TEHNICAL UNIVERSITY OF CLUJ-NAPOCA

CNC MACHINE WITH ARDUINO

Student: Gyarmathy Tímea

Group: 30433/1

Contents

Contents 2
Introduction 3
Literature review 3
Mini Cnc Machine Arduino Based by instructAbdo [1] 3
Sketch it (CNC plotter) [2] 4
New Design DIY Arduino Based Mini CNC Machine [3] 4
Design 4
Hardware 4
Software 7
Startup 10
Implementation 11
Hardware 11
Software 15
Processing file – User Interface 15
Arduino file – Board program 17
Creating G-code files 18
Testing 19
Conclusion ..... **Error! Bookmark not defined.**
References 22

Introduction

Computer numerical control (CNC) is the automation of machine tools by means of computers executing pre-programmed sequences of machine control commands. This contrasts with machines that are manually controlled by hand wheels or levers, or mechanically automated by cams alone.

Motion is controlled along multiple axes, normally at least two (X and Y), and a tool spindle that moves in the Z (depth). The position of the tool is driven by direct-drive stepper motor or servo motors in order to provide highly accurate movements, or in older designs, motors through a series of step down gears.

CNC-like systems are now used for any process that can be described as a series of movements and operations. These include laser cutting, welding, friction stir welding, ultrasonic welding, flame and plasma cutting, bending, spinning, hole-punching, pinning, gluing, fabric cutting, sewing, tape and fiber placement, routing, picking and placing, and sawing.

Industrial CNC machines are used in wood/metal working to cut or to carve these materials. To introduce this project, we describe that in case of cutting, the machine focuses on a 2D model, and in case of carving depth is also considered.

This project will consist of a mini CNC machine which will draw designs on paper, hence it resembles the cutting model, because we are only interested in the plane of the paper. We will tell the machine only when and where to let down or respectively to lift up the pen from the paper.

Literature review

This is a project that many enthusiasts could find interesting, thus there are some descriptions on the internet from computer science hobbyists describing their personal method of realizing it. These differ in many ways, from the positioning of the axes of the stepper motors to the code realization. Next, some of the best realizations that are relevant to introduce this project will be listed.

Mini Cnc Machine Arduino Based by instructAbdo [1]

This project description gives us an insight on what the project should consist of. To realize the project it needs two stepper motors for the X and Y axes of the CNC, and for this purpose it uses the motors of some DVD/CD drives. As for the Z axis, lifting and letting down the pen on the paper, it uses a servomotor. Additionally, the two stepper motors need to be driven by drivers, one each, and in this project a “Shield driver motor L293D adafruit v1” is used.

In order to be able to load drawings, the Gcode format is used. Gcode is a numerical control programming language widely used in computer-aided manufacturing to control automated machine tools. The instructions defined by it tell the computer “how” and “what” to do. In this

case a Gcode file will contain when and where to let down the pen (by the servo) as well as when to lift it.

Important feature to be noted about this project that it is realized on a fixed stand, and the X axis is mounted on the vertical plate which holds the pen, while the Y axis is mounted horizontally and it moves the little plate on which the paper is clipped.

Sketch it (CNC plotter) [2]

We can find a similar project on the official Arduino project hub.

It uses the same components as mentioned above.

Let's mention this time the software part. To be able to interact with the machine a small interface is used which simply waits for the user to input the key of the command he/she wants to execute. Such commands include selecting the gcode file, moving the pen to (0,0), and of course starting and stopping the streaming of the gcode.

This project also uses a fixed stand in the same way as described at the previous project.

New Design DIY Arduino Based Mini CNC Machine [3]

This project uses another approach of placing the axes.

In this project the X and Y axes are both placed horizontally, which means one stepper motor moves the entire carcass of the other. As for the paper, the entire structure is placed on top of the sheet of paper we desire to draw on.

This is the approach this project will follow.

Design

Hardware

First, we list the required components then we describe the role of each, separately:

- 2x scrap DVD writers (or stepper motors directly)
- 1x micro servo motor
- 2x L293D motor drivers
- 1x Arduino Nano

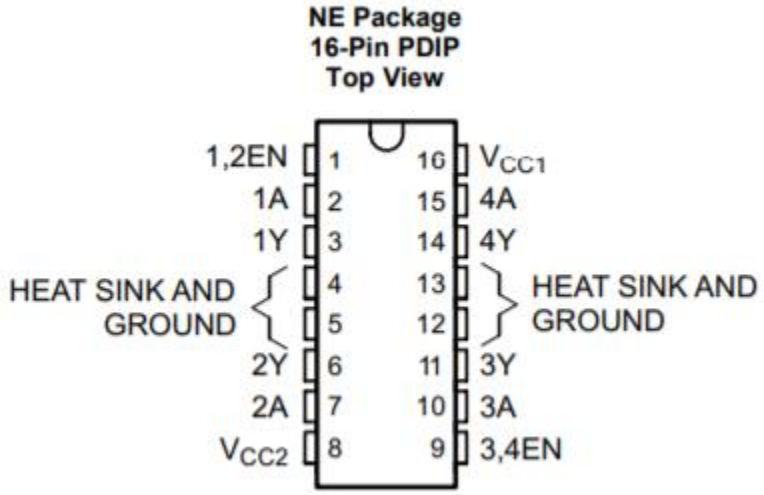
From the DVD writers we need the two stepper motors, and for this I need to disassemble the two cases until I am left with the minimal frame containing the two stepper motors. These two will make the movement along X and Y axes. I have two different DVD writer cases and the rotating bar of the two motors have a slightly different step. The calibration of their movements will be solved from software.

The micro servo motor's role is simply to hold up the pen whenever we are not drawing and to release it when it needs to touch the paper.

We need the two L293D motors to be able to drive the stepper motors in both directions. The L293D can provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. It is designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive supply applications.

The Arduino Nano is the brain of the system, we'll connect the components through it and program it.

To be able to draw the circuit, we examine the pins of the two motor drivers [4]:

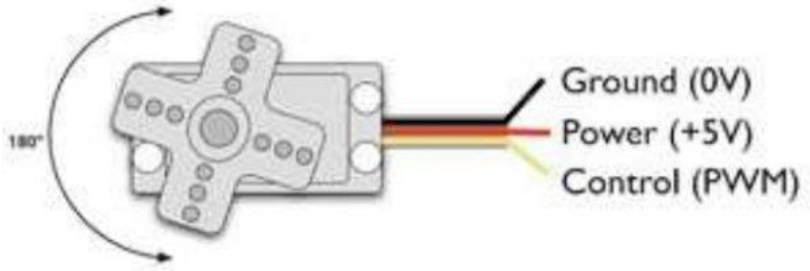


The meaning and the functionality of the pins is shown in the next table:

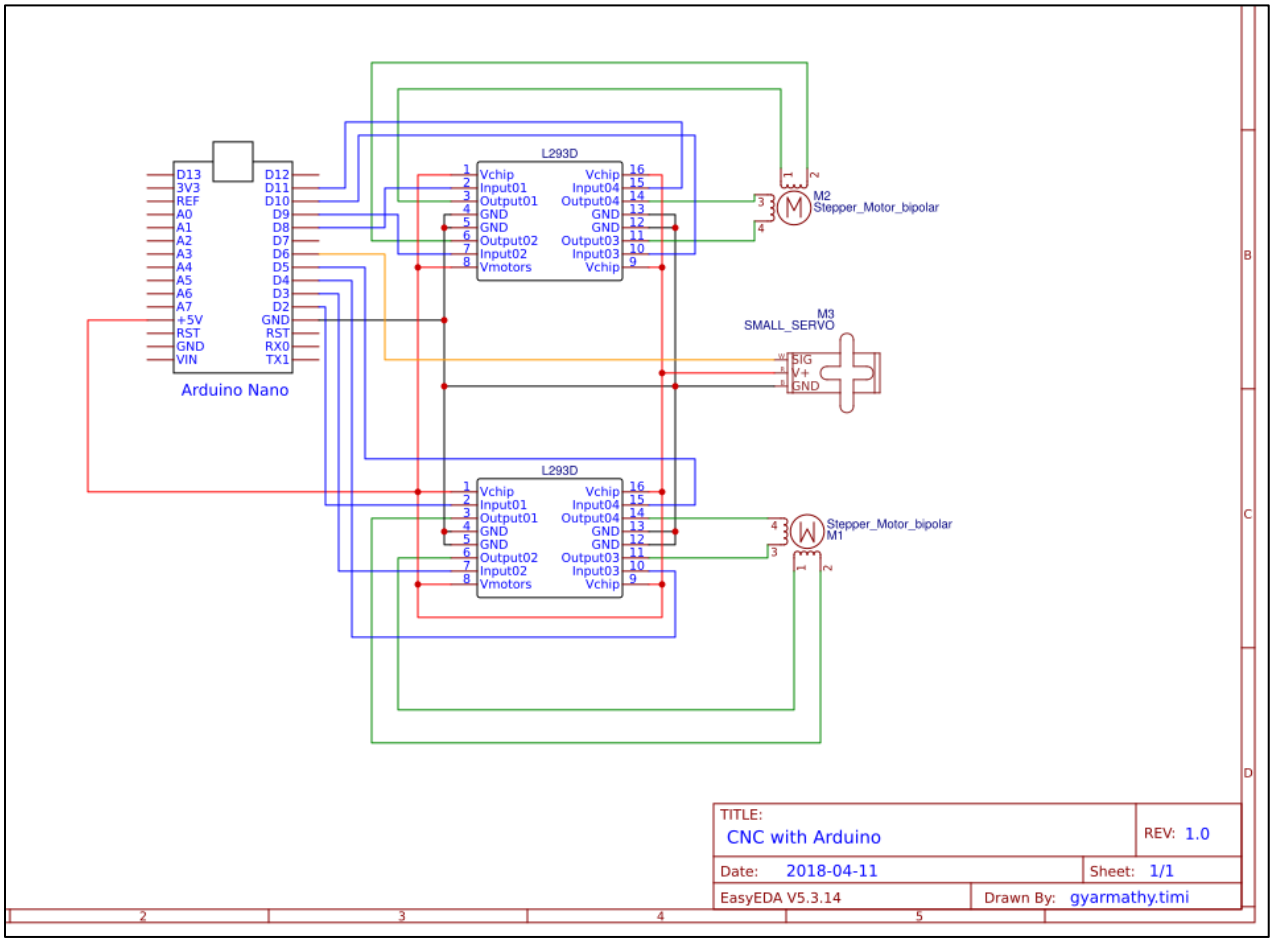
PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2 EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4 EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	-	Device ground and heat sink pin
VCC1	16	-	5V supply for internal logic translation
VCC2	8	-	Power VCC for drivers 4.5V to 36 V

From here it is straightforward that the output pins will be connected two by two to the stepper motor coils and the input pins to the digital output pins of the Arduino Nano.

The servo will be also driven by one of the digital output pin signals. Its wiring looks like the following:

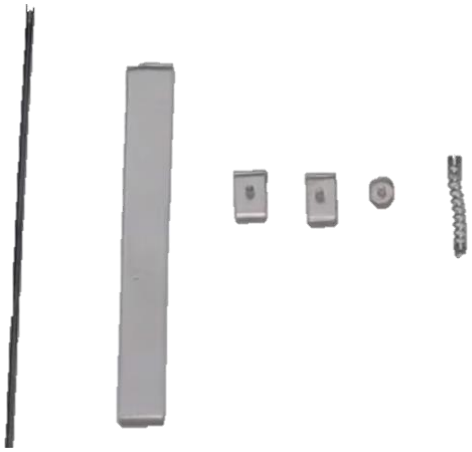


The full circuit wiring is shown in the next figure:



After this wiring is set up, we need to arrange our system to be able to draw along the X and Y axes. I will use the layout presented in the third bibliographic study, which installs the two motor cases on top of each other. To make it stand, we install four long screw on the corners of the lower frame to make it stable.

For the holder of the pen I will cut out from a harder plastic a spring based mechanism which holds the pen, has the servomotor glued on it and altogether it is glued on the system. To build this, we can follow an easy YouTube tutorial [5] which works on the same project idea.



Parts required for the pen holder can be seen on the image on the right. We need a small and rigid metal wire, and some cutout pieces from a strong plastic, as seen on the image. We also need a spring, and one from a ballpoint pen would suffice.

The mechanism would look like the following after assembly:

The pen needs to be tied to the metal bar part (on the right of the image), and the servo on the side of the flat plastic (the top part on the image).

My try on this looks like the following:

The servomotor holds up the pen and thus the spring is compressed. When the servo moves it releases the spring and thus the pen hits the paper. This is useful because if we realize the system inversely then we wouldn't know how far the paper is and we might force the pen by the servo. This way, with a spring strong enough, the pen should be held on the paper just right.

Software

The sketches to be drawn will be saved in a G-code format, thus we must stream G-code to the system, then process and transform them into physical coordinates.

To have an insight, this is how a fragment of a G-code file looks like:

```
(Polyline consisting of 2 segments.)
G1 X-6.72 Y5.81 F3500.00
M300 S30.00 (pen down)
G4 P150 (wait 150ms)
G1 X-6.72 Y6.95 F3500.00
G1 X-5.73 Y6.95 F3500.00
G1 X-5.73 Y5.81 F3500.00
G1 X-6.72 Y5.81 F3500.00
M300 S50.00 (pen up)
G4 P150 (wait 150ms)
```

These kinds of files are generated from a program such as Inkscape. From a list of G-code command meanings [6] we can see what commands interest us and what their meaning is:

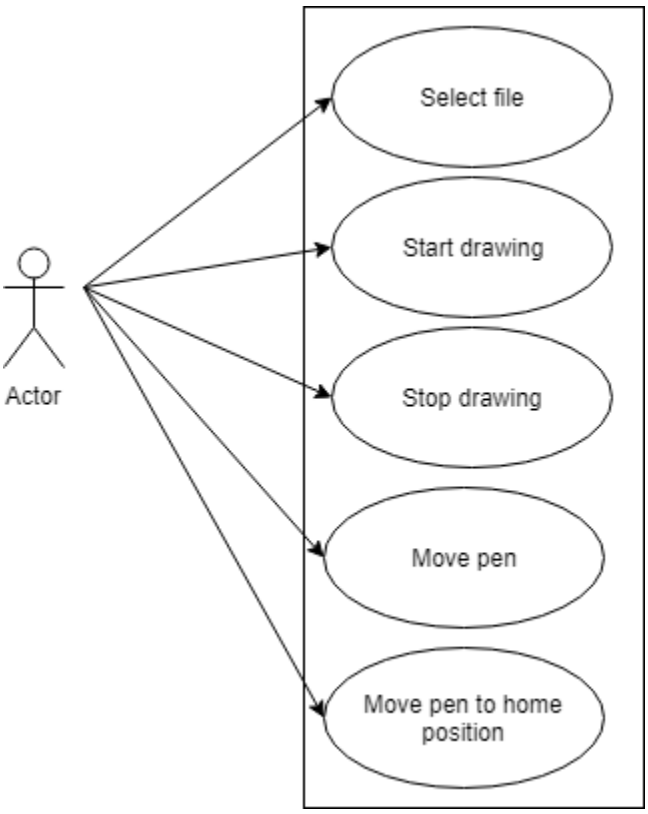
CODE	COMMAND	FORMAT	PURPOSE
G4	Dwell Time	G4/d	Specifies a programmed delay during a drill cycle.
G1	Linear Cutting Move	G1XxYyZz	Moves one or more of the axes along a straight line, at the cutting speed, to a specified location.
G90	Absolute Mode	G90	Switch to moving to the coordinates specified.
G91	Relative Mode	G91	Switch to moving with a distance dx and dy specified, relatively to the current position
M300	Pen up/down	M300 SN0.00	Tells the servomotor to let the pen down or lift it up
M114	Report position	M114	Write to the serial monitor the current coordinate positions

According to the Gcode scripter we use, we will add commands and/or modify the existing ones and write the interpreter accordingly.

