ARDUINO DRAWING MACHINE

1. INTRODUCTION

The aim of this project is to create an Arduino-based mini CNC plotter from scrap materials obtained from DVD drives. The functioning principle is the juxtaposition of two stepper motors and a servo motor in order to obtain movement on the three axes. The steppers represent the X (moving the drawing surface) and Y axis (moving across the drawing surface), while the Z-axis movement is ensured by the servo (which will raise or lower the pen on the drawing surface at particular time stamps).

Arduino is an open-source electronics platform based on easy-to-use hardware and software. The Arduino programming language, and the Arduino Software (IDE), based on Processing are used to control Arduino components such as microcontrollers, sensors and buttons. [1]

CNC, namely computer numerical control, is the automated control of machining tools (drills, boring tools, lathes) and 3D printers by means of a computer. A CNC machine processes a piece of material (metal, plastic, wood, ceramic, or composite) to meet specifications by following a coded programmed instruction and without a manual operator. Instructions are delivered to a CNC machine in the form of a sequential program of machine control instructions such as G-code and then executed. [2]

The plotter described in this documentation is such a CNC machine and uses transmits commands in the form of G-codes to the microcontroller. G-codes are usually used to command specific movements of the machine.

2. BIBLIOGRAPHIC STUDY

COMPONENTS REQUIRED:

- Arduino UNO
- L293D motor driver shield
- Micro Servo motor 9g SG90
- 2x stepper motors (from scrap DVD drivers)
- Dupont wires
- 5V 1A power supply

OPTIONAL COMPONENTS

- 8x M6 60 screws and nuts
- 3x M6 30 screws and nuts
- Soldering iron
- Drilling machine

- Glue
- Plexiglass
- 2x springs
- Rail system with support from (another) DVD drive

ARDUINO UNO

The Arduino UNO is a *microcontroller board* developed by Arduino.cc. It is open-source and based on the Microchip ATMega328P. The board is equipped with numerous digital and analog input/output pins that can have various shields (expansion boards) attached to it. It is programmable with the Arduino IDE via a USB cable and can be powered either through the USB cable or with external power sources. [5] The microcontroller board represents *the "brain" of the project* as it is programmed with the code used to interpret motor commands.

L293D MOTOR DRIVER SHIELD

The motor shield is an *expansion board* that can be attached to a microcontroller board. It can be connected to two servo motors and 2 stepper motors or 4 DC motors, having 2 sets of servo pins and 4 pairs of channels (M1, M2, M3, M4). The shield can be connected to a power supply of up to 16V. The chip is an *H-Bridge*, which is an electrical circuit that enables a voltage to be applied in either direction to an output (for example a motor). [6] The motor shield is *used to connect the 3 motors (servo and two steppers)* to the microcontroller. It was used due to its simplicity and efficiency (connecting all motors directly to a shield as opposed to connecting each motor separately).

MICRO SERVO MOTOR 9G SG90

The micro servo motor 9g SG90 is a lightweight servo with high output power. It can rotate *180 degrees* and can be used with any library. Its pins represent: PWM (orange), Vcc (red), ground (brown). It needs 4.8-6 V to function correctly. [7] The servo represents the *Z-axis movement*; it moves the pen up and down on the drawing surface with high precision. The servo control is done using the *Servo.h* library.

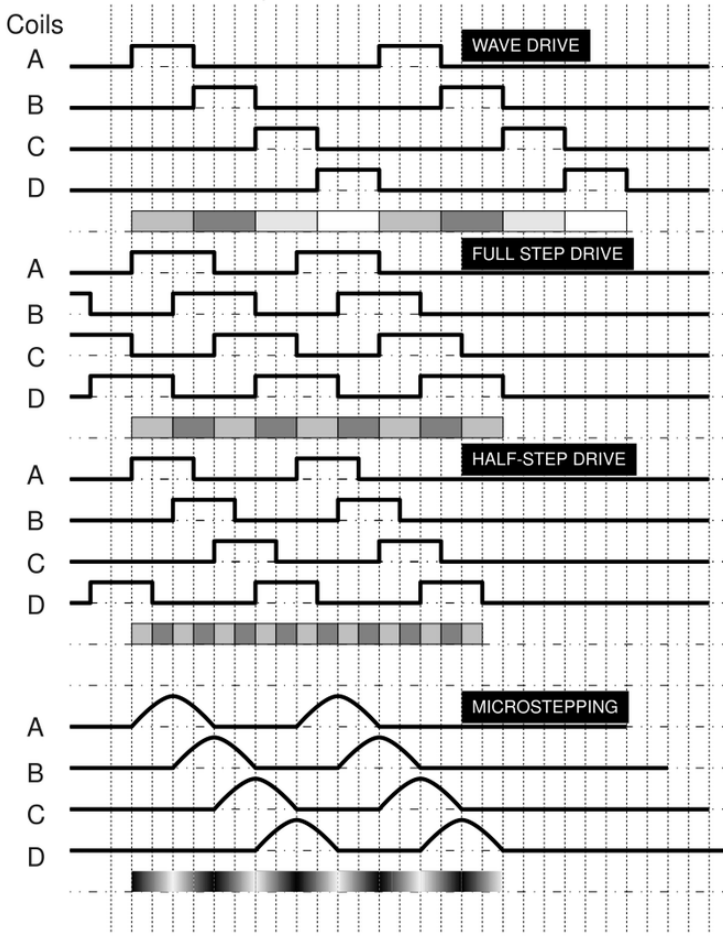
STEPPER MOTORS (FROM DVD DRIVERS)

A stepper motor is a brushless DC electric motor which divides a full rotation into a number of equal steps. It can be commanded to move to a certain position and hold its position without the need of sensors. While brushed DC motors rotate continuously, the stepper converts input pulses into precise increments of the position.

The motors found in DVD drives are bipolar and are generally used to move the laser across the DVD; bipolar motors can move forward or backward, according to the direction of the current. Typically, an H-Bridge is used to do this. The pattern of a two-coil bipolar stepper is: A+, B+, A-, B-; i.e. drive coil A with positive current, then drive coil B with positive current, drive coil A with negative current, then drive coil B with negative current and renew the cycle. The polarity of the current is flipped with the H-Bridge. In this project, DVD drive motors are used to provide the basis for the X

and Y-axis movement. Movement is precise and restricted to up to 4cm for each motor, thus offering a final 4x4 cm surface for the plotter. The X-axis has a drawing surface attached to it (enabling the surface to move forward and backward) and the Y-axis has the Z-axis assembly attached, ensuring the left and right movement.

The stepper is a polyphase motor and is ideally driven by a sinusoidal current. Various drive techniques have been developed to approximate a sinusoidal waveform. The microstep is the technique used in this project as it is the one closest to the ideal waveform; thus, producing the least vibration, leading to higher accuracy in the final product (the drawings done by the plotter). [8]



SOFTWARE REQUIRED:

- Arduino IDE (download from: https://www.arduino.cc/en/main/software)
- Processing IDE (download from: https://processing.org/download/)
- Inkscape version 0.48.5 (download from: https://inkscape.org/release/inkscape-0.48/?latest=1)
- AFMotor library (download from: https://learn.adafruit.com/adafruit-motor-shield/library-install)
- Makerboat G-code Inkscape extension (download from: https://github.com/martymcguire/inkscape-unicorn)

ARDUINO IDE AND AFMOTOR LIBRARY

Arduino IDE is an open-source software that can be used to write code and upload it to Arduino boards. In this project, the IDE is used to program the board with the code that interprets G-codes (namely, machine commands) received on the serial interface and accordingly send commands to the motors. The Adafruit motor shield library is used to control the motor shield and the motors. It must be installed to the Arduino IDE as such: in the Arduino IDE -> Sketch -> Include Library -> Manage libraries -> search for the AFMotor library and select it.

PROCESSING IDE

Processing IDE is a software used to write Processing programs. It includes a Text Editor, a compiler and a display window. The programs created with the IDE are called sketches and can draw two and three-dimensional graphics. In this project, Processing IDE was used to facilitate the communication between the microcontroller and the PC. The code (Gctrl) is used to connect to the port of the microcontroller, to test the axes movement and pressure (ensure correct position of the drawing surface and pen), to set the speed of the motors and finally to send a G-code to the programmed board through the serial interface in order to be interpreted and executed.

INKSCAPE AND MAKERBOAT G-CODE EXTENSION

Inkscape is a professional, open-source vector graphics editor. It is used to create the texts and images for the plotter. The plotter can interpret G-code which is a code that contains machine commands. Thus, a g-code extension should be added to the Inkscape editor in order to save texts and images in the g-code format. The Makerboat G-code extension allows the user to save Inkscape drawings as G-codes. It should be downloaded, then installed as such: copy the contents of *src/* to the extensions/ folder of the Inkscape application. An example location of the extensions folder is: *C:\Program files\Inkscape\share\extensions*.

3. ANALYSIS

The project is meant to be able to draw any G-code file on an area of at most 4x4 cm. The movement on the three axes is ensured by two stepper motors and a servo.

One stepper, placed horizontally and affixed to a plane (scrap DVD drives), will represent the X-axis and will move a drawing surface forward and backward up to 4 cm.

Placed at a 90-degree angle, the other stepper, representing the Y-axis and affixed to another plane, will move a writing utensil across the drawing surface, ensuring a left and right movement spanning 4 cm.

The Z-axis movement will be done by an assembly composed of a pen (or any writing utensil), a servo and a contraption actioned by the servo to move the pen up

and down vertically by a few millimetres such that the return to a fixed initial position (pen presses on drawing surface) is ensured.

The wiring will be described in the design chapter of the documentation. All 3 motors will be connected to a motor shield and an external 5V 1A power source will be supplied.

Using Inkscape, texts or images will be transformed into vectors and saved as G-code formats.

The Gctrl Processing program will be used to connect the PC to the port of the Arduino UNO. An initial test of the axes movements is recommended to check that the position (height) of the drawing surface is optimal across all corners and the pressure of the pen is not too high when touching the drawing surface. Afterwards, the G-code files can be streamed into the serial interface of the microcontroller.

The Arduino UNO will be programmed to interpret and execute G-codes, namely, machine commands. It will receive the G-code file from the Gctrl processing program one character at a time, reconstruct each line of the code and, according to the command lines, move the corresponding motors to new positions.

4. Design

Hardware

a) Disassemble two DVD drives in order to obtain the stepper motors and their frames (fig. 1).

