YILDIZ TECHNICAL UNIVERSITY

PEN PLOTTER

Ömer Faruk Bağcı

FINAL REPORT

Project Advisor: Assoc. Prof. Haydar Livatyalı

ISTANBUL, 2024

CONTENTS

REVISION HISTORY IV

SYMBOL LIST 1

LIST OF FIGURES 2

LIST OF TABLES 6

ABSTRACT 7

1. PROJECT DESCRIPTION 8
2. PROJECT OBJECTIVES 9
3. PROJECT SCOPE 10
4. LITERATURE ANALYSIS 11
5. BLACKBOX DIAGRAM 13
6. PROJECT NEEDS 13
7. PROJECT REQUIREMENTS 14
8. QUALITY FUNCTION DEPLOYMENT OF THE PROJECT 16
9. PROJECT SPECIFICATION 16
10. SYSTEM ARCHITECTURE 18
11. DESIGN 20
    - 11.1 Body Desing 21
    - 11.2 Mechanical Calculations 22
        - 11.2.1 Torque Calculations 23
    - 11.3 Comparing Preliminary and Revised Design Parts 26
        - 11.3.1 Mounting Block 26
        - 11.3.2 Profile Bracket 27
        - 11.3.3 Bed Idler Pulley Bracket 28
        - 11.3.4 Bed Stepper Motor Bracket 30
        - 11.3.5 Stepper Motor Mounting Top Plate 31
        - 11.3.6 Idler Pulley Mounting Top Plate 33

11.3.7 Corner L Bracket 34
11.3.8 Bed 35
11.3.9 Pen Holder 37
11.4 Overall Design 38
11.5 Realized Images 41
12. MOTOR SELECTION 45
13. ELECTRICAL SYSTEM DESIGN 46
13.1 Revised Electrical Schematic 46
13.2 Realized Electrical Circuit 49
14. POWER CALCULATION 51
15. STEP MOTOR DRIVER SELECTION 51
16. SENSORS 52
17. COMMUNICATION SYSTEM 52
18. CONTROL SYSTEM 53
19. SOFTWARE DESIGN 53
19.1 Mobile and Desktop Application 54
19.1.1 Mobile Application and Desktop Application User Flow 55
19.1.2 Mobile Application Images 57
19.2 Plotter Software 67
19.2.1 Linux Services 68
19.2.2 Server Program 69
19.2.3 Raster Image to Vector Image Converter 71
19.2.4 Vector Image to Gcode Converter 71
19.2.5 Raspi Hardware Controller 74
REFERENCES 79

REVISION HISTORY

Date	Rev. No	Definition	Author(s)
10/05/2024	1.0	Report Beginning	Ömer Faruk Bağcı
11/05/2024	1.1	Mechanical Design and Torque Calculation was Added	Ömer Faruk Bağcı
12/05/2024	1.2	Abstract, Project Definition, Aim, Scope, Literature Analysis was Added.	Ömer Faruk Bağcı
13/05/2024	1.3	Power Consumption Calculation, Software was Added	Ömer Faruk Bağcı
17/05/2024	1.4	Sensors, Project Management, Technical Requirements Analysis was Added.	Ömer Faruk Bağcı
20/05/2024	1.5	The Report was Edited Together According to Its Integrity.	Ömer Faruk Bağcı
22/05/2024	1.6	Content, Figure List, Table List was Created. Report Page Layout was Edited.	Ömer Faruk Bağcı
24/05/2024	1.7	Motor Selection was Added	Ömer Faruk Bağcı
25/05/2024	1.8	Equations were Named.	Ömer Faruk Bağcı
26/05/2024	2.0	Final Preliminary Report was Completed.	Ömer Faruk Bağcı
28/05/2024	2.1	Changed Titles and Typos were Fixed.	Ömer Faruk Bağcı

SYMBOL LIST

a: $Acceleration$

W: $Weight$

N: $Newton$

$Amps$: $Amper \times Second$

V: $Volt$

A: $Amper$

h: $hour$

W_h: $Watt \times Hour$

π: 3.14

LIST OF FIGURES

Figure 1 – iDraw 2.0 11
Figure 2 – iDraw A3 11
Figure 3 – Axi Draw V3 11
Figure 4 – Line-us 11
Figure 5 – Black Box Diagram 13
Figure 6 – Objective Tree 14
Figure 7 – Quality Function Diagram of The Project 16
Figure 8 – Pen Plotter Subsystems 18
Figure 9 – General System Architecture 19
Figure 10 – Preliminary Design 20
Figure 11 – Revised Design 21
Figure 12 – Cartesian Open Frame Design 21
Figure 13 – Pulley and Load 23
Figure 14 – Preliminary Mounting Block Design 26
Figure 15 – Revised Mounting Block Design 27
Figure 16 – Preliminary Profile Bracket Design 27
Figure 17 – Revised Profile Bracket Design 28
Figure 18 – Preliminary Bed Idler Pulley Bracket Design 29
Figure 19 – Revised Bed Idler Pulley Bracket Design 29
Figure 20 – Preliminary Bed Stepper Motor Bracket Design 30
Figure 21 – Revised Bed Stepper Motor Bracket Design 31
Figure 22 – Preliminary Stepper Motor Mounting Top Plate Design 32
Figure 23 – Revised Stepper Motor Mounting Top Plate Design 32
Figure 24 – Preliminary Idler Pulley Mounting Top Plate Design 33
Figure 25 – Revised Idler Pulley Mounting Top Plate Design 34
Figure 26 – Preliminary Corner L Bracket Design 34
Figure 27 – Revised Corner L Bracket Design 35
Figure 28 – Preliminary Bed Design 36

Figure 29 – Revised Bed Design 36
Figure 30 – Preliminary Pen Holder Design 37
Figure 31 – Revised Pen Holder Design 38
Figure 32 – Isometric View 38
Figure 33 – Front View 39
Figure 34 – Back View 39
Figure 35 – Left View 40
Figure 36 – Right View 40
Figure 37 – Top View 41
Figure 38 – Bottom View 41
Figure 39 – Machine Isometric View 42
Figure 40 – Machine Front View 43
Figure 41 – Machine Back View 44
Figure 42 – Machine Right View 44
Figure 43 – Machine Left View 45
Figure 44 – Preliminary Electrical Circuit of The Plotter Machine 46
Figure 45 – Revised Electrical Circuit Design 47
Figure 46 – PCB 47
Figure 47 – Top View 48
Figure 48 – Bottom View 48
Figure 49 – Realized Top Layer 49
Figure 50 – Realized Bottom Layer 50
Figure 51 – Finished Circuit 50
Figure 52 – A4988 Stepper Motor Driver 52
Figure 53 – Limit Switch 52
Figure 54 – Raspberry Pi Zero 2w 53
Figure 55 - Flow Chart of The Mobile and Desktop Applications 54
Figure 56 - User Flow of The Mobile Application 55
Figure 57 - User Flow of The Desktop Application 56
Figure 58 – Connect to the Plotter 57

Figure 59 – Check Connection 57
Figure 60 – Connection Error 58
Figure 61 – Auto home 59
Figure 62 – Manual Control 59
Figure 63 – Limit Error 60
Figure 64 – Plot Image 61
Figure 65 – Select an Image 61
Figure 66 – Detail Level 62
Figure 67 – Generating Svg 62
Figure 68 – Generated Vector Image 63
Figure 69 – Regenerate Vector Image 63
Figure 70 – Regenerated Vector Image 64
Figure 71 – Gcode Properties 64
Figure 72 – Generate Gcode 65
Figure 73 – Show Generated Gcode 65
Figure 74 – Start Plotting 66
Figure 75 – Auto Home 66
Figure 76 – Plotting 67
Figure 77 - Flow Chart of The Pen Plotter Machine 68
Figure 78 – Server Linux Service 68
Figure 79 – Embedded Linux Service 69
Figure 80 – Base Paths 69
Figure 81 – Check Device Router 70
Figure 82 – Machine Router 70
Figure 83 – Media Router 71
Figure 84 – Image to Svg Conversion Script 71
Figure 85 – Config File Fields 72
Figure 86 – Fill Field State 73
Figure 87 – Example Fill Field Output 74
Figure 88 – Api Header File 75

Figure 89 – Core Header File 1 76

Figure 90 – Core Header File 2 76

Figure 91 – Servo Header File 77

Figure 92 – Step Header File 77

Figure 93 – Utils Header File 78

LIST OF TABLES

Table 1– Pen Plotter Comparison 12

Table 2 – Project Needs 14

Table 3 – Project Requirements 15

Table 4 – Project Specifications 17

Table 5 – Pulley Specifications 22

Table 6 – Mechanical Components Specifications 23

Table 7 - Nema17 HS4401 Specifications 45

Table 8 - Power Consumption of The Components 51

ABSTRACT

Ink or laser for transferring digital drawings or writings onto a sheet of paper printers are used. In some cases, this information is not stored on paper or in other ways. It must be transferred to the surface. In cartography, technical training is required to draw detailed maps for drawing or drawing electronic circuits on PCBs (Printed Circuit Boards) Ink or laser printers are not useful when necessary. For example, to transfer an electronic circuit drawing onto a PCB, ink or laser It must be printed on wax paper using a printer. Then, the circuit drawing on paper is transferred to the PCB using heat. However, while the drawing is transferred onto the PCB using heat, the ink in some areas may not completely transfer to the PCB or the paper may tear. A pen plotter device can be used to solve the problems in this example. With the pen plotter device, drawings can be made on different surfaces easily and with high precision transferable.

There are two types of plotter machines. One of them is electrostatic plotter another one is pen plotter. We decided to design pen plotter because of our project requirements. Raspberry pi zero 2w single board computer will be used in plotter motion control and receiving data that will be plotted. In the motor motion side, Nema 17 step motor and Sg90 servo motor will be used. Our machine will be controlled with mobile or desktop application remotely. For these applications we use JavaScript and TypeScript languages. On the single board computer side we use JavaScript and C languages.

1. PROJECT DESCRIPTION

This project delves into the development and implementation of a Raspberry Pi Zero 2W-controlled Pen Plotter, designed to merge artistic creativity with technological precision. The project incorporates a 2-step motor for bed movement, a servo for pen adjustment, and operates on a 12V power supply. Its control mechanism allows users to interact with the machine effortlessly through a mobile or desktop application via wifi, offering a seamless and user-friendly experience. The mechanical aspect of the plotter employs a belt and pulley system for optimal movement precision.

2. PROJECT OBJECTIVES

The aims of the project can be summarized as follows.

- To transfer data using wireless communication technology
- Designing a simple and understandable interface
- To use current technological opportunities effectively
- Creating high precise plotting machine
- To support various surfaces for plotting
- Creating affordable machine by amateurs

3. PROJECT SCOPE

Pen plotters are technology products that enable drawing with high precission on different surfaces, which are becoming widespread in the market, and are used by businesses in advertising and brochure printing. We aim for amateur-level businesses with wireless use and a quality interface design. Since we aim for amateur users, we aimed for ease at every stage of the product. It can be used in different studies thanks to its easy installation, easy use, affordable maintenance costs and support of different file types.

4. LITERATURE ANALYSIS

The iDraw 2.0 features a working area of 219 mm x 297 mm and supports file formats such as JPEG, PNG, BMP, and SVG. Weighing 5kg, it employs a Cartesian Gantry motion system and is housed in an open case frame.

With a larger working area of 29 7mm x 420 mm, the iDraw A3 also supports JPEG, PNG, BMP, and SVG file formats. Weighing 2.5kg, it shares the Cartesian Gantry motion system and open case frame design with iDraw 2.0.

The AxiDraw V3 offers a working area identical to iDraw 2.0 (219 mm x 297 mm) and exclusively supports SVG files. Weighing 2.2kg, it utilizes a Cartesian Gantry motion system and an open case frame.

Featuring a smaller working area of 60 mm x 90 mm, Line-us supports files through its dedicated app. The weight is undefined, and it stands out with a SCARA motion system within an open case frame.

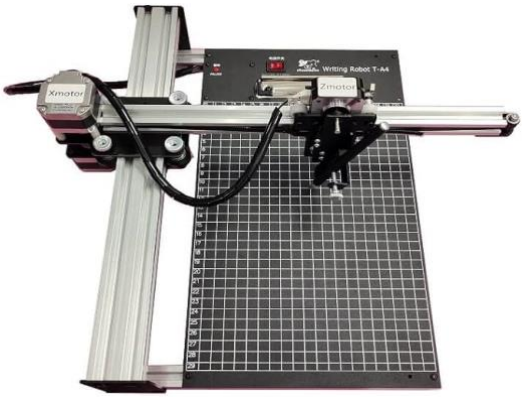

Figure 1 – iDraw 2.0

Figure 2 – iDraw A3

Figure 3 – Axi Draw V3

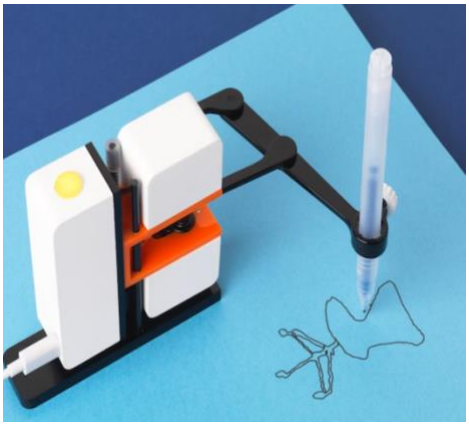

Figure 4 – Line-us

This detailed comparison provides a comprehensive overview of the specifications for each pen plotter, aiding in a nuanced decision based on individual preferences and requirements. The technical specifications of the pen plotters are given in Table 1.

Spec.	iDraw 2.0	iDraw A3	Axi Draw V3	Line - us
Working Area (mm x mm)	219 x 297	297 x 420	219 x 297	60 x 90
File Formats	JPEG,PNG,BMP,SVG	JPEG,PNG,BMP,SVG	SVG	Runs through Line-us app
Weight (kg)	5	2.5	2.2	Undefined
Motion Type	Cartesian Gantry System	Cartesian Gantry System	Cartesian Gantry System	SCARA System
Frame Type	Open Case	Open Case	Open Case	Open Case

Table 1– Pen Plotter Comparison

5. BLACKBOX DIAGRAM

The black box diagram of the system is as follows. The desired plotting part is given as an input to the system. In turn, the part in which the last desired plotting was made is taken as an output by going through the operations in the box.

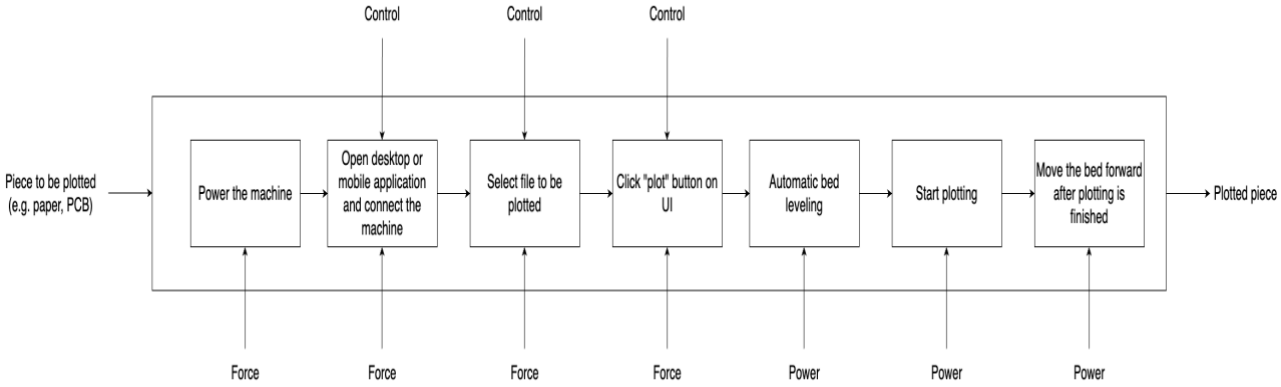

Figure 5 – Black Box Diagram

6. PROJECT NEEDS

While determining the project needs, a study was conducted on what advantages the products have or what problems they have in the literature research. In the fields of architecture, cartography and education, a study was conducted on which features our project can compete with other products in the market. In order for our product to be used in the field of education, our project should be reasonably priced compared to other products in the market. For use in areas such as cartography, architecture, it should be a machine with high precision. Project needs are sorted by their importance. You can see all the project needs below in Table 2.

No	Need	Importance
1	Lightweight	5
2	Can be controlled remotely without requiring a special device	5
3	Lasts a long time	5
4	Support different size of pens	5
5	High moving precision	5
6	Affordable for amateurs	4
7	Plug and play	4

8	No sharp corners	4
9	Adjustable speed	4
10	Sensitivity adjustment	3
11	Easy to assemble	3
12	Easy to install	3
13	Wide plotting area	3
14	Plot from various files	3
15	Can be easily maintained	2

Table 2 – Project Needs

7. PROJECT REQUIREMENTS

The objective tree for the product has been created according to the project needs. Accordingly, the lens tree is divided into 3 main parts. These are quality, cost-effectiveness, and what needs to be achieved.

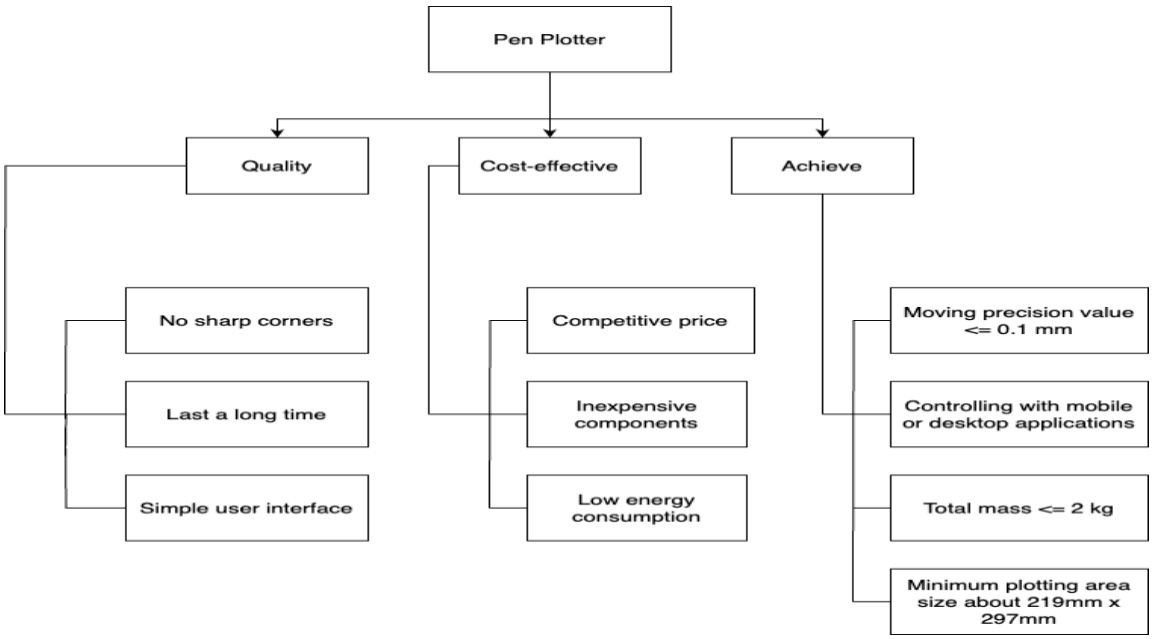

Figure 6 – Objective Tree

Project requirements created according to the project needs and objective tree. The items on the project needs list are not engineering terms so, we have listed the items in the project requirements in a way that complies with engineering terms. You can see the project requirements in Table 3 below.

Metric No.	Need Nos.	Metrics	Importance	Unit
1	1	Total mass	5	kg
2	2	Controlling with mobile and desktop applications	5	List
3	3	MTTF value	5	hr
4	4	Pen gripper diameter	5	mm
5	4	Pen height	5	mm
6	4	Pen diameter	5	mm
7	9, 5	Speed of base plate	4	mm/s
8	9, 5	Step count per full revolution	4	steps/rev
9	6	Unit product cost	4	$
10	7	No complex user interface	4	Subj.
11	7	Power on/off with one mechanical button	4	s
12	8	Buck edge radius	4	mm
13	9, 10	Microstepping range	3	List
14	11, 12, 15	Time to assemble/disassemble	3	s
15	13	Area of plotting base	3	mm^2
16	14	Support common document types	3	List

Table 3 – Project Requirements

8. QUALITY FUNCTION DEPLOYMENT OF THE PROJECT

The quality function diagram of the project is given below. The diagram has been prepared according to the needs and requirements of the project. iDraw 2.0, iDraw A3, axiDraw V3, and line-us have been added as competing products.

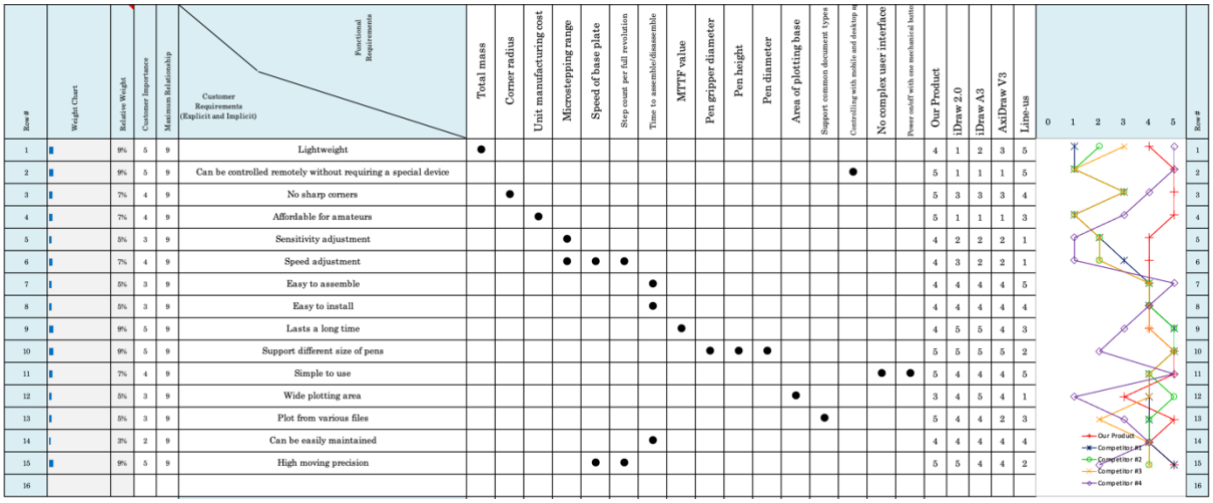

Figure 7 – Quality Function Diagram of The Project

9. PROJECT SPECIFICATION

The project requirements were created using engineering terms, but the values of these terms need to be determined. While determining these values, literature research was conducted, and an attempt was made to determine the ideal value. Marginal values were also determined based on ideal values. Some project requirements were left blank because the ideal values of the terms could not be found. You can see all the project specifications below in Table 4.

No	Need Nos.	Metrics	Imp.	Unit	Margianal Value	Ideal Value
1	1	Total mass	5	kg	< 2	< 1.5
2	2	Controlling with mobile and desktop applications	5	List	Android, Ios, macOs, Windows	Android, Ios, macOs, Windows
3	3	MTTF value	5	hr	20000 - 100000	25000 - 1300000
4	4	Pen gripper diameter	5	mm	-	-

5	4	Pen height	5	mm	-	-
6	4	Pen diameter	5	mm	8 - 12	6 - 20
7	9, 5	Speed of base plate	4	mm/s	0.1 - 3	0.01 – 5
8	9, 5	Step count per full revolution	4	steps/rev	200 - 3200	64 - 6400
9	6	Unit product cost	4	$	< 100	< 80
10	7	No complex user interface	4	Subj.	-	-
11	7	Power on/off with one mechanical button	4	s	-	-
12	8	Buck edge radius	4	mm	-	-
13	9, 10	Microstepping range	3	List	1, 4, 8, 16	1, 4, 8, 16, 32, 64
14	11, 12, 15	Time to assemble/disassemble	3	s	-	-
15	13	Area of plotting base	3	mm^2	148 x 210	219 x 297
16	14	Support common document types	3	List	JPEG, PNG, SVG	JPEG, PNG, SVG, BMP

Table 4 – Project Specifications

10. SYSTEM ARCHITECTURE

As seen in the below, the pen plotter machine consists of two subsystems. The first of these systems is the plotter machine and the other is the pen plotter machine control application. The file you want to plot is selected using the control application. The selected file is sent to the plotter machine and the drawing process is started on the part placed on the surface of the machine.