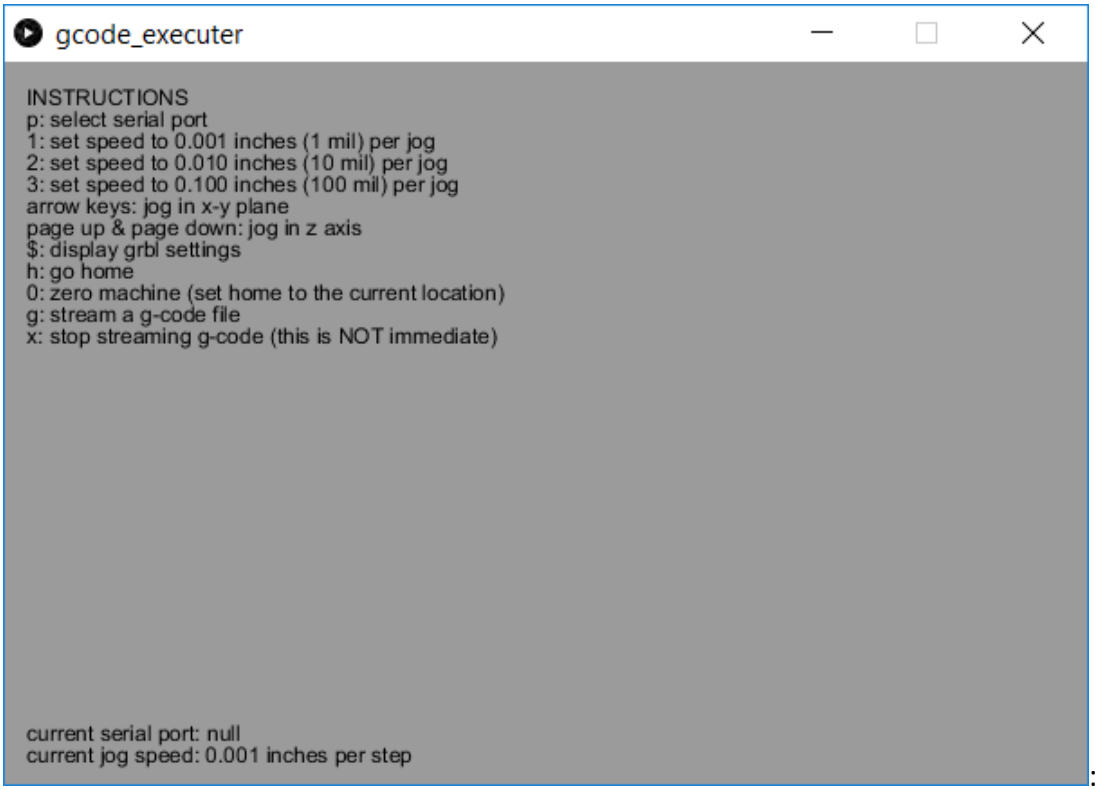
There will be two steps in making the project work from software.

First, using the Processing IDE, we will write a code to stream one of these files to the code running on the microcontroller. In this we can also implement a graphical user interface which responds to user input like choosing the file to be streamed, or pressing the arrow buttons on the keyboard to move the stepper motors and test their functionality.

A simple use-case diagram from the user point of view would look like the following:



The user interface would be a simple one, telling the user the information to which button to press for the desired operation. It would be similar to the following image.



Second, using the Arduino IDE, we need to write a code which processes coordinates and drives the 3 motors accordingly. This will basically convert all the commands it receives on the serial port to physical movement of the system by sending signals on the digital output pins accordingly.

According to the commands we identified, the algorithm would look like the following:

```
initialize StepperX;
initialize StepperY;
initialize Servo;
move StepperX to 0;
move StepperY to 0;
function ProcessIncomingLine(line)
{
Case of line starts with:
      G1 then
            take first argument as X;
            take second argument as Y;
            move StepperX to X;
            move StepperY to Y;
      G4 then
            take argument as Time;
            wait for Time;
      M300 then
            take argument as Command;
            if Command is S30 then
                  move Servo in position 'Down';
            else if Command is S50 then
                  move Servo in position 'Up';
}
```

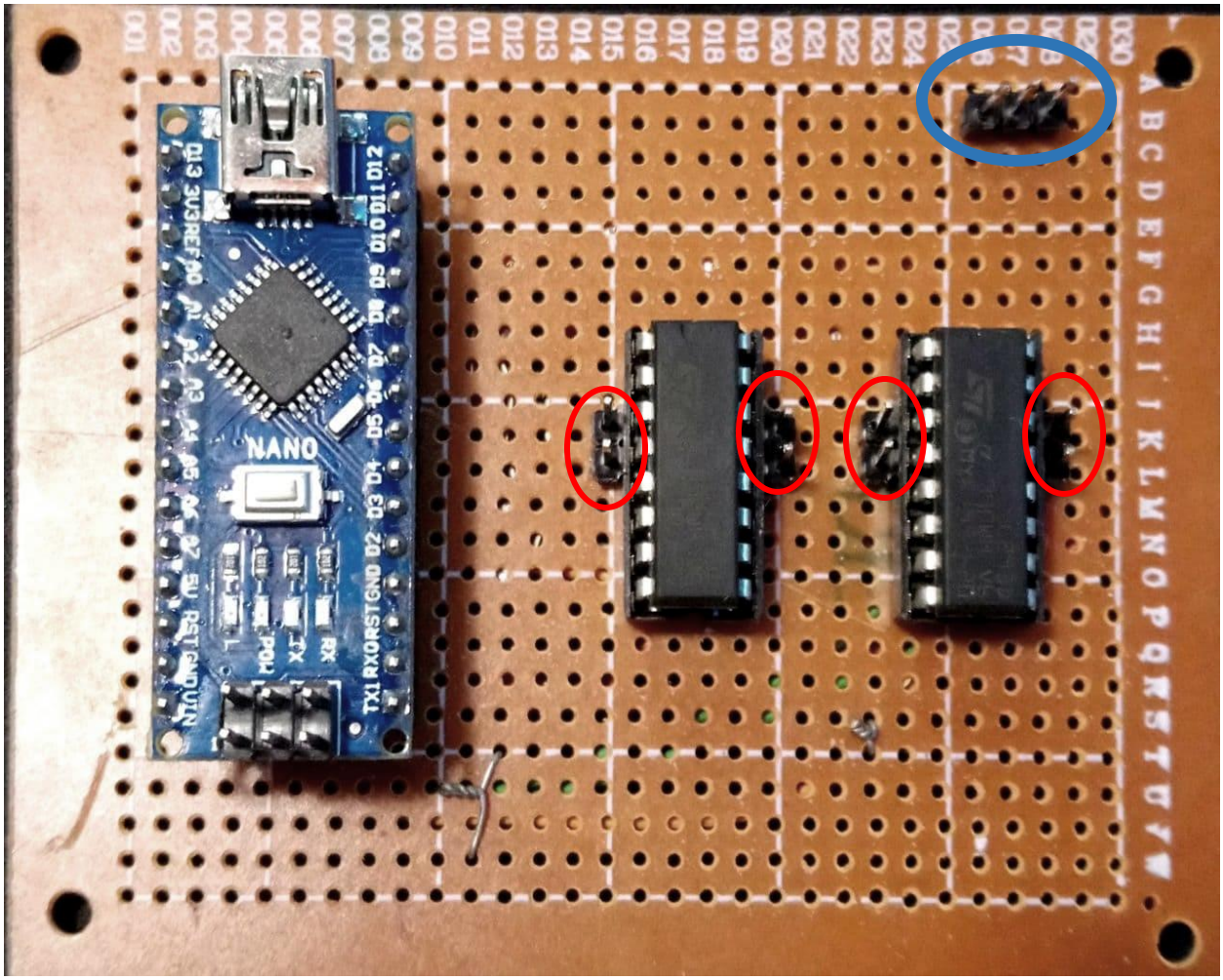
Startup

To make the system work, we need to put a sheet of blank paper under the system and plug it into the computer with a USB cable. We need to program the Arduino through the IDE with the program we will write to drive the motors. We need to open the Processing application and run our interface code. We need to select a .gcode file, then select start. This should start the streaming to the machine and the pen should move accordingly and draw what we desired to draw.

