Story Board: Laser Cutter

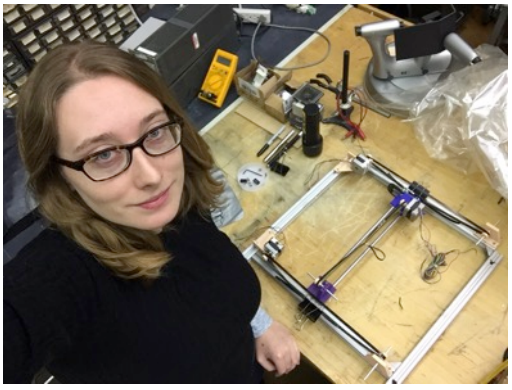
Larisa Thorne
2015-12-09

As my Term Project for the course 15-112 at CMU, I chose to make a laser cutter. This project will be divided into two parts: hardware and software.

For demos: use ArduinoPySerialTest2.ino + testSerial2.py (Arduino: servo), and LaserCutter4.py (PIL/Pillow, Tkinter).

I want to take advantage of the current DIY age of open source and affordable parts (most recently, Raspberry Pi's $5 microprocessor "Zero"!), and use this as a building block for later more advanced projects. I want to build a laser cutter because the basic parts (XY table, lasers, and programming) are sort of open-ended and come up in a lot of applications. These might include cases where one would want (1) extreme precision/control of motion or (2) control something without a human in the room. And example of the latter could be moving a radioactive target in/out of a ~12GeV electron beam's path.

For a more high-level treatment, see program's readme file.



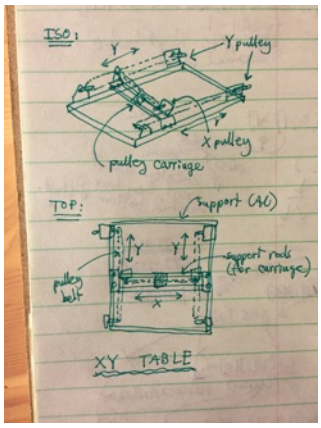
Hardware:

From "Project Proposal":

"On a most basic level, I will need to construct what is known as an 'XY table', and affix a laser to the carriage.

The XY table consists of an aluminum support frame, with two stationary Y stepper motors, which will support a moving carriage for the single X stepper motor. The Y stepper motors are attached to the aluminum frame, and the X stepper motor to the Y timing belt pulley, with specially 3D-printed parts. They will move/rotate a timing belt pulley.

The 'platform' which the two Y stepper motors support will be a rigid structure, supported by two rods. These rods will support the carriage, which will be attached to a point on the X timing belt pulley, which will contain the laser. "



For testing purposes, I choose to start by learning control of servos over serial with Arduino before moving on to stepper motor control. This is good practice in writing a Python script that will take a list of numbers (in this case, angles between 0 and 180) in a specific format ("###,") and feed them one by one to the Arduino. The goal is to make the Arduino do as little extra work as possible.

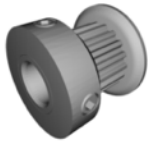
Having satisfactorily completed basic Arduino serial servo runs, I add the Adafruit Motor Shield v2.3 on top of the Arduino, and supply 12VDC from extral DC regulated power supply (since the power Arduino gets via USB is not enough to power the steppers).

Components (currently constructed), where * denoted 3D printed part, and the corresponding picture is a screenshot of the .OBJ file

- Structural:
  - 4 x Aluminum "X" bars:

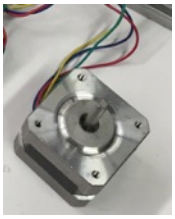
  - 3 x XL timing pulley belts

  - 6 x Timing pulleys, matching stepper motors and belts*



- 2 x Window hardware (roller)

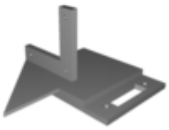
- 3 x Stepper motors (12V 4-cable bipolar unit, where 2 cables per coil)



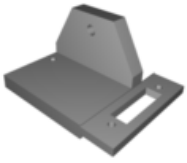
- 2 x Y-Stepper-to-Bar adapter*

- 2 x Pulley-to-Bar adapter*

- 1 x X-Stepper-to-Pulley adapter (top, bottom)*



- 1 x Pulley-to-Carriage adapter*



- Electronics:
  - 1 x Arduino Uno

- Jumper cables (+ associated soldering supplies)
- 2 x Adafruit Motor Shield, v2.3 (for stepper driving)

- Wishlist:

  - Heat sink/fan
  - Laser that will burn paper (>20mW)
  - Laser driver

Some pictures of current progress in assembling:

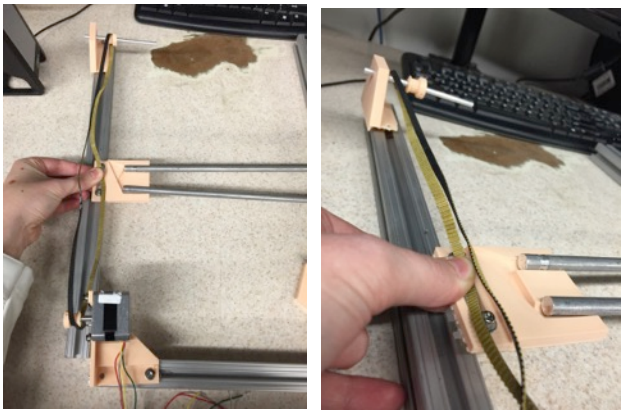


Figure 1:
Left: Demonstrates how the X carriage will be attached to the Y stepper.
Right: View of the window hardware roller used as a sliding mechanism for the X-carriage-to-Y-Pulley motion.

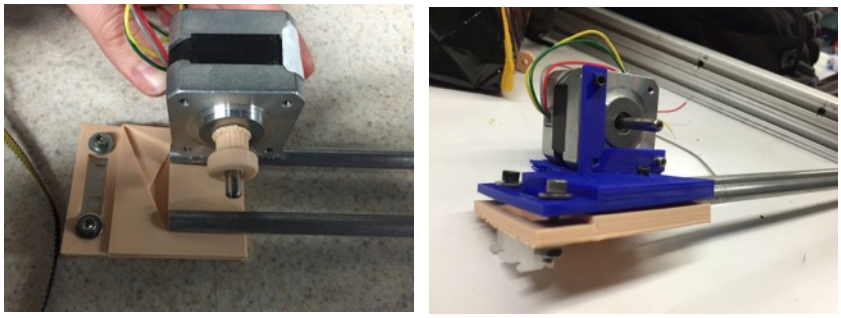


Figure 2:
Left: How the X stepper will be affixed to the X carriage.
Right: X stepper mounted to carriage.

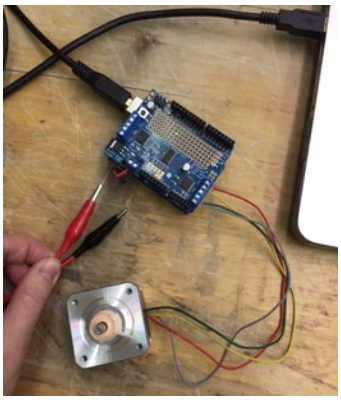


Figure 3: Arduino + Adafruit Motor Shield, v2.3. (See video)

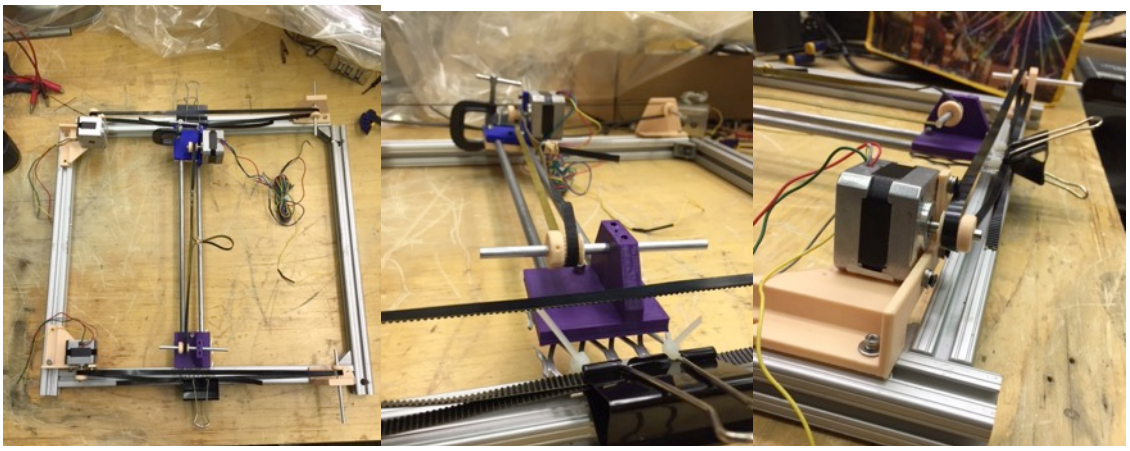


Figure 4:
Left: Complete XY table. Middle: X carriage. Right: Y carriage.

