CHAPTER 1

Introduction

1.1 Background

In the early days of the computer with the advent of computer graphics, its use as a design-aid for both – text processing and product design grew. Due to the lack of a system standard, files compatible with one design were completely incompatible with most others. Thus, paper remained a major medium of communication. To overlay the paper with the design inside a computer's memory, two methods are available: Printing and Plotting. A printer works on the principle of raster graphics, where the design is maintained by the device as a set of colored lines that form the design when stacked appropriately during reproduction. A plotter works on the principle of vector graphics, that maintains the design through the vertices present in it, which the machine then joins appropriately to reproduce the design on paper. This project aims at designing and implementing a plotter.

1.2 Relevance

Foremost, there is a train of problems to solve. First of those that come to the mind are, for example, "how to measure distances traversed by a motor?" On a little further examination, one might encounter: "how to move the X- and Y- motors independently using just a single core?"

Then there are problems that aren't really problems in the beginning when scale isn't an issue; but as libraries are shipped, questions like "should the user be able to swap X- and Y- axes programmatically?", or "should the origin be accessible/adjustable by a user?"; even "how should the machine behave if a point specified to draw was outside its bounds? Should it ignore the call? Or should it draw till the edge and then return an error and proceed to the next line in code as if nothing happened?" start to arise. Where there are problems, there are solutions; to find which us engineers are the relevant beings to consult.

Now for the electronics part: any movement won't be possible without powering the machine. Ensuring sufficiency of power is the responsibility of a "supply", something that requires to be designed, which in our case, converts ~240V AC mains to 5V and

12V constant DC sufficient to drive 2 1.5A Max. rated stepper motors, a puny servo, a microcontroller and overcome a bunch of other resistances which could lead to severe voltage drops or sudden surges if improperly designed. Additionally, for mainly but not limited to aesthetics, this must be done on a PCB, where some basic electronics knowledge is useful.

Finally, the project has moving parts, which calls also for some basic mechanics an algorithmic mind, and a lot of common sense: likely something that an engineer with some experience in creation has acquired.

1.3 Literature Survey

A 2D X-Y Plotter using the Arduino Board has been implemented but is intended to be programmed using G-Codes, not always familiar to computer programmers. [1]

Another similar project was also undertaken, again through the Arduino platform, this time using pre-built libraries and G-Code conversion software. [2]

1.4 Motivation

Much of the motivation for this project arose from the fascination of experiencing first-hand the olden days of revolutionary technology, “to see something come to life.” We therefore deliberately did not choose to use anything pre-existent and started from scratch.

1.5 Aim

To design and implement a minimalistic 2-dimensional graphics plotting machine that uses the vector scheme.

1.6 Objectives

a. To design simple hardware;
b. To create a PCB for the mounting components comprising the hardware;
c. To create a simple C library for user programs by providing a small set of powerful system calls that could exhaustively render any geometry on paper;
d. To build a mechanical assembly for the machine.

1.7 Scope

1. The library implemented has code for only linear interpolation between 2 specified points in the movev() function.
2. Creating mechanical drawings does **not** come under the scope of this project.

CHAPTER 2

Description

2.1 Technical Approach

As is hinted above, we avoided the use of G-codes to program the machine so that any interested computer programmer interested in the inner workings of vector graphics can tinker without any knowledge of CNC programming.

For the most part, the project is a server-client system where the server is the library that we provide, and the client is a program that uses it. Without a client, the library or the machine is of no use.

Only 3 functions to drive the hardware on user request, that we call *system calls* or just *syscalls*, are implemented: penup(), pendown(), and movev(). As their name suggests, the first two either engage or disengage the pen with paper. The third one moves the pen from where it currently is, to a supplied target point according to the supplied method of interpolation (which, as mentioned in the scope, is restricted to linear as of now).

Implementing the first 2 syscalls was easy as they only rotate the servo arm by some angle. The algorithm is as follows:

```
algorithm: pendown
input: nothing
output: the pen is put in contact with paper if it isn't already
{
        set the reference so that the servo minimizes the error between the angle of
        the servo arm such that the pen comes in contact with paper, and the current
        angle of the arm;
        return;
}
```

Algorithm for penup() is similar to that of pendown().

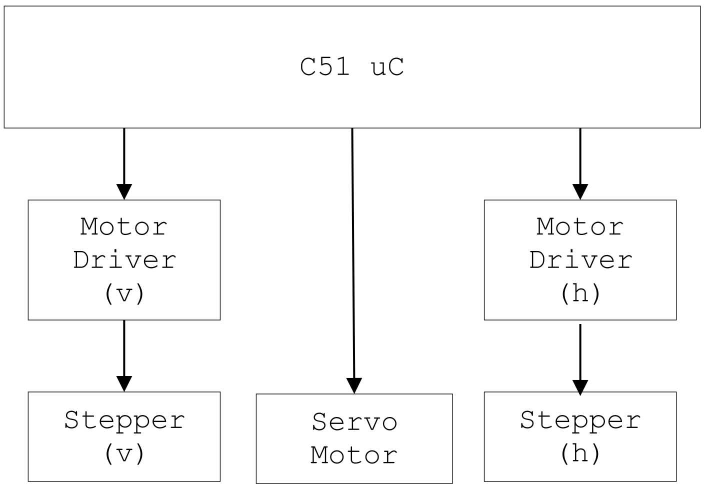
The third syscall requires special treatment because the plotter is supposed to be able to draw lines at any angle from either axes. For this reason, the 2 motors must run at speeds in proportion to the rise-over-run ratio (the slope of the line) or be supplied pulses at their STEP input at 2 different rates independently. However, there is only 1 processor core in all C51-based controllers!

