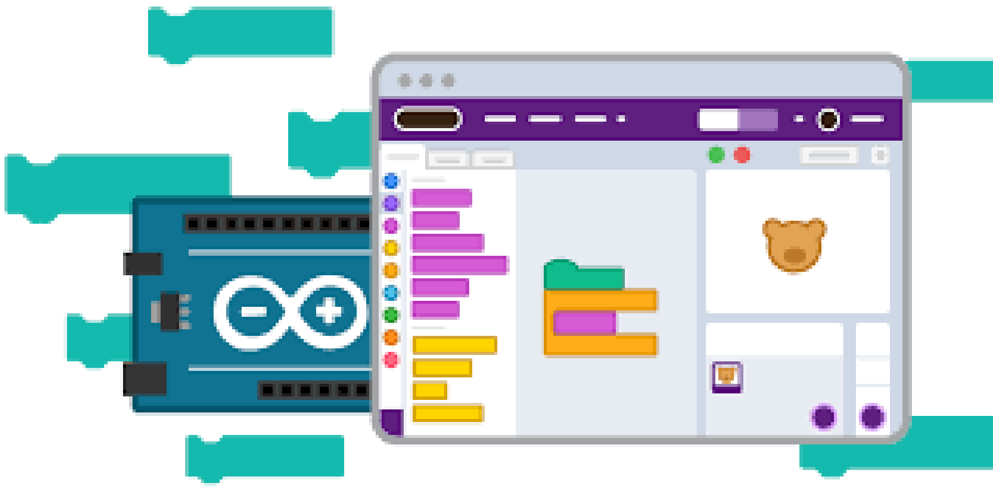
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Drawing Shapes with Arduino



Student: Valcauan Laura-Maria
Group: 30432
Teacher: Hangan Anca
Academic Year: 2022-2023

Table of contents:

1. Introduction

1.1. Context

The goal of this project is to design, implement and test different shapes using a robotic arm which will be controlled by Arduino. Basically the program has the following logic: a given shape will be translated in g code, sent to the Arduino code and then the arm will draw what was requested.

The device can be used by any one because it has a friendly and easy to use graphical interface in which the image/ shape is loaded and transmitted to the program.

1.2. Specifications

The device will be implemented in the IDE provided by Arduino and then programmed on an Arduino UNO board; and for the graphical user interface it will use the IDE provided by JetBrains. It will be able to represent internally the shapes in g code representation (presented in the next chapter), convert from and into coordinates for the Arduino to know how and when to draw the shape.

1.3. Objectives

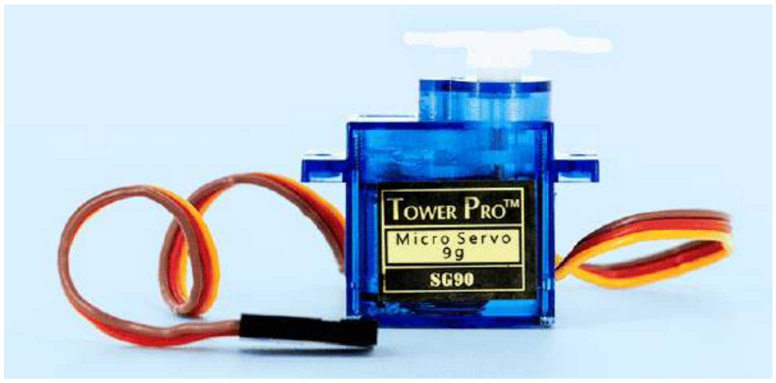
Design and implement a way to encode a shape in g code, to de-code and send the coordinates to the Arduino board so the robotic arm can use its motor and rotary potentiometer and a pen (that it will hold) to draw the shape/ image. For the interface, being the user's first interaction with the program, it is very simple, insert an image or write a command (i.e. draw a line) and press the button “ok” which will bring the program to work.

2. Bibliographic Study

Arduino is **an open-source electronics platform based on easy-to-use hardware and software**. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.
Source: Arduino board. For this project additional accessories for the robotic arm are:

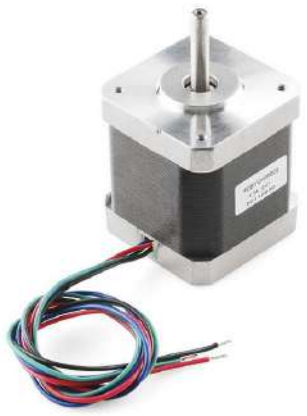
- Micro-servo motor
- 2 x Stepper motor Nema 17 (bipolar)
- Jumper wires
- 12 V Voltage source
- CNC Shield
- 2 x Stepper Driver A4988
- 2 x MGN15H Linear Rail for each axis
- 2 x Double Tooth Pulley
- GT2 Belt
- 2 x Idler Pulley
- M3 and M5 bolts

…and ofcourse one Arduino UNO.



source: https://www.google.com/search?q=micro+servo+motor+arduino&sxsrf=ALiCzsaLipcUqA0vhUdHaTD0Nd5eS9lKzA:1667749938732&source=lnms&tbm=isch&sa=X&ved=2ahUKEwif_a_V9Jn7AhX2_rsIHYTaDyYQ_AUoAXoECAIQAw&biw=1536&bih=664&dpr=1.25#imgrc=EF8ghc8r5gAY8M

Fig1. Micro-servo motor



source: https://www.google.com/search?q=stepper+motor+Nema+17+for+arduino&sxsrf=ALiCzsaVQrfkcA-bz1ivhKKv0fgOU7_sDg:1667749989297&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi8lb7t9Jn7AhUBg_0HHcDdCGwQ_AUoAnoECAEQBA&biw=1536&bih=664&dpr=1.25#imgrc=mA7AgNRuyHxFnM

Fig2. Stepper Motor Nema 17

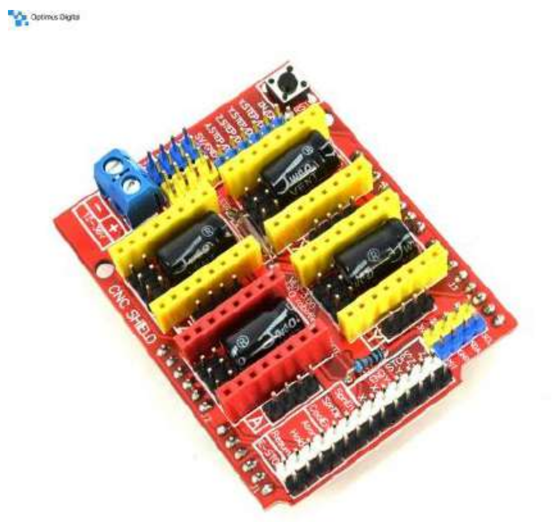
source: https://www.google.com/search?q=jumper+wires+for+arduino&tbm=isch&sxsrf=ALiCzsZdk-HdHMYuqqpUf2AN

nlNGFODUgw:1667750039225&source=lnms&sa=X&ved=0ahUKEwipqqWF9Zn7AhULgf0HHR5BCswQ_AUI1AcoAQ&biw=1536&bih=664&dpr=1.25#imgrc=wlKKC14B4En3EM

Fig4. Jumper Wires

source: https://ro.wikipedia.org/wiki/Arduino

Fig5. Arduino UNO 2560



source: https://www.optimusdigital.ro/en/arduino-shields/467-a4988-cnc-shield-v3-for-arduino.html

Fig6. CNC Shield

source:
https://www.optimusdigital.ro/ro/drivere-de-motoare-pas-cu-pas/866-driver-pentru-motoare-pas-cu-pas-a4988-rosu.html
Fig7. A4988 Stepper Driver

source:
https://www.hta3d.com/en/mgn15c-linear-carriage-for-mgn15-linear-guide
Fig8. MGN15H Linear Rail

source:
https://ardushop.ro/en/3d-printing/380-gt25-pulley-for-5mm-axis.html
Fig9. Double Tooth Pulley

source:
https://ardushop.ro/en/3d-printing/381-gt25-timing-belt-1-meter.html
Fig10. GT2 Belt

