|  |  |
| --- | --- |
| A close up of a suitcase  Description automatically generated  **CNC Research Project** | Computer Numerical Control Machine Design, Hardware, Firmware, and Processor Architecture  14 May 2019  Anneka Bath  Grace Adams  Seungman Chang |

**Table of Contents**

|  |  |
| --- | --- |
| **Introduction** | 2 |
| Purpose | 2 |
| Mission | 3 |
| Project Scope | 3 |
| **Research** | 4 |
| Firmware: Marlin | 4 |
| Software: G-Code | 6 |
| Hardware: Arduino | 8 |
| **Build: Designing a CNC Machine** | 11 |
| Framework | 11 |
| Drag Knife Mount | 12 |
| Wiring | 13 |
| Marlin Configuration | 14 |
| Slicer | 16 |
| **Conclusions** | 17 |
| **Picture Sources** | 18 |
| **References** | 18 |

# **Introduction**

3D printers and computer numerical control (CNC) machines are at the heart of many emerging technologies and have revolutionized manufacturing. CNC manufacturing has replaced much of the machining done by hand, reducing variation between pieces and increasing efficiency.

The difference between 3D printers and CNC machines is that 3D printing is an additive machining process, and CNC machining is a subtractive manufacturing process. "While subtractive processes remove layers of material from the workpiece to produce custom shapes and designs, additive processes assemble layers of material to produce the desired form and formative processes deform and displace stock material into the desired shape" (Ronquillo). To achieve the "desired shape," the CNC machine can be outfitted with many different tools, including but not limited to drills, lathes, and mills.

At their core, both types of manufacturing utilize precisely timed commands from the computer to create the desired product. 3D printers and CNC machines both use an embedded firmware microcomputer to automatically position and move a tool according to preprogrammed coordinate instructions. This archetype positions motors set along the X, Y, and Z axis to build layers of material or cut material away. And while 3D printers have become relatively inexpensive and widely available, CNC machines have not. In theory, taking a commercially produced 3D printer and converting it into a CNC machine can save money and allow a small business to gain the manufacturing capabilities of a large company.

## **Purpose**

The purpose of this research project is to convert a 3D printer into a CNC machine. This conversion will allow us to gain knowledge and experience with the Arduino hardware, Marlin firmware, and G-code production. As an outcome, our team will have hands-on experience in processing capabilities of the Arduino platform paired with the most widely used build platform, the three-axis CNC (Computer Numerical Control) as well as giving us the insight of designing a machine from start to finish.

## **Mission**

Our team seeks to understand how the CNC design architecture works in every step of the process. Namely, how it communicates instructions from a CAD file into g-code, then how the G-code is executed to control the machine, and finally how the machine mechanically follows the instructions. The reason we’ve chosen this task is to gain experience with real world applications of processor architecture. Gaining background knowledge and hands-on experience with this type of machine will contribute to our marketable skillsets, as well as our maker abilities. Understanding the controller architecture of the most popular tool, the 3d printer, will make us more versatile candidates in the job market.

## **Project Scope**

The scope of our project includes the physical components of the CNC architecture (processors, chip boards, adaptors, and motors) as well as how the processor’s firmware code handles, decodes, and executes the GUI’s (graphical user interface) program files, aka CAD files, and creates an executable g-code file that the machine can understand.

In this project we will describe the process in which our team was able to upgrade a Monoprice Maker Select 3d Printer V2 13860 to an Arduino Mega 2560 board and convert it into a CNC machine so it can be for any number of CNC applications. For the purpose of our in-class demonstration we’ve decide to use a drag knife attachment. This attachment allows us to cut different materials into complex shapes that would be hard to make with conventional scissors, and hard to reproduce with precision by hand.

# **Research**

Before starting our build, we needed a firm grasp of the components we'd be using. With this research, we were able to make informed design decisions and understand what was required to complete our build.

## Firmware: Marlin

One of the first decisions that we made was to use Marlin as the firmware for our CNC Cutter Machine. Marlin is an open source firmware initially intended for 3D printers, specifically the replicating rapid prototypers. It derived from Sprinter and grbl, then became a standalone open source project with its Github release in 2011. Marlin can be true firmware, hard programmed onto the CPU of a headless printer, or it can be used to control a CNC machine from a networked computer.

*Why Marlin?*

There are many reasons why we chose Marlin over other available CNC Firmware. The first was the price. Due to the nature of opensource licenses, it is free for any application. The second was flexibility. Marlin's primary goal is to be adaptable to as many boards and configurations as possible. It is configurable, customizable, and extensible for hobbyists and vendors alike. Features are enabled as-needed to adapt Marlin to added components, or disabled when elements are unnecessary, and there are a lot of users created packages available for specific modifications. The third is universality. Marlin was built by and for 3D printer enthusiasts and used by several respected commercial 3D printers. We wanted to choose 'industry standard' options as much as possible so that we can carry this experience forward into our careers. The final reason was convenience. Marlin runs on inexpensive 8-bit Atmel AVR micro-controllers. These are the chips on the Arduino Mega2560, which we used to replace the original board in the 3D printer. It also made sense to use 3D printer firmware on a 3D printer, even if we were modifying it to be a CNC machine.

Relevant Main Features:

* Full-featured G-code
* Complete G-code movement suite
* Smart motion system with lookahead, interrupt-based movement, linear acceleration
* Support for Cartesian, Delta, SCARA, and Core/H-Bot kinematics
* LCD Controller UI with more than 20 language translations
* Bed Leveling Compensation — with or without a bed probe
* Print Job Timer and Print Counter

An important detail to note, Marlin's designed for 3D printers which require movement but also heaters, extruders, and many other components. The nature of our project only needed positioning from the stepper motors. However, the modular design of Marlin was essential. We were able to disable any components we didn't need without altering the function of any other part of the code. i.e., no code transferred to the e1/e0 drivers on the RAMPS 1.4 board. These components control the extruders and heatbed.

