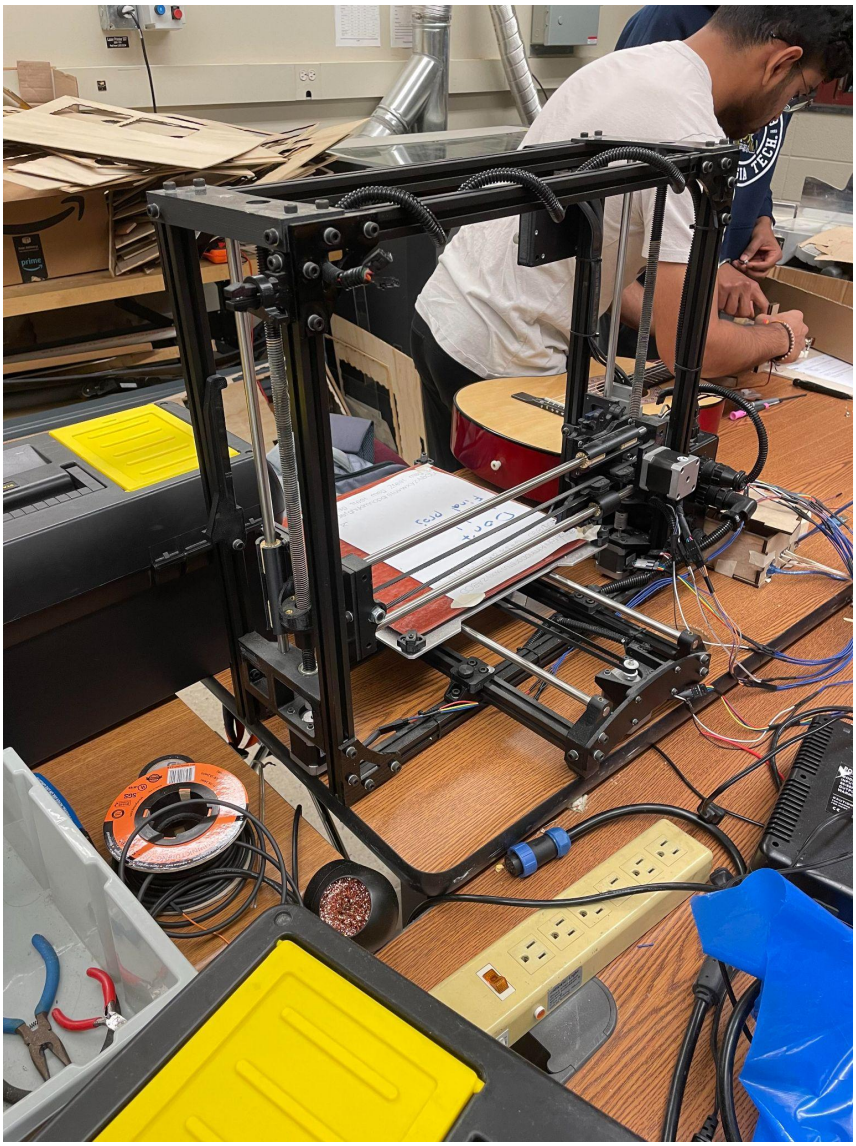
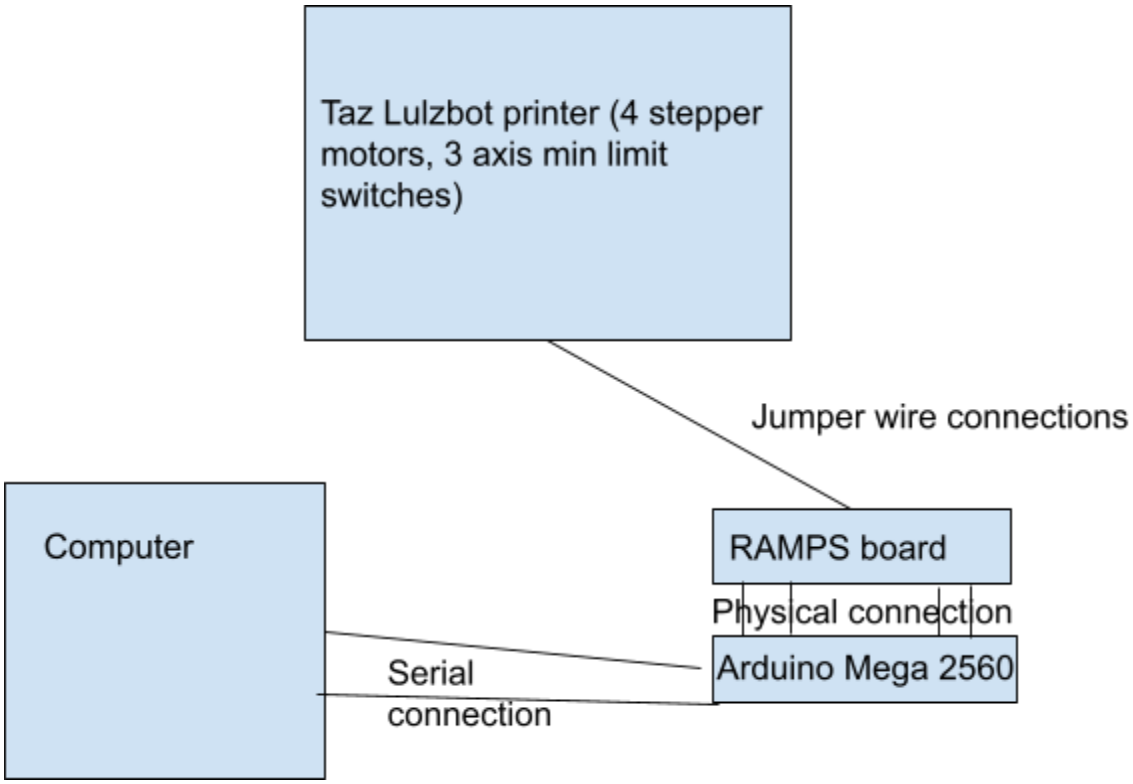
Making a plotter using a 3D-printer