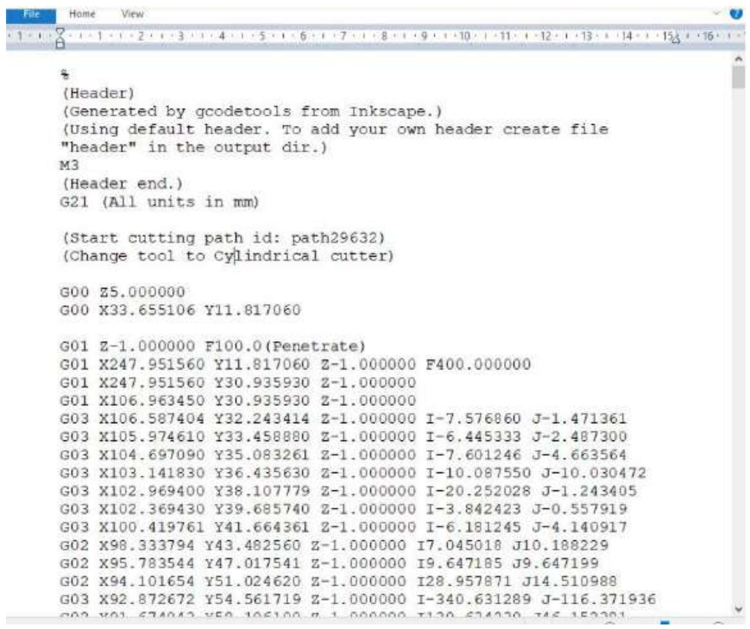
source:
https://ardushop.ro/en/3d-printing/469-idler-pulley-gt2-6mm.html
Fig11. Idler Pulley

source:
https://www.optimusdigital.ro/ro/drivere-de-motoare-pas-cu-pas/866-driver-pentru-motoare-pas-cu-pas-a4988-rosu.html
Fig12. Voltage Source

What is G-code?

G-code is a programming language for CNC (Computer Numerical Control) machines. G-code stands for “Geometric Code”. This language is used to tell a machine what to do or how to do something (in this case, to Arduino). The G-code commands instruct the machine where to move, how fast to move and what path to follow.

How to read a G-code Commands?



```
%
(Header)
(Generated by gcodetools from Inkscape.)
(Using default header. To add your own header create file
"header" in the output dir.)
M3
(Header end.)
G21 (All units in mm)

(Start cutting path id: path29632)
(Change tool to Cylindrical cutter)

G00 Z5.000000
G00 X33.655106 Y11.817060

G01 Z-1.000000 F100.0(Penetrate)
G01 X247.951560 Y11.817060 Z-1.000000 F400.000000
G01 X247.951560 Y30.935930 Z-1.000000
G01 X106.963450 Y30.935930 Z-1.000000
G03 X106.587404 Y32.243414 Z-1.000000 I-7.576860 J-1.471361
G03 X105.974610 Y33.458880 Z-1.000000 I-6.445333 J-2.487300
G03 X104.697090 Y35.083261 Z-1.000000 I-7.601246 J-4.663564
G03 X103.141830 Y36.435630 Z-1.000000 I-10.087550 J-10.030472
G03 X102.969400 Y38.107779 Z-1.000000 I-20.252028 J-1.243405
G03 X102.369430 Y39.685740 Z-1.000000 I-3.842423 J-0.557919
G03 X100.419761 Y41.664361 Z-1.000000 I-6.181245 J-4.140917
G02 X98.333794 Y43.482560 Z-1.000000 I7.045018 J10.188229
G02 X95.783544 Y47.017541 Z-1.000000 I9.647185 J9.647199
G02 X94.101654 Y51.024620 Z-1.000000 I28.957871 J14.510988
G03 X92.872672 Y54.561719 Z-1.000000 I-340.631289 J-116.371936
```

If we take a closer look at the code, we can notice that most of the lines have the same structure. It seems that the “complicated” part of the G-code are all those numbers, which are just Cartesian coordinates.

For example, in this line:

G01 X247.951560 Y11.817060 Z-1.000000 F400.000000

The line has the following structure:

G## X## Y## Z## F##

- ➔ First is the G-code command and in this case that’s the G01 which means “move in a straight line specific position”
- ➔ Declare the position or the coordinates with the X, Y, Z values
- ➔ Lastly, with F value is set the feed rate, or the speed at which the move will be executed.

In conclusion, the line G01 X247.951560 Y11.817060 Z-1.000000 F400 tells the CNC machine to move in a straight line from its current position to the coordinates X247.951560, Y11.817060 and Z-1.00000 with speed of 400 mm/min.

Source: https://howtomechatronics.com/tutorials/g-code-explained-list-of-most-important-g-code-commands/

3. Project Plan and Proposal

3.1. Project Plan

Week	**Tasks**
Week 1 and Week 2 (3.10.2022 - 16.10.2022) - Project Meeting 1	★ Choose theme of project ★ Getting acquainted with chosen topic
Week 3 and Week 4 (17.10.2022 - 30.102022) - Project Meeting 2	★ Writing project introduction and bibliographical research ★ Research on g-code and arduino components
Week 5 and Week 6 (31.10.2022 - 13.11.2022) - Project Meeting 3	★ Project scheduling, weekly plan of tasks ★ Research on algorithms used for the design of the project
Week 7 and Week 8 (14.11.2022 - 27.11.2022) - Project Meeting 4	★ Design of components needed ★ Implementation of the base program: which gives the coordinates for the arm
Week 9 and Week 10 (28.11.2022 - 11.12.2022) - Project Meeting 5	★ Implementation of the robotic arm ★ Implementation of the commands (drawings) that will be executed
Week 11 and Week 12 (12.12.2022 - 25.12.2022) - Project Meeting 6	★ (cont.) Implementation of the commands (drawings) ★ Fixing potential bugs ★ Finalizing documentation
Week 13 and Week 14 (9.01.2023 - 22.01.2023) - Project Meeting 7	★ Presentation of the final project and documentation

3.2. Project Proposal and Analysis

The goal of this project is to design, implement and test different
shapes using a robotic arm which will be controlled by Arduino. Basically the program has the following logic: a given shape will be translated in g code, sent to the Arduino code and then the arm will draw what was requested.

For drawing the basic lines we will use 2 algorithms: Bresenham and MidPoint. The result of these 2 algorithms will be converted in g-code and sent to our robot.

- **Bresenham Algorithm**

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly. In this method, the next pixel selected is that one who has the least distance from the true line.

The method works as follows:

Assume a pixel $P_1'(x_1',y_1')$,then select subsequent pixels as we work our may to the night, one pixel position at a time in the horizontal direction toward $P_2'(x_2',y_2')$.

Once a pixel in choose at any step

The next pixel is:

1. Either the one to its right (lower-bound for the line)
2. One top its right and up (upper-bound for the line)

The line is best approximated by those pixels that fall the least distance from the path between P_1',P_2'.

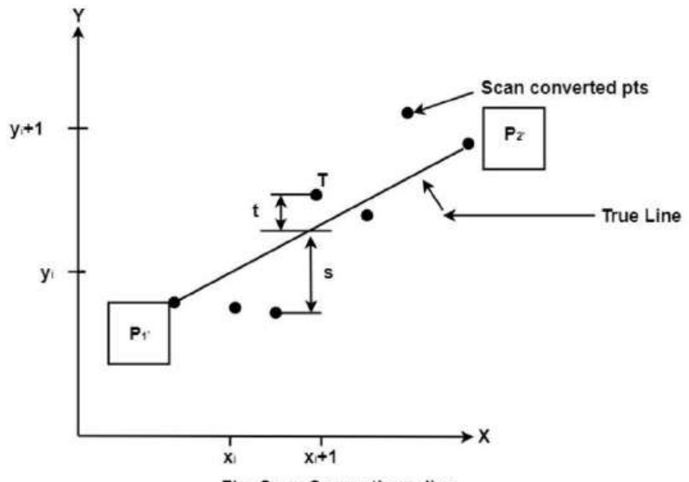


Fig: Scan Converting a line.