Implementation

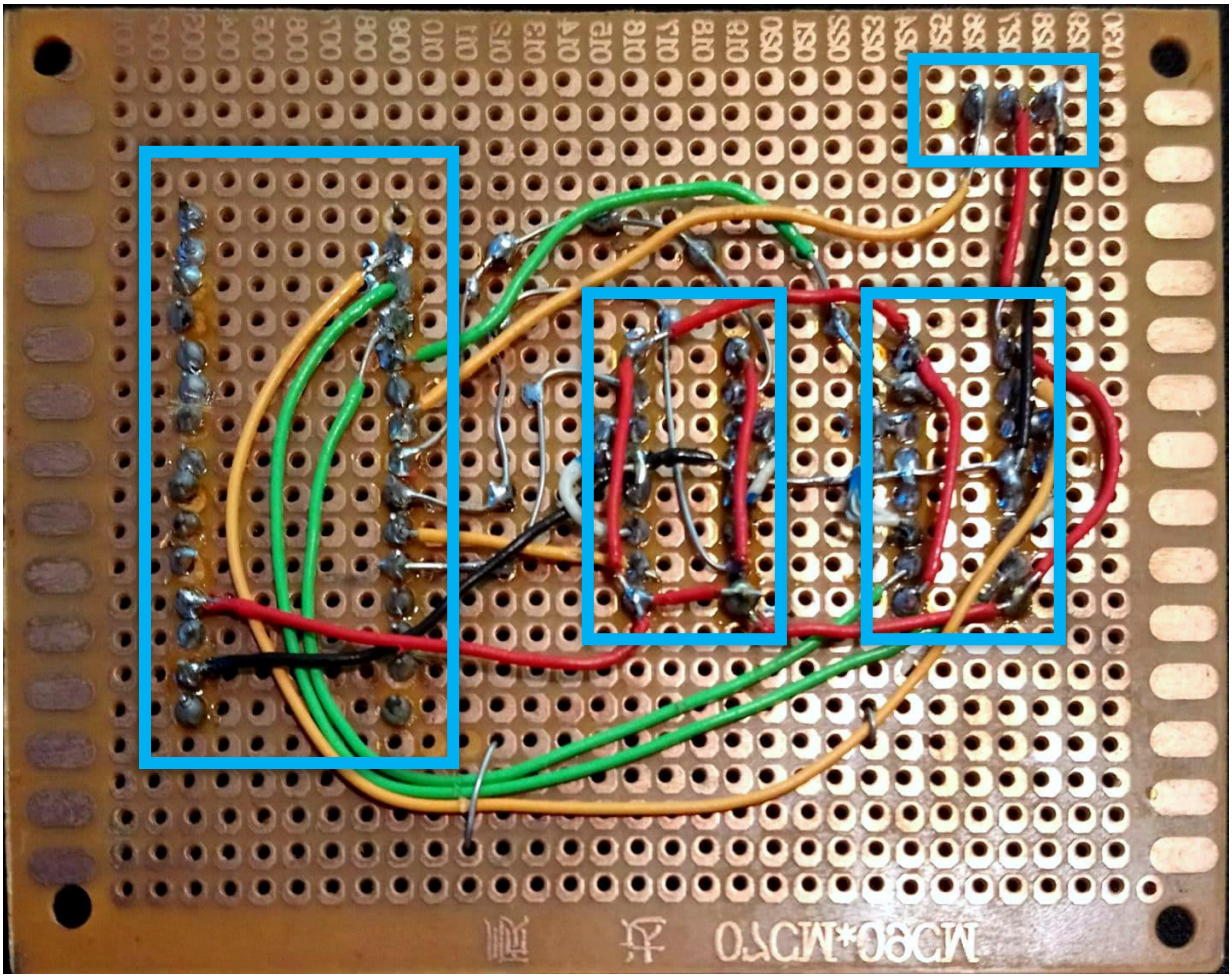
Hardware

On the hardware part we first start by realizing the circuit. I am using a small PCB (Printed Circuit Board) and solder the pieces accordingly. First, I solder the beds for the Arduino Nano and the two ICs (Integrated Circuits, referring to the two motor drivers). To be able to connect the motors, I solder some output pins on the PCB. The layout looks like the following:



The servomotor will be connected to the pins in the upper right corner of the image, circled with blue. The pins circled with red mark the places where the coils of the two stepper motors will be connected, pair by pair.

After soldering, the back of the PCB can be seen on the next image. To make sure that no two wires which should not be connected touch each other, we need to verify each connection with a multimeter to avoid hardware bugs. If everything is connected to where it should, there should be no hazard of burning out the drivers or the microcontroller.



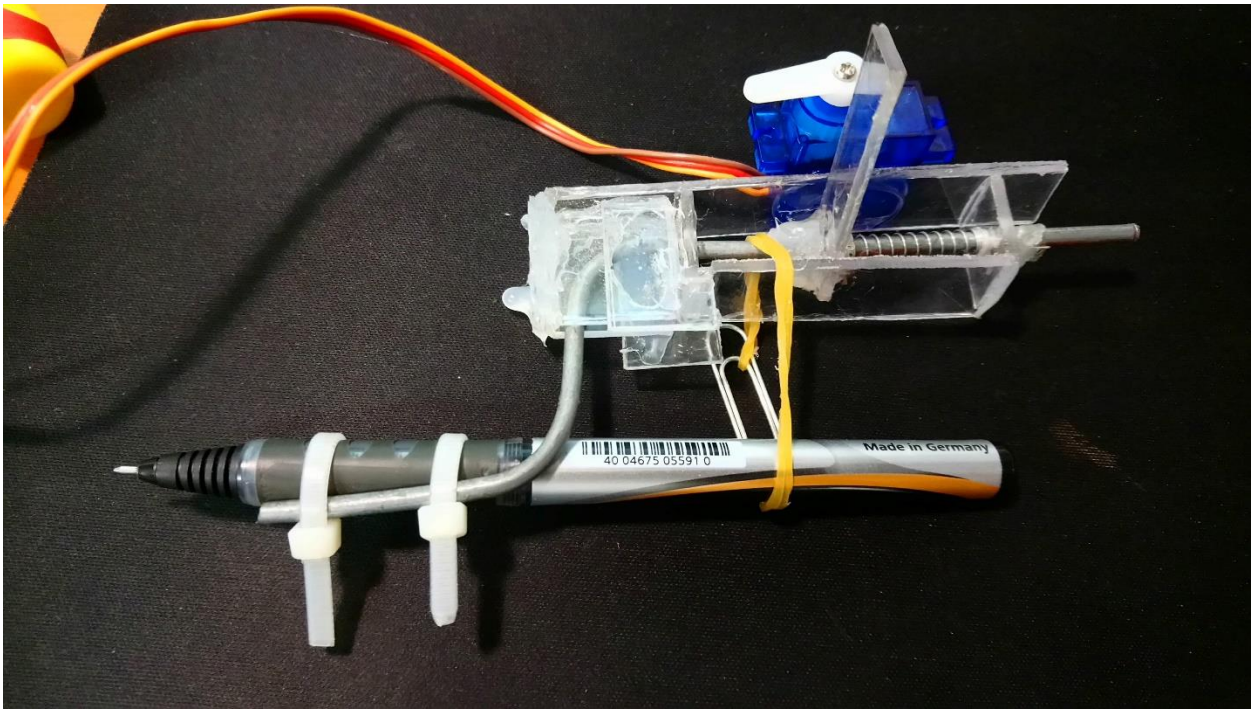
This image is in fact the mirroring of what an observer could see on the actual system, but this way, the parts that overlay this structure can be illustrated: as organized in the previous picture, on the left we have the Arduino, near it to the right the two ICs, and in the top right corner we have the three pins to which the servo will be connected.

For the pen holder mechanism, I cut out the pieces from a 2.5 mm wide plexiglass, glued them together with a glue gun and used a piece of galvanized wire for the pen holder axis. This was adequate, because it is rigid enough not to bend when pressure is applied, and that is because it is partly made of steel. The spring is from a usual ballpoint pen.