Calix Tang



High Level Design

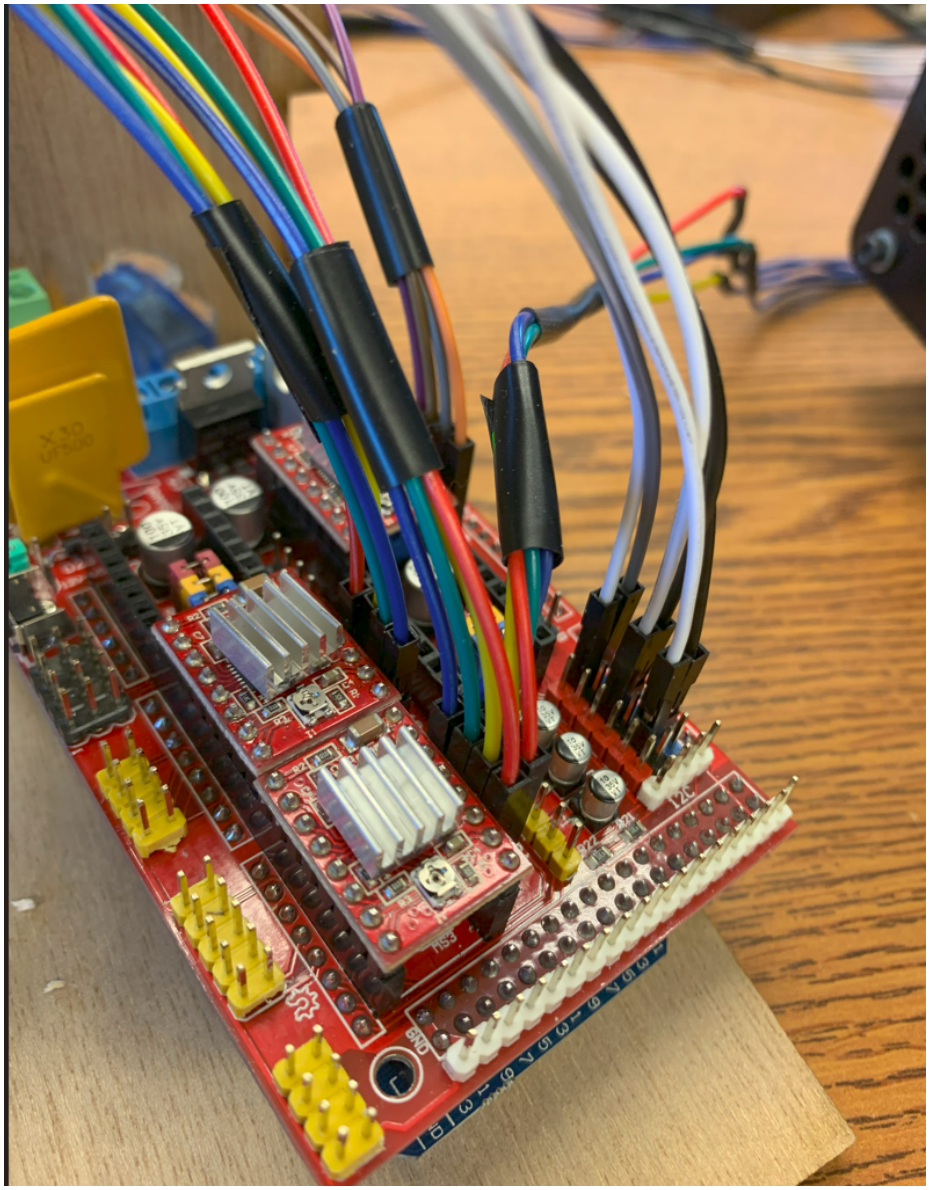


- The hardware components of your project

The Taz Lulzbot (big 3d printer) is the main hardware ‘component’ of my project. The gantry, steppers, and bed are really all I needed, but making those myself would have been far too expensive and time-consuming for me.

I needed a way to control the stepper motors. Peter suggested the RAMPS v1.4 kit, which I bought and ended up using. The RAMPS board itself shields the underlying arduino from high voltage and has integrated spots for wiring up stepper motor drivers and limit switches.