To choose the next one between the bottom pixel S and top pixel T.

If S is chosen

We have $x_{i+1}=x_i+1$ and $y_{i+1}=y_i$

If T is chosen

We have $x_{i+1}=x_i+1$ and $y_{i+1}=y_i+1$

The actual y coordinates of the line at $x = x_{i+1}$is : $y=mx_{i+1}+ b$

The distance from S to the actual line in y direction : $s = y-y_i$

The distance from T to the actual line in y direction : $t = (y_i+1)-y$

Now consider the difference between these 2 distance values : $s - t$

When $(s-t) <0 \Longrightarrow s < t$

The closest pixel is S

When $(s-t) \geq 0 \Longrightarrow s < t$

The closest pixel is T

This difference is

$s-t = (y-yi)-[(yi+1)-y]$

$= 2y - 2yi -1$

$$\boxed{s-t = 2m(x_i+1) + 2b - 2y_i - 1}$$

Substituting m by $\frac{\Delta y}{\Delta x}$ and introducing decision variable

$d_i=\triangle x\ (s-t)$

$d_i=\triangle x\ (2\ \frac{\Delta y}{\Delta x}\ (x_i+1)+2b-2y_i-1)$

$=2\triangle xy_i-2\triangle y-1\triangle x.2b-2y_i\triangle x-\triangle x$

$d_i=2\triangle y.x_i-2\triangle x.y_i+c$

Where $c= 2\triangle y+\triangle x\ (2b-1)$

We can write the decision variable d_{i+1} for the next slip on

$$d_{i+1}=2\triangle y.x_{i+1}-2\triangle x.y_{i+1}+c$$

$$d_{i+1}-d_i=2\triangle y.(x_{i+1}-x_i)- 2\triangle x(y_{i+1}-y_i)$$

Since x_(i+1)=x_i+1,we have

$$d_{i+1}+d_i=2\triangle y.(x_i+1-x_i)- 2\triangle x(y_{i+1}-y_i)$$

Special Cases

If chosen pixel is at the top pixel T (i.e., $d_i \geq 0) \Longrightarrow y_{i+1}=y_i+1$

$$d_{i+1}=d_i+2\triangle y-2\triangle x$$

If chosen pixel is at the bottom pixel T (i.e., $d_i<0) \Longrightarrow y_{i+1}=y_i$

$$d_{i+1}=d_i+2\triangle y$$

Finally, we calculate d_1

$$d_1=\triangle x[2m(x_1+1)+2b-2y_1-1]$$

$$d_1=\triangle x[2(mx_1+b-y_1)+2m-1]$$

Since $mx_1+b-y_i=0$ and m = $\frac{\Delta y}{\Delta x}$, we have

$$d_1=2\triangle y-\triangle x$$

Advantages:

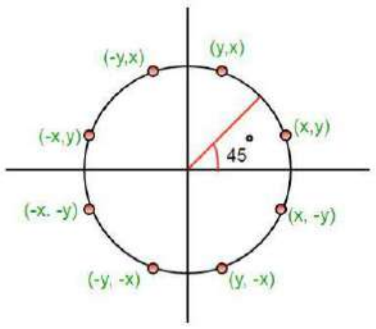
- It involves only integer arithmetic, so it is simple.
- It avoids the generation of duplicate points.
- It can be implemented using hardware because it does not use multiplication and division.
- It is faster as compared to DDA (Digital Differential Analyzer) because it does not involve floating point calculations like the DDA algorithm.

Disadvantages:

- This algorithm is meant for basic line drawing only initializing is not part of Bresenham's line algorithm. So to draw smooth lines, this one is not very helpful.

- Midpoint Circle Algorithm

The midpoint circle drawing algorithm is an algorithm used to determine the points needed for rasterizing a circle. We use the midpoint algorithm to calculate all the perimeter points of the circle in the first octant and then print them along with their mirror points in the other octants. This will work because a circle is symmetric about its center.



It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin:

$$f(x, y) = x^2 + y^2 - r^2 \quad \begin{bmatrix} < 0 \text{ for } (x,y) \text{inside the circle} \\ = 0 \text{ for } (x,y) \text{on the circle} \\ > 0 \text{ for } (x,y) \text{outside the circle} \end{bmatrix} \text{.....equation 1}$$

Now, consider the coordinates of the point halfway between pixel T and pixel S

This is called midpoint (x_{i+1},y_i- $\frac{1}{2}$) and we use it to define a decision parameter:

$$P_i=f(x_{i+1},y_i-\frac{1}{2}) = (x_{i+1})^2+(y_i-\frac{1}{2})^2-r^2 \text{ ....equation 2}$$

If P_i is -ve ⇒midpoint is inside the circle and we choose pixel T

If P_i is+ve ⇒midpoint is outside the circle (or on the circle)and we choose pixel S.

The decision parameter for the next step is:

$$P_{i+1}=(x_{i+1}+1)^2+(y_{i+1}-\frac{1}{2})^2- r^2\text{............equation 3}$$

Since x_{i+1}=x_{i+1}, we have

$$P_{i+1} - P_i = ((x_i + 1) + 1)^2 - (x_i + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - (y_i - \frac{1}{2})^2$$

$$= x_i^2 + 4 + 4x_i - x_i^2 + 1 - 2x_i + y_{i+1}^2 + \frac{1}{4} - y_{i+1} - y_i^2 - \frac{1}{4} - y_i$$

$$= 2(xi+1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$

$$P_{i+1} = P_i + 2(x_i + 1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i) \text{..........................equation 4}$$

If pixel T is chosen ⇒P_i<0

We have y_{i+1}=y_i

If pixel S is chosen ⇒P_i≥0

We have y_{i+1}=y_i-1

$$\text{Thus,} \quad P_{i+1} = \begin{bmatrix} P_i + 2(x_i + 1) + 1, & \text{if } P_i < 0 \\ P_i + 2(x_i + 1) + 1 - 2(y_i - 1), & \text{if } P_i \geq 0 \end{bmatrix} \text{.............equation 5}$$

We can continue to simplify this in n terms of (x_i,y_i) and get

$$P_{i+1} = \begin{bmatrix} P_i + 2x_i + 3, & \text{if } P_i < 0 \\ P_i + 2(x_i - y_i) + 5, & \text{if } P_i \geq 0 \end{bmatrix} \text{............................equation 6}$$

Now, initial value of P_i (0,r)from equation 2

$$P_1 = (0 + 1)^2 + (r - \frac{1}{2})^2 - r^2$$

$$= 1 + \frac{1}{4} - r^2 = \frac{5}{4} - r$$

We can put $\frac{5}{4}$ ≅1

∴r is an integer

So, P_1=1-r

4. Design

The construction of the machine uses 2 NEMA 17 stepper motors for the X and Y axis, and a micro-servo motor for the gripper. The brain of this robot is an Arduino UNO board in combination with a CNC Shield and two A4988 stepper drivers.

The work area is fairly big, 360x280mm. The motion for the X and the Y axes is provided through GT2 Belts and some GT2 pulleys and the micro-servo motor is used for the gripper. Now, we need to tell our machine how many steps is allowed to make on each axis and for that we will use the Bresenham Line Algorithm with Gcode.

4.1. A4988 Stepper Driver

What is an A4988 Driver?

Fig9: A4988 Stepper Driver

The A4988 is a microstepping driver for controlling bipolar stepper motors (NEMA17) which has a built-in translator for easy operation. This means that we can control the stepper motor with just 2 pins from our controller, or one for controlling the rotation direction and the other for controlling the steps. The driver provides five different step resolutions: full-step, half-step, quarter-step, eight-step and sixteenth-step. Its logic voltage is from 3 to 5.5 V and the maximum current per phase is 2A.