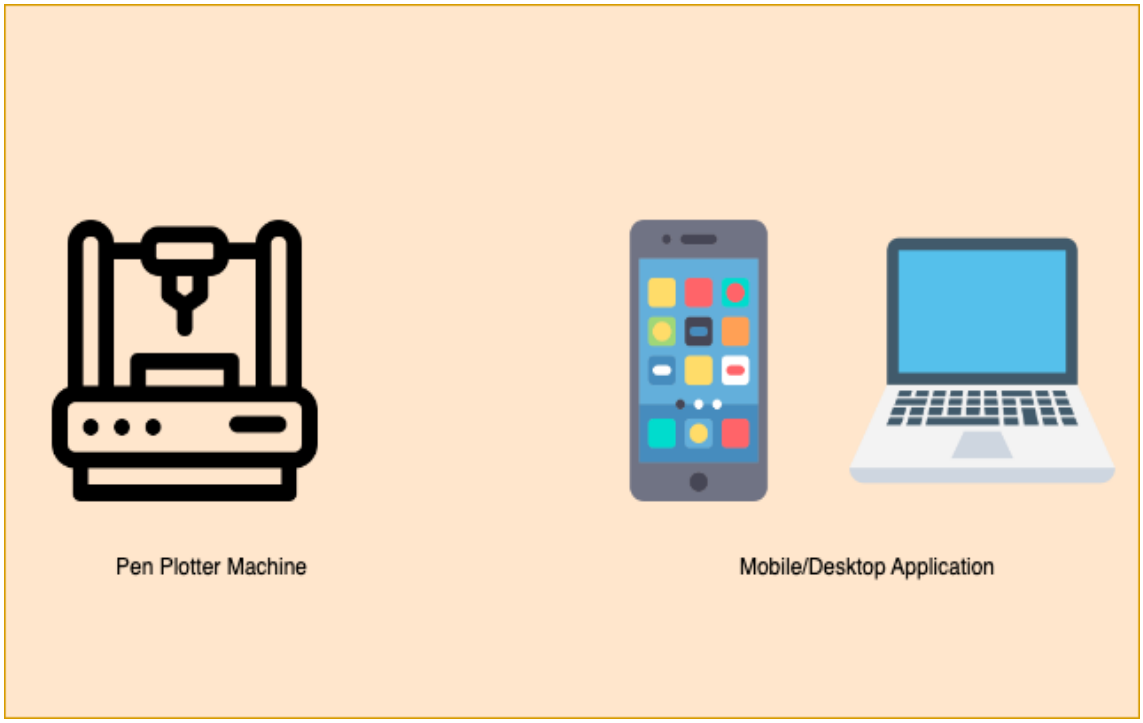

Figure 8 – Pen Plotter Subsystems

Whichever part we want to print on is placed in the device. The file we want to plot is sent via raspberry wifi using the mobile and desktop application. In Raspberry, this file is converted to g-code. After the G-code generation, the necessary signals are sent from the drivers depending on the g-code to the stepper and servo motors. After the G-code processing is completed, the desired file is plotted on the part placed in the machine. You can see the general system architecture in Figure 9.

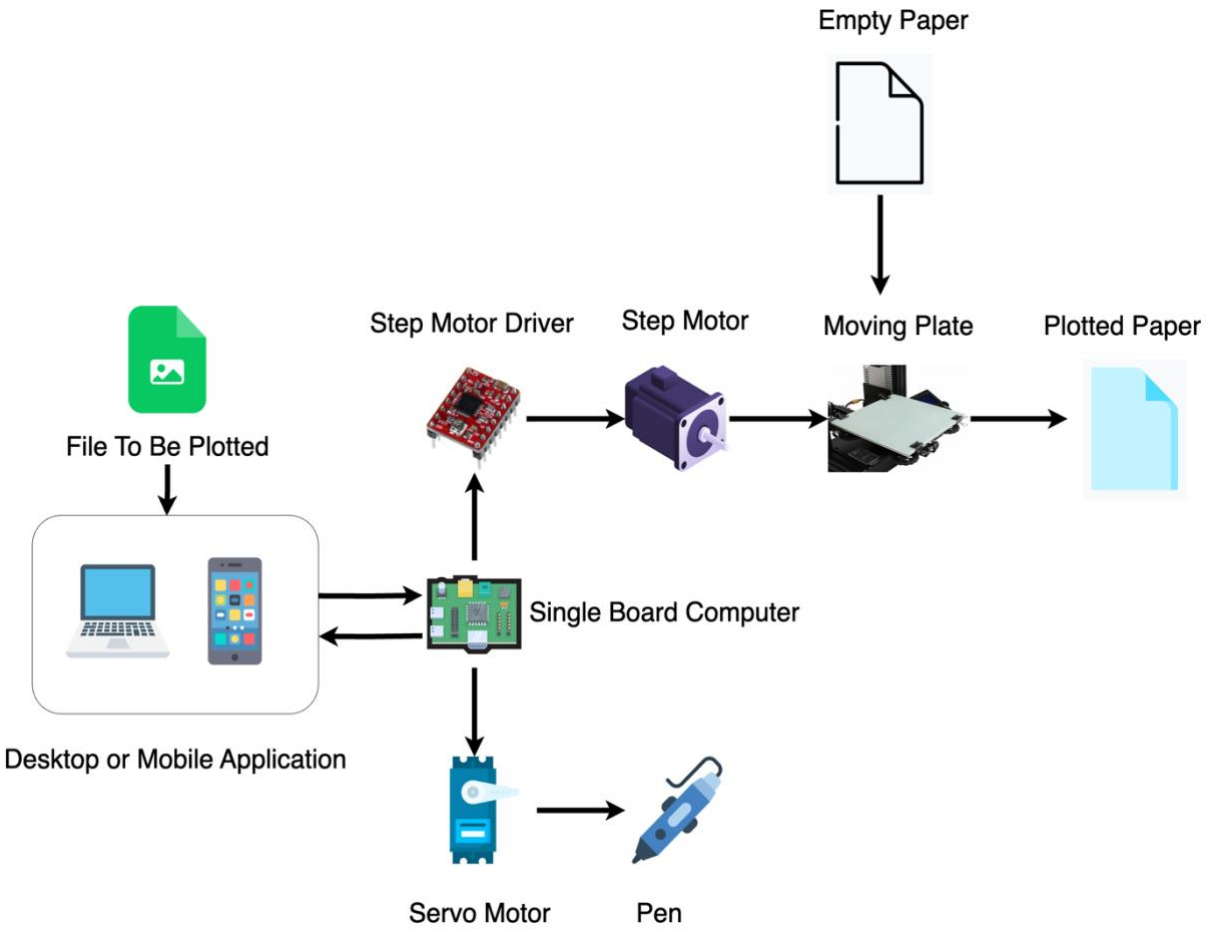

Figure 9 – General System Architecture

11. DESIGN

Pen plotter machine is divided into two subsystems as mobile/desktop application and mechanical part. In the mechanical part, 2 stepper motors were used to move the pen in the desired directions in the x and y directions. The stepper motors are mounted on the bed and pen gripper with a belt system. The belts, on the other hand, receive the movement from the step motor with a pulley. The belts are again supported by a pulley. The pen gripper and the bed both slide on two axles. A servo motor is used to move the pen up and down. Preliminary design can be seen in Figure 10.

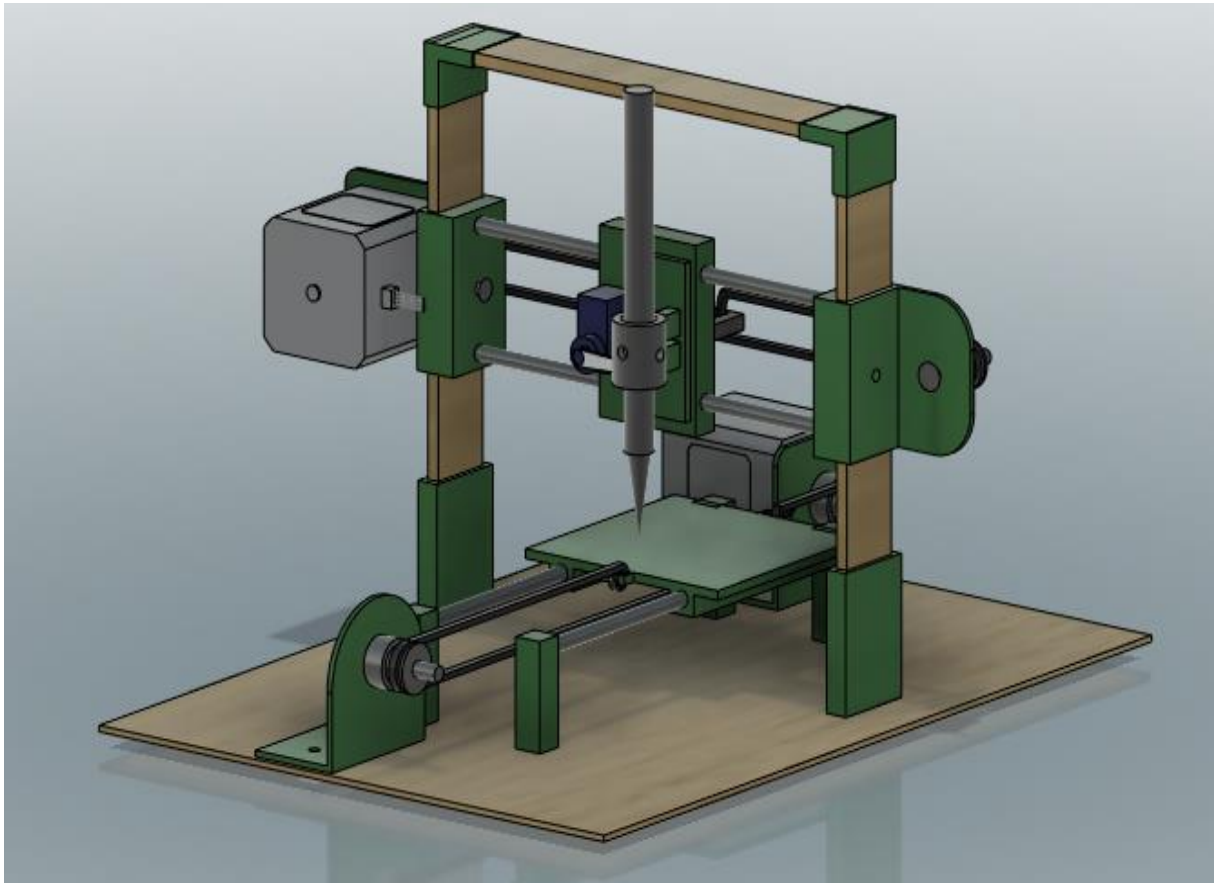

Figure 10 – Preliminary Design

The preliminary design seen in figure 10 above has been completely designed from scratch and finalized as in figure 11.

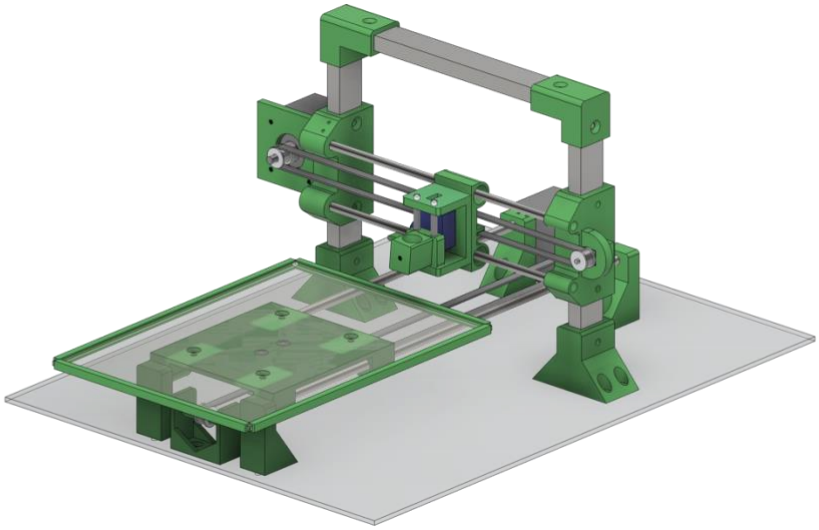

Figure 11 – Revised Design

11.1 Body Desing

There are 3 common type for body design based on motion type. One of them is cartesian gantry system, another one is scara system and the last one is cartesian system. We decided to use cartesian system, so our design for this motion type is open frame.

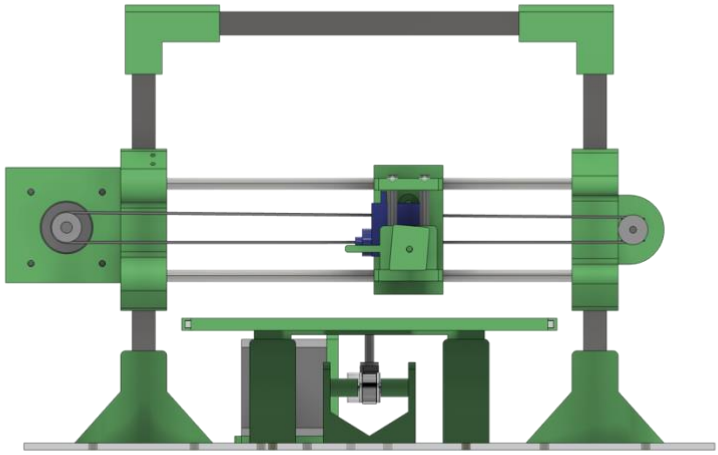

Figure 12 – Cartesian Open Frame Design

11.2 Mechanical Calculations

We determined sensitivity of the plotter machine as 0.01 mm. The operation of the machine with this precision does not depend only on the step motor. Since the machine will move with a belt and pulley system, this sensitivity depends on the pulley, motor angle and microstepping resolution of step driver. When we investigate the stepper motors sold on the market, the step angles range from 0.9° to 5.625°. There are many types of motors on the market for 1.8° step angle. Because of the easy access and stability we selected Nema17 HS4401 step motor. This step motor has 1.8° step angle. We selected the belt that is GT2 timing belt which has width of 6 mm and GT2 16 teeth pulley. Tooth pitch of the pulley is 2 mm.

Specifications	Value
Number of teeth	16
Pitch	2 mm
Circumference	16 x 2 = 32 mm

Table 5 – Pulley Specifications

The belt will move at a distance equivalent to 32 mm per rotation. Motor takes 1.8° per step. One step will move plotter by a distance of $(32\ mm/360°)\ x\ 1.8° = 0.16\ mm$. For 0.01 mm distance we selected the motor driver which has 1/16 microstepping resolution. Using 1/16 microstepping operation one step will become (1.8°/16) = 0.1125°. One step will move plotter by a distance of 32mm/360° x 0.1125° = 0.01mm.

11.2.1 Torque Calculations

We have to check torque of the motor is enough for pencil gripper mechanism and bed.

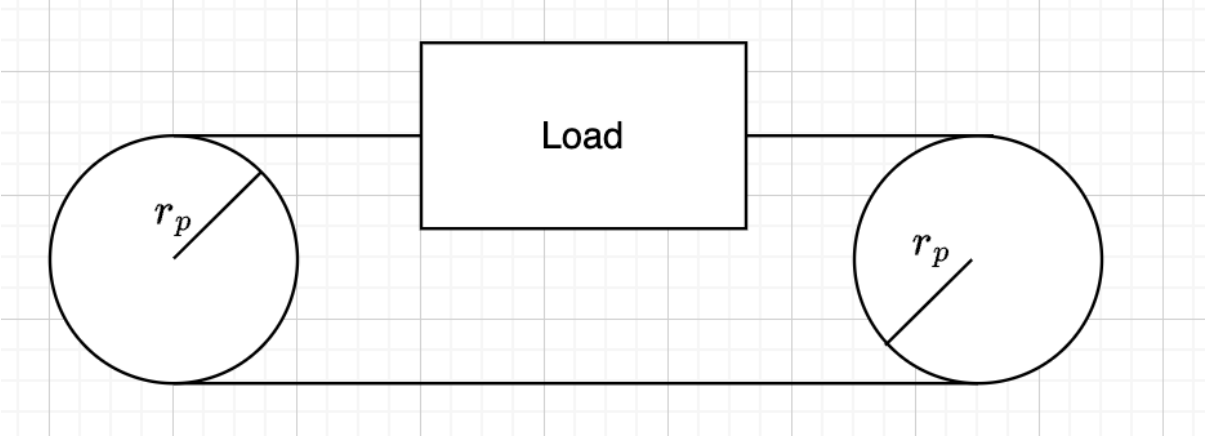

Figure 13 – Pulley and Load

Required information are listed Table 6 below.

Specification	Value
Load weight (m_l)	$300g = 300x10^{-3}kg$
Belt weight (m_b)	$12g = 12x10^{-3}kg$
Pulley weight (m_p)	$5g = 5x10^{-3}kg$
Inertia of the motor (J_m)	$54gcm^2 = 5.4x10^{-6}kgm^2$
Outer diameter of the pulley (r_o)	$12mm = 12x10^{-3}m$
Inner diameter of the pulley (r_i)	$5mm = 5x10^{-3}m$

Table 6 – Mechanical Components Specifications

First calculate the inertia of the pulley.

$$J_p = \frac{1}{2} m_p (r_o^2 + r_i^2) \qquad \text{(Eq. 11.1)}$$

where,

$J_p \gg Inertia\ of\ the\ pulley$

$m_p \gg Weight\ of\ the\ pulley$

$r_o \gg Outer\ diameter\ of\ the\ pulley$

$r_i \gg Inner\ diameter\ of\ the\ pulley$

$$J_p = \frac{1}{2} x(5x10^{-3})(12^2 + 5^2)x10^{-6} = 0.423\ x10^{-6}\ kgm^2 \qquad \text{(Eq. 11.2)}$$

For load calculation there are two load for two belt system pen gripper section and bed. Because of the weight of the bed is higher than pen gripper, we selected load as weight of the bed. Also extra weight is included that is weight of the belt.

$$J_l = (m_b + m_l)x(r_o^2) \qquad \text{(Eq. 11.3)}$$

where,

$J_l \gg Inertia\ of\ the\ load$

$m_b \gg Weight\ of\ the\ belt$

$m_l \gg Weight\ of\ the\ load$

$r_o \gg Outer\ diameter\ of\ the\ pulley$

$$J_l = (12x10^{-3} + 300x10^{-3})x(12^2)x10^{-6} = 44.928x10^{-6}\ kgm^2 \qquad \text{(Eq. 11.4)}$$

Now, we can calculate total inertia of the system. Total inertia of the system summation of the inertia of the motor, inertia of the pulleys and inertia of the load.

$$J_t = J_m + J_{p1} + J_{p2} + J_l \quad \text{(Eq. 11.5)}$$

where,

$J_t \gg Total\ inertia\ of\ the\ system$

$J_m \gg Inertia\ of\ the\ motor$

$J_{p1} \gg Inertia\ of\ the\ pulley\ 1$

$J_{p2} \gg Inertia\ of\ the\ pulley\ 2$

$J_l \gg Inertia\ of\ the\ load$

$$J_t = (5.4 + 0.423 + 0.423 + 44.928)x10^{-6} = 51.174x10^{-6} kgm^2 \quad \text{(Eq. 11.6)}$$

We can calculate the required torque with total inertia and angular acceleration of the motor. Low angular acceleration is enough for this machine. So, it is selected $0.1\frac{rad}{s^2}$. Using equation below we can calculate the torque value.

$$T_{acc} = J_t\, x\, \alpha \quad \text{(Eq. 11.7)}$$

where,

$T_{acc} \gg Acceleration\ torque$

$J_t \gg Total\ inertia\ of\ the\ system$

$\alpha \gg Angular\ acceleration\ of\ the\ motor$

$$T_{acc} = (51.174x10^{-6})x0.1 = 5.1174x10^{-6}\ Nm \quad \text{(Eq. 11.8)}$$

The detent torque (acceleration torque) of the motor is given that 2.2 N.cm as maximum. Which is equal to $22x10^{-3}\ Nm$. Our calculated torque value is so smaller than the maximum torque of the motor, so this motor can be used in this project.

11.3 Comparing Preliminary and Revised Design Parts

In this section, you will see the differences between the previous design and the new design. All parts of the machine have been redesigned due to strength problems, balance problems or installation problems.

11.3.1 Mounting Block

Since these mounting blocks are thin, they will not be able to lift it when the axles are placed in the holes. The mounting blocks have been redesigned because they contain one screw hole, which will cause them to rotate around themselves, and the chassis is not suitable for fixing to the chassis.

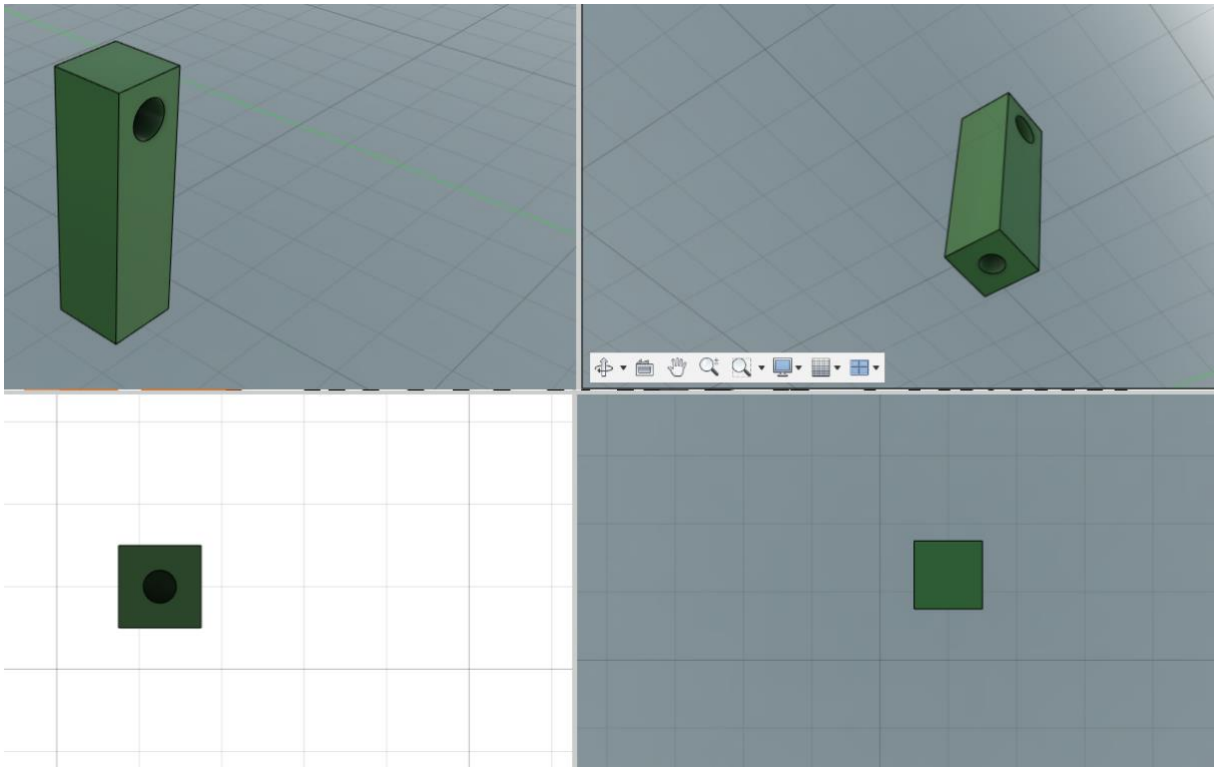

Figure 14 – Preliminary Mounting Block Design

In the new design, the number of screw holes has been increased to 2 to prevent it from rotating around itself. In order to make the installation easier, one of the screws is designed to be from the top and one from the bottom. The thickness of the wall has been increased to be durable. The axle hole has been enlarged according to the new design.

