Plotter Documentation

Aarre Kilpeläinen, Sebastian Coffeng, Petri Lassila

January 11, 2021

1 Introduction

This text documents the hardware and software of an Arduino-based plotter. The plotter uses stepper motors to move a pen and a platform in two perpendicular dimensions and a servo motor to raise and lower the pen at desired times. This creates a plot, drawing or other desired result on a surface placed on the platform. This way, the plotter can be used for various different tasks, for example to draw sketches for PCB, to draw schematics or to make art. The plotter can be seen in figure 1.

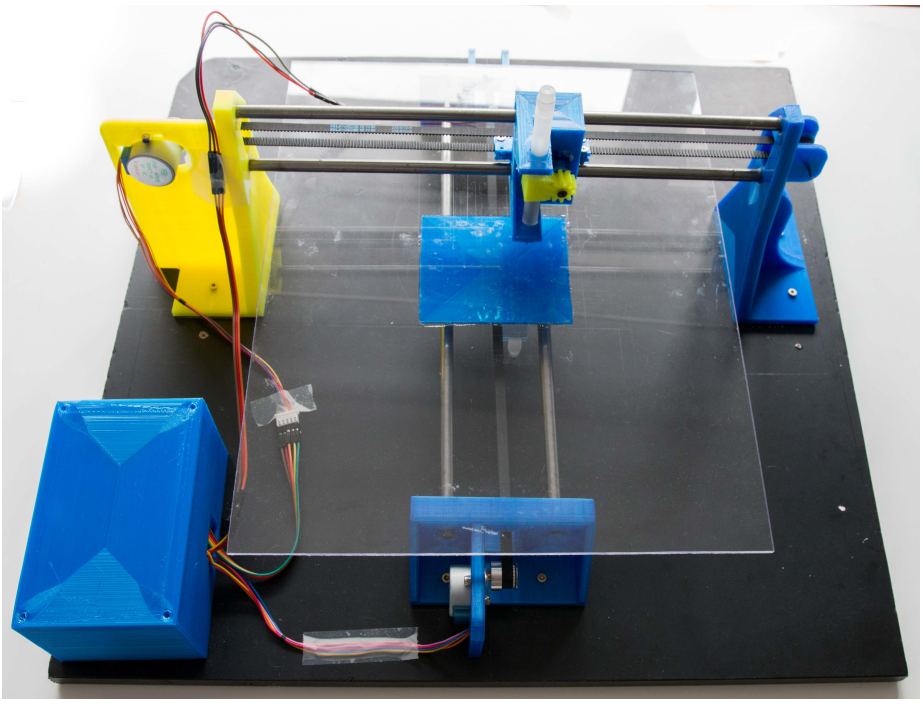

Figure 1: The plotter

The following sections describe the hardware that makes up the plotter, then the software necessary to efficiently use the plotter, a summary regarding the use of the plotter, and an estimate of its accuracy. The final section contains a summary of the group's process of creating the plotter, including a weekly time usage log of the group.

2 Hardware

The parts used are as follows:

1. Arduino UNO microcontroller
2. 2x 9V5V DC-DC Converter
3. 2x 9V1A Adapter
4. Servo motor SG90
5. 2x 28BYJ-48 5V DC Stepper motor
6. 2x ULN2003 stepper motor driver module
7. 4x 210mm x 8mm metal rails
8. 4x LM8UU Linear bearing
9. 2x GT2 Belt for moving the stage
10. Custom designed 3D-printed PLA plastic parts

The servo motor, the stepper motors, and the UNO microcontroller were chosen primarily due to them already being available to us. All the parts are very cheap off-the-shelf parts. This allows for cheap and accessible design for a wide range of users. Despite the cheap components, it is possible to get relatively accurate results. The stepper motors have a built-in gearbox which allows for very small and precise steps. The motor driver modules make the wiring more straightforward. Two power sources were used to deal with potential overheating problems in powering the stepper motors that were noticed in earlier prototypes with only one source.