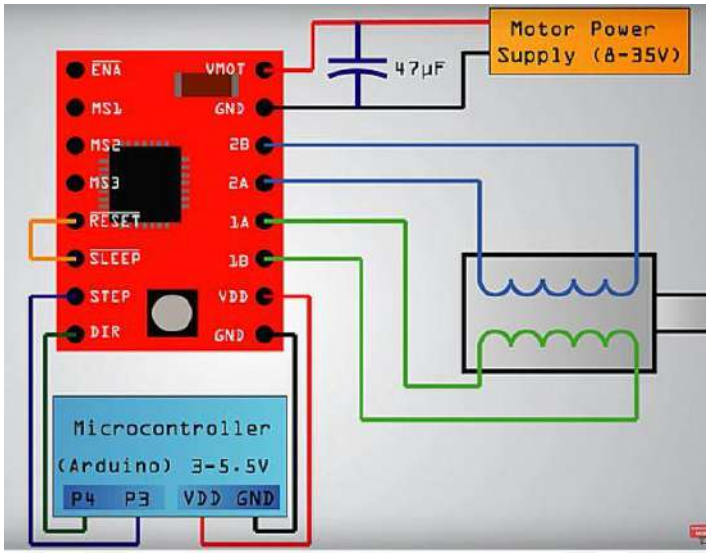


Fig10. A4988 Stepper Driver Pinout

Now let’s take a look at the pinout of the driver and hook it up with the stepper motor and the controller. So we will start with the 2 pins on the button right side for powering the driver, the VDD and Ground pins that we need to connect them to a power supply of 3 to 5.5 V and in our case that will be our controller, the Arduino Board which will provide 5 V. The following 4 pins are for controlling the motor. The 1A and 1B pins will be connected to one coil of the motor and the 2A and 2B pins to the other coil of the motor. For powering the motor we use the next 2 pins, Ground and VMOT that we need to connect them to Power Supply for let’s say 8-10 V and also we need to use a decoupling capacitor with at least 47 µF for protecting the driver board from voltage spikes. The next 2 pins, Step and Direction are the pins that we actually use for controlling the motor movements. The Direction pin controls the rotation direction of the motor and we need to connect it on one of the digital pins on our microcontroller, or in our case I will connect it to the pin number 4 of my Arduino Board.

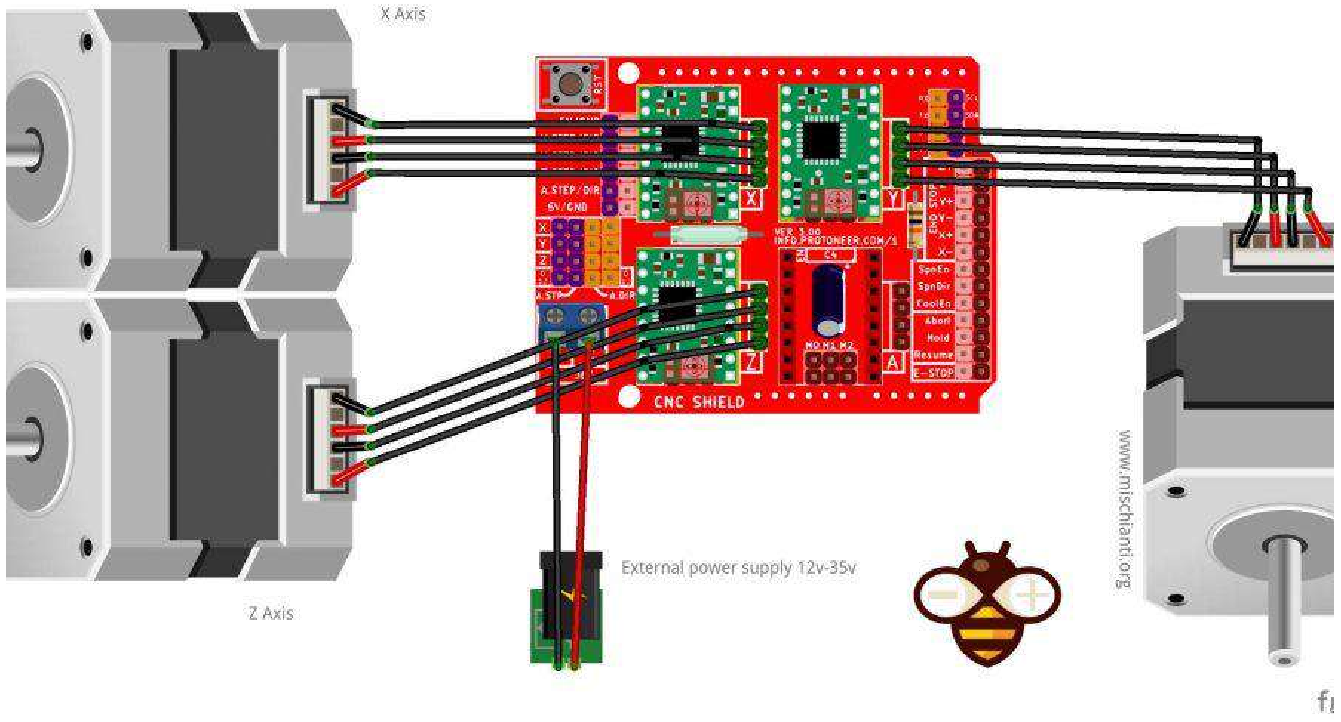
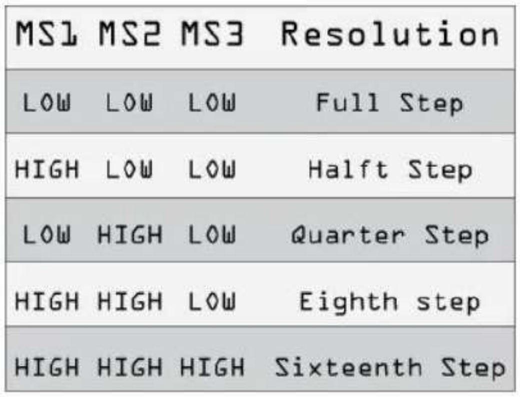


Fig12. CircuitSchematic_phase1

With the Step pin we control the mirosteps of the motor and with each pulse sent to this pin the motor moves one step. So that means that we don’t need any complex programming, phase sequence tables, frequency control lines and so on, because the built-in translator of the A4988 Driver takes care of everything. Here we also need to mention that these 2 pins are not pulled to any voltage internally, so we should not leave them floating in our program.

Next, the RESET pin sets the translator to a predefined Home state. This Home state or Home Microstep Position can be seen from these Figures from the A4988 Datasheet. So these are the initial positions from where the motor starts and they are different depending on the microstep resolution. If the input state to this pin is a logic low all the STEP inputs will be ignored. The Reset pin is a floating pin so if we don’t have intention of controlling it within our program we need to connect it to the SLEEP pin in order to bring it high and enable the board.



MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

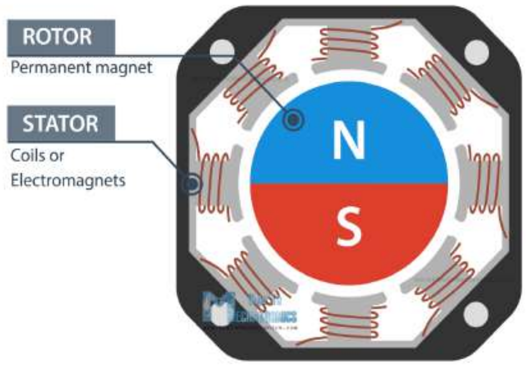
Fig13.TruthTable

The next 3 pins (MS1, MS2 and MS3) are for selecting one of the five step resolutions according to the above truth table. These pins have internal pull-down resistors so if we leave them disconnected, the board will operate in full step mode.

The last one, the ENABLE pin, is used for turning on or off the FET outputs. So a logic high will keep the outputs disabled.