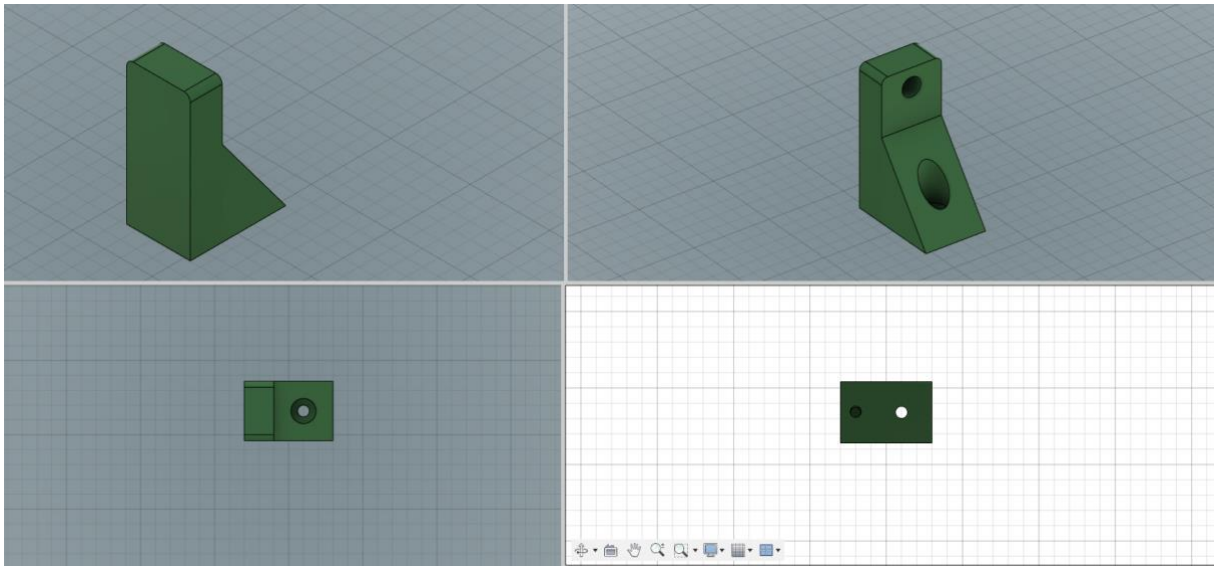

Figure 15 – Revised Mounting Block Design

11.3.2 Profile Bracket

The profile bracket is designed to hold the poplar board in the initial design. Since steel profile is preferred instead of poplar board in the new design, wall thickness and hole dimensions are not suitable. The screw holes are 2, and this will not be enough for the steel profile.

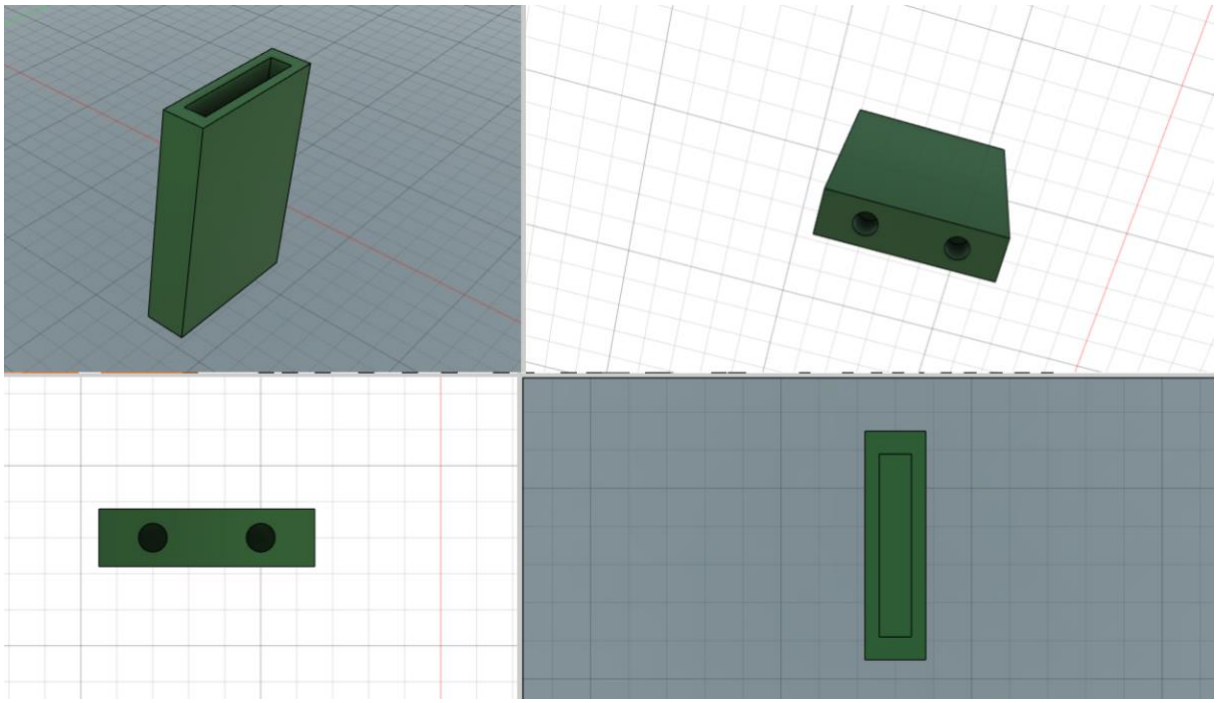

Figure 16 – Preliminary Profile Bracket Design

In the new design, the hole size has been enlarged to match the steel profile. The wall thickness has been increased for strength. The screw holes are designed to be from the top for ease of installation and more stable fastening to the chassis. A screw hole has also been added to the side of the profile bracket so that the steel profile does not become dislodged if the machine is held from above.

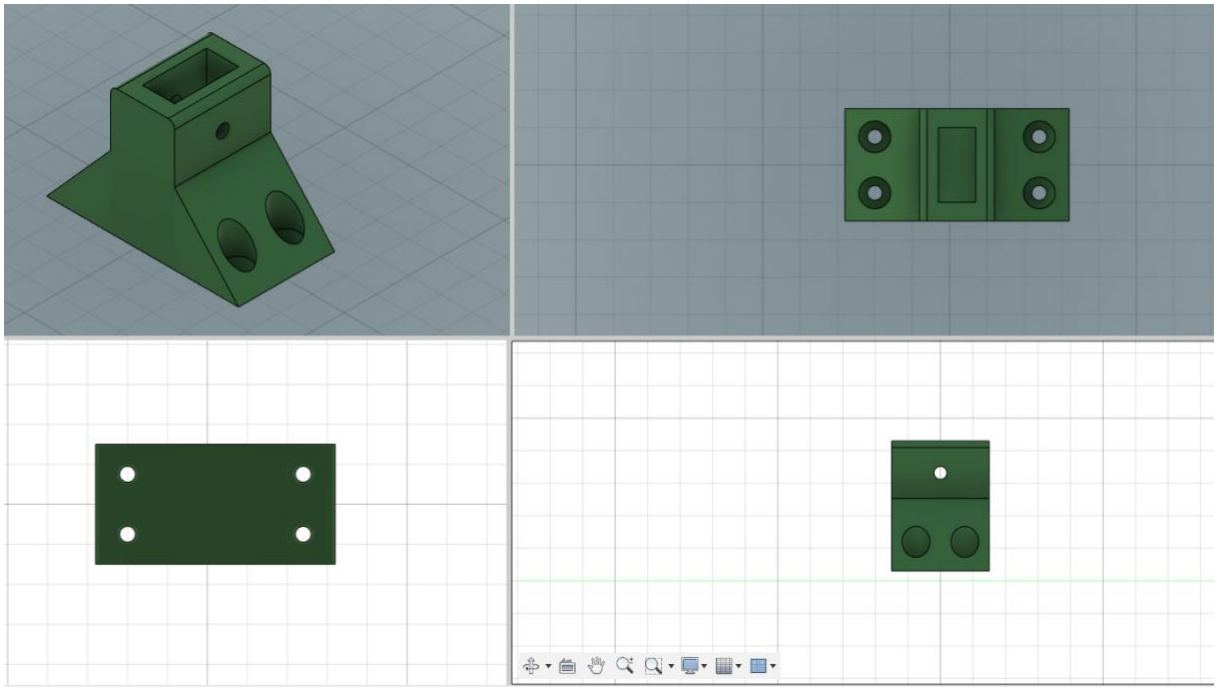

Figure 17 – Revised Profile Bracket Design

11.3.3 Bed Idler Pulley Bracket

The idler pulley bracket is weak because the wall thickness of the holder is thin. The idler pulley bracket cannot resist bending because it provides one-sided grip.

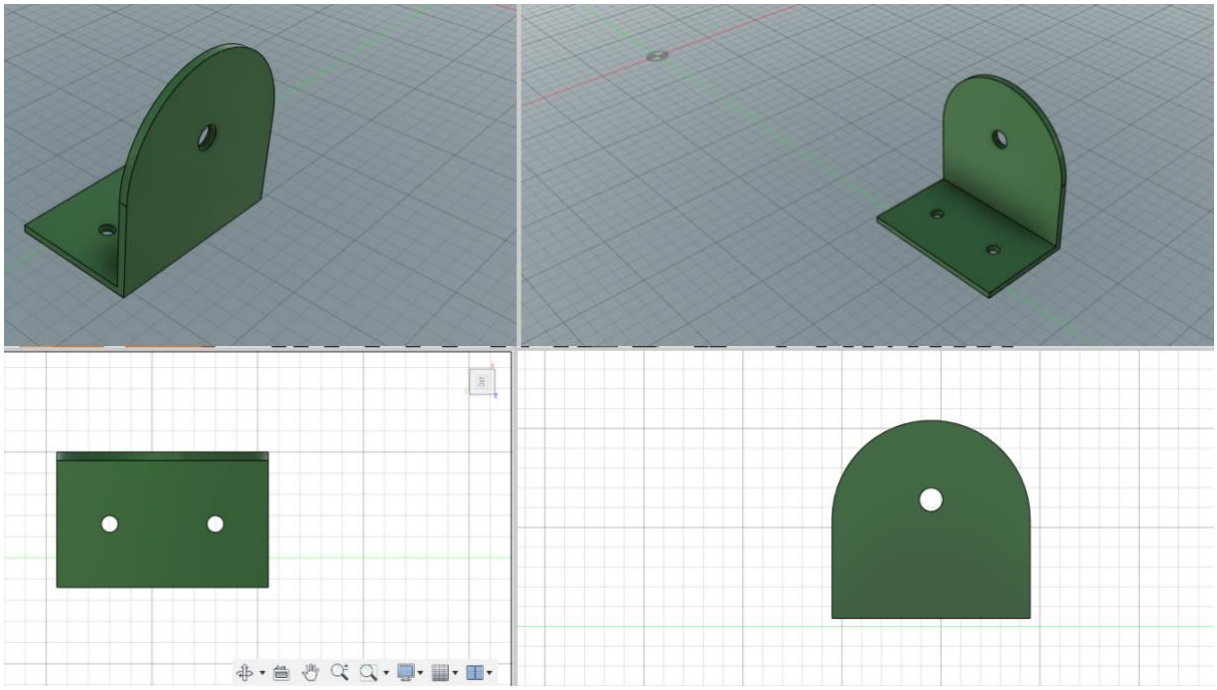

Figure 18 – Preliminary Bed Idler Pulley Bracket Design

In the new design, the idler pulley breacket has been updated to provide a double-sided grip. The part strength has been increased by increasing the wall thickness. Rib is added to prevent the part from tilting inward during the belt movement.

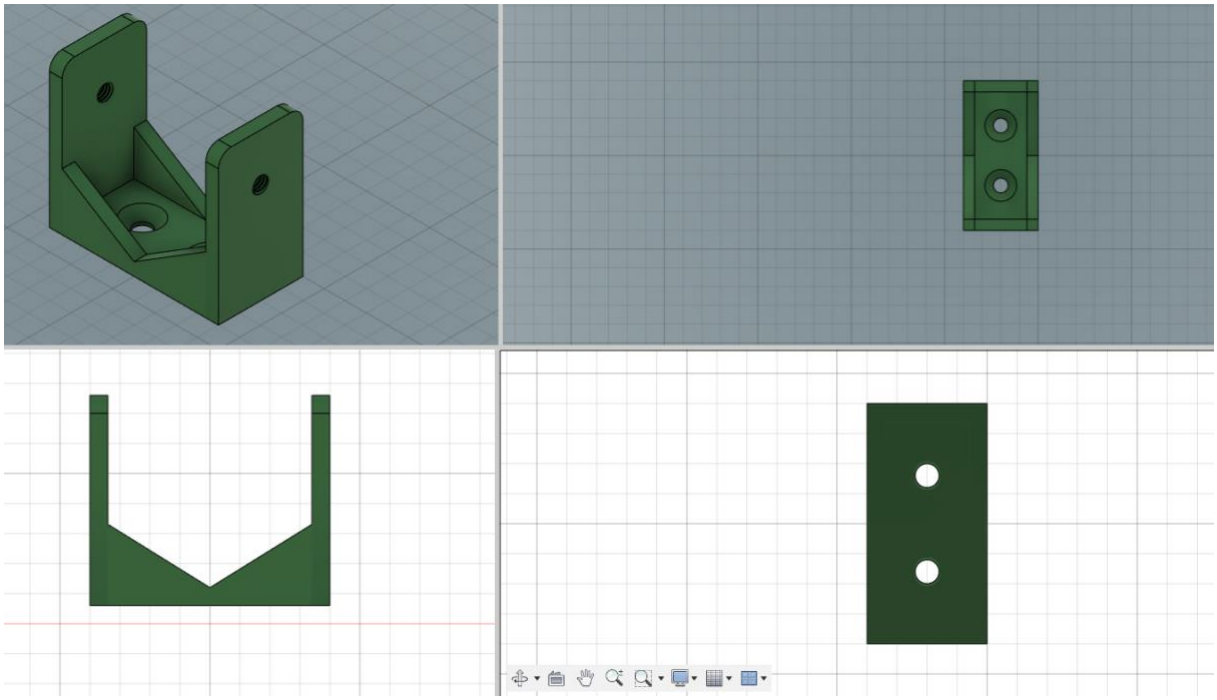

Figure 19 – Revised Bed Idler Pulley Bracket Design

11.3.4 **Bed Stepper Motor Bracket**

In the old design, the stepper motor housing is not suitable for the selected stepper motor. Since the structure is thin, belt movements will lead to bending of the part. There are not enough screw holes in the screw to fix it to the chassis.

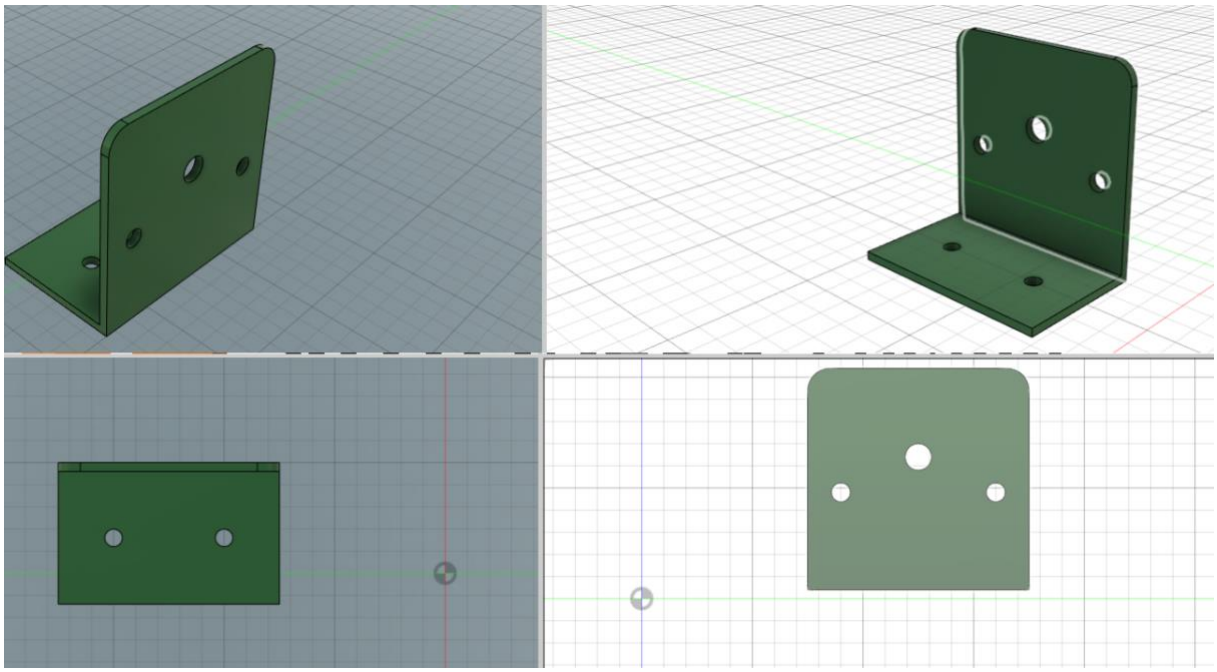

Figure 20 – Preliminary Bed Stepper Motor Bracket Design

In the updated design, an stepper motor housing has been added to the stepper motor bracket to match the stepper motor. Screw hole is added to the 4 corner to fix the stepper motor to the part. 4 screw holes have been added to hold the part firmly to the chassis. The wall thickness has been increased to prevent bending.

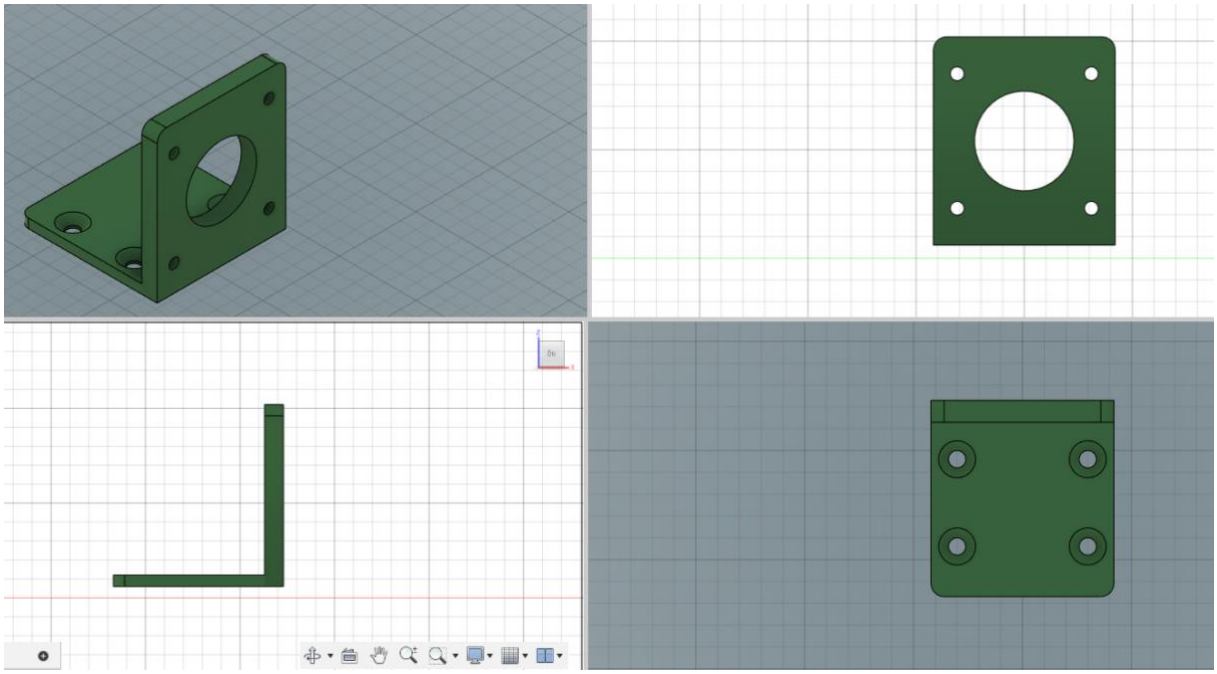

Figure 21 – Revised Bed Stepper Motor Bracket Design

11.3.5 Stepper Motor Mounting Top Plate

Since the previous design was designed to be screwed to the poplar board, it is not suitable for the steel profile used in the new design. There is no part that supports the stepper motor from below. The size of the axle slots is not for the new axle size. The structure is not resistant to bending during belt movement.

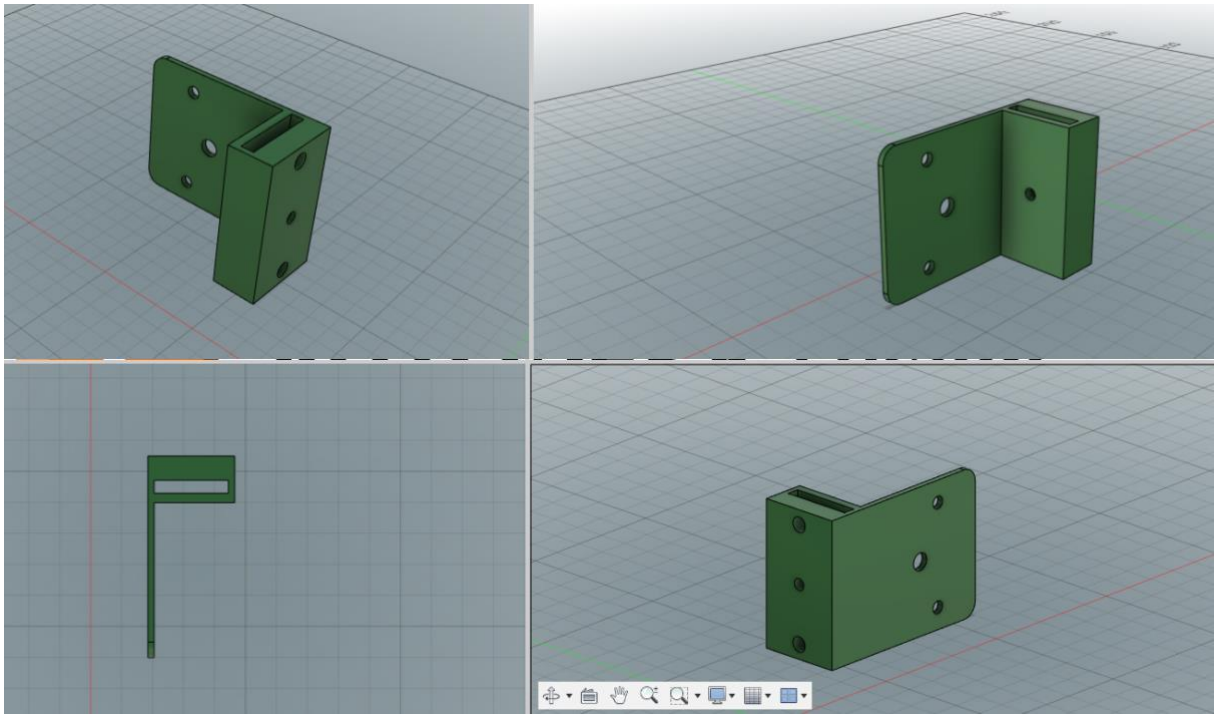

Figure 22 – Preliminary Stepper Motor Mounting Top Plate Design

The hole size has been changed so that the steel profile will fit. The hole is designed according to the new axle size. An addition has been made to support the stepper motor from below. A screw hole has been added for the placement of the limit switch.

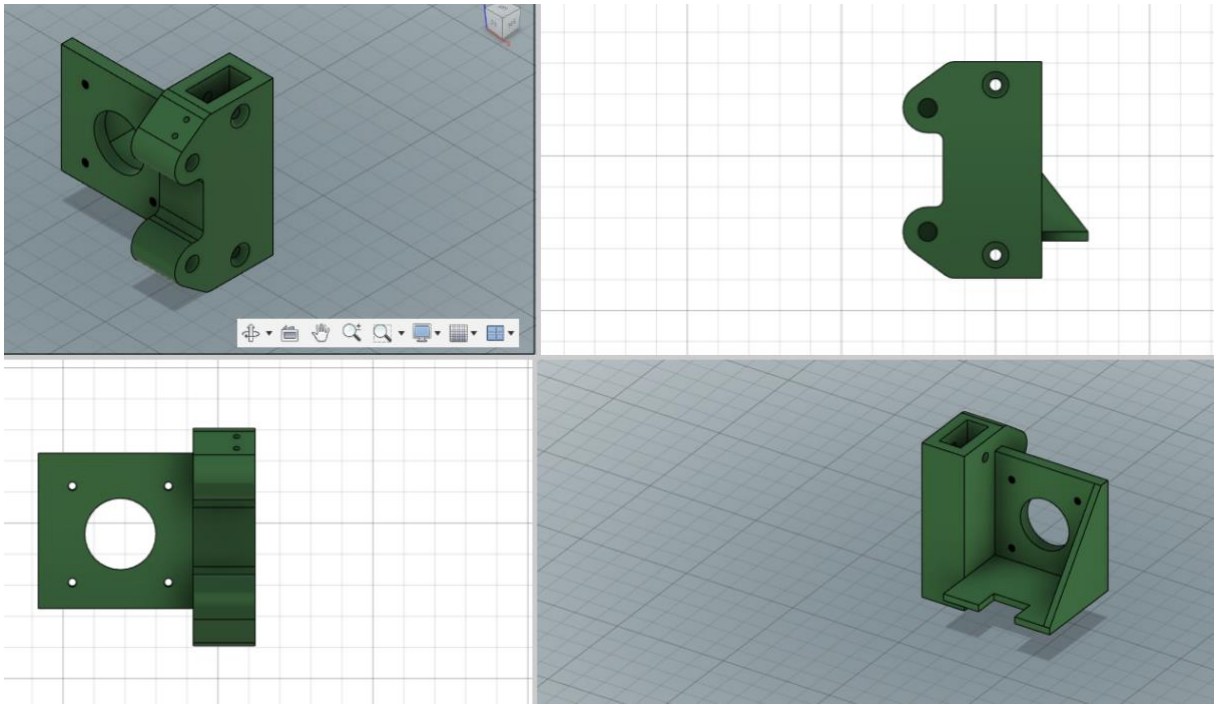

Figure 23 – Revised Stepper Motor Mounting Top Plate Design

11.3.6 Idler Pulley Mounting Top Plate

Since the previous design was designed to be screwed to the poplar board, it is not suitable for the steel profile used in the new design. The size of the axle slots is not for the new axle size. The structure is not resistant to bending during belt movement.