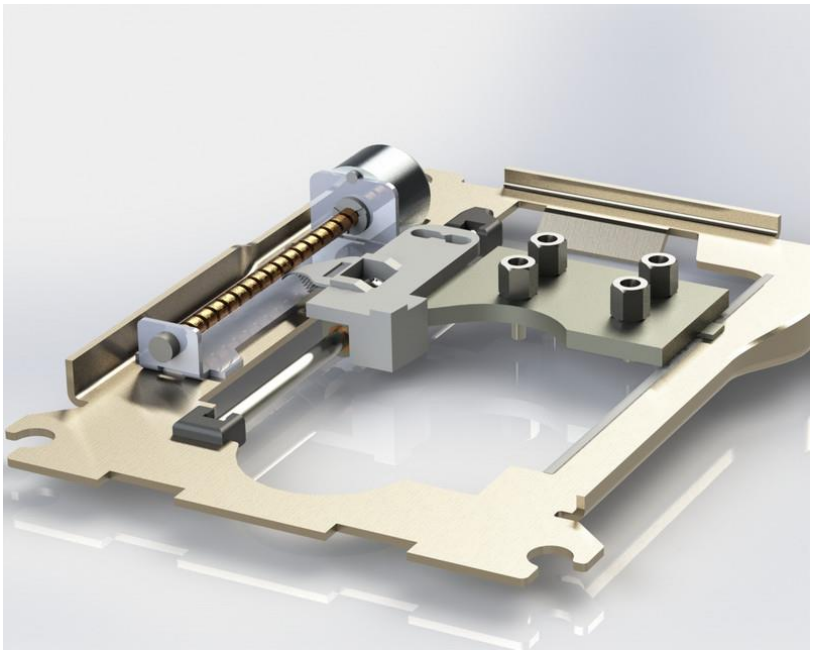


FIGURE 1

Check the four pins of each motor to see which of them constitute pairs (A+ A- and B+ B-) as in fig. 2 and fig. 3. This can be done either using a multimeter or using

the method illustrated in [9]: hold the two wires of a diode (LED) against two of the pins of a motor and move the support located between the rails of the motor frame. If the LED illuminates, the two pins constitute a pair. If the LED does not light up, two other pins should be verified.

FIGURE 2

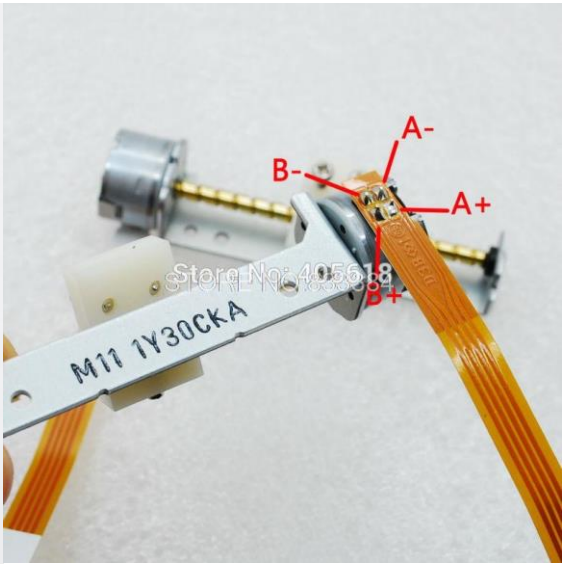


FIGURE 3

Solder wires to the motor pins as exemplified in fig. 4 and fig. 5 so that the motors can be connected to the shield.

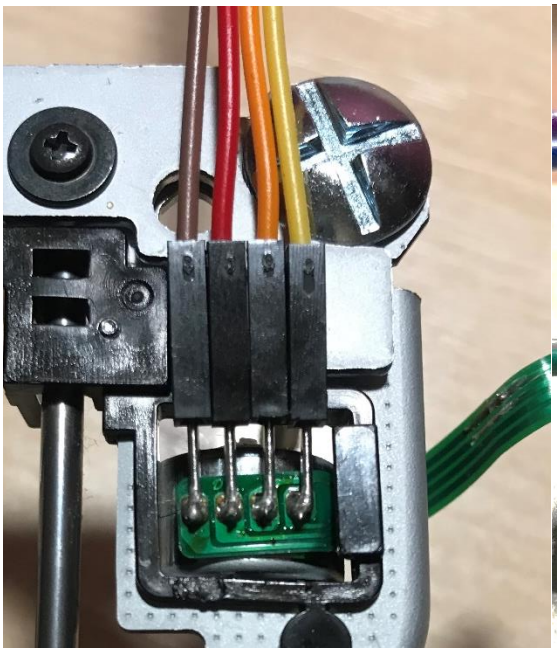


FIGURE 4

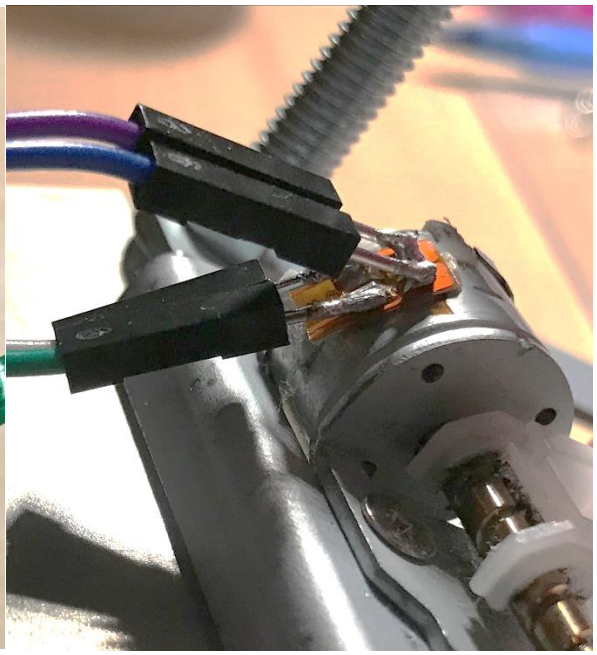


FIGURE 5

b) Use scrap DVD boxes to create two perpendicular planes on which the stepper motors will be placed. (fig. 6)

The stepper frames should be elevated so that the support moves smoothly along the rails, without bumping into anything. In order to do this, 4 M6 60 screws and 4 nuts were used for each stepper frame (fig. 7, fig. 8). Place the stepper frames mounted on screws onto a DVD box and carefully trace the screw positions onto the plane. The X-axis (horizontal plane) stepper should be placed length-wise on the plane and the Y-axis (vertical plane) stepper should be placed width-wise. Drill holes into the DVD boxes to insert the stepper frames.

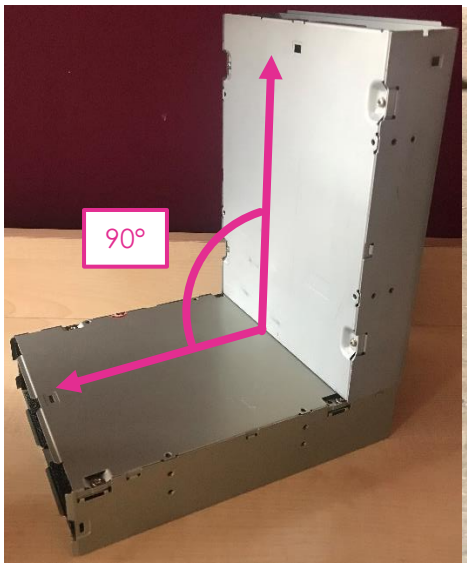


FIGURE 6

FIGURE 7

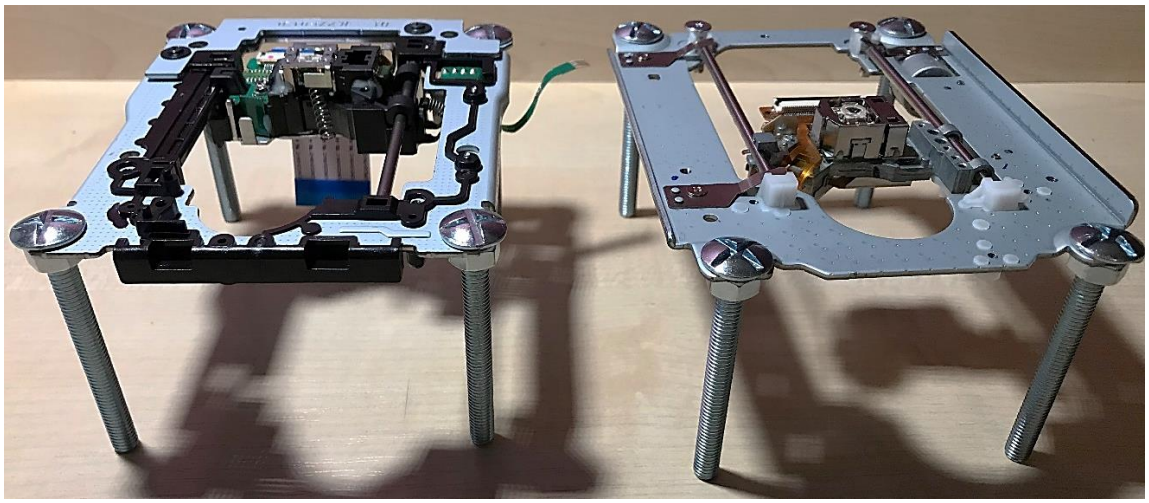


FIGURE 8

Next, the boxes should be affixed at a 90 degree angle using 3 M6 30 screws. (fig. 9) Any other screws and materials for the planes can be used as long as the result is a stable L-shaped construction on which the motors are mounted.

