Design And Development Of A Mini CNC Machine Using Arduino

Dissertation submitted to

SGTB Khalsa College, DU

For the award of degree of

Bachelor of Science

in

ELECTRONIC SCIENCE

By

Tarun Thakur

(2017ELE2001)

Under the supervision of

Dr. Inderpreet Singh

Dr. P. Arun

Department Of Electronics

Sri Guru Tegh Bahadur Khalsa College

University of Delhi

CERTIFICATE

This is to certify that the dissertation entitled **"Design And Development Of A Mini CNC Machine "** being submitted by Mr. Tarun Thakur to the SGTB Khalsa College, University of Delhi, Delhi for the award of degree of Bachelor of Science in Electronic Science, is the original work carried out by him. The results contained in the thesis have not been submitted anywhere in part or full for the award of any other diploma or degree.

Tarun Thakur

(Candidate)

Dr. Inderpreet Singh	Dr. P. Arun
Department Of Electronics	Department Of Electronics
Sri Guru Tegh Bahadur Khalsa College	Sri Guru Tegh Bahadur Khalsa College
University Of Delhi, North Campus	University Of Delhi, North Campus
New Delhi - 110007, India	New Delhi - 110007, India

CONTENTS

Chapter1: Introduction

1.1 What is CNC?

It refers to a protocol or logic by which computer is able to draw a predefined pattern or design through a hardware setup. The logic involves representation of the drawing patterns by their corresponding coordinates (x, y, z) numeric values. These numeric values help in precisely controlling the movement of machinery parts for replication of the designed pattern in the real world [1]. The software develops an instructruction set that helps in step by step designing of the predefined patterns by controlling the movement of drawing tool. CNC has also find ample scope in the cutting and drilling industry. The CNC machines are so precise that even nanometeric scale patterns can be easily designed by them. A user can produce any design without specifically programming the hardware every time a new design is to be made.

1.2 Basic Components of a CNC machine:

Input Device: A USB port is utilized for transferring the data from computer to the memory card of CNC machine.

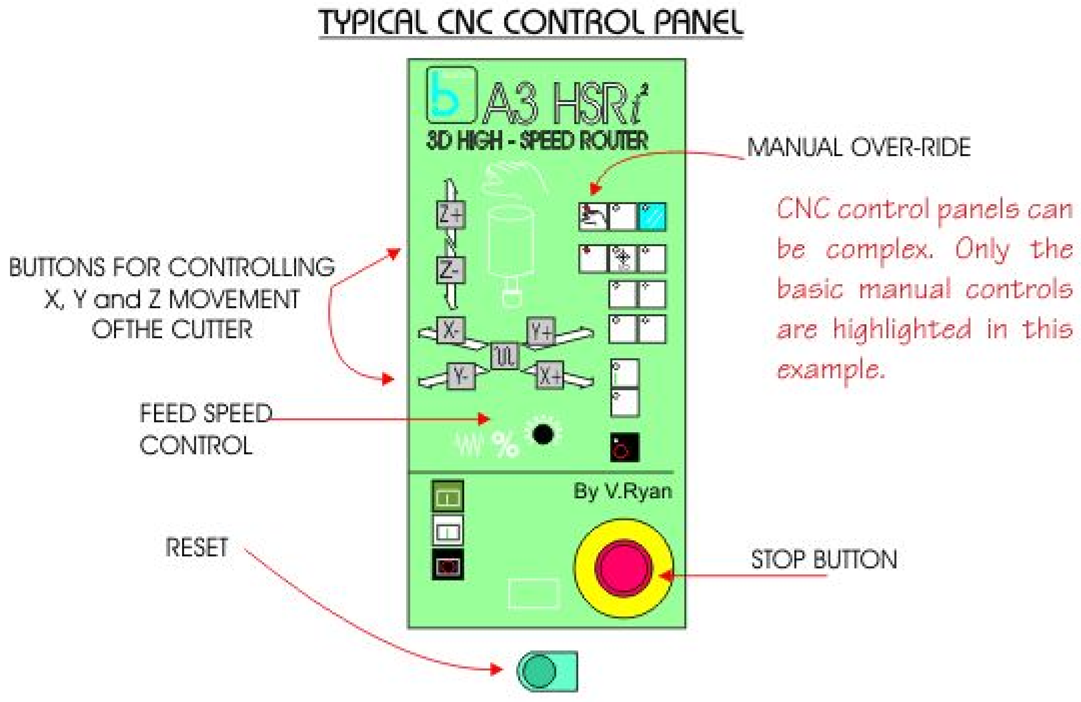
Central Processing Unit: It reads and decodes the data in the memory unit and generates appropriate signals to generate position control and velocity control signals for the hardware attached.

Motion Control: The drawing or cutting tool of the machine can move in two or more directions. The position of the tool along with these directions is controlled by a stepper motor (for horizontal directions x and y) and a servo motor (for vertical direction z).

Machine Control Panel: It consists of some buttons used to initialize the machine or terminate operation in case of emergency. A typical machine control panel is shown in Figure 1.

Reset Button: It is used to move the tool to origin i.e. (0,0,0) position. If not pressed, it is possible that the machine starts cutting the material at wrong position.

Stop Button: It is used to stop the machine immediately. It is pressed when the user feels that the machine is not working in expected manner.



TYPICAL CNC CONTROL PANEL
A3 HSRt²
3D HIGH - SPEED ROUTER
MANUAL OVER-RIDE
CNC control panels can be complex. Only the basic manual controls are highlighted in this example.
BUTTONS FOR CONTROLLING X, Y and Z MOVEMENT OFTHE CUTTER
Z+
Z-
X-
Y+
Y-
X+
FEED SPEED CONTROL
%
By V.Ryan
RESET
STOP BUTTON

Figure 1: A Typical CNC Control Panel [2]

1.3 Working

As already explained, CNC Machine starts working after a file containing an instruction set with coordinates (in a predefined manner) is loaded into the system. The user has to load this file, which is also known as G-code file, and initial position of the tool is set to origin and finally, the run button is pressed.

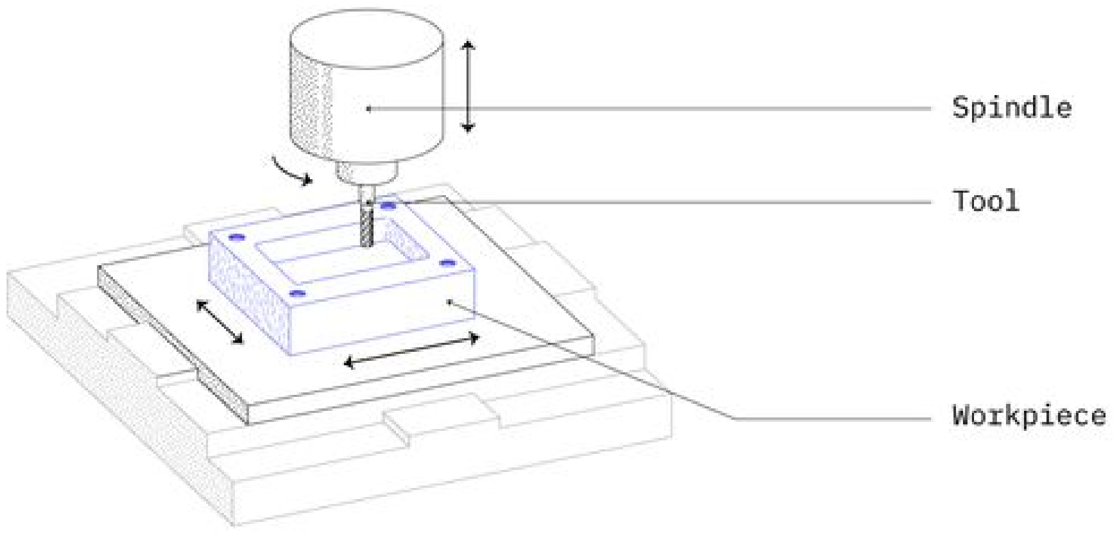


Figure 2: A Typical Setup Of A CNC Tool [3]

After that each line of the G-Code file is read and decoded by processing unit of the machine. Subsequently, depending upon the instructions in the G-Code file, position and speed control signals are generated by the processor to drive the stepper and servo motors to produce the desired pattern.

1.4 History Of CNC Machine

The idea of numerical control started when the automation of machine tools originally incorporated specific concepts of programmable logic [4]. The development of Numerical Control machines started back in 1940s [5]. Afterwards, a slightly more advanced machines came in 1950s, which were though based on existing tools but modified with motor controls. These controls were developed to follow specific instructions that were fed into the machine using punched tape [6]. These early mechanisms were soon improved with both analog and digital computers in 1990s [7]. The introduction of computer technology into the concept of numerical control led to what we now know as Computer Numerical Control.

1.4.1 Early MIT Research on Numerical Control

After World War II, John Parsons researched ways to improve aircraft by creating stiffened skins for them [8]. This eventually led to a series of important Air Force research projects, which were conducted at the Massachusetts Institute of Technology (MIT). This research began in 1949. After the early planning and research phases, an experimental milling machine was designed at MIT. Professor J.F. Reintjes and his team of researchers were involved in this project [9].

1.4.2 The First CNC Machine

Before the MIT projects, a system was developed by Parsons Corporation in Traverse City, Michigan to produce templates for helicopter blades. John Parsons, the founder of the company , discovered a method to calculate airfoil coordinates on an IBM 602A multiplier. He then fed these data points into a Swiss jig borer [The **jig borer** is a type of machine tool invented at the end of World War I to make possible the location of hole centers quickly and precisely]. It is considered to be the first true numerical control machine as it was able to manufacture goods – helicopter blade templates, in this case.

As numerical control technology moved into the 1960 and 1970s, the CNC machine started taking its modern shape [10]. As digital technology evolved, automation in production processes became more efficient than ever. Nowadays, many individuals can purchase – and even design – their own homemade CNC machines. Because of advancement in computer industry , it’s more common than ever to find CNC machines in all industries.

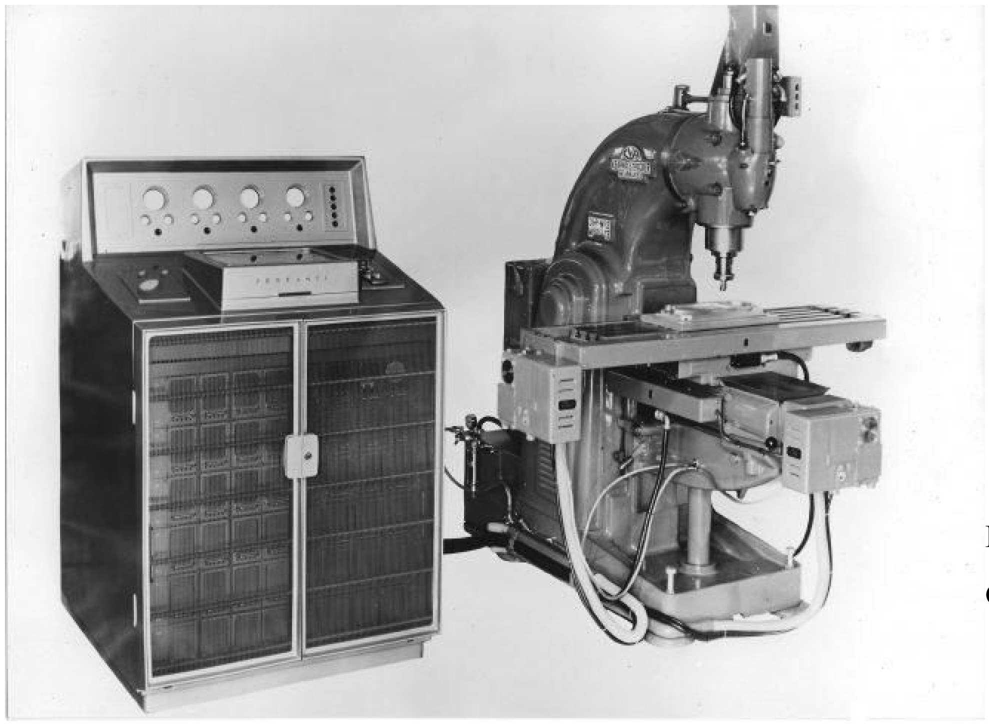


Figure3: First CNC machine

1.5 Present Scenario

Owing to its ability to work continuously and low labour requirement, CNC makes it possible to create anything with reduced cost of production. Today, all kinds of businesses rely on CNC to produce precisely machined parts with unmatched consistency. Some of these industries are:

- Aerospace Industry: While manufacturing aircrafts, precision is very important. Without precise parts, aircrafts are not safe to fly in. Hence, aerospace industry favors CNC machining because it can easily manage hard to cut materials.
- Automative Industry: The parts in vehicles vary in size and shapes. CNC makes possible the production of these parts at large scale keeping the production lines moving.
- Electronic Industry : As the hardware is getting more compact, precision has become very important. CNC is used to make printed circuit boards (PCB) that can work with consistency and reliability.

1.6 CNC Machine As Hobby And Motivation For Present Work

Presently, numerous Do It Yourself enthusiasts are trying to design a CNC Machine that is capable of 2D printing but the best that could be done till now is a X-Y plotter using Arduino software, inkspace, processing and G-Code. Using four different softwares for a simple 2-D printing makes it a complex task. Apart from the technical qualification required to operate the software, there have also been compatibility issues everytime a new update is released. Also the proposed prototypes till date have been heavily dependent on G-Code, which although quite popular in the industry, is out of the scope of a layman's understanding. Therefore, in the present work, we have developed a GUI by which CNC machine can be operated by drawing patterns on a canvas. In addition to this, the software is designed in such a way that conventional G-Code files can also be used for designing. The option of drawing a 2D plot by loading an image file has also been kept in the software. So, it allows three distinct ways to read the pattern to be designed i.e. drawing, G-Code, and image file.

