CDEFO Week 10 (#9)

Timeline:

Week 8 - End:

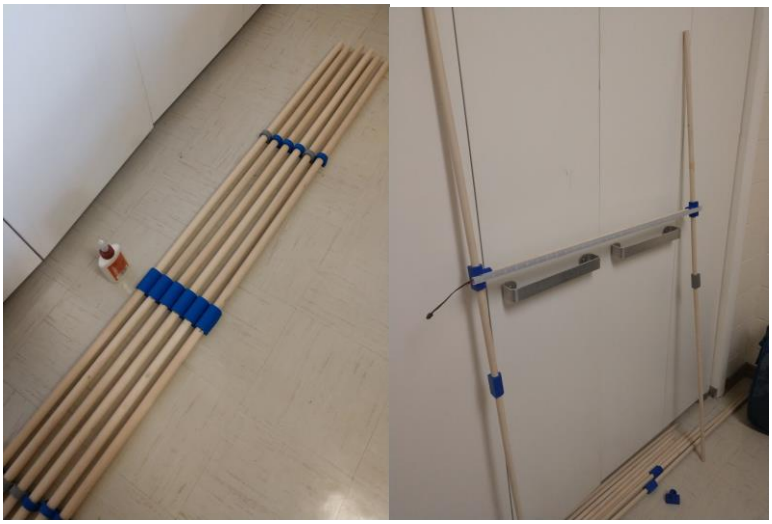
- Build user experiences, curate content, and test stability. (until the end)
  - Write any remaining C++ functions
  - Make the experiences pretty and bug free
- Building the user experiences is probably going to take the longest, as it is going to have some QA involved to make sure they are actually enjoyable. Lots of talking to users.
- Write documentation for the C++ library (until the end)
- Prepare for demonstrations

Point by point, I wrote the music visualizer C++ function, and also built the prototype for the actual module itself. I'm going to end up using it demonstrate my progress as my Alpha prototype. In terms of what I have for my alpha prototype, I have 1 LED frame, 1 Curtain frame, the CDEFO box with its ability to power the 5 LED strips I plan on using, custom cabling for the LED and the Music Module. I also have a the same little inkwell I've been using as a lightweight NFC tag holder. The functions I have prepared for the Alpha Prototype can record audio and video, play it back, visit websites, play/search for spotify tracks, sync up with music being played, and accept keyboard input to trigger events. That's pretty much all I need to demonstrate my personal user experience.

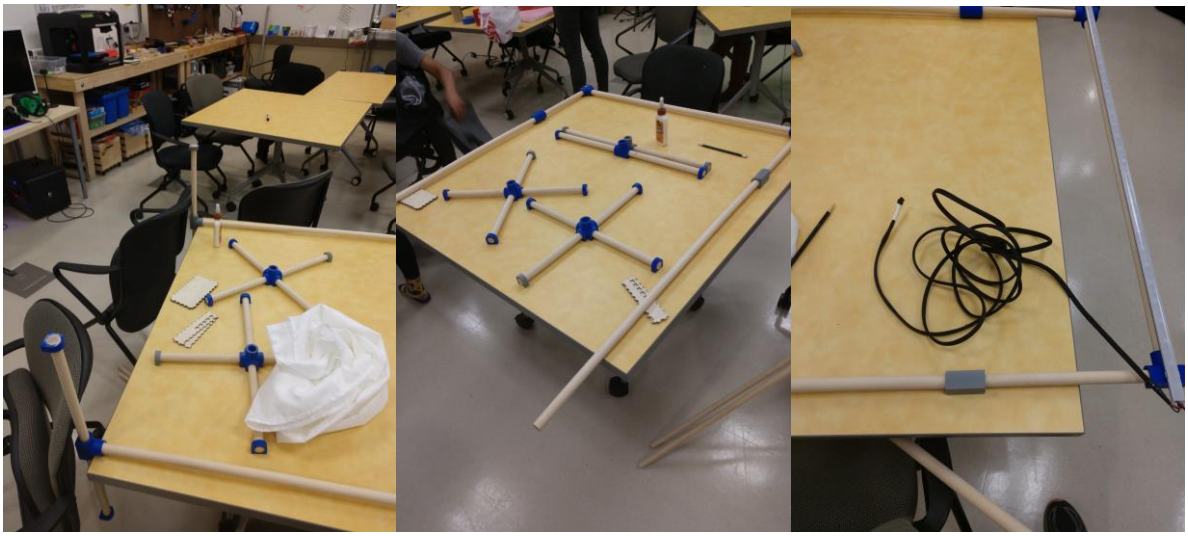
Also, in order for the visualizer to work, it can be blocked by anything, such as another LED strip running its lighting script, so I finally merged the TimedAction protothreads into master. The only issue with it, though, was that TimedAction was written to accept function pointers without arguments, so I had to go in and do a little tweaking to get it to accept a pointer to an LED strip object and a pointer to a structure that stores all of the necessary values (i.e. pointers to color arrays, unsigned ints for volume, character arrays for the lighting scripts). This was a lot simpler than I thought it would be originally, as the TimedAction library actually isn't all that complicated, it's pretty much just a wrapper for using millis() to implement delays instead of delay() itself, since delay() blocks the whole program from running, which would prevent concurrency.
I'm also almost entirely finished the Python script, and I have to say, since I was really just making a serial monitor which processes the content it receives. Ir's braindead easy, really, the hardest part about it was relearning regex to pull integers out of strings. Considering how involved tweaking the musical visualizer was, though, it definitely made up for how easy and wonderful it's been to code in Python. It's actually been so great that I'm considering expanding the EQ function to accept input from the Python script to provide 7 discrete frequency channels so that I can make a graphical EQ instead, and one that will ultimately be even smoother because of the increased amount of data. Right now, the Sparkfun Sound sensor that I have just outputs the amplitude(volume) as a value between 0 and 255, the presence of sound as a binary, or the audio wave itself as a wave. Now, in order for my to make the EQ without Python, I'd need to implement a Fast Fourier Transform or the YIN function on the Arduino using the audio output from the sensor (I'd also need to "bias" the analog input of the Arduino to 2.5V, since the envelope of the audio output is -2.5V to 2.5V, and the negative voltage can destroy the Analog-Digital Converter on the Arduino). Every FFT I've seen though is far to large and far

