Automated Pen Plotter

Anthony DeMore, Patrick Caughey, Al Moatasem Al-Abri, Peregrino Quansah

Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450

Abstract **- A pen plotter is a device with a similar goal in mind as a standard inkjet or laser printer, but with a different process and appearance of outputs that cannot be replicated. Our pen automated pen plotter design has been created for pen plotting hobbyists to create unique works of art in an easy and efficient manner. We are able to accomplish this by the use of a robust graphical user interface which the user is able to interact with in many different ways to create their perfected output. This is paired with a visually interesting and mechanically sound plotting design and hardware which allow the user's design to take shape on paper. Many of the current pen plotters on the market today are rather expensive and it was our goal to create a cheaper design that still has any feature a user may desire. This paper presents the elements which make up our design, our testing process as well as our reasoning and thought processes behind the making of our pen plotter.**

I. Introduction

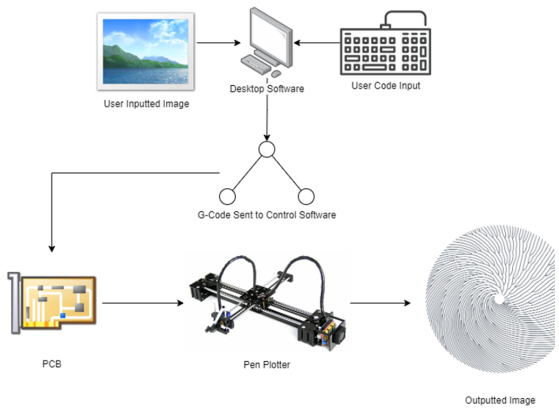
Printing as we know it has been around for quite some time now. The majority of printers we see on the market are inkjet and laser. A somewhat forgotten form of mechanically outputting images to paper format is known as pen plotting. A major reason for this is the lower cost and speed of inkjet and laser printers cannot be matched by pen plotters. However, the unique output style and artistic expression ability of the pen plotter cannot be matched by modern printers and that is what our group hopes to showcase with our project.

Many existing pen plotters are expensive and difficult to use. There is also a very small market so creating competition within this market is a goal of ours which would spark innovation and choice for the user. We are able to achieve these goals by the use of powerful and relatively inexpensive mechanical and electronic parts, paired with software based on carefully thought out industry-standard open source software.

Our capable and robust user interface allows the user to control the plotter in many different ways and allows for versatility when creating an output image. The user interface is connected to the device's firmware, this firmware translates the user interface commands and allows for the movement along our plotter's three separate axes. The control software is controlled by our PCB design which is based on an Arduino Uno which we used throughout our development and testing process. Each axis is controlled by a stepper motor which requires a driver PCB for each unit. Each of the three axes is powered by separate Nema 17 Stepper motors and is moved by a pulley system along their respective linear rails.

The resulting output image can only occur with the precise movement of our plotter which can only be achieved by each layer of our design described in the following sections working simultaneously for the best user experience.

Below is a block diagram depicting the overall workflow and structure of our pen plotter design:



Overall Block Diagram

II. System Components

At the broadest scope, the system is made up of three components, which are the software, which runs on the computer, the firmware, which runs on the microcontroller, and the hardware, which includes all of the moving parts. These three components follow a chain of command, where the software sends G-code instructions to the firmware, which then interprets them and controls the hardware using pulse width modulation (PWM).

The software used is a modified version of a free-to-use G-code sender. The first component of the software is the G-code generator. With this, the software is capable of generating G-code from a variety of sources. It can generate G-code from scratch for simple rectangles and text based on fonts specifically made for it. It can also generate G-code from vector-based image files such as *.svg files. The default drawing method for *.svg files is to trace around the shape, though there are also options to fill in the shape. The software is multipurpose, so it has options that are not useful for this project, and would not be worth removing.

The second component of the software is the direct control and calibration tools. The software gives the user a control panel to manually move the pen around the working area. This is useful for manually calibrating the zero position of the system, which is useful for testing and allows the system to be tested.

The third component of the software is the actual G-code sender portion. The G-code sender sends individual G-code instructions to the plotter. It also keeps track of a simulated plotter, so it knows where the pen is at all times, which the user can compare with the real pen to visually confirm that everything is going well.