To solve the problem, we decided to make use of timer interrupts. The algorithm for movev() is as follows:

```
algorithm: movev
input: target position vector, method of interpolation, time taken for this move call
output: position vector of the pen after the call finished
{
        calculate from current pen coordinates and user supplied coordinates, the change
        vector;
        set the direction of rotation of the 2 stepper motors;
        calculate the number of steps required by each motor to reach the target;
        calculate the delays between successive steps from the time taken for this move
        call, single pulse duration and count of steps to be taken in the X and Y direction;
        enable timer interrupts;
        while(target not reached)
        {
                setup 2 timers to generate interrupts after respective delays for both
                motors;
                wait for interrupts to arrive;
                step the appropriate motor in respective ISRs;
        }
        disable timer interrupts;
        return(current position vector of the pen);
}
```

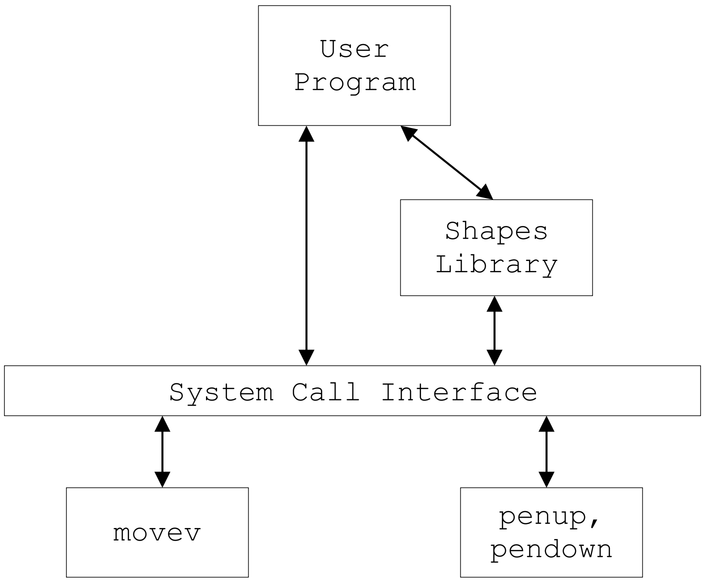
2.2 Block diagram

As stated before, we wanted the hardware to be as simple as it can be. The block schematic consists only of the following elements:



Apart from the 2 drivers, function of every other element has been cleared above. We chose the classic A4988 stepper motor drivers because they are functionally the easiest to understand for anyone who can surf the web. They act as a one-way interface between the microcontroller and the stepper, isolating the controller from power systems on the other end.

From a software perspective, the block diagram appears to be as follows:



Where a Shapes Library can be another wrapper over the library we supply, implemented by anyone seeking further interest in the machine.

2.4 Hardware Used

1. 8051-based microcontrollers (x1) [3];
2. A4988 Motor Driver Modules (x2) [4] [5];
3. NEMA-17 Stepper Motors (x2) [5];
4. S100 Mini Hobby Servo (x1);
5. Transformer, Diodes, Capacitors, and 5V, 12V 2A Regulators;
6. Linear Rail (x1);
7. 5mm OD Steel Rods (x2);
8. GT2 Timing belts and an assortment of compatible timing pulleys;
9. Some 3D Printed parts [6];
10. A Plywood Board; and
11. Connectors, Power tools, and other miscellaneous items.

CHAPTER 3

System Design

3.1 Calculating Interleaved Variable Delays

Suppose a user supplied the target vector as $(usrX, usrY)$ and the pen is currently at $(penX, penY)$.

Define:

$$\Delta x = |usrX - penX|$$

$$\Delta y = |usrY - penY|$$

The pulse train for motors begin and end with a pulse (P) with a delay (D) interleaved between successive pulses,

$$Begin - P - D - P - D - P - \ldots - D - P - End$$

i.e., the number of delays is 1 less than the number of pulses, or

$$D_* = S_* - 1; \ \ where * can\ be\ one\ of\ \{x, y\}$$

Let T be the time to complete 1 move operation. This parameter is user configurable.

Let t_p be the width of a single pulse to either motor in either direction.

Let t_{d*} be the duration of delay between successive pulses (* can be one of {x, y}).

Thus, we can say:

$$T = S_x t_p + D_x t_{dx} = S_x t_p + (S_x - 1)t_{dx} = S_y t_p + D_y t_{dy} = S_y t_p + (S_y - 1)t_{dy}$$

or,

$$t_{dx} = \frac{T - S_x t_p}{D_x} = \frac{T - S_x t_p}{S_x - 1}; \ \ and$$

$$t_{dy} = \frac{T - S_y t_p}{D_y} = \frac{T - S_y t_p}{S_y - 1}$$

hold.

If one has the distance the motor covers in one step, l_* (each for X and Y). Hence, the number of steps to cover to reach the target for either motors can be calculated as:

$$S_x = \frac{\Delta x}{l_x};\ and$$

$$S_y = \frac{\Delta y}{l_y}$$

l_x and l_y can be calculated as follows:

- Assemble the plotter completely;
- Attach the pen;
- Write a program that rotates the X-motor once by 360^o whilst keeping the pen down. Measure the length of the line and scale down by 200 for full stepping mode, 400 for half stepping mode, etc.;
- Rewrite the program to rotate the X-motor, this time by 180^o. Measure the length and scale down by 100 for full stepping mode, 200 for half stepping mode, etc.;
- Rewrite the program one last time to rotate the X-motor, by 90^o. Measure the length and scale down by 50 for full stepping mode, 100 for half stepping mode, etc.;
- Average out the scaled lengths recorded to get the value of l_x.
- Repeat the above steps for the Y-motor to get the value of l_y.

Note: more than 3 readings obviously won’t harm.

CHAPTER 4

Implementation

This chapter consists of all images taken throughout the course of the development of this project, right from 3D printing parts to assembly.