Figure 5: Tapped pulleys, added set screws, so no sliding.

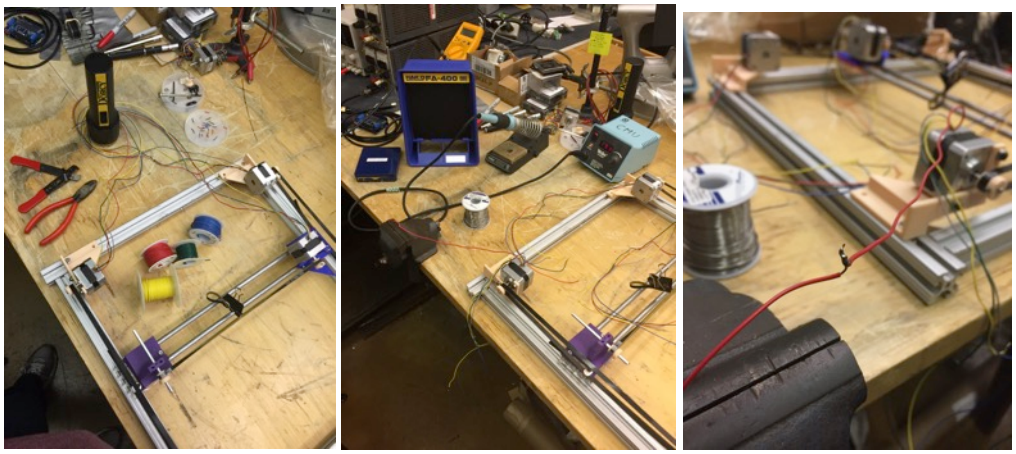


Figure 6:
Left: Adding extra wiring. Middle: Preparing to solder connections. Right: Solder connection.

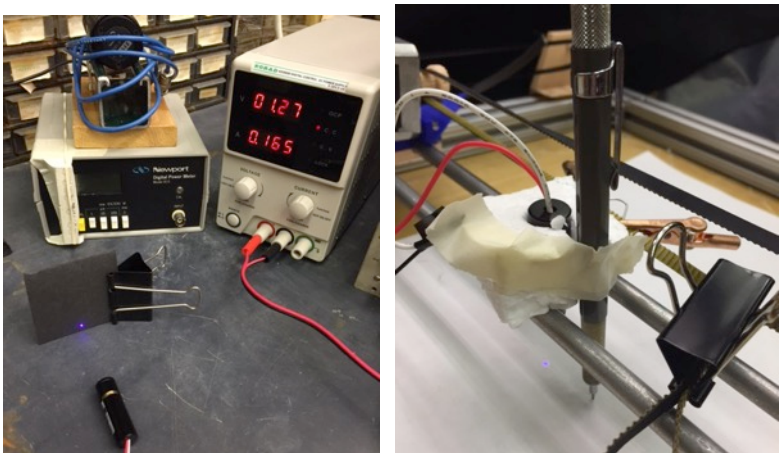


Figure 7: Left: Testing laser. Right: X-Carriage, with laser (L) and pencil (R).

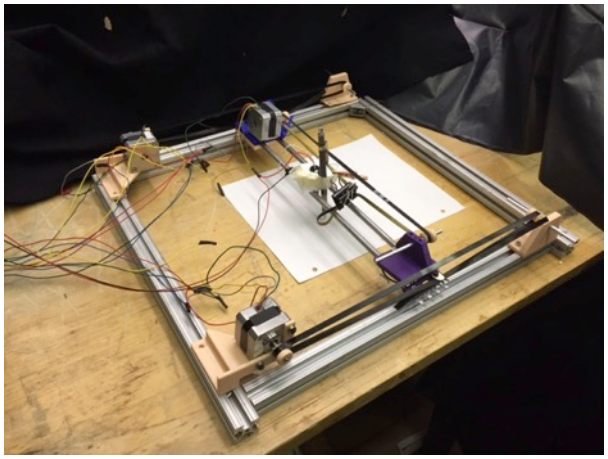


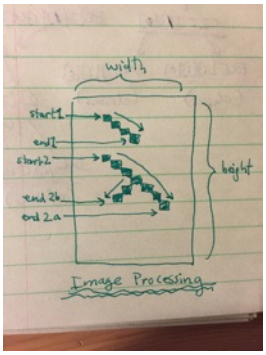
Figure 8: The finished table.