The software has been written in python and is designed to automatically detect the port to which Arduino is connected and is able to automatically configure various initial hardware and software parameters.The interesting part about this CNC machine is that the hardware part is developed by material available in old computer lab. At last, it also provides provision for attaching multiple tools for 2D pattern generation i.e. ink, laser and needle.

Chapter2 : Implementation

2.1 Hardware

The X and Y axes for the CNC machine are derived from CD drives. The stepper motors alongwith their to and fro mechanism are taken from old CD drives. The drives are carefully disassembled and the assembly was taken off the rails as shown below.

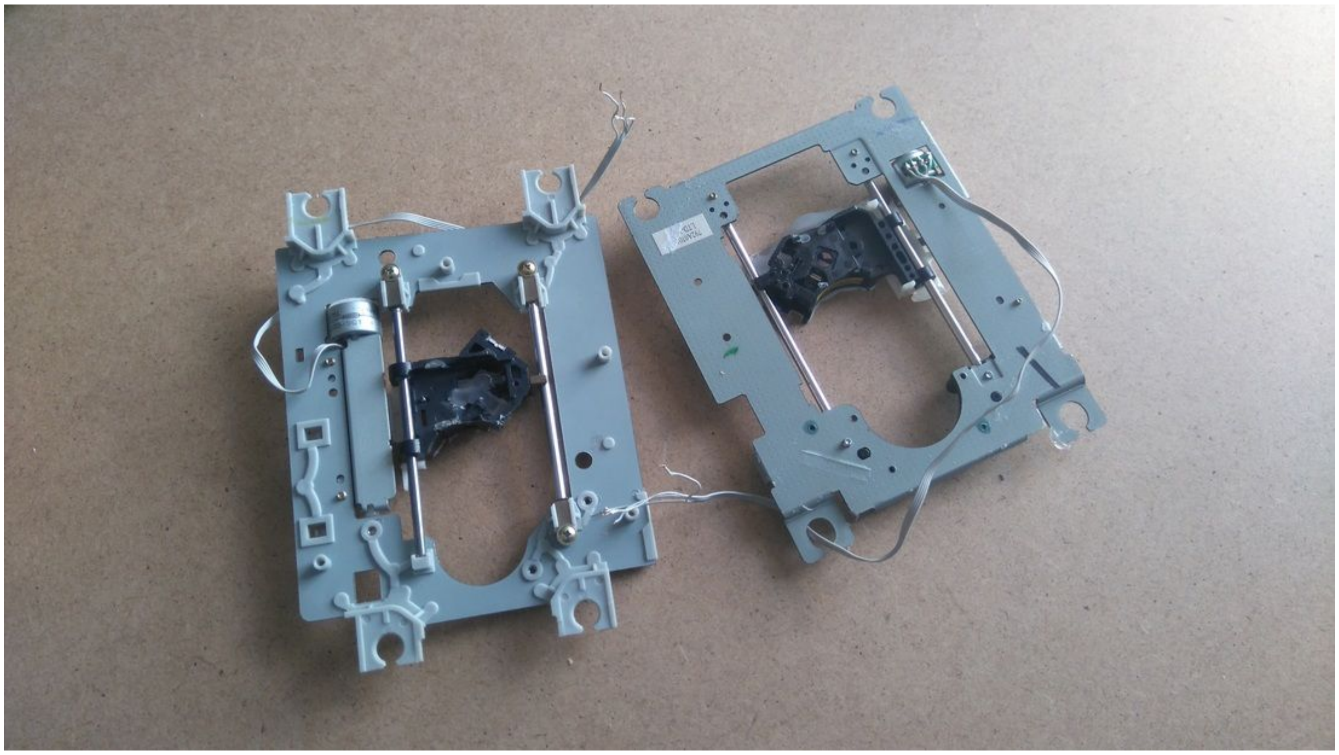


Fig 4(a) : Disassembled CD drives used for X and Y axis motors in the project

The Z axis motion is controlled using a servo motor with rack and pinion gear assembly. Also, a solid base is attached to the bottom. Finally, the wires are arranged in a way so that they fit inside the base and a small surface at the side is cut to make way for the usb port.

The X axis is attached to a plastic frame and then screwed to the base perpendicularly. On the surface of the Z axis, we attach a pen holder. Now, we attach the z axis on the X axis so that the pen can move in horizontal direction. At last, we attach a solid surface on the Y axis on which we put a paper/pcb board. The printing area is 10.5 cm X10.5 cm.

Sss

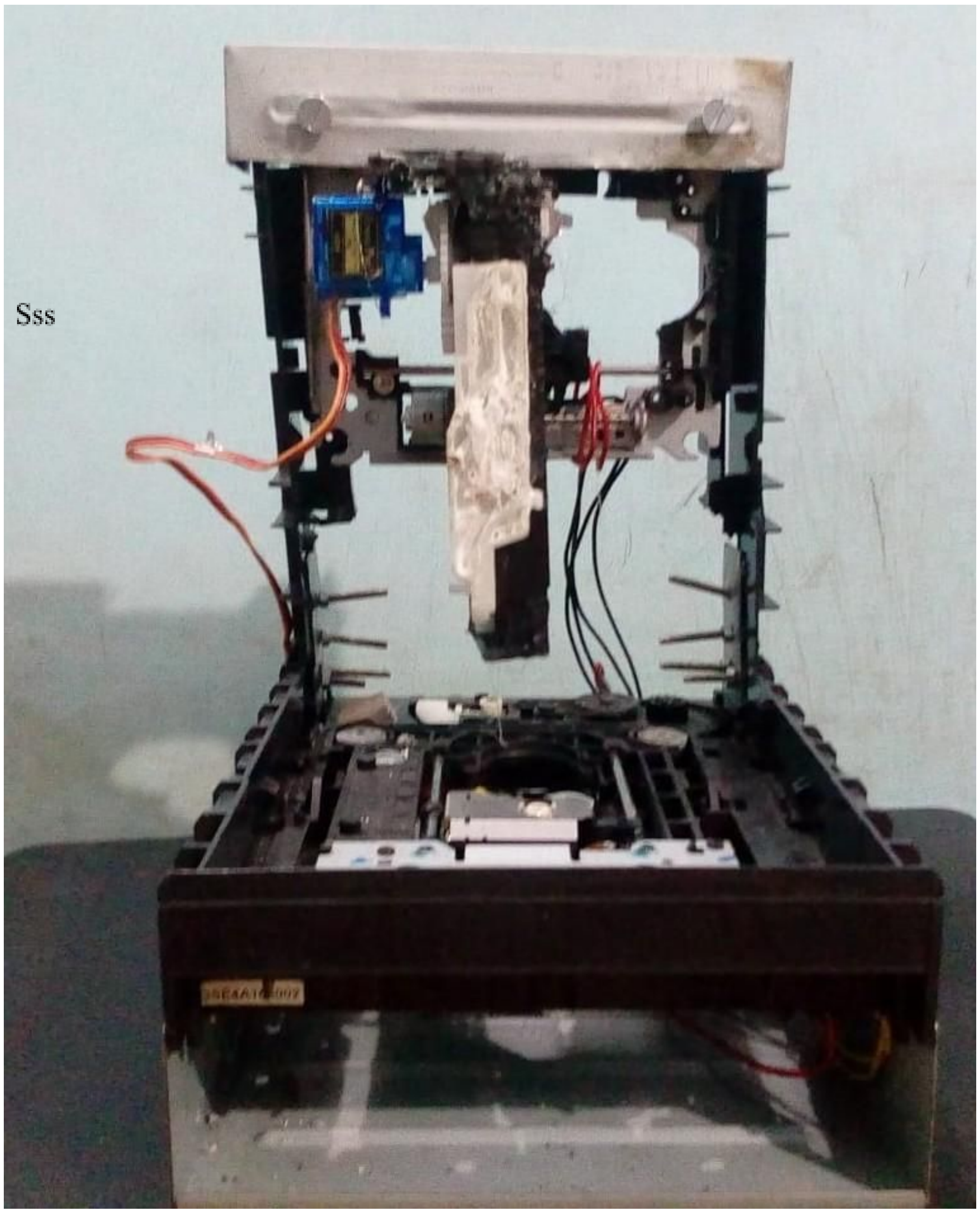


Fig 4(b) Front view of the proposed CNC machine

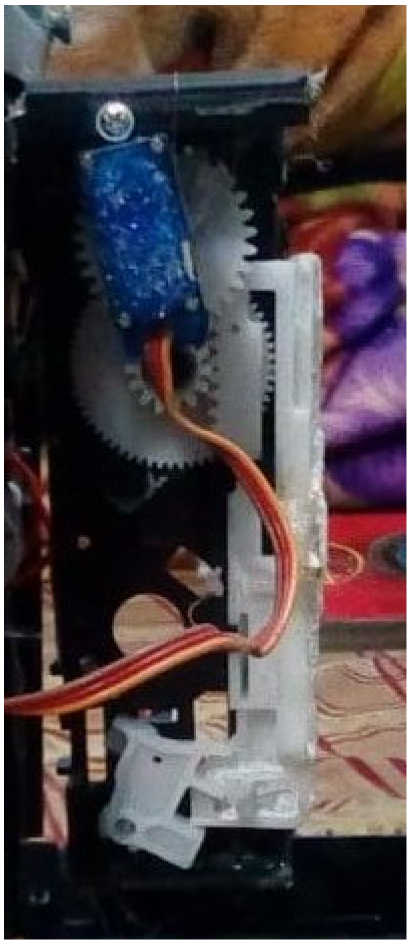


Fig 4(b) Z axis gear assembly

2.1.1 Working Of Stepper Motor

In a stepper motor, the angle through which the shaft of the motor rotates by a fixed angle for each discrete pulse. The angle through which the stepper motor shaft turns for each pulse is referred as the step angle. The step angle is decided by the number of input pulses given. Hence, the position of shaft is controlled by controlling the number of input pulses.

The internal diagram and connection of stepper motor in the machine is as follow:

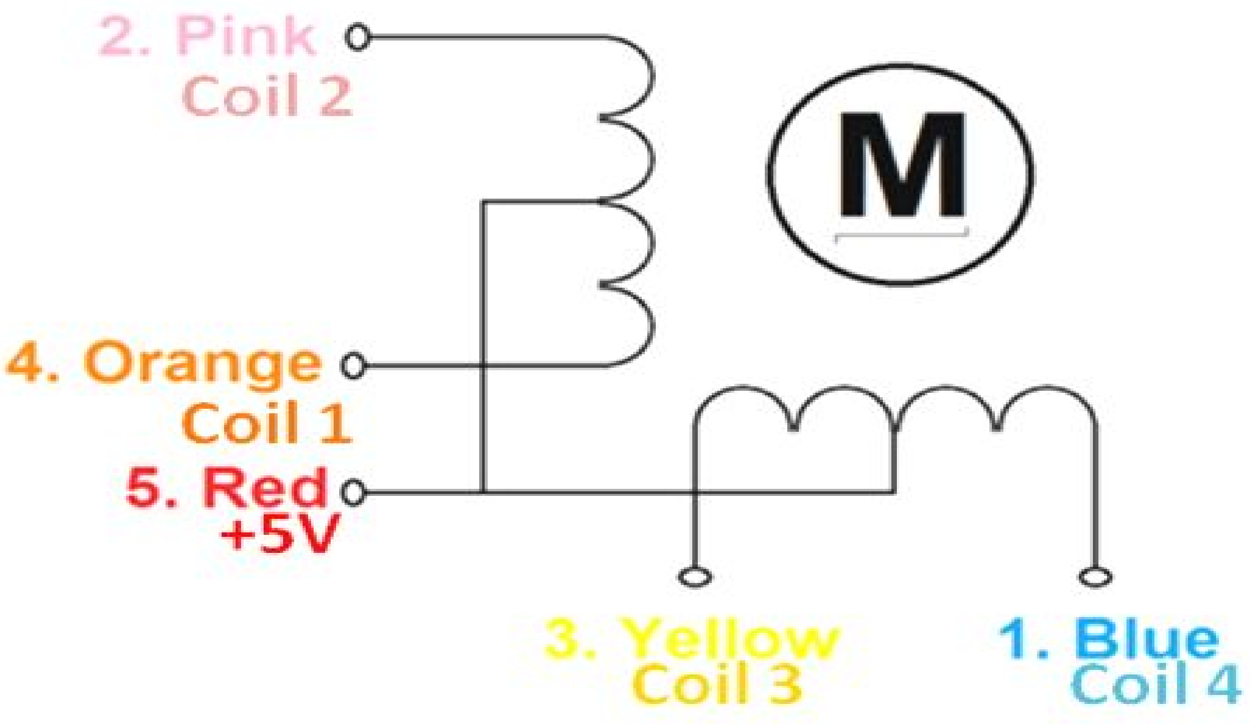


Fig 6(a) : Stepper motor internal

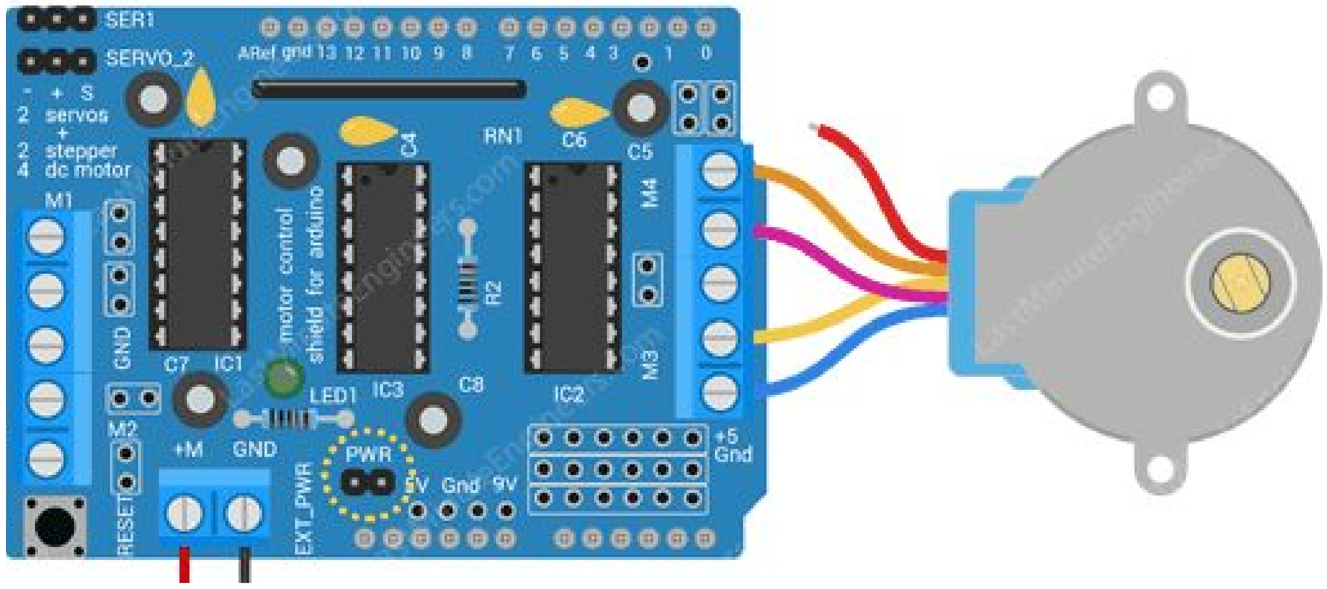


Fig 6(b) : Stepper Motor Connections

2.2.1 Working Of Servo Motor

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly and a controlling circuit. The gear assembly reduces the RPM and increases the torque of the motor. Initially, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. When an electrical signal is given to another input terminal of the error detector amplifier, the difference between these two signals, one comes from potentiometer and another comes from other source, will be processed in feedback mechanism and output will be provided in term of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with potentiometer and as motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

For testing the stepper and servo motor, we write a simple arduino program to print a square. In case the stepper motors don't work in expected manner, the wires should be connected to different pins or the L293D IC should be changed

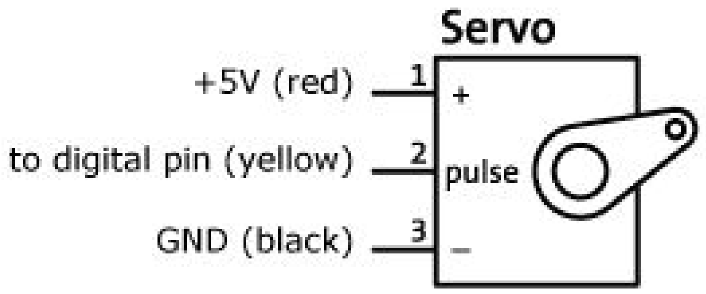


Fig 7 : Pin diagram of Servo motor with gear assembly

2.1.3 Connections :

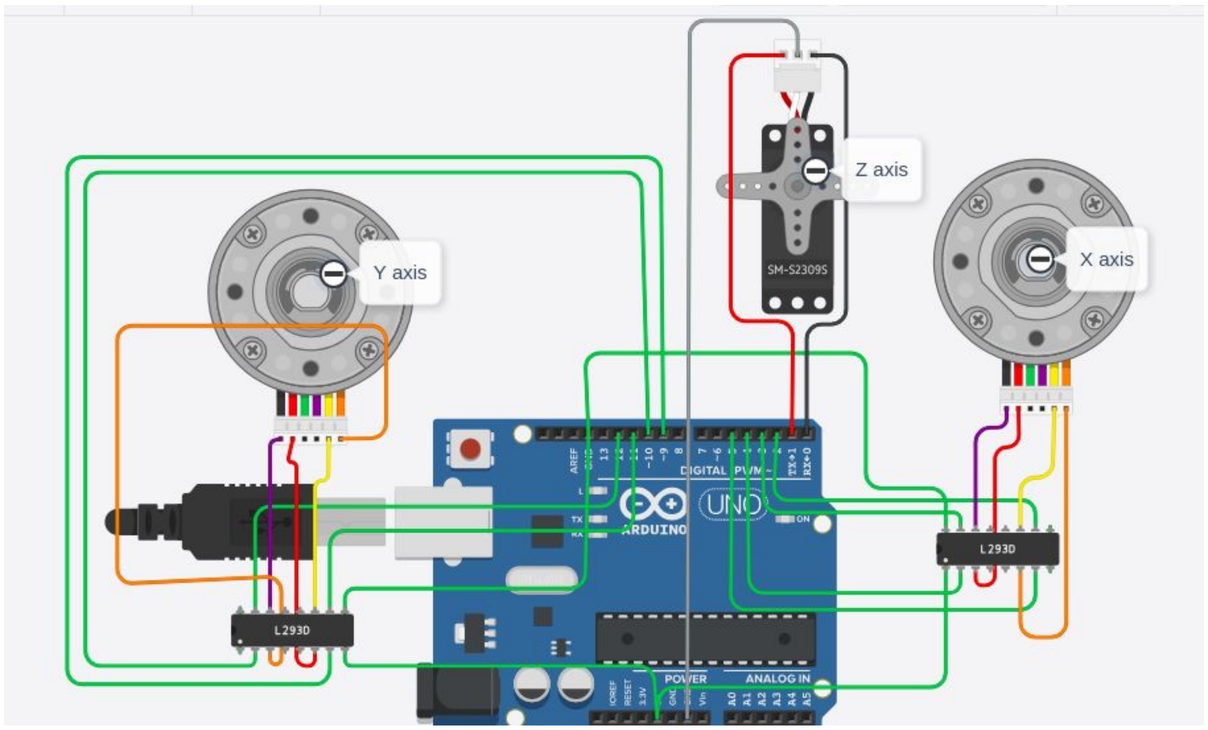


Fig 5 : Hardware Connections

2.2 Graphical User Interface

The graphical user interface (GUI) for the software is developed in python. Python is an interpreted, object oriented, high-level programming language with dynamic semantics. Its high level built in data structures make it very attractive for Rapid Application Development, as well as for use as a scripting language. Python supports modules and packages, which encourages program modularity and code reuse. There are a number of libraries available for python development that can be used without any change on any operating system [11-13]. One of the python libraries widely used for developing GUI is ‘tkinter’. It is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python.[14-16].

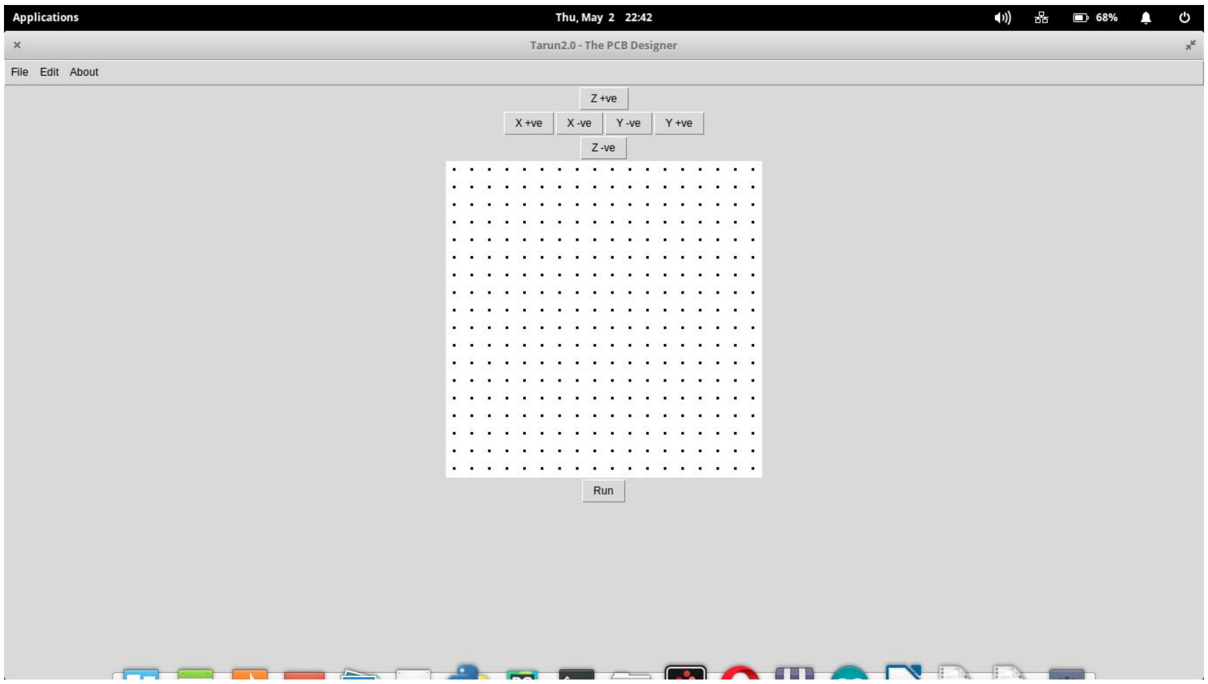
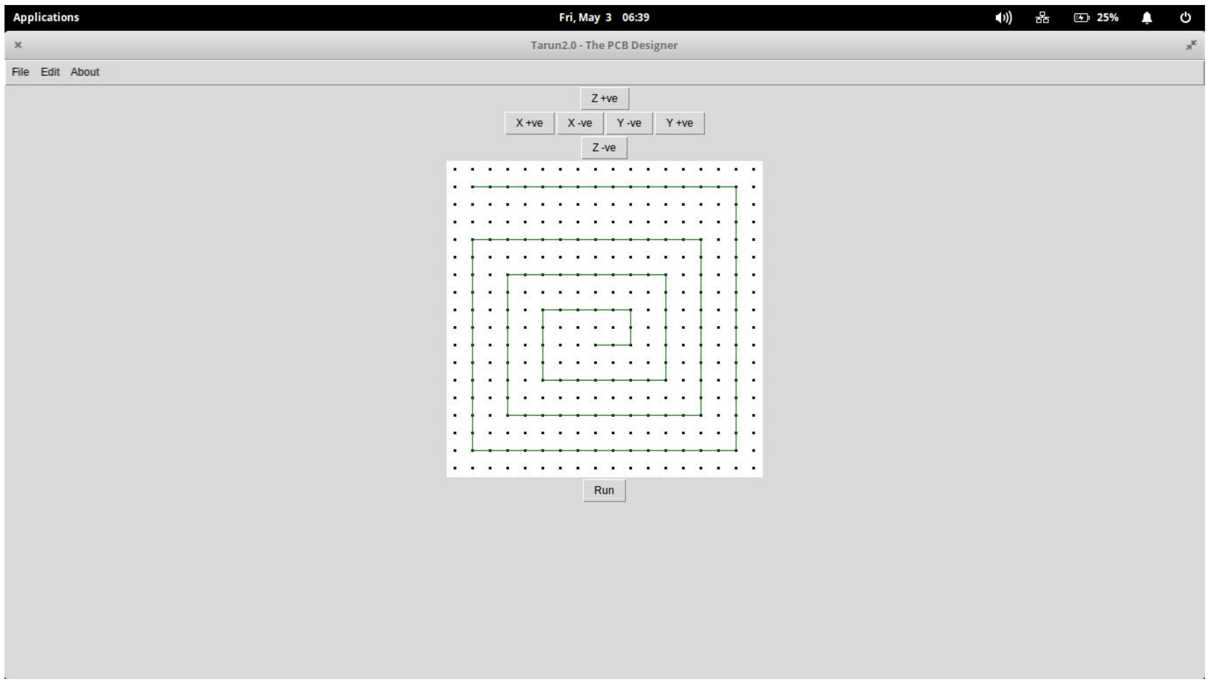
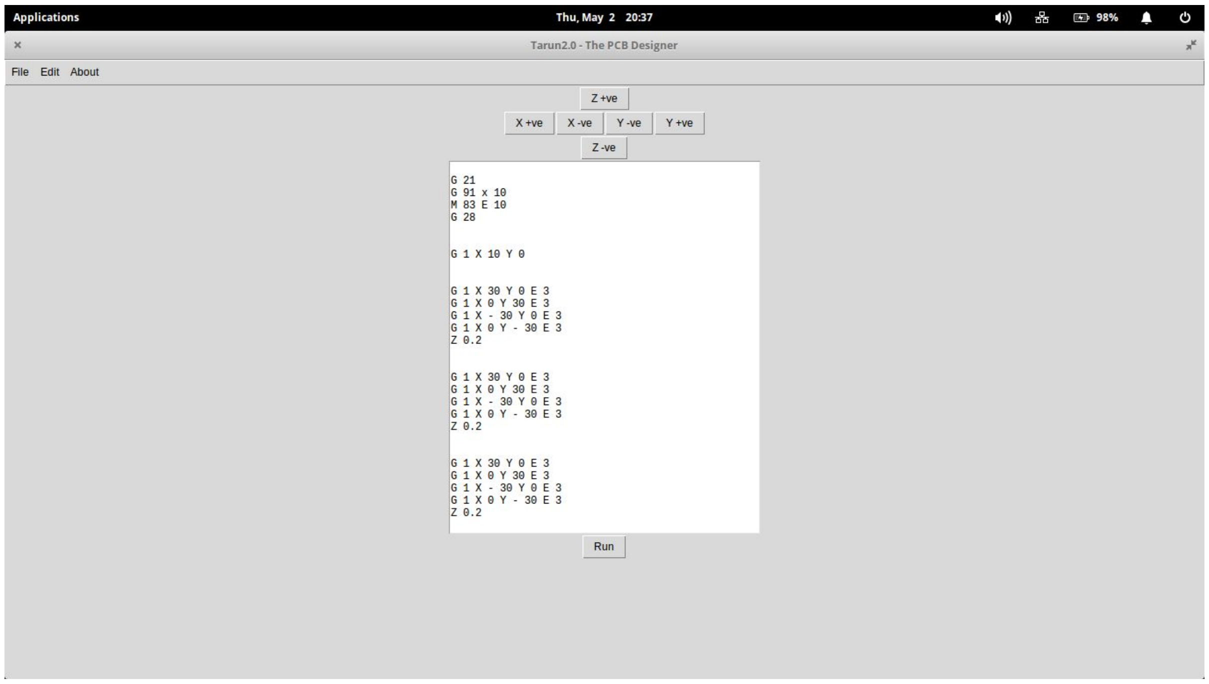