Metal rails are used for the free movement of the platform and the pen holder. The rails are 8 mm thick rods that fit standard LM8UU linear bearings. These metal rods give structural rigidity to the frame and ensure smooth operation of the axis. This enables more precise movements to the machine. Lubricant was also added to improve the the movements. Standard GT2 belts were used to further ensure precision. Simpler designs with just some sort of wire tied around the motors could also have been satisfactory, but using more

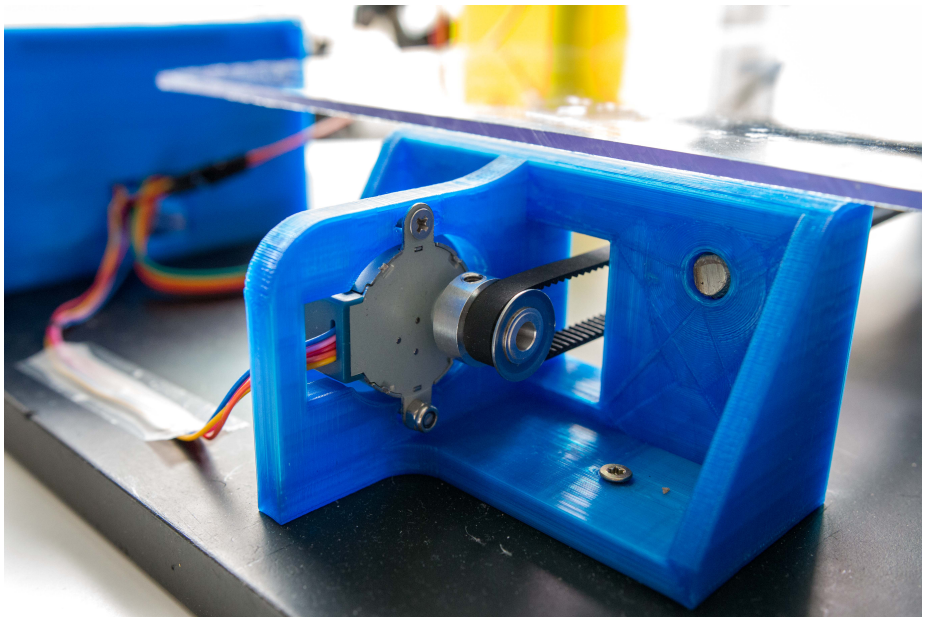

Figure 2: Stepper motor and rail holder

advanced parts prevents slippage and stretching.

The supporting structure, including the pen holder, were 3D-printed PLA plastic based on custom designs. Several iterations of the design were made. The end result has useful features such as adjustable mechanical pen pressure, enclosed electronics and a wide plexiglass drawing plate, and also adjustable screw positions in the holders making it easy to adjust the belt tension. The pen movement was implemented using a servo actuated gear system. The gears have high tolerances and thus the pen can be moved in small increments up and down. Figures 3 and 4 show this design and in figure 2 the stepper motor mount can be seen.

The assembly is bolted on to thick wooden base to ensure rigidity. The linear bearings are press fitted to the pen holder and the drawing platform and then inserted trough the metal rails. The rails are tightly mounted to the main holders and the holders are bolted to the platform by screws. After the construction is finished the electronics and belt drive system can be installed. The stepper motors are fastened onto the structures, and the belt is put onto the stepper motors. The motors are connected to drivers and ultimately to the UNO microcontroller. The pins used on the controller depend on the software. The power sources are connected to give power to the motors. The electrical components are housed in a 3D-printed compartment. The microcontroller is connected to a computer using USB. Figure 5 is a schematic of the stepper motor and servo

arrangement.