4.2. Stepper Motors

A stepper motor is a unique type of brushless DC motor whose position can be precisely controlled even without any feedback. The working principle of a stepper motor is based on its magnetic fields. It has two main components, a stator and a rotor. The rotor is usually a permanent magnet and it's surrounded by some coils on the stator.



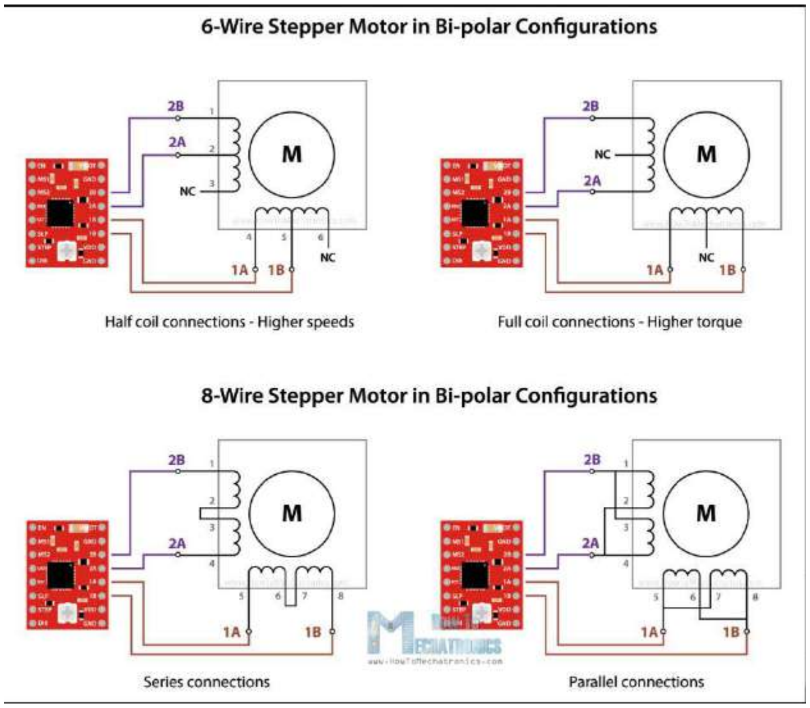
Source:
https://howtomechatronics.com/wp-content/uploads/2022/05/Stepper-Motor-main-components-stator-and-a-rotor.png
Fig15. Inside of a Stepper Motor

When we energize or let current flow through the coils, particular magnetic fields are generated in the stator that either attract or repel the rotor. By activating the coils, step by step, one after another in a particular order, we can achieve continued motion of the rotor, but also, we can make it stop at any position. So, that's why these motors are called stepper motors, they move in discrete steps.

The stepper motor that we use, Nema17 has 50 stopping points or steps on the rotor. On the other hand, the stator can have several coils organized in two phases which provide four different magnetic field orientations or positions. So, the 50 steps of the rotor multiplied by the 4 different magnetic field orientations, make a total of 200 steps for completing a full rotation. Or if we divide 360 degrees by 200 steps, that's a resolution of 1.8 degrees per step.

I mentioned that the stator coils are organized in two phases, and we can also notice that if we take a look at the number of wires of a stepper motor. It has 4 four wires, two for each phase. The four different magnetic field orientations are possible as we can let current flow through the phases in both directions.

The thing with them is that they can provide different performance characteristics, like more torque or more speed, depending on how we connect these wires on the four control terminals. Nevertheless, with this brief explanation, now we understand that for driving a stepper motor, we cannot just connect power to it as nothing will happen. Instead, we have to energize the two motor phases in both directions, and activate or send pulses to them in particular order, in a timely sequence. So, that's why we need drivers for controlling stepper motors.


6-Wire Stepper Motor in Bi-polar Configurations
2B
2A
NC
M
1A
1B
NC
Half coil connections - Higher speeds
2B
NC
2A
M
1A
NC
1B
Full coil connections - Higher torque
8-Wire Stepper Motor in Bi-polar Configurations
2B
2A
M
1A
1B
Series connections
2B
2A
M
1A
1B
Parallel connections
www.HowToMechatronics.com

There are many types and sizes of drivers, corresponding to the many types and sizes of stepper motors. However, the basic working principle of all of them is that they have two H-Bridges that allow energizing the motor phases in both directions.

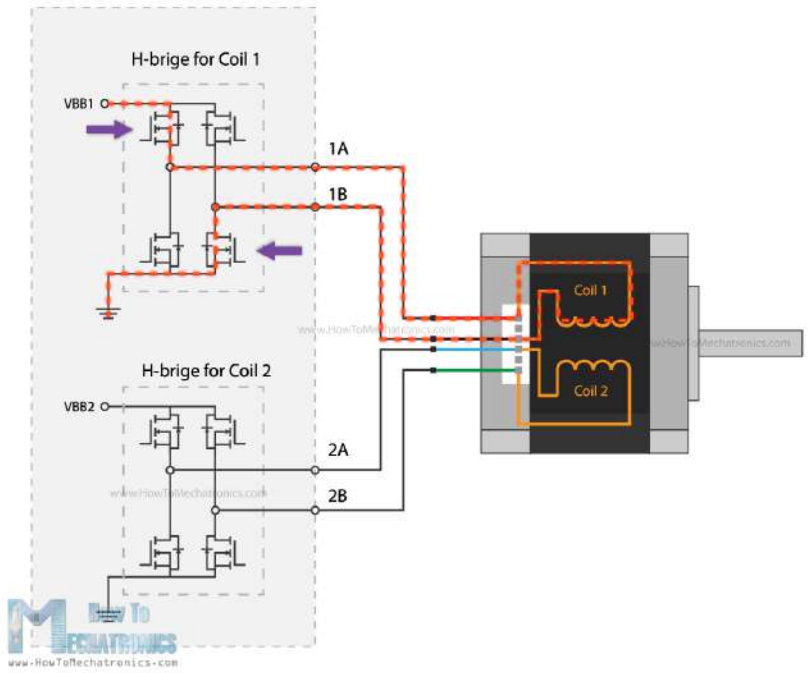


Fig16. H-Bridge inside a stepper motor
Source: Inside A Stepper Motor

4.3. Servo Motors

What is a Servo Motor?

A servo motor is a closed-loop system that uses position feedback to control its motion and final position. There are many types of servo motors and their main feature is the ability to precisely control the position of their shaft. The actual position captured by these devices is fed back to the error detector where it is compared to the target position. Then according to the error the controller corrects the actual position of the motor to match with the target position.

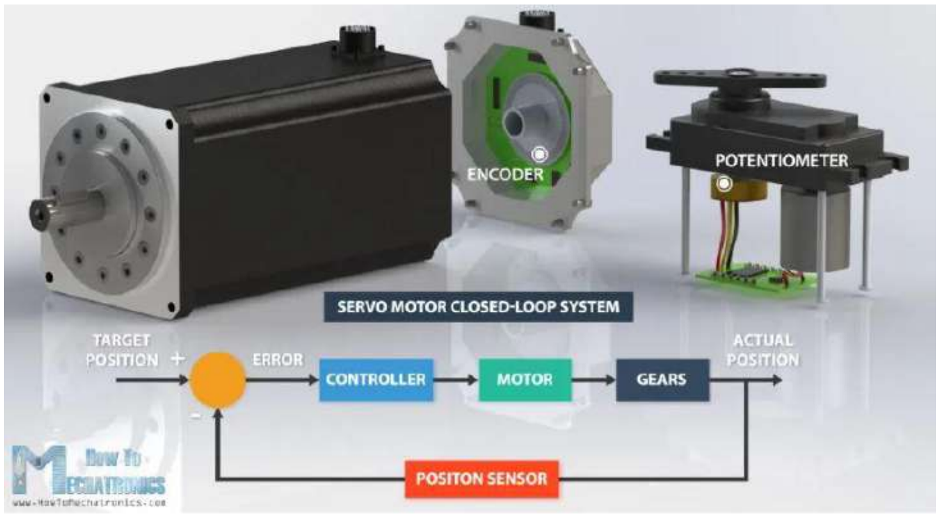


Fig17. How Servo Motor Works

How does Servo Motor Work?