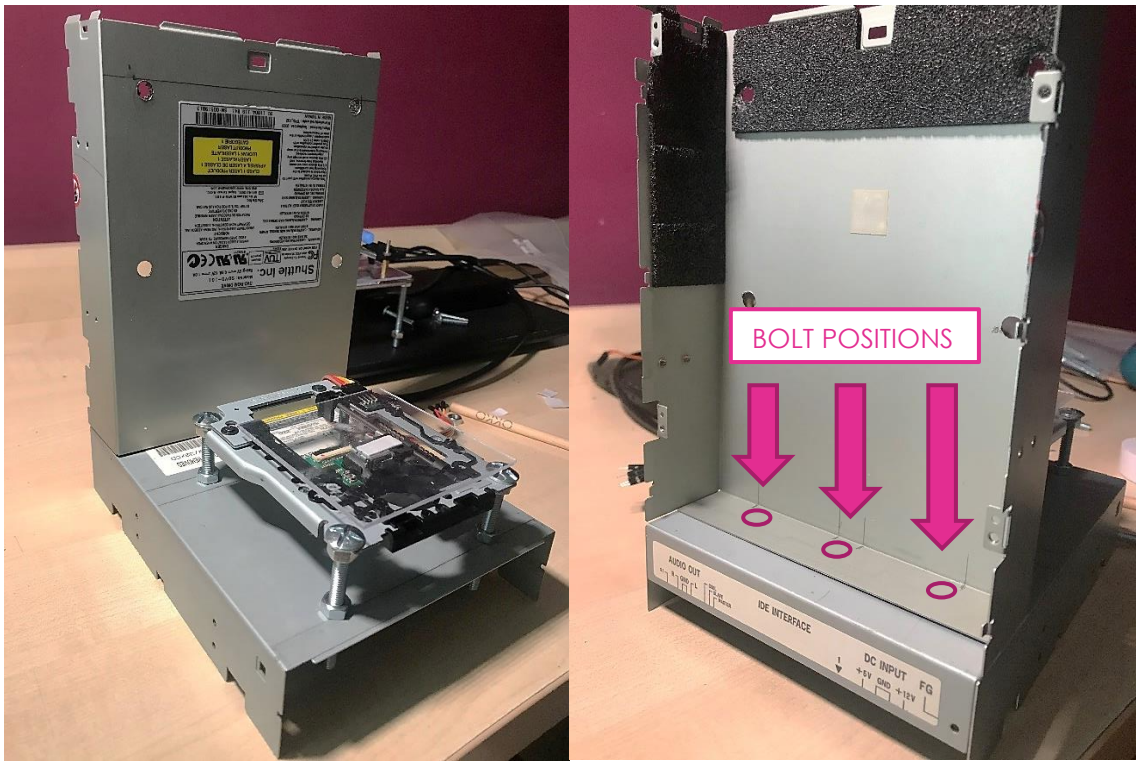


FIGURE 9

c) Cut an 8x8 cm plexiglass surface and glue it onto the X-axis (fig. 10). The X and Y axis are finished. At this point, the machine should look like fig. 11.

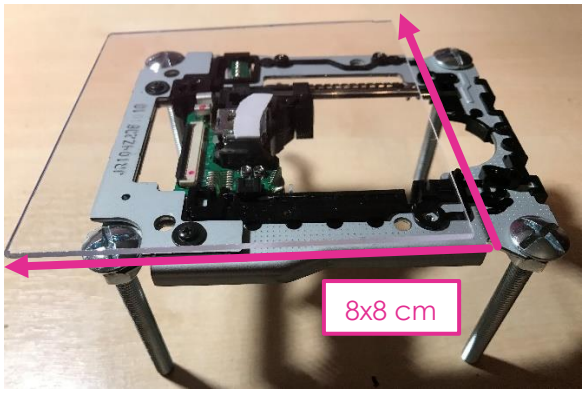


FIGURE 10

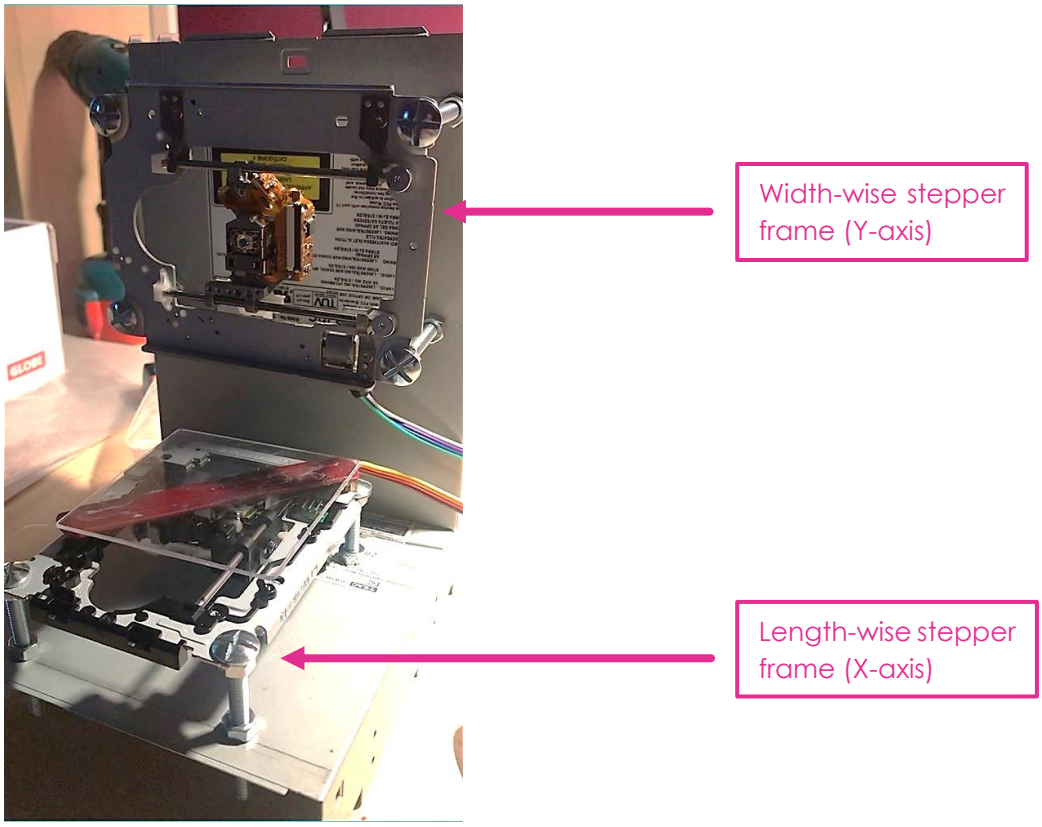


FIGURE 11

To construct the Z-axis, several things must be taken into consideration. The servo must push or pull the pen up and down, thus it cannot be connected directly to the pen (the servo rotates, describing a circular motion rather than a straight one). If the pen was being pulled directly by the servo, it would drag onto the paper instead of simply lifting during the pen-up command. The initial position must hold the pen on the drawing surface under tension, such that the pen is not being lifted while moving across the paper. The pen's height (and the pen itself) must be modifiable in order to fine tune the machine. Several versions were devised, before ending up to the final one.

A compass's pen holder was used to fix the pen onto the contraption as it is easily adjusted (fig. 12 and 13).

FIGURE 12

FIGURE 13

The rail and support system of another DVD drive was mounted on an 8x11 cm plexiglass using screws (fig. 14). Springs were inserted onto the rails to ensure that the pen will return to the initial position and remain under tension unless lifted by the servo. The servo was glued above the support between the rails and bound to it using string. As such, when the servo is functioning, it lifts the support vertically, along the parallel rails (fig. 15).

FIGURE 14

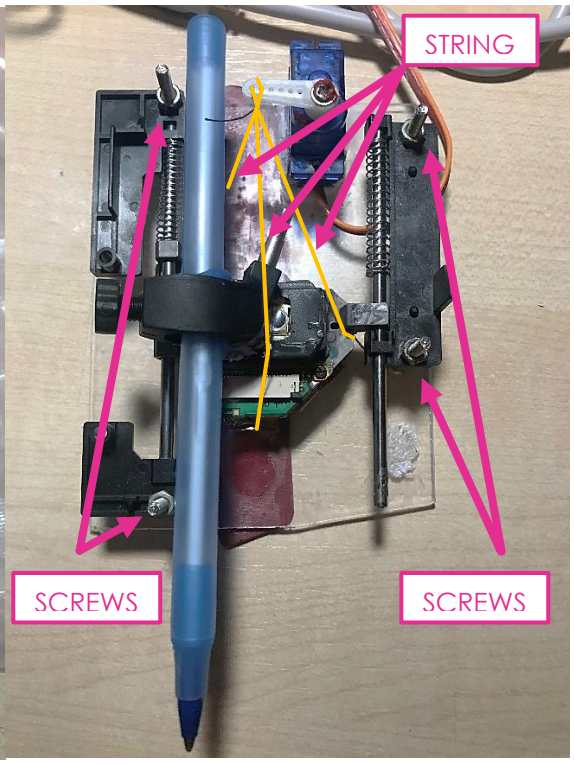


FIGURE 15

To prevent the railing from brushing onto the plexiglass, 1 cm standoffs were used to distance the two parts (fig. 16). The machine should look as in fig. 17 and 18.

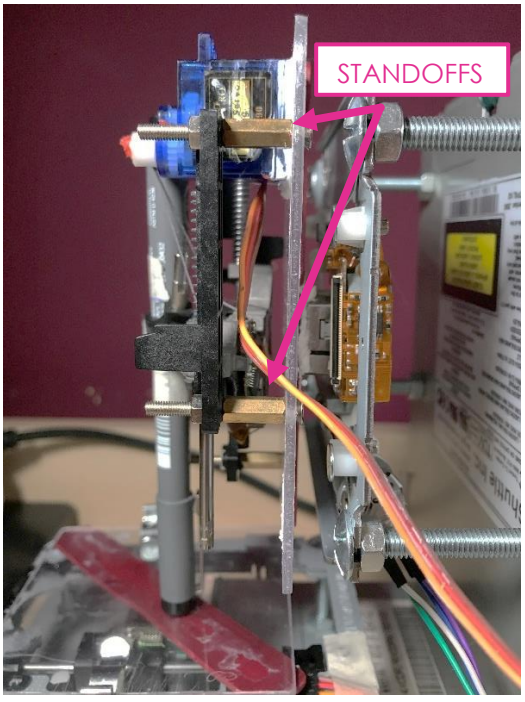


FIGURE 16

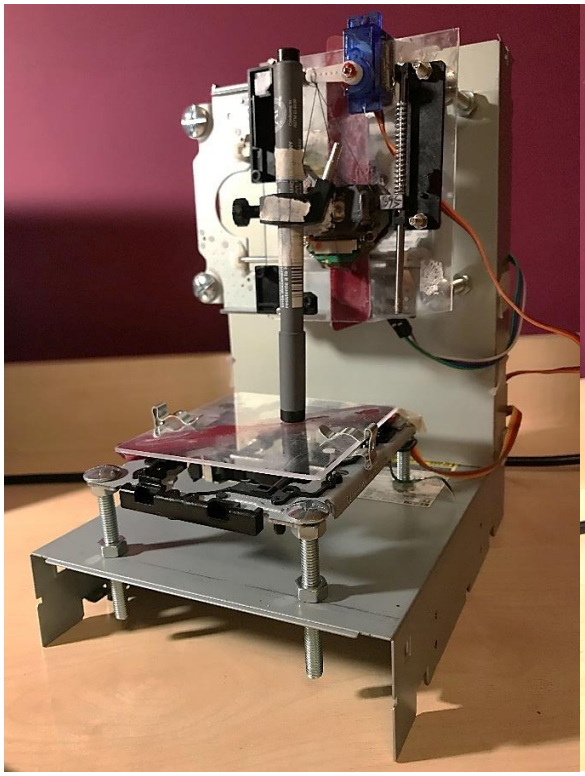


FIGURE 17

FIGURE 18

d) The assembling process is now complete. The final hardware step is the wiring and it should be done as in figure 18 provided by [12].