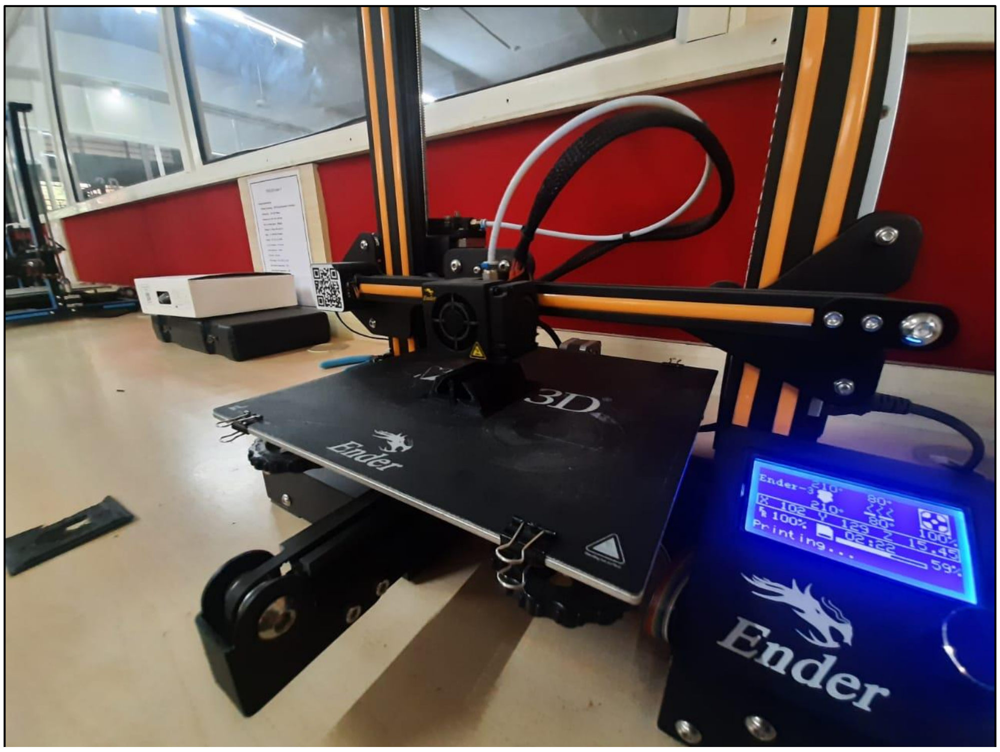


Fig. 1 Printing the pen holder [7]