Here’s the board with all the necessary wires (for this project) plugged in:



A small but crucial component was the pen mount, which is described and shown in the parts section below.

- The libraries you used

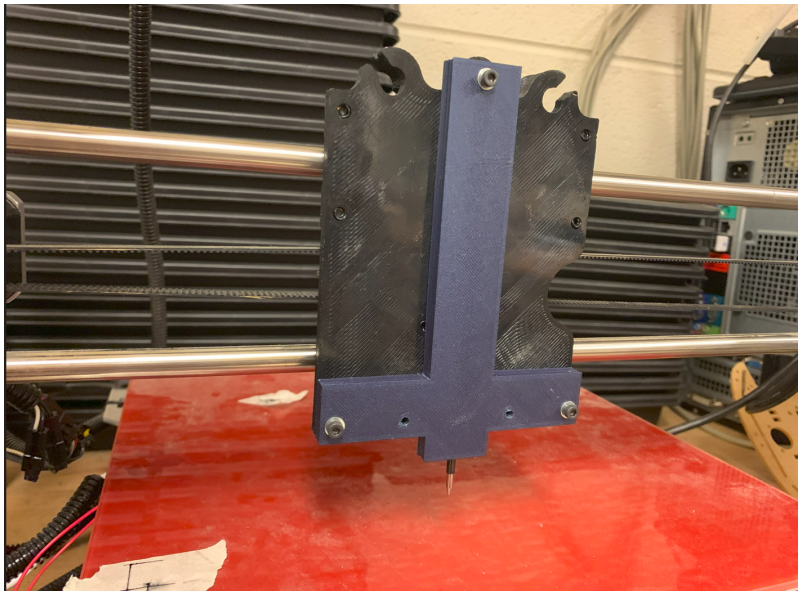
I used a RAMPS library (https://github.com/momostein/Ramps) to easily use the RAMPS v1.4 kit I bought. I had to slightly modify it because my x-stepper driver position stopped working, so I modified a header file in the library to make it use Q stepper pins instead.