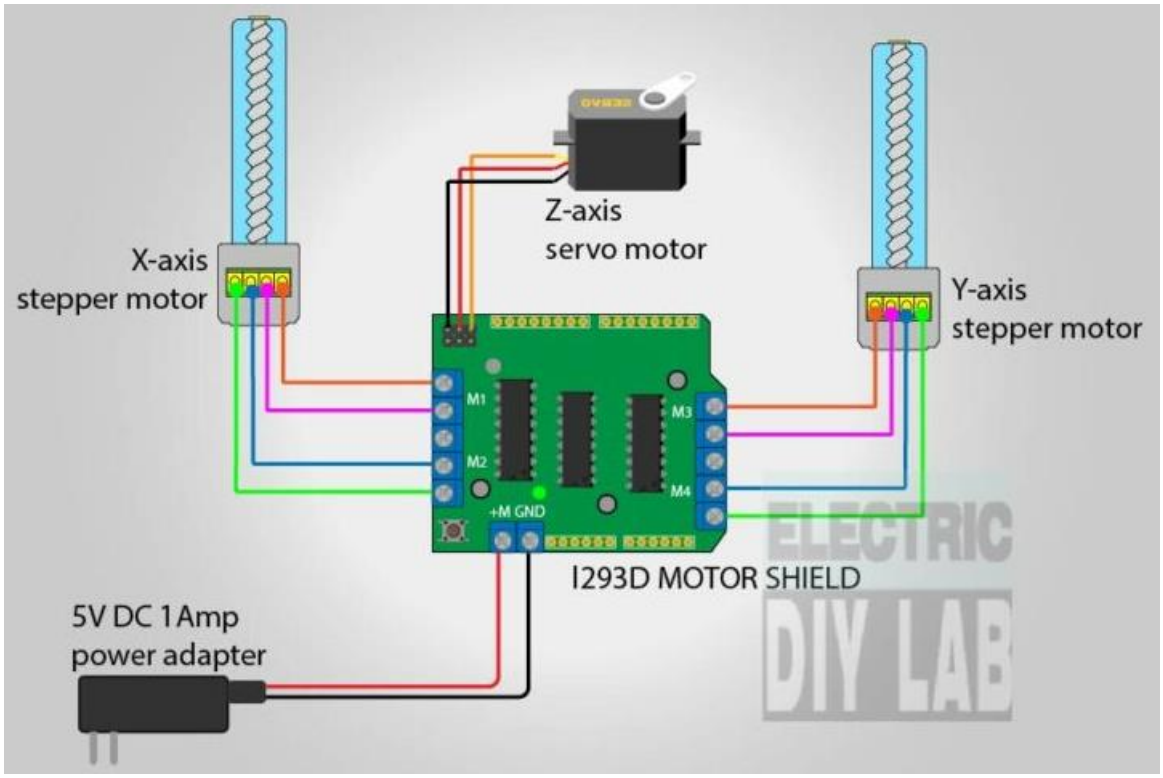


FIGURE 19

SOFTWARE

i. Check the functionality of the three motors using Arduino IDE. It is an optional step, but it is strongly recommended. It is also needed to find the servo motor pen up and pen down angles. For this project, the servo pen up position is 80˚ and the pen down position is 140˚.

Stepper test

```
#include <AFMotor.h>
AF_Stepper motorX(48, 1);  //stepper X with 48 steps/revolution, pin 1
AF_Stepper motorY(48, 2);  //stepper Y with 48 steps/revolution, pin 2

void setup() {
  motorX.setSpeed(150);     //stepper speed
  motorY.setSpeed(150);
  delay(1000);
}

void loop() {
//SINGLE - One coil is energized at a time.
//DOUBLE - Two coils are energized at a time for more torque.
//INTERLEAVE - Alternate between single and double to create a half-
//step in between.  This can result in smoother operation, but because
//of the extra half-step, the speed is reduced by half too.
//MICROSTEP - Adjacent coils are ramped up and down to create a number
//of 'micro-steps' between each full step. This results in finer
//resolution and smoother rotation, but with a loss in torque

  //X-axis motor
  motorX.step(256, FORWARD, SINGLE);     //steps, direction, style
  motorX.step(256, BACKWARD, SINGLE);

  motorX.step(256, FORWARD, DOUBLE);
  motorX.step(256, BACKWARD, DOUBLE);

  motorX.step(256, FORWARD, INTERLEAVE);
  motorX.step(256, BACKWARD, INTERLEAVE);

  motorX.step(256, FORWARD, MICROSTEP);
  motorX.step(256, BACKWARD, MICROSTEP);

  //Y-axis motor
  motorY.step(256, FORWARD, SINGLE);
  motorY.step(256, BACKWARD, SINGLE);

  motorY.step(256, FORWARD, DOUBLE);
  motorY.step(256, BACKWARD, DOUBLE);

  motorY.step(256, FORWARD, INTERLEAVE);
  motorY.step(256, BACKWARD, INTERLEAVE);

  motorY.step(256, FORWARD, MICROSTEP);
  motorY.step(256, BACKWARD, MICROSTEP);
}
```

Servo test

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos;        // variable to store the servo position

void setup() {
  myservo.attach(10);// attaches servo on pin 9 to servo object
}

void loop() {
  for (pos = 80; pos <= 140; pos += 1) {
    // goes from 80 degrees to 140 degrees
    myservo.write(pos);
    // tell servo to go to position in variable 'pos'
    delay(50);
    // waits 15ms for the servo to reach the position
  }
  for (pos = 140; pos >= 80; pos -= 1) {
    // goes from 140 degrees to 80 degrees
    myservo.write(pos);
    delay(50);
  }
}
```

After testing the motors, the Arduino UNO can be programmed with the code that interprets and executes the G-code files received on the Serial interface. The code used is mostly the one provided by [12], with a few essential changes. The G-code interpretation is an adaptation of Grbl, which is an open-source free software for CNC motion control that can run on Arduino IDE.

AFMotor library is used to control the steppers. The AF library reference mentions: “Step is a synchronous command and will not return until all steps have completed. For concurrent motion of two motors, you must handle the step timing for both motors and use the "onestep()" function below.”, thus the onestep function is the one needed.

CNC code 1/6

```
#include <Servo.h>
#include <AFMotor.h>

#define LINE_BUFFER_LENGTH 512

char STEP = MICROSTEP ;

// Servo position for Up and Down
const int penZUp = 80;
const int penZDown = 140;

// Servo on PWM pin 10
const int penServoPin = 10 ;

// Should be right for DVD steppers, but is not too important here
const int stepsPerRevolution = 48;

// create servo object to control a servo
Servo penServo;

// Initialize steppers for X- and Y-axis using this Arduino pins for the
L293D H-bridge
AF_Stepper myStepperY(stepsPerRevolution, 1);
AF_Stepper myStepperX(stepsPerRevolution, 2);

struct point {
  float x;
  float y;
  float z;
};

// Current position of plothead
struct point actuatorPos;

//  Drawing settings
float StepInc = 1;
int StepDelay = 1;
int LineDelay = 0;
int penDelay = 50;

// Motor steps to go 1 millimeter.
// Use test sketch to go 100 steps. Measure the length of line.
// Calculate steps per mm. Enter here.
float StepsPerMillimeterX = 100.0;
float StepsPerMillimeterY = 100.0;

// Drawing robot limits, in mm
float Xmin = 0;
float Xmax = 40;
float Ymin = 0;
float Ymax = 40;
float Zmin = 0;
float Zmax = 1;

float Xpos = Xmin;
float Ypos = Ymin;
float Zpos = Zmax;

// Set to true to get debug output.
boolean verbose = false;
```

CNC code 2/6

```
//  Needs to interpret
//  G1 for moving
//  M300 S140 (pen down)
//  M300 S80 (pen up)
//  Discard anything with a (
//  Discard any other command!

void setup() {
  Serial.begin( 9600 );

  penServo.attach(penServoPin);
  penServo.write(penZUp);
  delay(100);

  // Decrease if necessary
  myStepperX.setSpeed(600);
  myStepperY.setSpeed(600);

  //  Notifications
  Serial.println("Mini CNC Plotter alive and kicking!");
  Serial.print("X range is from ");
  Serial.print(Xmin);
  Serial.print(" to ");
  Serial.print(Xmax);
  Serial.println(" mm.");
  Serial.print("Y range is from ");
  Serial.print(Ymin);
  Serial.print(" to ");
  Serial.print(Ymax);
  Serial.println(" mm.");
}
```

CNC code 3/6

```
void loop(){
  delay(100);
  char line[LINE_BUFFER_LENGTH];
  char c;
  int lineIndex;
  bool lineIsComment, lineSemiColon;
  lineIndex = 0;
  lineSemiColon = false;
  lineIsComment = false;

while (1) {
    // Serial reception - Mostly from Grbl, added semicolon support
    while (Serial.available() > 0) {
      c = Serial.read();
      if (( c == '\n') || (c == '\r') ) { // End of line reached
        if ( lineIndex > 0 ) {            // Line is complete. Then execute!
          line[lineIndex] = '\0';         // Terminate string
          if (verbose) {
            Serial.print("Received : ");
            Serial.println(line);
          }
          processIncomingLine(line, lineIndex);
          lineIndex = 0;
        }
        else { // Empty or comment line. Skip block.}
        lineIsComment = false;
        lineSemiColon = false;
        Serial.println("ok");
      }
      else {
        if ( (lineIsComment) || (lineSemiColon) ) {
        // Throw away all comment characters
          if ( c == ')' )
                lineIsComment = false;// End of comment. Resume line.
        }
        else {
          if ( c <= ' ' ) { // Throw away whitepace and control characters
          }
          else if ( c == '/' ) { // Block delete not supported. Ignore char.
          }
          else if ( c == '(' ) {
         // Enable comments flag and ignore all characters until ')' or EOL.
            lineIsComment = true;
          }
          else if ( c == ';' ) {
            lineSemiColon = true;
          }
          else if ( lineIndex >= LINE_BUFFER_LENGTH - 1 ) {
            Serial.println( "ERROR - lineBuffer overflow" );
            lineIsComment = false;
            lineSemiColon = false;
          }
          else if ( c >= 'a' && c <= 'z' ) { // Upcase lowercase
            line[ lineIndex++ ] = c - 'a' + 'A';
          }
          else {
            line[ lineIndex++ ] = c;
          }
        }
      }
    }
  }
}
```

CNC code 4/6