*How Marlin Works*

Marlin manages all the real-time activities of the machine from the main board. It coordinates the components involved in the 3D printing, or our case, 2D cutting, process. The control-language for Marlin is a derivative of G-code. As Marlin receives movement commands it adds them to a dedicated movement queue, then the "stepper interrupt" processes the queue. Linear movements convert into electronic pulses to the stepper motors. Marlin generates thousands of stepper pulses every second, and the CPU speed limits how fast the machine can move. Heaters and sensors manage a second interrupt that executes at a much slower rate, but we disabled this entire queue for our project

Software : G-Code

G-code is a language based on an XYZ-coordinate system. It specifies detailed instructions on how to make a physical object based on the digital model. These instructions include, but are not limited to direction, speed, and extrusion temperature and rate. The instructions communicate to the CNC machine where to move, how fast to move, and what path to follow. G-code was initially a limited language that lacked many features that programmers are familiar with, such as loops or conditional operators, programmer-declared variables with natural-word-including names, and the ability to encode logic. Now many versions of G-code include macro-language capabilities closer to a high-level programming language.

A few examples of G-code commands and their meaning are:

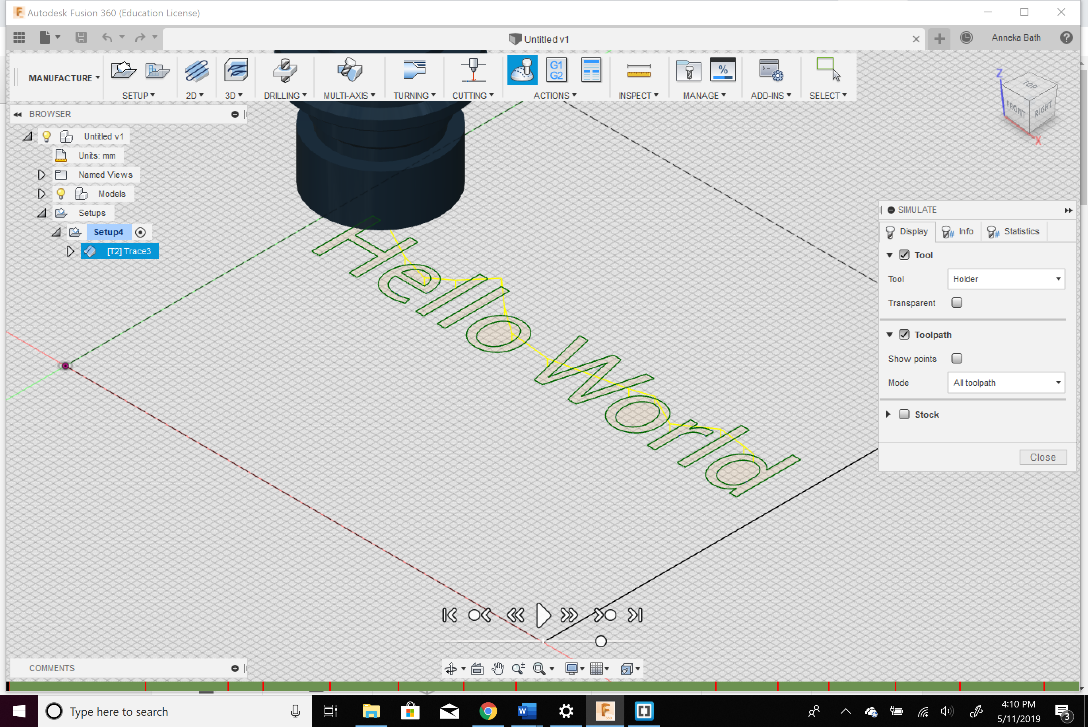
* G00 – Rapid Transverse Motion
* G01 – Motion by a feed rate
* G02 – Motion in an arc clockwise
* G03 – Motion in an arc counter-clockwise
* G04 – Dwell
* G17 – XY Plane selection
* G18 – XZ Plane selection
* G19 – YZ Plane selection
* G28 – Return to Machine zero for axis selected

*How We Made Our G-Code*

Thankfully we did not have to code directly in G-code. Once we made a CAD/CAM model of the pattern we wanted, we were able to generate the G-code for that model using Fusion 360.

We started the process using a vector design program, specifically Adobe Illustrator, to draw our design. This design was exported as an SVG design file and loaded into Fusion 360. Then, using the Manufacture design setting, we were able to develop a CAM trace along with the SVG design. The CAM tace created a path that the tool head would follow to cut out our design (figure 1).

Figure 1: Fusion 360 Simulation Screenshot



Once the design was rendered and simulated to check for errors, we could then create our G-code in post-processing. However, this took an extra step because we needed G-code precisely for a 3D printer. We were able to find a .cps file which had post-processing configuration handling for Fusion 360 that was compatible with the Marlin 1.1.4+. The file we used came from Tech2C, who is a member of the Thingiverse.com community. He also is an active member of YouTube; our team referenced his videos on how to use Fusion 360 to create G-code. Since none of us had designed a CNC machine, his walk-through of creating G-code and his .cps post-processing configuration was a vital resource for completing our CNC machine.

Once we had usable G-code, the file uploaded to the Marlin interface, called Pronterface. Pronterface is a G-code handler, meaning it sends the movement signals to Marlin Firmware, which then controls the motors and receives information from the hardware such as the endstops. The endstops are responsible for telling the printer it’s about to go out of bounds.

