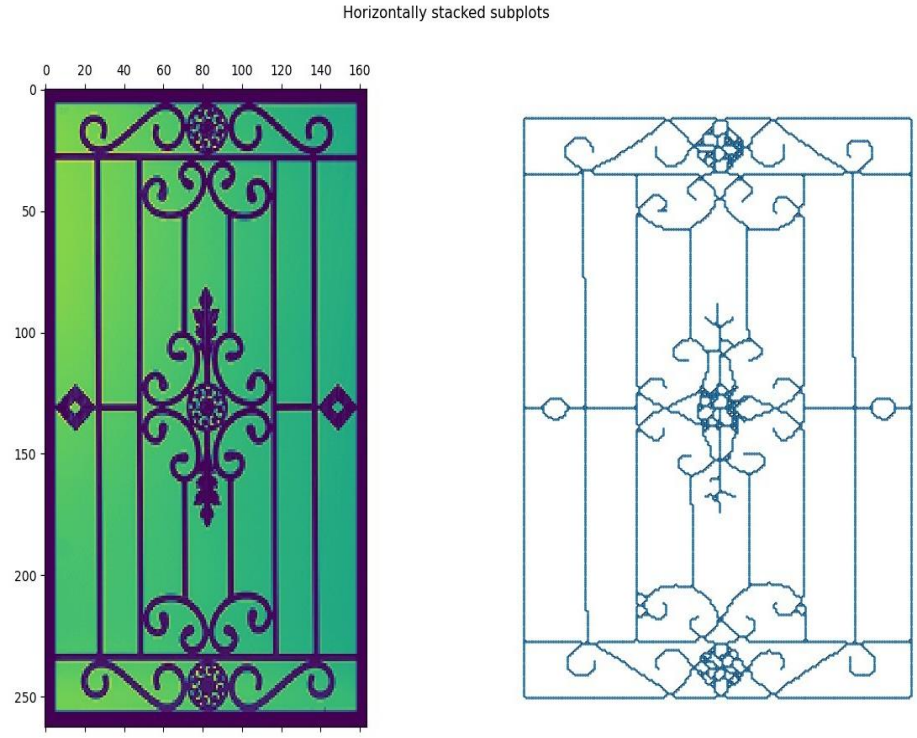


Figure 4-24: Image to graph transformation (1)

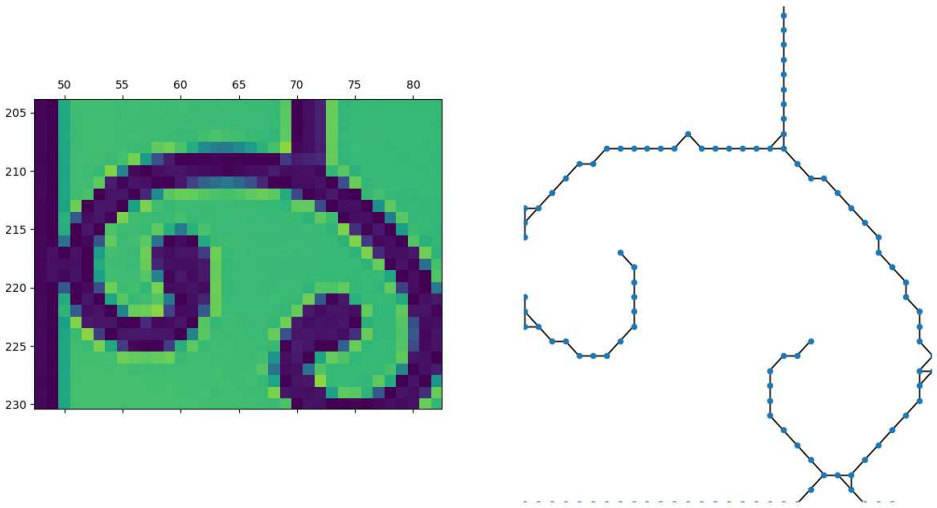


Figure 4-25: Image to graph transformation (2)

Chapter 5 Assembly:

In this chapter, we will explain the mechanical assembly and the electronics.

5.1 The Mechanical Frame Assembly:

5.1.1 The Mechanical Parts:

In this section, mechanical parts we used for this project are explained.

1. **The Aluminum Extrusions:** Here we used four 40 cm-long aluminum V-slot profiles (cross section: 20×20 mm) for the main frame.

Figure 5-1: Aluminum Extrusion Profile

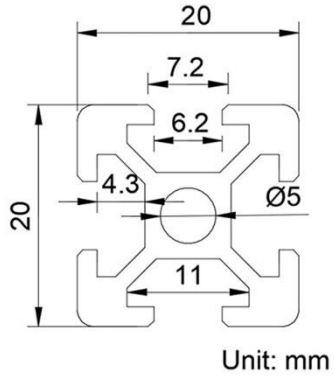


Figure 5-2: Dimensions of the Aluminum profile cross section

2. **The Linear Rails:** We used three 40 cm long MGN12H linear rails for the linear motion system of the project.

Figure 5-3: The MGN12 Linear Rails

Here are the dimensions of the linear rails:

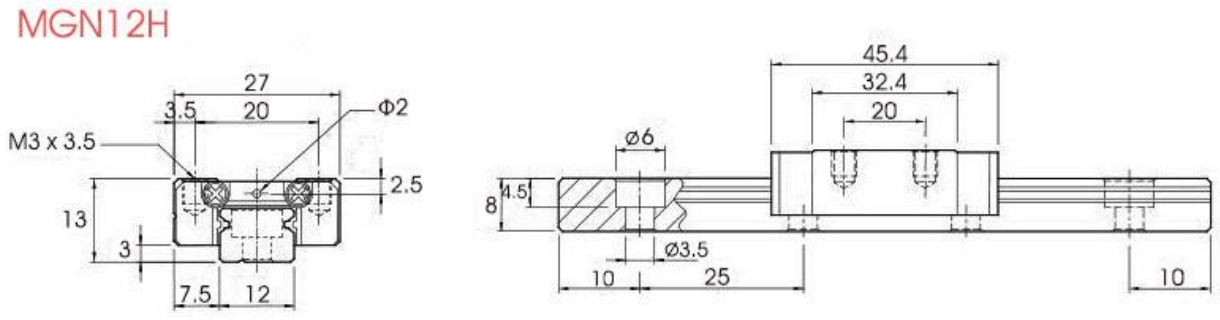


Figure 5-4: MGN12H Linear Rail Dimensions

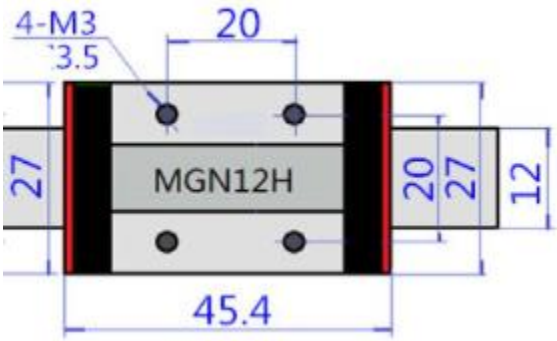


Figure 5-5: MGN12H Dimensions

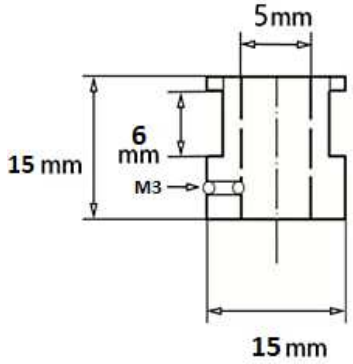
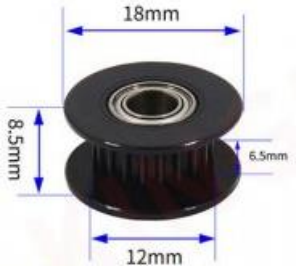
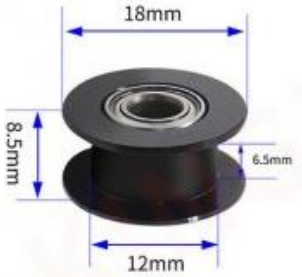
3. The Pulleys: We used three types of GT2 aluminum pulleys; two 5mm bore toothed pulleys for the stepper motors, four 4mm-bore smooth idler pulleys for the bridge motion system and two 4mm toothed idler pulleys, as shown:

Figure 5-6: Stepper Pulley

Figure 5-7: Smooth Idler Pulley

Figure 5-8: Toothed Idler Pulley

The specifications of each pulley are shown in this table:

Pulley Type	Count	Dimensions	Pitch	# of teeth
Stepper	2	 **Figure 5-9: GT2 Stepper Pulley Dimensions**	2mm	20
Toothed Idler	2	 **Figure 5-10: GT2 Idler Pulley Dimensions**	2mm	20
Smooth Idler	4	 **Figure 5-11: Smooth Pulley Dimensions**	-	-

4. The Timing Belt: We used a 5M long 6mm wide toothed rubber GT2 timing belt for the motion system to connect the motion elements of the systems together. The pitch of the timing belt teeth is 2mm. The width is 6mm.