too complex for the Arduino to do by itself, let alone when it's already running several other routines. The YIN function is the same story, however it *is* possible to run it on an Arduino, but only when literally nothing else is running. What I can do, though, is do the processing in Python using my laptop's microphone and then transmit it over serial. It would be fairly simple, given the modularity of my Arduino sketch now that I've implemented protothreading. ANYWAYS… as a result of all of this experimenting/research, I spent around 12 hours this week on coding alone.

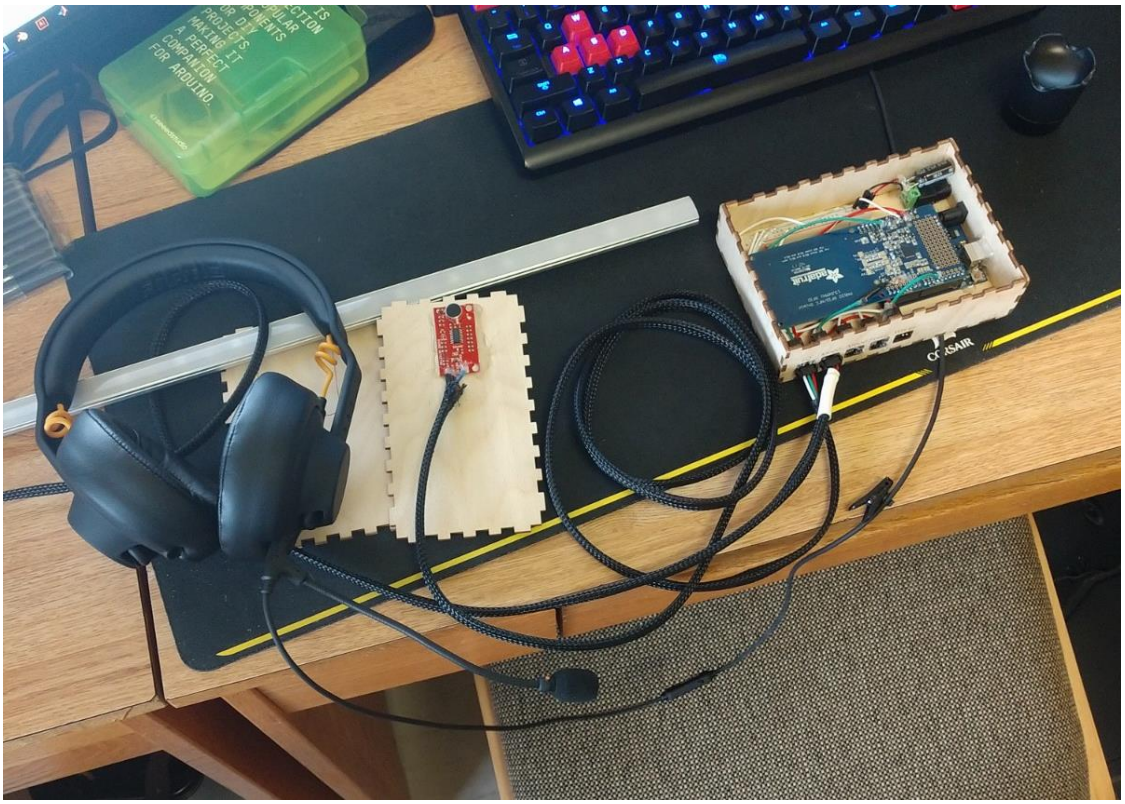
In terms of building the frames, It'd make more sense to just show you photos (I forgot to get photos of me making the cabling, since I didn't want to have my photo near the soldering irons, but the gluing of the frames is still there, cabling and all, along with the testbench I'm using for the Arduino). The whole building process took about 6 hours, not including dry time.