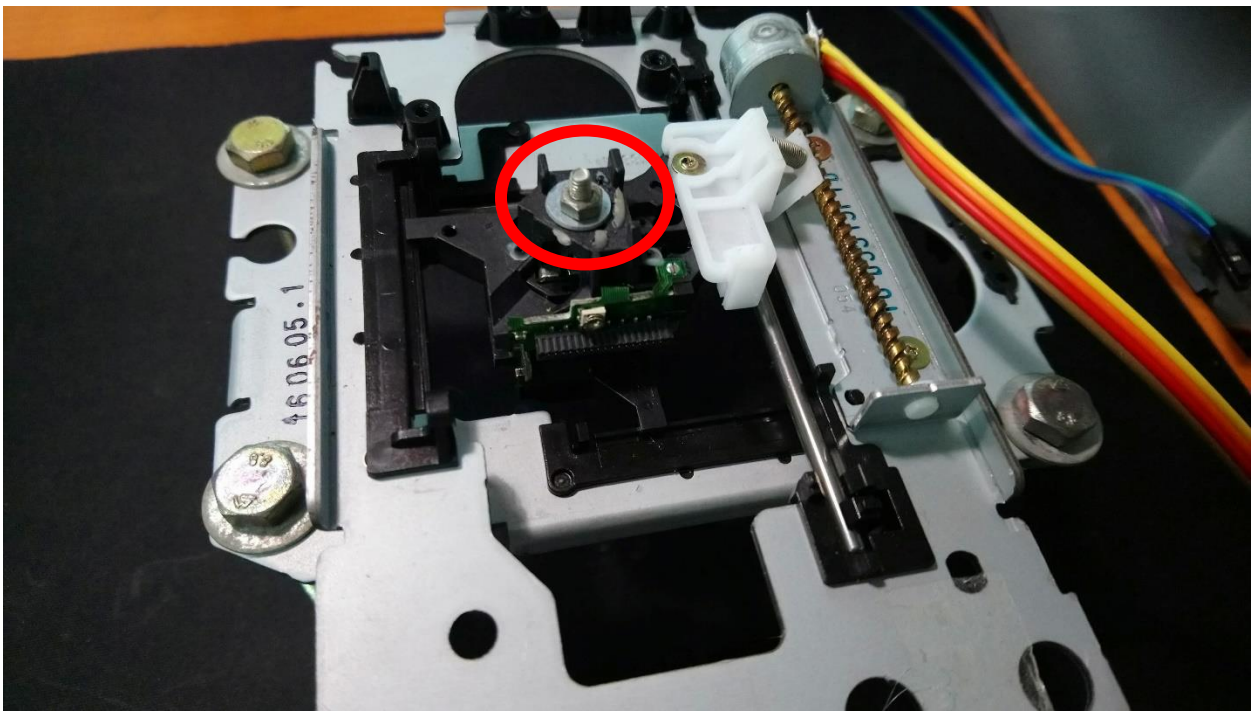
The pen itself to draw is chosen carefully, as the mechanism is not able to apply a huge amount of pressure to draw on the paper. Considering this, the pen must be able to draw as it touches the paper and this requires a liquid pen paste. I chose a black ink rollerball pen after carefully testing out more models.

The servomotor is glued to the side of this mechanism and even though in the picture its arm is facing left (by the orientation of the picture), in fact it should face downwards (by the orientation of the picture) and hold the spring tense by that piece of plexiglass glued to the bottom of the spring.

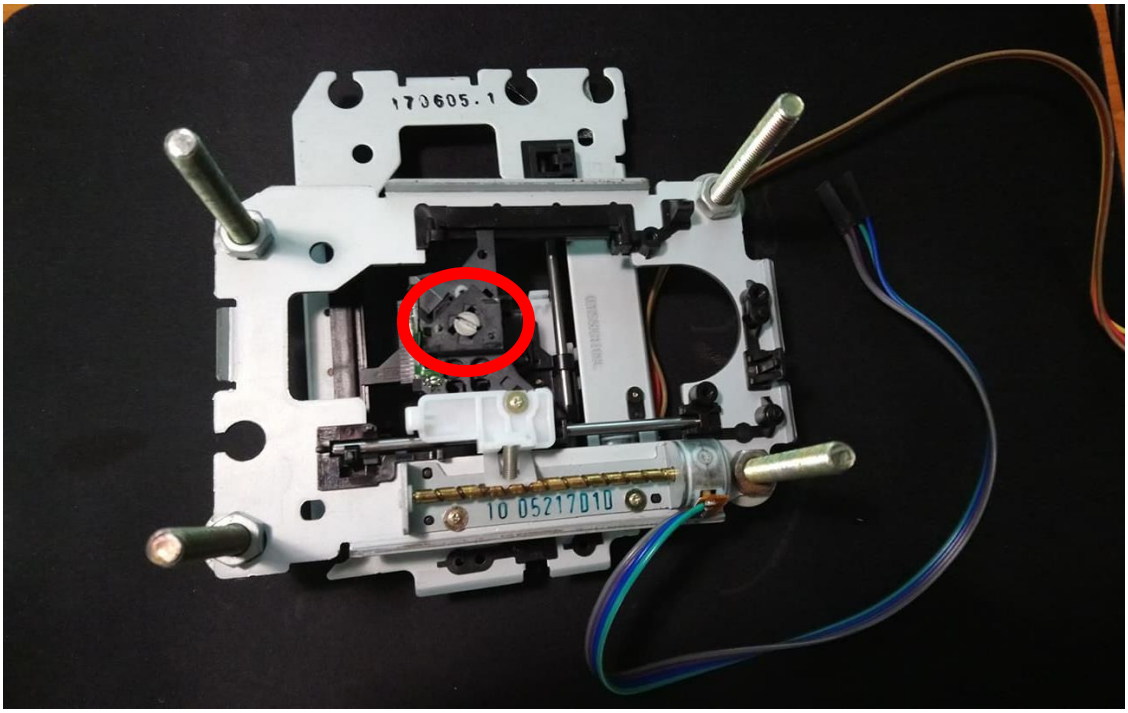
The pen holder mechanism all built up can be seen in the next image:



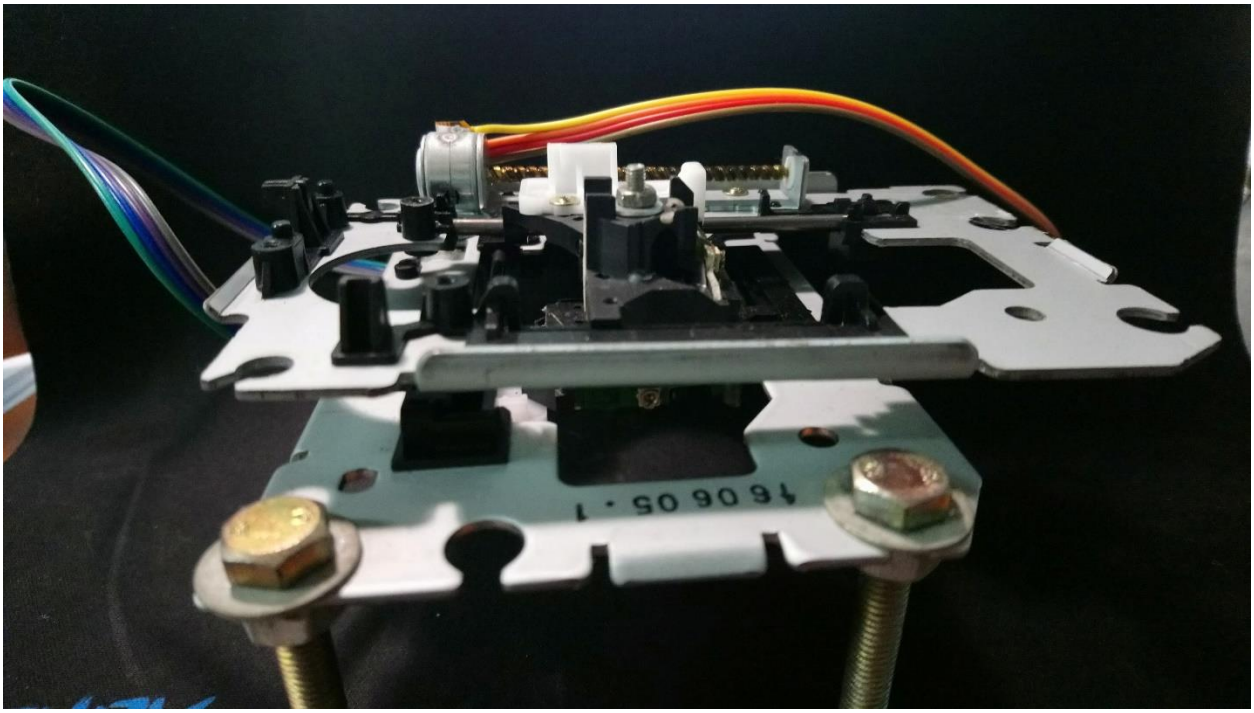
For the drawing axes, I took apart two DVD drives until only the case of the stepper motors were left. These all had in the middle a moving part connected, which had the laser eye in the center. I took these “eyes” out and thus fastened the two cases together with a screw by these holes that were left in the center.



One of the cases is facing upwards and the other one is facing downwards. In the one facing downwards I inserted long screws at the corners, which became the legs of the stand.