There are four main components inside of a hobby servo, a DC motor, a gearbox, a potentiometer and a control circuit. The DC motor is high speed and low torque but the gearbox reduces the speed to around 60 RPM and at the same time increases the torque.

The potentiometer is attached on the final gear or the output shaft, so as the motor rotates the potentiometer rotates as well, thus producing a voltage that is related to the absolute angle of the output shaft. In the control circuit, this potentiometer voltage is compared to the voltage coming from the signal line. If needed, the controller activates an integrated H-Bridge which enables the motor to rotate in either direction until the two signals reach a difference of zero.

A servo motor is controlled by sending a series of pulses through the signal line. The frequency of the control signal should be 50Hz or a pulse should occur every 20ms. The width of pulse determines angular position of the servo and these types of servos can usually rotate 180 degrees (they have a physical limits of travel).

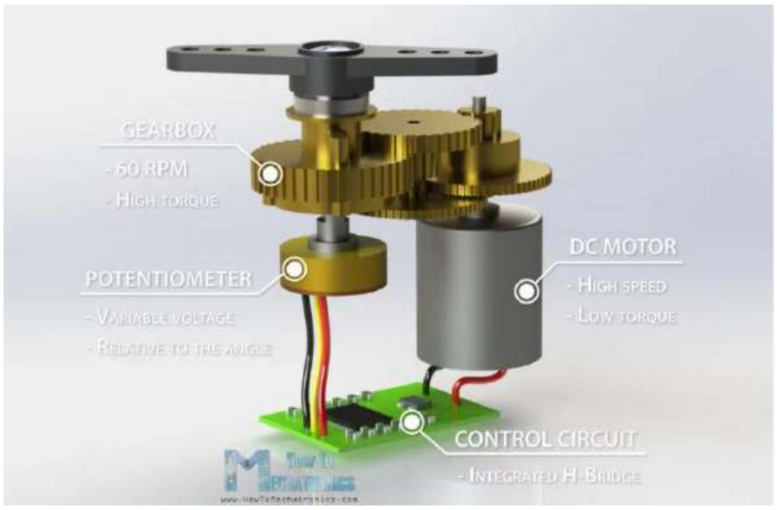
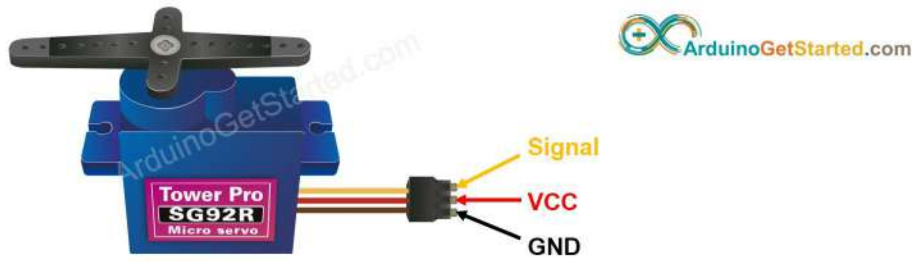


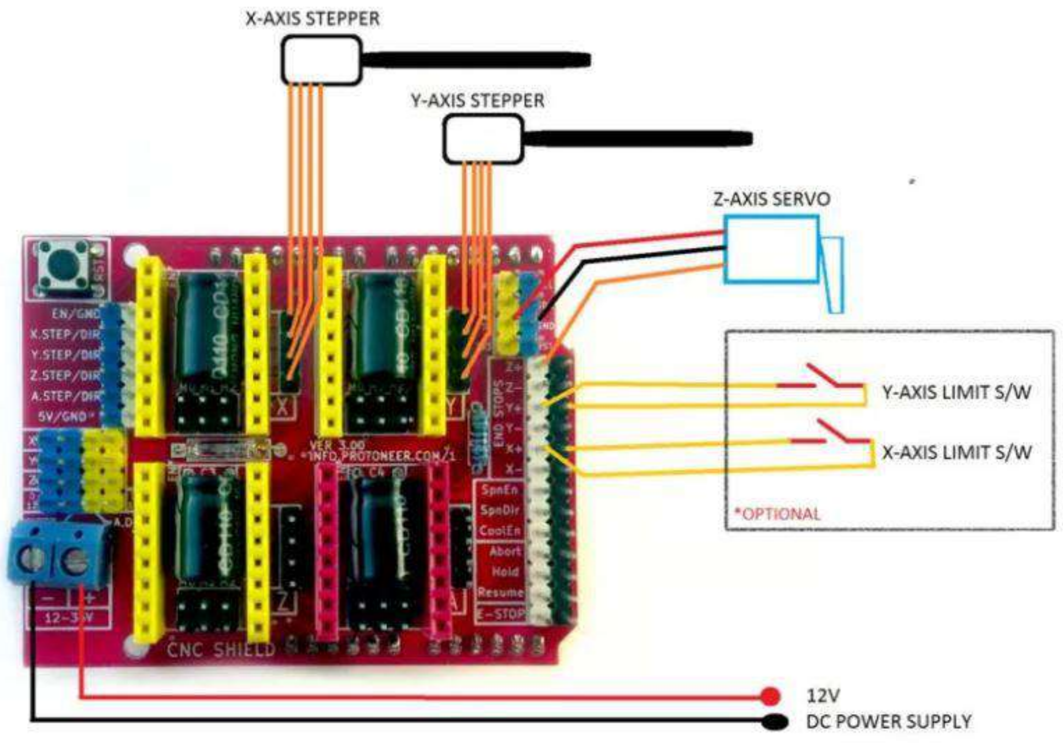
Fig18. Inside a Servo Motor
Source: How to Control Servo Motors with Arduino - Complete Guide

Once we settle the stepper motors, the next thing to secure in place is the servo-motor. It will be used as a gripper, letting and taking the pen. To do that we need to see how to connect it on the CNC Shield, but firstly we need to make sure we know what servo-motor's pins mean.



Source: https://arduinogetstarted.com/tutorials/arduino-servo-motor
Fig19. Servo Motor Pinout

Now we are ready to connect it to our CNC Shield and Arduino UNO board.