Figure 5-12: The GT2 Timing Belt

5.1.2 The 3D Printed Parts:

1. The Corner Bracket: It is used to connect the four sides of the frame (The 20×20 cm Aluminum extrusions) together. Four of these were printed:

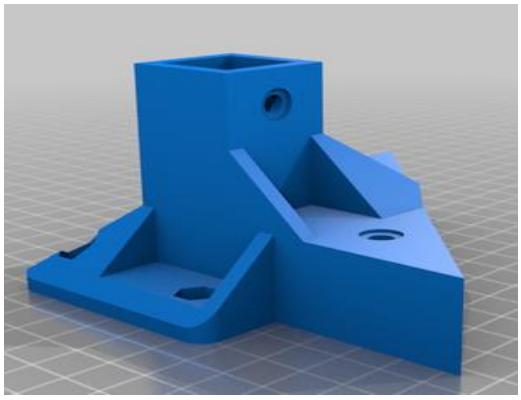


Figure 5-13: Corner Isometric View

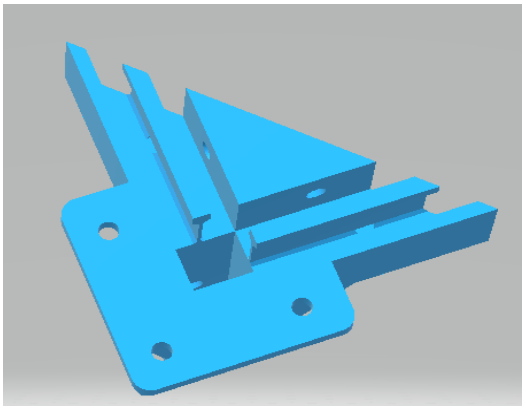


Figure 5-14: Corner Bottom View

2. **The Stepper Motor Mount:** Used to fix the stepper motor on the machine. Two of these were printed.

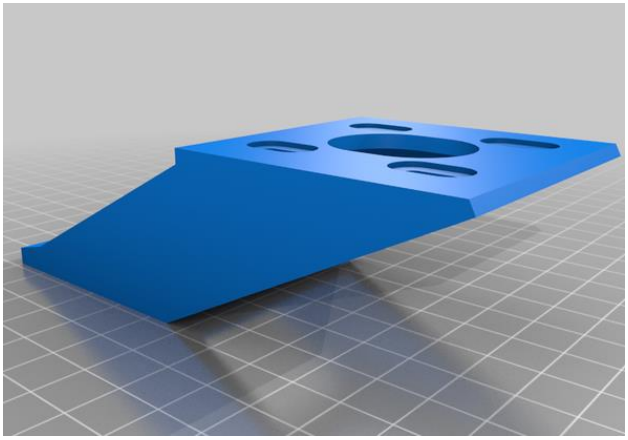


Figure 5-15: The Stepper Mount View 1

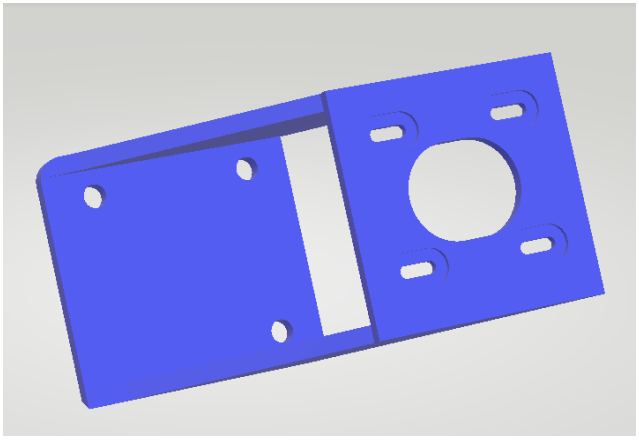


Figure 5-16: Stepper Mount Top

3. **The XY-Brackets:** It is used to connect the three linear rails together. The smooth pulleys are fixed on it. Two XY-Brackets were printed.

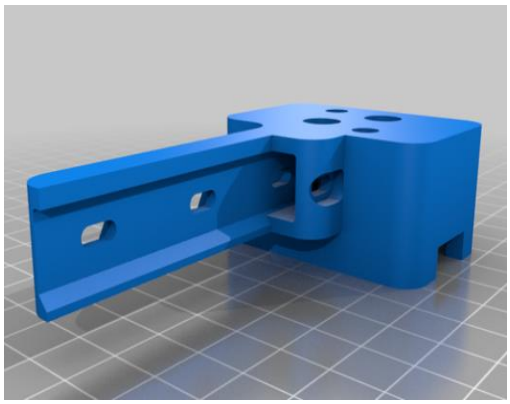


Figure 5-17: XY-Bracket View 1

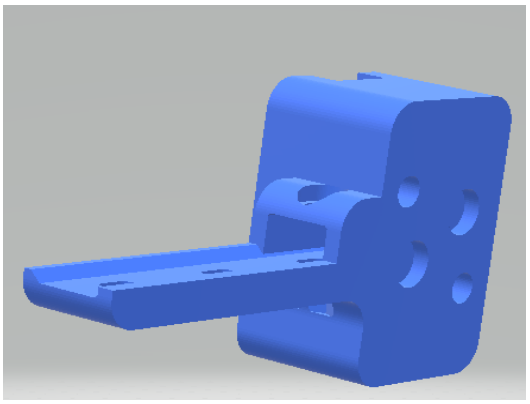


Figure 5-18: XY-Bracket View 2

4. **The Pulley Bracket:** It is used to fix the toothed idler pulleys on it. Two were printed.

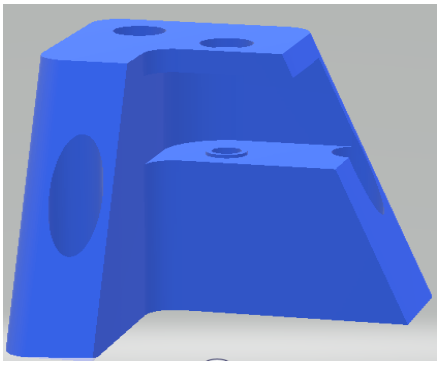


Figure 5-19: Pulley Bracket front

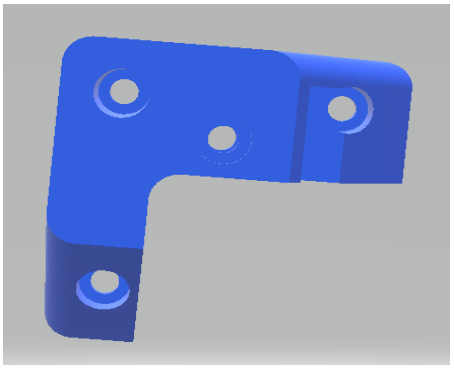


Figure 5-20: Pulley Bracket Top

5. **The Painter Carriage Mount:** It is used to fix the painter head and the ends of the timing belts.

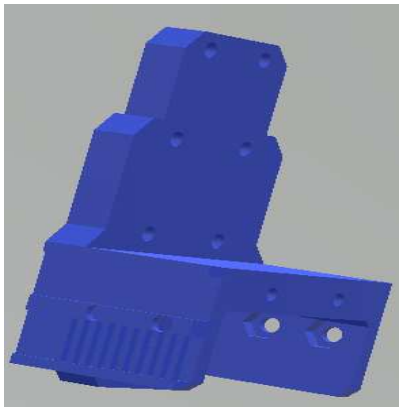


Figure 5-21: The Carriage Mount Side

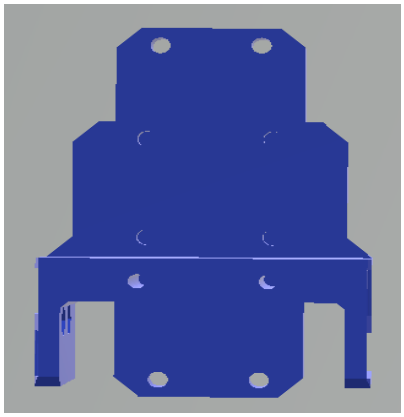


Figure 5-22: The Carriage Mount Front

6. **The Calibration Cube:** Four of these were printed. They are put on the cross area in the corner brackets. Face area is 20×20 mm^2

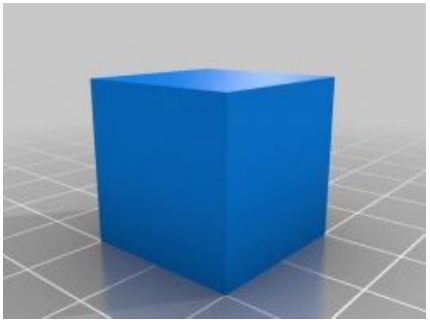


Figure 5-23: The Calibration Cube

5.1.3 The Mechanical Frame Assembly Process:

First of all, the aluminum extrusions and the calibration cubes were installed on the corner brackets as shown in fig. 5-24:

Figure 5-24: Mechanical Frame Assembly 1

The side linear rails are fixed on the frame with M3 screws and T-nuts as shown in fig. 5-25:

Figure 5-25: Mechanical Frame Assembly 2

The XY brackets had two close diagonally dug holes for the pulleys that do not work for this system, so we had to dig vertical holes for the smooth pulleys. The smooth pulleys are installed with M4 screws and nuts:

Figure 5-26: Smooth pulleys installed on the brackets

The toothed idler pulleys are installed with M4 screws and nuts on the pulley brackets:

Figure 5-27: Pulley Bracket Assembly

The 3D printed parts (Stepper Mounts, XY brackets, pulley brackets and the carriage mount), along with the middle rail and the motors are installed on the frame:

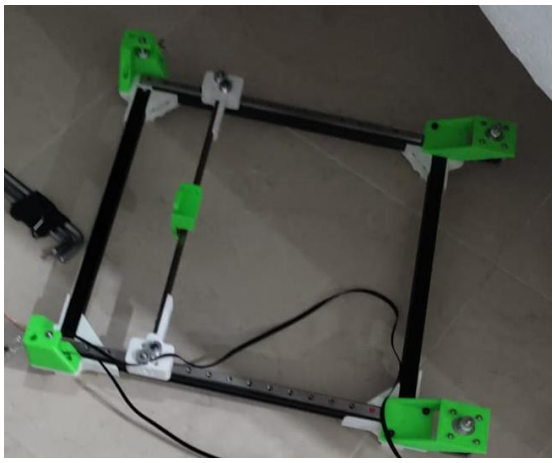


Figure 5-28: Mechanical Frame Assembly 3

The timing belt is installed on the system and fixed on the carriage mount:

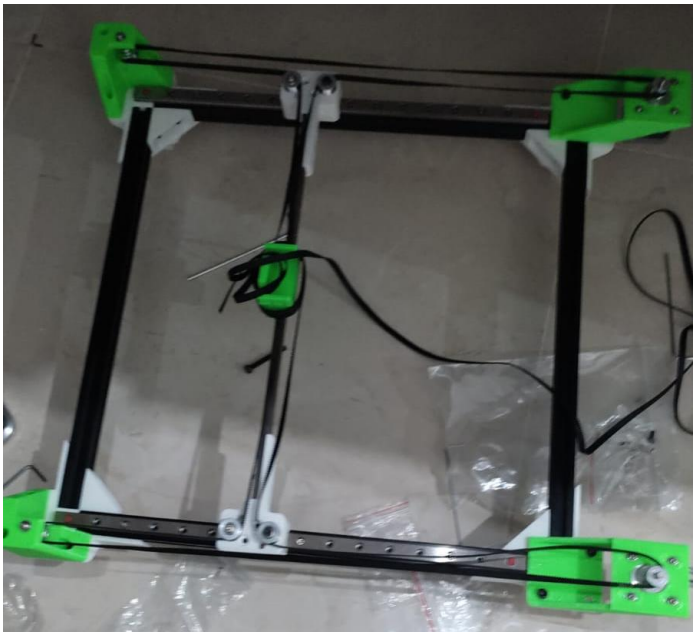


Figure 5-29: Installing the timing belt

The H-Frame is complete as shown in the figure 5-30:

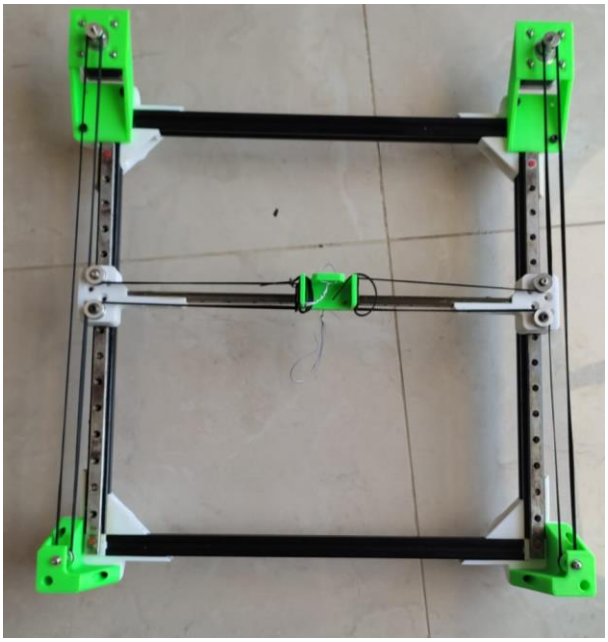


Figure 5-30: The H-Frame

Finally, we added a stand for the camera (phone) to take the images.

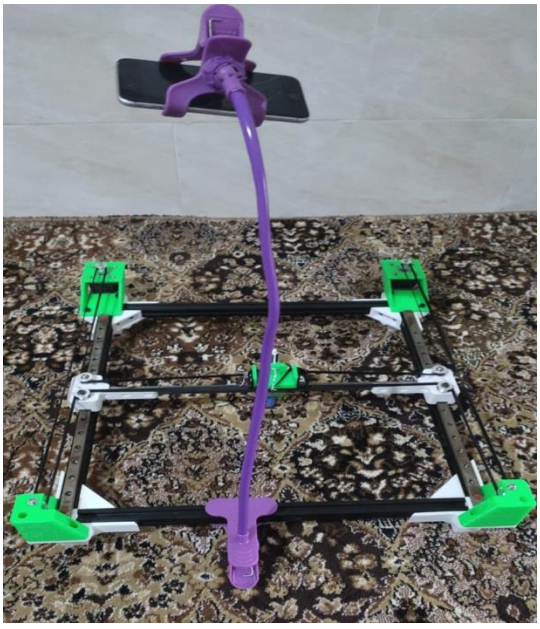


Figure 5-31: The H-Frame with a stand

5.2 The Electronics:

5.2.1 The Raspberry Pi Controller:

For this system, we used the Raspberry Pi 4 controller. This controller has an open source Linux operating system (RPi OS) which is easy and simple to use. The memory of this controller is large enough (4 GB RAM) to perform the image processing algorithms and transfer the signals to the stepper motor drivers. The components of the controller are shown in the figure 5-31:

Figure 5-32: Rasperry Pi 4 Controller

The pin configuration of the controller is shown here:

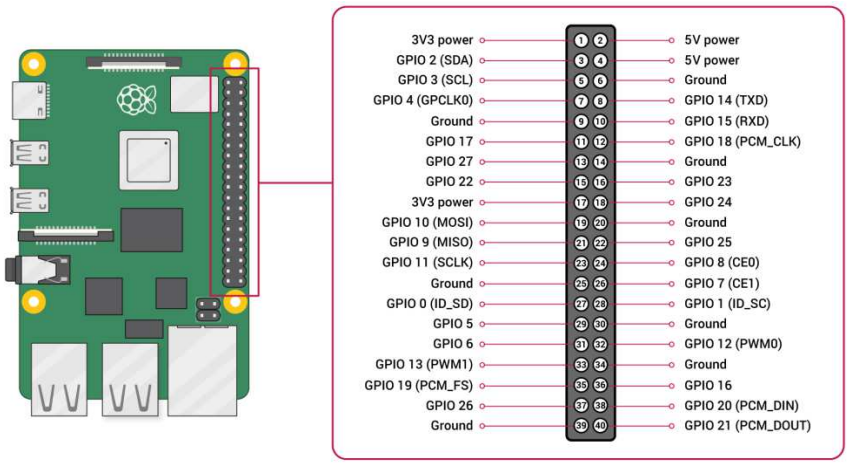


Figure 5-33: Raspberry Pi Pinout

5.2.2 The Stepper Motor Drivers:

We used two EasyDriver stepper drivers for the stepper motors. These drivers are open source and have a current controller potentiometer which can prevent overheating. Only two pins are connected to the controller, which minimizes the number of wires used for the system. For more information about this driver, refer to the Appendix.

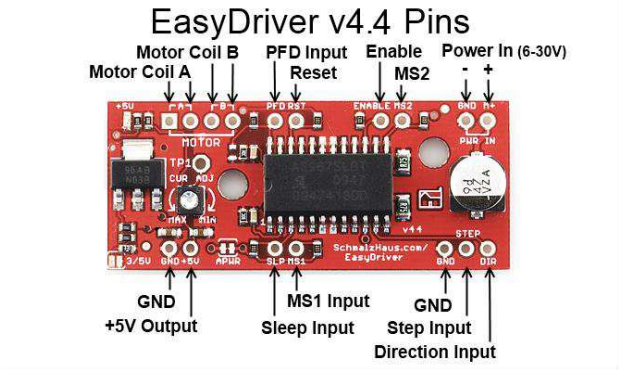


Figure 5-34: The EasyDriver

5.2.3 Stepper Motors:

For the prototype, we used the Nema17 Step Motor (SL42STH40-1684MA-23). These stepper motors are capable of carrying high torques up to 4000 g.cm.

Specifications: 1.8^{o} / step, 5mm diameter shaft, 1.7 A rated current.

Figure 5-35: The Stepper Motor

5.2.4 The Power Supply:

We used the ZONESTAR Switching power supply. This power supply gives an output of 12V and 20A DC, which is enough to run the stepper motors at the fullest.

Figure 5-36: The Power Supply

5.2.5 The Wiring:

The coils of the stepper motors are connected to the four-pin inputs of the EasyDriver module. The step and direction pins are connected to the Raspberry Pi Controller pins (GPIO); 27 and 22 for the first motor and 23 and 24 for the second. The ground line connects the power supply, the drivers and the controller. We connected the power supply to a separate breadboard for safety measures and a more organized system. The wiring is shown in the fig. 5-36.

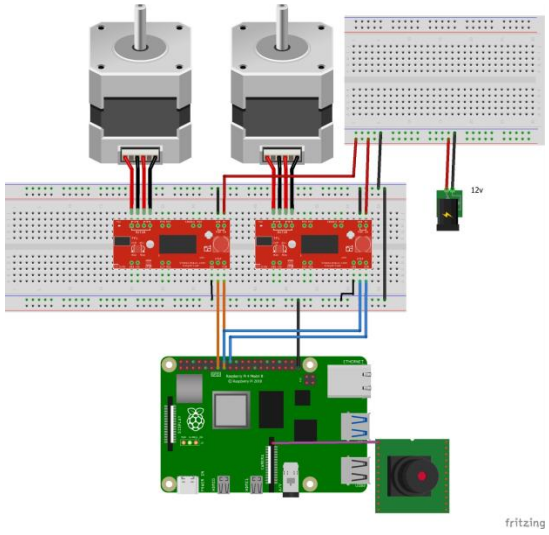


Figure 5-37: The Wiring

Chapter 6 Calibration and Experimental Work:

6.1 Calibration

In this section we will explain how we match the coordinates of the image with the coordinates of the machine and the use of the parameters of the calibration in order to get the correct calibration.

