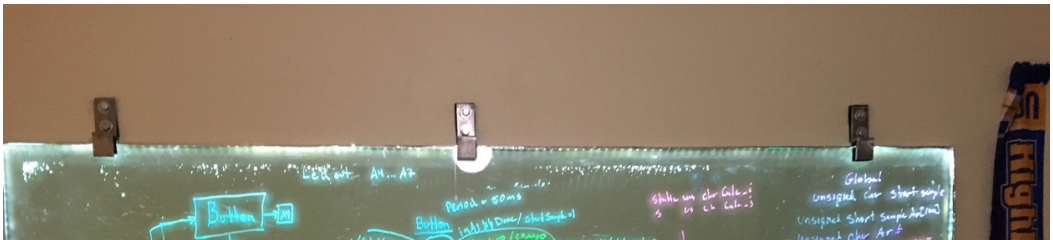
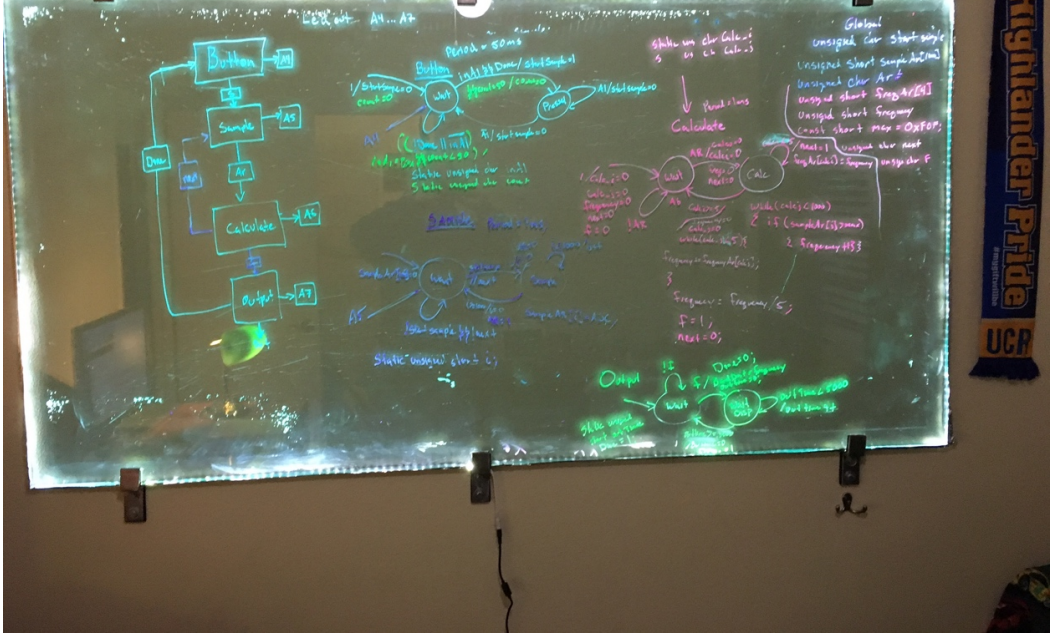
Custom lab report
My project consists of a guitar tuner. Using the ADC, a microphone with opamp, and a low-pass filter for input. For the output I ended up using the 7 segment display to show the note.
My proposal was to build a guitar tuner that was to include a LCD screen, a rotary switch, and two speakers.
When I initially proposed this project I did not know the details required to convert an analog signal into a digital signal. Initially I planned to convert a speaker into a mic, but I quickly realized that the added complexity of repurposing the speaker and using a FFT without having experience with either was a mistake. I instead ordered a broken out microphone with preamp built in. This gave me the opportunity to tune the ADC to work with my microphone and created my first build upon.
Researching analog signal conversion led me to the Fast Fourier Transform and two open source pieces of software called KISSFFT and FFTW. More than a week was spent researching this code and attempting to implement it without success. I had developed an algorithm that I thought would work, that sampled the ADC and then counted the number of times the signal crossed the DC offset point, but after several days of testing and correcting bugs, the output from the algorithm was still inaccurate and unpredictable. The second algorithm I developed was much simpler. After using a ADC program to observe the microphones response, I was able to come up with a maximum ADC value. My algorithm samples the ADC at 1 ms, if the ADC value is equal or greater than the max ADC value than a 1 is written to that element, if it is not than a 0 is written. Once the sampling is complete, the algorithm counts the 1's and that is the estimated frequency. Using this algorithm, I was able to interpret the incoming signals with a usable resolution, and completed my second build upon. Two other students were doing a similar project and were not able to get a working transformation of their signals, so I gave them my algorithm to use. The third build upon was outputting a note (A,B,C,D..) to the 7-segment display. My fourth build upon used my signal algorithm as a sound switch. This switch required 50 consecutive high ADC values before the machine starts to sample. My sound switch made the usability of the tuner much nicer as there are no buttons that you need to push to sample. My fifth and final build upon was a low pass filter. I designed this filter to have a frequency cutoff at around 480 Hz, this greater improved my accuracy in the conversion.
Most of the bugs encountered had to do with implementation of the signal transformation and inaccurate calculations. I used a series of LED lights and a variable named debugLED to keep track of what state machine I was in, and in some cases, which logical instance I was in. I fixed these bugs by rechecking my calculations and in some cases reducing the complexity of states.
The algorithm that was developed does not have very high resolution. High frequency samples tend to be less accurate than low frequency samples. This could possibly be resolved by using a 10KHz sampling rate, and averaging the samples.

To use this machine, simply power it up and play a note into the microphone. The 7-segment display will then show the note that is being played.

Link to youtube video:
https://www.youtube.com/watch?v=PaREa1fg0R8

Original state machine (not used):





Simplified state machine (implemented version):