Building a synthesizer for this semester's Electronics I final project was no simple task. However, the principles involved in its construction are. In this application note, I will discuss the planning stages and how the synthesizer works at a granular scale.

1 Planning the Synth

During our first lab of the semester, one of the tasks was to construct an astable multivibrator with an oscillation frequency of 500Hz. Before building it, I did not have a great understanding of how they worked, but once I worked through the math, I was amazed. Using fairly imprecise components, we were able to achieve an oscillation frequency of 507Hz, not too bad.

I grew up watching videos about old school computers and synthesizers, specifically, the Commodore 64's SID chip. I became familiar with simple waveforms, and high school I played around with Fourier series on my graphing calculator to construct these waveforms. So when I was able to quickly create a square wave with only a handful of readily available parts, my imagination went straight to synthesizers.

In February, Elise and I started making schematics to create more waveforms from the astable multivibrator. I used an integrator to turn the square wave into a triangle wave, and I was able to use a two stage diode shaper to approximate a sine wave. Unfortunately, here we ran into trouble. Since we were integrating a square wave, the amplitude of the created triangle wave would shrink as frequencies got higher. We should have anticipated this since our integrator was a low pass filter. At this point the project took a turn into the world of embedded systems.

2 How the Synth Works

By employing a microcontroller, the higher frequency attenuation was no longer an issue. Instead, timers were chosen to select an output value for each waveform at each point in time. Elise was able to put together the code that would allow the microcontroller to select the frequency based on a button press, while I focused on generating the signals. Since the GPIO pins are strictly digital, I opted to convert the values from the timers to pulse width modulation (PWM). My strategy for doing this was too computationally intense, and I was not able to create PWM signals any faster than 1KHz, which would not cut it for the frequencies we wanted to generate.

Simulating the electronics portion went very smoothly. I was able to put together a python script that made preset PWM signals in the form of a PWL file that would control a voltage source in LTspice. Using a low pass filter, some voltage followers, and a summing amplifier to act as a mixer, we were able to simulate triangle and sawtooth waves that looked quite good. In addition, we were able to combine the two waveforms together with the summing amplifier

to create a sort of bent sawtooth wave.

3 Conclusions

Overall, the project served as a learning experience more than anything. If we had another week or so, I think we could have used a system similar to the python script I made to generate some preset PWM values for the microcontroller to use. We also found some wiring errors on our printed circuit board that would have caused problems once we got the microcontroller to generate the desired output signals. We were fairly close to a working product, and I am happy that I got to work on such a fun and rewarding project.