Suppose the camera is perpendicular to the frame of the machine, no rotation and the point (0,0) of the frame map to (0,0) in the image. In this case we can just find the ratio between the length of the axis in the plane (x or y) or any straight piece on the machine and the number of pixels that represent it. This ratio represents the scale between the real length measurements and the pixels of the image. We can resize (rescale) the image pixels to the real measurements, for instance, the dimensions of an A4 paper is 210×297 mm and the dimension of the image should be the same after resizing (210×297 pixels):

$$\text{Ratio} = \frac{Length}{\#\ \text{of pixels}}$$

But what if we could not put the camera in that perfect position?

Some of the problems we will face are:

1. Shifting between the coordinates of the image and the machine coordinate
2. Rotation

The “worst” of all these problems is that the camera orientation is not perpendicular to the frame of the machine. We can solve these problems using a new coordinate system called the **homogenous coordinates**.

6.1.1 Homogeneous Coordinates [15]

The 2d point in the homogenous coordinate system is represented as: $[x \quad y \quad 1]^\top$.

And, $[x \quad y \quad 1]^\top = [ax \quad ay \quad a]^\top$, where a is a constant.

For instance, the point (2,3,1)=(4,6,2)=(1,1.5,0.5).

So there are infinite values that represent the same point. The projection of a ray passing the center of the camera in the image is a point. This ray contains infinite points with this relation.

Take any point on this ray $[\boldsymbol{x}` \quad \boldsymbol{y}` \quad \boldsymbol{z}`]^\top$, make a projection on the image plane and represent it in the homogenous coordinates as follows:

$$[\boldsymbol{x}`/\boldsymbol{z}` \quad \boldsymbol{y}`/\boldsymbol{z}` \quad \boldsymbol{z}`/\boldsymbol{z}`]^\top = [\boldsymbol{x} \quad \boldsymbol{y} \quad \boldsymbol{1}]^\top$$

This value is equal for any point after projection. That means that all points project on one point in the image plane as shown in the image below:

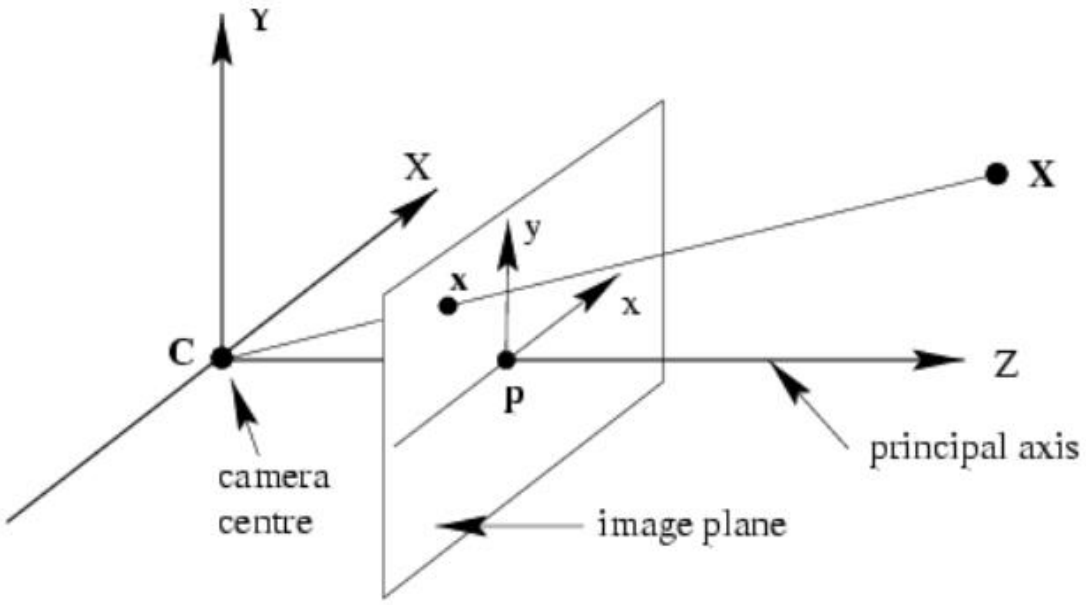


Figure 6-1: Projection to the Image Plane

6.1.2 What Is the Use of the Homogenous Coordinates?

Back to the problem, we represent the change from the plane of the machine to the image plane as transformation, and when the points are represented using homogenous coordinates, it turns out that these transformations are linear. [16]

The following table shows matrix representations for two-dimensional transformations in homogeneous coordinates:

1. Translation	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ t_x & t_y & 1\end{bmatrix}$ or $\begin{bmatrix}1 & 0 & t_x\\ 0 & 1 & t_y\\ 0 & 0 & 1\end{bmatrix}$
2. Scaling	$\begin{bmatrix}S_x & 0 & 0\\ 0 & S_y & 0\\ 0 & 0 & 1\end{bmatrix}$
3. Rotation (clockwise)	$\begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$
4. Rotation (anti-clock)	$\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$
5. Reflection against X axis	$\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$
6. Reflection against Y axis	$\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
7. Reflection against origin	$\begin{bmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$
8. Reflection against line Y=X	$\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$
9. Reflection against Y= –X	$\begin{bmatrix}0 & -1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$
10. Shearing in X direction	$\begin{bmatrix}1 & 0 & 0\\ Sh_x & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
11. Shearing in Y direction	$\begin{bmatrix}1 & Sh_y & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
12. Shearing in both x and y direction	$\begin{bmatrix}1 & Sh_y & 0\\ Sh_x & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

Figure 6-2: Homogenous Transformations

Here are matrices with its properties :

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, $\mathbf{I}, \mathbf{J}$ (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Figure 6-3: Homogenous transformations 2

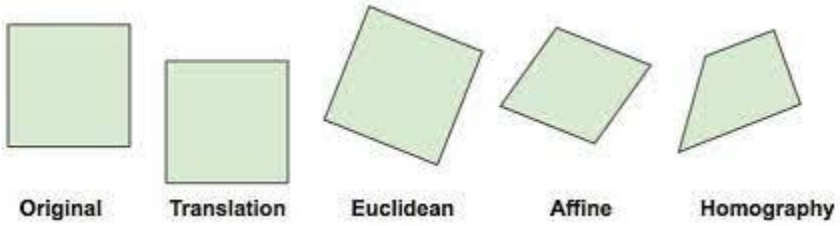


Figure 6-4: Effect of transformation.

Suppose there is a matrix transforming the plane of the machine to the second plane which is the image:

$$\begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Take the inverse, we get to the real value of the frame plane:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H^{-1} \begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix}$$

This matrix is called **homography matrix,** which maps between two planes [17]

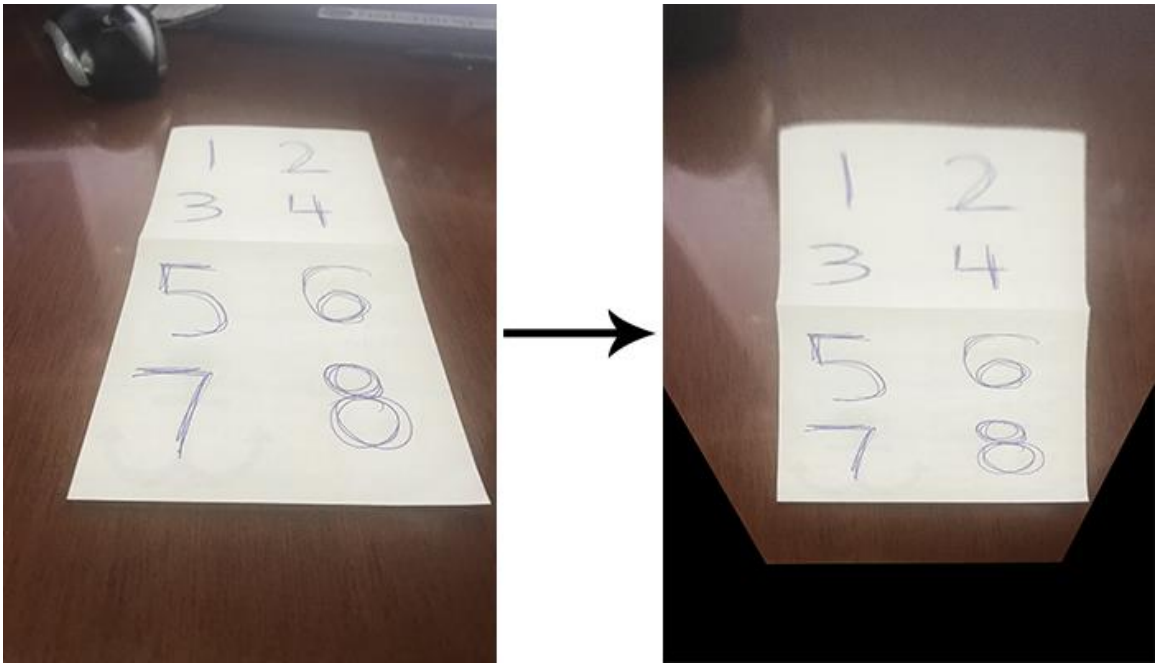


Figure 6-5: A Homographic transformation

Since the degree of freedom of the matrix is equal to eight, we need four corresponding points to find it.

