Week 4

This week was spent trying to figure out more about the MIDI standard in order to figure out how to proceed with our project – specifically in trying to figure out the “correct” way in integrating the keyboard with the python program.

Using the notes from last week and performing a more careful read, I’ve realized that at the end of the day, the MIDI standard simply refers to a real-time continuous feed of information on all of the controllers involved in a synthesizer. What that means is that the standard output of a MIDI for synthesizers is continuous – therefore the solution is to properly take in this continuous stream and process it in some code.

Current thoughts are to use either the pygame module or the mido module to take input from the synthesizer keyboard. Pygame has a built in MIDI function that is reportedly well supported and mido is a module built specifically for processing MIDI. While I imagine using pygame would correspond well with JT’s synth program, as he also uses pygame in his code, I am leaning more towards using mido as the documentation examples of pygame I feel are not nearly as abundant as mido’s documentation examples since the last time I looked it up. In any case, the purpose is to get the inputs from the keyboard via USB and parse said inputs. Once said inputs are parsed we can proceed towards entering them into JT’s program.

With that said, there was a large misconception that we’ve had regarding how JT’s program works and how we planned on integrating the synthesizer with the keyboard. At first the thought was that the keyboard would play a note and JT’s synth program would be able to take that note and output a transformation of that note, the specifics of which would be determined by the weights which could be configured by the sliders on the synthesizer connected to the program. JT however informed me that his program only cares about the weights determined by the sliders and nothing else – his program only produces “one” sound based on the weights in the program. To produce different notes, we would have to shift the pitch of that sound up and down the correct ratios.

What that means is that the process of integrating the keyboard and the program should proceed as thus: We must parse the continuous MIDI input and grab only the slider controls, inputting that into JT’s program. We only update our inputs towards JT’s program when the settings of the sliders change. Once that’s done, we wait for the .wav file to be generated with the new weights. As soon as that happens, the program we need to write has to be able to shift the pitch of the .wav file based on how the keys are pressed. The exact details of this would need to be hashed out, but that’s the most basic representation of the path moving forward.

The issues at hand mainly revolve around how to take the keys input such that they are consistent with the synthesizer. Specifically, it needs to be able to handle that multiple keys could be played at the same time – there needs to be a way to turn on and off every note played out at the program’s end because the time a note is played is entirely dependent on when the key is pressed and released, not some static time instance.

In other news, there was also a thought that we could try to install the filter onto the keyboard using MIDI controls so that JT’s filter would create an output at the synthesizer’s speakers. This is more or less impossible – MIDI standards only states that the instruments of any device should be designated as specific values – how the sounds of this instruments play out is independent to each manufacturer. In other words, there is no real way of knowing how to program any one controller and even if we did find out, it would only be for that specific product.