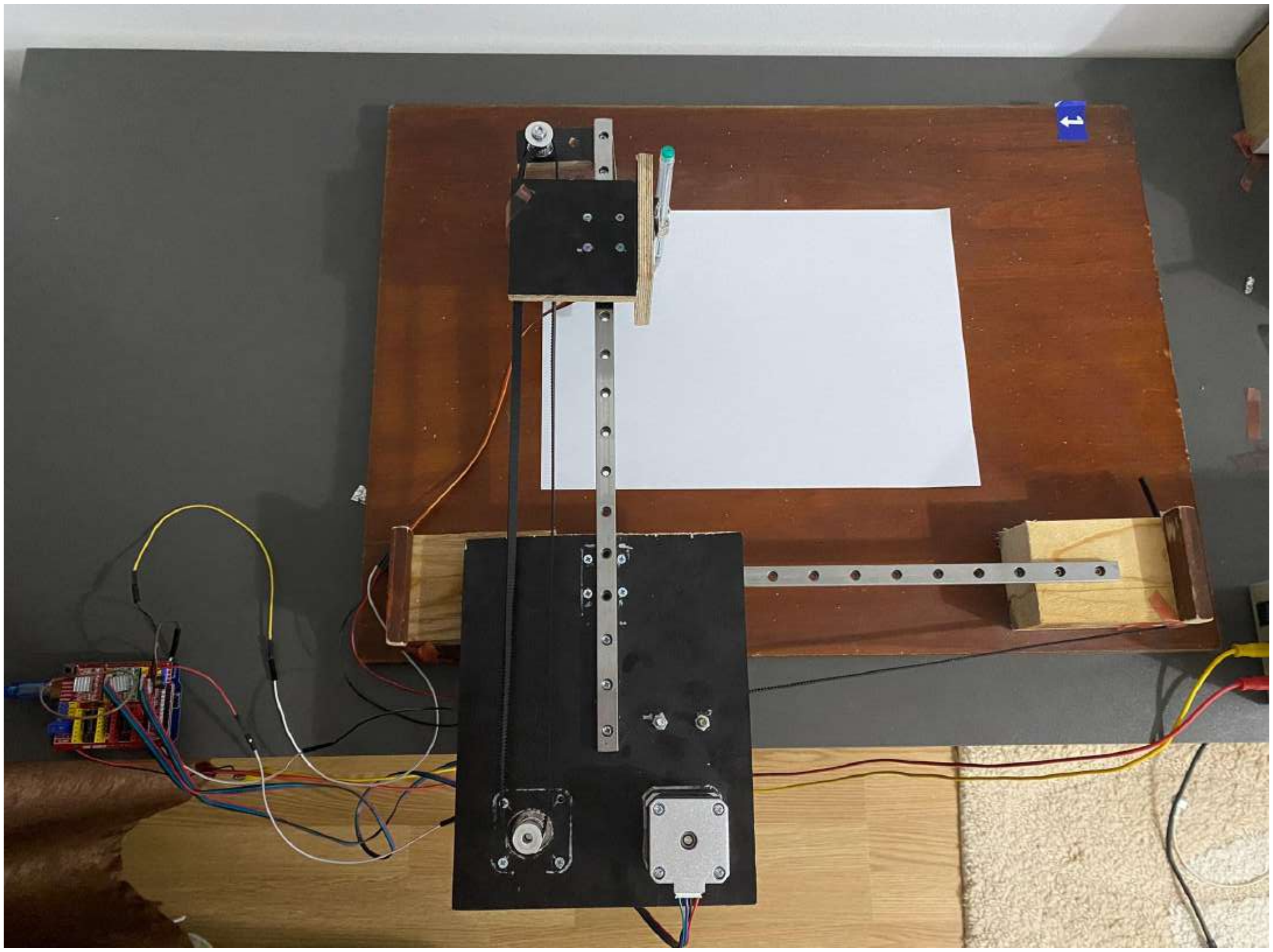
Source: https://electricdiylab.com/grbl-cnc-shield-z-axis-servo-migrbl/
Fig. How to Connect Servo on CNC Shield

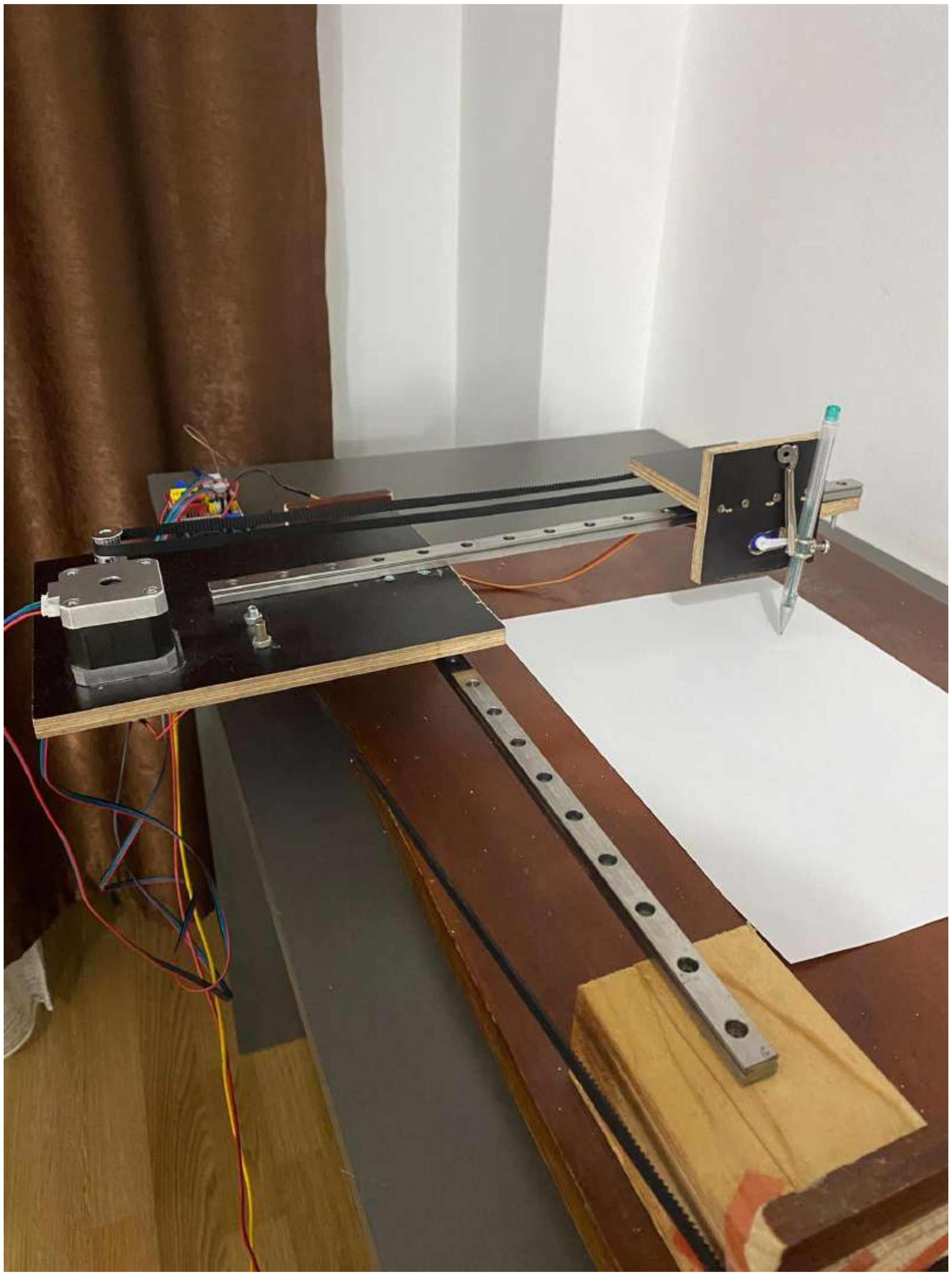
As we can see from the figure above we need to connect the VCC and GND pins of the servo motor to the VCC and GND pins of the CNC Shield and the signal pin of the servo motor goes on the “Z+” pin of the CNC Shield.