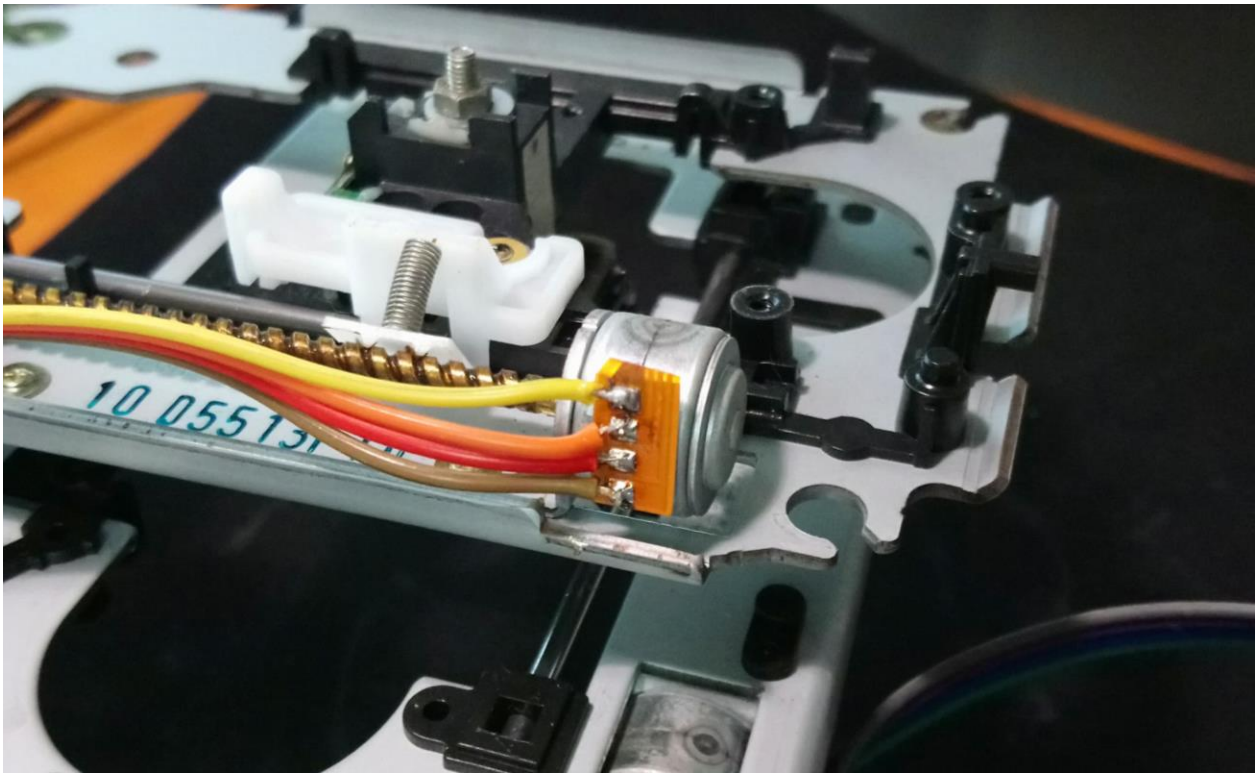


The red circles in the two previous images highlight the screw which holds together the mechanism.



Here we can see the mechanism standing. To draw, the stepper motor at the bottom will move the entire case of the one on the top, while the pen stand will be glued to the stand on the top, and will move together with both axes.

A stepper motor has four inputs, and these can be found on the side of their cases. After cutting off the old connections, I soldered four-four jumper wires to these so I can connect them to the pins on the PCB.



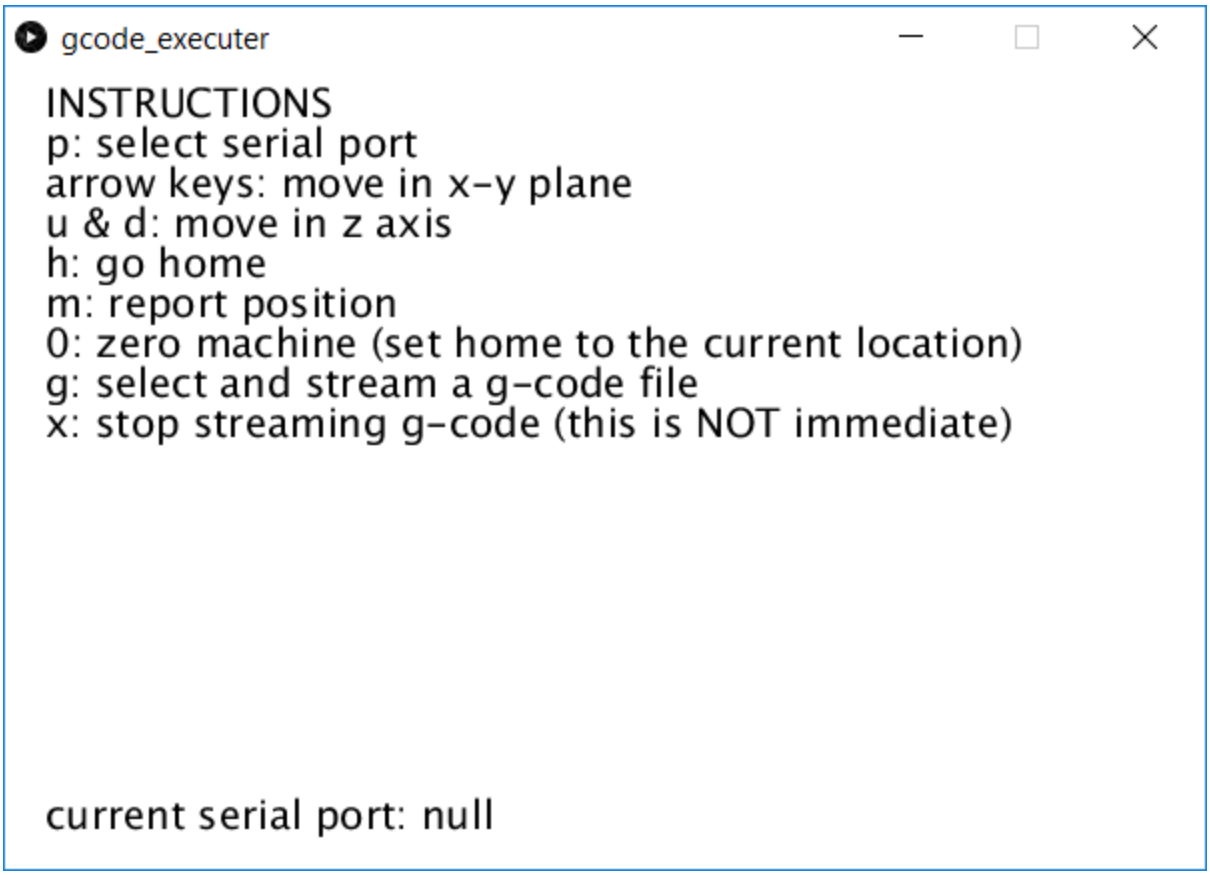
The wires near each other by pairs are the two-two ends of the two coils in the stepper motor. We need to take this into consideration when connecting them to the drivers.

Software

In order to make this project work, we need two files, one which programs the Arduino and transforms the commands it receives to physical movements, and one which streams the .gcode file lines one by one (or other commands) to the Arduino by the user choices.

Processing file – User Interface

The user interface is written via Processing [7], a flexible software sketchbook and a language for learning how to code within the context of the visual arts. In Java language, we create a simple user interface which lists available commands to the user and corresponding keys to be pressed. The user interface looks like the following:



It's main background logic consists of sending the corresponding G-code commands via the open Arduino port to the board.

For example, by pressing the arrow keys, the user can move the pen in the x-y directions. This is done by sending a command which tells the Arduino to:

1. Switch to relative movement, send: "G91\n"
2. Move one step in the corresponding direction, send: "G00 X+" + speed + " Y0.00\n"

Where *speed* specifies a real number to tell the board how much it should move.

When the user presses the key *g* on the keyboard, a user prompt appears where the user should select the .gcode file to stream. In a later section it'll be described how to create such a file. As soon as the file is selected at this step, the program starts streaming it to the boards via the selected port. To interrupt this process, the user can press *x* to stop streaming the file.

The following table presents all the available commands the user can select:

Option	Key	Description
Select port	P	A drop list appears from which the user should select the serial port corresponding to the Arduino
Left arrow	←	Move one step left (-x direction)

Up arrow	↑	Move one step up (+y direction)
Right arrow	→	Move one step right (+x direction)
Down arrow	↓	Move one step down (-y direction)
Lift pen	u	Lift up the pen from the paper
Lower pen	d	Lower down the pen onto the paper
Go home	h	Move the system back to the specified (0, 0) location
Report position	m	Print the current coordinates to the serial output
Set zero location	0	Set the current position as the (0, 0) location of the system
Stream file	g	Select a file from the pop-up explorer window and start streaming it
Stop streaming	x	Interrupt the current file streaming

The user can see the success of some commands, as well as the output of the "report position" command in the console window of the Processing software.