The fourth component of the software is the preview window, which allows the user to preview the drawing that the pen plotter will make, and to move, rotate and scale components of it to edit the drawing. It is also the window that shows where the simulated plotter is and how much of the drawing has been sent as a G-code to the plotter.

The G-code is sent from the software to the plotter from one of the computer's USB A ports to the PCB's USB B port.

The firmware used is a modified version of a free-to-use grbl plotter firmware. The firmware of the plotter needs to take in G-code commands and control the hardware. The first component of the firmware is the G-code accepter, which takes in G-code and puts them into a queue which is then used to calculate motion values for the motors.

The second component of the firmware is the motion calculator, which calculates the best speeds for the motors in order to follow the path at the target speed.

The third component of the firmware is the motor drivers, which control the motors using PWM.

The hardware must be capable of moving the pen in all three dimensions and holding the pen. The first component of the hardware is the X and Y axes, which use belts, linear rails, and stepper motors to move the pen in two dimensions.

The second component of the hardware is the Z axis, which uses a stepper motor to lift the pen holder or let it come down.

The third component of the hardware is the pen holder, which uses a clamp to hold the pen in place.

III. Hardware Design and electrical components

A. System Block Diagram.

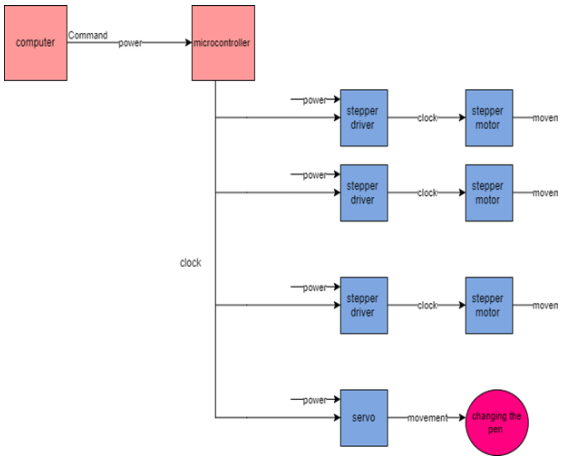


Fig. 1. System Block Diagram.

The figure above is for our system block diagram. Our system block diagram is divided into two parts: the first part is the microcontroller and the computer and the second part represents the mechanical movement. For the first part, the microcontroller will be connected directly to the computer, and the user will be using the computer to send data and commands to the microcontroller. For the mechanical movement part, the microcontroller will be connected to three stepper drivers and a servo, and the microcontroller will be responsible for sending clock signals to them, where the stepper drivers will control the direction and the movement of the stepper motors and the stepper motors will move the rails along the x-axis, y-axis, and z-axis. The servo will be used to change the pen color when it is needed, so if there is an output that uses two different colors the servo will be responsible for holding and releasing the pen.

B. Power Distribution.

For the power distribution in our project, there will be two different power sources used. For the first power source, we will be using an AC to DC power converter which will be converting the 120V that is coming from the wall outlet to a useful 12V, and this 12V will be used by the

stepper motors. For the second power source, we will use the USB port that is coming from the computer which will be providing us with 5V which is needed to power the microcontroller, three stepper drivers, and the servo.

In order to control the whole project will need to use two different chips one is the microcontroller chip and the other one is a USB chip.

C. Microcontroller.

For the microcontroller chip, we decided to use the ATMEGA328p. This chip has 32KB

flash memory and 2KB SRAM which is enough to operate a project such as our project. Also, this chip operates between 1.8-5.5 volts and it can be found for $6.28. the other chip that we will need in our project is the connector between the USB port and the microcontroller where this chip will be used to send commands and data from the computer to the microcontroller, so we chose to use the FT232R which is a USB to serial UART interface with optional clock generator output. This chip will help in sending the user's input to the microcontroller, this chip has 28-pins and it can work with 3.3V - 5.25V.

D. Stepper motors.

For the movement in the project and to move the rails across the x-axis, y-axis, and z-axis we will need to use motors. There are several types of motors that can be found in markets, however, there is one famous motor that is commonly used in 3D printing which is the Nema 17 stepper motor, this type of motor can be found in many other projects such as Robotics, CNC Machining, and Laser cutters. This type of stepper motors provides high torque at low speed and the torque rating for them is 45 N*cm, also these stepper motors have a 1.8-degree step size which means that for every step the motor increments it will rotate 1.8 degrees in the direction that is needed where they have 200 steps per revolution. The Nema 17 stepper motors have an open-loop system which means that there is no feedback.