Hardware : Arduino

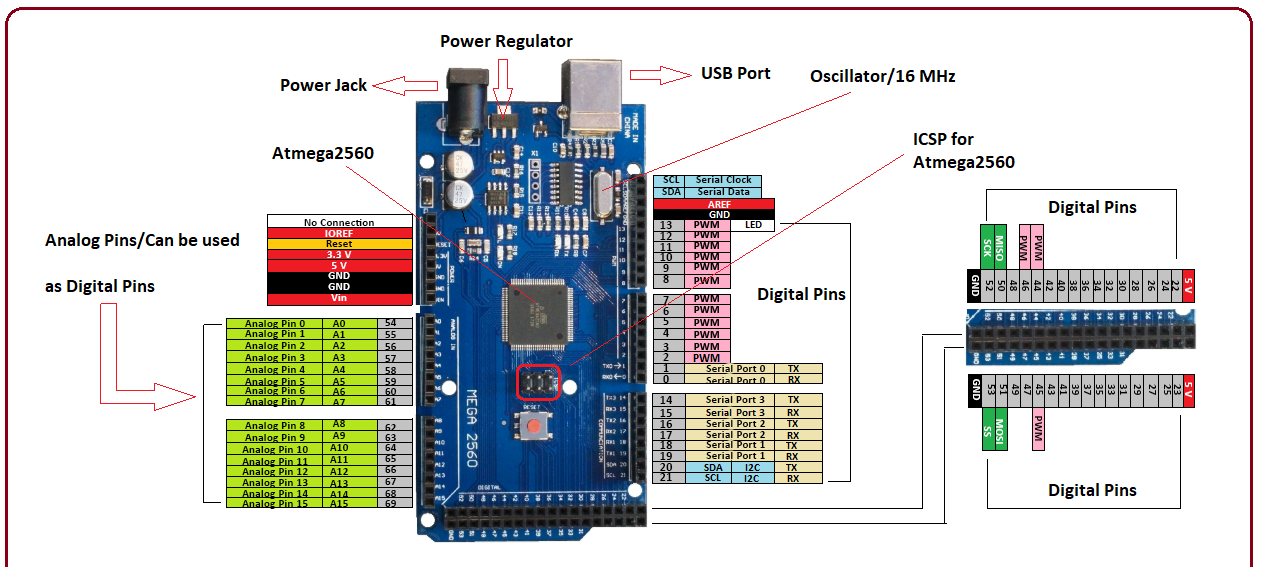
The Arduino Mega2560 is the higher-end member of a line of computers designed for more straightforward hobbyist projects. Rather than a microprocessor, powered by an Atmel ATmega2560, an 8-bit AVR microcontroller running at 16 MHz. 

Figure 2: Arduino Pin Diagram

*Specifications:*

* Length 101.52 mm
* Width 53.3 mm
* Weight 37 g
* 54 Digital I/O pins
* 16 Analog Input pins
* Flash Memory 256 KB
* SRAM 8 KB
* EEPROM 4 KB
* Clock Speed 16 MHz
* Operating Voltage 5 V
* 32 8-bit general purpose registers
* 16-bit program counter
* 480 8-bit I/O registers

*Registers*

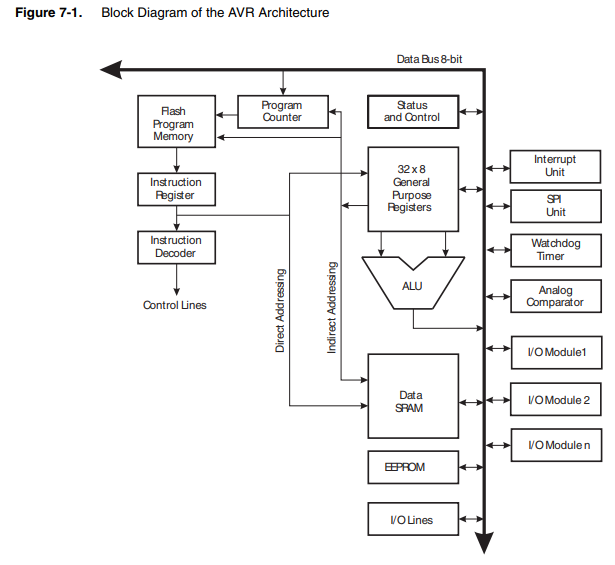
There are 32 general-purpose 8-bit registers, as well as a 16-bit-wide program counter and 480 8-bit-wide I/O registers, including the stack pointer and the program counter.

Memory writes in little-endian format.

*Instruction Set Architecture*

The microcontroller sports an AVR Enhanced RISC (Alf-Egil Bogen Vegard Wollan RISC) architecture, designed with high-level languages in mind, namely C. While similar to RISC, it supports a large number of instructions (including some CISC-like instructions) for smaller code size. Unlike ARM, AVR Enhanced RISC was designed from the ground up for microcontroller use and tended to see more use with lower-end CPUs and smaller amounts of memory.

Instructions are 16-bit and fixed-length with a word size of 8 bits. Each instruction accepts 1-2 operands. AVR is generally more limited than either MIPS or ARM; there is no register-indexed addressing or PC-relative addressing for load/store and jumps. The calls must perform relatively. There is also no conditional execution except for branching. Because the microcontroller does not feature a floating point unit, all floating point data types must emulate in the software.

 The ATmega2560's Harvard architecture allows for a single level of pipelining – an instruction is fetched from program memory while the previous instruction executes. At least three different formats exist for instructions, featuring opcodes occupying 4, 6, or 8 bits of the instruction. One notable feature of the encoding scheme is that bits representing various fields may not always be contiguous in the instruction. For example, in an ADD instruction, the left-most bit of the field representing the address of the source register appears to the left of the one representing the destination register, while the remaining four bits appear to the right of the other address.

*Datapath*

Figure 3: Block Diagram of AVR Architecture

A RAMPS (RepRap Arduino Mega Pololu Shield) board is designed to accommodate all the electronic components needed for a RepRap or other 3D-printer-like device on a single board. The board controls the following:

* Up to 5 stepper motors with 1/16 stepping precision and interface with a hotend, a heatbed, and a fan (or a second hotend)
* An LCD controller
* A 12V (or 24V with appropriate modification) power supply
* Up to three thermistors
* Up to six end stoppers

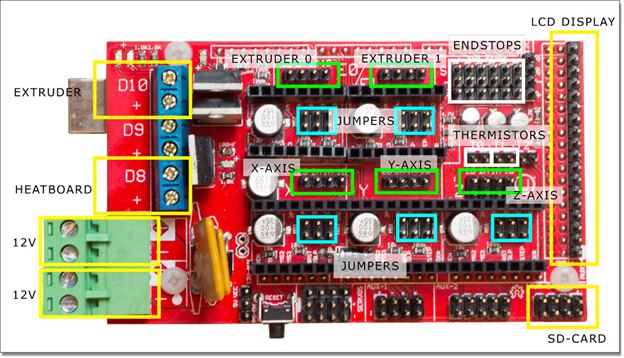


Figure 4: RAMPS 1.4 Wiring Diagram

*Drivers*

Stepper motor drivers are necessary to power the stepper drivers on the RAMPS board without compromising the functionality of the microcontroller. These drivers also allow the motors to make smoother rotations. Microcontroller-based drivers become especially crucial at relatively high speeds, as the coils in the motors must be tightly synced for optimal performance.