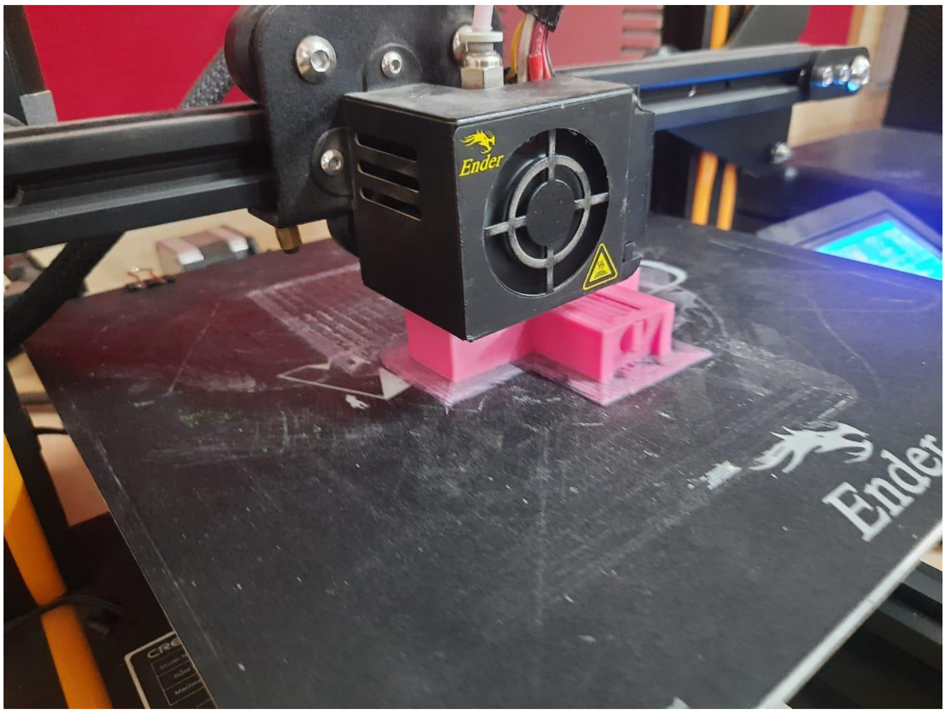


Fig. 2 Printing the raftmount

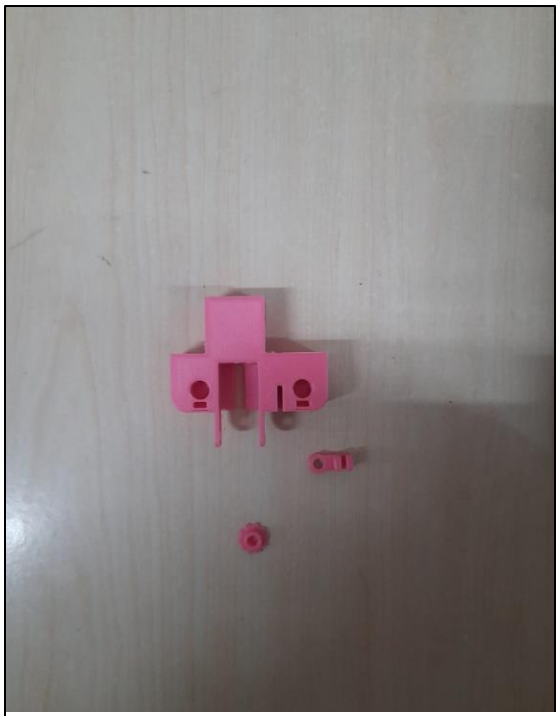


Fig.3 Printed the Y- pen end, servoarm and a finger dial

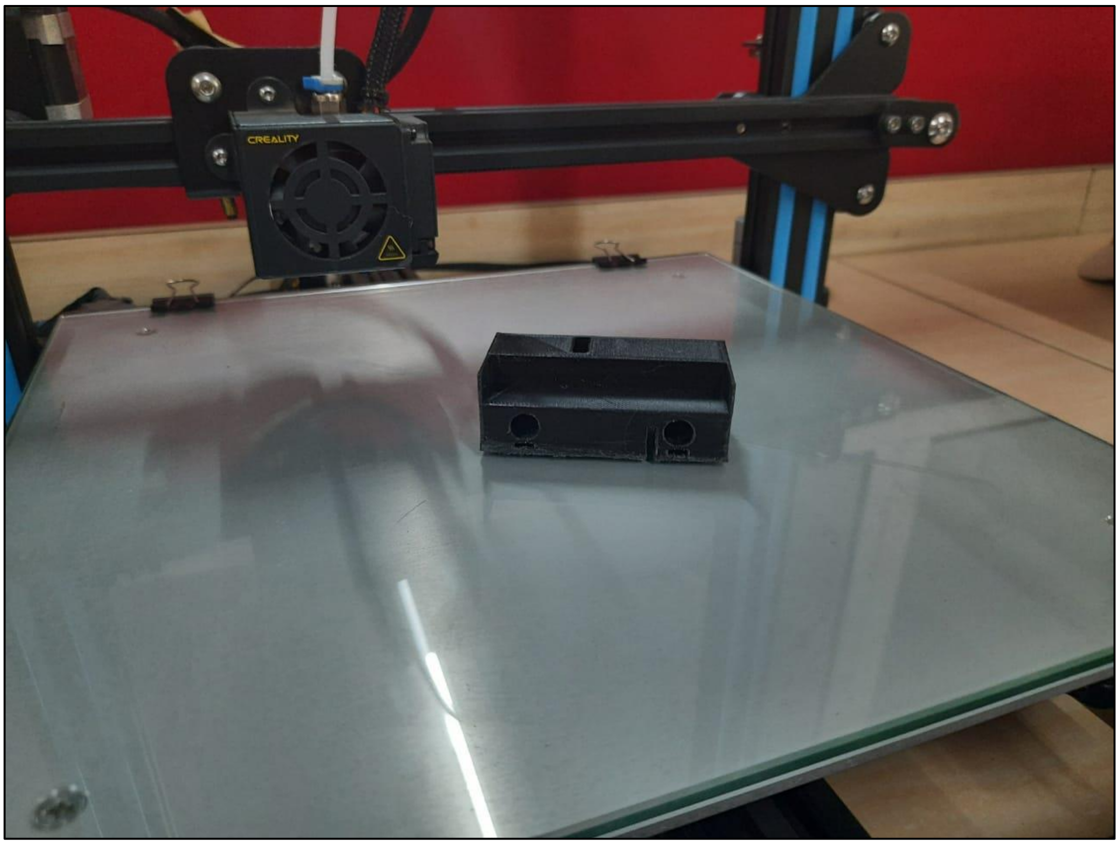


Fig. 4 Printed the Y- servo-end

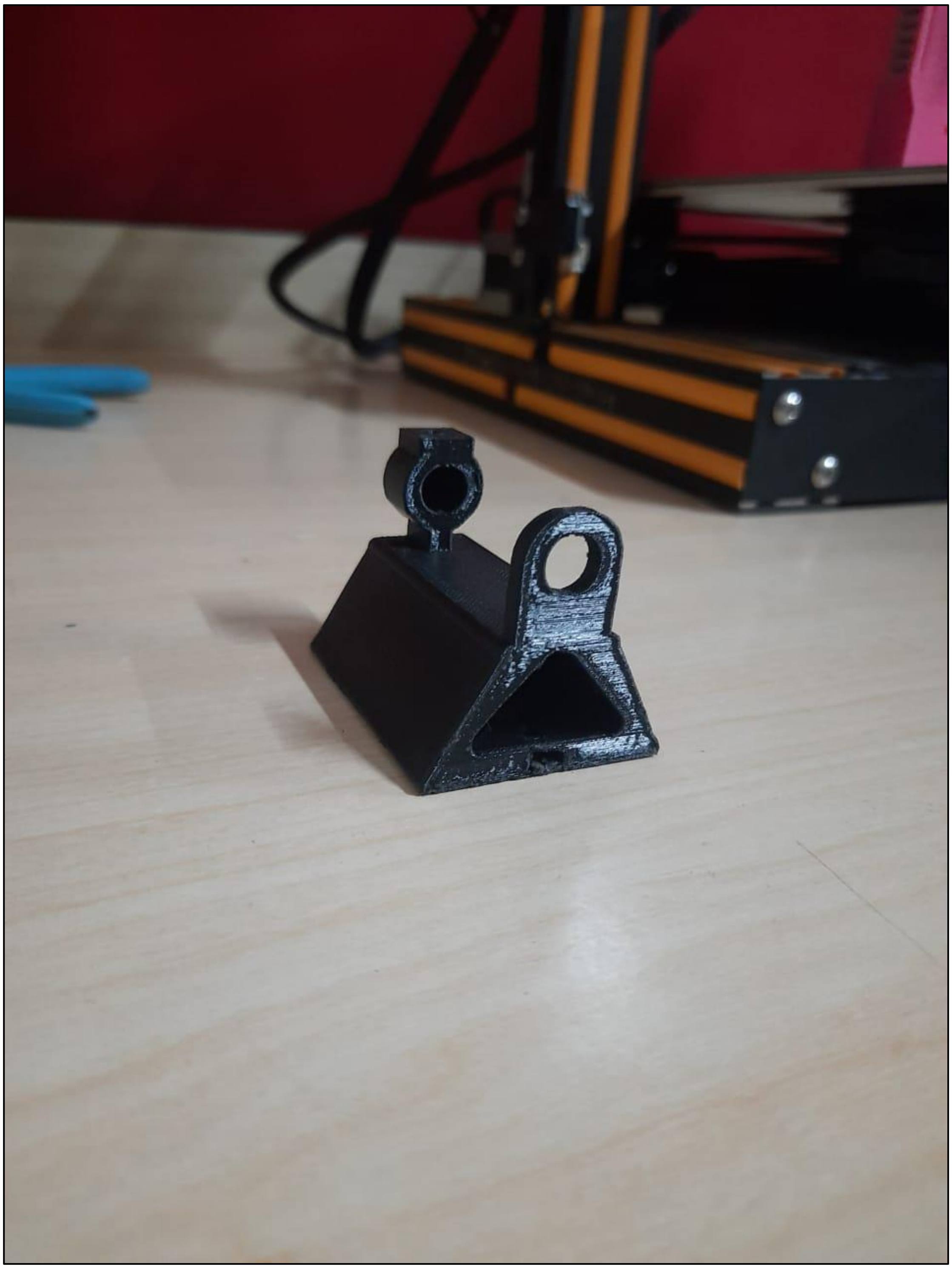


Fig. 5 Printed the pen holder

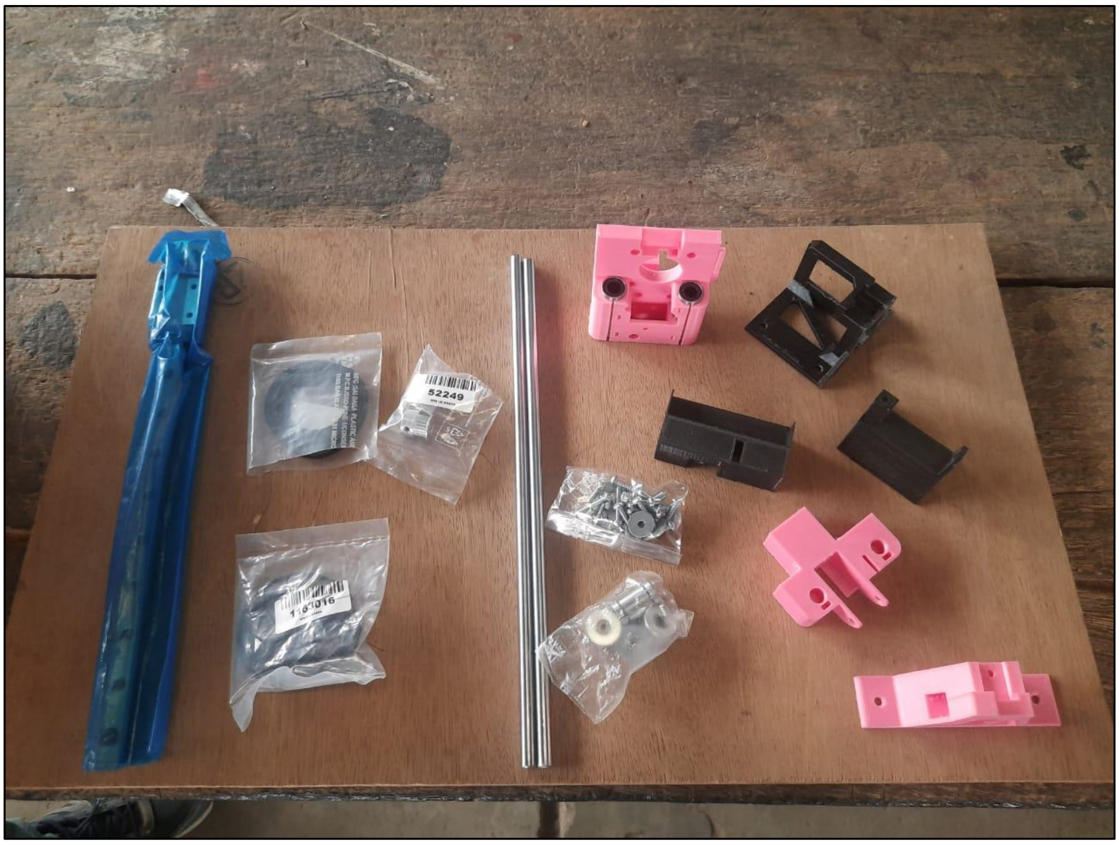


Fig. 6 All the mechanical material collected in one place

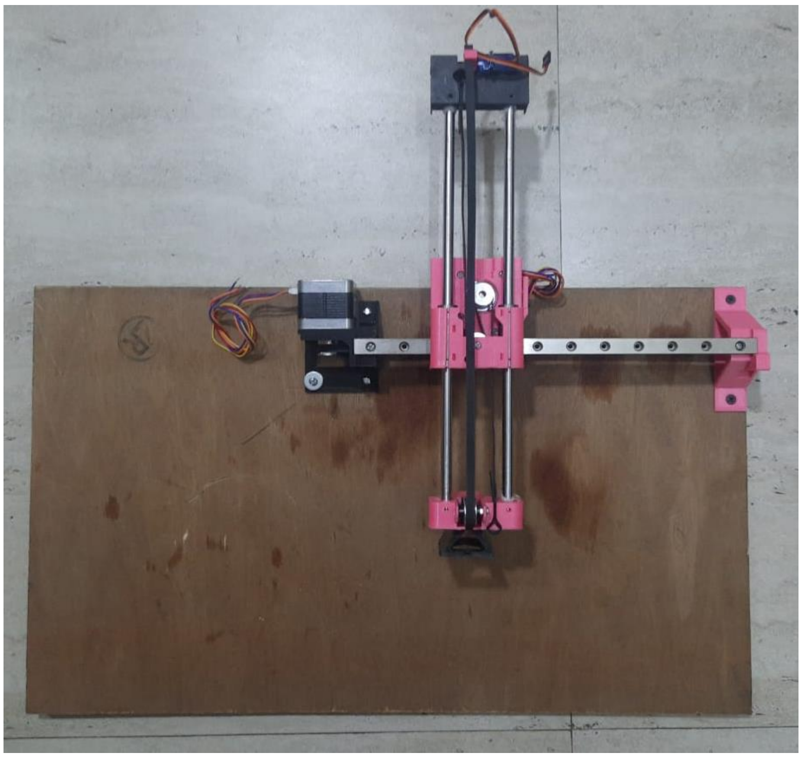


Fig. 7 Top view after complete assembly

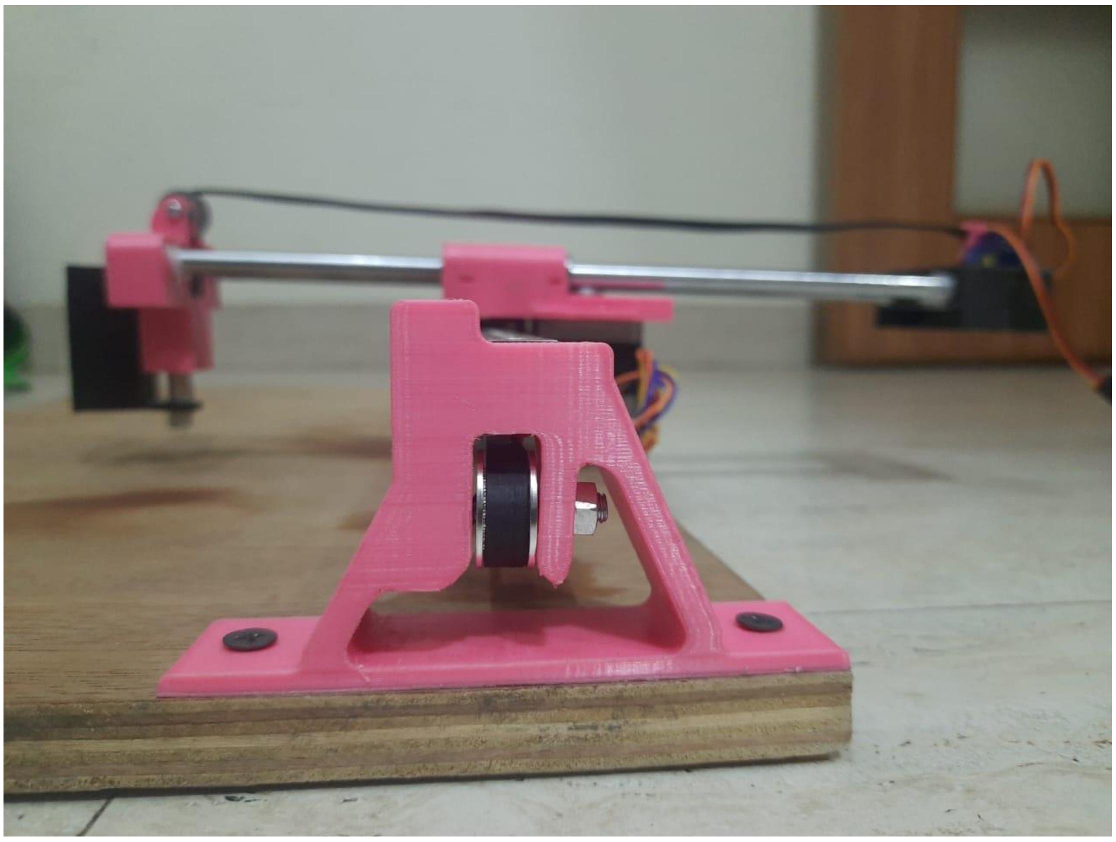


Fig. 8 RHS view after complete assembly