Software:

From “Project Proposal”:

“The role of the software is to take an input bi-color image (black and white), and to do some operation on it that will translate to motion of the carriage containing the laser head.

The method for translating bi-color image to Arduino movement instructions involves the use of backtracking. First, create a 2D list containing pixel location and color from original input image. This then becomes a ‘backtracking/floodfill’ problem, different in flavor than previously encountered: we want to go through each pixel of the image and ask if it’s the beginning of a line. If so, look at the adjacent eight pixels for more black pixels, with indicate a continuation of the line in that direction. This involves recursion, where the base case would be arriving at a black pixel that is either (1) surrounded by non-black pixels or (2) any black pixels around it have already been visited. The recursive case allows us to ‘climb’ back up the line to investigate branches, where our original line split into multiple lines.



The total path traced will be stored in a list, which will be handed to the Arduino when it’s finished, so that it’s not done in real-time.”

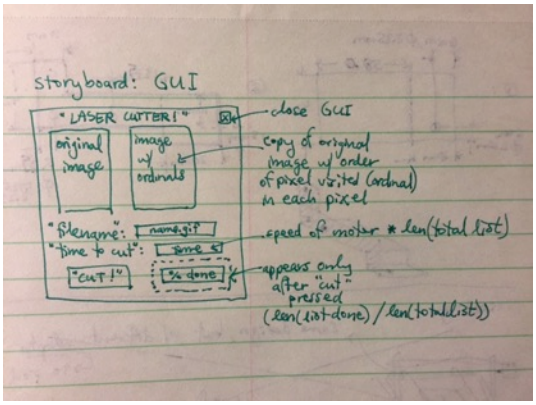
I wrote the software code in steps (version denoted by the # in file of form “LaserCutter#.py). A description of the goal of the version will be provided at the top. Simple test images will be used, which will be located in the same directory. I will create separate files for Arduino control (version denoted by the # in “ArduinoPySerialTest#.ino”).

Created an additional function within main code to format the current pixel path: the first location is in (row, col) format; each following pixel location is (drow, dcol). Split the path locations into two independent components, X and Y, and feed them (alternating) to the Arduino, which relays the command to the two stacked Adafruit MotorShields.

GUI information currently includes: file name, dimensions in terms of pixels, estimates for cut time (based on serial sleep times, number of pixels); and after cut initiated: countdown timer tells how much time left. Five buttons: one to open file, one to start the cut, one to stop the cut, one to save the pixel path of the cut image, and one to return to the welcome page. At start, “cut”, “save” and “stop” are disabled.

Once run begins, “stop” becomes available and “open” is disabled. When run done, “stop” is disabled, but “save” is enabled. “Stop” kills both the stepper motion and the laser, for safety.

How the GUI should look:



How GUI currently looks:

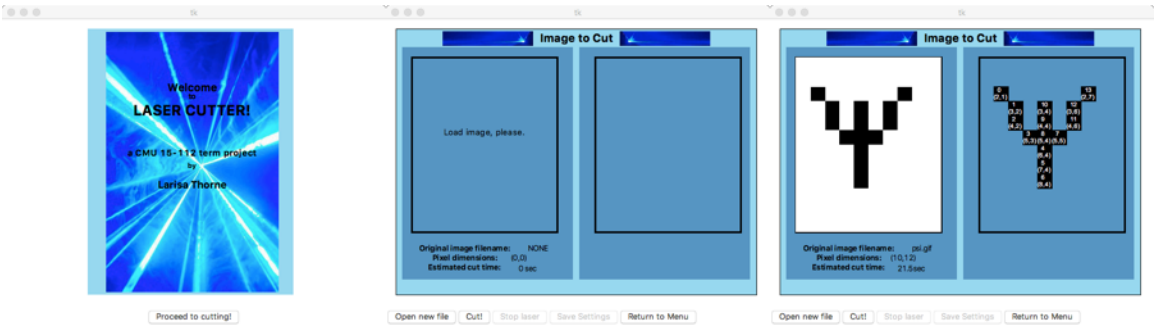


Figure 1:
Left: Welcome page. Middle: Load file request. Right: Loaded file, ready to cut.