E. Stepper Drivers

To control and operate stepper motors we will need to use stepper drivers. The stepper drivers that we are using will be working as the voltage supply for the stepper motors where they can operate under a minimum voltage of 8V and a maximum voltage of 36V, also stepper drivers help on providing the required current by the stepper motors which is 1.2A. stepper motors can also change the polarity so they can move the stepper motors in the opposite direction, additionally, they can a very accurate movement at low speed by changing the step size. There are different types of stepper drivers that can do the same work that we need and for our project, we are using the A4988 stepper drivers.

F. Limit Switches

One important component we will need to use in our project is limit switches. Limit switches are electromechanical devices used to detect the presence or absence of movement; they are commonly used in projects that have moving parts such as our project where the rails will be moving in different directions. Limit switches work on controlling the movement of the moving parts where sometimes they prevent the moving part from moving and sometimes they allow the moving part to move, so limit switches work when the moving part touches them and whether they open or close the circuit depends on what they are needed to do. In our project, we are using three limit switches, one for the x-axis, one for the y-axis, and one for the z-axis. They work to stop the motors when it reaches a certain limit by closing the circuit when the moving part touches the limit switch.

IV. Software Design

The successful operation of this project cannot be achieved without the use of the appropriate software components. Most of the software used in this project is open source but has been modified to meet the specifications of our project. This section gives an insight into the kind of tools used for

the software composition and the various software combined to bring life to the hardware. The software components can be broken down into two:

1) The firmware
2) The control software.

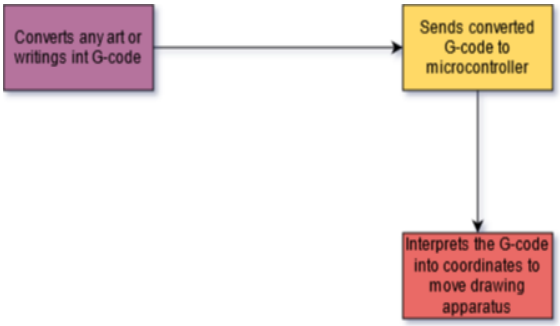


Fig 2. High-level Software Flow

A. Software tools.

Putting together the various software for this project required the use of various software tools. The first software tool we used was the Arduino Integrated Development Environment. We needed the Arduino IDE to edit the selected firmware for our microcontroller. The Arduino IDE was also used to flash the firmware onto the microcontroller. The reason we opted for the Arduino IDE over the various IDEs out there is that the Arduino IDE is specific to the microcontroller we used for the project. It also has a fairly simple user interface that allows the user to accomplish their task with minimum complications. It is also popular, hence it has loads of resources and support on the internet which makes it easy to get help when needed.

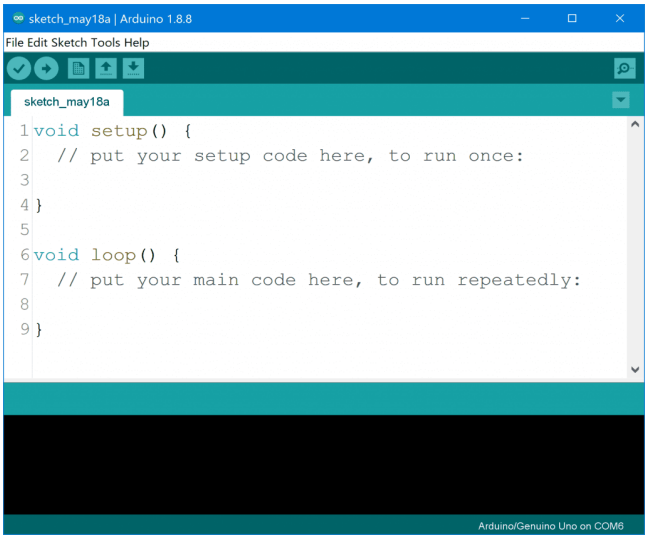


Fig 3. Arduino IDE

The second tool used for software development is the famous Visual Studio code IDE. We used the VS code to modify our chosen open-source controller software. VS code enabled us to remove certain unwanted features and also make some additions. VS code comes with the ability to install packages specific to our project which makes it very useful and versatile.