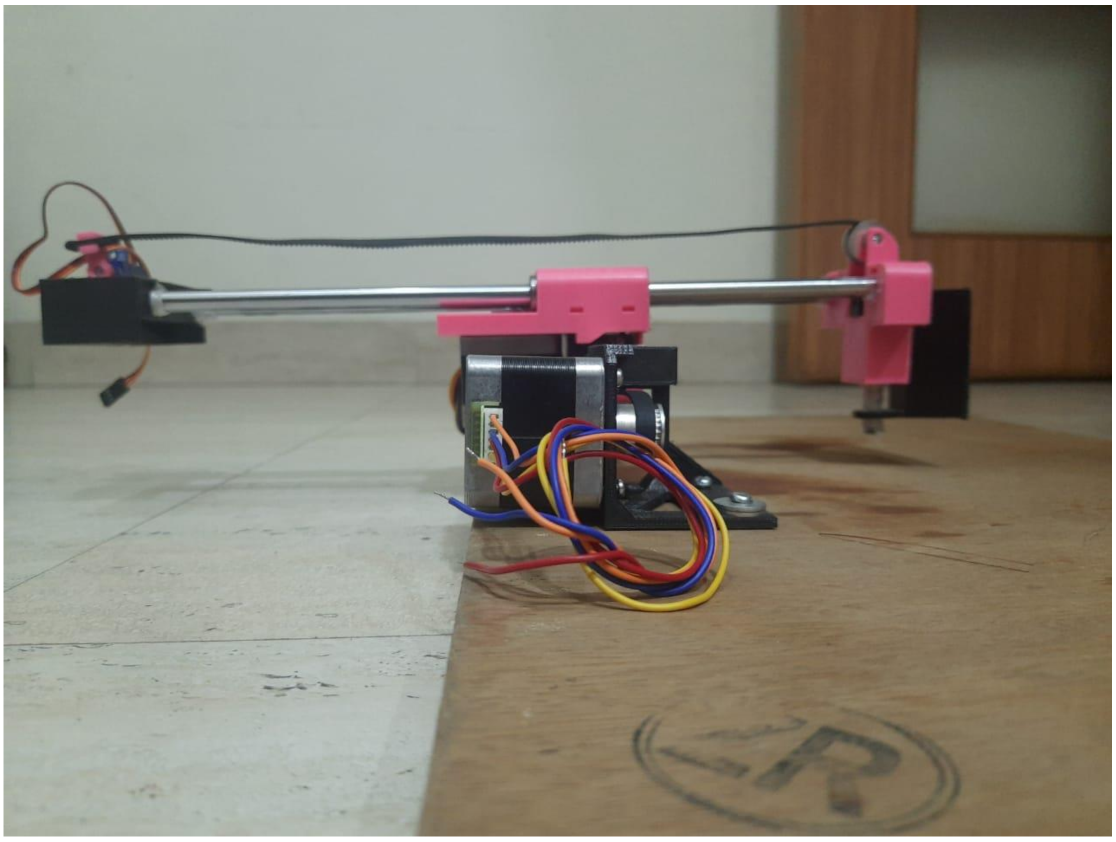


Fig. 9 LHS view after complete assembly

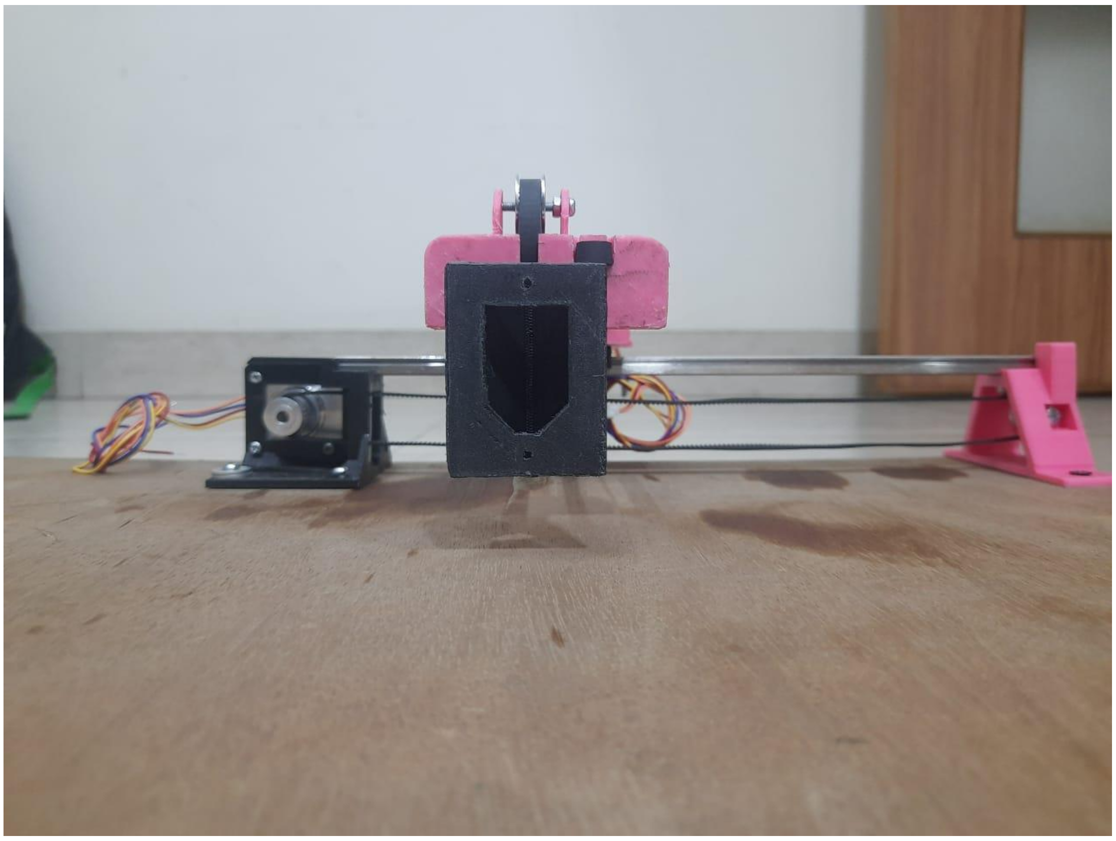


Fig. 10 Front view after complete assembly

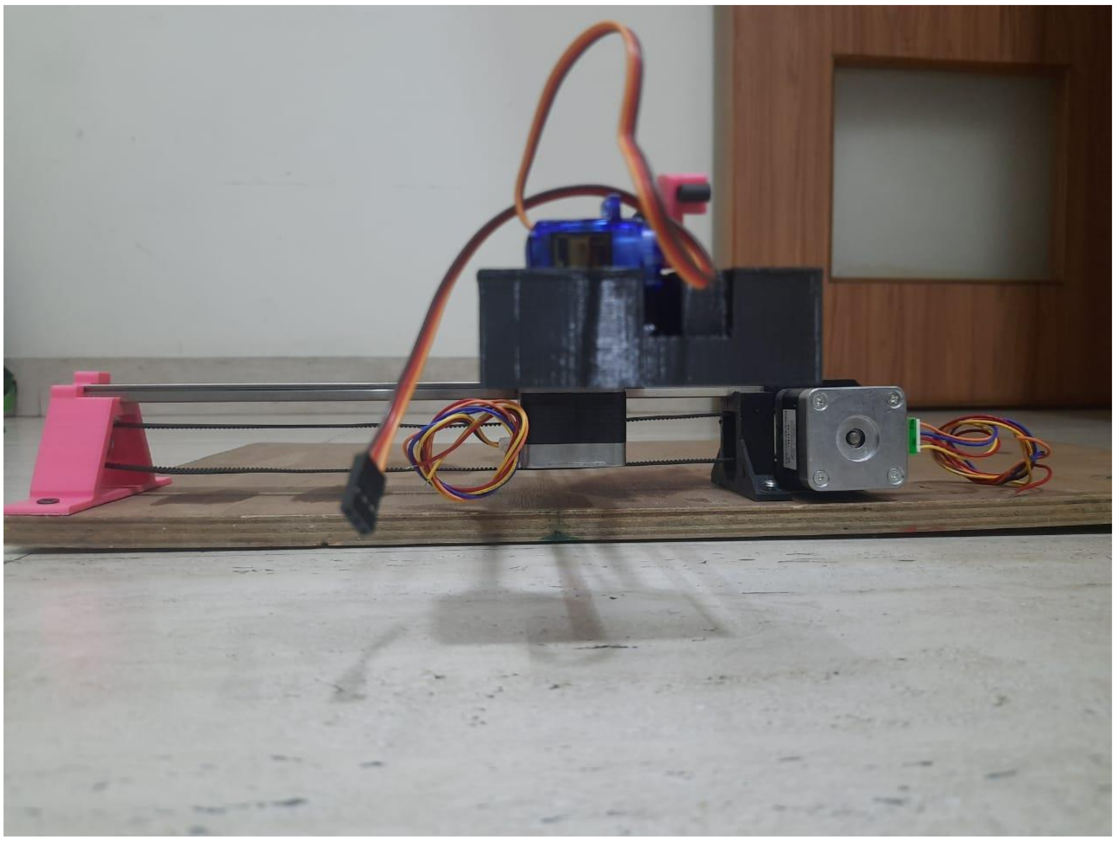


Fig. 11 Rear view after complete assembly

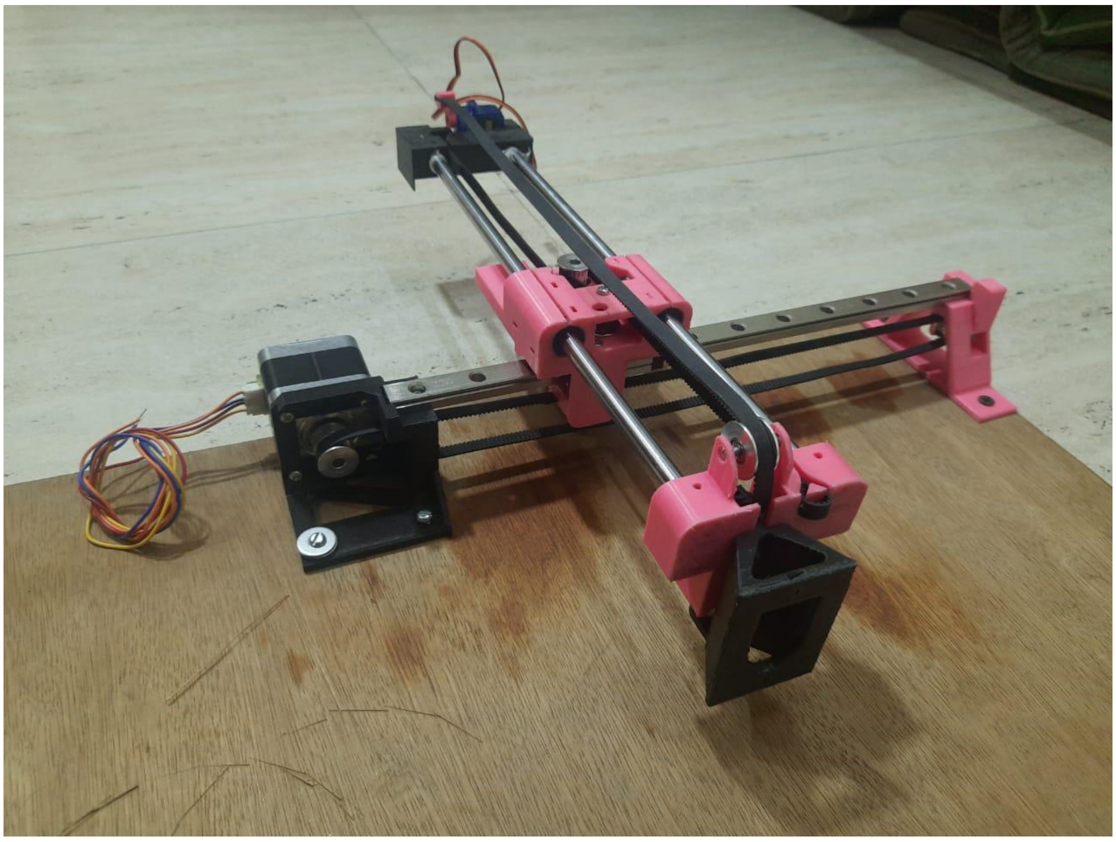


Fig. 12 Overall appearance from an angle of the assembly

CHAPTER 5

Results and conclusion

5.1 Results

The results obtained were near original requirements. A working, minimal plotter was what we expected and indeed what we had with a small C library to abstract hardware, and that is easy enough to learn to use in a day.