A chip embedded in the stepper driver supplies power with pulse-width modulation so that the motor can run without overheating. There is also a controller attached to the driver to control the orientation of the motor (i. e. clockwise vs. counterclockwise) and grounding.

*Endstops*

Three pairs of pins situated on the upper-left corner of the board (X-min, X-max, Y-min, Y-max, Z-min, Z-max) constitute the endstop pins. These allow the machine to know when the moving components (I. e. those controlled by the motor) have reached the ends of their movement axes. Mechanical contact switches are the most straighforeward implementation, but because the contact switches may not be as repeatable, optical endstops are a slightly more popular choice. These send signals to allow the motor software to reference its axes.

# **Build: Designing a CNC Machine**

Once we decided to create a CNC machine, we had to decide on what kind of CNC machine we would make. Several constraints influenced our decision, including the need to do an in-class presentation. We did not want to create a lot of debris or excessive noise. Upon researching different CNC implementations, we chose a drag knife CNC.

### Framework



The 3D printer we modified was a Monoprice Maker Select 3d Printer V2 13860 (figure 2). The 3D printer was donated to the project, due to a recent machine upgrade.

*CNC Hardware Update*

Figure 5: Monoprice Maker Select 3D Printer V2

The original 3D printer design ran off a Melzi board which runs similarly to a RAMPS 1.4. When reviewing CNC machine architecture, we found that Ramps 1.4 was widely used for machine controllers. Thus, we decided to incorporate a RAMPS 1.4 controller paired with an Arduino Mega 2560 to operate the CNC motors.

### *CNC Firmware*

Next, we had to decide on which firmware to use to configure the machine’s hardware. Grbl is the industry standard for CNC machines; however, it was agreed to go with Marlin for three main reasons:

1. Marlin is the firmware of choice for a 3D printer architecture. Since we were working from a 3D printer framework, it made sense to stick with Marlin.
2. We decided it would be smart to work with the Marlin firmware in case the machine was needed to convert back to a 3D printer. Grbl doesn’t support 3D printing since its primary purpose is positioning. In later research, we realized this was the optimal choice because Marlin incorporates Grbl for its motion support. We gained the industry standard quality of Grbl and the flexibility of Marlin all at once.
3. The 3D printer is the most popular maker tool on the market. This logic made the most practical sense from a marketability perspective. By having experience with Marlin, our skills would translate well to the job market for this field.

Drag Knife Mount

 After the deciding on firmware, we needed a way to hold a drag knife along the x-axis bar. First, we chose a small, precision blade drag knife, the “GAOHOU 15pcs 30 45 60 Degree GCC Cutting Plotter Vinyl Cutter Blade + Holder” to mount as our tool head. With the tool’s dimensions, we were able to find an existing 3D print template on a site called Thingiverse.com to help create our mounting device. Searching for a mounting device led us to the “MPCNC Roland Drag Knife Holder,” produced by Mjacobs518. This design was designed and paired with a different Thingiverse.com design called the “Universal Mount 525 for MPCNC” by HicWic.

Figure 6: GAOHOU Blades and Holder

*A picture containing indoor, table, wall, sitting

Description automatically generated* We were able to print the drag knife holder as is. However, the Universal Mount design needed to be modified to fit on the platform where the extruder was previously mounted. We were able to commission Adam Bath, a CAD/CAM engineer, to draft up a 3D model of the Universal Mount to fit on the modified Maker Select tool mount platform. Once he made the changes to the original design, we were able to print and assemble the pieces with the Drag Knife Holder, and the Vinyl Cutter Blade.