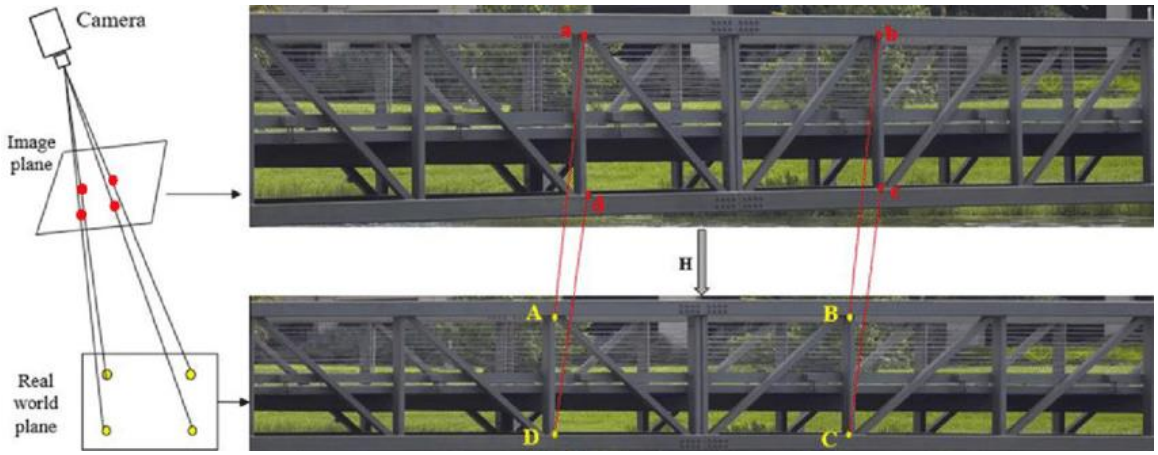


Figure 6-6: 4 points to map between image plane and real world plane [18]

After mapping between the coordinates, we know that the new image points ($H^{-1}[x` \quad y` \quad 1]^{\top}$) represents the real point values of the frame coordinates. For instance, map pixels to cm or mm.

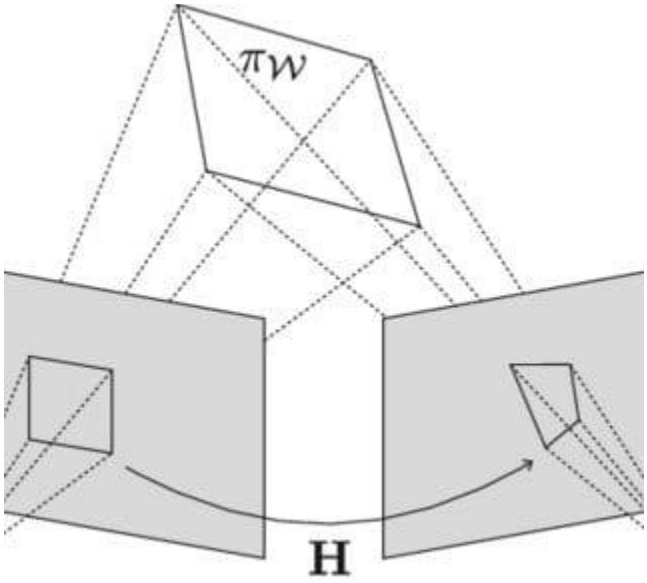


Figure 6-7: Effect of The Homography Matrix

The table on the next page shows the steps of the calibration and the path generation:

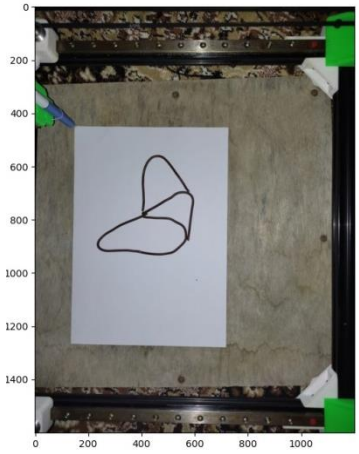


Figure 6-8: Original image

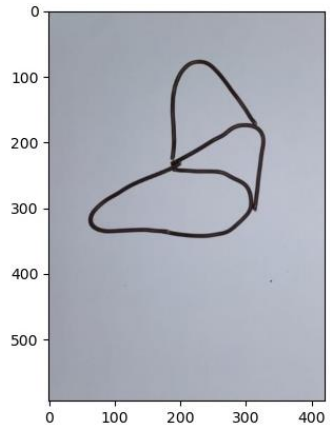


Figure 6-9: Transformed image

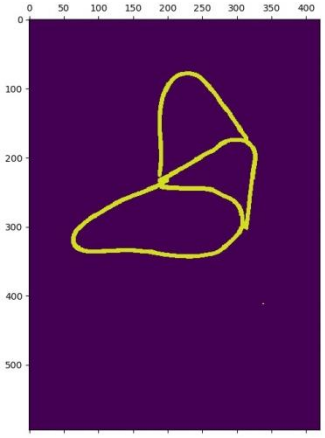


Figure 6-10: Thresholded Image

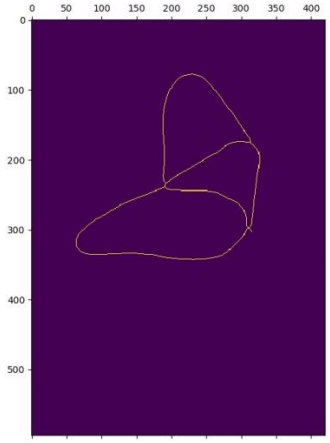
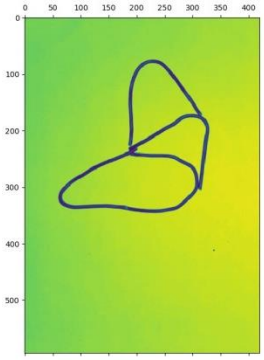


Figure 6-11: Skeletonized Image



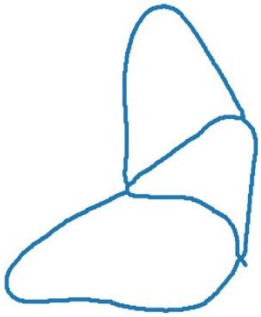


Figure 6-12: Generated Path

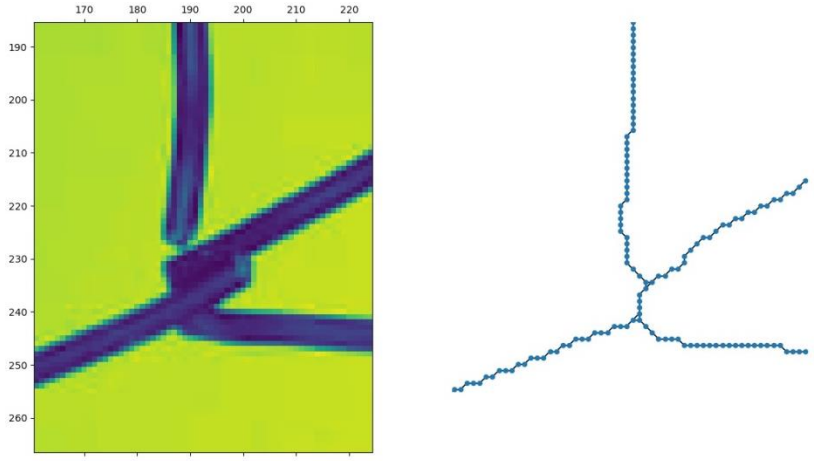


Figure 6-13: A zoomed view of the generated path

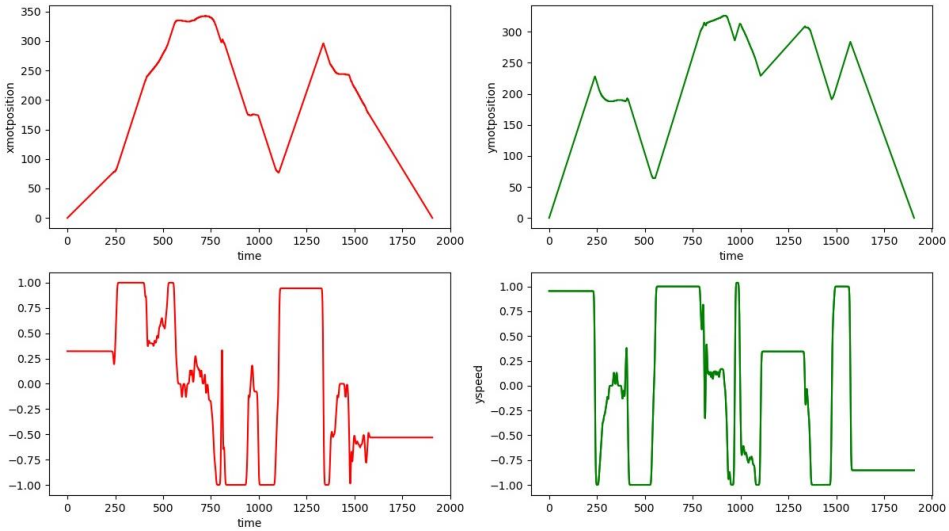


Figure 6-14: The position and speed signals of the two motors

6.1.3 The Mechanism and the Motor Effect on the Signals:

Before sending the signals to the motors, there is another transformation to be done, which is the motion mechanism of H-Frame. This transformation is linear and we can remove the effect of this transformation by multiplying the signal by the inverse of H-Frame kinematics matrix (Refer to chapter 3).

$$\begin{bmatrix} x` \\ y` \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Where $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is the H-Frame matrix.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} x` \\ y` \end{bmatrix}$$

The path generated from the image is a set of coordinates executed sequentially. In the code, it is just an array; more accurately, two arrays; each one of them is sent to motor. In general, the motor takes the speed (frequency) as an input and the output is the position. That means that it is similar to an integral (low pass filter). So, we need to differentiate the signal before sending it to the motor in order to eliminate motor effect on the signal.

$$\int_{x0}^{x} \frac{df(x)}{dx} dx \ = f(x) \ - f(x_0) = f(x), assume\ f(x_0) \ = 0$$

6.1.4 Moving the Head to the Start Point (Homing):

In most cases, the start point for painting an object is not the origin, so we need to move the spray head (end-effector) from the origin point to the start point of the path, and once the path is finished, the spray head must return to the origin point. Refer to the code in the Appendix.