Fig 8 : Screenshot of graphic user interface

The GUI provides a set of buttons – up,down,right and left – to initially configure the axes to origin. It also provides an option to either use your existing G-Code and image files or to make your own pattern from the scratch. The procedure to generate a 2D pattern using the graphic user interface is given as follows:

- For making his/her sketch, the user is provided with a dotted canvas. This helps user to draw the pattern. Apart from these, a python script is written to fix any possible human error during drawing(say, a line that is not perfectly straight). When the user enters “Run” button, the algorithm designed calculates the length of each section with direction in which it is located and a corresponding command to the microcontroller is sent for moving the tool.
- To generate the pattern using a G-Code file or image file, the user has to load the source file. After loading the source file, the user just has to press ‘run’ button to start printing.The algorithm is capable of decoding the G-Code file and image file and convert its content into micro-controller command using python regular expressions.





```
G 21
G 91 x 10
M 83 E 10
G 28


G 1 X 10 Y 0


G 1 X 30 Y 0 E 3
G 1 X 0 Y 30 E 3
G 1 X - 30 Y 0 E 3
G 1 X 0 Y - 30 E 3
Z 0.2


G 1 X 30 Y 0 E 3
G 1 X 0 Y 30 E 3
G 1 X - 30 Y 0 E 3
G 1 X 0 Y - 30 E 3
Z 0.2


G 1 X 30 Y 0 E 3
G 1 X 0 Y 30 E 3
G 1 X - 30 Y 0 E 3
G 1 X 0 Y - 30 E 3
Z 0.2
```

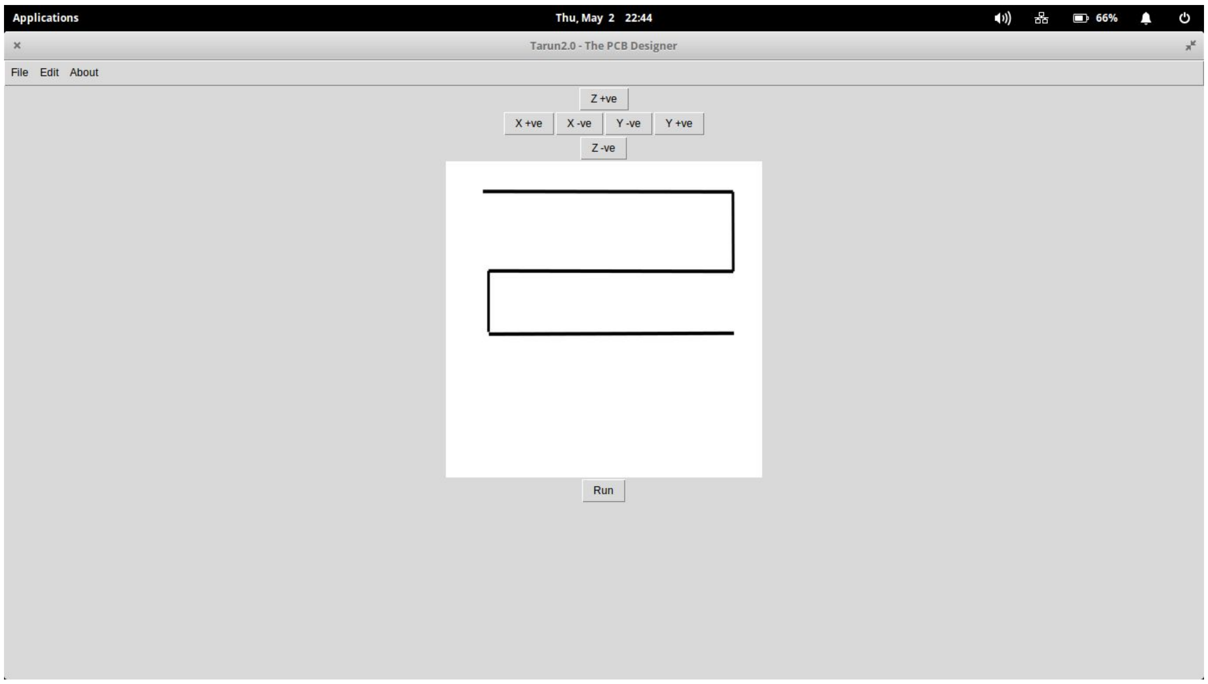


Fig 9: Screenshots of designs made/uploaded by user

2.3 Arduino

Arduino is an open source single-board microcontroller kit for building digital devices and interactive objects that can sense and control both physically and digitally. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. [17]

2.4 Serial Communication

To integrate the hardware and the software, serial communication is used. serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. [18] For every movement, a particular command is sent to the microcontroller which then drives the direction and duration of the motor's movement. The rotational movement of the stepper motor is converted into linear motion by using lead-screw mechanism and the same for servo motor at Z axis is done using gear-assembly.

2.5 Motor Driver

In this machine, a L293D controller chip has been used to control the stepper motor. The block diagram and connections are shown as follows:

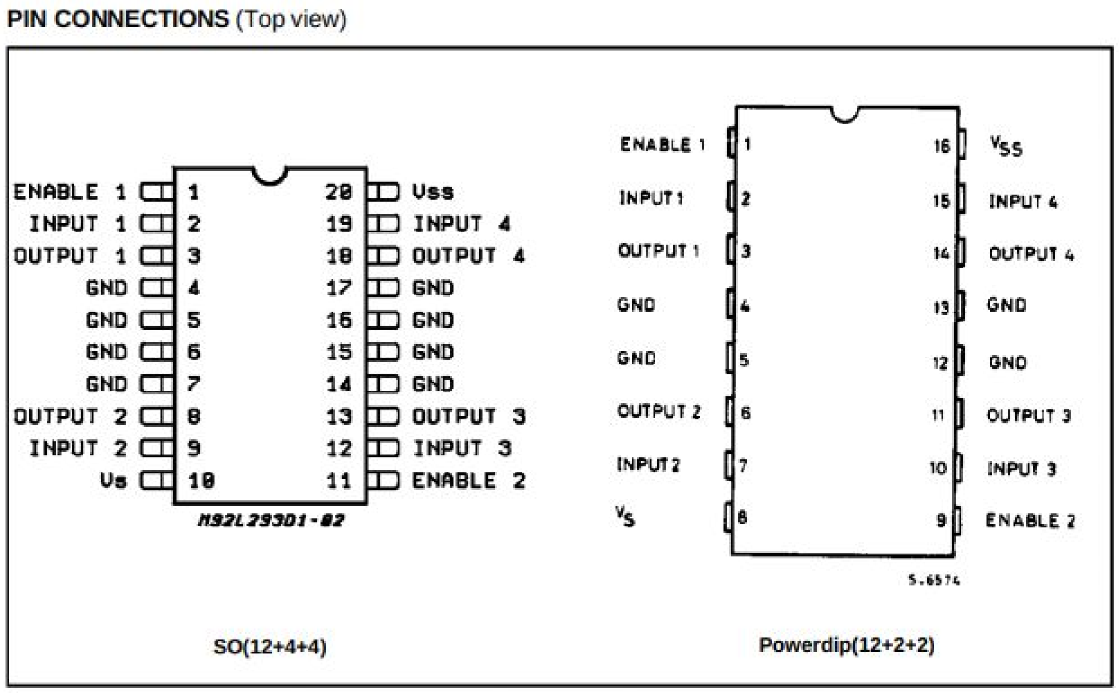


Fig 10(a) : Pin diagram of IC L293D

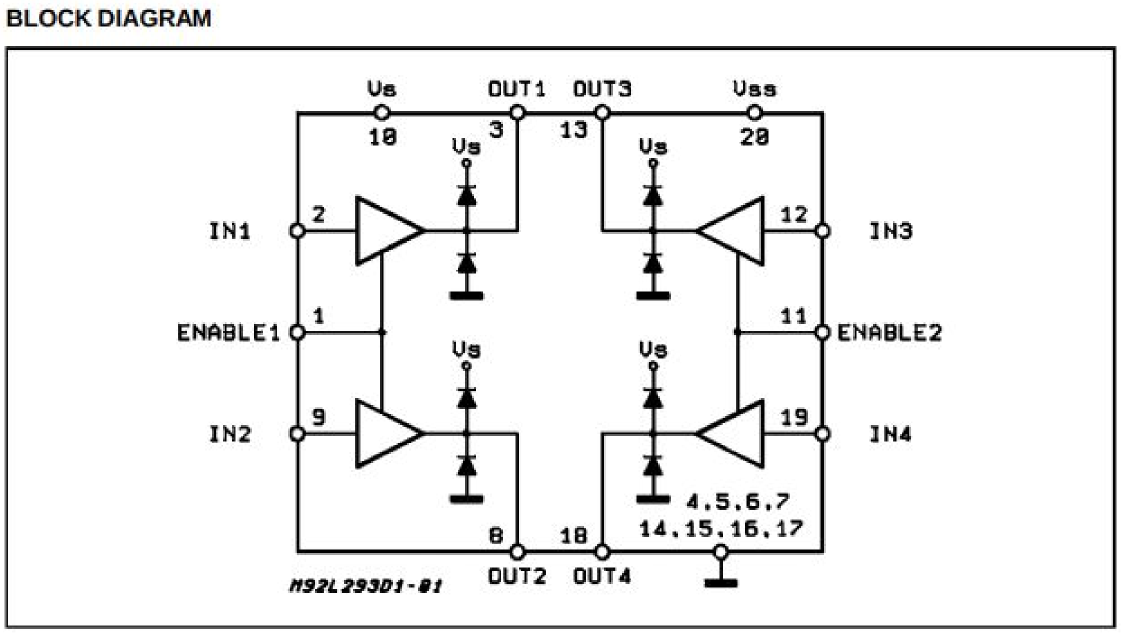


Fig 10(b) : Block diagram of IC L293D

- **Why stepper motors were used?**

  It can be controlled at low cost and obtain high torque at the startup and low speed .

  The construction of the motor is simple and it is able to operate in open loop control system.

  The rotation angle of the motor is propotional to the input pulse and the motor has full torque at stand still.

2.6 Lead Screw Mechanism

A lead-screw turns rotatory motion into linear motion combining a screw and a nut where the screw thread is in direct contact with the nut thread.