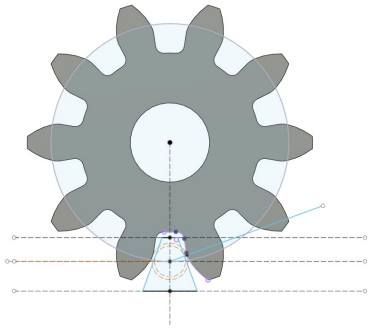

Figure 3: Schematic of the gear system design

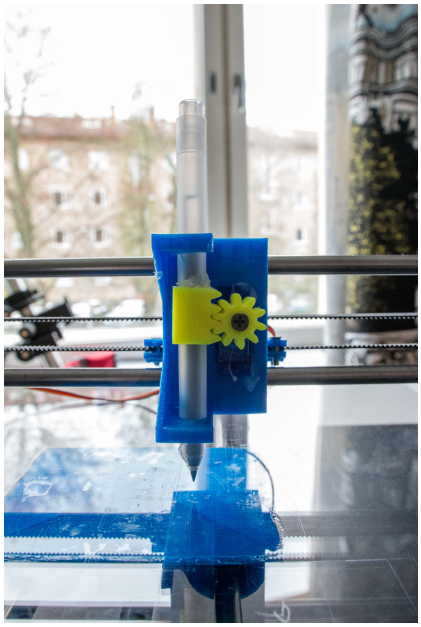

Figure 4: Pen holder

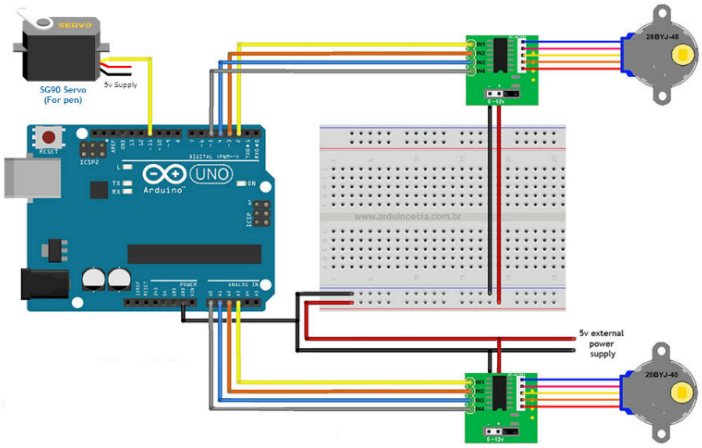

Figure 5: Schematic of the stepper motor and servo arrangement

3 Software

The software for the plotter is relatively simple, easy to operate and easy to modify to the users' needs. The software is open source but the code presented in this document has been modified to fit specifically to the needs of this project [1]. However, the hardware can be used with a large number of different software, and the user may want to write their own code to ensure behaviour of their own preference.

A computer is used to communicate with the microcontroller. The Arduino code consists of a loop that constantly reads serial input. The Arduino interprets all its commands using standard G-code. Based on these inputs, the plotter moves accordingly. The serial input is generated using the program Processing in the computer. It takes inputs either directly from keyboard commands, such as the arrow keys, or from G-code from a file stored on the computer.

The G-code can be written by the user, or more conveniently generated using a suitable program that can transform pictures into G-code. In this case, the free software Inkscape was used. Inkscape can transform pictures into bitmaps. This bitmap can be transformed into G-code using MakerBot Unicorn G-code extension for Inkscape [2]. Example picture of this process can be seen in figure 6.

The standard G-code that the plotter reads consists of coordinates line by line that the plotter can follow. When the plotter is given new coordinates through the serial port, it either raises or lowers the pen or moves from its current location to the next location. By doing this it can create lines, and as these lines are connected, images are formed. Figure 7 shows an example picture of G-code and the tool paths it creates.