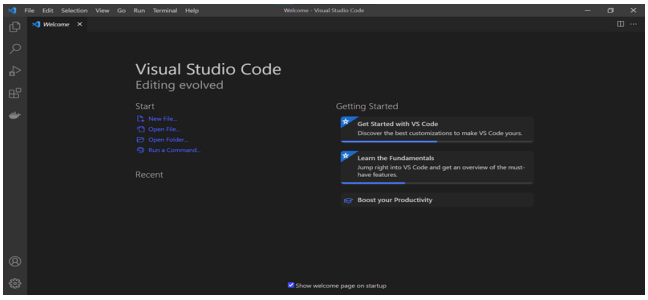


Fig 3. VS Code

B. The Firmware

The firmware we used for our project is the GRBL firmware. It is an open source high performance, low-cost alternative to parallel-port-based motion control for CNC projects. It runs with most of the Arduino boards out on the market, another reason for

choosing this firmware. It achieves precise timing and asynchronous operations. We make a few changes to the original software like changing the default frequencies of the control pin, and also add a few lines of code to enable 3-axis homing instead of the 2 default. Its main responsibility is to accept G-code from the control software, interpret it and send the interpretations to the motors onboard to act on it.

C. The control software

For the control software, we used an open source software called the GRBL-Plotter which is solely dedicated to controlling plotters. This control software comes with a graphic converter through which the user can generate G-code from images and graphics. For this software, we removed certain features like the laser controlling feature that were not utilized in this project and added a few lines of code to meet our design specification. Using the control software, the user can upload images, write texts, upload SVG files and generate G-code from them. It also provides the feature where the user can see the progress of the plotting and also track movements of the pen while displaying the G-code as it runs. There is also a debug window that enables the user to make changes in real-time.

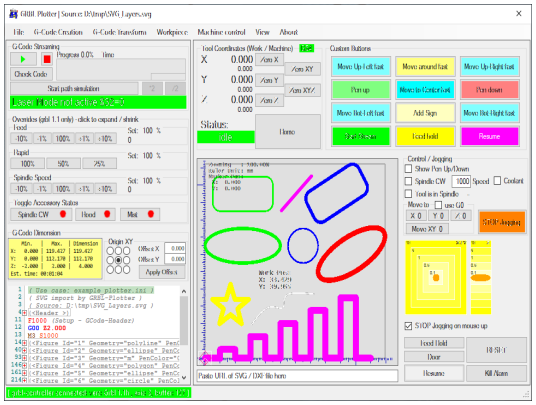


Fig 4. GUI of control software

V. Testing Results

The initial stages of testing involved determining if there was communication between the development board controlling the stepper motors on the plotter and the desktop application interface. Once this connection was established, initial movement tests were conducted. This was a somewhat meticulous process at first, as the limit switches were not installed, and the movement was done through a tool on the interface and not via g-code commands sent by a specified output image. There was a good amount of trial and error getting the speed of the plotter and distance traveled per motor rotation correct for our purposes. During this testing, we realized some modifications needed to be made to our mechanical design in order to increase stability and precision. Our z-axis movement was the thing that was most hindered by the 3d printed pen holder of our design. It had to be modified later in order to create our desired product. We tested the range of motion of our plotter, the working area we decided on was larger than that of a standard 8 x 11 paper and we were able to achieve movement in our working area of 12 x 16 inches, which was determined to be more than enough to create large and detailed works of art.

We then began testing moving and plotting speed. We desired roughly 20 mm/s plotting speed based on rough estimates and known pen plotter movement speeds. We initially were running at a much slower rate than our desired time. Around 8 mm/s were the testing results seen by drawing the same shape of a known drawing distance and timing the time to completion. We were able to increase this speed as well as maintain the integrity of the outputs by increasing the parameters controlling the stepper motors.

Our next goal for testing was the precision and accuracy of artistic outputs based on the desired output shown in the user interface. This is likely the most important testing process as the output is the most important part of our design. The output needs to show a quality image with no discernible differences from the desired output the user set in the interface. Initially, we experienced some inaccuracies in our output due to some of the pulley systems not functioning correctly. As seen below, the two headphone drawings below have slight variations and are not as accurate as desired.