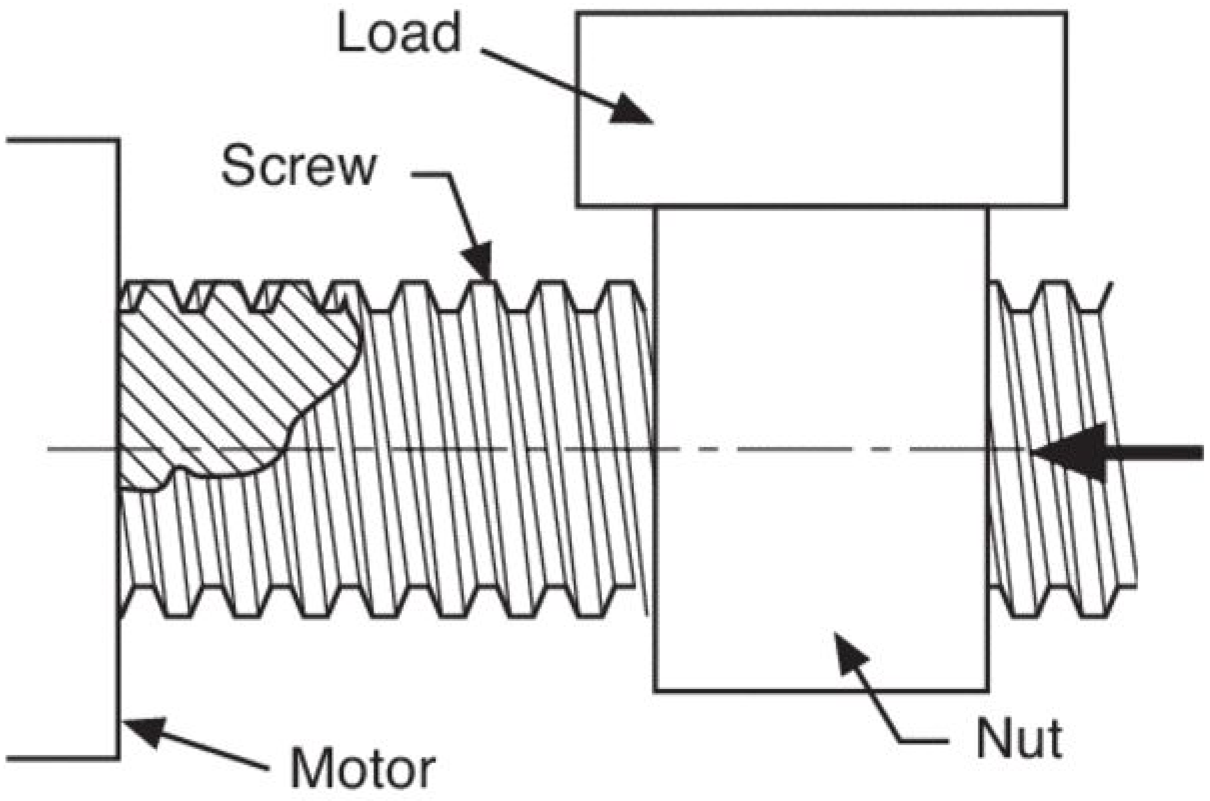


Fig 11 : Lead-screw mechanism

Lead-screw uses a helix angle of the thread to convert rotatory motion into linear motion. Small springs are inserted to the lead screw construction to maintain traction of the thread and the skid. It also acts as an emergency apparatus ,that is , to absorb overshoot problem of the motor. The advantages of lead-screw mechanism are as follows:

- Large load carrying capacity
- Precise and accurate linear motion
- Smooth, quite and low maintenance
- Self lock

2.6.1 Mechanics

The mechanics of the lead-screw mechanism is as follows:

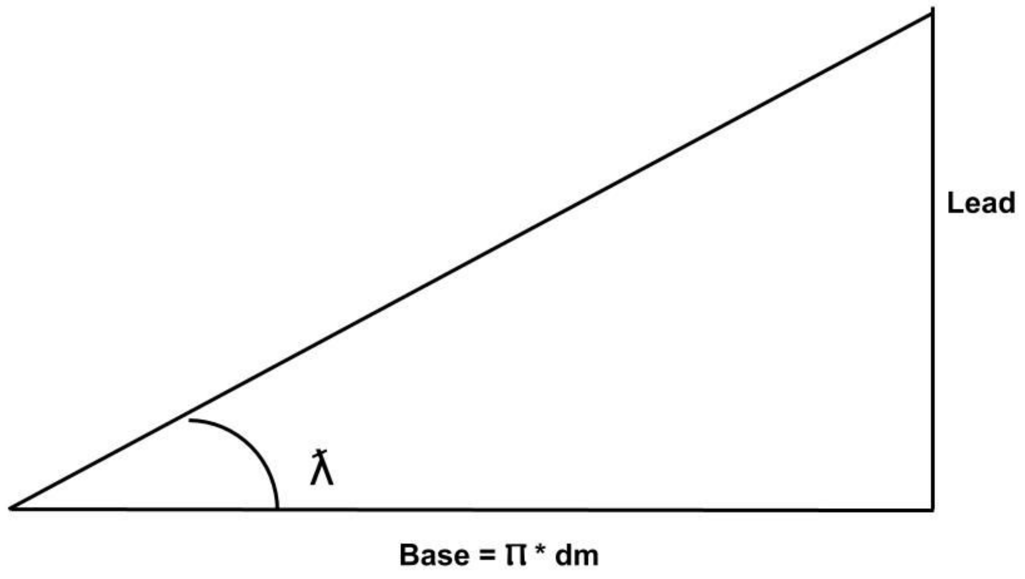


Fig 12 . Diagram of an “unwrapped” thread

The torque required to lift or lower the load can be calculated by “ unwrapping” one revolution of a thread. The unwrapped thread forms a right angle triangle where the base is π*dm long and the height is lead.

The force of the load is directed downwards, the normal force is perpendicular to the hypotenuse of the triangle, the frictional force is directed in the opposite direction of the direction of the motion.

Using the free body diagram, the torque required to the lift or lower a load can be calculate

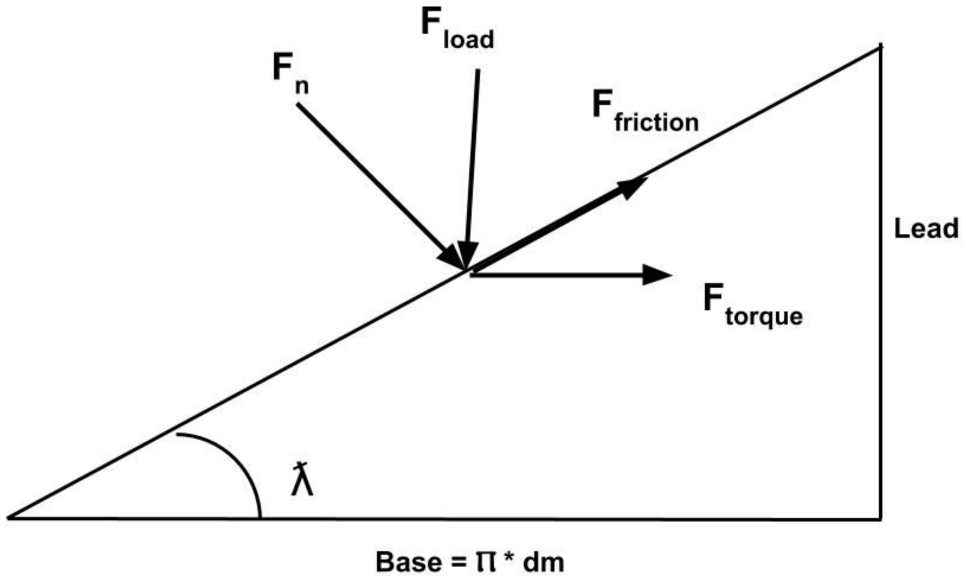


Fig 13 : Free Body Diagram Of An Unwrapped Thread

$$T_{lower} = \frac{Fload * dm}{2} \cdot \frac{[\Pi * \mu * dm - l]}{\Pi * dm + \mu * l}$$

$$= \frac{F\,load * dm}{2} \cdot \tan(\phi - \lambda)$$

and

$$T_{raise} = \frac{Fload * dm}{2} \cdot \frac{[\Pi * \mu * dm + l]}{\Pi * dm - \mu * l} = \frac{F\,load * dm}{2} \cdot \tan(\phi + \lambda)$$

Where ,

dm = mean diameter

ϕ = angle of friction

Based on T_{lower} equation, it can be found that the screw is self-locking when the coefficient of friction is greater than the tangent of the lead angle. When this is not true, the screw will back-drive or lower under the weight of the load.

The efficiency, calculated using the above torque equation is:

$$efficiency = \frac{To}{Traise} = \frac{Fload}{2\pi * Traise} = \frac{(\tan(\lambda))}{(\tan(\phi+\lambda))}$$

2.6 Rack and Pinion arrangement

A rack and pinion is a pair of gears which convert rotational motion into linear motion. The circular pinion engages teeth on a flat bar – the rack. The rotational motion applied to the pinion will cause the rack to move to the side.

The gear ratio of gears G1 and G2 is 5:1 which means that one complete rotation of G1 corresponds to 5 rotations of G2. This helps us control the vertical movement of the pen using a servo motor. A small angular displacement of stepper motor is capable of producing significant vertical displacement on the pen.

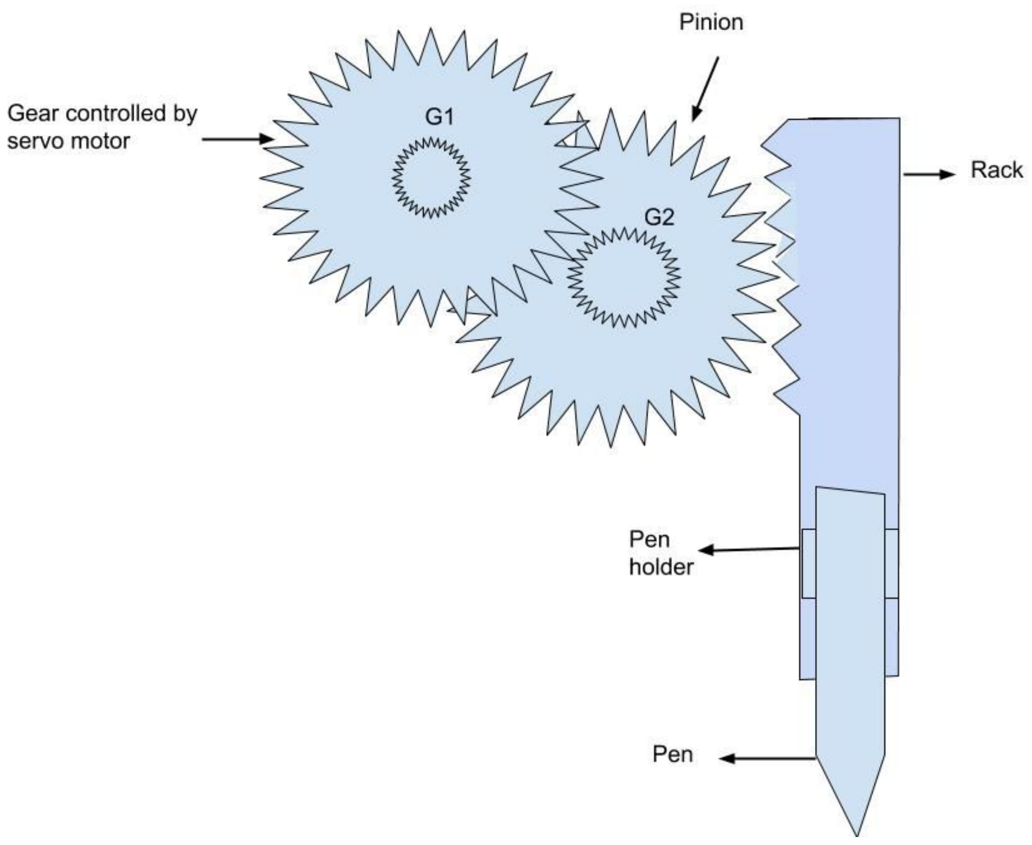


Fig 14: Rack And Pinion Arrangement

Chapter 3: Programming

Arduino Programming:

Flow chart:

The flow chart for the arduino program is as follows:

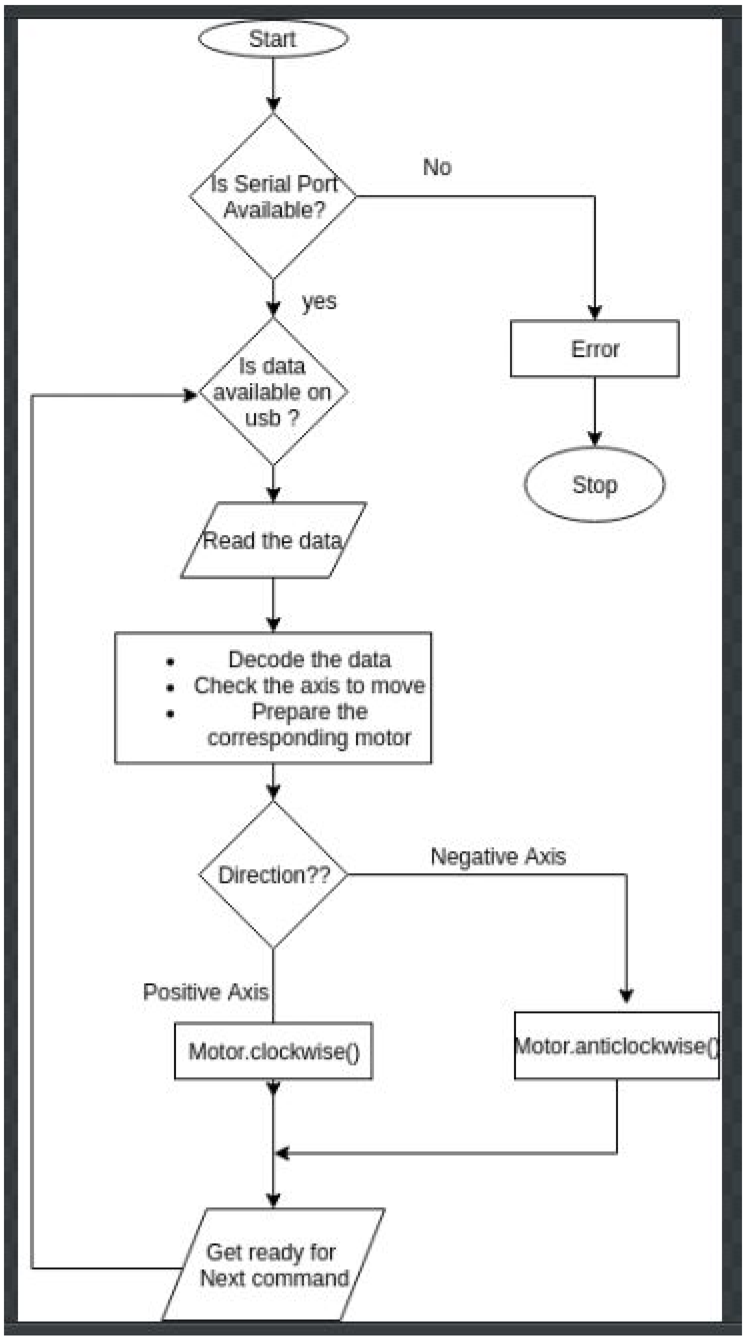


Fig 15 : Flow chart of arduino programming

Code:

```
#include <AccelStepper.h>
#include <Servo.h>                                          // Add library

Servo my_servo;

int servo_position = 0 ;

                                                           // Motor shield has two motor ports,
                                                           //  now we'll wrap them in
                                                           //  an AccelStepper object
AccelStepper stepper1(forwardstep1, backwardstep1);
AccelStepper stepper2(forwardstep2, backwardstep2);
int convertToInt( char a){
  switch (a){
    case '0': return 0;
          break;
    case '1': return 1;
          break;
    case '2': return 2;
          break;
    case '3': return 3;
          break;
    case '4': return 4;
          break;
    case '5': return 5;
          break;
    case '6': return 6;
          break;
    case '7': return 7;
              break;
    case '8': return 8;
              break;
    case '9': return 9;
              break;

  }

}

void setup()
{   Serial.begin(9600);                                        // setting the baud rate
    Serial.println("hello");

    my_servo.attach (10);

    stepper1.setMaxSpeed(300.0);
    stepper1.setAcceleration(100.0);
    stepper1.moveTo(0);
    stepper2.setMaxSpeed(300.0);
    stepper2.setAcceleration(100.0);
    stepper2.moveTo(0);

  }

void loop() {

    if (Serial.available() > 0) {                    // checking for serial communication
```

```
      String serialListener=Serial.readString();            // read the command from the software
      int current_position= my_servo.read();               // read the current position of z axis


                                                           //up or down servo motor
      if(serialListener[0] == 'U'){                        // pen is lifted
       my_servo.attach (10);
              for (servo_position =current_position; servo_position <=current_position + 90;
servo_position +=1){

           my_servo.write(servo_position);
           delay(10);
          }
         my_servo.detach();

        }
       else if (serialListener[0] == 'D'){                  // pen is dropped
         my_servo.attach (10);

               for (servo_position=current_position; servo_position >= current_position - 90;
servo_position -=1){

            my_servo.write(servo_position);
            delay(10);

          }
          my_servo.detach();
        }
```

```
if (serialListener[0] == 'Y') {                          // check the axis to move
 if (serialListener[1] == 'N') {                         // check the direction of movement
                                                          //along  the axis


    int a=convertToInt(serialListener[2]) ;              // convert the number of steps
                                                          // from the command into integer
  a=10*a;
  Serial.print(" a is ");
  Serial.print(a);
  int b=convertToInt(serialListener[3]);
  Serial.print(" b is ");
  Serial.print(b);
  int increment = 12*(a+b);                                         // multiply with minimum
                                                                          //resolution factor
  stepper1.moveTo(stepper1.currentPosition()-increment);                  // move the axis

 }
 else if (serialListener[1] == 'P') {
  int a=convertToInt(serialListener[2]);
  a=10*a;
  Serial.print(" a is ");
  Serial.print(a);
  int b=convertToInt(serialListener[3]);
  Serial.print(" b is ");
  Serial.print(b);
  int increment = 12*(a+b);
  stepper1.moveTo(stepper1.currentPosition()+increment);
 }
}
else if (serialListener[0] == 'X') {
  if (serialListener[1] == 'P') {
  int a=convertToInt(serialListener[2]);
  a=10*a;
  Serial.print(" a is ");
  Serial.print(a);
  int b=convertToInt(serialListener[3]);
  Serial.print(" b is ");
  Serial.print(b);
  int increment = 12*(a+b);
  stepper2.moveTo(stepper2.currentPosition()-increment);


  }
  else if (serialListener[1] == 'N') {
  int a=convertToInt(serialListener[2]);
  a=10*a;
  Serial.print(" a is ");
  Serial.print(a);
  int b=convertToInt(serialListener[3]);
  Serial.print(" b is ");
  Serial.print(b);
  int increment = 12*(a+b);
  stepper2.moveTo(stepper2.currentPosition()+increment);



  }
```

```
    }
                                                          // for diagonal Movement
    else if (serialListener[0] == 'T') {
        if (serialListener[1] == 'X' && serialListener[2] == 'N' && serialListener[3] == 'Y'
&& serialListener[4] == 'N'){
       int a=convertToInt(serialListener[5]);
       a=10*a;
       Serial.print(" a is ");
       Serial.print(a);
       int b=convertToInt(serialListener[6]);
       Serial.print(" b is ");
       Serial.print(b);
       int increment = 12*(a+b);
       stepper1.moveTo(stepper1.currentPosition()- increment);
       stepper2.moveTo(stepper2.currentPosition()+increment);
      }
       else if  (serialListener[1] == 'X' && serialListener[2] == 'N' && serialListener[3] ==
'Y' && serialListener[4] == 'P'){
       int a=convertToInt(serialListener[5]);
       a=10*a;
       Serial.print(" a is ");
       Serial.print(a);
       int b=convertToInt(serialListener[6]);
       Serial.print(" b is ");
       Serial.print(b);
       int increment = 12*(a+b);
       stepper1.moveTo(stepper1.currentPosition()+ increment);
       stepper2.moveTo(stepper2.currentPosition()+increment);
      }
```

```
        else if  (serialListener[1] == 'X' && serialListener[2] == 'P' && serialListener[3] ==
'Y' && serialListener[4] == 'N'){
        int a=convertToInt(serialListener[5]);
        a=10*a;
        Serial.print(" a is ");
        Serial.print(a);
        int b=convertToInt(serialListener[6]);
        Serial.print(" b is ");
        Serial.print(b);
        int increment = 12*(a+b);
        stepper1.moveTo(stepper1.currentPosition()- increment);
        stepper2.moveTo(stepper2.currentPosition()-increment);
       }
        else if  (serialListener[1] == 'X' && serialListener[2] == 'P' && serialListener[3] ==
'Y' && serialListener[4] == 'P'){
        int a=convertToInt(serialListener[5]);
        a=10*a;
        Serial.print(" a is ");
        Serial.print(a);
        int b=convertToInt(serialListener[6]);
        Serial.print(" b is ");
        Serial.print(b);
        int increment = 12*(a+b);
        stepper1.moveTo(stepper1.currentPosition()+ increment);
        stepper2.moveTo(stepper2.currentPosition()-increment);
       }
      }


   }
```

```
    stepper2.run();
    stepper1.run();
}
```

3.2 Python Programming:

3.2.1 Flow-chart:

The folow chart for the python based software is given below:

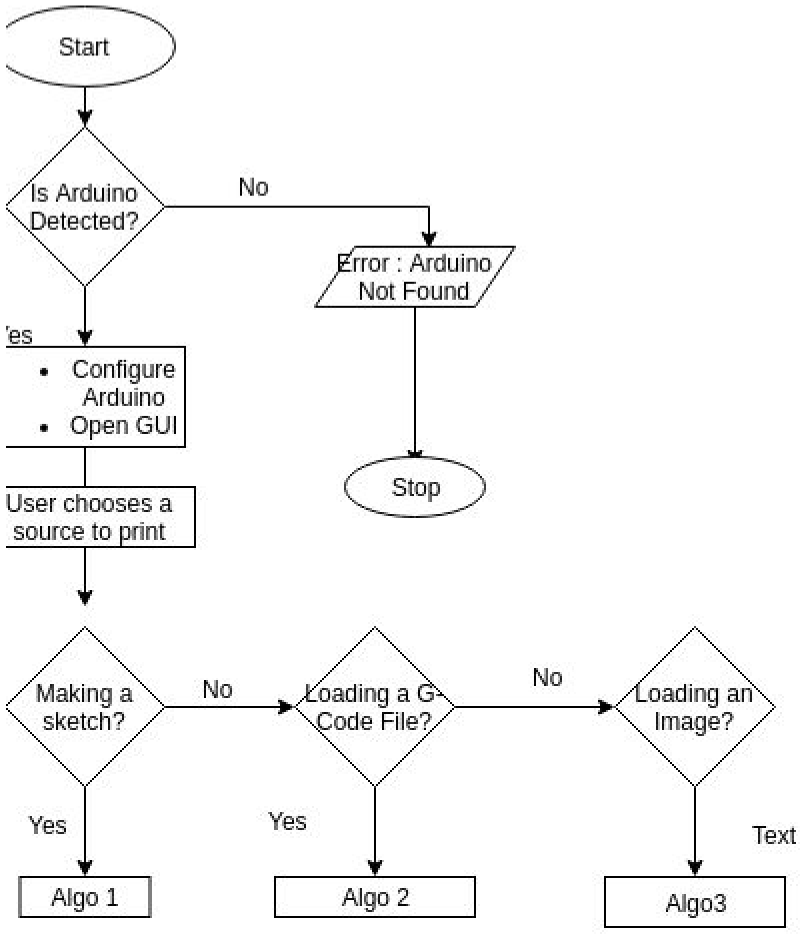
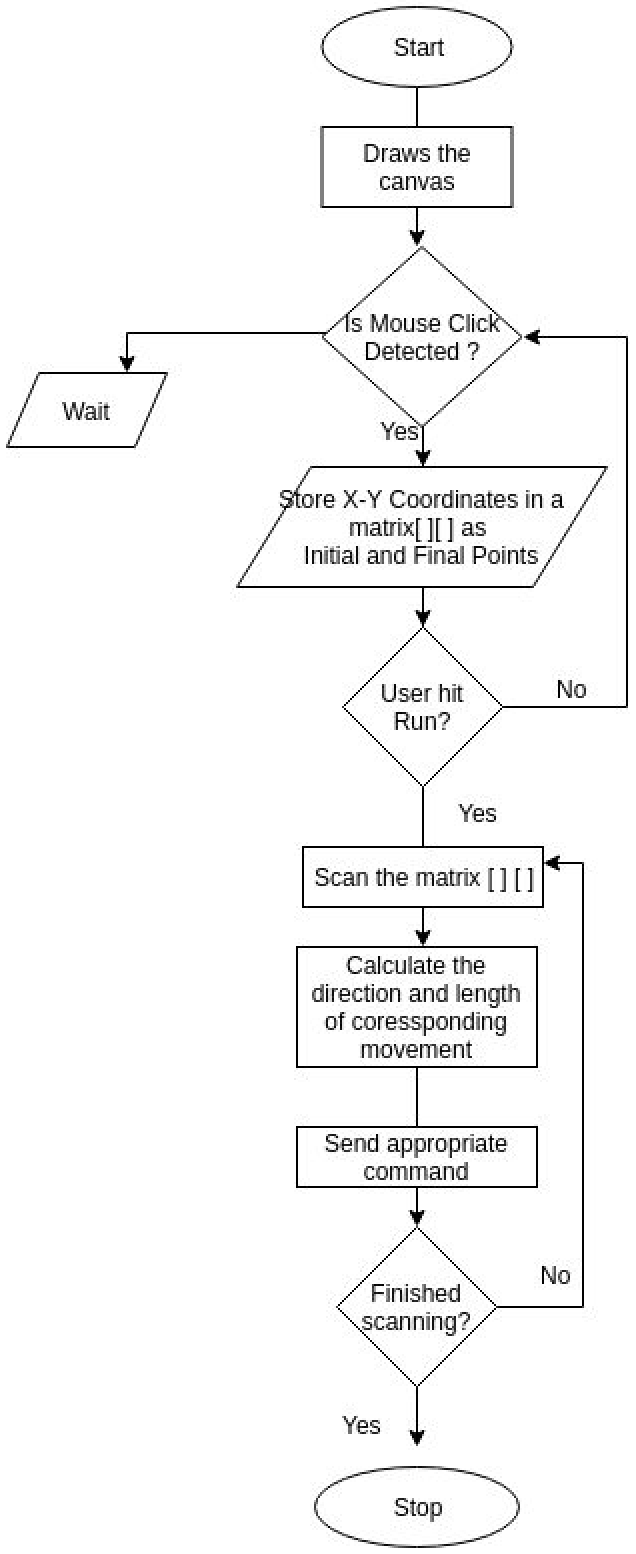


Fig 16 (a),(b),(c) : Flow chart of Python Code

Algo 1


Start
Draws the canvas
Is Mouse Click Detected ?
Wait
Yes
Store X-Y Coordinates in a matrix[ ][ ] as Initial and Final Points
User hit Run?
No
Yes
Scan the matrix [ ] [ ]
Calculate the direction and length of coressponding movement
Send appropriate command
Finished scanning?
No
Yes
Stop