Figure 7: Final Drag Knife Setup

Wiring

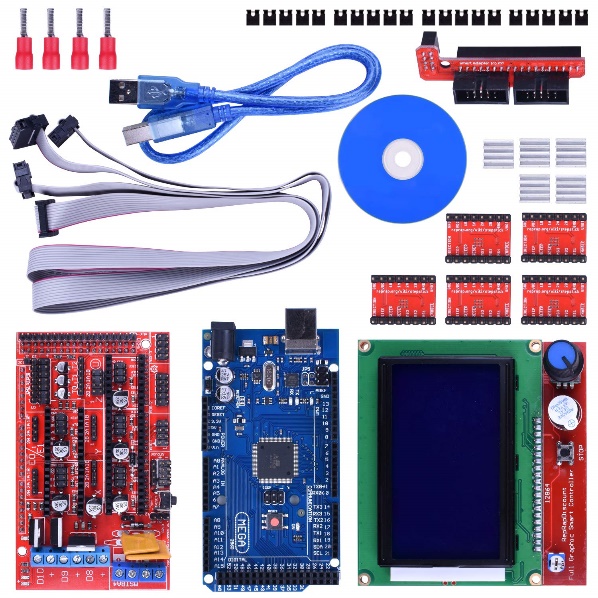
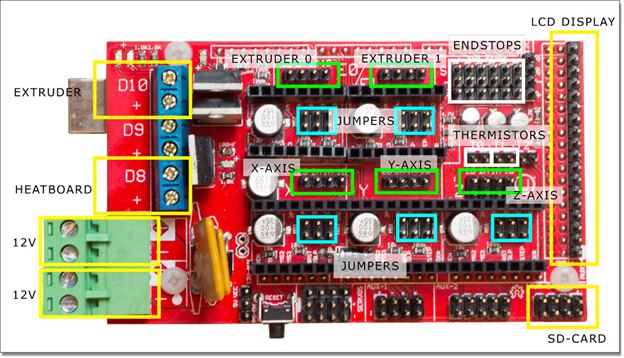
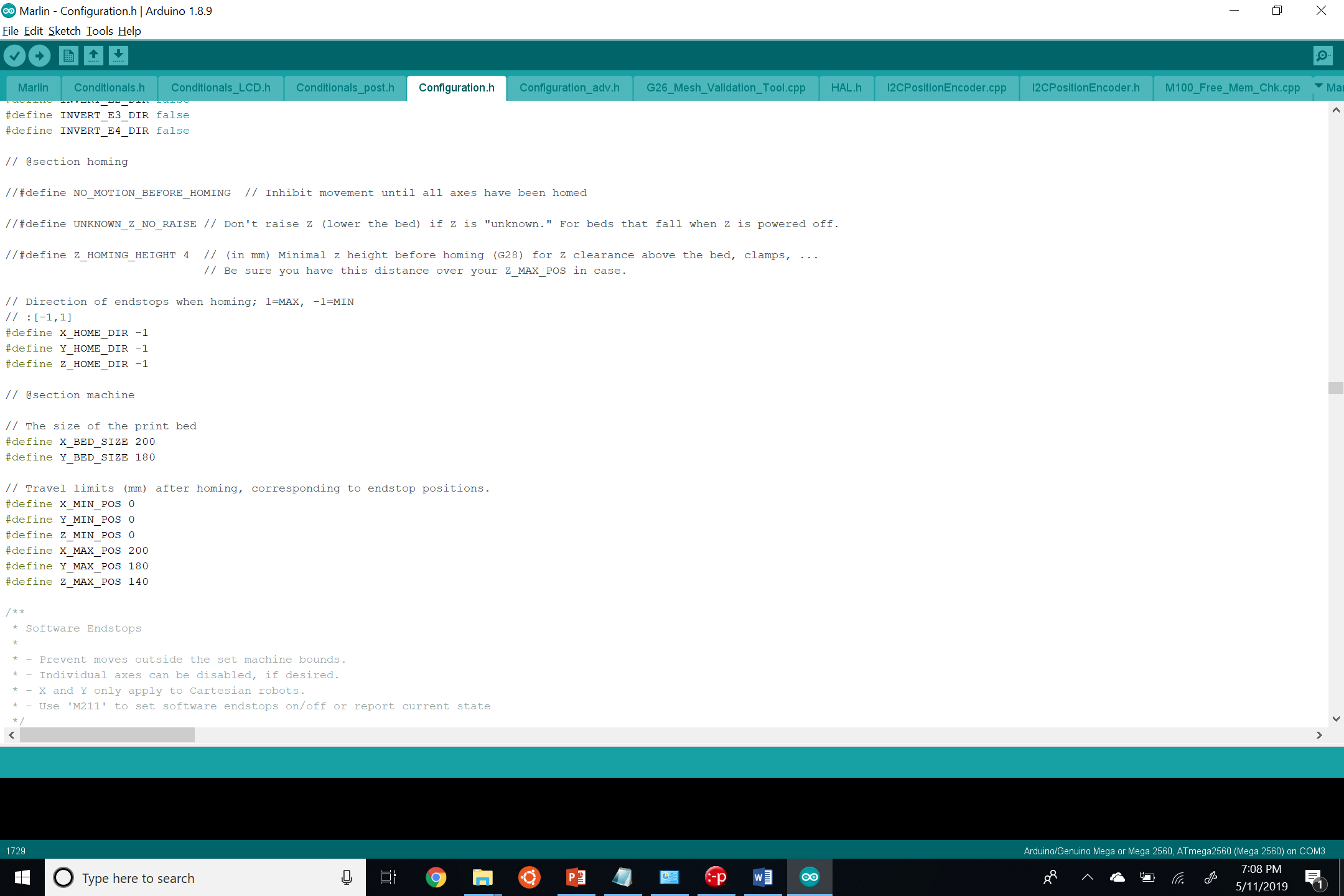
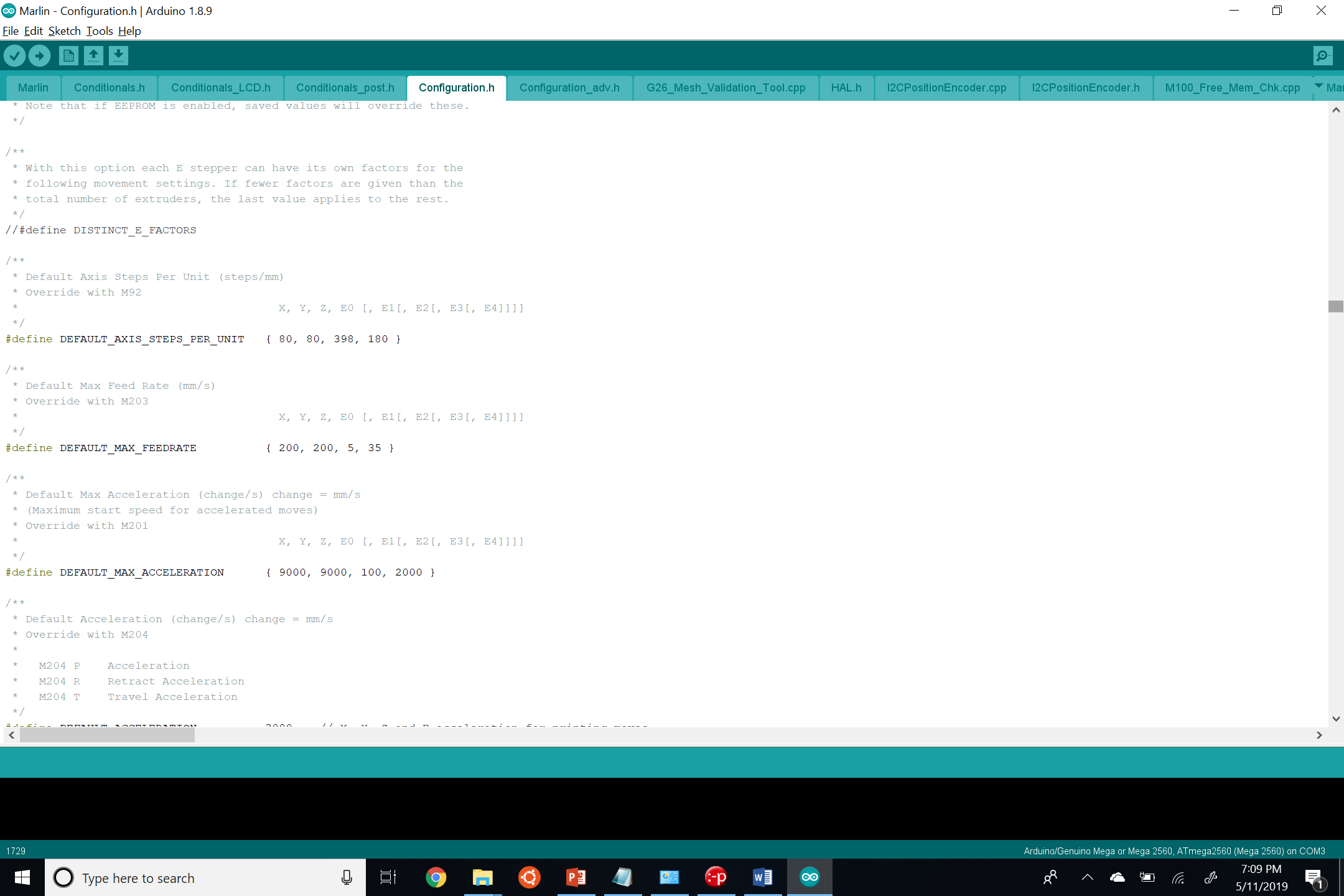
Next step was to wire up the motors and endstops to the RAMPS controller. Our team purchased an Arduino mega kit online, which came with an LCD with a board adaptor, A4988 Stepper Motor Drivers with heat sinks, RAMPS 1.4, and micro jumpers (Figure 5). The Ramps 1.4 controller mounts onto Arduino Board and the micro jumpers are outfitted across the pins placed in the X, Y, and Z partitions of the ramps board. In discovery, this was found to be a crucial step in the process. Initially, we didn’t insert the micro jumpers on the driver pins, which resulted in a difficult time in getting X, Y, and Z stepper motors to move correctly. We were able to conclude that this is due to the type of stepper motors the printer uses. The Monoprice is equipped with MOONS Type C17HD40102-01N stepper motors which use high precision movements. When referring to the wiring diagram (Figure 6) we realized that to all three pins under the driver needed the micro jumpers. This setup tells the driver to perform a 1/128 step size, which gives the motor a smoother transition along its pathway. (RAMPS 1.4) When we finally had the pins set up, we plugged in our X, Y, and Z motors. The Z motor has two wire sets to operate the Z axis. The Z-axis rods work in parallel through the movement of two motors running simultaneously to fit the pair of wires onto the Z driver pins. We needed to do a bit of rewiring. The pins were reoutfitted with a thinner pin-set adaptor as well as female pin sockets. We performed this step to the endstops as well. The final wiring we had to do was connect the endstops to their designated pins. Once completed, we had to join the power box, the Arduino board, and the heatsink fans. Once the components were lighting up, we could check our wiring after the Marlin mounted to the Arduino.