Arduino file – Board program

This file is used to program the board, and transform the commands received via the serial port into signals sent through the digital pins of the board to the electronic components. In this file, the main logic consists of parsing the line. We take the incoming line, decompose it into parts, verify what characters we received and switch to the corresponding command option.

There are two modes, absolute and relative, and if we are in absolute mode, we move from the current position to the new specified position. In relative mode, we move from the current position the number of specified steps in the command.

To move the motors, we use the Arduino libraries <Servo.h> and <Stepper.h> which give us the prebuilt functions to command the motors via the connected pins.

For debugging purposes, if the user sets the value of the variable *verbose* to true, the ongoing actions will be printed to the serial monitor.

A detailed description of how the movement happens:

1. Let's suppose the current position of the pen, as saved by the software in two variables, *Xpos* and *Ypos* is saved, for example, (0, 0)
2. There is an incoming line through the serial port which is recorded up until the first newline character
3. The main decision function is called on the line, suppose it was "*G1 X20.00 Y30.00"*
4. The switching function takes the line's first characters, recognizes it is a G1 command for movement and processes the line further until it gets the two values, 20.00 for X and 30.00 for Y
5. The *drawLine* function is called on the read coordinates

6. This checks if the newly received coordinates are within bounds of drawing and converts it to values to be sent to the stepper by:
   x1 = (int)(x1*StepsPerMillimeterX);
   y1 = (int)(y1*StepsPerMillimeterY);
7. Movement distance is calculated by subtracting the current position from the new position
8. Step by step, until the new position is reached, a command is sent to the steppers via prebuilt functions, which looks like the following:
   myStepperX.step(sx);
   myStepperY.step(sy);
   Where myStepperX and myStepperY are the two declared stepper motors in X and Y directions, and sx and sy are two real numbers to specify the size of a single step

As for the servomotor, its movement is determined by two functions, *penUp()* and *penDown()* which call functions on the sefined servomotor to move to a certain angle. In our case the angles are defined as:

```
const int penZUp = 80;
const int penZDown = 40;
```

This code has to be uploaded on the board via the Arduino software [8] if everything goes well and the board remains plugged in, the user interface program should be able to connect via opening the serial port the Arduino communicates on.

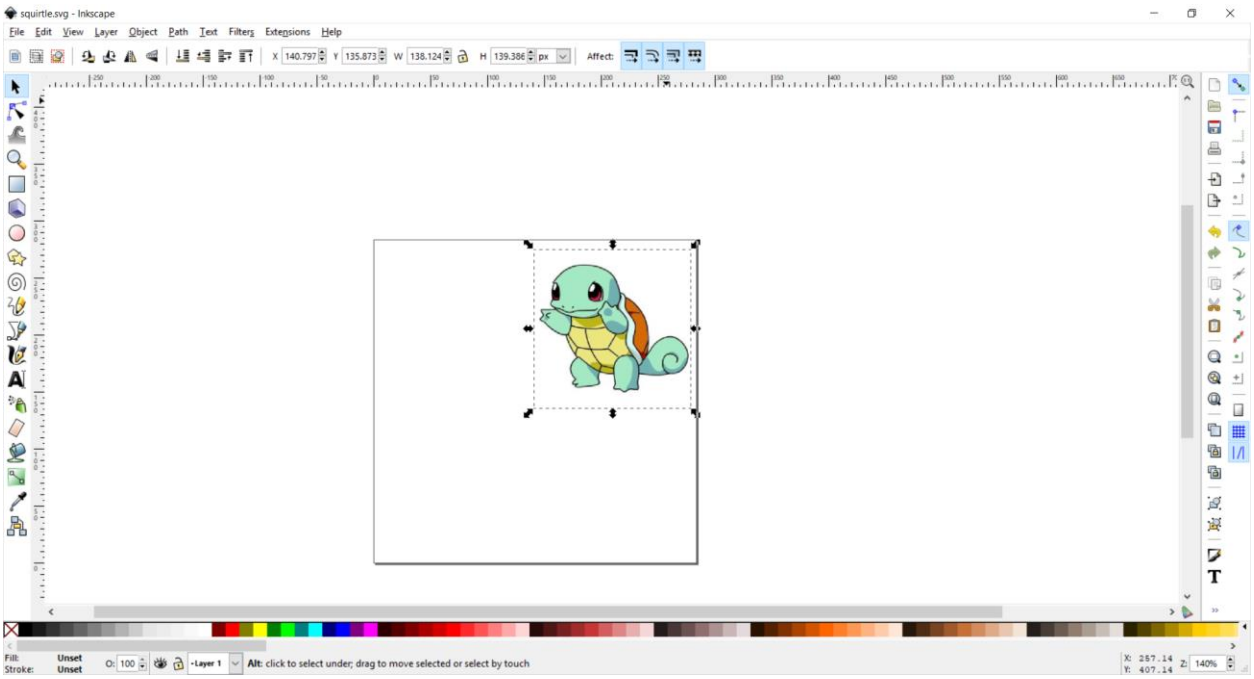
Creating G-code files

To have self-made G-code files we need the software *Inkscape* [9], which is a professional vector graphics editor for Windows, Mac OS X and Linux. It's free and open source. It doesn't support saving to .gcode format but it is really easy to add an extension to it. This extension is *MakerBot Unicorn G-Code Output for Inkscape* [10], which unfortunately was last successfully tested on Inkscape 48.5 (the current version being 92), but it works really well with it, and the features implemented in the new version do not concern this project.

After downloading Inkscape 48.5 and the folder containing the scripts to save as .gcode file, we copy the scripts into Inkscape's *extensions* folder in its installation folder. This way when saving the project, we should have a *Save as->MakerBot Unicorn G-Code (*.gcode)* option.

To prepare an image for saving as .gcode, first open a document and resize it to 80x80 mm.

Drag an image onto the canvas and place it into the top right quadrant of it, as in the next figure:



To convert it, first choose the option *Trace bitmap…* from the menu option *Path* and proceed with it. Then on the created bitmap (delete the colored image) choose *Path->Object to path.* This should ensure that you can save it as .gcode. You can also choose to simplify it a bit so the machine would have to work with less points.

The bug I ran into with this feature was that the .svg format which was parsed by the .gcode scripter saved the points in a dynamic, relative offset. To fix this, in Inkscape preferences I unchecked the option *Allow relative coordinates.*

After this the project should be saved as .gcode and fed to the running system.

Testing

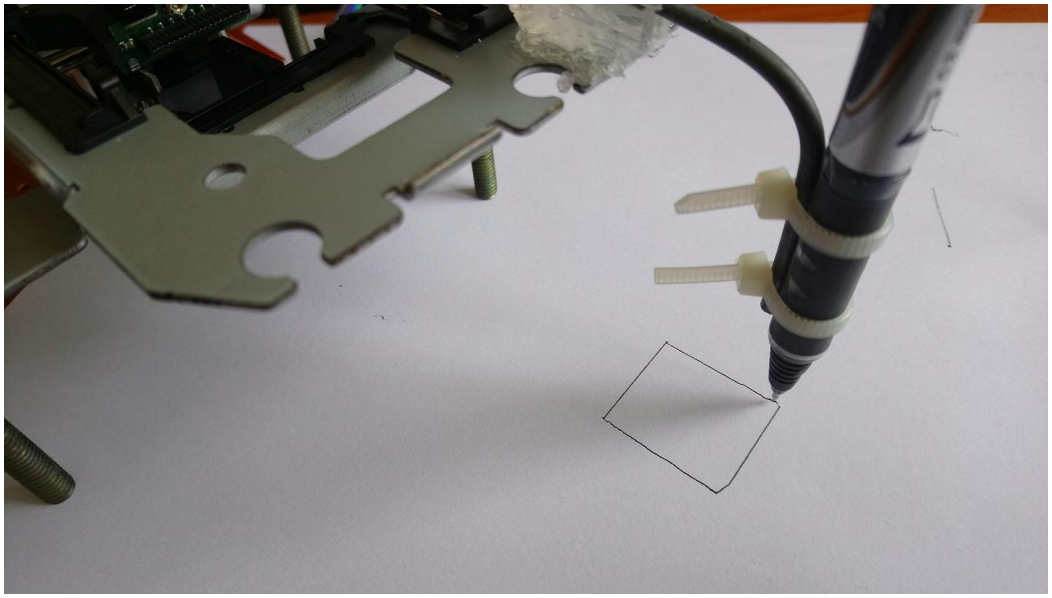
First I had to calibrate the system in some way so first I set the *stepsPerMilimeter* value to 1. By using the feature implemented by the arrow keys, I moved 10 steps then measured the drawn line and calculated the actual value correspondingly, which turned out to be ~5 steps per millimeter.