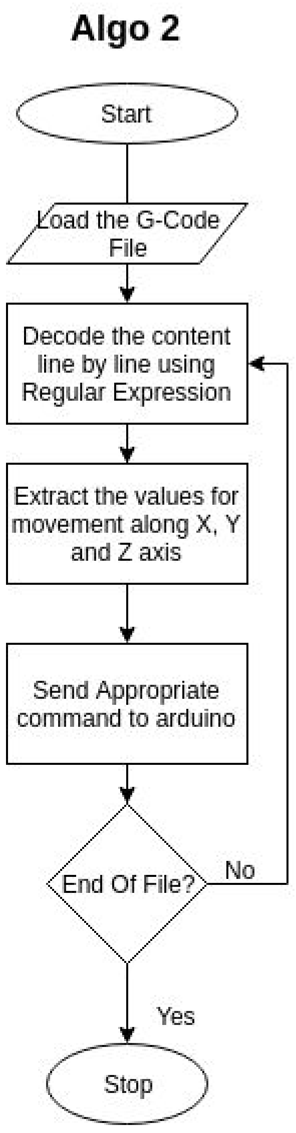

Algo 2
Start
Load the G-Code File
Decode the content line by line using Regular Expression
Extract the values for movement along X, Y and Z axis
Send Appropriate command to arduino
End Of File?
No
Yes
Stop

Fig 16(b)

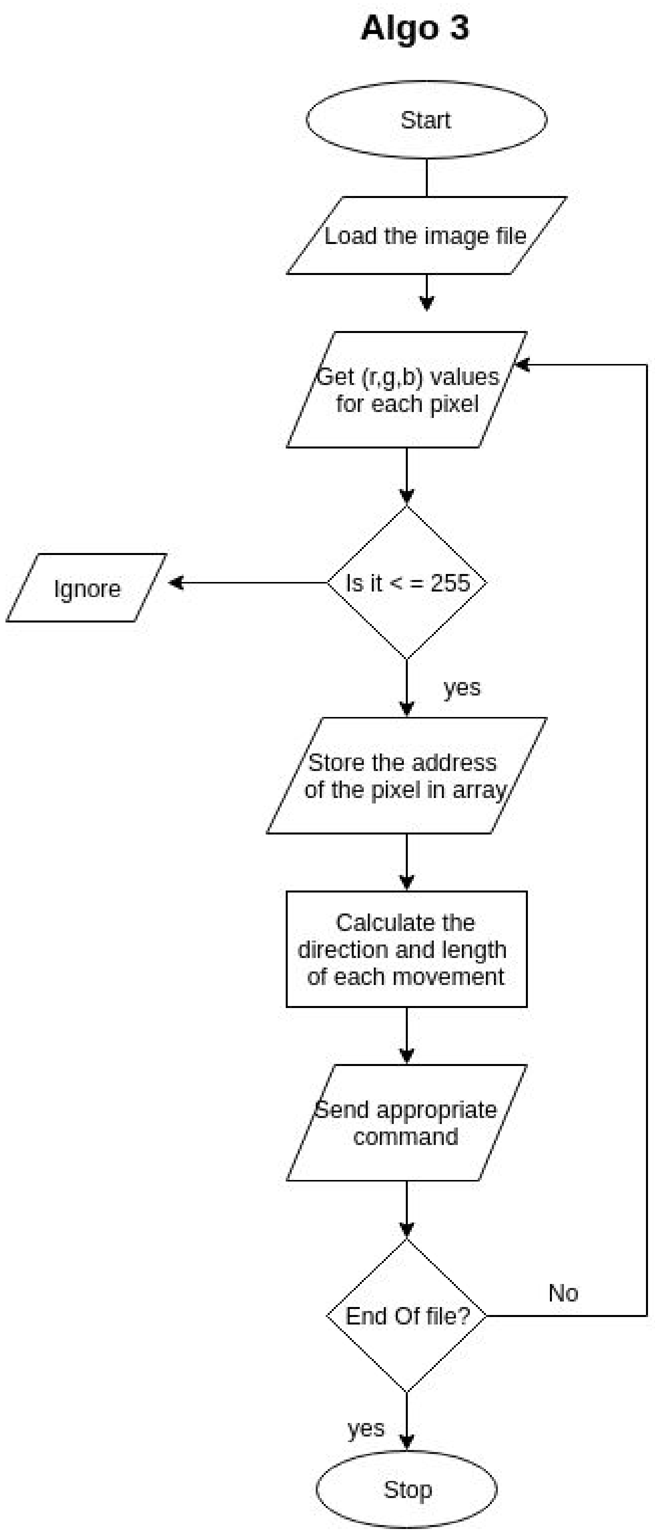

Algo 3
Start
Load the image file
Get (r,g,b) values for each pixel
Is it < = 255
Ignore
yes
Store the address of the pixel in array
Calculate the direction and length of each movement
Send appropriate command
End Of file?
No
yes
Stop

Fig 16(c)

Code:

```
from tkinter import *                                   // GUI framework for python
import re                                              // for using regular expressions
import os                                              // for configuring the hardware
import warnings                                        // for displaying warnins
import serial                                          // for setting up serial commuinication
import serial.tools.list_ports                         // for differentiating between the ports
import time                                            // for using delay
import getpass                                         // for password
from PIL import Image, ImageTk                         // for image processing

arduino_ports = [                                      // check the ports available
   p.device
   for p in serial.tools.list_ports.comports()
   if re.search(r'ttyACM.','ttyACM0').group() or re.search(r'ttyS.','ttyS0').group()
]                                                       // identify the ports used by arduino
if not arduino_ports:
   raise IOError("No Arduino found")
   print(" ")
if len(arduino_ports) > 1:
   warnings.warn('Multiple Arduinos found - using the first')

print("Please Enter Your Computer's Password (it is neccessary for giving permission to the
board):")

sudoPassword = "      "

command = 'sudo chmod 666  '+arduino_ports[0]                   // configuring arduino
os.system('echo %s|sudo -S %s' % (sudoPassword, command))
```

```
arduino = serial.Serial(arduino_ports[0],9600)                    // setting baud rate for serial
                                                                    //communication
print(arduino)
servo_position = 0
def isPendown(servo_position):                                      // check if pen is down
    if(servo_position == 0):
        return True
    else:
        return False



def isPenup(servo_position):                                        // check if pen is up
    if(servo_position == 90):
        return True
    else:
        return False
// function to lower the pen
def pen_down():
    global servo_position
    time.sleep(1)
    go="D"                                                          // prepare command for the
                                                                    // harware
    arduino.write(go.encode())                                      // send  command to arduino
    time.sleep(2)
    servo_position=0
    print("pen down")

// function to lift the pen
def pen_up():
    global servo_position
    time.sleep(1)
    go="U"
    arduino.write(go.encode())
    time.sleep(2)
    servo_position=90
    print("pen up")



                                    // function for fixing the resolution of initial and final point
def fixRes(x,a):
    for n in range(len(a)):
        if(x> a[0] and x < a[len(a)-1]):
            if (x>a[n] and x<a[n+1]):
                if (x <= (a[n] + a[n+1])/2):
                    x=a[n]
                else:
                    x=a[n+1]


        else:
            if(x<a[0]):
                x=a[0]
            elif(x>a[len(a)-1]):
                x=a[len(a)-1]

    return x



                                                        // function to convert G code length
```

```
                                                    //component into integer
def toInt(x):


   if(float(x) < 1 or float(x)< (-1)):
      if float(x) > 0 :
         x=int(1)
      elif float(x) < 0:
         x= int(-1)
      else:
         x= int(0)



   else :
      for i in range(-50,50):
         if (float(x)>i and float(x)<i+1):
            x = int(i)
         elif(float(x) == float(i)):
            x= int(i)
   return x



                                             // functions for movement of axes
def movement(xi,yi,xf,yf):
   if(xi == xf and yi != yf):
       moveY(yi,yf)
   elif(yi == yf and xi != xf):
       moveX(xi,xf)
   else:
       move_diagonal(xi,yi,xf,yf)
```

```
    return



def moveY(yi,yf):
    if (yi>yf):
        moveUp(yi - yf)
    else:
        moveDown(yf - yi)

def moveX(xi,xf):
    if (xi>xf):
        moveLeft(xi - xf)
    else:
        moveRight(xf - xi)



def move_diagonal(xi,yi,xf,yf):
    if (xi>xf and yi>yf):
        #move_diagonal_lift_up(xi-xf, yi-yf)
        time.sleep(1)
        steps=xi-xf
        if((steps/20)<10):
            go="DXNYP"+"0"+str(int(steps/20))
        else:
            go="DXNYP"+str(int(steps/20))

        print(go)
        arduino.write(go.encode())
        time.sleep(5)

    elif(xi<xf and yi > yf):
        #move_diagonal_right_up(xf-xi, yi-yf)
        time.sleep(1)
        arduino.write('B'.encode())
        time.sleep(60)

    elif(xi > xf and yi < yf):
        #move_diagonal_down_left(xi-xf, yf-yi)
        time.sleep(1)
        arduino.write('C'.encode())
        time.sleep(60)
    elif(xi<xf and yi < yf):
        #move_diagonal_down_right(xf-xi, yf-yi)
        time.sleep(1)
        arduino.write('D'.encode())
        time.sleep(60)




def moveUp(steps):
    time.sleep(1)

    if((steps/20)<10):
        go="YP"+"0"+str(int(steps/20))
    else:
        go="YP"+str(int(steps/20))

    print(go)
    arduino.write(go.encode())
    print("up here")
    time.sleep(5)








def moveDown(steps):
    time.sleep(1)

    if((steps/20)<10):
        go="YN"+"0"+str(int(steps/20))
    else:
        go="YN"+str(int(steps/20))

    print(go)
    arduino.write(go.encode())
    print("down here")
    time.sleep(5)
```

```
def moveRight(steps):
    time.sleep(1)

    if((steps/20)<10):
        go="XP"+"0"+str(int(steps/20))
    else:
        go="XP"+str(int(steps/20))

    print(go)
    arduino.write(go.encode())
    print("right here")
    time.sleep(5)

def moveLeft(steps):
    time.sleep(1)
    if((steps/20)<10):
        go="XN"+"0"+str(int(steps/20))
    else:
        go="XN"+str(int(steps/20))
    arduino.write(go.encode())
    print("left here")
    time.sleep(5)



// programming for GUI components
```

```
# Here, we are creating our class, Window, and inheriting from the Frame
# class. Frame is a class from the tkinter module. (see Lib/tkinter/__init__)
class Window(Frame):
    disp_count=0                                                    # number of displacements
    i=0
    j=0
    drawing_tool = "line"                                     # Stores current drawing tool used
     left_but = "up"                                               # Tracks whether left mouse is
down
    x_pos, y_pos = None, None                             # x and y positions for drawing
                                                                      # with pencil
    x_points=[]
    y_points=[]
                                                                      # Tracks x & y when the mouse is
                                                                      # clicked and released
    x1_line_pt, y1_line_pt, x2_line_pt, y2_line_pt = None, None, None, None
    coordinates=[[None for _ in range(4)] for _ in range(50)]

                                                                      # Define settings upon initialization.
    def __init__(self, master=None):

        # parameters that you want to send through the Frame class.
        Frame.__init__(self, master)

                                                                     # reference to the master widget,
                                                                     # which is the tk window
         self.master = master
        self.init_window()
        self.g_code= [ [None for _ in range(5)]for _ in range(100)]

    def init_window(self):                                          #Creation of init_window



        self.master.title("Tarun2.0 - The PCB Designer")          # changing the title of
                                                                   # our master widget

        # allowing the widget to take the full space of the root window

        self.pack()



        menu = Menu(self.master)                                   # creating a menu instance
        self.master.config(menu=menu)



        file = Menu(menu)                                          # create the file object)
        file.add_command(label="Open", command=self.__openFile)
        file.add_command(label="New", command=self.__newFile)

        # adds a command to the menu option, calling it exit, and the
        # command it runs on event is client_exit

        file.add_command(label="Exit", command=self.client_exit)
```

```
        menu.add_cascade(label="File", menu=file)                # added "file" to our menu
        edit = Menu(menu)                                        # create the file object)

        # adds a command to the menu option, calling it exit, and the
        # command it runs on event is client_exit

        edit.add_command(label="Undo")

        #added "file" to our menu
        menu.add_cascade(label="Edit", menu=edit)

        about = Menu(menu)
        about.add_command(label="Application", command=self.__showAbout)
        menu.add_cascade(label="About", menu=about)

        self.box=Frame(root)
        self.box.pack()

        self.box1=Frame(root)
        self.box1.pack()

          increase_Z = Button(self.box1,text='Z +ve',command=self.__increase_Z).pack(side =
"top")
          decrease_Z = Button(self.box1,text='Z -ve',command=self.__decrease_Z).pack(side =
"bottom")

         increase_X = Button(self.box1,text='X +ve',command=self.__increase_X).pack(side =
"left")
```

```
        decrease_X = Button(self.box1,text='X -ve',command=self.__decrease_X).pack(side =
"left")

        increase_Y = Button(self.box1,text='Y +ve',command=self.__increase_Y).pack(side =
"right")
        decrease_Y = Button(self.box1,text='Y -ve',command=self.__decrease_Y).pack(side =
"right")


      #print(widget)

      self.box2=Frame(root)
      self.box2.pack()

      self.drawing_frame=Frame(self.box2)
      self.drawing_frame.pack()
      drawing_area = Canvas(self.drawing_frame,bg="white",width=360,height=360)
      drawing_area.pack()
      drawing_area.bind("<Motion>", self.motion)
      drawing_area.bind("<ButtonPress-1>", self.left_but_down)
      drawing_area.bind("<ButtonRelease-1>", self.left_but_up)



                                                        // Puts dots on canvas

      for x in range(10,360,20):
         for y in range (10,360,20):
            drawing_area.create_rectangle(x-1, y-1, x + 1, y + 1,fill="midnight blue")
            self.x_points.append(x)
            self.y_points.append(y)

      self.btn2 = Button(self.drawing_frame,text='Run', command=self.run_grid).pack()



                                                              // function for opening the G code File


    def __openFile(self):
      root=self.master
      self.drawing_frame.pack_forget()

      filename = filedialog.askopenfilename()
      i=0
      f_extns = filename.split(".")
      print(f_extns)
      if(f_extns[-1] == "png" or f_extns[-1] == "jpg" or f_extns[-1] == "bmp"):




         frame = Frame(self.box2)
         frame.pack()
         load = Image.open(filename)
                                                                     #resizing height and width of image
         new_width  = 360
         new_height = 360
         load = load.resize((new_width, new_height), Image.ANTIALIAS)
         render = ImageTk.PhotoImage(load)
         label = Label(frame, image=render)
         label.image = render
         label.pack()
         #load.place(x=0, y=0)


         im = Image.open(filename, "r")


         print("it works")



         pix_val = list(im.getdata())
         n=0
         pix_val_flat = [x for sets in pix_val for x in sets]
         for i in range(0,len(pix_val_flat),10):
            if(pix_val_flat[i] < 10):
                if(n == 0):
                    self.coordinates[0][0]= i/1000
                else:
                    self.coordinates[0][1] = i/1000
                n=n+1

         btn_image = Button(frame,text='Run', command=self.__run_image).pack()


      else:
         T = Text(self.box2, height=30, width=50)
         T.pack()
```

```
with open(filename, 'r') as f_gcode:
   for line in f_gcode:


      try:
         str(line)


         s=line.split(" ")
         for item in range(len(s)):


                                                                 r_string_and_num_unsigned   =
re.compile("([a-zA-Z]+)([0.0-9.9]+)").match(s[item])
                                                                    r_string_and_num_signed   =
re.compile("([a-zA-Z]+)([-+]+)([0.0-9.9]+)").match(s[item])
            if r_string_and_num_unsigned:
              T.insert(INSERT,r_string_and_num_unsigned.groups())
               T.insert(INSERT," ")
               self.g_code[i][item]=r_string_and_num_unsigned.groups()
            elif r_string_and_num_signed:
               T.insert(INSERT,r_string_and_num_signed.groups())
               T.insert(INSERT," ")
               self.g_code[i][item]=r_string_and_num_signed.groups()
      except:
         pass


      i=i+1
      T.insert(INSERT, "\n")


btn_gcode = Button(root,text='Run', command=self.__run_gcode).pack()
```

```
 // Function to decode the G Code file
def __run_gcode(self):
   j=0
   command=list()
   for row in range(50):
      for col in range(4):
         if (self.g_code[row][col] != None):
                     if(self.g_code[row][col][0] == 'X' or self.g_code[row][col][0] == 'Y' or
self.g_code[row][col][0] == 'Z' ):

                #print(self.g_code[row][col][0],end=" ")
                command.append(self.g_code[row][col][0])
                if(self.g_code[row][col][1] != '-'):

                   command.append(toInt(self.g_code[row][col][1]))
                else:

                   command.append(-1*toInt(self.g_code[row][col][2]))

   for i in range(len(command)):
      if command[i] == 'X':
         if command[i+1] !=0:
            if command[i+3] != 0:
               move_diagonal(command[i+1],command[i+3])
            else:
               moveX(command[i+1])
         elif command[i+1] == 0:
            if command[i+3] != 0:
               moveY(command[i+3])
```

```
  def __newFile(self):
      self.master.title("Untitled - Tarun2.0")
      self.__file = None
      self.box.pack_forget()
      self.init_window()

 def __increase_X(self):
      go="XP"+"0"+str(int(5))
      arduino.write(go.encode())
      time.sleep(1)

  def __decrease_X(self):
      go="XN"+"0"+str(int(5))
      arduino.write(go.encode())
      time.sleep(1)

  def __increase_Y(self):
      go="YP"+"0"+str(int(5))
      arduino.write(go.encode())
      time.sleep(1)

  def __decrease_Y(self):
      go="YN"+"0"+str(int(5))
      arduino.write(go.encode())
```

```
    time.sleep(1)

def __increase_Z(self):
    arduino.write("u".encode())
    time.sleep(1)

def __decrease_Z(self):
    arduino.write("d".encode())
    time.sleep(1)

def __showAbout(self):
    messagebox.showinfo("Tarun2.0","Tarun Thakur")

# ---------- CATCH MOUSE UP ----------
def left_but_down(self, event=None):
    self.left_but = "down"

    # Set x & y when mouse is clicked
    self.x1_line_pt = event.x
    self.y1_line_pt = event.y

    self.x1_line_pt =fixRes(self.x1_line_pt,self.x_points)
    self.y1_line_pt =fixRes(self.y1_line_pt,self.y_points)

    self.coordinates[self.i][0]= self.x1_line_pt
    self.coordinates[self.i][1]= self.y1_line_pt

    print("x1 = " + str(self.x1_line_pt)+"\n y1 = "+str(self.y1_line_pt))
```

```
# ---------- CATCH MOUSE UP ----------

def left_but_up(self, event=None):
    self.left_but = "up"

    # Reset the line
    self.x_pos = None
    self.y_pos = None

    # Set x & y when mouse is released
    self.x2_line_pt = event.x
    self.y2_line_pt = event.y

    self.x2_line_pt =fixRes(self.x2_line_pt,self.x_points)
    self.y2_line_pt =fixRes(self.y2_line_pt,self.y_points)
    self.disp_count= self.disp_count +1

    self.coordinates[self.i][2]= self.x2_line_pt
    self.coordinates[self.i][3]= self.y2_line_pt
    self.i = self.i + 1
    print("x2 = " + str(self.x2_line_pt)+"\n y2 = "+str(self.y2_line_pt))

                                                  # If mouse is released and line tool is selected
                                                  # draw the line
    if self.drawing_tool == "line":
        self.line_draw(event)



# ---------- CATCH MOUSE MOVEMENT ----------
```