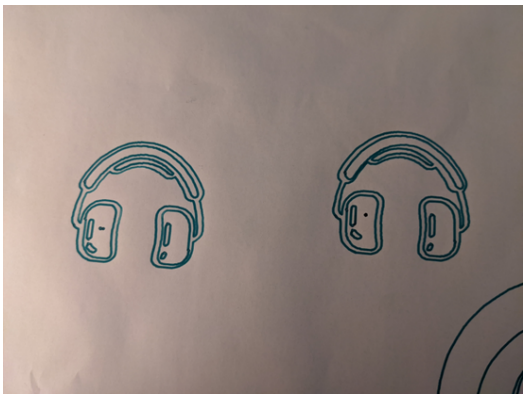


Image 1: Slight Inaccuracies During Initial Testing

This was corrected by tightening our pulley system and making sure there was no slippage on the teeth of the pulley during longer plots. We also needed to make sure the writing surface was secured in order to avoid any paper movement which could cause an undesirable output. After we improved our pulley accuracy, the outputs improved tremendously as seen in the after image of three more of the same headphone outputs seen below. These are the same outputs as seen above, but they are all virtually indistinguishable from each other as we desired with our accuracy specifications set during our design stages.

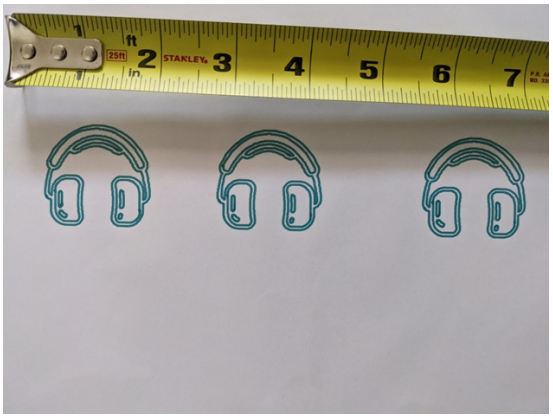


Image 2: Inaccuracies Corrected

VI. Conclusion

We believe that our pen plotting design was a success overall and an enjoyable process throughout. We achieved our goals of creating a usable and mechanically sound plotter with outstanding movement and precise drawing capabilities. The user interface which we were able to integrate with our design also worked very well alongside the onboard firmware. The outputs of our project exceeded our expectations and it was enjoyable to see the reactions of others when our device was creating a work of art.

There were a few aspects of our design and design process as a whole that could be improved upon, but this was a great medium for challenging us as engineers and preparing us for our futures. This project overall was a great learning experience for all of us. Not only in the design aspects of software and hardware, but also in terms of working as a team, establishing goals, creating a managing a budget, and other important aspects of engineering which are not taught in typical class curriculums. These experiences are something we can all consider when managing our future careers and there are many valuable lessons we are learning throughout this process.

VII. The Engineers

Almoatasem Alabri is a 23-year-old senior Electrical Engineering student at UCF. He expects to graduate in August 2022.

Peregrino Quansah is a graduating senior Computer Engineering Student at the University of Central Florida. He hopes to work with a reputable Software Engineering firm where he can put the knowledge acquired to practice.

Anthony DeMore is a Computer Engineering student who will be graduating in August 2022. He currently works as an intern at Leidos, a defense contractor, and has accepted an offer to work post-graduation. He is interested in working in simulation software and computer vision. He is also considering obtaining a master's degree in computer science.

Patrick Caughey is a senior in Computer Engineering graduating in August. He hopes to work as a video game developer to develop his dream game.

VIII. References

[1] written by John, John. “Best CNC Software [2022] for Hobbyists and Pros [Free and Paid].” MellowPine, 20 Feb. 2022

[2] “Platformio vs Arduino Eclipse Plugin.” Compare Differences & Reviews?,https://www.saashub.com/compare-platformio-vs-arduino-eclipse-plugin.

[3] “Top 15 Best Embedded Systems Programming Languages.” UbuntuPIT, 29 Apr. 2021, https://www.ubuntupit.com/best-embedded-systems-programming-languages/.