```
void processIncomingLine( char* line, int charNB ) {
  int currentIndex = 0;
  char buffer[ 64 ]; // Hope that 64 is enough for 1 parameter
  struct point newPos;
  newPos.x = 0.0;
  newPos.y = 0.0;

  while ( currentIndex < charNB ) {
    switch ( line[ currentIndex++ ] ) { // Select command, if any
      case 'G':
        buffer[0] = line[ currentIndex++ ];
        buffer[1] = '\0';

        switch ( atoi( buffer ) ) { // Select G command
          case 1:
            char* indexX = strchr( line + currentIndex, 'X' );
            // Get X/Y position in the string (if any)
            char* indexY = strchr( line + currentIndex, 'Y' );
            if ( indexY <= 0 ) { //Y not found, set only X to new position
              newPos.x = atof( indexX + 1);
              newPos.y = actuatorPos.y;
            }
            else if ( indexX <= 0 ) {
              newPos.y = atof( indexY + 1);
              newPos.x = actuatorPos.x;
            }
            else { //if both were found, change both positions
              newPos.y = atof( indexY + 1);
              *indexY = '\0';
              newPos.x = atof( indexX + 1);
            }
            drawLine(newPos.x, newPos.y );
            actuatorPos.x = newPos.x;
            actuatorPos.y = newPos.y;
            break;
        }
        break;
      case 'M':
        buffer[0] = line[ currentIndex++ ];
        buffer[1] = line[ currentIndex++ ];
        buffer[2] = line[ currentIndex++ ];
        buffer[3] = '\0';
        switch ( atoi( buffer ) ) {
          case 300:
              char* indexS = strchr( line + currentIndex, 'S' );
              float Spos = atof( indexS + 1);
              Serial.println(Spos);
              if (Spos == penZDown) {
                penDown();
              }
              if (Spos == penZUp) {
                penUp();
              }
              break;
          default:
            Serial.print( "Command not recognized : M");
            Serial.println( buffer );
        }
    }
  }
}
```

CNC code 5/6

```
/*********************************
   Draw a line from (x0;y0) to (x1;y1).
   int (x1;y1) : Starting coordinates
   int (x2;y2) : Ending coordinates
 *********************************/
void drawLine(float x1, float y1) {

  if (verbose)
  {
    Serial.print("fx1, fy1: ");
    Serial.print(x1);
    Serial.print(",");
    Serial.print(y1);
    Serial.println("");
  }

  //  Bring instructions within limits
  if (x1 >= Xmax) {
    x1 = Xmax;
  }
  if (x1 <= Xmin) {
    x1 = Xmin;
  }
  if (y1 >= Ymax) {
    y1 = Ymax;
  }
  if (y1 <= Ymin) {
    y1 = Ymin;
  }

  if (verbose)
  {
    Serial.print("Xpos, Ypos: ");
    Serial.print(Xpos);
    Serial.print(",");
    Serial.print(Ypos);
    Serial.println("");
    Serial.print("x1, y1: ");
    Serial.print(x1);
    Serial.print(",");
    Serial.print(y1);
    Serial.println("");
  }

  //  Convert coordinates to steps
  x1 = (int)(x1 * StepsPerMillimeterX);
  y1 = (int)(y1 * StepsPerMillimeterY);
  float x0 = Xpos;
  float y0 = Ypos;

  //  Let's find out the change for the coordinates
  long dx = abs(x1 - x0); //number of steps to be done on X-axis
  long dy = abs(y1 - y0); //number of steps to be done on Y-axis
  int sx = x0 < x1 ? StepInc : -StepInc; //step forward/backward
  int sy = y0 < y1 ? StepInc : -StepInc;

  ng i;
  long over = 0;

//FUNCTION CONTINUES ON NEXT CODE BOX
```

CNC code 6/6

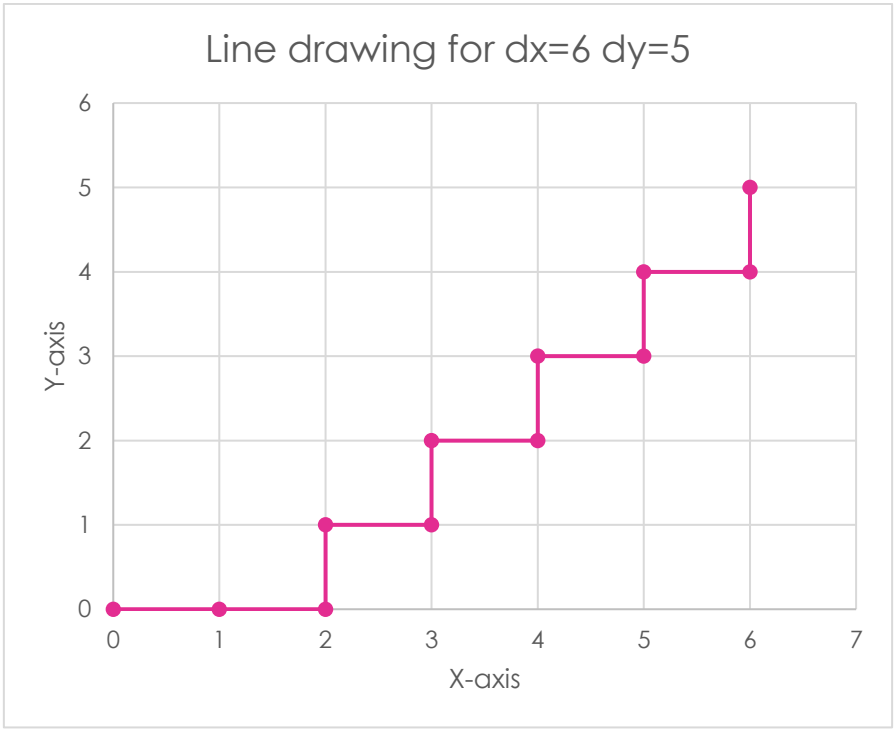
```
   if (dx > dy) { //start with X-axis
    for (i = 0; i < dx; i++) {
      myStepperX.onestep(sx, STEP); //step on X-axis in sx direction
      over += dy; //if over >= dx, it is time to step on the other axis
      if (over >= dx) { //check if the Y-axis position must be incremented
        over -= dx; //reset the Y-axis increment "timer"
        myStepperY.onestep(sy, STEP); //step on Y-axis in sy direction
      }
      delay(StepDelay);
    }
  }
  else {        //start with Y-axis
    for (i = 0; i < dy; i++) {
      myStepperY.onestep(sy, STEP); //step on Y-axis in sy direction
      over += dx; //if over >= dy, it is time to step on the other axis
      if (over >= dy) { //check if the X-axis position must be incremented
        over -= dy; //reset the X-axis increment "timer"
        myStepperX.onestep(sx, STEP); //step on X-axis in sx direction
      }
      delay(StepDelay);
    }
  }
  if (verbose)
  {
    Serial.print("dx, dy:");
    Serial.print(dx);
    Serial.print(",");
    Serial.print(dy);
    Serial.println("");
    Serial.print("Going to (");
    Serial.print(x0);
    Serial.print(",");
    Serial.print(y0);
    Serial.println(")")
  }
  delay(LineDelay); //  Delay before any next lines are submitteD
  //  Update the positions
  Xpos = x1;
  Ypos = y1;
}