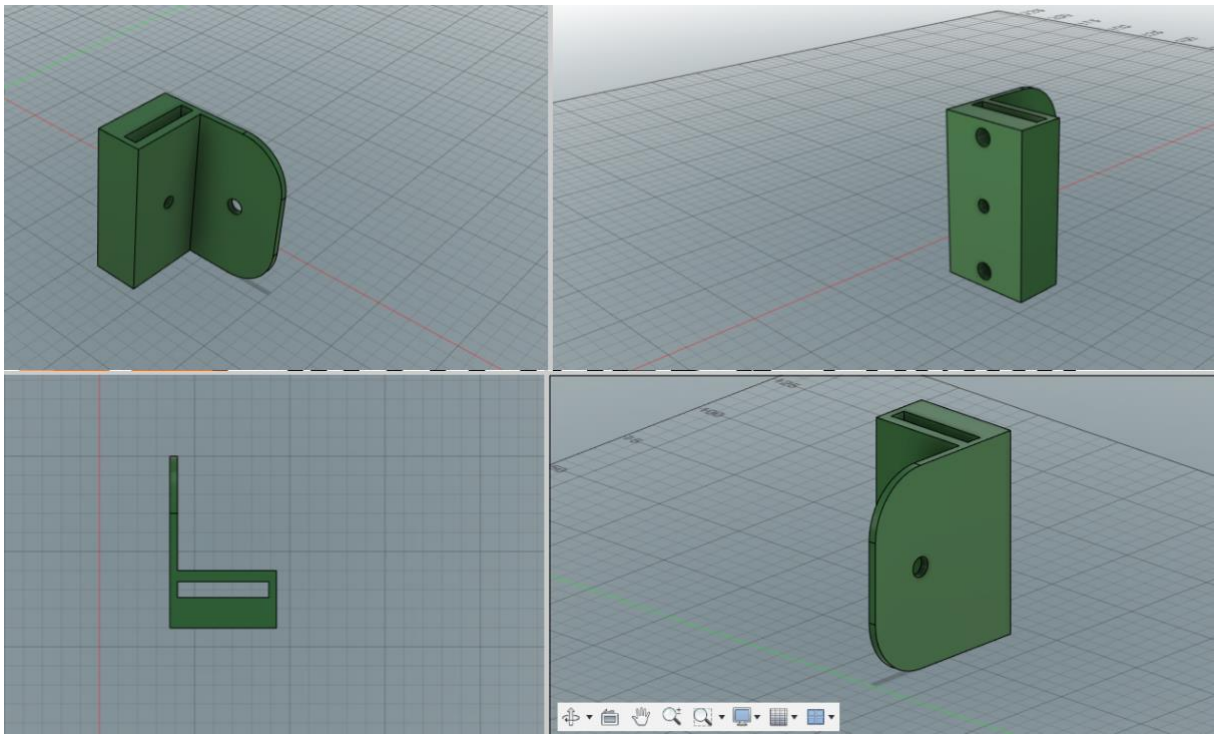

Figure 24 – Preliminary Idler Pulley Mounting Top Plate Design

The hole size has been changed so that the steel profile will fit. The hole is designed according to the new axle size. A square slot has been added in case the spindle size of the idler pulley changes. A part that will hold axle up to 8mm thick can be placed in this slot.

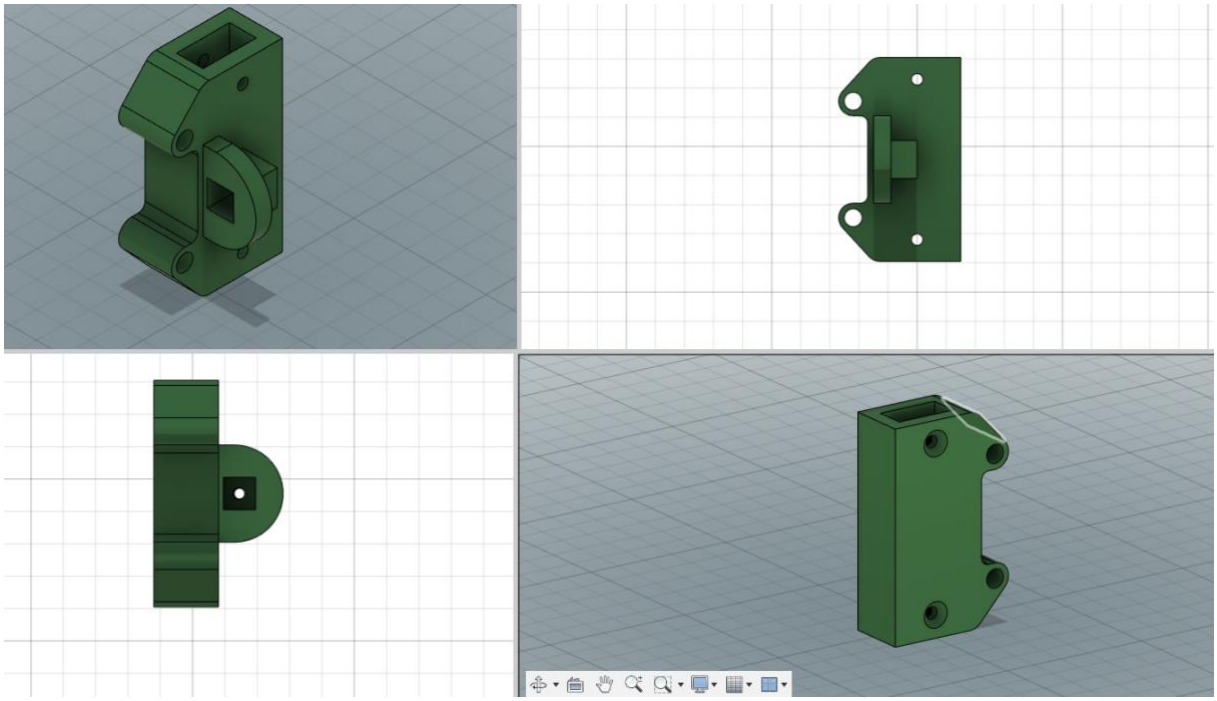

Figure 25 – Revised Idler Pulley Mounting Top Plate Design

11.3.7 Corner L Bracket

Since the previous design was designed to be attached to the poplar board, it is not suitable for the steel profile used in the new design. There are no screw holes.

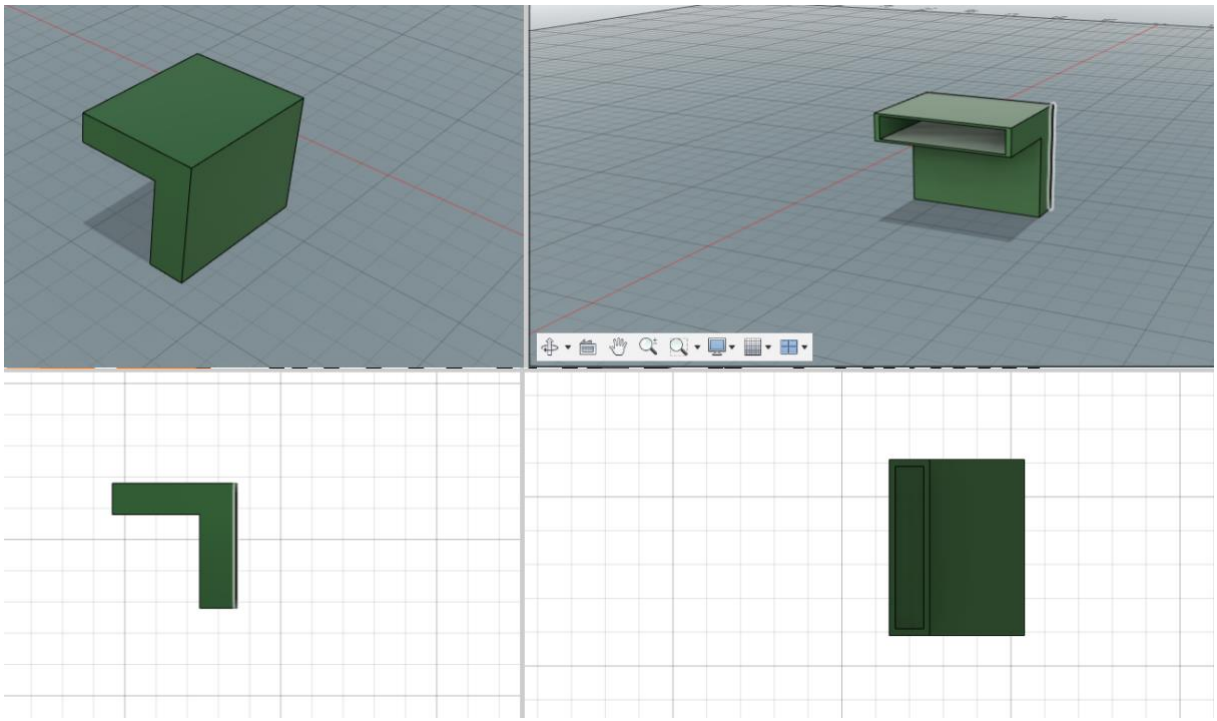

Figure 26 – Preliminary Corner L Bracket Design

The new design is designed to fit the steel profile. Screw holes have been added to keep the profile intact. The edges of the piece are rounded so as not to be sharp.

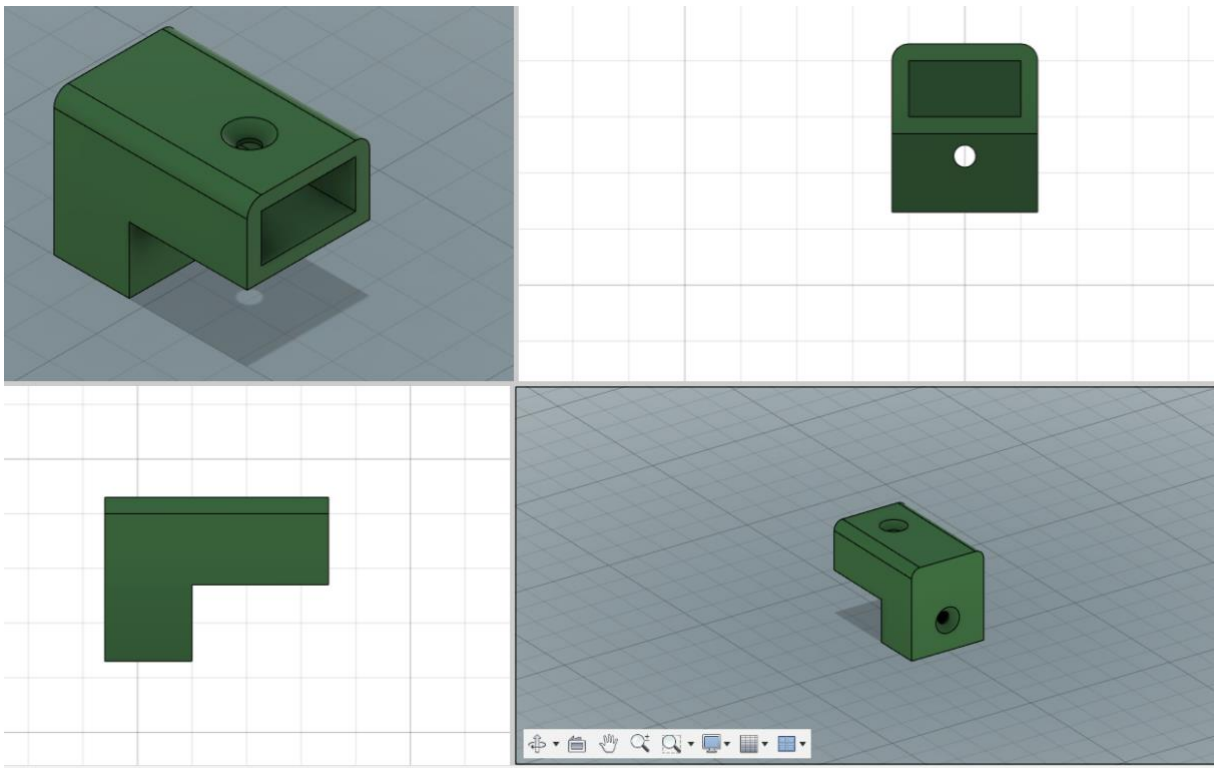

Figure 27 – Revised Corner L Bracket Design

11.3.8 **Bed**

In the old design, the fastening places of the belt are weak and resistant to tensile stress. The bearing hole is designed to slide directly on the axles, but this increases the roughness a lot. there is no hole for the plexiglass part to be connected to it.

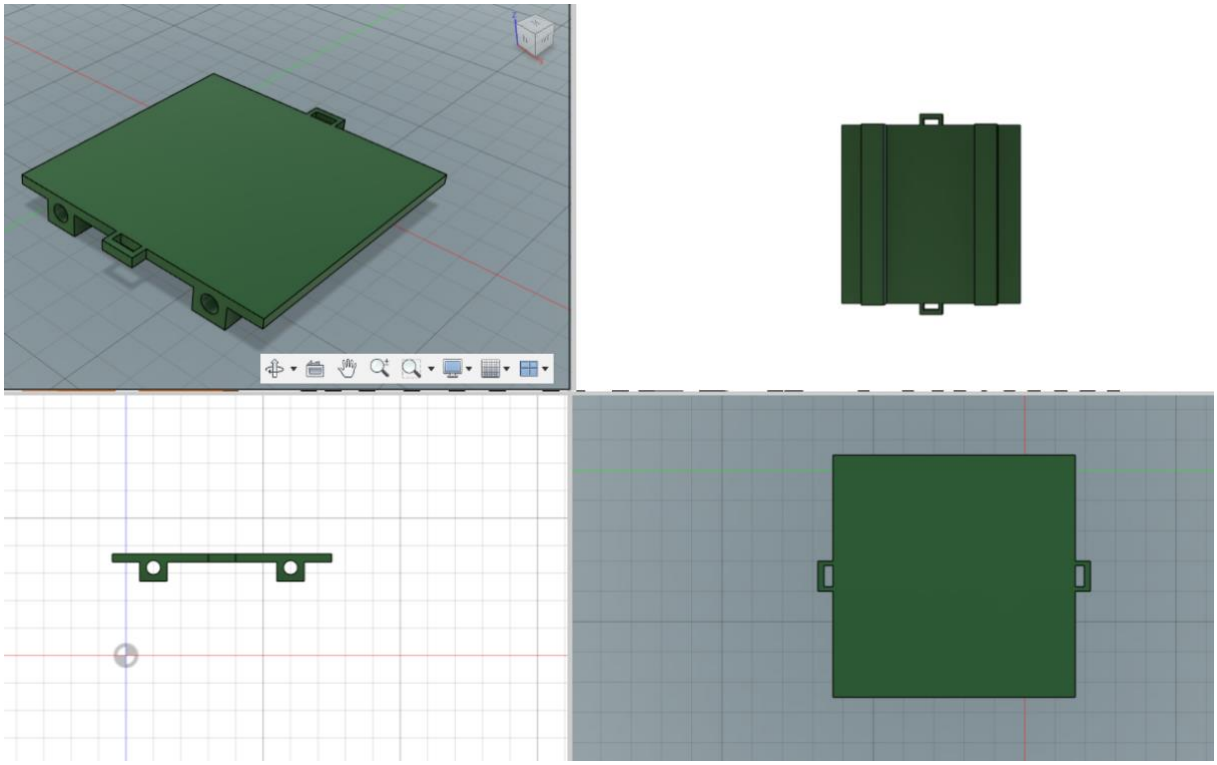

Figure 28 – Preliminary Bed Design

Holes have been added to the part to attach the belt to the lower part of the part. Linear bearing holes have been added instead of the old axle holes, and the new axles have been placed inside these bearings. Screw holes are added to fix the plexiglass part.

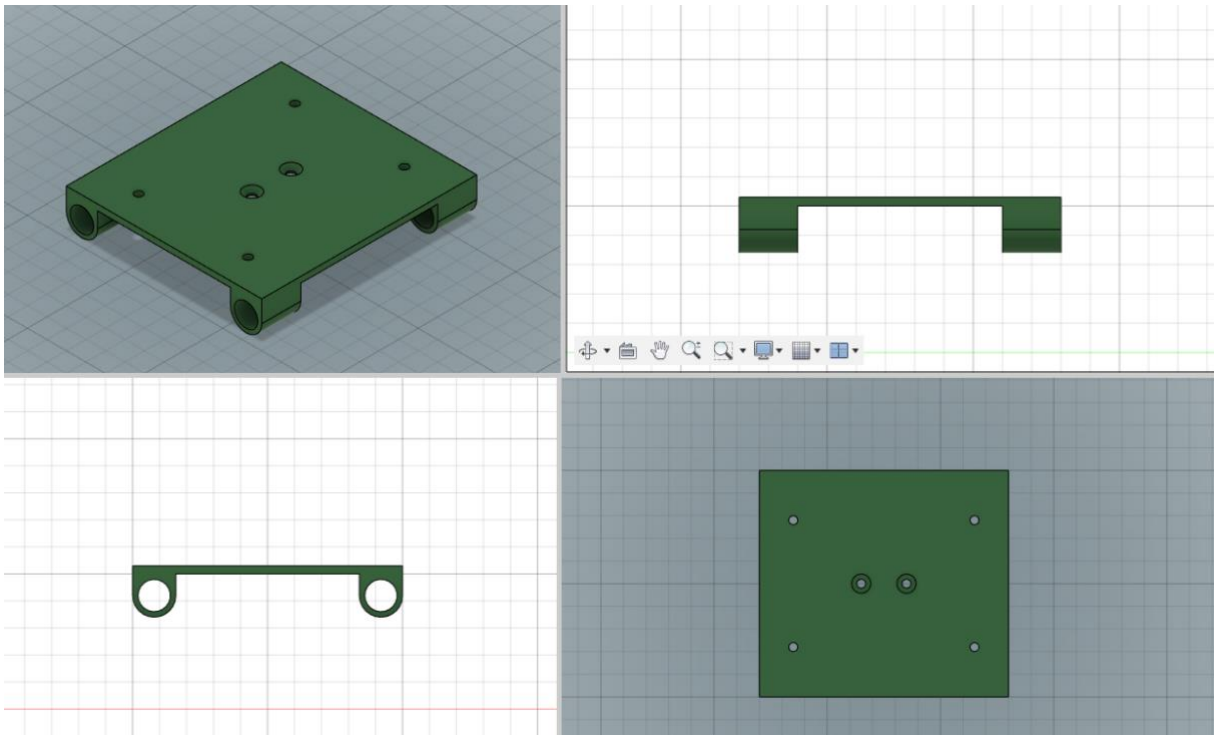

Figure 29 – Revised Bed Design

11.3.9 **Pen Holder**

In the old design, there is no spring that will make the effect to move the pen from the up position to the down position. The belt fastening part is not resistant to tensile stress. The part slides directly on the axles, which causes a lot of friction.

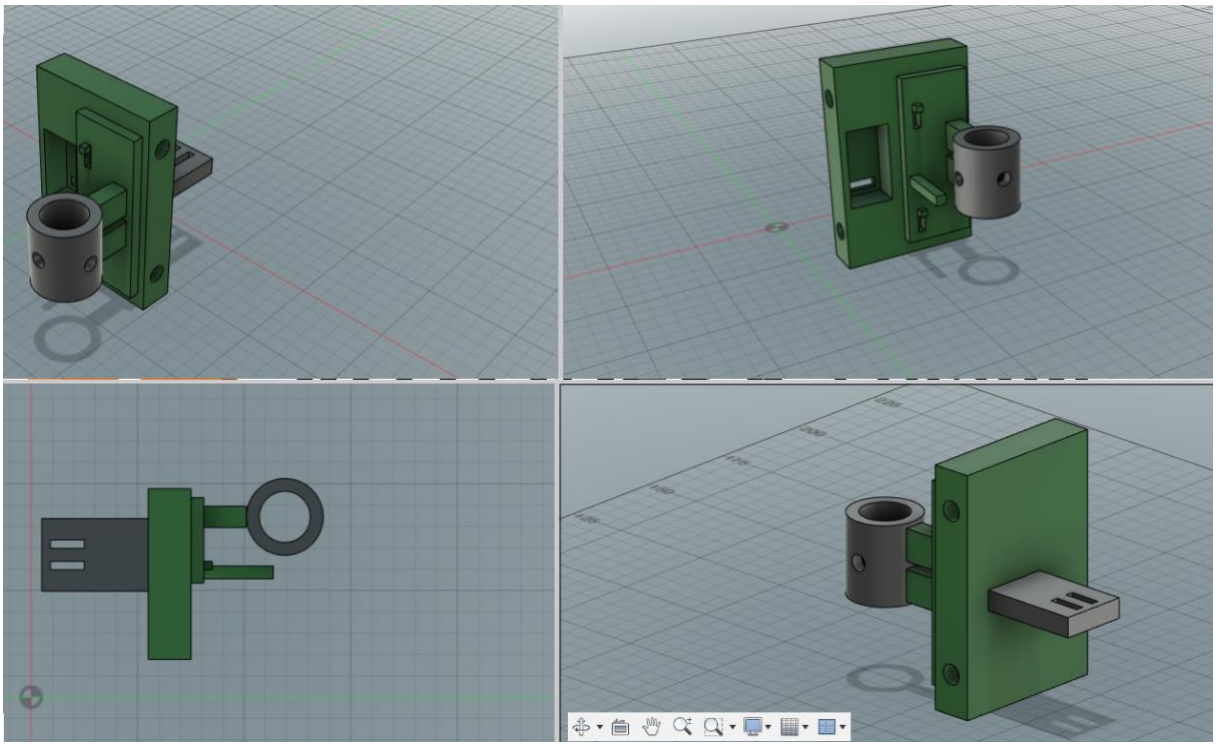

Figure 30 – Preliminary Pen Holder Design

In the new design, slots have been added for the axle that will hold the spring that will move the pen from the up position to the down position. The belt fastening location has been changed. Linear bearing holes have been added instead of axle holes, so the part moves on the axle using rollers and unnecessary friction losses have been eliminated.

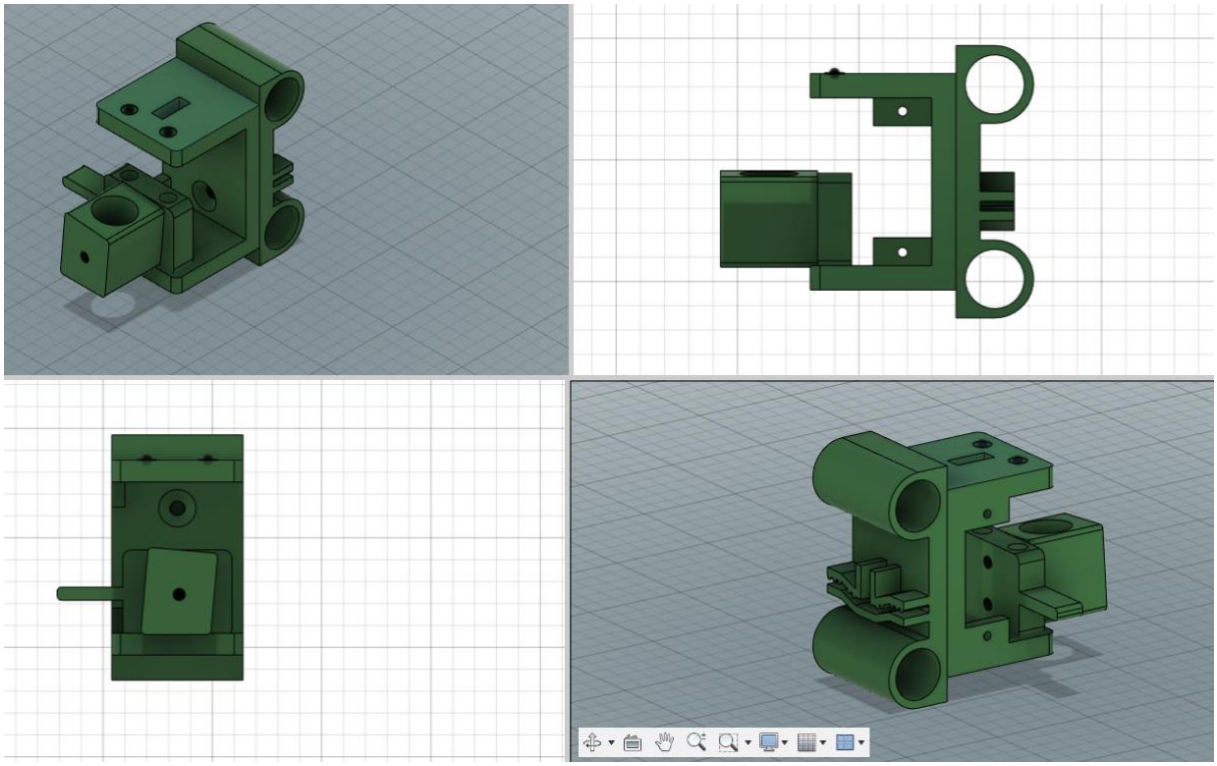

Figure 31 – Revised Pen Holder Design

11.4 Overall Design

The revised design of the machine can be seen below.