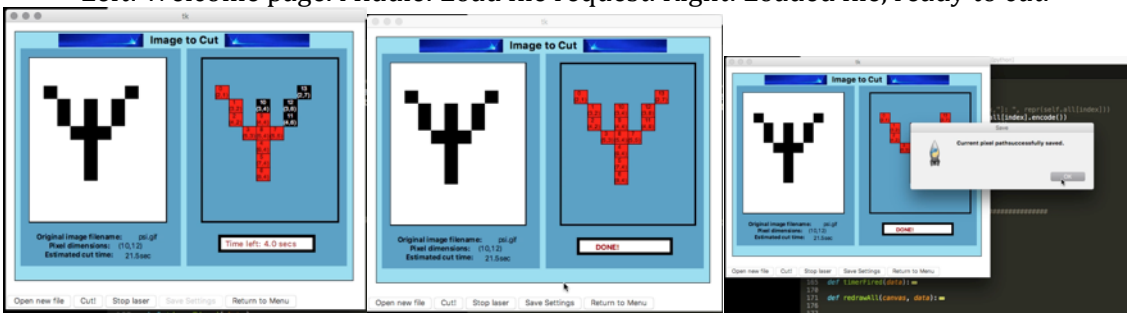


Figure 2:
Left: Cutting in progress. Middle: Finished cut. Right: Saving pixel path.

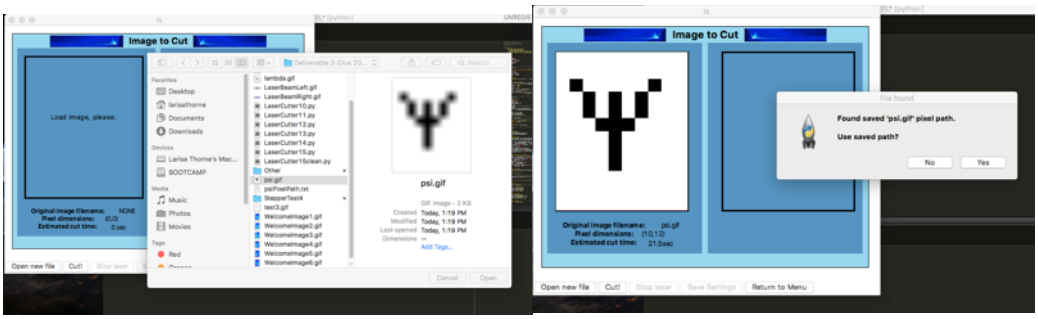


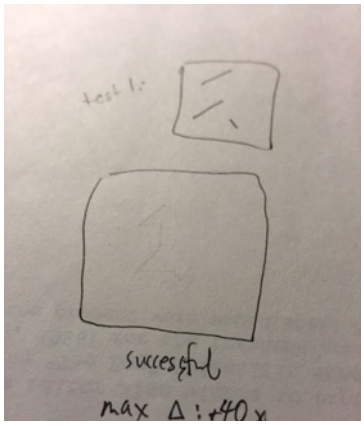
Figure 3:
Left: In a separate instance, loading same image. Right: Sees that pixel path already exists, asks if should use it.

Improvements I can make:

- No pencil: drags/snags and draws inaccurate representation of what laser is doing.
- Change track design:
    - Instead of having x platform attached to pulley/rollers, maybe can have y's sliding along rods.
    - Currently, sometimes the jerking motions from x carriage throw the y's off track.
- Use different connection, other than serial.
    - Seems that serial connection must sleep at least 1.5 - 2 seconds in between movements, or you get either nothing, or steppers that won't stop rotating.
- A laser that actually burns through paper!
    - Currently have 20mW of 405nm. Not enough to even scorch black paper. Research suggests need 50mW, maybe even 100mW. See laser pointer forum link in "Resources".
- During TP User Study, the only suggestion that was made was for me to add a feature where I can allow the user to edit the image to be cut from the interface (ex., add a pixel). It sounds like a great idea, but I couldn't integrate it in the time that was left. [Most people thought I did an ok job; no other suggestions.]

Image results:

***Disclaimer: the pencil snagging and track derailing were real problems here! The laser was doing what the serial said, but the pencil has not been reproducing properly. Even between two separate runs of the same instructions, there were differences. I've attached the best I could get. Seems that it's easier to get best results with smaller step size and simpler/smaller images.



This does exactly what I told it to: need to keep in mind that I'm artificially blowing up the images, and I am intending to not visit the same pixel location twice, because the laser shouldn't be resting in the same position more than once (will burn what's under the paper).

Here are two of the cases where I had problems in just one location, but rest was fine. The pencil wasn't quite touching the surface of the paper in the beginning of the second drawing (the surface they were mounted to was uneven; I tried to compensate, but not too much because a stuck pencil is worse than a snagged one). Also, I stopped the first run after I saw that it had snagged again, but I'm including it here because it shows a better view of the top part of the image.