6.1.5 The Effect of the Pulleys:

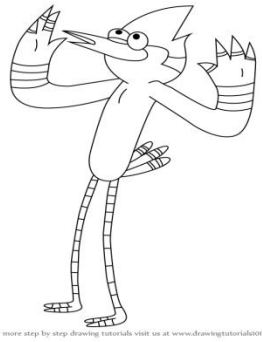
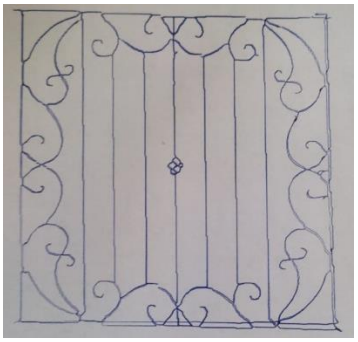
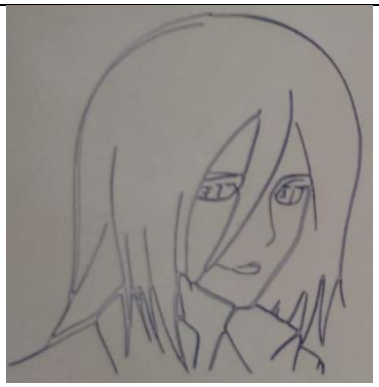
The size of the pulleys (the diameter) affects the scaling. In the code, we can control the speed and also the scaling, so in this step the calibration is done manually by adapting the scale variables in the code.

6.2 Experimental Work

6.2.1 Images converted to paths:

The first experiment we did on the machine was using it as a plotter and making it draw different drawings and images.

Here are comparisons between the images and the plotted paths:

Original	Plotted
 Figure 6-15: Image 1	 Figure 6-16: Plot 1
 Figure 6-17: Image 2	 Figure 6-18: Plot 2
 Figure 6-19: Image 3	 Figure 6-20: Plot 3

Check out the video of the plotting process of the window guard (image 2). Short link is in the Appendix D.

6.2.2 Using the Camera and Painting a Real Pattern:

For this, we used a camera of an android phone and the IP Camera app. Then, we used the calibration and the homing algorithms mentioned in section 6.1 for the pictures taken by the camera. The machine crossed (painted) the path on the patterns accurately. The patterns are made of wires. Go to the shortened links in the appendix to see the processes of two of the patterns. (Or click here and here if you are using an e-version of this thesis).

Figure 6-21: Pattern 1 Painting

For the second pattern:

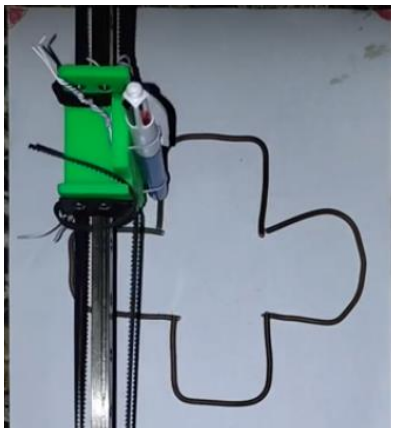


Figure 6-22: Pattern 2 painting

6.2.3 Repeatability Test

For the repeatability test, we used the machine as a plotter and made it draw a line and a square with three different colors. The machine repeated the same line as desired.

The Line:

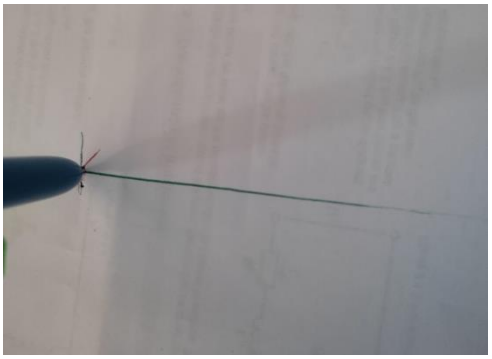


Figure 6-23: Line 1

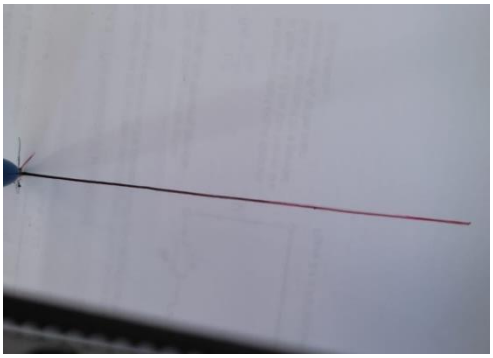


Figure 6-24: Line 2

Figure 6-25: Line 3

Note: In the first picture, we notice that the line became lighter towards the end. That is an error in the ink pen itself.

The square:

Figure 6-26: Square1

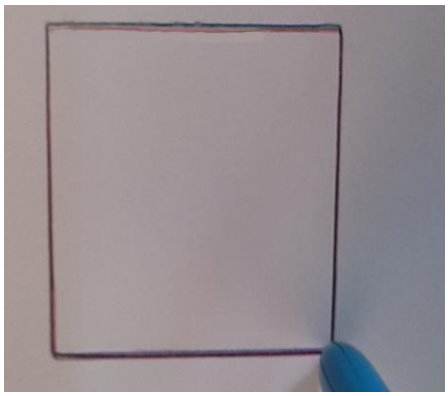


Figure 6-27: Square 2

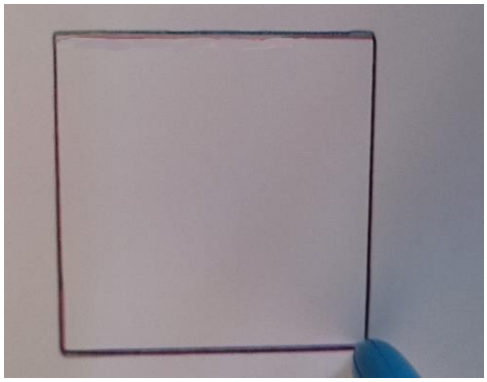


Figure 6-28: Square 3

The Accuracy and the Resolution:

From these tests, the two shapes; the line and the square, were drawn with high accuracy and repeatability. For the line, the accuracy between the three drawn lines was about 0.2 mm. The resolution was about 0.1 mm.

For the squares, between the four experiments, the accuracy was about 0.5 mm, the resolution was about 0.3 mm.

Conclusion

In this project, an H-Bot-based painter was designed and implemented using simple, lightweight mechanical parts. This system is fast and accurate enough for practical applications, such as plotters, painters, engravers and more. Since the system is lightweight, the control system designed in Chapter 3 was not needed for this project; for the motors did not need much torque to move the end-effector; which means no torque control was needed. The image processing algorithm worked greatly with this mechanism, creating highly accurate paths for patterns with high complexity in a significantly low time.

Recommendations and Future Work

For future improvements, we recommend the following:

- This system has no feedback, due to the use of stepper motors. For future improvements, servomotors should be used for larger scales; for they have torque and position feedback systems. Encoders can also be used with the stepper motors. Keep in mind, for larger scales; the stiffness of the timing belt can play a role in the errors; for that reason, in the future dynamic models, the stiffness should be taken into consideration, as well as the position and torque feedback systems.
- For this prototype, the flipping is done manually by the user, due to time constraints. An automatic flipping system can be added to this system for better painting results.
- A spraying system (nozzle) can be added to the project.
- This system can be improved into other forms of CNC machines, such as a laser engraving, drilling, welding, etc.
- Currently, the machine works with only one pattern or object in an image. In the future, this machine can work with multiple patterns in the same image.
- The image processing algorithm does not identify the thicknesses of the patterns. In the future, the algorithm should be able to work with different thicknesses.
- The system in the future should be able to paint in the third dimension. The algorithm should be able to identify the depth of the pattern.

Appendices

Appendix A: The Stepper Motors

Introduction (Definition):

Stepper motors are brushless, synchronous DC motors that move in discrete steps, converting electrical pulses into discrete mechanical movements. In other words, they can divide a full rotation into an expansive number of steps.

The motor's position can be controlled accurately without any feedback mechanism, as long as the motor is carefully sized to the application, which makes it an open-loop control system. With a computer controlled stepping we can achieve very precise positioning and/or speed control. For this reason, stepper motors are the motor of choice for many precision motion control applications.

Principle [19]:

The stepper motor consists of a rotor that is generally a permanent magnet and it is surrounded by the windings of the stator. As we activate the windings step by step in a particular order and let a current flow through them they will magnetize the stator and make electromagnetic poles respectively that will cause propulsion to the motor.

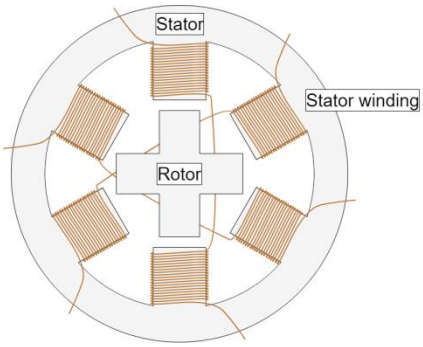


Figure A-1: Cross-Section of a Stepper Motor

By energizing one or more of the stator phases, a magnetic field is generated by the current flowing in the coil and the rotor aligns with this field. By supplying different phases in sequence, the rotor can be rotated by a specific amount to reach

the desired final position. The figure below shows a representation of the working principle. At the beginning, coil A is energized and the rotor is aligned with the magnetic field it produces. When coil B is energized, the rotor rotates clockwise by 60° to align with the new magnetic field. The same happens when coil C is energized. In the pictures, the colors of the stator teeth indicate the direction of the magnetic field generated by the stator winding.