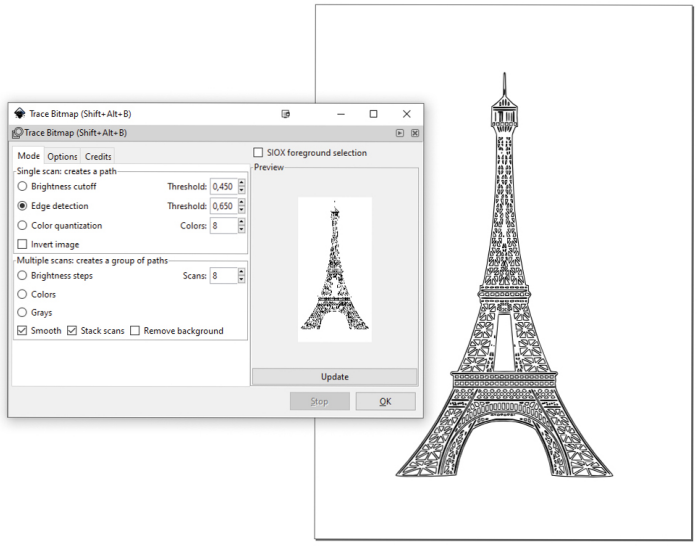

Figure 6: Creating bitmap from an image

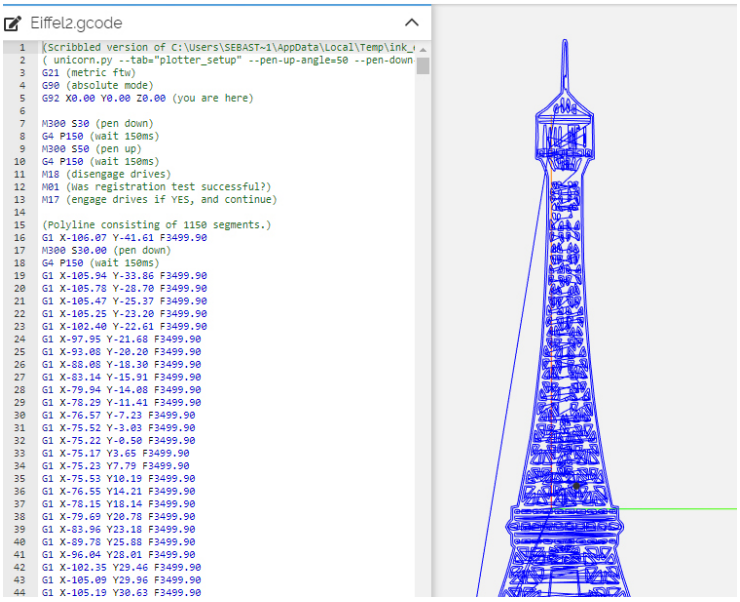

Figure 7: Example of a G-code and toll paths.

4 Plotter operation

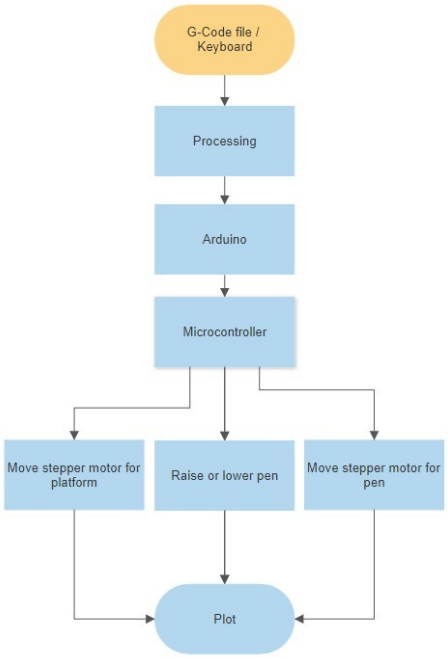

Figure 8: Chart of the operation of the plotter

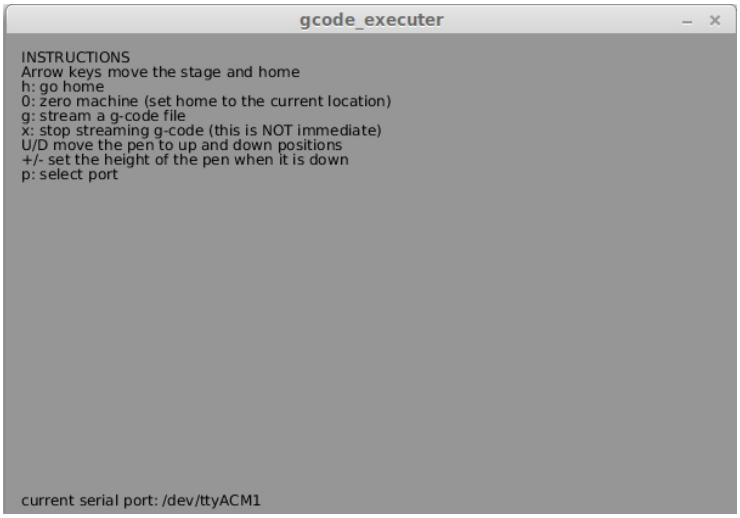

Figure 9: Picture of the interface

The plotter software is divided into two parts. Arduino code is only for interpreting G-code commands and the Prossesing program on the user's computer stores all the user commands for the plotter. The command structure of the

plotter can be seen in figure 8. When the program is run on the computer the user is presented with clear instructions for all the commands available. Figure 9 shows these user commands.