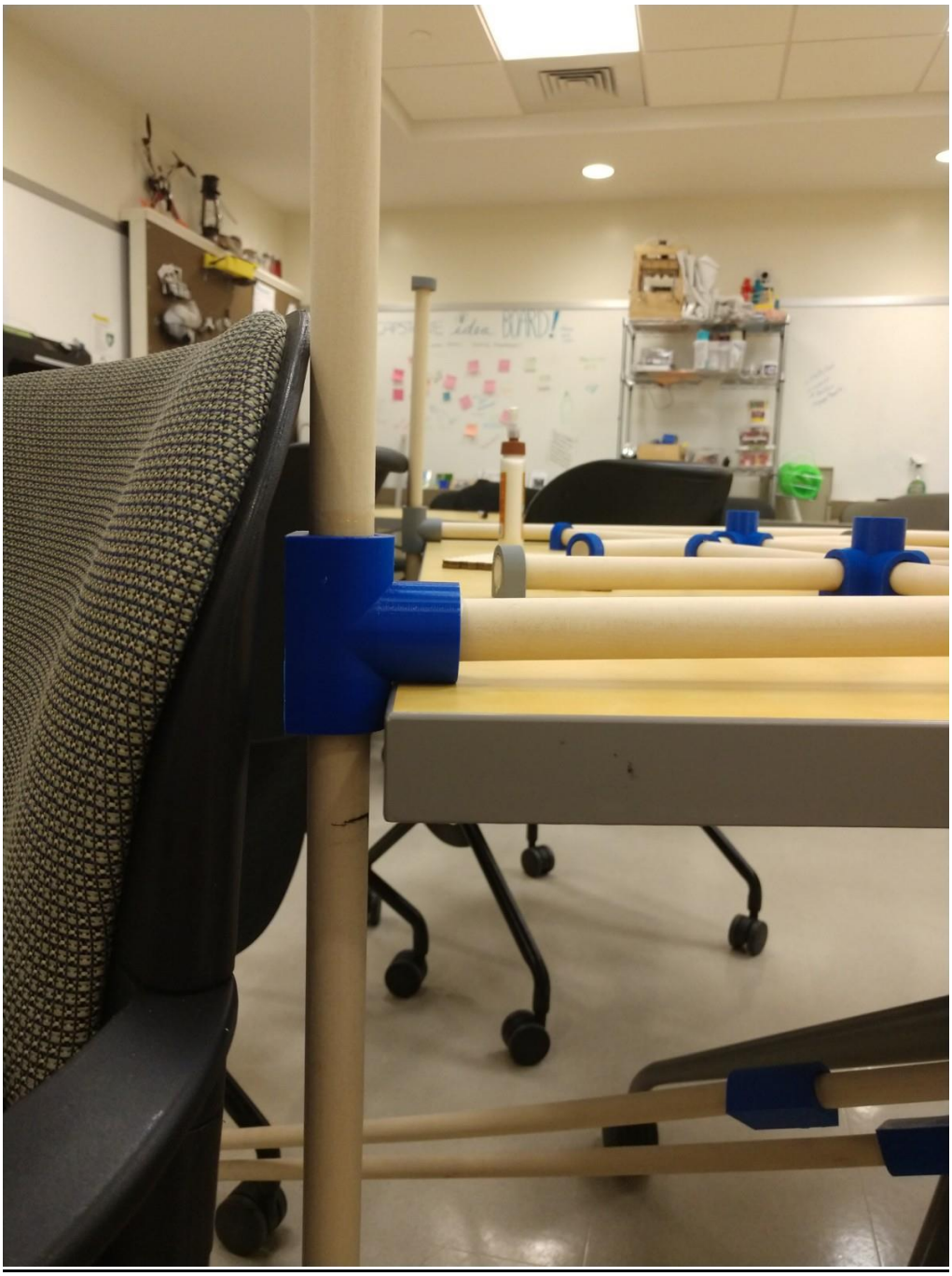
Gluing together most of the curtain masts, and the cross pieces for the LED frames in my dorm.



Gluing the feet in the DCC lab after some of the custom cabling.



The testbench for the Arduino, I used my headphones as a speaker in order to test the microphone so that I wouldn’t disturb my neighbors. The extra long cables I made make this so much easier too.



Using chairs as clamps because DCC doesn’t have clamps