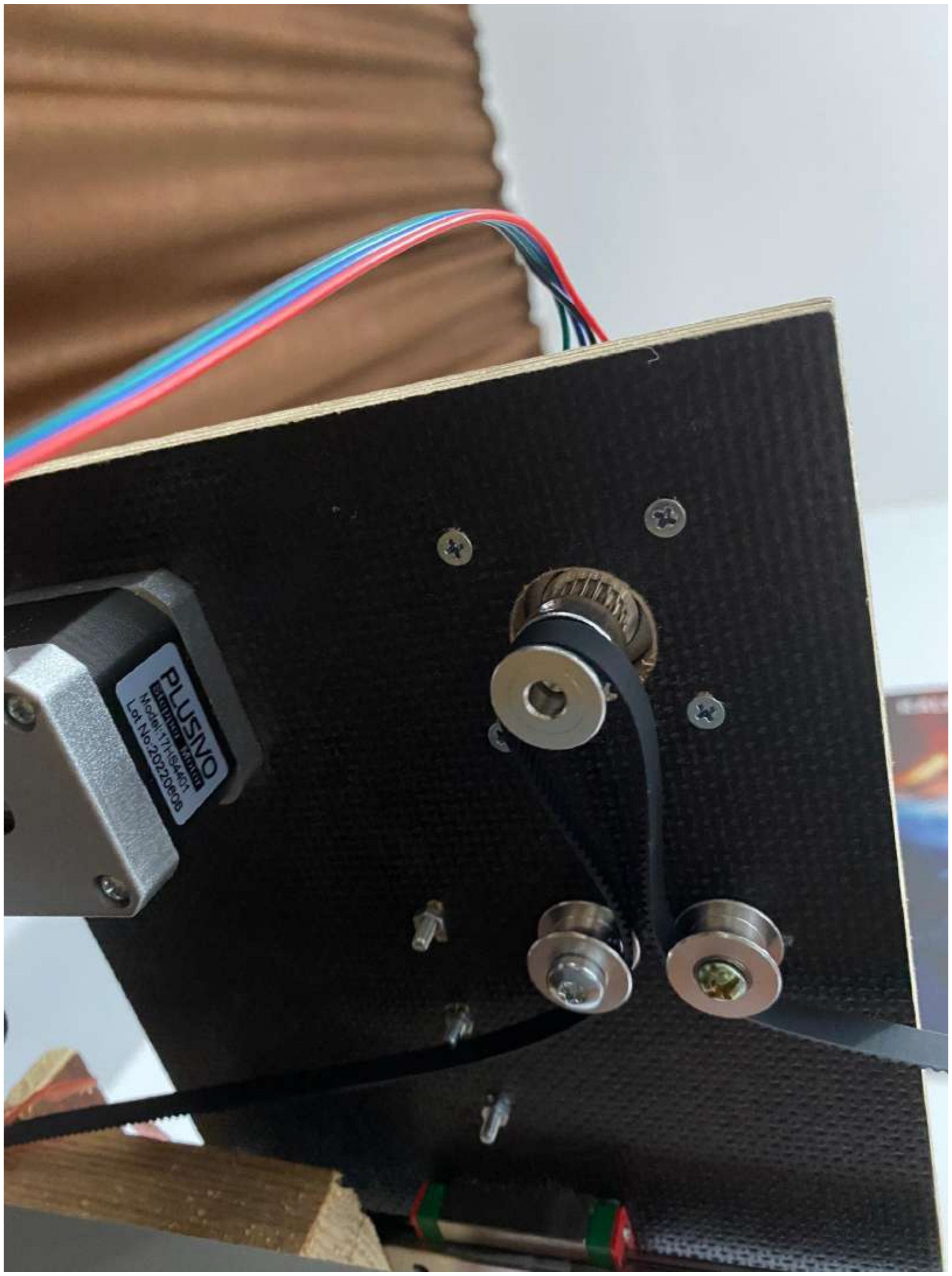
4.4. Final Design

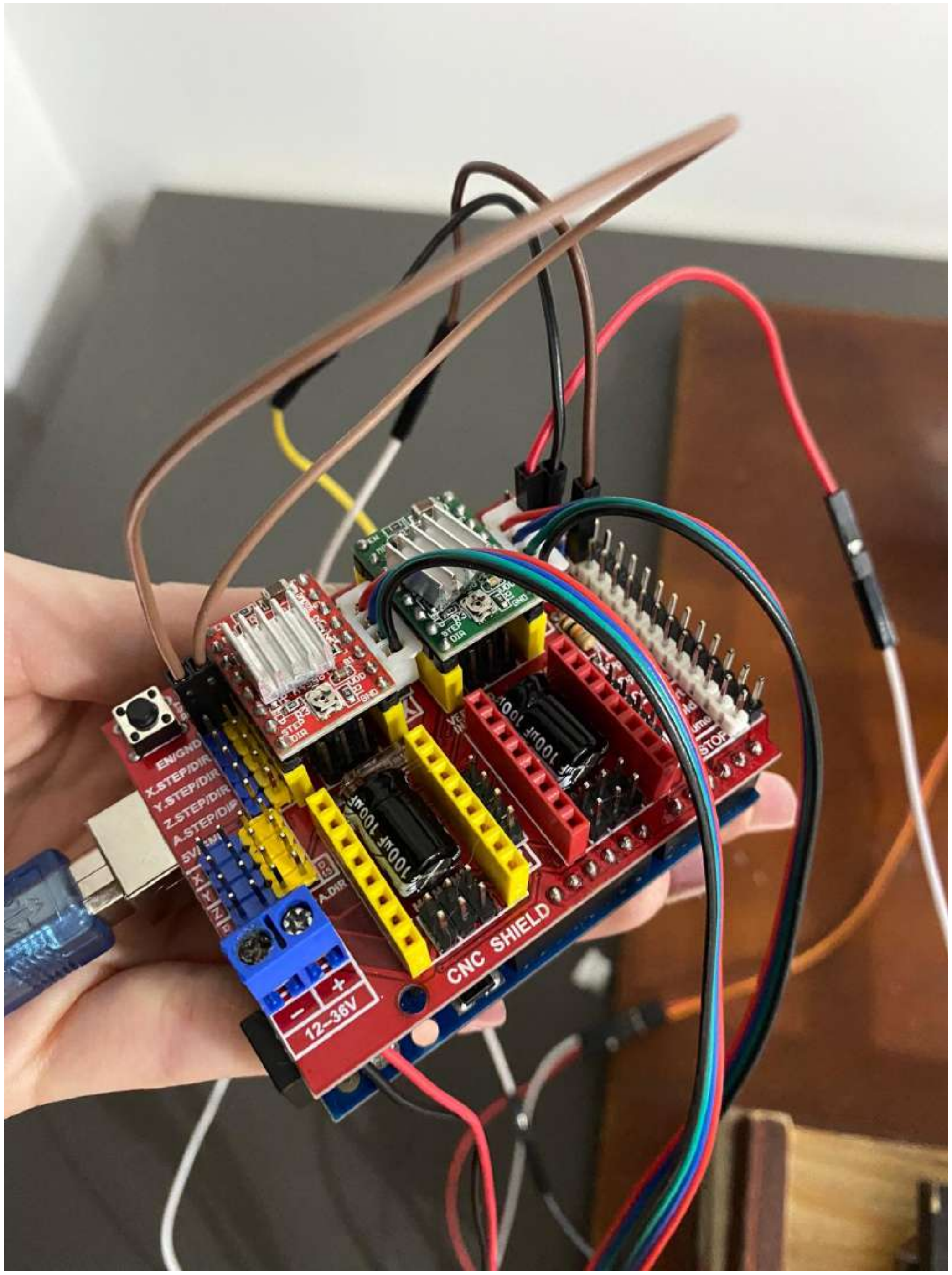
Now we have all motors ready to be connected to the power supply and do the work. The machine will consist of two MGN15H linear rails for each axis and they will provide the motion, sliding with the motors help. The two Nema17 stepper motors will be secure on one of the rails on a wood stand. On each servo motor will be placed a double tooth pulley and in front of one stepper will be placed the 2 idler pulleys and these 3 connected through a GT2 belt will provide the motion on the X axis. On the other rail will place the servo motor with the pen and at the edge of the rail will put a tooth pulley which will be connected to the other double pulley (that is on the stepper motor) through a GT3 belt and those will provide the movement on the Y axis. All the mechanics will be placed on a wooden plate on which will have the piece of paper that will be our work area and the mechanism is ready to work!

Here it is the final result:






PLUSIVO
Model:17HS4401
Lot No:20220606


EN/GND
X.STEP/DIR
Y.STEP/DIR
Z.STEP/DIR
A.STEP/DIR
5V
A.DIR
CNC SHIELD
12-36V
100uF

5.Bibliography

- **https://howtomechatronics.com/tutorials/g-code-explained-list-of-most-important-g-code-commands/**
- **https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/**
- **https://howtomechatronics.com/tutorials/arduino/stepper-motors-and-arduino-the-ultimate-guide/**
- **https://www.allaboutcircuits.com/projects/servo-motor-control-with-an-arduino/**
- **https://www.youtube.com/watch?v=zUb8tiFCwmk**
- **https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/**
- **https://www.javatpoint.com/computer-graphics-midpoint-circle-algorithm**
- **https://en.wikipedia.org/wiki/G-code**
- **https://howtomechatronics.com/tutorials/g-code-explained-list-of-most-imp ortant-g-code-commands/**
- **https://en.wikipedia.org/wiki/Arduino**