If the angle of a step is smaller, the number of steps will be greater per revolution and so the accuracy of obtained position will be higher. The largest angle can be 90 degrees and small angle will be as 0.72 degrees. Anyway, the most common steps of the angle which are used are 1.8°, 2.5°, 7.5° and 15°.

The shaft rotation direction depends on the pulses sequence applied to the stator. The shaft speed is directly proportional to the input pulses frequency. So, when a frequency is low, the stepper motor rotates in steps and when the frequency is high, it rotates in continuous speed.

Advantages and disadvantages of stepper motors:

Advantages:

- The rotation angle of the motor is proportional to the input pulse.
- The motor has full torque at standstill.
- Precise positioning and repeatability of movement since good stepper motors have an accuracy of 3 – 5% of a step and this error is noncumulative from one step to the next.

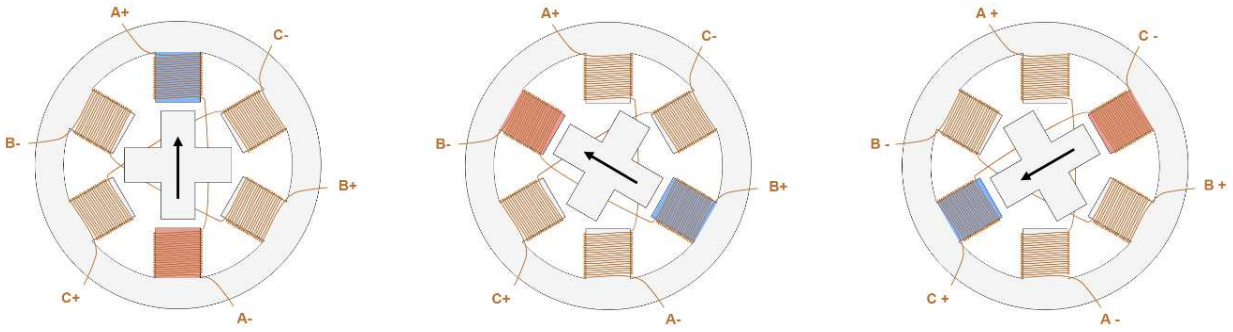


Figure A-2: Stepper Motor Steps

- Excellent response to starting, stopping, and reversing.

- Very reliable since there are no contact brushes in the motor. Therefore the life of the motor is simply dependent on the life of the bearing.
- The motor's response to digital input pulses provides open-loop control, making the motor simpler and less costly to control.
- It is possible to achieve very low-speed synchronous rotation with a load that is directly coupled to the shaft.
- A wide range of rotational speeds can be realized as the speed is proportional to the frequency of the input pulses.

Disadvantages:

- **Low Efficiency:** Unlike DC motors, stepper motor current consumption is independent of load. They draw the most current when they are doing no work at all. Because of this, they tend to run hot.
- **Limited High Speed Torque:** In general, stepper motors have less torque at high speeds than at low speeds. Some steppers are optimized for better high-speed performance, but they need to be paired with an appropriate driver to achieve that performance.
- **No Feedback:** Unlike the servo motors, most steppers do not have integral feedback for position. Although great precision can be achieved running 'open loop'. Limit switches or 'home' detectors are typically required for safety and/or to establish a reference position.

Stepper Driving techniques[20]:

Stepper drives control how a stepper motor operates; there are three commonly used excitation modes for stepper motors, **full step**, **half step** and **microstepping**. These excitation modes have an effect on both the running properties and torque the motor delivers.

1. Full Step Mode: For each 360° rotation of the motor shaft, the rotor proceeds through 200 distinct steps, each exactly 1.8°. During full step operation, two of the phases on the stator are always energized. This provides maximum torque, but angular resolution is limited by the number of teeth on the rotor.

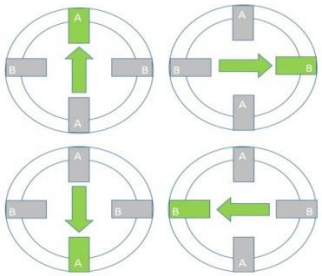


Figure A-3: Full-Step mode1

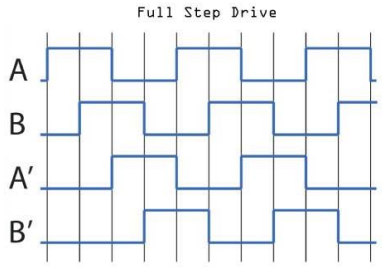


Figure A-4: Full-Step Mode Waves

2. Half-Step mode: For each 360° rotation of the motor shaft, the rotor proceeds through 400 distinct steps, each exactly 0.9°. During half step operation, there is an alternation between having one or two phases on the stator energized. This provides twice the level of angular resolution for increased positioning accuracy but comes at the expense of torque.

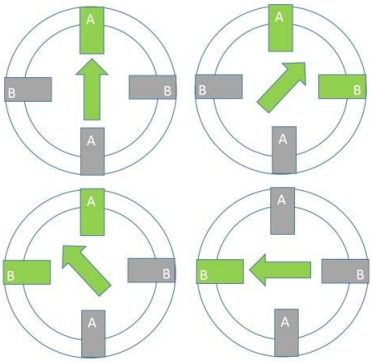


Figure 0A-5: Half Step Mode

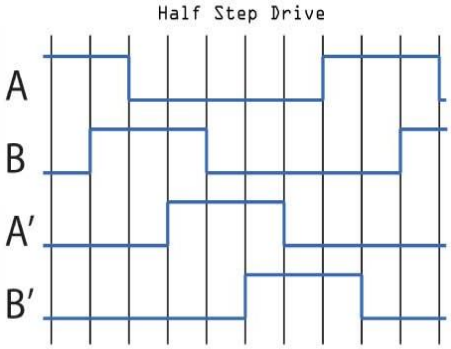
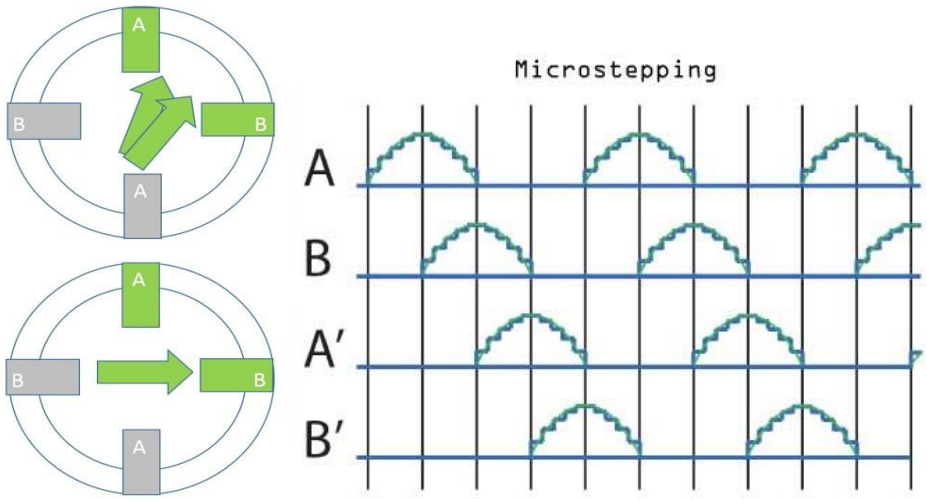


Figure 0-6: Half-Step Mode Waves

3. Microstepping Mode: For each 360° rotation of the motor shaft, the rotor proceeds through 51,200 distinct steps, each exactly 0.007°. During micro-step operation, phases on the stator can be either energized, de-energized or partially energized. This mode is used in applications where highly accurate positioning is needed, although torque rating can be reduced by as much as 30%.