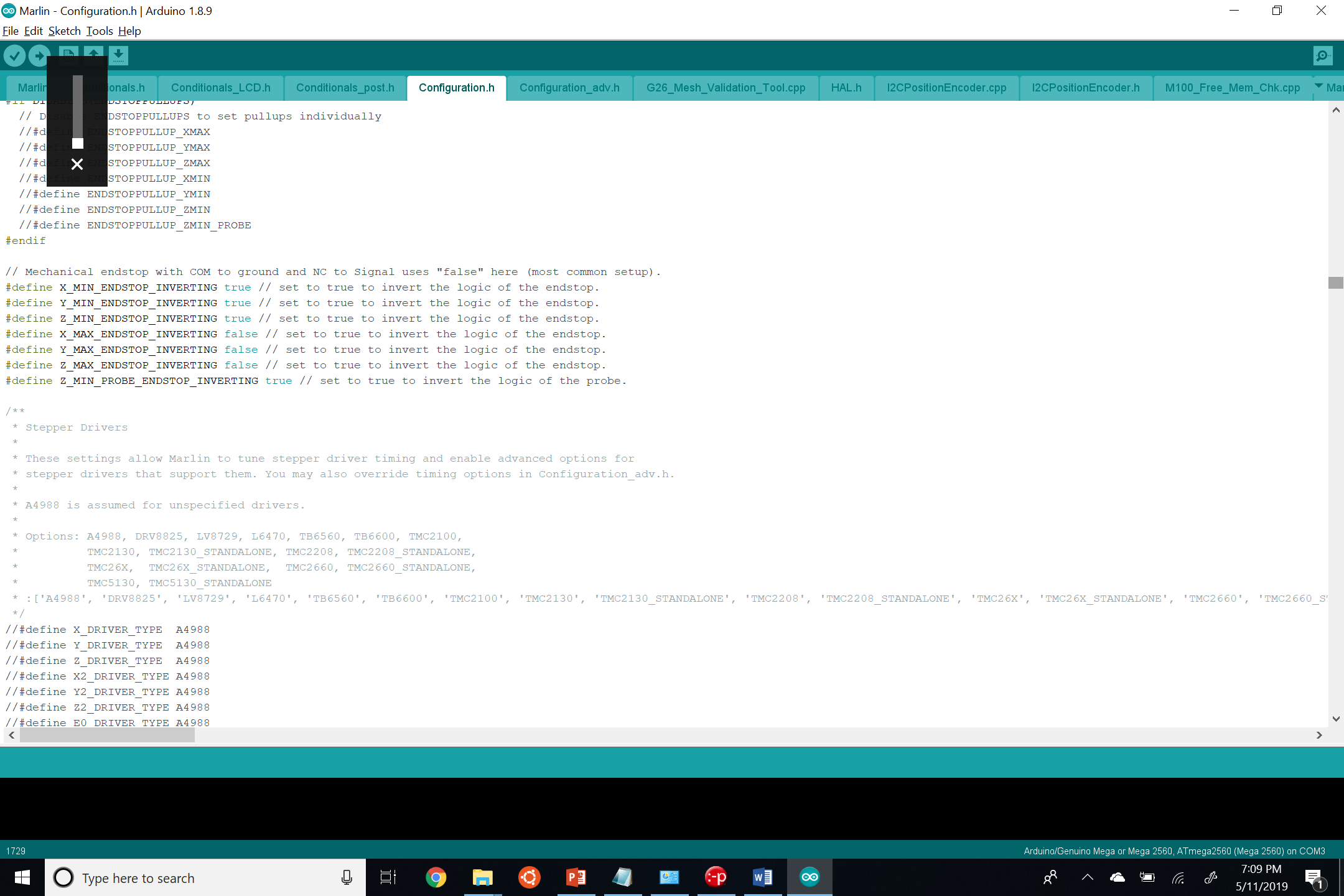
Figure 8: Arduino Mega 2560 with RAMPS 1.4 kit