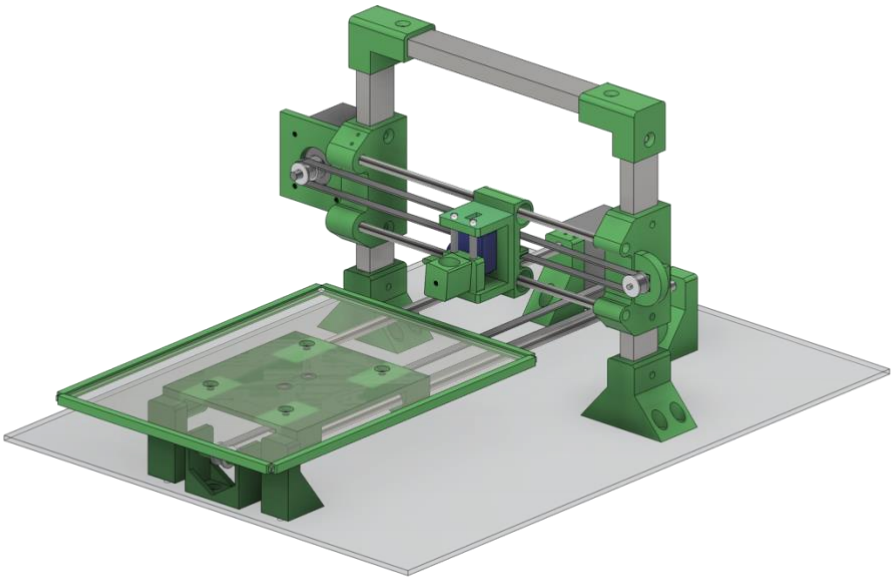

Figure 32 – Isometric View

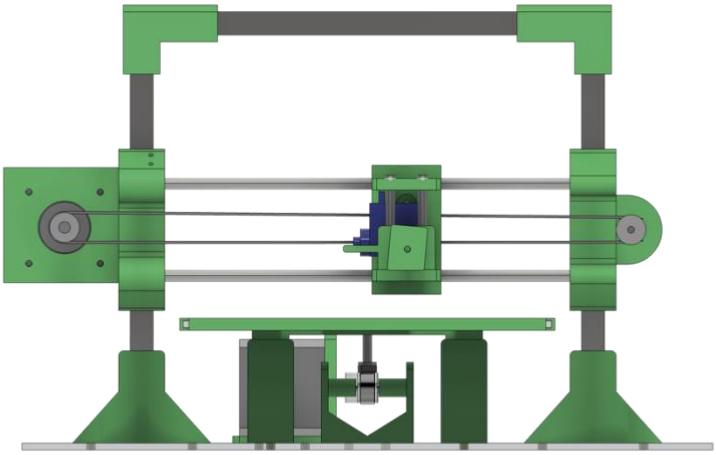

Figure 33 – Front View

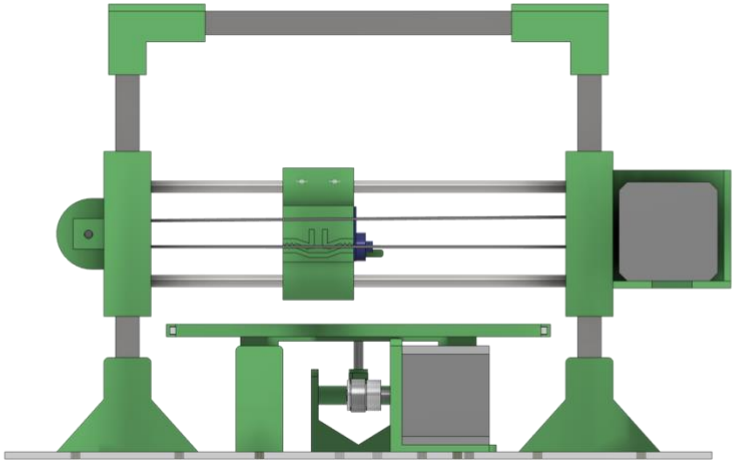

Figure 34 – Back View

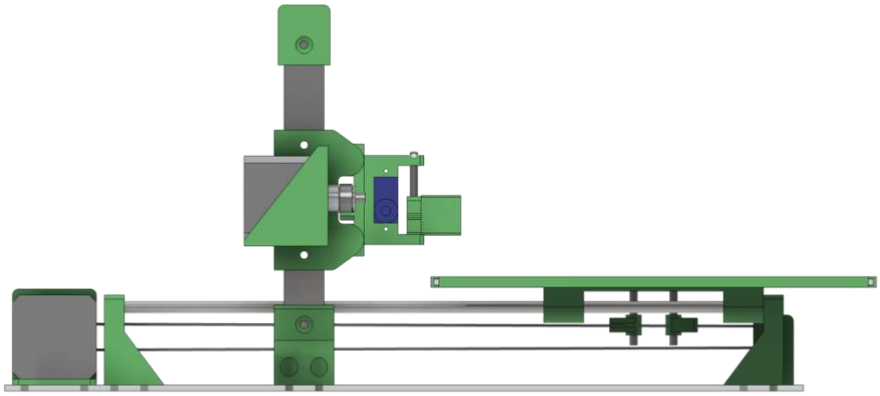

Figure 35 – Left View

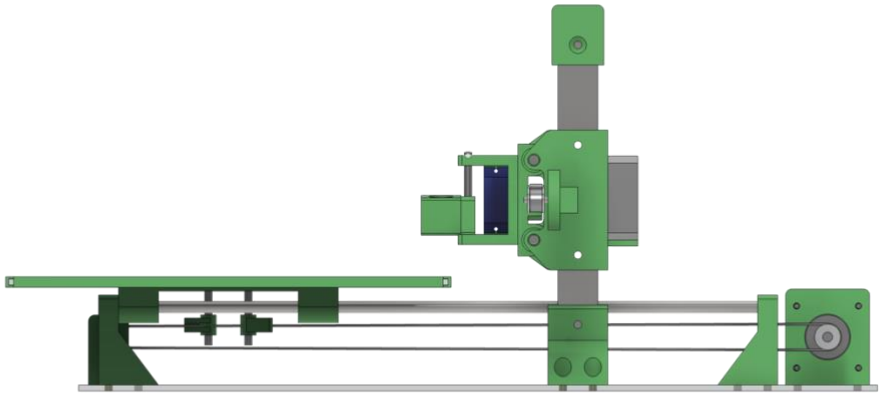

Figure 36 – Right View

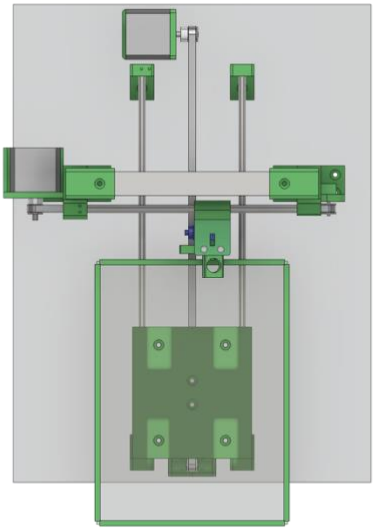

Figure 37 – Top View

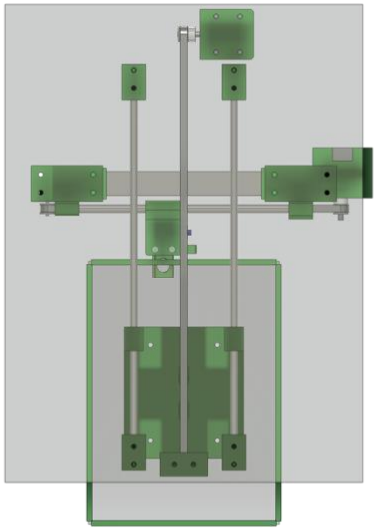

Figure 38 – Bottom View

11.5 Realized Images

Here you can see pictures of the realized state of the machine

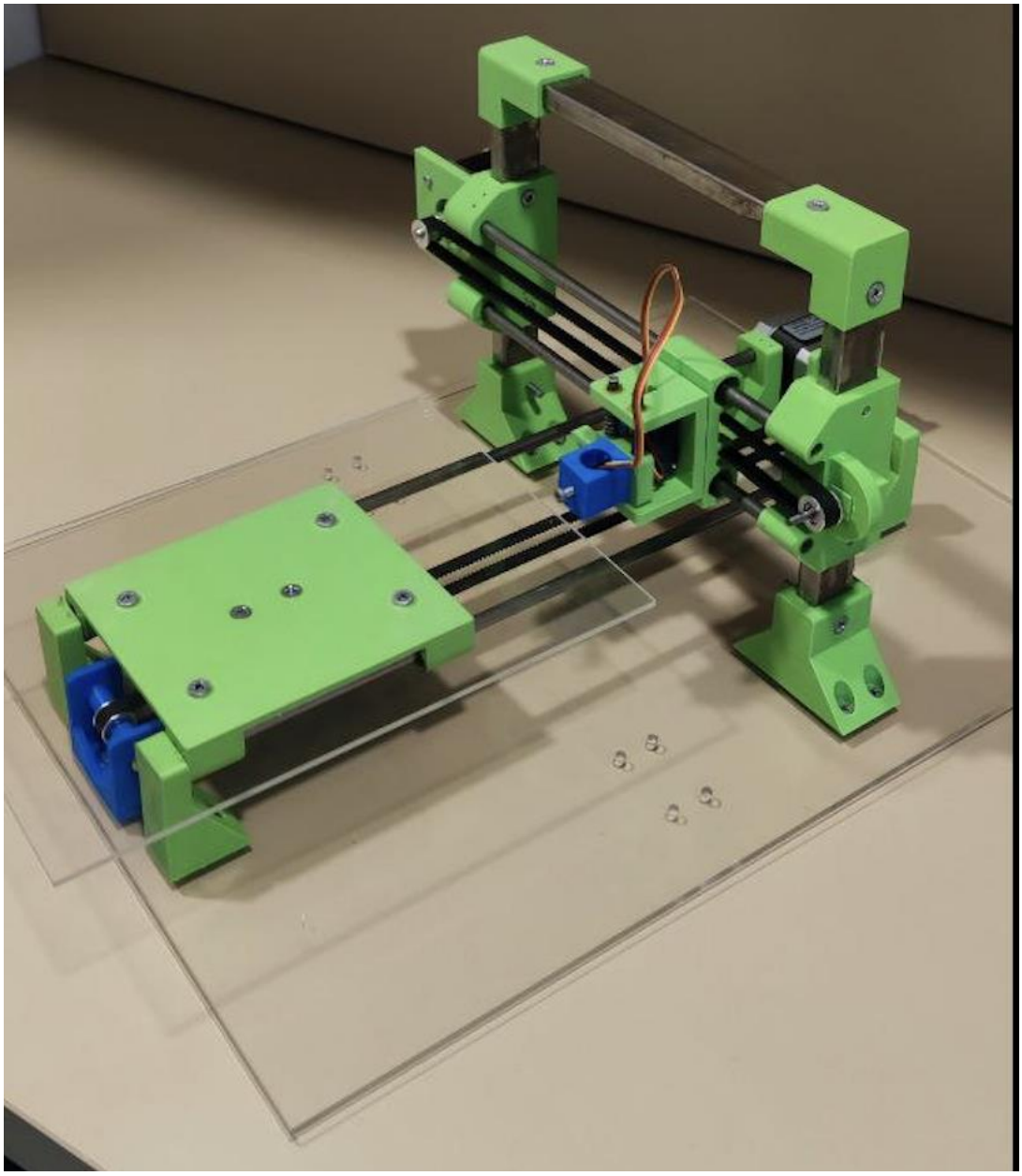

Figure 39 – Machine Isometric View

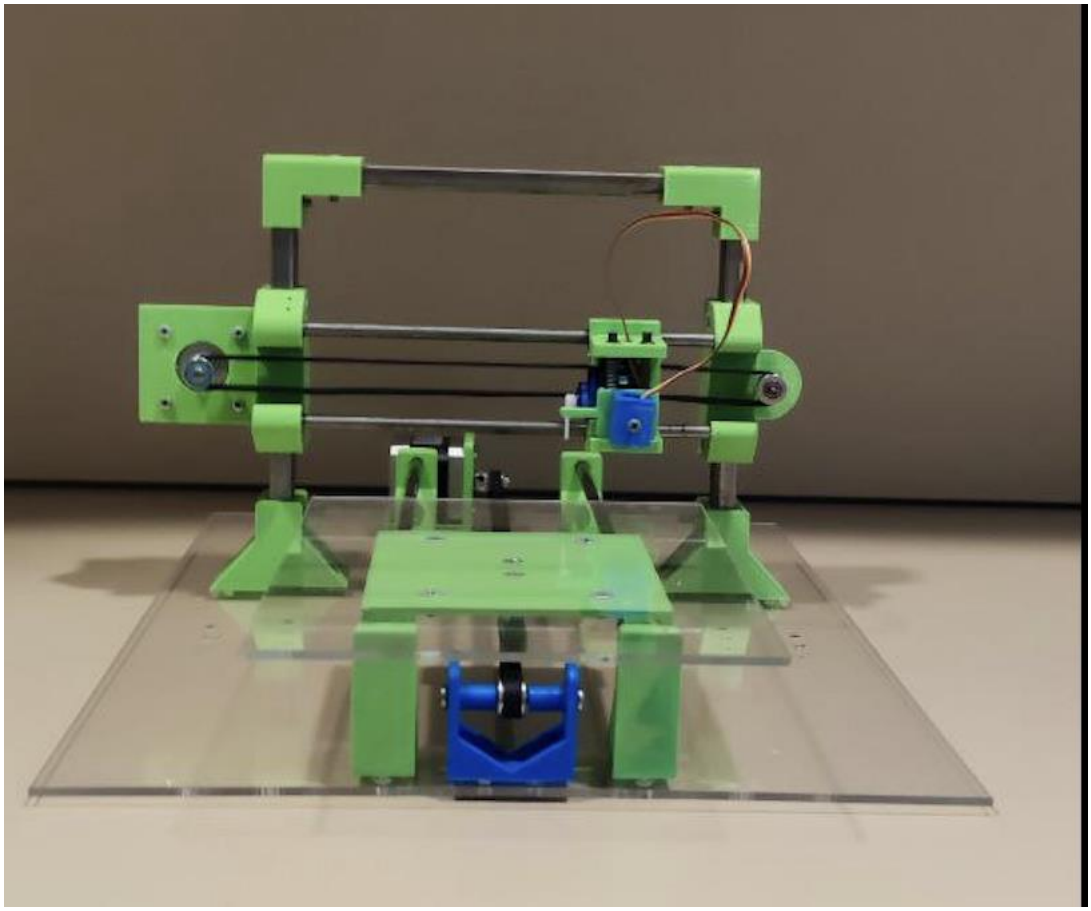

Figure 40 – Machine Front View

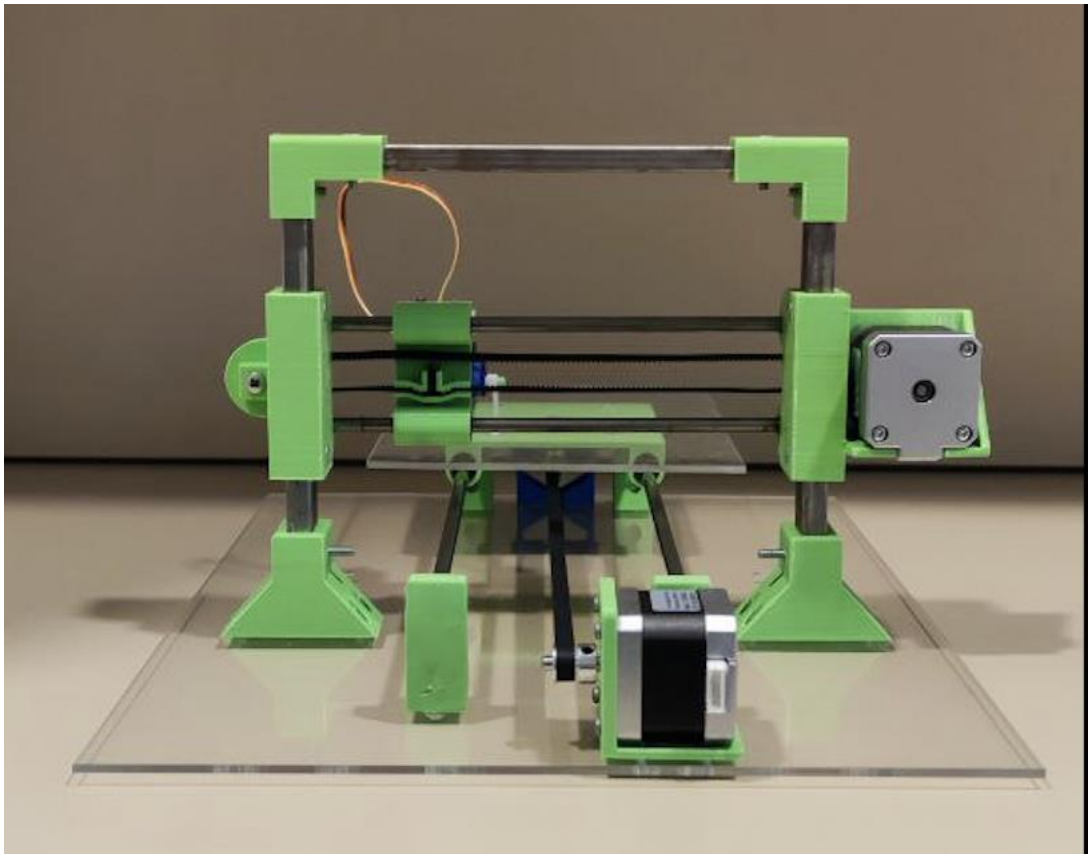

Figure 41 – Machine Back View

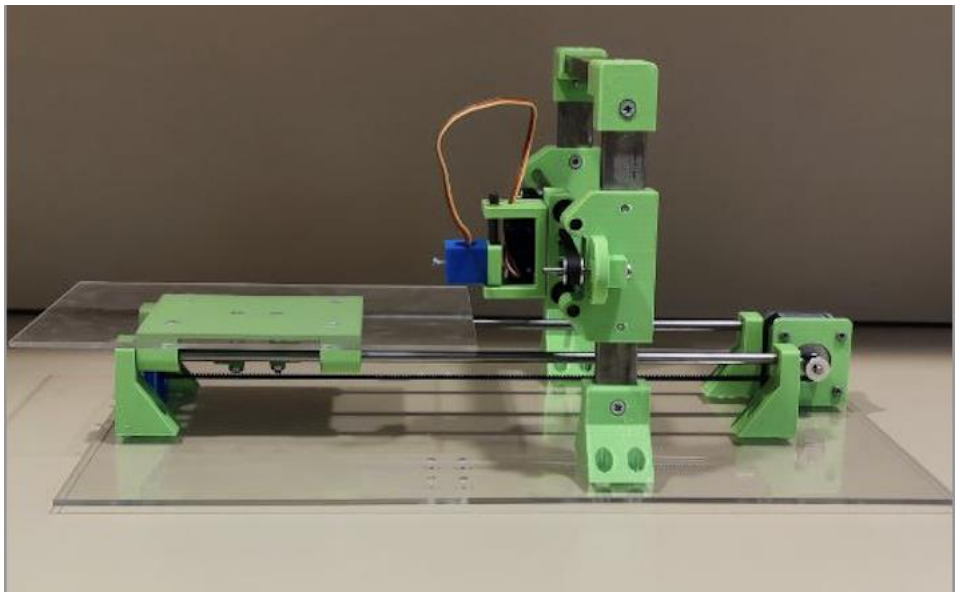

Figure 42 – Machine Right View

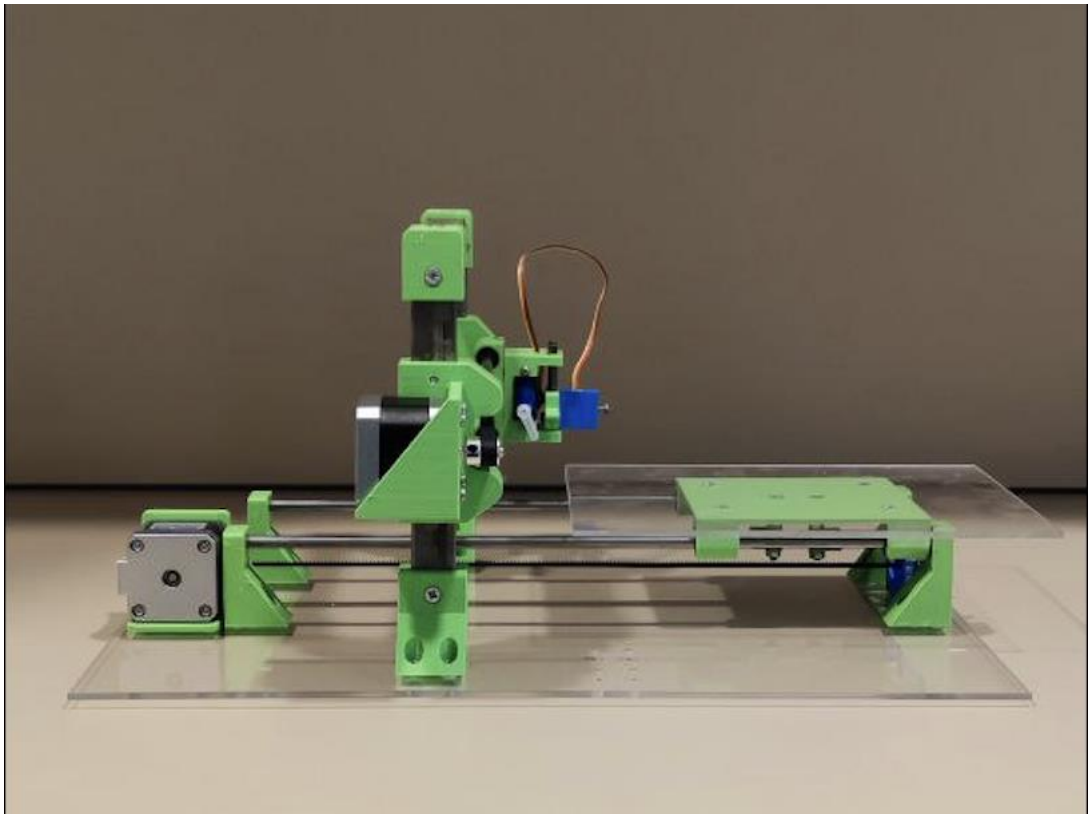

Figure 43 – Machine Left View

12. MOTOR SELECTION

Nema17 HS4401 step motor specifications required for our project are listed in the Table 7 below.

Specification	Value	Image
Step Angle (θ_s)	200	
Maximum Current (I_{max})	1.7 A	
Inductance (L)	2.8 mH	
Inertia of the motor (J_m)	$54gcm^2$	

Table 7 - Nema17 HS4401 Specifications

13. ELECTRICAL SYSTEM DESIGN

The electrical drive system of the plotter will consist of three main systems. These are DC step motor, step motor driver and power supply respectively. The step motor configuration of the machine will be selected in such a way that it can move the base plate with determined speed that specified in the technical requirements. The step motor driver will be selected according to the technical specifications of the step motor.

In the circuit below, there are two step motors, two step motors driver, two limit switches, tow pull-down resistors, one single board computer and one servo motor. Step motor drivers are powered by 5 volts and its motor voltage input is powered by 12 volts. Servo motor and raspberry pi is powered by 5 volts. The figure below is preliminary circuit design.