This ramps library also used a Polulu stepper driver library. I looked at it to understand how it worked, but never modified it.

This wasn't a library, but a very useful resource I used in my project: a text-to-gcode repo (https://github.com/Stypox/text-to-gcode/tree/master) with a script to turn ascii into g-code (which I didn't use) but also a bunch of common ascii characters already in gcode (which I did use).

- The parts that you designed or made yourself

The only part I really designed myself was a mount to hold a pilot G-2 pen at the right depth for the plotter to use. It took 3 iterations to get the tolerances and mount hole positions right, but the end product works great. It holds the pen in place without any wiggling and is springloaded so that there's up to a 3/4-in range where the pen is held down and can write.



I also designed a small box and laser cut it to hold the ramps board in place.

- Skills you had to learn (beyond the skill demos) to accomplish your goals

I had to learn how to modify arduino libraries. Since I accidentally shorted the board at one point (oops), the board's designated position for the x stepper motor didn't work anymore. I switched a few pins around in the ramps library I was using to let ramps use the designated spot for the "Q" motor (either bed heating or extruder, I forget which) for my X motor.

I learned how to read and write g-code and eventually figured out how to use python to translate g-code into C (arduino) code.

- The iterative process you went through

I started the project not really knowing what to do or how to do it. After I decided to use the taz lulzbot, I bought a ramps board and messed around with other printers in the room while waiting for it to arrive. By using PuTTy to send g-code over serial, I could make some of the small white printers (iii-3d?) to move to any legal position I wanted them to.

Once my RAMPS board arrived, I wanted to try it on the lulzbot. A few problems I ran into immediately were: (1) I didn't have a 12V power source on hand and (2) I had no idea how to wire up the stepper drivers to the actual motors. With Peter's help, I salvaged a laptop charger that could supply enough power for the RAMPS board and motors. Using the long female-to-female wires I soldered together, I tested first with one stepper driver and motor and just made it move back and forth with the ramps library I found. After making the pen move back and forth in all 3 axes, I added in wiring for the limit switches. When I figured out how to get a signal from them, I worked on a homing script (making all 3 axes move until their min-travel limit switches were hit). Finally, I wrote functions for easy control over movement. These included functions for making the pen move up above the plane of the bed, for making the pen move down for solid contact with the paper, for making the pen move left and right without going out of bounds, and for moving the pen around with relative coordinates to where it was before.

I was pretty indecisive about what to do with the controls I made. At first, I tried using some pushbuttons and potentiometers to manually control each axis of movement, but initial testing on a breadboard already showed that I couldn't read signal reliably. My other idea was sending text over serial for the arduino to process and turn into control signals, but I initially didn't want to do this because of how tedious making each character in c code would be. However, when Peter showed me an online text to g-code converter, I changed my mind. I found a github repo with all keyboard-writable ascii characters already in g-code, so all I had to do was convert these into c-code. It turned out to be easier than I expected, and within the afternoon I had header files with functions for drawing each character. I linked these together with another header file with a big switch statement that chose the right draw function for an input character and it worked!

Minor changes after that included allowing for variable width of characters instead of making each a fixed width (it looked goofy) and making the printer 'wrap around' when reaching the edge of the page.

Here’s an example of what it prints now (all printable characters on my keyboard)