void penUp() {//  Raises pen
  penServo.write(penZUp);
  delay(penDelay);
  Zpos = Zmax;
  digitalWrite(15, LOW);
  digitalWrite(16, HIGH);
  if (verbose) {
    Serial.println("Pen up!");
  }
}
void penDown() {//  Lowers pen
  penServo.write(penZDown);
  delay(penDelay);
  Zpos = Zmin;
  digitalWrite(15, HIGH);
  digitalWrite(16, LOW);
  if (verbose) {
    Serial.println("Pen down.");
  }
}
```

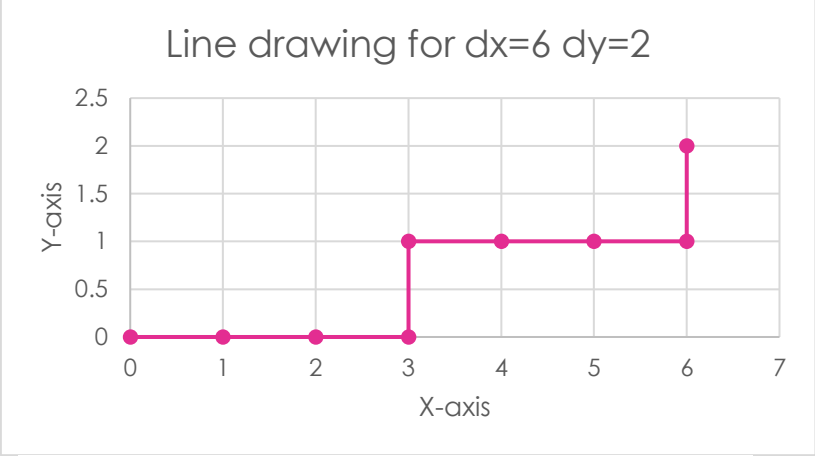
The tracing of an algorithm used in the drawLine function is represented in the graphs below, with the intent of easing the understanding process of the algortihm's functioning principle. The starting point of each line is considered to be (0,0) for the simplicity of the tracing. Thus, the dx and dy values exemplified will be the final coordinates of the line (dx-0, dy-0). The algorithm illustrated is:

Line drawing algorithm

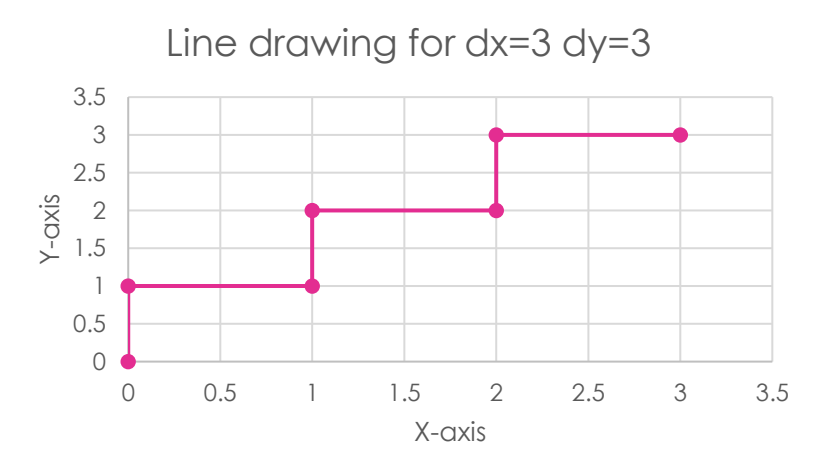
```
if (dx > dy) { //start with X-axis
  for (i = 0; i < dx; i++) {
    myStepperX.onestep(sx, STEP); //step on X-axis in sx direction
    over += dy; //if over >= dx, it is time to step on the other axis
    if (over >= dx) { //check if the Y-axis position must be incremented
      over -= dx; //reset the Y-axis increment "timer"
      myStepperY.onestep(sy, STEP); //step on Y-axis in sy direction
    }
    delay(StepDelay);
  }
}
else {        //start with Y-axis
  for (i = 0; i < dy; i++) {
    myStepperY.onestep(sy, STEP); //step on Y-axis in sy direction
    over += dx; //if over >= dy, it is time to step on the other axis
    if (over >= dy) { //check if the X-axis position must be incremented
      over -= dy; //reset the X-axis increment "timer"
      myStepperX.onestep(sx, STEP); //step on X-axis in sx direction
    }
    delay(StepDelay);
}
```



dx	6										
dy	5										
x	1	2	2	3	3	4	4	5	5	6	6
y	0	0	1	1	2	2	3	3	4	4	5
i	0	1	1	2	2	3	3	4	4	5	5
over	5	10	4	9	3	8	3	7	1	6	0
x-axis step	Y	Y	N	Y	N	Y	N	Y	N	Y	N
y-axis step	N	N	Y	N	Y	N	Y	N	Y	N	Y



dx	6							
dy	2							
x	1	2	3	3	4	5	6	6
y	0	0	0	1	1	1	1	2
i	0	1	2	2	3	4	5	5
over	2	4	6	0	2	4	6	0
x-axis step	Y	Y	Y	N	Y	Y	Y	N
y-axis step	N	N	N	Y	N	N	N	Y



dx	3						
dy	3						
x	0	0	1	1	2	2	3
y	0	1	1	2	2	3	3
i	0	0	1	1	2	2	
over	3	0	3	0	3	0	3
x-axis step	Y	N	Y	N	Y	N	Y
y-axis step	N	Y	N	Y	N	Y	N

ii. Next, download and install Inkscape version 0.48.5 from the Bibliographic Study/Software required section. Add the Makerboat G-code extension as described in the Bibliographic Study/Software required/INKSCAPE AND MAKERBOAT G-CODE EXTENSION section. Then, create a G-code file as follows.

TURN A TEXT INTO G-CODE

1. Make the following changes (fig. 20):

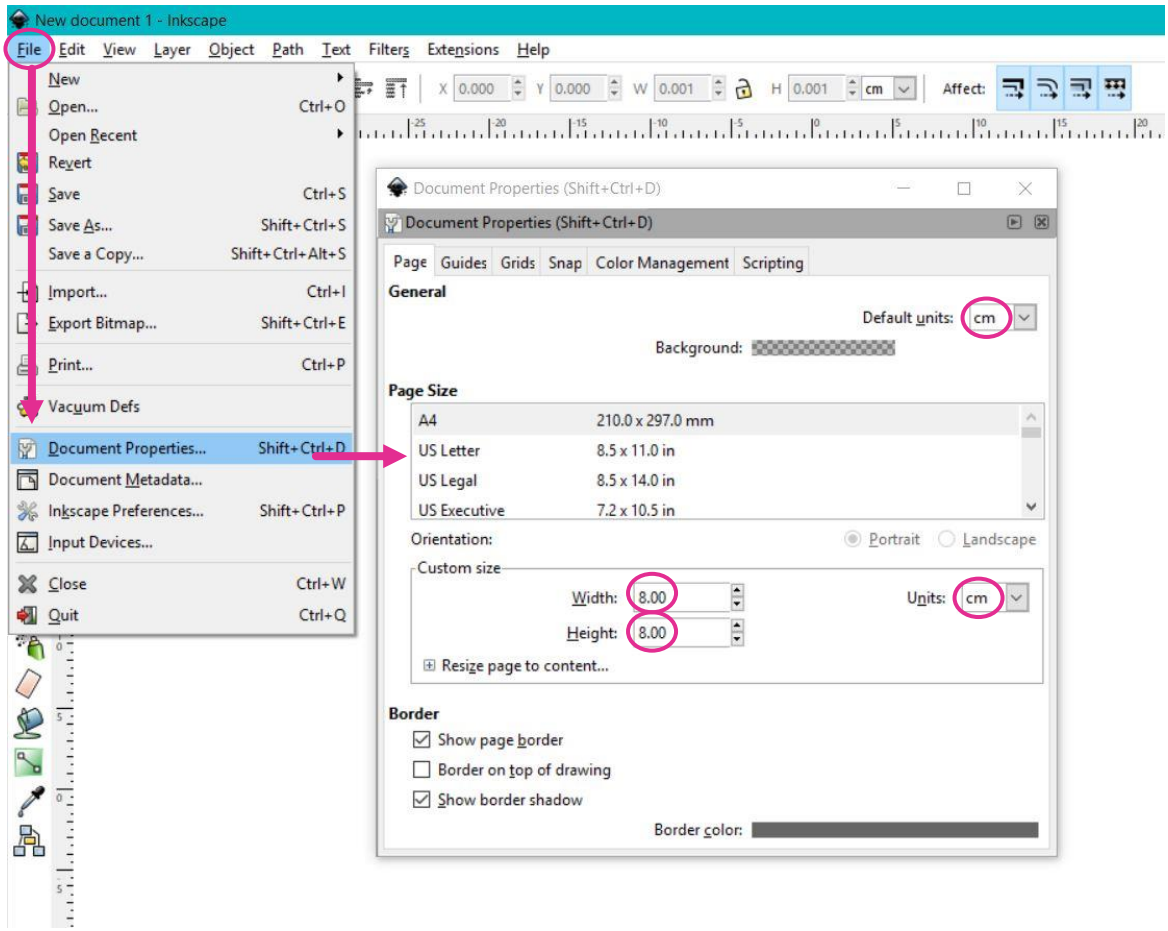


FIGURE 20

2. Create a text object with the desired text. Resize it to fit the 4x4 cm upper right corner of the 8x8 cm surface. Convert the object into a vectorial drawing by selecting the object and clicking *Path->Object to Path and Path->Dynamic Offset*. This will turn the text into a set of nodes. (fig. 21)

3. Save the image as a G-code in the desired destination (fig. 22). A prompt will pop-up, to set the CNC machine specifications. The most important ones to check are pen-up and pen-down (fig. 23).

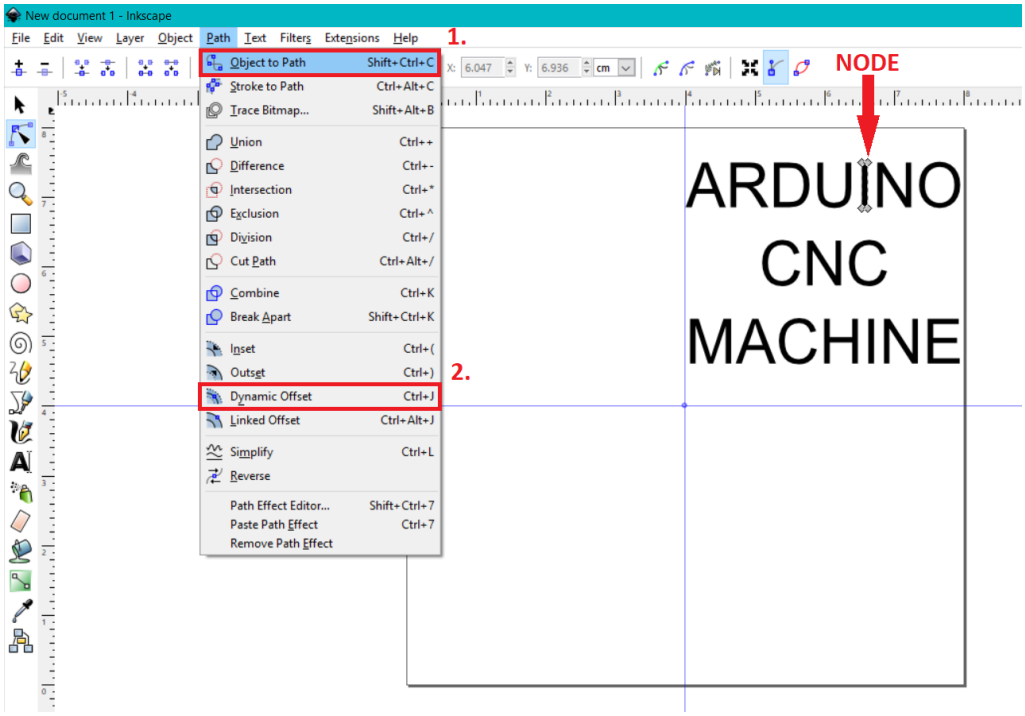


FIGURE 21

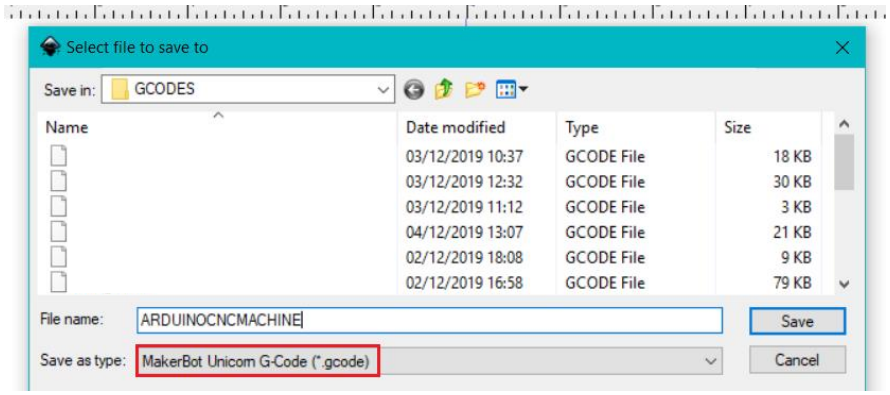


FIGURE 22

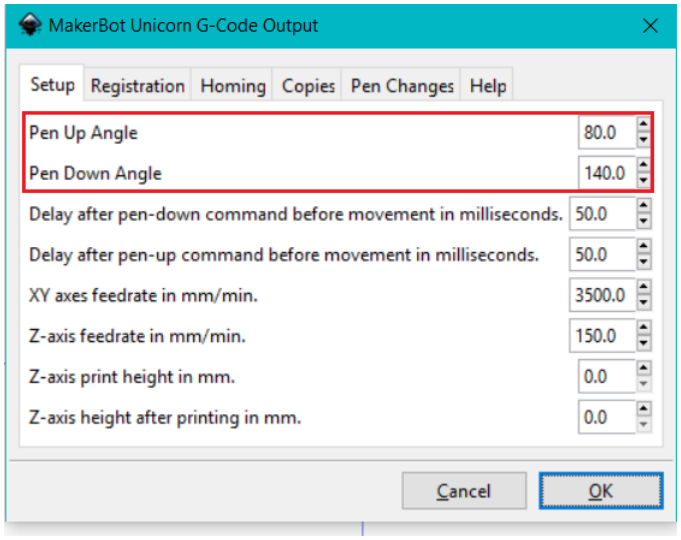


FIGURE 23

TURN AN IMAGE INTO G-CODE

1. Step one from Turn a text into G-code is to be applied.
2. Drag and drop the image and select it. Resize it to fit the 4x4 cm upper right corner of the 8x8 cm surface. Whilst selected, go to *Path->Trace Bitmap…*, modify the brightness cutoff threshold and the edge detection threshold if needed, click *update*, then *ok*. A bitmap of the original image will be created. The original image can be discarded as any further actions will be done on the bitmap. (fig. 24)
3. Convert the object into a vectorial drawing by selecting the object and clicking *Path->Object to Path and Path->Dynamic Offset*. The option *Path->Stroke to path,* though optional, can be also be selected to create a path (outline) out of a stroke (fig. 25)

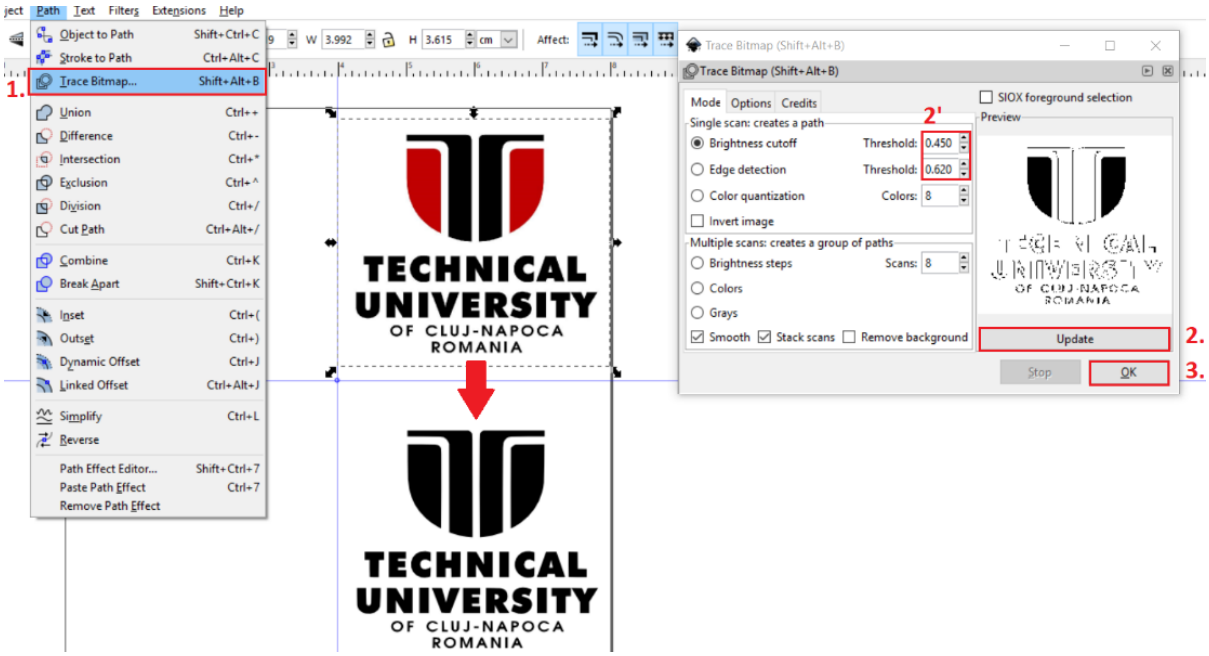


FIGURE 24

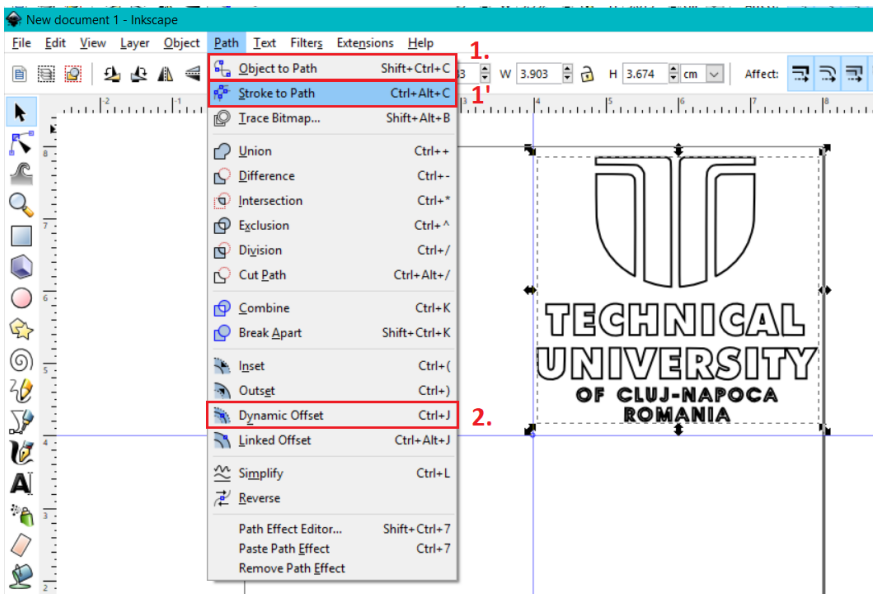


FIGURE 25

4. Step three from Turn a text into G-code is to be applied.

iii. Then, download and install Processing IDE from the Bibliographic Study/Software required. Create a sketch named GCTRL.pde and write the following code provided by [12] and modified according to the machine's specifications:

GCTRL 1/2