Steps per millimeter[21]:

For the X-Y movement, we will use the following formula to calculate the number of steps per millimeter required for the motion of the motors:

$$Steps = \frac{Motor\ Steps\ per\ Revolution \times \frac{Driver\ microstep}{Belt\ pitch}}{\#\ of\ teeth\ on\ the\ pulley}$$

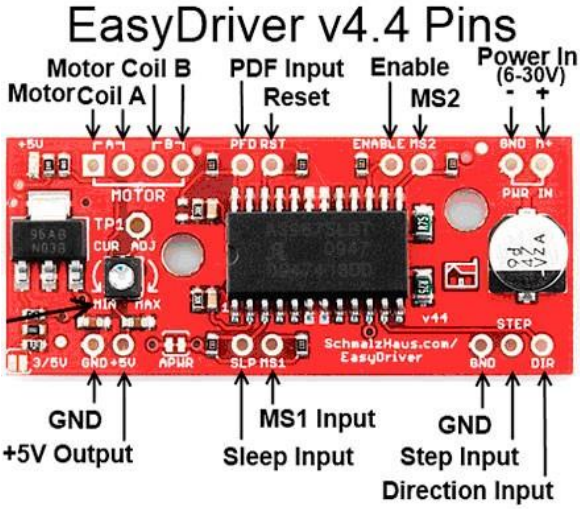
The NEMA 17 stepper motor has a step revolution of 1.8°, so in order to for the shaft to rotate one complete revolution, in full step operation; the number of pulses is calculated as follows: $360° \div 1.8° = 200\ pulses$

The driver of the stepper is on the microstepping mode, which allows the stepper motor in increments of the normal step. For instance, a 1/16th microstepping stepper driver is able to move the stepper motor 16 times for each 1.8° step. The belt pitch is determined by the distance between the teeth on the timing belt. So a T2.5 belt has teeth spaced every 2.5 millimeters. Finally, the number of teeth on the stepper pulley determines the overall gear ratio of the timing-belt drive. So, applying the equation above gives: $Steps = \frac{200 \times \frac{16}{2.5}}{16} = 80\ steps/mm$

Appendix B: The EasyDriver Module: [22]

Introduction:

For this project, we used the EasyDriver V4.4 module. An open-source, low-cost stepper motor module. Each EasyDriver can drive up to about 750mA per phase of a bi-polar stepper motor. It defaults to 8 step microstepping mode. (So if your motor is 200 full steps per revolution, you would get 1600 steps/rev using EasyDriver.) This setting can be easily overridden by tying the MS1 and/or MS2 pin to ground to set the driver to use 1/8, 1/4 or 1/2 microstep mode (See the datasheet for the table of values). It is a chopper microstepping driver based on the Allegro A3967 driver chip. For the complete specs of the design, read the A3967 datasheet. It has a variable max current from about 150mA/phase to 750mA/phase. It can take a maximum motor drive voltage of around 30V, and includes on-board 5V regulation, so only one supply is necessary.



Quick Pin Description:

- **GND**: There are three GND (Ground) pins on the Easy Driver. They are all connected together inside the board. Connect the negative side of your power supply, as well as from any other boards you are using to drive the Easy Driver to one or more of the GND pins.
- **M+**: This is the power input to the Easy Driver. Connect this to the positive power supply lead. This should be a 6V to 30V, 2A (or more) power supply that is clean (low ripple).
- **A and B:** (four pins) These are the motor connections. See below diagrams for how to hook these up. A and B are the two coils of the motor, and can swap the two wires for a given coil (it will just reverse the direction of the motor). This connection to the motor is solid, and not through a connector that has any chance of intermittent contact (which will burn the motor driver chip).
- **STEP**: This needs to be a 0V to 5V (or 0V to 3.3V if you've set your Easy Driver that way) digital signal. Each rising edge of this signal will cause one step (or microstep) to be taken.
- **DIR (Direction)**: This needs to be a 0V to 5V (or 0V to 3.3V if you've set your Easy Driver up that way) digital signal. The level if this signal (high/low) is sampled on each rising edge of STEP to determine which direction to take the step (or microstep).
- **MS1/MS2:** These digital inputs control the microstepping mode. Possible settings are (MS1/MS2) : full step (0,0), half step (1,0), 1/4 step (0,1), and 1/8 step (1,1 : default).
- **RST (reset)**: This normally high input signal will reset the internal translator and disable all output drivers when pulled low.
- **SLP (sleep):** This normally high input signal will minimize power consumption by disabling internal circuitry and the output drivers when pulled low.
- **ENABLE:** This normally low input signal will disable all outputs when pulled high.

- **PFD:** This one is complicated - please see the datasheet for more information. We default it to slow decay mode, but you can over-ride with your own voltage on this pin. (or by populating R17)
- **5V:** This is an OUTPUT pin that will provide either 5V (default) or 3.3V from the voltage regulator, at a small amount of current (say 50mA - depends on input voltage) to power a circuit that you may need powered. If you cut jumper APWR (SJ1) then you can use the 5V pin as a VCC input to the Easy Driver, powering it with your own VCC supply.

Microstepping:

MS1	MS2	Resolution
low	low	Full Step (2 phase)
high	low	Half step
low	high	Quarter step
high	high	Eight step

Appendix C: Coding:

MATLAB Code:

The MATLAB code is used for the control system. Four different controllers were designed; Tracking, Robust Tracking, Discrete-Time Tracking and Discrete-time Robust Tracking.

```
clc

clear all

close all

format long

a1=18.688e-5

a2= 14.652e-5

b1=7.48e-3

b2 = 1.67e-3

%For the state-space representation in the dynamic modeling

E=[1, 0, 0, 0; 0, a1+a2 , 0,  -a2; 0, 0, 1, 0; 0, -a2, 0, a1+a2]

Ei=inv(E)

Q=[0 1 0 0; 0 b1+b2 0 -b2; 0 0 0 1; 0 -b2 0 b1+b2]

A=Ei * Q

U=[0 0; 1 0; 0 0 ; 0 1 ]

B=Ei * U

C=[1 0 0 0; 0 0 1 0]

D=0;

n=4; m=2;
```

```
%controllability
Cm=ctrb(A,B)
cont=rank(Cm)

%Design requirements and desired poles
w1=10; w2=10; z1=0.85; z2=0.95; w3=10; z3=.8;
p1=-z1*w1 + i*w1*sqrt(1-z1^2)
p2=conj(p1)
p3=-z2*w2 + i*w2*sqrt(1-z2^2)
p4=conj(p3)
p5=-z3*w3 + i*w3*sqrt(1-z3^2)
p6=conj(p5)
%For the regulator and the tracking controller
P=[p1 p2 p3 p4]
Kt=place(A,B,P)
Ni=-C*inv(A-B*Kt)*B
N=inv(Ni)
%For the Robust Tracking controller
Ae=[A zeros(n,m); -C zeros(m,m)]
Be=[B; zeros(m,m)]
Pe=[p1 p2 p3 p4 p5 p6]
Ke=place(Ae,Be,Pe)
K=Ke(:, 1:n)
Ki=-Ke(:,n+1:n+m)
```

```
%Digital Tracking
Wmax=10;
T=.05 %sampling time
[F,G]=c2d(A,B,T); %Discrete matrices
Ma=ctrb(F,G) %Controllability matrix
rnk=rank(Ma) %Controllability check
Pcz=exp(P*T) %Converts desired poles to the discrete
Kc=place(A,B,Pcz)
Nk=inv(-C*inv(F-G*K-eye(n))*G)
%Digital Robust-Tracking
Fe=[F zeros(n,m); -C eye(m)]
Ge=[G; zeros(m,m)]
Pez=exp(Pe*T)
Ked=place(Fe,Ge,Pez)
Kd=Ked(:,1:n)
Kid=-Ked(:,n+1:n+m)
```

Appendix D: Simulation and Experiment videos:

You can type these short links on your browser to see the videos mentioned in the chapters four and six.

One-time painting simulation from section 4.11: t.ly/wetA

Pattern 1: t.ly/bns5

Pattern 2: t.ly/MmZU

Window guard plotting from section 6.2.1: t.ly/3jdo

References

1. Lim, H., J.-W. Seo, and C.-H. Choi. *Position control of XY table in CNC machining center with non-rigid ballscrew*. in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*. 2000. IEEE.
2. Hace, A., et al. *Robust motion control of XY table for laser cutting machine*. in *IECON'98. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 98CH36200)*. 1998. IEEE.
3. Phanomchoeng, G. and R.J.E.J. Chanchareon, *Adaptive gain control for a two-axis, H-frame-type, positioning system.* 2017. **21**(3): p. 223-234.
4. Sollmann, K.S., M.K. Jouaneh, and D.J.I.A.t.o.m. Lavender, *Dynamic modeling of a two-axis, parallel, H-frame-type XY positioning system.* 2009. **15**(2): p. 280-290.
5. Yin, M., et al. *Dynamic Modeling and Characterization of the Core-XyCartesian Motion System*. in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. 2018. IEEE.
6. Craig, K.C., R.J.T. Greenheck, and M. World, *So, you want to build an H-bot?* 2011. **31**(5): p. 22.
7. Marion, A., *Introduction to image processing*. 2013: Springer.
8. Madisetti, V., *The digital signal processing handbook*. 1997: CRC press.
9. Ballard, D.H. and C.M.J.J.P.H. Brown, *Computer vision. englewood cliffs.* 1982.
10. West, D.B., *Introduction to graph theory*. Vol. 2. 1996: Prentice hall Upper Saddle River, NJ.
11. Lippman, D. *Euler and Hamiltonian Paths and Circuits*. 2018.
12. Mejia, A. *8 time complexities that every programmer should know*. Adrian Mejia Blog 2019.
13. Lippman, D. *Eulerization and the Chinese Postman Problem*. 2018.
14. Soille, P., *Morphological image analysis: principles and applications*. 2013: Springer Science & Business Media.
15. OpenCV. *Basic concepts of the homography explained with code*. 2018; Available from: https://docs.opencv.org/4.5.2/d9/dab/tutorial_homography.html.
16. Roth, D.G., *Homography*. 2017.
17. Nemeth, J., C. Domokos, and Z. Kato. *Recovering planar homographies between 2D shapes*. in *2009 IEEE 12th International Conference on Computer Vision*. 2009. IEEE.
18. Dong, C.-Z., et al., *Structural displacement monitoring using deep learning-based full field optical flow methods.* Structure and Infrastructure Engineering, 2020. **16**(1): p. 51-71.
19. Fiore, C., *Stepper Motors Basics: Types, Uses, and Working Principles*. 2018.
20. Doujan. *How a Stepper Motor Works*. How to Mechatronics Website 2016.
21. Isra'a Rabbaa, Z.N., *Fused-Deposition-Modeling 3D printer*, in *Mechanical Engineering Department*. 2015, Palestine Polytechnic University.
22. Schmalz, B. *Easy Driver Stepper Motor Driver*. Available from: https://www.schmalzhaus.com/EasyDriver/.