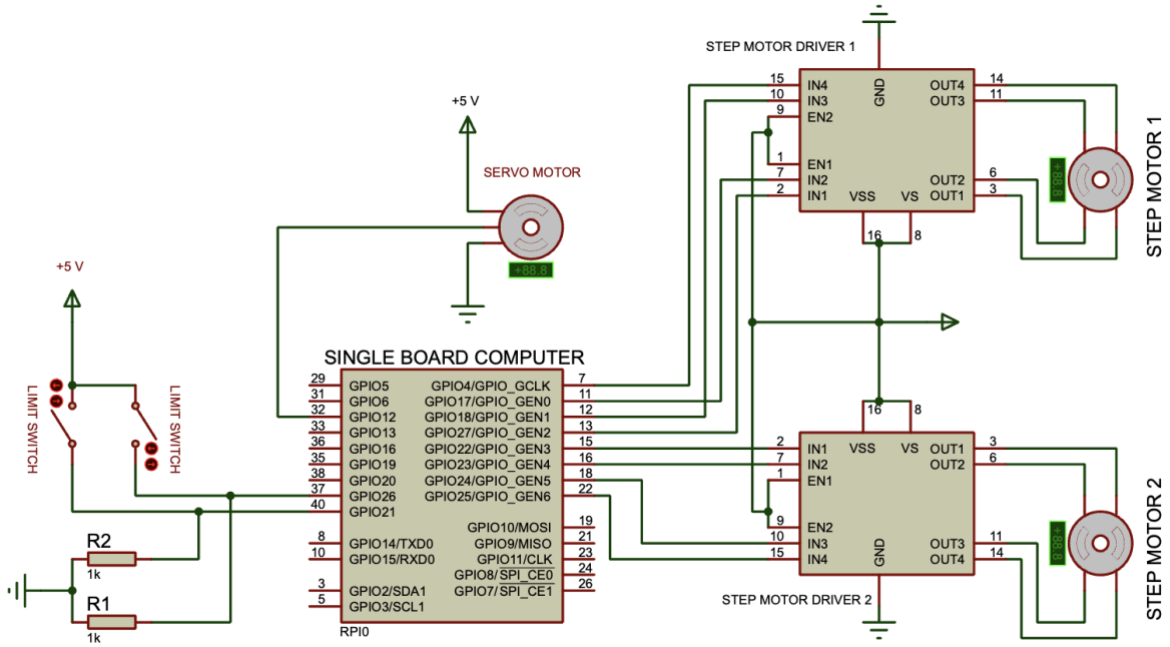

Figure 44 – Preliminary Electrical Circuit of The Plotter Machine

13.1 Revised Electrical Schematic

The revised electrical control circuit diagram was made with the EasyEDA program. The Vmot pin of the stepper motor drivers is connected to the 12V power supply with a 100uF capacitor. EN pins and MS pins are connected to raspberry for full control of the stepper motor driver. Header connections have been made to the A and B pins of the stepper motor drives in order to connect the stepper motors. Headers have been added for the servo motor connection again. 5V and 12V power input locations have been added. Led connection has been added to show the machine activity status.

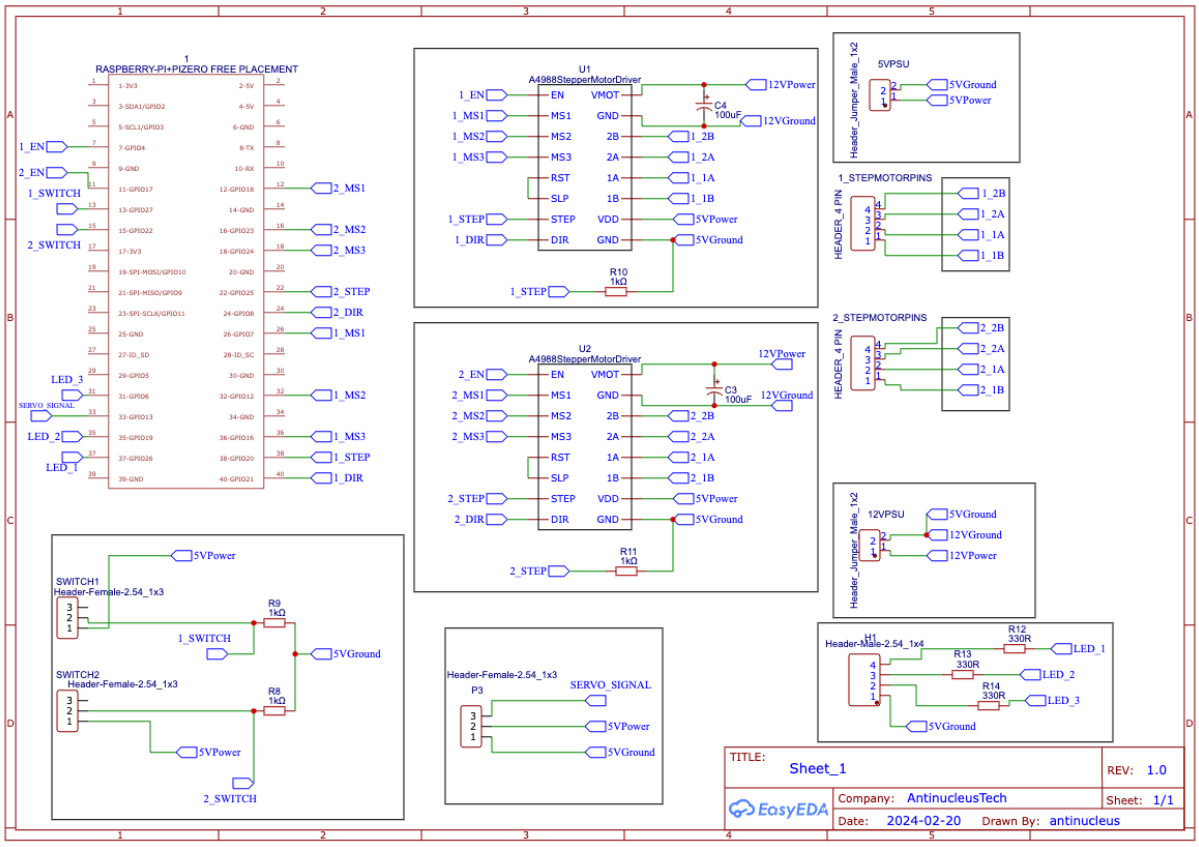

Figure 45 – Revised Electrical Circuit Design

The PCB design of the circuit diagram in figure 45 above is as shown in figure 46 below. Blue colored paths show the paths on the lower layer of the PCB, red colored paths show the paths on the upper layer of the PCB.

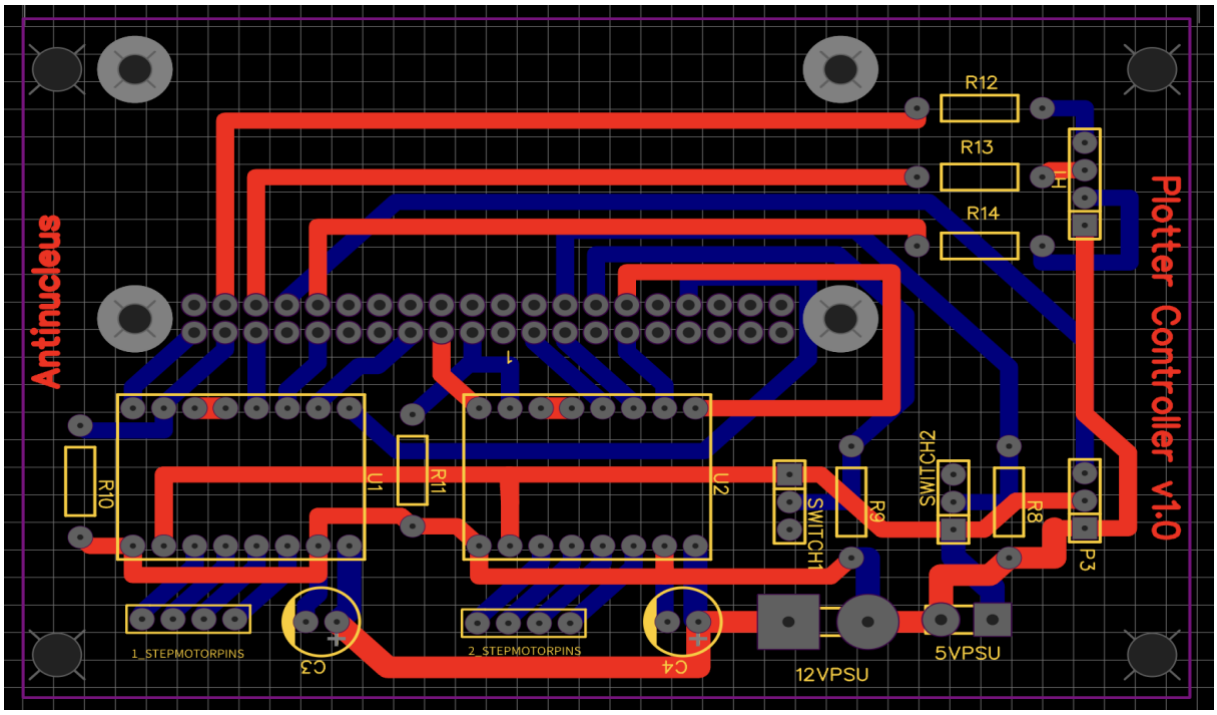

Figure 46 – PCB

In Figure 47, the paths belonging to the top layer of the PCB design are seen.

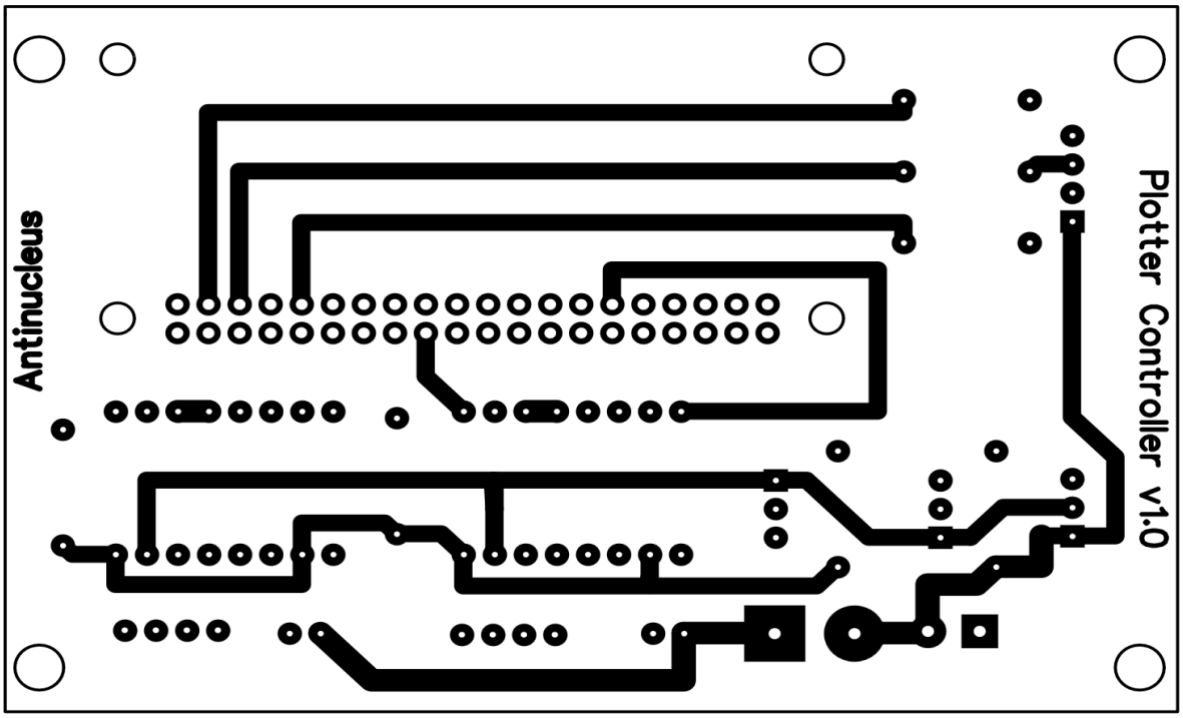

Figure 47 – Top View

In Figure 48, the paths belonging to the bottom layer of the PCB design are seen.

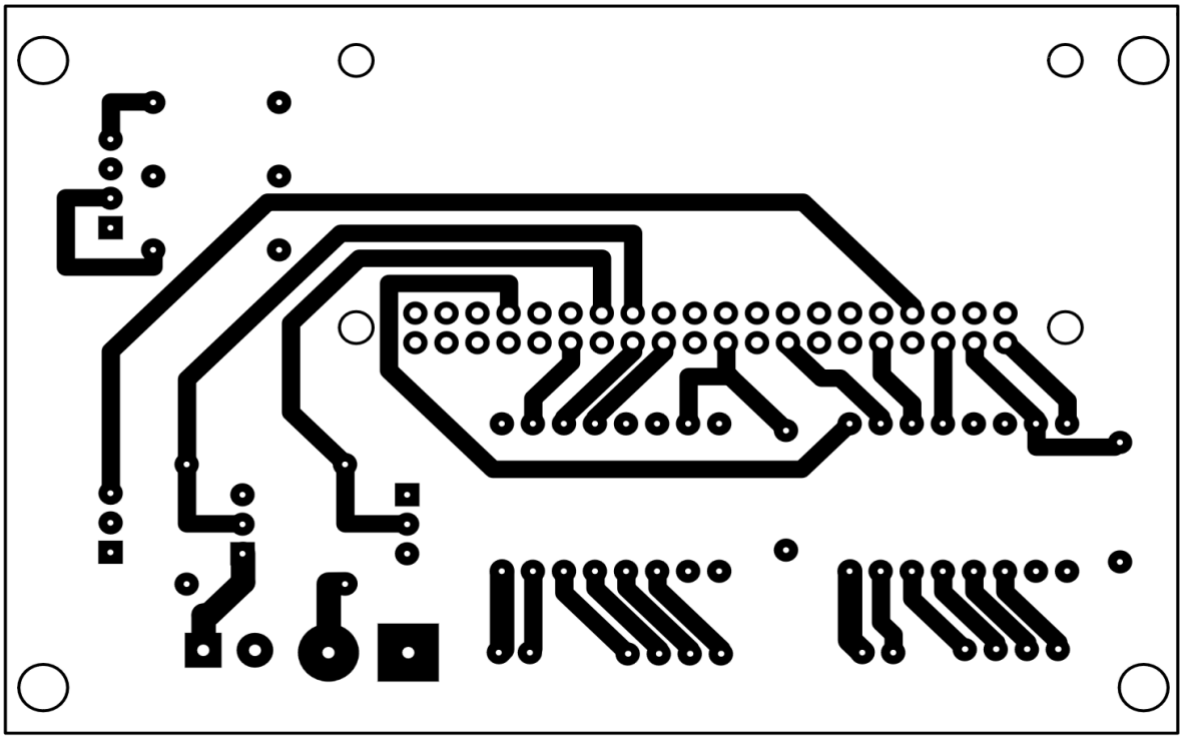

Figure 48 – Bottom View

13.2 **Realized Electrical Circuit**

Below are the realized pictures of the electrical circuit diagram. The circuit is made on a double-sided copper plate. First, the top layer image was printed out from an ink printer on glossy paper and transferred. The circuit on this glossy paper is transferred to the copper plate by temperature. After the transfer is completed, the glossy paper is removed.

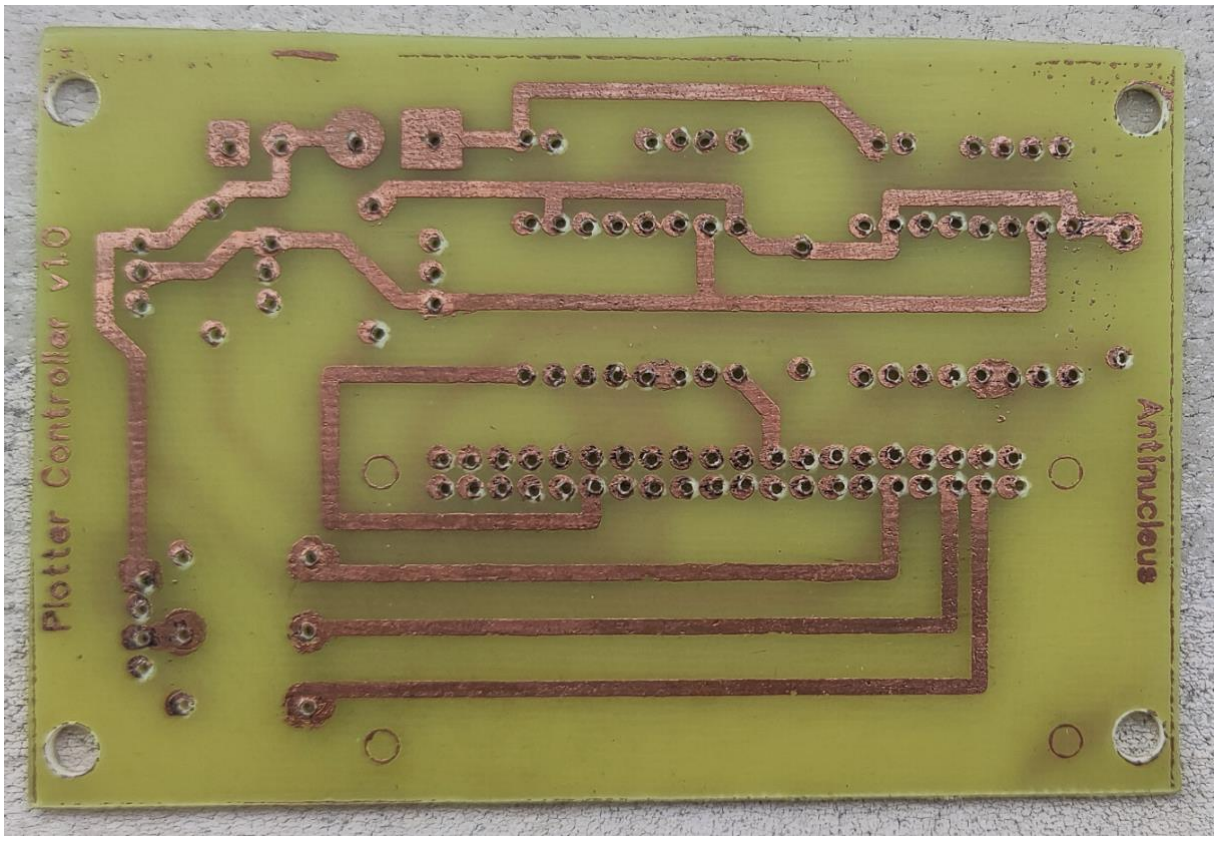

Figure 49 – Realized Top Layer

The same process is performed on the other side of the copper plate in the bottom layer. After the bottom layer circuit is transferred to the copper plate, the copper plate is stored by throwing it into perhydrol and hydrochloric acid to remove unnecessary copper. The remaining coppers contain the paths necessary for our circuit. After the inks on these paths are also cleaned, holes are drilled into the plate.

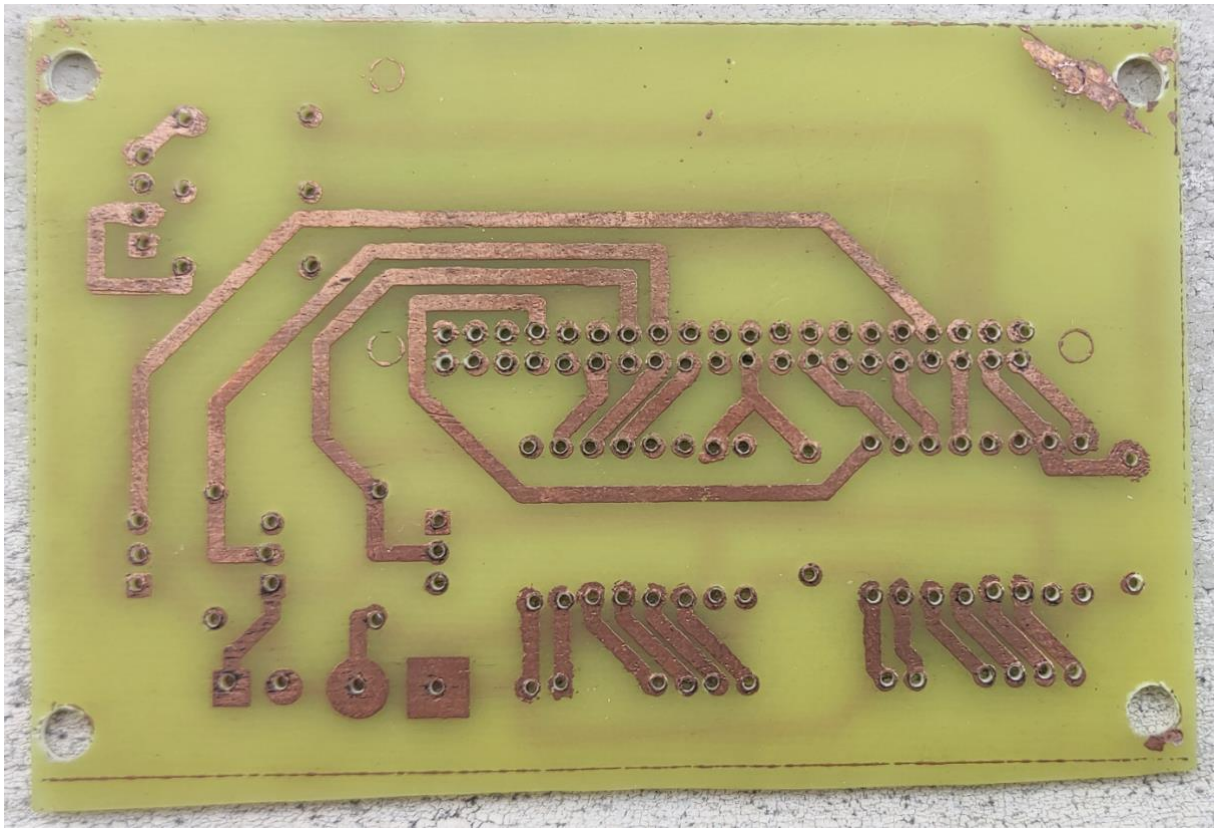

Figure 50 – Realized Bottom Layer

The finished circuit is shown in figure 51 below. Stepper motor connection headers, stepper motor drive headers, servo motor connection headers, limit switch headers, led headers, resistors and capacitors are soldered. Stepper motor drivers and raspberry pi zero 2w controller are mounted on the headers.

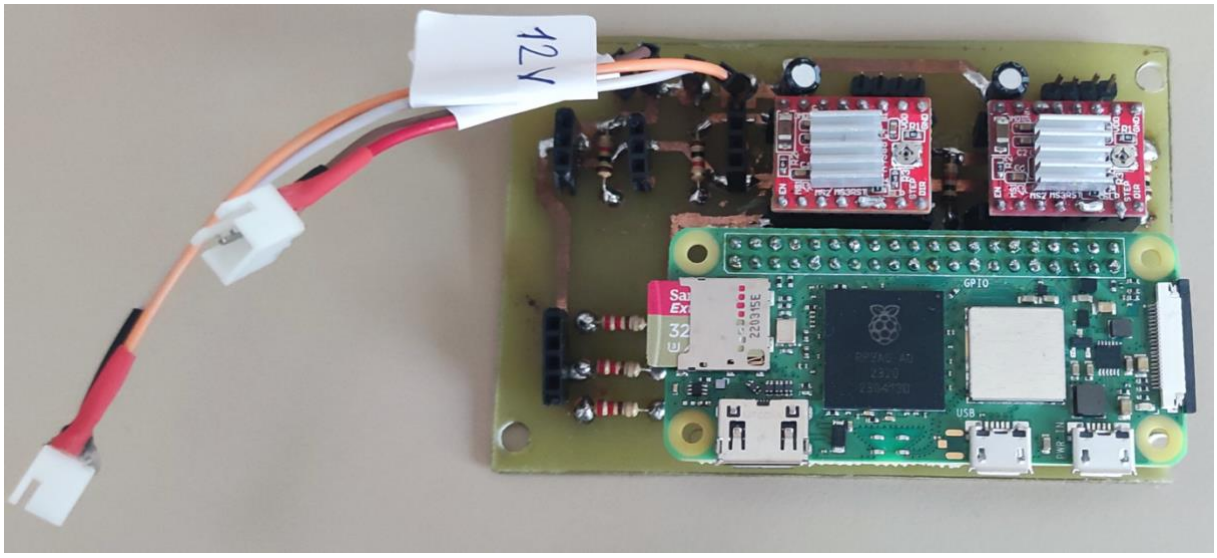

Figure 51 – Finished Circuit

14. POWER CALCULATION

It is important to know how much power the entire system consumes. Because in the electrical circuit design part, it is necessary to make a power supply system and a power distribution system that can provide these powers. The main consumer in the power system is the step motors.