```
import java.awt.event.KeyEvent;
import javax.swing.JOptionPane;
import processing.serial.*;

Serial port = null;
String portname = null;
boolean streaming = false;
float speed = 0.001;
String[] gcode;
int i = 0;

void openSerialPort(){
  if (portname == null) return;
  if (port != null) port.stop();
  //create a new serial interface connection
  port = new Serial(this, portname, 9600);
  port.bufferUntil('\n');
}

void selectSerialPort(){
  String result = (String) JOptionPane.showInputDialog(frame,
    "Select the serial port that corresponds to your Arduino board.",
    "Select serial port", JOptionPane.PLAIN_MESSAGE, null, Serial.list(),0);
  //connect to microcontroller's board
  if (result != null) {
    portname = result;
    openSerialPort();
  }
}

void setup(){
  size(500, 250);
  openSerialPort();
}

void draw(){
  background(0);
  fill(255);
  int y = 24, dy = 12;
  text("INSTRUCTIONS", 12, y); y += dy;
  text("p: select serial port", 12, y); y += dy;
  text("arrow keys: jog in x-y plane", 12, y); y += dy;
  text("5 & 2: jog in z axis", 12, y); y += dy;
  text("$: display grbl settings", 12, y); y+= dy;
  text("h: go home", 12, y); y += dy;
  text("0: zero machine (set home to the current location)", 12, y); y +=
dy;
  text("g: stream a g-code file", 12, y); y += dy;
  text("x: stop streaming g-code (this is NOT immediate)", 12, y); y += dy;
  y = height - dy;
  text("current jog speed: " + speed + " inches per step", 12, y); y -= dy;
  text("current serial port: " + portname, 12, y); y -= dy;
}
```

GCTRL 2/2

```
void keyPressed(){
  if (key == '1') speed = 0.001;
  if (key == '2') speed = 0.01;
  if (key == '3') speed = 0.1;
  if (!streaming) {
    if (keyCode == LEFT) port.write("G21/G90/G1 X-10  F3500\n");
    if (keyCode == RIGHT) port.write("G21/G90/G1 X10 F3500\n");
    if (keyCode == UP) port.write("G21/G90/G1 Y10 F3500\n");
    if (keyCode == DOWN) port.write("G21/G90/G1 Y-10 F3500\n");
    if (key == '5') port.write("M300 S80\n");
    if (key == '2') port.write("M300 S140\n");
    if (key == 'h') port.write("G90\nG20\nG00 X0.000 Y0.000 Z0.000\n");
    if (key == 'v') port.write("$0=75\n$1=74\n$2=75\n");
    if (key == 's') port.write("$3=10\n");
    if (key == 'e') port.write("$16=1\n");
    if (key == 'd') port.write("$16=0\n");
    if (key == '0') openSerialPort();
    if (key == 'p') selectSerialPort();
    if (key == '$') port.write("$$\n");
  }
  if (!streaming && key == 'g') {
    gcode = null; i = 0;
    File file = null;
    println("Loading file...");
    selectInput("Select a file to process:", "fileSelected", file);
  }
  if (key == 'x') streaming = false;
}
void fileSelected(File selection) {
  if (selection == null) {
    println("Window was closed or the user hit cancel.");
  } else {
    println("User selected " + selection.getAbsolutePath());
    gcode = loadStrings(selection.getAbsolutePath());
    if (gcode == null) return;
    streaming = true;
    stream();
  }
}
void stream(){
  if (!streaming) return;
  while (true) {
    if (i == gcode.length) {
      streaming = false;
      return;
    }
    if (gcode[i].trim().length() == 0) i++;
    else break;
  }
  println(gcode[i]);
  port.write(gcode[i] + '\n');
  i++;
}

void serialEvent(Serial p)
{
  String s = p.readStringUntil('\n');
  println(s.trim());