5.2 Conclusion

a. We designed hardware simple enough;
b. We created a PCB for the mounting components comprising the hardware;
c. We created a simple C library for user programs by providing a small set of powerful system calls that could exhaustively render any geometry on paper;
d. We built a mechanical assembly for the machine.

5.2 Future Scope

There are some obvious future enhancements/additions to the project:

1. With some additional electronics and software support, a third axis can be added so that the machine is capable of creating 3D items as well.
2. For those who know only G-code, a polite programmer may contribute a wrapper over the provided library to translate G-code into C.
3. Support for other interpolation methods like circular, elliptical, parabolic, etc. can be added.
4. The pen can be replaced by a cutting blade or a cutting laser to suit the requirement of a user.

And so on.

References

[1] A. Boban, V. S. Anandakrishnan, S. Unnikrishnan and P. B. Shibil, "2D Robotic Plotter," Painavu, 2016.

[2] M. Raut, K. Pable and P. Mulay, "Drawing Robot," International Journal of Advanced Research in Science & Technology (IJARST), Pune, 2020.

[3] ATMEL, "Electronic Components Datasheet Search," ATMEL, [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/82390/ATMEL/AT89S52.html. [Accessed 03 2022].

[4] Pololu Robotics and Electronics, "A4988 Stepper Motor Driver Carrier," [Online]. Available: https://www.pololu.com/product/1182. [Accessed 04 2022].

[5] Schneider, "NEMA17 Datasheet Preview," Schneider, [Online]. Available: https://datasheet4u.com/datasheet-pdf/Schneider/NEMA17/pdf.php?id=1260602. [Accessed 04 2022].

[6] D. Machines, "Super Easy 3D Printed Arduino CNC Drawing Machine | GRBL Plotter Elegoo," [Online]. Available: https://www.youtube.com/watch?v=XYqx5wg4oLU.