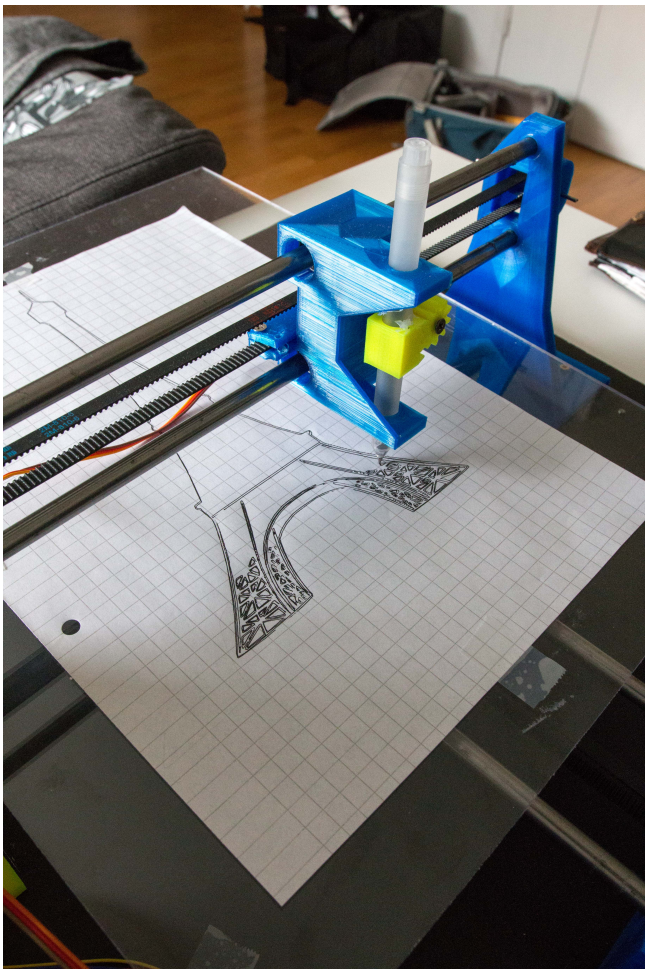

Figure 10: Drawing the Eiffel Tower

First the user needs to choose the USB port where the Arduino is connected. In our G-code examples, the origin is set to the center of the drawing plate and the user therefore has to move the head of the pen to the center of the platform with arrow keys. When the platform is centered the drawing pressure needs to be set by using the + and - keys. Once this setup has been done, the G-code file can be opened by pressing G. After this selection the plotter draws the selected picture automatically. There are also additional commands like returning "home" to the initial position of the pen, stop printing, and serial selection. Un-

finished drawing can be seen in figure 10.

Additionally the pen pressure can be adjusted during the print mechanically by screwing the end cap of the pen. With this feature the user can take into account the flexing of the drawing platform or the drawing material. In figure 11 the finished Eiffel Tower drawing can be seen.

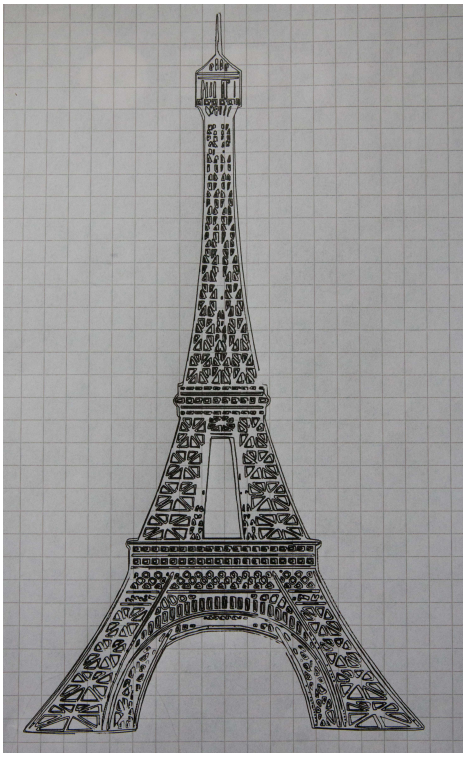

Figure 11: Finished Eiffel Tower

5 Accuracy

The accuracy of the system was determined by drawing different shapes and lines and measuring the drift of these drawings. These are only rough estimates: The measurement systems available to the group weren't particularly precise. Triangle ruler was used for the angle drift and length drift measurements. The result were that the angle drifts ± 0.5 degrees and the line drift is $1mm \pm 0.005$. The true limiting factor of the resolution is however the size of the pen, which in this case is about 0.3mm. This drift might be the result of the axis not being exactly 90 degrees from each other and possible imperfections inside of the stepper motors. Some of the inaccuracy is also due to the flex of the frame. In figures 12 & 13 the drift measurements can be seen.

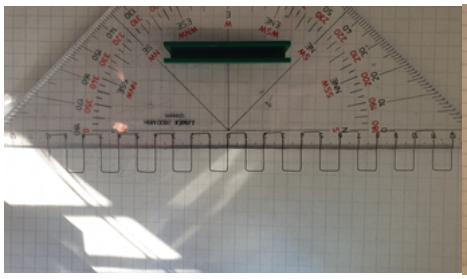

Figure 12: Picture showing the line drift

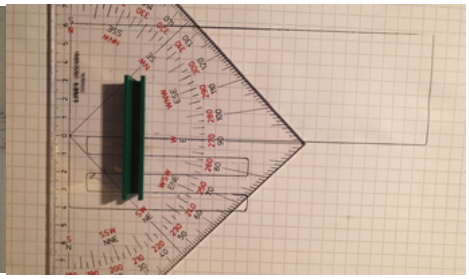

Figure 13: Picture showing the angle drift

6 Group work

The hours spent on the project are visualized in figures 14 and 15.