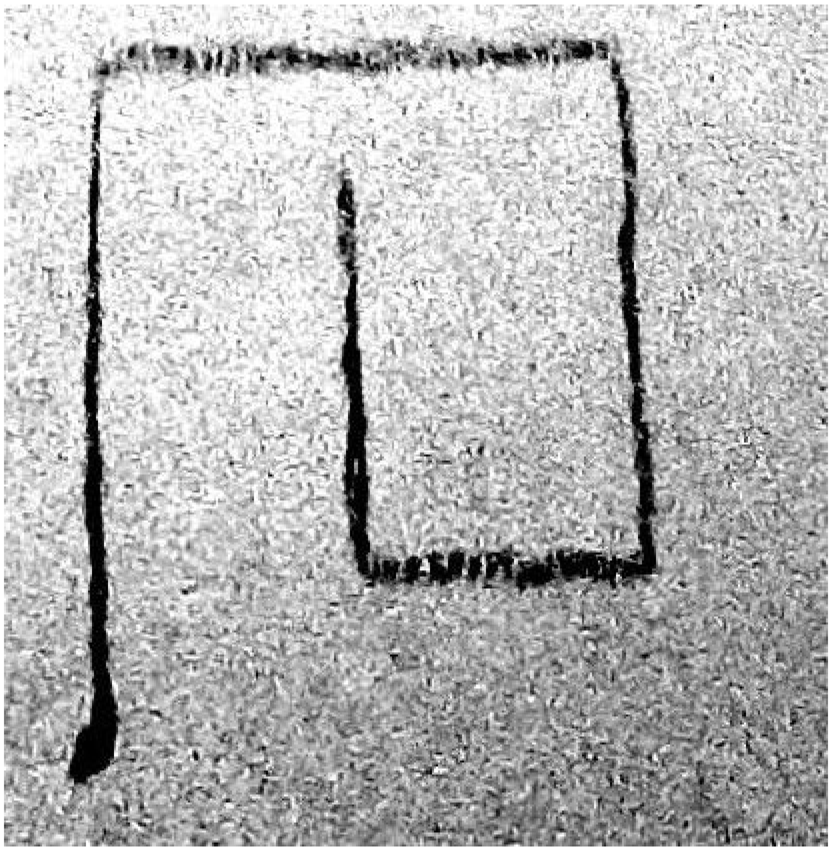
```
    def motion(self, event=None):

        if self.drawing_tool == "pencil":
            self.pencil_draw(event)



  # ---------- DRAW LINE ----------

    def line_draw(self, event=None):

        # Shortcut way to check if none of these values contain None
        if None not in (self.x1_line_pt, self.y1_line_pt, self.x2_line_pt, self.y2_line_pt):
                        event.widget.create_line(self.x1_line_pt,  self.y1_line_pt,  self.x2_line_pt,
self.y2_line_pt, smooth=TRUE, fill="green")





    def run_grid(self):
        del self.coordinates[self.disp_count : ]
        print(str(len(self.coordinates)))
        for k in range(-1,len(self.coordinates)):
            print(str(k))


            if k == -1:
                print("")
        elif k == 0:
            print("moving")
            #if ( not isPendown(servo_position)):
            #pen_down()# under if

movement(self.coordinates[k][0],self.coordinates[k][1],self.coordinates[k][2],self.coordinate
s[k][3])
                print("X1 = " + str(self.coordinates[k][0])+"Y1 = " + str(self.coordinates[k][1]) +
"X2 = " + str(self.coordinates[k][2]) + "Y2 = " + str(self.coordinates[k][3]))

        else:
                if self.coordinates[k][0] == self.coordinates[k-1][2] and self.coordinates[k][1] ==
self.coordinates[k-1][3] :
                print("moving")
              #  if ( not isPendown(servo_position)):
              #  pen_down()# under if
                print("servo position" + str(servo_position))



movement(self.coordinates[k][0],self.coordinates[k][1],self.coordinates[k][2],self.coordinate
s[k][3])


                   print("X1 = " + str(self.coordinates[k][0])+"Y1 = " + str(self.coordinates[k][1])
+ "X2 = " + str(self.coordinates[k][2]) + "Y2 = " + str(self.coordinates[k][3]))

            else:
                # if ( not isPenup(servo_position)):
                pen_up()#under if
                #print("servo position" + str(servo_position))
                print("problem is here \n")

movement(self.coordinates[k-1][2],self.coordinates[k-1][3],self.coordinates[k][0],self.coordi
nates[k][1])
                print("k-1 = " +str(k-1)+"k"+str(k))
                                          print("X1  =  "  +  str(self.coordinates[k-1][2])+"Y1  =  "  +
str(self.coordinates[k-1][3])  +  "X2  =  "  +  str(self.coordinates[k][0])  +  "Y2  =  "  +
str(self.coordinates[k][1]))
                pen_down()
                print("servo position" + str(servo_position))

movement(self.coordinates[k][0],self.coordinates[k][1],self.coordinates[k][2],self.coordinate
s[k][3])
                  print("X1 = " + str(self.coordinates[k][0])+"Y1 = " + str(self.coordinates[k][1])
+ "X2 = " + str(self.coordinates[k][2]) + "Y2 = " + str(self.coordinates[k][3]))
                print("k= "+str(k))




    def client_exit(self):
        exit()
```

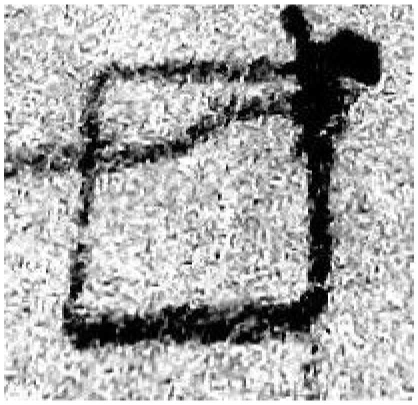
```
# root window created. Here, that would be the only window, but
# you can later have windows within windows.
root = Tk()

root.geometry("1200x700")

#creation of an instance
app = Window(root)

#mainloop
root.mainloop()
```

CHAPTER 4: RESULTS

The machine designed was successfully tested for different designs. The minimum precision that could be achieved by varying the multipication factor in the code was found to be 0.5 mm for M.F equal to 5.

Some of the designs printed are as follows:





Refrences:

1. Mike Lynch, "Key CNC Concept #1—The Fundamentals Of CNC", *Modern Machine Shop*, 4 January 1997.

2. http://www.technologystudent.com/cam/cncman2.htm

3. https://nptel.ac.in/courses/112103174/pdf/mod4.pdf

4. Noble, David F. (1984), *Forces of Production: A Social History of Industrial Automation*, New York, New York, USA

5. Reintjes, J. Francis (1991), *Numerical Control: Making a New Technology*, Oxford University Press

6. Siegel, Arnold. "Automatic Programming of Numerically Controlled Machine Tools", *Control Engineering*, Volume 3 Issue 10 (October 1956), pp. 65–70.

7. The Evolution of CNC Machines (2018). Retrieved October 15, 2018, from Engineering Technology Group

8. www.cnc.com/the-history-of-computer-numerical-control-cnc/

9. http://news.mit.edu/2008/obit-reintjes-tt0305

10. https://www.baronmachine.com/news/the-evolution-of-the-modern-cnc-machine-shop/

11. https://www.python.org/doc/essays/blurb/

12. https://www.pythonforbeginners.com/learn-python/what-is-python/

13. https://www.w3schools.com/python/python_intro.asp

14. https://www.geeksforgeeks.org/python-gui-tkinter/

15. https://wiki.python.org/moin/TkInter

16. https://docs.python.org/2/library/tkinter.html

17. https://www.arduino.cc

18. https://www.oreilly.com/library/view/arduino-cookbook/9781449399368/ch04.html