Repeat of Figure 4: RAMPS 1.4 Wiring Diagram

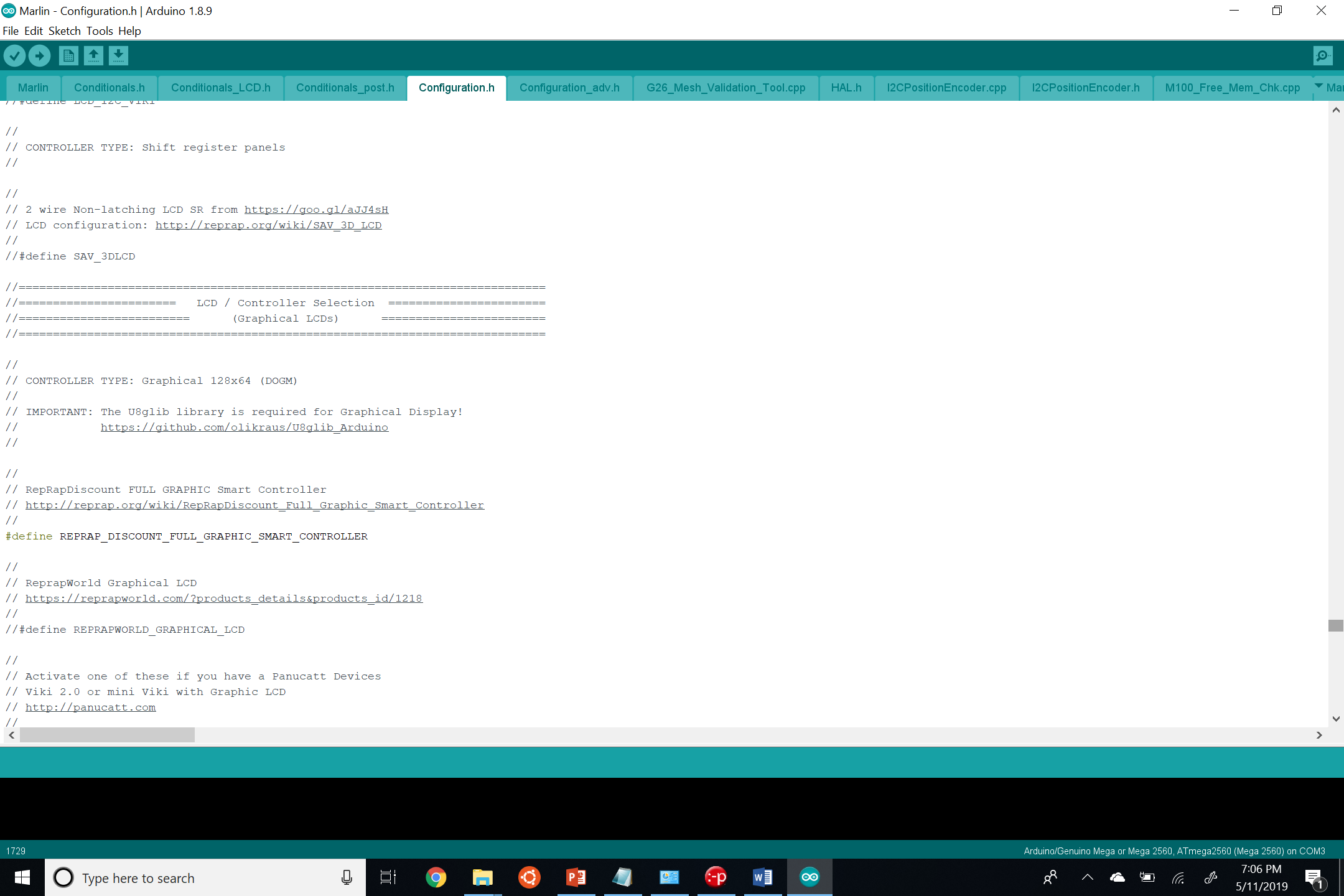
Marlin Configuration

At this point, the Arduino Board is very stupid and needs to be told what it is, what it’s hooked up to, and how to handle instructions given to it. It just sits there turns on and off and flashes a light to let you know it’s alive. Using the Arduino IDE, we were able to open Marlin and edit the firmware. The configuration.h in that one file that needed information about our machine frameworks. Luckily, we were able to find some help in our settings through another maker on YouTube who had had the right axis steps per unit for our model and machine. The following sections were updated:





To get the LCD interface working, we had to download its library drivers and add the configuration files into the Marlin folder. This step allowed our drivers to be able to recognize the LCD screen. The following is a screenshot of the modified LCD code in the configuration.h file, as well as it's hyperlink download for the LCD libraries.



Once we modified the header file, we then used the sketch function to compile in the Arduino IDE and then upload to the Arduino.