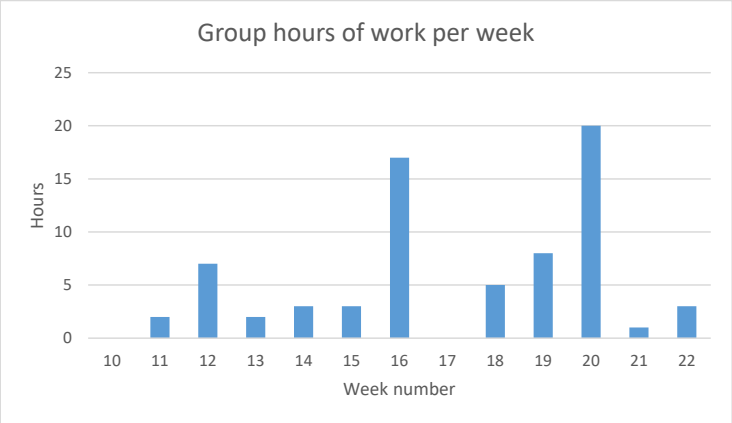

Figure 14: The hours worked by the group for each week.

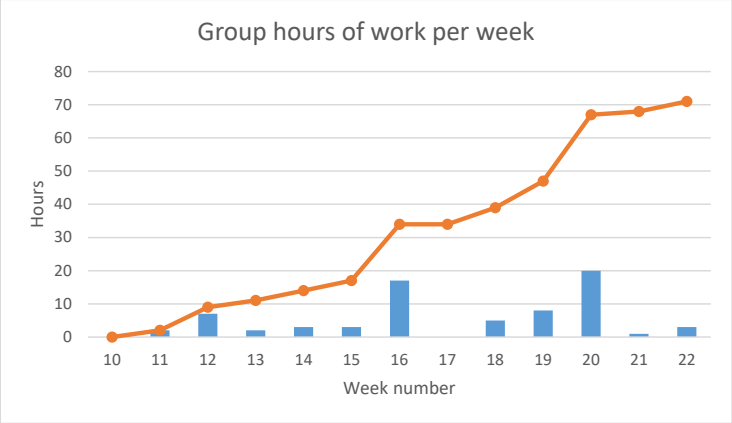

Figure 15: The hours worked by the group for each week, with cumulative hours plotted.

The work began in the beginning of March around week 10, when a plotter was agreed upon as the subject of the project. During Week 11, first plans for 3D-printed parts were created. During week 12, the first parts were created for testing. A plan for the project was agreed upon, written, and turned in. The group discussed which parts would be necessary for the printer.

The next weeks saw continued development and further refining of design of the 3D-printed parts. The group met in person during week 16 to discuss the progress so far and to solve problems relating to the stepper motors and power supply. Report "version 1" was written. No work was officially logged during week 17, as the group focused on other studies. Due to problems with the new stepper motors, the group ultimately ended up using previously used motors instead.

In the end of April and beginning of May, weeks 18 and 19, after several tests during the previous weeks, a working prototype was assembled, the design finalized, and the first proper drawings were made. Some final improvements were added, such as the protective case for the electronic components. Week 20 had the most work hours. The plotter was tested further by plotting more advanced and time-intensive figures, and some measurements were made to try to estimate its accuracy. The presentation materials were written, discussed, and reviewed, the group met in person to prepare for the demonstration, and finally the group held the demonstration. The following 2 weeks include a few hours more which were used to finish this documentation.

In total, about 70 hours were spent on the project. A great deal of effort was avoided thanks to the availability of suitable open-source code that could be adapted for the plotter. The most labour-intensive part was the designing, printing, and assembly of the 3D-printed parts. Sebastian focused on this part, and as a result he contributed the most, over 30 hours, to the project.

References

[1] Easy Arduino Projects - YouTube
https://www.dropbox.com/sh/n9n4vf6zqslcsgd/AAC6FO5gXUnHi6kOJ82ZvWPUa?dl=0

[2] MakerBot Unicorn G-code extention for Inkscape
https://github.com/martymcguire/inkscape-unicorn