We will drive our stepper motors with 12 volts and the total power to be consumed in the system is calculated as 44.3 watts. A device with a 12-volt 4 ampere can be selected as the power supply, but a power supply with a 12-volt 5 ampere specification will be selected so that there are no glitches in the system.

No	Component	Voltage	Ampere (A)	Power (W)
1	Raspberry pi zero 2w	5	0,120	0,7
2	Nema17 HS4401 - 1	12	1.7	20,4
3	Nema17 HS4401 - 2	12	1.7	20,4
4	Sg90	5	0,550	2,75
5	Step motor driver	5	0,008	0,04
	Total			44,3

Table 8 - Power Consumption of The Components

15. STEP MOTOR DRIVER SELECTION

According to mechanical calculation section, microstepping resolution of the driver must be minimum 1/16. In market we found a4988 step motor driver. This step motor driver can supply 2A output. Nema17 HS4401 stepper motor has maximum 1.7A and this driver can be work at 8-35 voltage range. Also we can use this driver with 1/16 microstepping resolution.

Figure 52 – A4988 Stepper Motor Driver

16. SENSORS

The sensors to be used in the project were selected in line with the literature research and the technical requirements of the project. Accordingly, only one sensor is need for our project that is limit switch. We use limit switch for determining that the base plat is collide with boundiries.

Figure 53 – Limit Switch

17. COMMUNICATION SYSTEM

In this project the pen plotter machine will communicate via wifi. Wifi module is embedded in raspberry pi zero 2w.

Wireless communication will be made between the machine and the remote-control device with the help of the wifi. Image transfer and other datas will be transferred in this way. Thus,

with the interface designed for the machine on the computer or mobile application, it will be possible to follow the progress of the machine such as time, speed, coordinates, and pen status.

18. CONTROL SYSTEM

Raspberry pi zero 2w single board computer will be used as a basis for plotter motion control and data acquisition from sensors. Wireless data transfer to the remote-control device will be carried out with the help of the wi-fi module. Also, file will be converted to g-code using raspberry.

Figure 54 – Raspberry Pi Zero 2w

19. SOFTWARE DESIGN

Three Different software is required for the pen plotter project. These softwares are mobile application software, desktop application software and embedded system software. Control applications will be written for mobile and desktop that will send the necessary data to the machine and receive the necessary information from there. On the embedded system application side, software will be developed that will send the data from mobile or desktop applications to the workplace and the necessary signals to the electronic parts connected to the raspberry pi.

19.1 Mobile and Desktop Application

You can see flow chart of mobile and desktop application below. First user starts the application then connect to wifi hotspot of the plotter machine. If connection is not successful application can not go further. After connection, user can select the file for plotting. Selected file will send from mobile/desktop application to the plotter via wifi. If there is an error while transferring file to the plotter or file process error, then user will be notified what is the error. If there is no error user can start the plotting by pressing the start button.

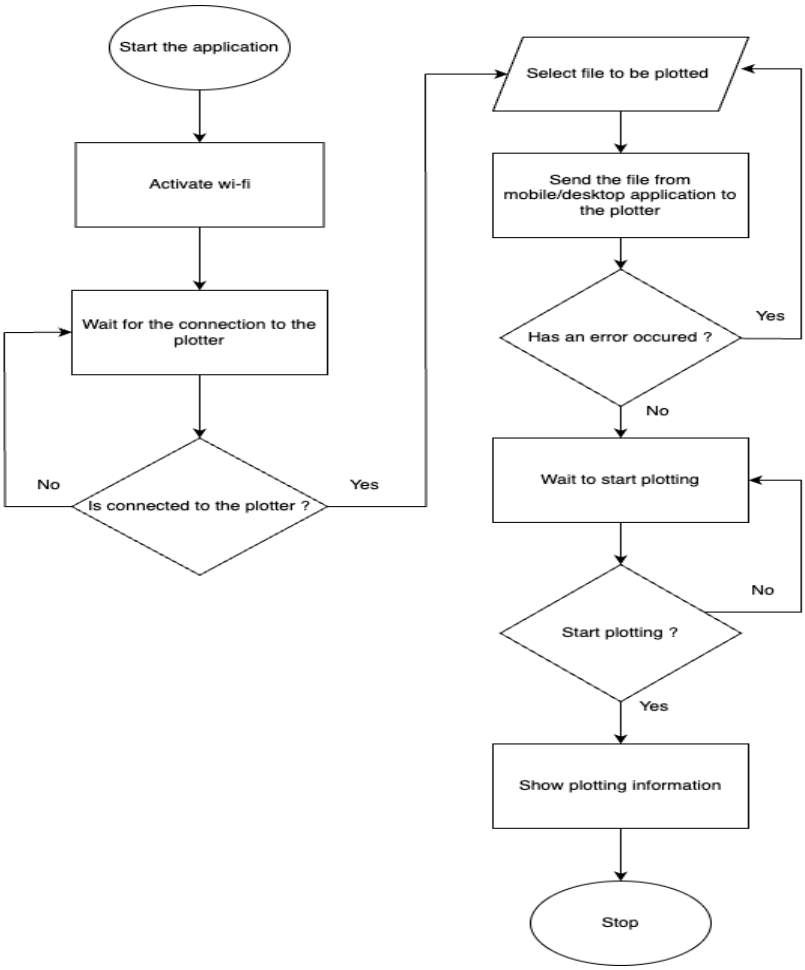

Figure 55 - Flow Chart of The Mobile and Desktop Applications

19.1.1 Mobile Application and Desktop Application User Flow

User flow diagrams are shown in below. All the steps of mobile application can be seen in Figure 56.

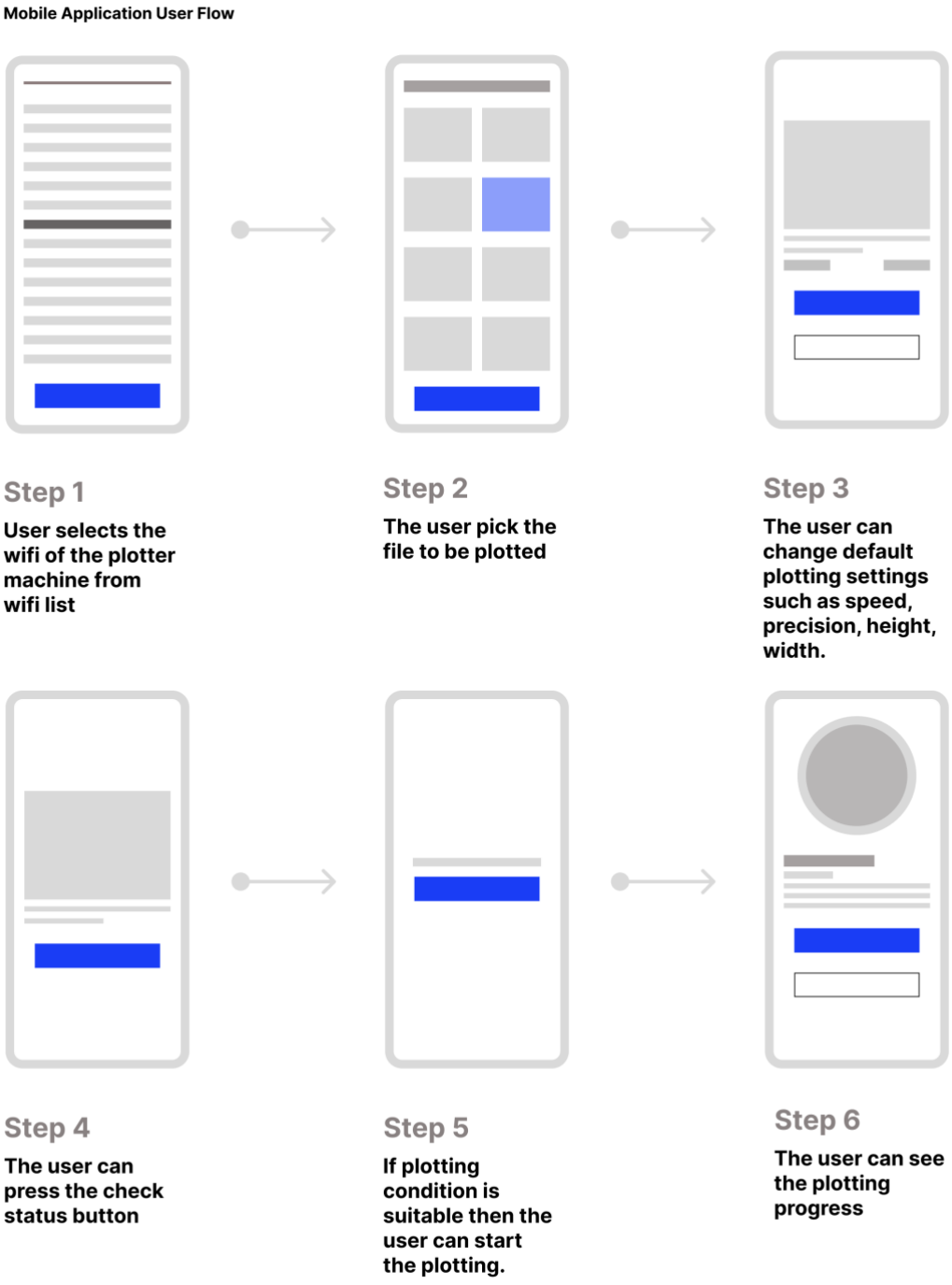

Figure 56 - User Flow of The Mobile Application

User flow diagrams are shown below. All the steps of desktop application can be seen in Figure 57.

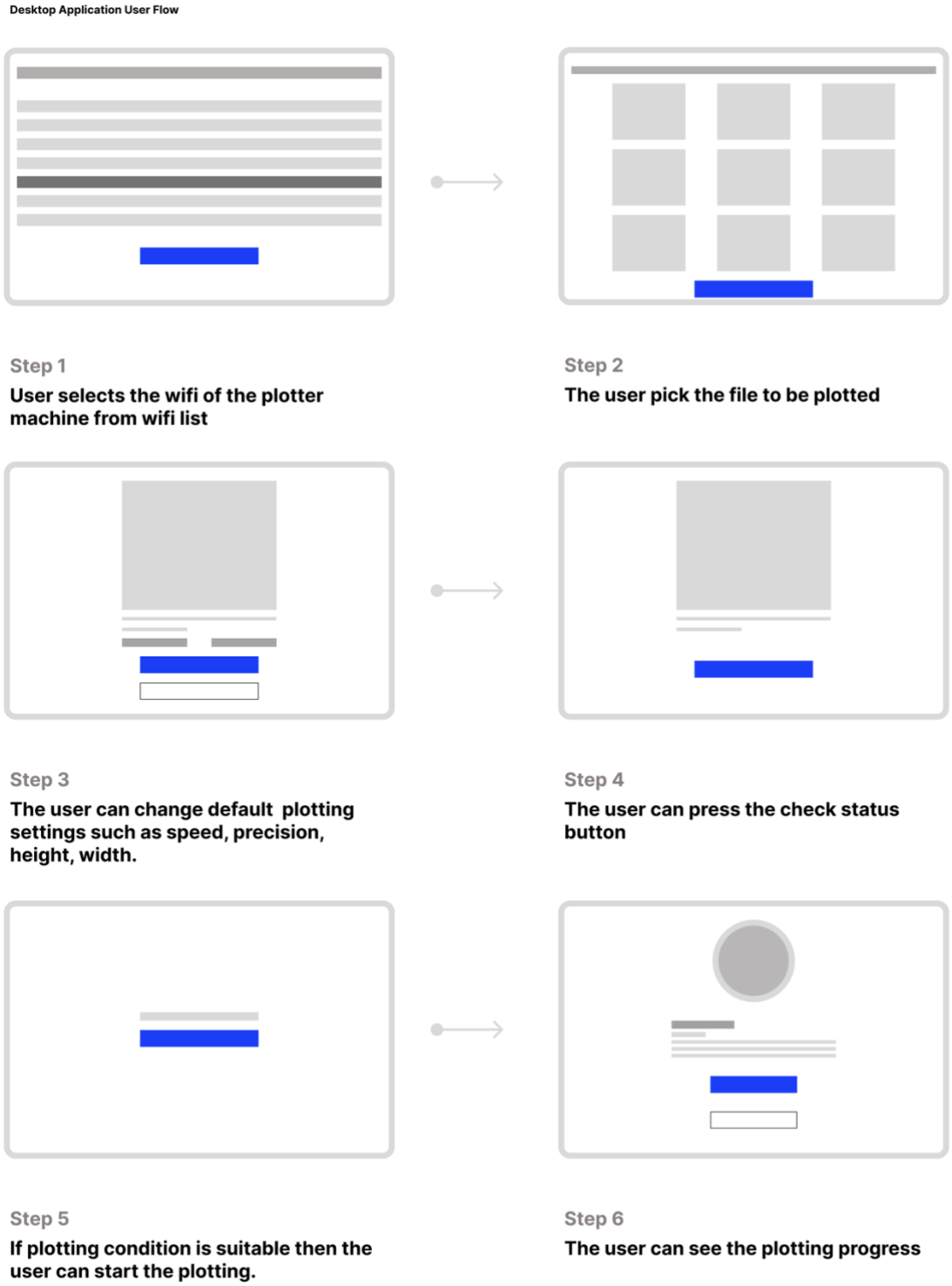

Figure 57 - User Flow of The Desktop Application

19.1.2 **Mobile Application Images**

First of all, we need to provide a connection to the machine.

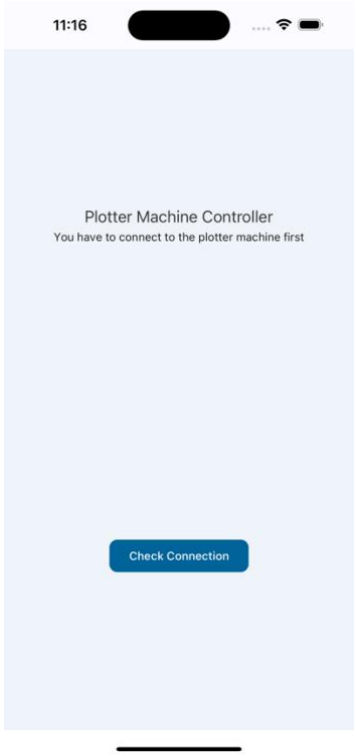

Figure 58 – Connect to the Plotter

The user is informed via the loading indicator until the connection is made.

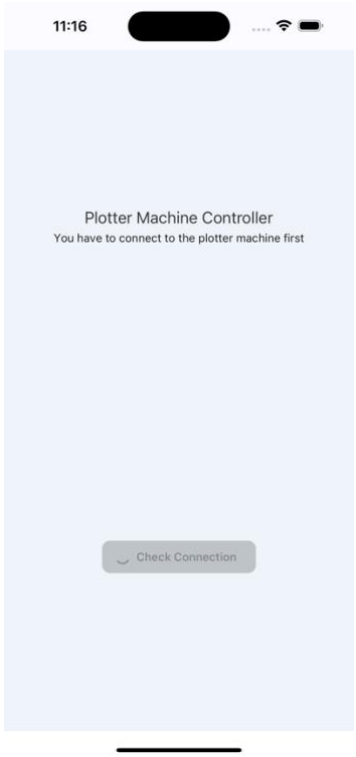

Figure 59 – Check Connection

If the user is not connected to the access point or is not connected to the machine's access point, he will see a warning.

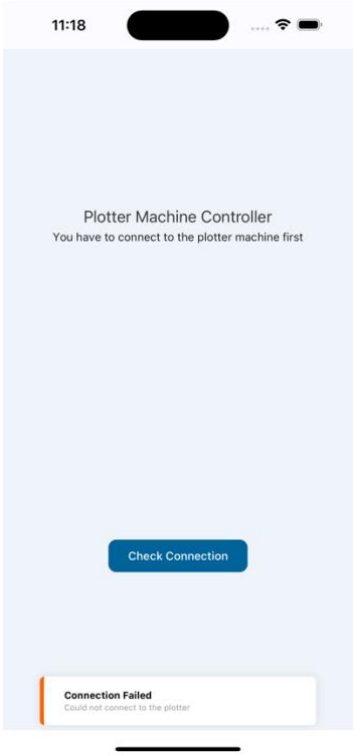

Figure 60 – Connection Error

After the connection is established, the user can switch to the machine control section. In this part, the machine moves itself to the starting position at the entrance. The user is waiting until this process is completed.

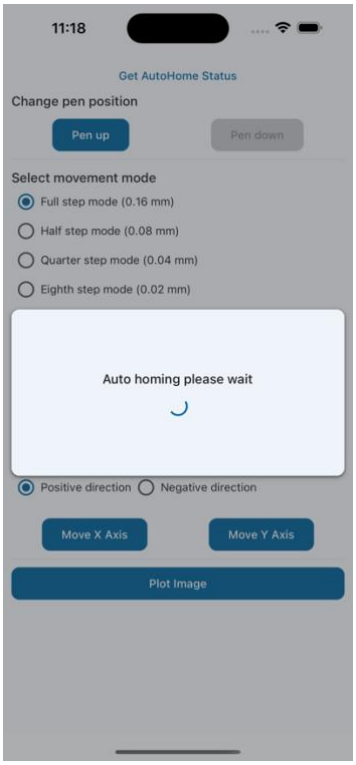

Figure 61 – Auto home

After reaching the starting position, the machine can be controlled manually.

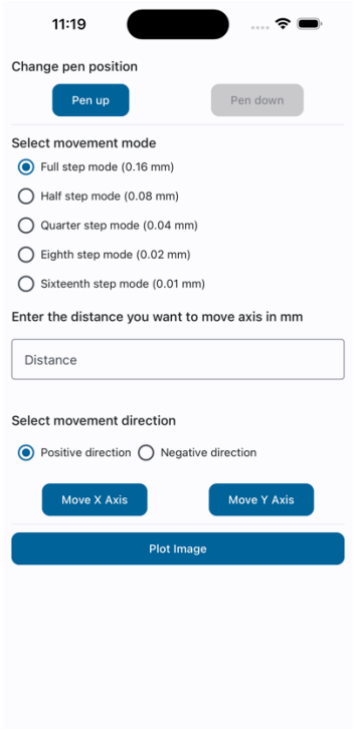

Figure 62 – Manual Control

You can move the pen of the machine up or down, it can be ensured that it moves on the x and ye axes. The movement speed in the x and y axes can also be adjusted from the anywhere mode section. For example, if the fastest mode, full step mode, is selected, 0.16 mm Octet is provided at each pulse of the axis. If the desired distance to be moved on the axes exceeds the machine limits, the user receives a warning as follows.

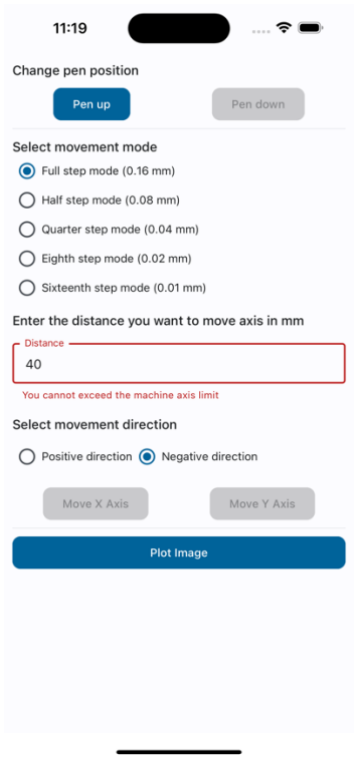

Figure 63 – Limit Error

To draw the selected image, the plot image button is pressed and the transition to the following page is made.

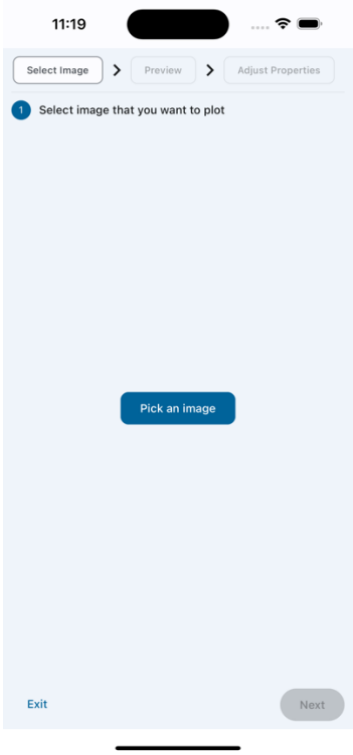

Figure 64 – Plot Image

The desired photo to be drawn from the albums is selected.

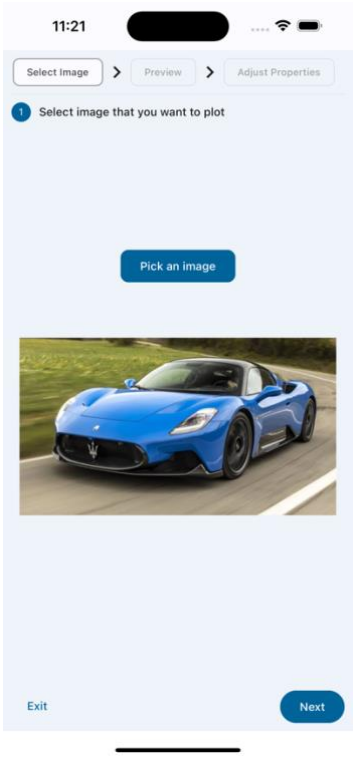

Figure 65 – Select an Image

The drawing detail can be adjusted.

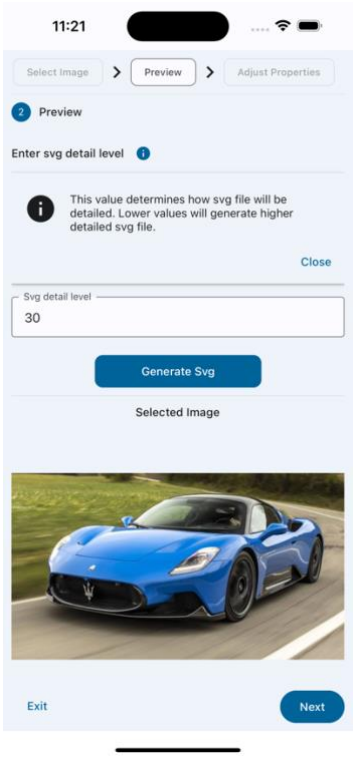

Figure 66 – Detail Level

The generate svg button is pressed to produce a vector image according to the selected detail level. When producing a vector image, the user is informed as follows.

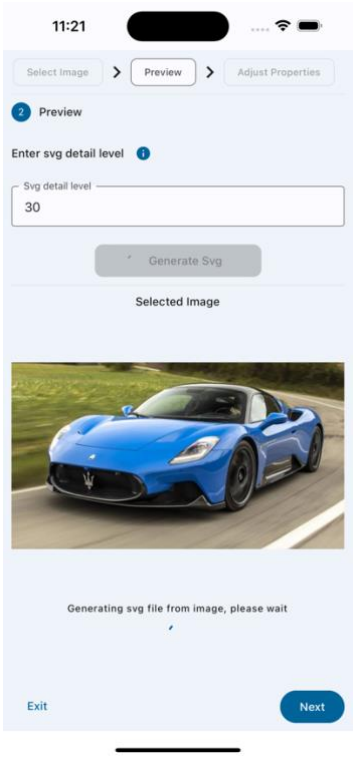

Figure 67 – Generating Svg

After the vector image is generated, the user can see it as follows.

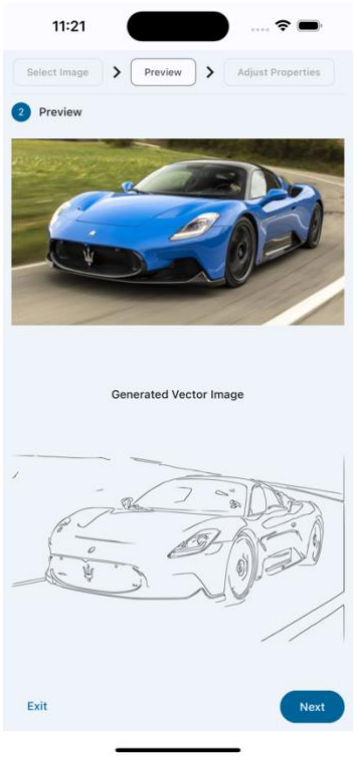

Figure 68 – Generated Vector Image

If he wants to change the level of detail, he can enter the desired value and have the image reproduced. For example, the new detail level is set to 60 below.