Slicer

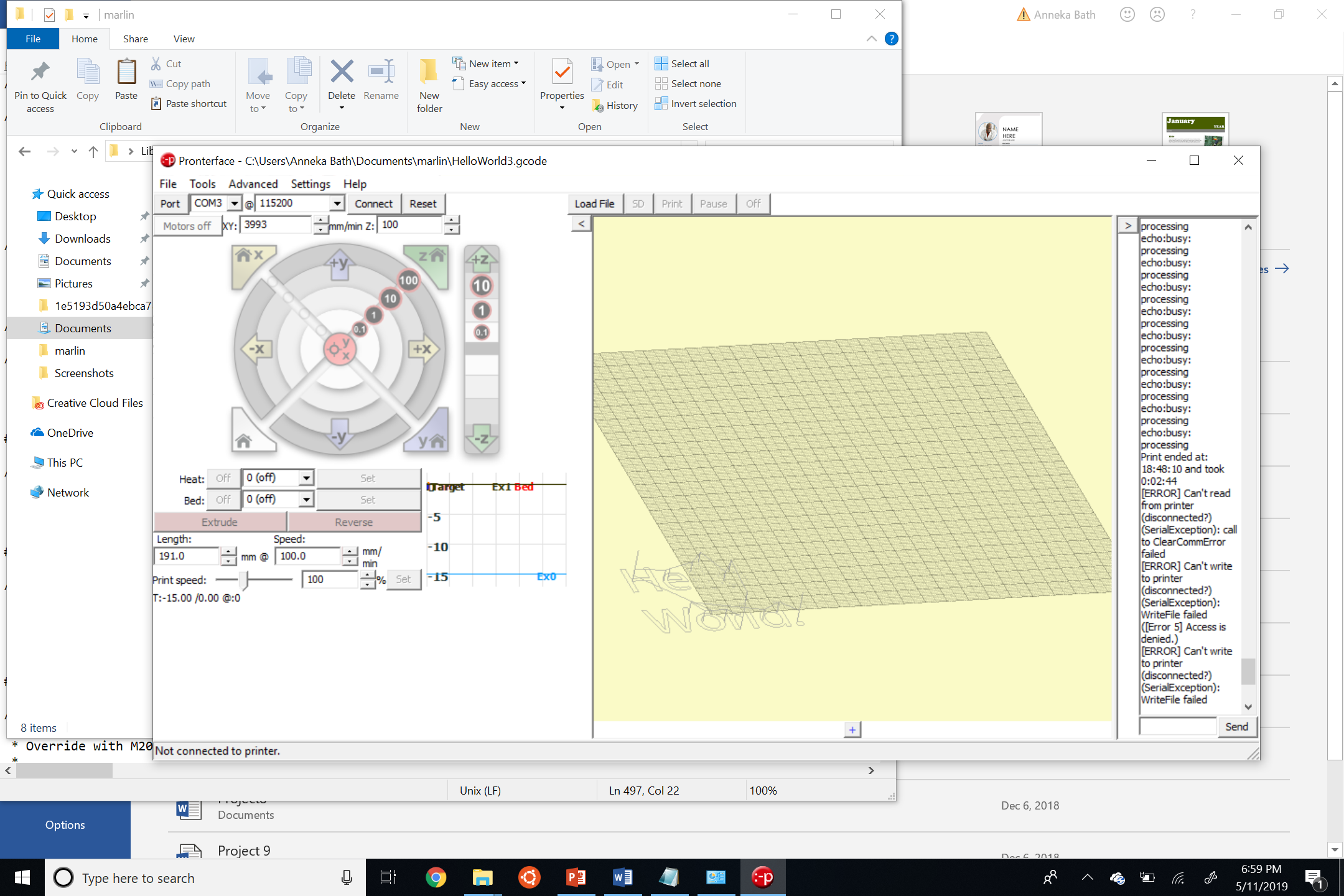
The last step in the build process was to check our wiring and stepper motors. The wiring and stepper motors use a slicer interface called Ponterface to test their functionality and precision. We encountered a few obstacles. Initially, we had reversed our Z-axis pins; this made the up and down motions move in the wrong direction. By flipping the pin adaptors over the ramps pins, we fixed the issue. We found our endstops were incorrectly set on the endstop pins; we corrected this by setting the correct adaptors onto the min position on the XYZ endstops on the RAMPS 1.4 board. This stage is when we were able to recognize and solve the micro jumper issue mentioned in the wiring section. After a lot of tinkering and trying to figure out the correct stepper movements, we ended up finding someone who had already done a ramps conversion for our 3D printer and had uploaded their Marlin configuration on (yet another) YouTube video. The user ‘inside the mind of matt’ saved us a lot of frustration in tampering this axis steps. Once the hardware seemed to be working correctly, we could start loading G-code and print our design. 

Figure 9: Pronterface Screenshot

**Conclusions**

Our most significant conclusion during this project was that this task was better suited for an intermediate CNC builder. For three Computer Science majors who were novices in this field, this was a much more complicated project than we had expected. We learned new software applications (Marlin, Fusion 360), coding languages (G-code), and a lot about instruction handling. We now understand how Arduino Mega 2560 and RAMPS 1.4 can control the movements of high precision motors. We got firsthand experience in working with the Marlin firmware, and its Slicer interface. We worked through issues and found just how helpful a robust open source community is. Without the open source community, we probably wouldn’t have been able to push our project through the many obstacles we encountered. In the end, we’re incredibly proud of the finished product. We developed, designed, and accomplished the task we set out to do. As we demonstrated in the class demo, we were able to cut out the words “hello world” from a piece of vinyl and transfer the design on fabric such as a t-shirt, using an iron.

## **Picture Sources**

Figure 4: <https://www.deviceplus.com/wp-content/uploads/2017/05/image005-02.jpg>

Figure 5: <https://www.monoprice.com/product?p_id=13860>

Figure 6: <https://images-na.ssl-images-amazon.com/images/I/71U3wmZY4IL._SL1000_.jpg>

Figure 8: <https://images-na.ssl-images-amazon.com/images/I/81KRAjPJQ2L._SL1500_.jpg>

**References**

CNC Code Programming. Custom Laser Cutting Minnesota, Sheet Metal Fabricators Minnesota, Carlson Manufacturing, Inc., 2018.

https://carlsonmfg.com/cnc-g-code-m-code-programming

G-Code. Wikipedia, Wikimedia Foundation, 10 May 2019.

https://en.wikipedia.org/wiki/G-code

HicWic. Universal Mount 525 for MPCNC. 30 December 2015. website. 1 May 2019. https://www.thingiverse.com/thing:1234989

HyperCube CNC: Drawing with Fusion360. 30 July 2017. website. 24 April 2019. https://www.youtube.com/watch?v=q8memLK8rdo

HyperCube CNC Toolheads. 30 July 2017. website. 1 May 2019. https://www.thingiverse.com/thing:2459624

Jbrazio. What Is Marlin?. Marlin Firmware, Jbrazio, 5 May 2019,

https://marlinfw.org/docs/basics/introduction.html

Matt, Inside The Mind of. My Marlin Settings and Octoprint EEPROM Editor Plug in (Ramps Conversion Series Conclusion). 16 September 2017. website. 25 April 2019. https://www.youtube.com/watch?v=AXIgT5ZwM4g

Mjacobs518. MPCNC Roland Drag Knife Holder. 17 December 2016. website. 1 May 2019. https://www.thingiverse.com/thing:1975507

RAMPS 1.4. 7 March 2019. 11 May 2019. https://reprap.org/mediawiki/index.php?title=RAMPS\_1.4&oldid=185110

Ronquillo, Romilla. “Understanding CNC Machining.” Thomas, Thomas Publishing Company, www.thomasnet.com/articles/custom-manufacturing-fabricating/understanding-cnc-machining

Tech2C. HyperCube 3D Printer/CNC. 4 September 2016. website. 1 May 2019. https://www.thingiverse.com/thing:1752766