Week 2:

The first week I was learning the basics of the Raspberry Pi, experimenting with potentiometers and the GPIO pins. The potentiometer values are processed by the MCP3008 microchip, and then passed to the Raspberry Pi vie the I2C bus in its GPIO pins. This week I have started thinking about the design, and how I plan to identify audio files and loop the files in sequence. I have decided to do this using a multidimensional array. Each position in the array represents a spot that can be filled by an audio file, and the integer in the position determines which audio file should be played(0 means there is nothing there). Each position will also have an incomplete circuit. If the circuit is completed, that will tell the raspberry pi that there is an audio file present in that position. To determine which audio file is present, there will be unique resistors in the physical audio file, and the raspberry pi can read that value(or the current resulting from the resistor, V=IR). This is the schematic to read the Analog values in:A picture containing drawing

Description automatically generated

There will be three different channels, and the sounds will be played through the different channels synchronously. This is accomplished by using pygame’s built in channel and mixer objects. There will be a loop that works its way through the two dimensional array, playing a new column every 2 seconds(at least for now). That should be scalable to adjust tempo in the future. I have already written all this code, and it works; however, there are some weird delays. At first, I used time.sleep() method between each column, but that would not work because the sleep method would not be called until after an audio file was finished playing. So next, I tried several multithreading options, but I could not get them to work. Finally, I coroutines with the python’s built in asyncio dictionary. This is the best working option that I have figured out thus far, but there is still some inconsistency to the rhythmic elements. I have tested this by adding a 16 bar 120 bpm drum beat in channel0 and a synth hit on every down beat. Around 85% of the time the synth hit will be on time, and the other 15% it will be slightly early or late. Another observation I have made is that it does not go out of time more and more the longer the loop runs, which is a problem most people who post advice for projects like mine have run into. I will continue researching other options for keeping time in this sequencer. I also might try out the method I have right now, but use fewer columns, and make each of them last for a longer time. This would mean fewer time delays, and each one for a longer time. This would make the timing more accurate, but it would also mean that there could be fewer horizontal positions to start different audio tracks. I also ran into a problem with the .wav files that can be played by pygame. Some were working and some would raise errors. I figured out that the ones that worked I had converted from mp3 files to .wav files instead of bouncing them from logic as .wav files. Something in the conversion process makes the files readable. I am also considering the possibility of trying to get access to a 3d printer so that the finished product looks as good as possible. I am not even close to that stage, but I figure it would be a good idea to get started on that as soon as possible. I have also been practicing with Max, and I hope to use it to experiment with prototyping ideas and hopefully make some unrelated visual art with Jitter as well.