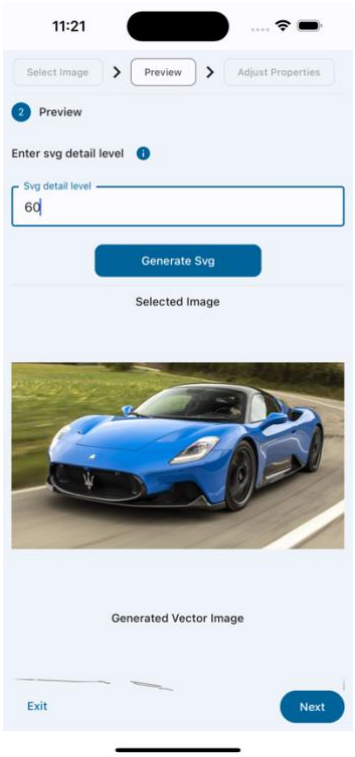

Figure 69 – Regenerate Vector Image

The vector image produced according to the new level of detail is as follows

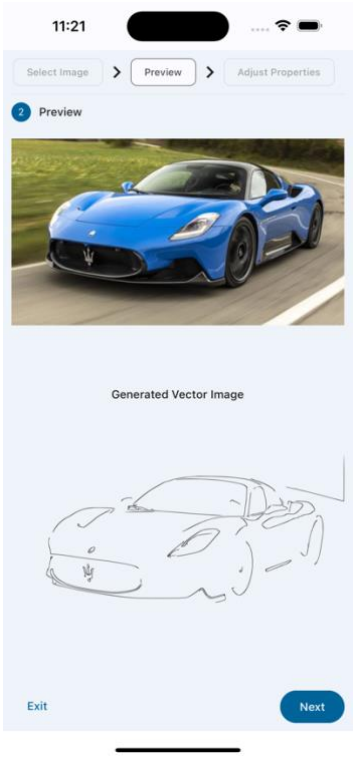

Figure 70 – Regenerated Vector Image

At the next stage, adjustments are being made for gcode production.

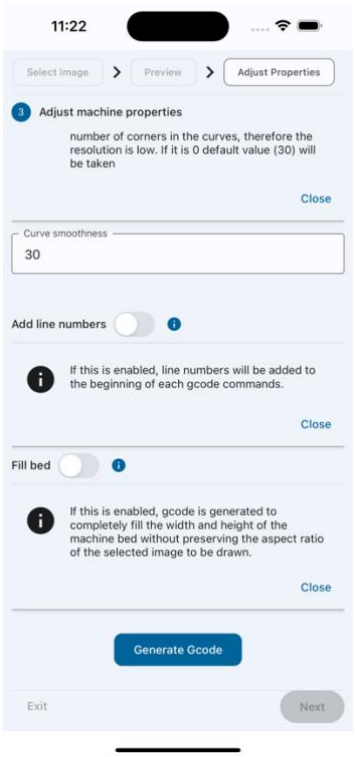

Figure 71 – Gcode Properties

After the adjustments are made, the gcode is generated by pressing the generate gcode button.

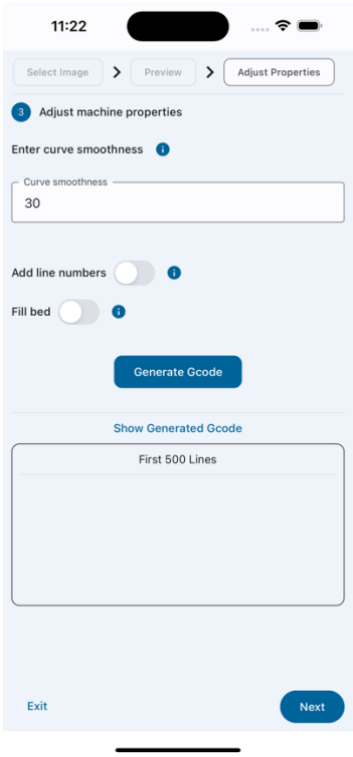

Figure 72 – Generate Gcode

If you want to view the generated gcode, it can be viewed by pressing the show generated gcode button.

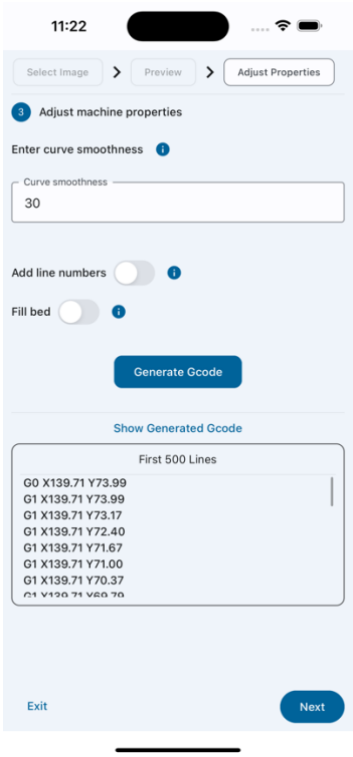

Figure 73 – Show Generated Gcode

In the next part, the gcode is ready to be processed by the machine.

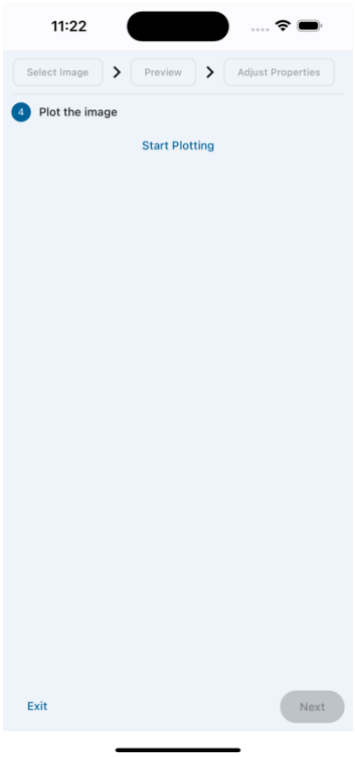

Figure 74 – Start Plotting

When the start plotting button is pressed, the machine is first taken to the starting position and the picture starts to be drawn.

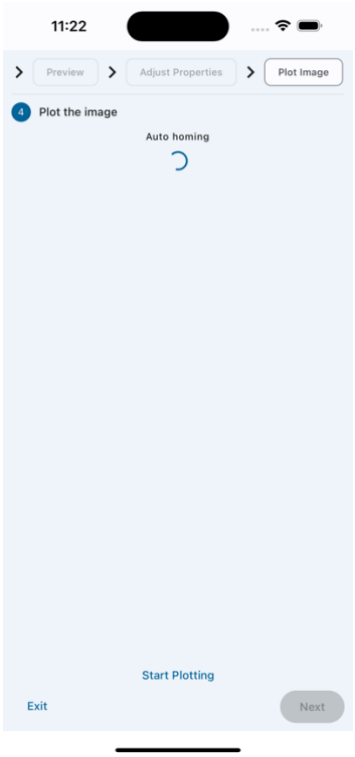

Figure 75 – Auto Home

The position of the machine can be seen instantly as shown in the figure, and if it is desired to stop the drawing, it is stopped by pressing the stop plotting button.

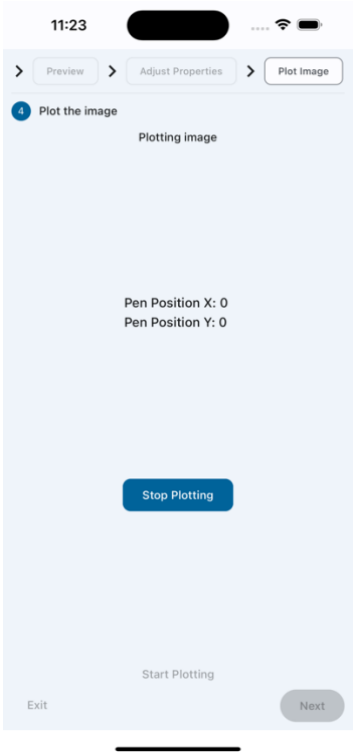

Figure 76 – Plotting

19.2 Plotter Software

In plotter machine side, user first start the plotter then operating system automatically activate hotspot after the controller booted up. Also, our server software is started when controller booted up. Server waits for mobile/desktop application connection. After device connected to the plotter, then it will get the file from mobile/desktop application. If file type is true, then file will be converted to g-code. If error occurred while generation of g-code, user will be informed. Otherwise, plotter wait for start drawing command. If plotting is started machine will plot the data to the surface. Flow chart of the plotter can be seen in below.

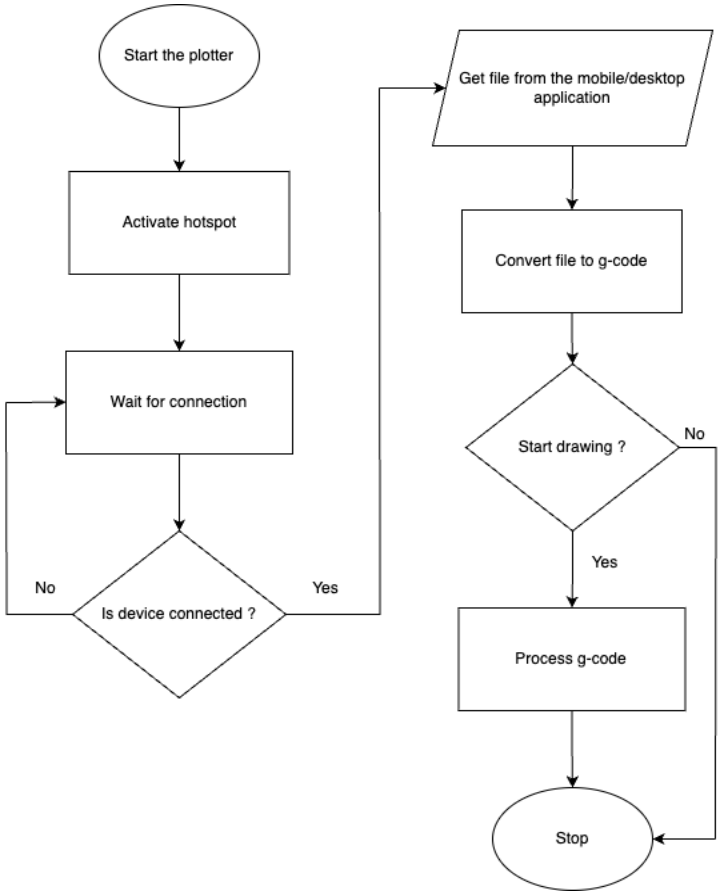

Figure 77 - Flow Chart of The Pen Plotter Machine

19.2.1 **Linux Services**

There are 2 subsystems on the plotter side. These are server program and raspi hardware controller program. The Raspberry Pi OS, which has a load on the Raspberry pi, runs 2 linux services when it boots.

The first of these services is the linux service that starts the server. The mobile application and the raspi hardware controller communicate with each other via this server. Below is the linux service that uploads the server in figure 78.

```
[Unit]
Description=Cnc Controller Service
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/node /home/oxygen/Desktop/PlotterController/plotter-express-server/index.js

[Install]
WantedBy=multi-user.target
```

Figure 78 – Server Linux Service

The other linux service runs the raspi hardware controller. The raspi hardware controller controls all the components in the electrical circuit design, namely stepper motors, servo motor, limit switches and LEDs. Figure 79 shows the linux service running the raspi hardware controller .

```
[Unit]
Description=Embedded Controller Service
After=multi-user.target

[Service]
Type=idle
ExecStart=/home/oxygen/Desktop/PlotterController/embedded/run

[Install]
WantedBy=multi-user.target
```

Figure 79 – Embedded Linux Service

19.2.2 Server Program

The server program is written in express.js. After the server is activated, it responds to requests received both by the mobile application and by the raspi hardware controller. Server contains 5 base path. These can be written as root path (/), machine path (/machine), media path (/media), gcode path (/gcode) and public path (/public). Below you can see how these are defined in express.js.

```
app.use("/", checkDeviceRouter);
app.use("/machine", machineRouter);
app.use("/media", mediaRouter);
app.use("/gcode", gcodeConfigRouter);
app.use("/public", express.static(path));
```

Figure 80 – Base Paths

These base paths contain their own routers. The first of these, / base path, uses a router called checkDeviceRouter. There is a route path in this router named /check as shown in the figure below. The checkDevice controller is working, which checks the connection status of the device when a request is made using the /check route path ine GET method.

```
const checkDeviceRouter = express.Router();

checkDeviceRouter.get("/check", checkDevice);
```

Figure 81 – Check Device Router

The machine (/machine) base path contains the machineRouter. There is a route path named /status. Requests are made here using the GET and POST methods. When a request is made using the GET method, the getMachineStatus controller is running. When a request is made using the POST method, updateMachineStatus works. movePen controller works when a request is made using the /pen route path with POST method. The moveAxis controller works when a request is made using the /axis route path with POST method.

/coordinates route path accepts GET and POST. The getCoordinates controller works when a request is made using the GET method. When a request is made using the POST method, the updateCoordinates controller works.

The resetAllValues controller works when a request is made using the ine GET method to /reset route path.

```
const machineRouter = express.Router();

machineRouter.get("/status", getMachineStatus);
machineRouter.post("/status", updateMachineStatus);
machineRouter.post("/pen", movePen);
machineRouter.post("/axis", moveAxis);
machineRouter.get("/coordinates", getCoordinates);
machineRouter.post("/coordinates", updateCoordinates);
machineRouter.get("/reset", resetAllValues);
```

Figure 82 – Machine Router

The uploadImage controller works when a request is made using the /image/upload route path POST method located in the media (/media) base path.

```
const mediaRouter = express.Router();

mediaRouter.post("/image/upload", upload.single("image"), uploadImage);
mediaRouter.post("/image/generate", convertImagetoSvg);
mediaRouter.get("/image/get", getSvg);
mediaRouter.get("/gcode/generate", generateGcode);
mediaRouter.get("/gcode/get", getGcode);
```

Figure 83 – Media Router

19.2.3 Raster Image to Vector Image Converter

In order for the plotter machine to draw the selected image from the mobile application, the image must be converted to gcode. For this, the selected image first if in vector image format (.svg) if not, the image must be converted to a vector image. For this purpose, we use the open source imagemagick and potrace software. Below you can see the shell script that performs this process on Raspberry.

```
filename=$1
level=$2
extension=${filename##*.}
convert ${filename} -canny 0x1+10%+${level}% -negate out.${extension}
convert out.${extension} result.svg
rm out.${extension}
```

Figure 84 – Image to Svg Conversion Script

In this script, the path of the png or jpg image should be entered as the first parameter, and the number specifying the detail level of the vector image should be entered as the second parameter. These values are selected from the mobile application interface and transferred here. The raster image is converted to vector image by using canny edge detection operator.

19.2.4 Vector Image to Gcode Converter

After the vector image is obtained, it must be converted to gcode. In this section, we have developed software that we have published as open source and that does this job. You can find the link in the References section. This program is a vector image that is thrown into the public folder located under the root directory, gcode, which is also located in the root directory.config.it converts gcodea according to the properties set in the json file. The fields shown in the figure below can be added to the config file.

Field	Default Value	Type	Options	Required
`svgFileName`	""	string	-	✓
`initialCommand`	[ ]	string[]	-	✘
`lineNumbering`	false	boolean	true,false	✘
`sampleCount`	30	number	-	✘
`unit`	"mm"	string	mm,in	✘
`width`	undefined	number	-	if height is not provided
`height`	undefined	number	-	if width is not provided
`centerX`	0	number	-	✘
`centerY`	0	number	-	✘
`fill`	false	boolean	true,false	✘

Figure 85 – Config File Fields

The svgFileName field is a mandatory field. The vector added to the Public folder must be the same as the name of the image. If more than one vector image has been added under the public folder, the name of which vector image the gcode output is desired to be should be written.

The initialCommand field is not required. By default, it is an empty array. The commands added in this field are added to the very beginning of the generated gcode file.

The lineNumbering area is not mandatory. By default, the value is false. If true is made, the line number is added per each gcode produced. This is useful for gcode parsers that use line number verification.

The sampleCount field is not required. The default value is 30. This value determines how soft the bezier curves in the vector image will be when the gcode output is received. If low values are entered, curves, indentations, protrusions begin to appear. For a smooth curve, this value should be high. A high value will cause the generated gcode file to be long and the drawing to take a long time.

The unit field is not required. The default is oalrak mm. It is the space required when converting digital units to physical units for the gcode to be produced. The values of mm or in can be written in this field.

The width field is mandatory if there is no height field. The width value of the area to be drawn must be entered.

The height field is mandatory if there is no width field. The length value of the area to be drawn must be entered.

The centerX area is not mandatory. The default value is 0. This value can be used to shift the x value of the generated gcode.

The centerY area is not mandatory. The default value is 0. This value can be used to shift the y value of the generated gcode.

The fill field is not mandatory. Stop false by default. If true, the image is expanded or collapsed without preserving the aspect ratio so that it fills the entire area entered as width and height while the gcode is being generated. You can see these values in the table below.

Condition	`fill`	Aspect Ratio Maintenance
Both `width` and `height` are supplied	false	✓
Only `width` is supplied	false	✓
Only `height` is supplied	false	✓
Both `width` and `height` are supplied	true	✘
Only `width` is supplied	true	✘
Only `height` is supplied	true	✘

Figure 86 – Fill Field State

For a better understanding, you can examine the figure below.

Svg Image Size	`fill`	AspectR	`width` and `height` value in config file	Image Size as Gcode
width: 100 height: 200	false	1/2	`width`: 250 `height`: undefined	`width`: 100 `height`: 200
width: 100 height: 200	false	1/2	`width`: undefined `height`: 250	`width`: 100 `height`: 200
width: 100 height: 200	false	1/2	`width`: 250 `height`: 250	`width`: 100 `height`: 200
width: 100 height: 200	false	1/2	`width`: 50 `height`: 250	`width`: 50 `height`: 100
width: 100 height: 200	false	1/2	`width`: 250 `height`: 50	`width`: 25 `height`: 50
width: 100 height: 200	false	1/2	`width`: 80 `height`: 120	`width`: 60 `height`: 120
width: 100 height: 200	false	1/2	`width`: 80 `height`: 170	`width`: 80 `height`: 160

Figure 87 – Example Fill Field Output

19.2.5 **Raspi Hardware Controller**

This control program is written with cpp. Raspberry gpio ports are controlled with the Wiring pi library. Communication with the server is made using the Curl library. there are 6 main files. These are api, core, servo, step, utils and main.

The api file contains the functions that will be used in communication with the server. You can see the api header file below. The request is made using the getData function with the backend e GET method. The request is made using the backed-up e-MAIL method with the PostData function. The MovePen function takes the position status of the pen backendend. The UpdateAuthHome function updates this on the backend after the auto home operation is completed. The UpdateMovingAxisStatus function updates the status of the end of the movement in manual mode on the x or y axis on the backend. updateMovingPenStatus

updates the pen position information on the backend. The upadateStartPlottingStatus function is used to update the start status of the drawing on the backend. With the sendCoordinates function, the position information on the x and y axes of the machine is sent in the backend.

```
#ifndef _API_H
#define _API_H
#include <string>

std::string getData(std::string path);
std::string postData(std::string path, std::string body);
void movePen(std::string direction);
void updateAutoHomeStatus(std::string autoHome);
void updateMovingAxisStatus(char axis, std::string movingStatus);
void updateMovingPenStatus(std::string movingStatus);
void updateStartPlottingStatus(std::string startPlottingStatus);
void sendCoordinates(double x, double y);

#endif
```

Figure 88 – Api Header File

The core file contains Raspberry GPIO port information and some hardware functions. The initpins function sets all the required pins as input or output. The disableOutput function is used to disable the pins after they are set. With the setDriverStatus function, drivers can be activated or deactivated. The setDriverMode function adjusts which driving mode the drivers will operate in. The getStepPerRev function hesps the number of steps required for a rotation according to the specified mode. The getOneStepDistance function calculates the distance taken in a pulse according to the given mode. The readSwitch function can be used to read values from switches.

```
#ifndef _CORE_H
#define _CORE_H

#define HIGH 1
#define LOW 0
#define DRIVER_ENABLE !HIGH
#define DRIVER_DISABLE !LOW
#define FULL_STEP 1
#define HALF_STEP 2
#define QUARTER_STEP 4
#define EIGHTH_STEP 8
#define SIXTEENTH_STEP 16
#define ENABLE_PIN_X 7
#define DIRECTION_PIN_X 29
#define STEP_PIN_X 28
#define MS1_X 11
#define MS2_X 26
#define MS3_X 27
#define ENABLE_PIN_Y 0
#define DIRECTION_PIN_Y 10
#define STEP_PIN_Y 6
#define MS1_Y 1
#define MS2_Y 4
#define MS3_Y 5
#define LIMIT_SWITCH_X 2
#define LIMIT_SWITCH_Y 3
#define STEP_PER_REV 200
#define SERVO_PIN 23
#define DELAY 1000 // microseconds
#define PULLEY_CIRCUMFERENCE 32.0f
#define ONE_CYCLE 360.0f
#define DEFAULT_ANGLE 1.8f
```

Figure 89 – Core Header File 1

```
void initPins();
void disableOutput();
void setDriverStatus(char axis, int status);
void setDriveMode(char axis, int mode);
int getStepPerRev(int mode);
double getOneStepDistance(int mode);
int readSwitch(char axis);

#endif
```

Figure 90 – Core Header File 2

The servo file contains the functions required for servo motor control. Using MoveTool, the pen can be positioned up or down.

```
#ifndef _SERVO_H
#define _SERVO_H

void moveTool(char direction);

#endif
```

Figure 91 – Servo Header File

The step file contains the functions necessary for stepper motor control. With the PulseStep motor function, a pulse is sent to the motor. With moveAxis, the pen is taken to the desired distance at the desired speed by the motor on the selected axis. The stayMinimumDistanceFromSwitch function allows the motors to stop in the closest position that does not touch the buttons after auto home is made.

The autohome function allows it to be set to the 0,0 position on the x and y axes.

The servebed function can be used to forward the bedi after the drawing is finished.

```
#ifndef _STEP_H
#define _STEP_H
#include <string>

// directionX, directionY: -, +
// targetDistanceX, targetDistanceY: desired distances in mm
// oneStepDistance: one step will move plotter by a distance of 0.16 mm in full-step mode

void pulseStepMotor(char axis);
void moveAxis(std::string directionX, double targetDistanceX, std::string directionY, double targetDistanceY,
void stayMinimumDistanceFromSwitch();
void autoHome(double *currentX, double *currentY);
void serveBed();

#endif
```

Figure 92 – Step Header File

The utils file contains some auxiliary functions. The setDriveModeManualControk function updates the driver mode for manual mode. The ManualMode function includes steps that allow manual control of the machine. The checkStopPlotting function checks whether the drawing is finished or not.

```
#ifndef _UTILS_H
#define _UTILS_H
#include <string>

void setDriveModeManualControl(char axis, std::string driveMode);
double getOneStepDistanceManualControl(std::string driveMode);
void manualMode(std::string response, double *currentX, double *currentY, short *isStarted);
short checkStopPlotting(short *isStarted);
#endif
```

Figure 93 – Utils Header File

REFERENCES

[1] https://www.creativebloq.com/buying-guides/best-pen-plotters

[2] https://idrawpenplotter.com/shop/ols/products/xn-idraw-pen-plotter-2-a3

[3] https://shop.pimoroni.com/products/axidraw-v3-a3?variant=32196192895059

[4] https://www.line-us.com/

[5] https://www.alldatasheet.com/datasheet-pdf/pdf/338780/ALLEGRO/A4988.html

[6] https://aus3d.com.au/products/gt2-pulley-16-teeth-5mm-bore

[7] https://www.pidramble.com/wiki/benchmarks/power-consumption

[8] https://engineering.stackexchange.com/questions/23565/determining-required-torque-for-a-chain-and-sprocket-conveyor-belt

[9] https://www.mechatronicscenter.com/uploads/7/5/6/2/75629763/introduction_to_motor_sizing.pdf

[10] https://www.linearmotiontips.com/how-to-account-for-belt-and-pulley-inertia-during-system-design/ [11] https://maker.robotistan.com/robot-kontrolculeri-sensorler/

[12] https://www.linearmotiontips.com/how-to-calculate-motor-drive-torque-for-belt-and-pulley-systems/

[13] Ugural, Ansel C - Mechanical Design of Machine Components, Second Edition-CRC Press (2015)

[14] https://github.com/antinucleus/convert-svg-to-gcode

[15] https://github.com/antinucleus/plotter-mobile-application

[16] https://github.com/antinucleus/plotter-embedded-controller

[17] https://github.com/antinucleus/plotter-express-server