  if (s.trim().startsWith("ok")) stream();
  if (s.trim().startsWith("error")) stream();
}
```

Click *Run*. A window will pop up, containing the usage instructions. Click *p* to connect to the serial port of the microcontroller. Test the CNC servo using keys 5 and 2. Test the steppers using the arrow keys to move the pen forward, backward, left and right 1cm. Click *h* to return to the initial position of the CNC. Click *g* to start streaming a G-code file of your choice; click x to prematurely stop the streaming of the file. A few other instructions are described in the window (fig. 26).

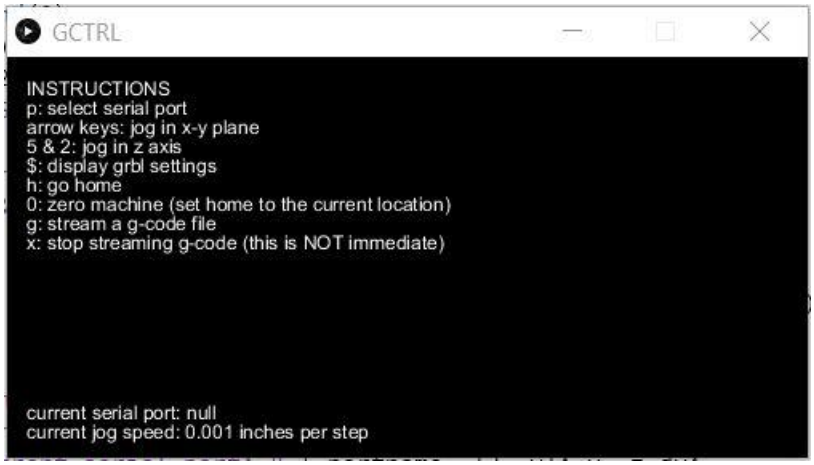


FIGURE 26

5. TESTING

The machine needs to be calibrated before usage for best result. This is done while running Gctrl Processing, after connecting the port. The height of the pen and of the drawing surface were verified and modified so that an even pressure is being applied to any corner of the surface. If the pressure is not evenly distributed, the pen will either not touch or drag on the surface.

The calibration test involves a trial and error method consisting in drawing squares in all extremes of the surface (NE, NW, SE, SW) using the arrow keys for moving the X and Y-axis and keys 5 and 2 for the Z-axis (pen-up, pen-down) (fig. 27).

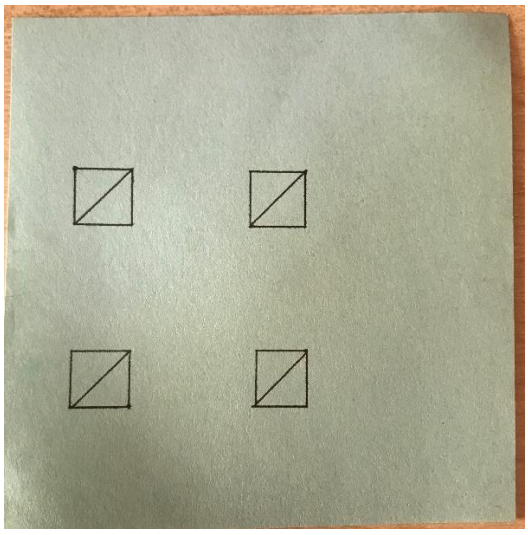


FIGURE 27

I. TEXT BASED G-CODE TEST

The text "ARDUINO DRAWING MACHINE" was converted into G-code using the method described in the Design section. The result is presented in figure 28.

II. IMAGE BASED G-CODE TEST

The following images were converted into G-code using the method described in the Design section (results are presented in fig. 29-31):

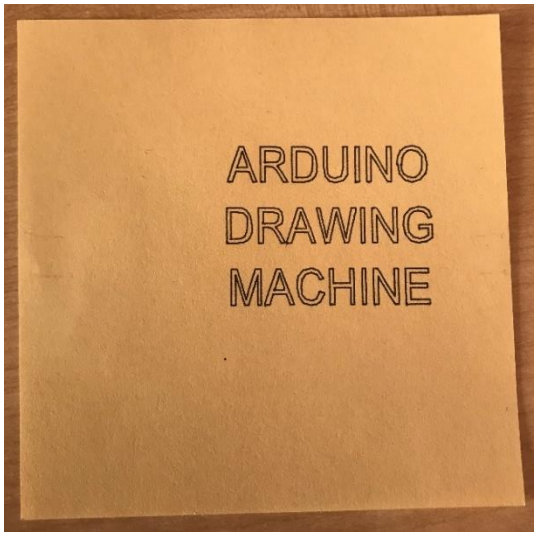


FIGURE 28

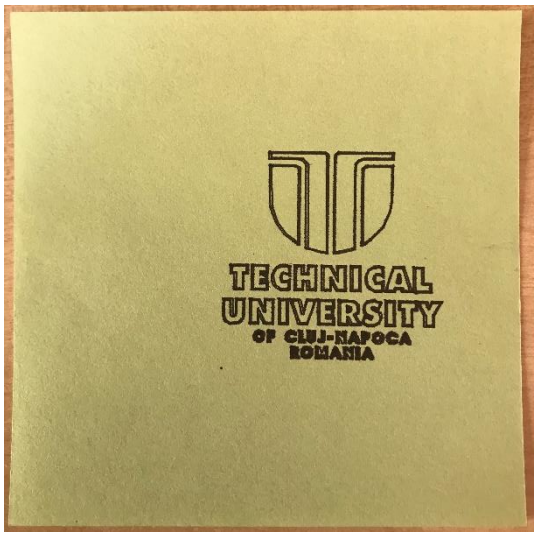


FIGURE 29

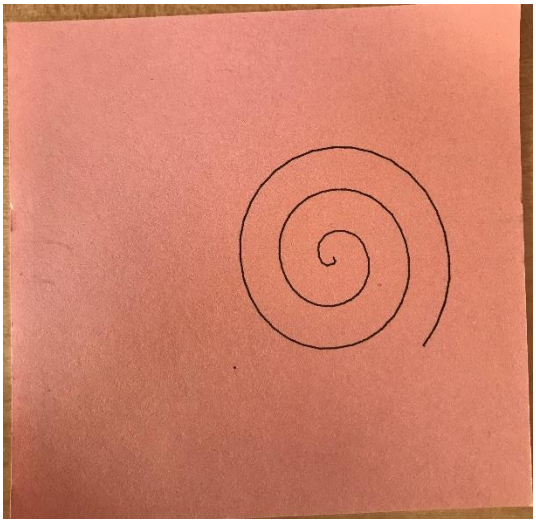


FIGURE 30

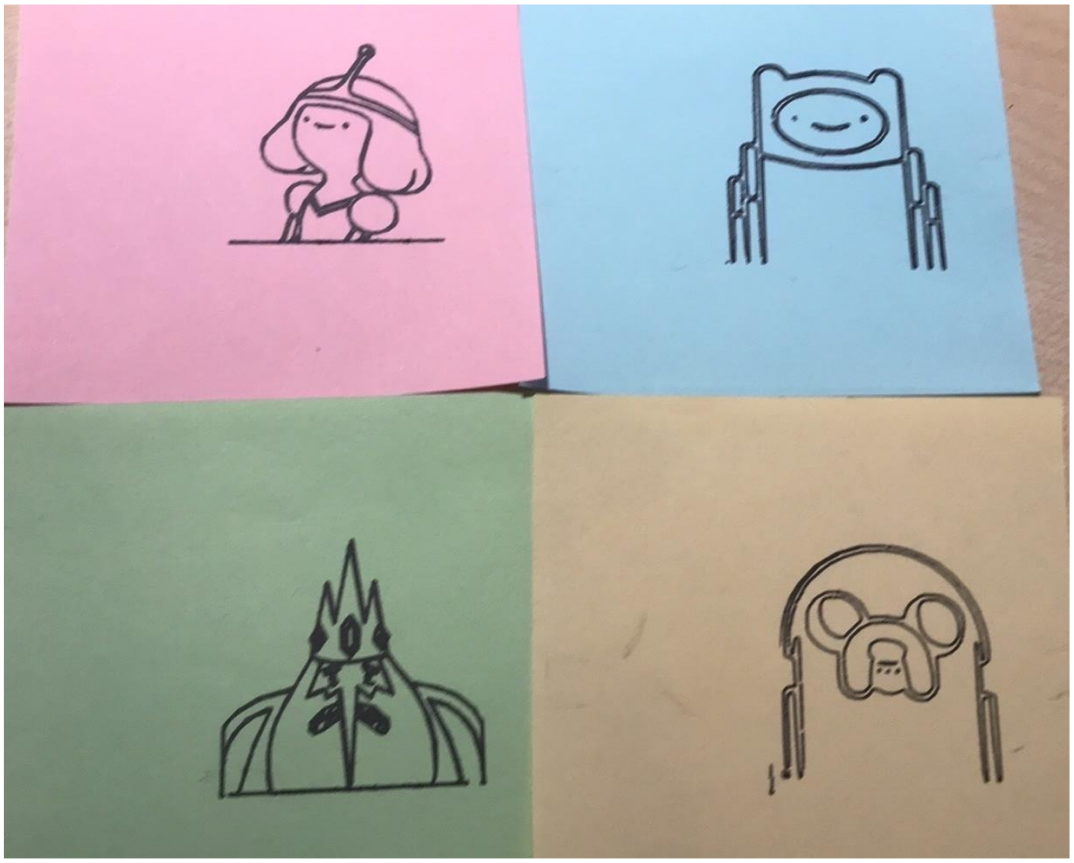


FIGURE 31

7. CONCLUSIONS

In conclusion, the Arduino CNC Machine is a beginner to intermediate level project, yet an ambitious one, as it requires both time and resources.

The realisation of the project represents the fusion of many different fields pertaining to automation and computer science. It has a mechanical/automatic component, since it entails the construction of the machine from scrap DVD materials and encourages creative

thinking in the assembling process (especially the Z-axis module). The wiring and code constitute the electronic part, encompassing knowledge of microprocessors, expansion boards (shields), servo and stepper motors and basic electric circuits (the H-Bridge) and Arduino and Java language development skills.

The result is a miniature version of a simplified plotter. It substitutes the time consuming and imprecise manual operation with an automated operation mode via the use of G-code interpretation and execution. Granted the work surface (4x4 cm) is limited due to the stepper motor rail system size, the sketches done by the CNC machine are quite precise if the recommended fine tuning is performed. The quality of the machine's drawings is satisfactory.

The project enables further developments by replacing the Z-axis pen module with a laser to create an engraving machine, or by replacing the whole Z-axis module with another stepper motor frame to which a 3d printer pen can be attached in order to obtain a 3d printer.

Finally, the project is worthwhile from an academic point of view, because it links and develops the expertise of various subject matters

8. BIBLIOGRAPHY

- [1] Arduino. Introduction. 2019. https://www.arduino.cc/en/guide/introduction
- [2] Wikipedia. Numerical control. 2019. https://en.wikipedia.org/wiki/Numerical_control
- [3] HarshalK12. Arduino based CNC Plotter From Old CD-ROM. https://tinyurl.com/uuoyrsa
- [4] Ardumotive_com. Arduino Mini CNC Plotter Machine From Dvd Drives. https://tinyurl.com/rtwtcsp
- [5] Wikipedia. Arduino UNO. 2019. https://en.wikipedia.org/wiki/Arduino_Uno
- [6] Wiki sunfounder. L293D Motor Shield Driver Shield. 2018 https://tinyurl.com/t6j5zds
- [7] Peter Y. K. Cheung. Servo motor SG90 data sheet. 2019 https://tinyurl.com/y85795m3
- [8] Wikipedia. Stepper motor. 2019. https://en.wikipedia.org/wiki/Stepper_motor
- [9] Ingo Lohs. Check a Stepper Motor From the DVD-ROM/Pairing the 4 Stepper Cables. 2019. https://tinyurl.com/warlfsh
- [10] Tinkernut. Hack old CD-ROM into a CNC Machine. 2015. https://tinyurl.com/pku3d95
- [11] Hardware Boy. Mini CNC Machine simple frame Assembling. 2019. https://tinyurl.com/yxrfe54o
- [12] Sandeep. How to make Arduino mini CNC plotter machine. 2019. https://tinyurl.com/vua6m2h
- [13] Adafruit. AF Stepper Class. 2012. https://tinyurl.com/lzn3oho