Max Kawula

Capacitive Touch Keyboard

For my final project I created a 12-key keyboard that uses capacitive touch to detect key presses. Some of the main aspects include using a Raspberry Pi to produce the sound. This allows me to use software to playback samples instead of just using buzzers. A few other benefits include being able to play audio from an audio jack, instead of having to drive a speaker, and polyphony which lets me play multiple notes at the same time. My project also includes building a physical keyboard so the key layout is familiar to those who have played piano before.

Building this was pretty straight forward. I needed to create an enclosure for the Pi and Sensor. And then I needed to create keys using copper fabric. To create the enclosure I made measurements on how wide and tall I needed the keyboard and then how much space I would need for all the internals. I added in slots for the raspberry pi and the keys so wires could run through to the outside. To create the keys I made some precise measurements in adobe illustrator to fit all 12 keys tightly together on the enclosure. The rest just involves soldering wire to each key and connecting it to the sensor.

The software portion of this required many steps to reach a operational level. As I said before, I am using software, Sonic Pi, to create the sounds. This software doesn't communicate directly with the pins on the Pi so it is necessary to use scripts to send key presses to sonic pi. I found a script online that does just that. This script uses something called Open Sound Control (OSC) which is a communication protocol. With this script,

whenever the mpr121 detects a touch, it will send an event to Sonic Pi. On Sonic Pi, you can use these events to trigger sounds. The code behind the keyboard simply involves checking for when these events come through and playing a sound. While this works fine as it is, adding more to this proved very difficult.

First of all, the software only responds to key presses. Sonic Pi plays a sound for a given duration. So everytime I hit a key it would play it for equal length, regardless of how long I held it. Ideally, I want there to be a key press, hold and release, but this required a better understanding of python to fully integrate. I had a solution in the works that never made it to the final project. It involved playing each note for 10 seconds and whenever a key was released, another event would come through and interrupt that duration. While this is easier to do with midi events (like from an actual midi keyboard). OSC events were much less usable. Another issue that came up was the latency. Sonic Pi is more suited for sequencing music and playing it back, and not so much for real time playing. The result is this noticeable lag for everytime I hit a key. It makes it unplayable since every key press is behind. There are some solutions for this but none of them were enough to fully get rid of it.

If I were to redo this project I would not use the raspberry pi because it relies too heavily on other people's code. With my other projects I knew exactly how things worked because I created them, but this one was trying to reverse engineer something way beyond my ability. That being said, I think it's worth it to explore more solutions to make sounds on raspberry pi. There could be other software capable of playing back samples that won't require a lot of coding to get it working.