Then I wrote a .gcode file manually to draw a square between coordinates (0, 0) and (20, 20) and the system seemed to handle the situation nicely.

However, in some cases when the pen was lifted and I was trying to make the machine draw an oblique line, the axes of the stepper motor skipped over some steps and gave a serious problem. I considered the problem should be solved by greasing the axes more.

The other problem I ran into was the distance between the paper and the pen. If it pressed too much onto the paper, it became imprecise.

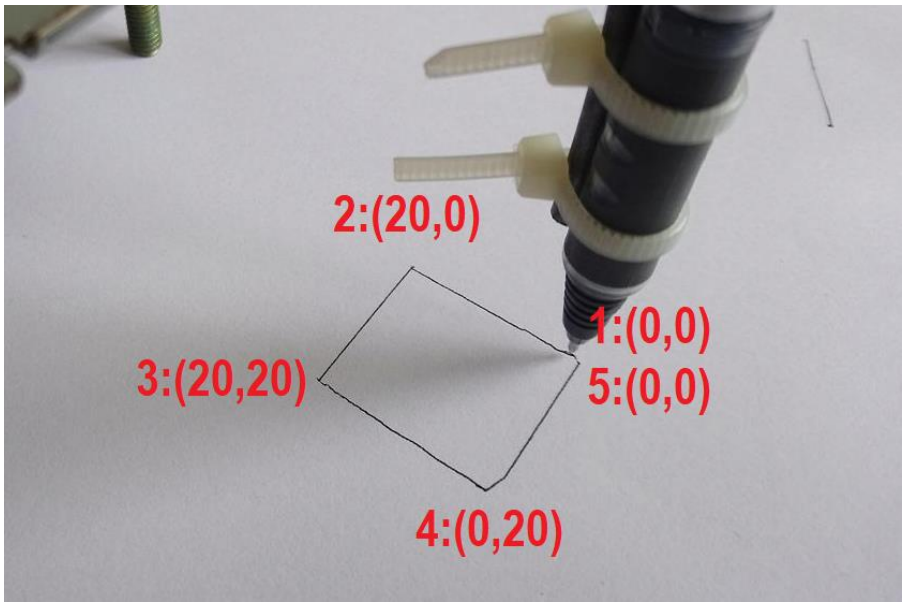
We can see on the next picture how the square was drawn:



The coordinates of this square were given to the machine as:

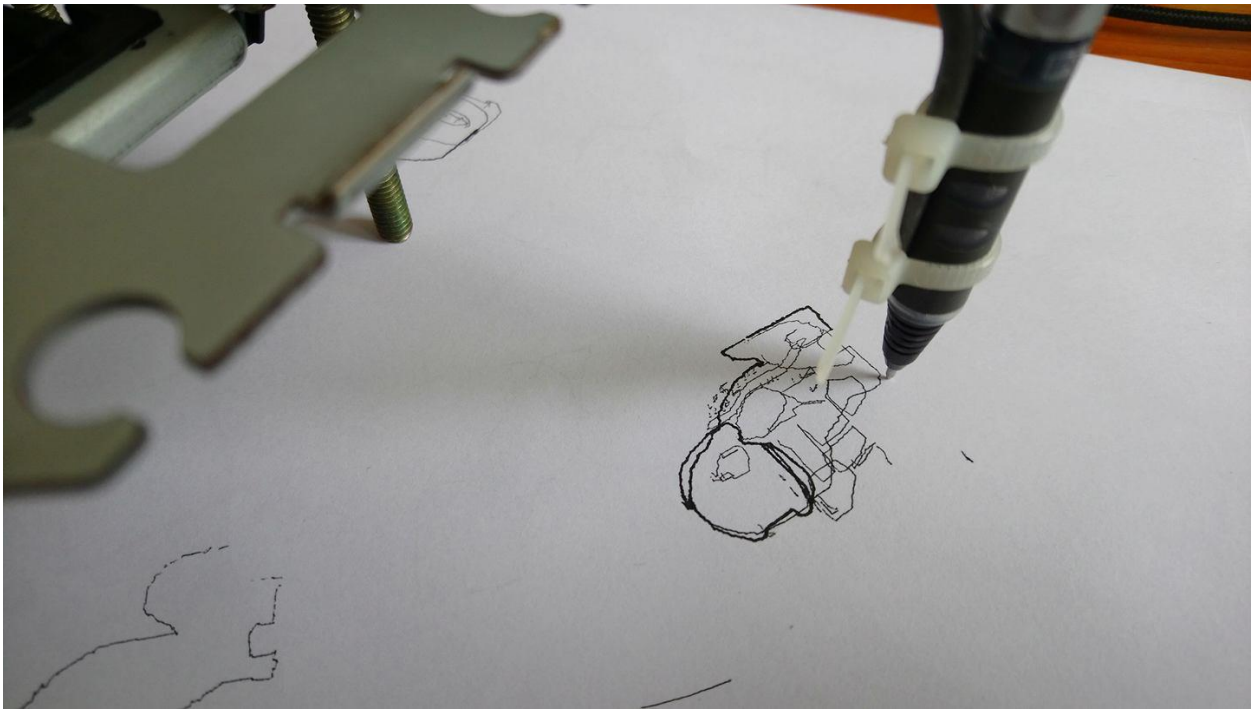
- (start from X:0 Y:0)
- X:20 Y:0
- X:20 Y:20
- X:0 Y:20
- X:0 Y:0

To understand the drawing process, the next image labels the corners of the square by order in which they were reached and their placement (coordinates):



The slight anomality on the sides of the square comes from the way the pen touches the paper. To have a perfect square drawn, only the tip should touch the paper when drawing. I couldn't reach this perfectly because the table surface itself wasn't perfectly flat.

When testing with scripted gcode, I had a problem of saving the file such as it contained only positive coordinates. Apparently the newer version of Inkscape changed the .svg format so the script didn't work properly. With Inkscape 0.48 (the current version being 0.92) I managed to script the squirt image. When fed into the machine, it cut off (pruned) its sides which meant my scaling wasn't right. After a bit more testing this is the resulting image that was drawn:



As we can see it resembles, but it is far from perfect. The factors which conclude to the mistakes in drawing are:

- The two stepper motors I use have different scaled axes
- The motor drivers are powered through Arduino, they might not get enough power
- The X axis sadly skips some steps because of the high scaling it has and it slips
- The gcode scripter is not reliable, the coordinates it produces may not be accurate

Overall the results are fairly satisfying.

Further developments

This project has a wide variety of further development options such as to comply with the newer versions of Inkscape (but that part relies on rewriting the scrips, not the drawing files themselves), or to power the motor drivers from the computer not through the board.

References

[1] "Mini Cnc Machine Arduino Based & Adafruit Driver Motor L293d V1 & 2*Mini Stepper Cd/Dvd Player," [Online]. Available: http://www.instructables.com/id/Mini-CNC-Machine-Arduino-Based-Adafruit-Driver-Mot/.

[2] "Sketch It (CNC plotter)," [Online]. Available: https://create.arduino.cc/projecthub/Yogeshmodi/sketch-it-cnc-plotter-95019d.

[3] "New Design DIY Arduino Based Mini CNC Machine," [Online]. Available: http://electricdiylab.com/new-design-diy-arduino-based-mini-cnc-machine/.

[4] "L293x Quadruple Half-H Drivers," Texas Instruments, [Online]. Available: http://www.ti.com/lit/ds/symlink/l293.pdf.

[5] "How to Make a CNC machine at home," Tapendra Mandal, [Online]. Available: https://www.youtube.com/watch?v=2VFOU-WUQIY.

[6] "Summary Of GCODE Commands By Category (HTT0196)," [Online]. Available: http://www.science.smith.edu/cdf/pdf_files/Techno_GCODE%20Commands.pdf.

[7] "Processing," [Online]. Available: https://processing.org/.

[8] "Arduino IDE," [Online]. Available: https://www.arduino.cc/en/Main/Software.

[9] "Inkscape," [Online]. Available: https://inkscape.org/en/.

[10] "MakerBot Unicorn G-Code Output for Inkscape," [Online]. Available: https://github.com/martymcguire/inkscape-unicorn.