In a few words, this project is basically an Arduino instrument. It consists of 4 capacitive touch sensors (1 used for mode switching), 6 potentiometers, (4 for pitch, 1 for modulation tuning, 1 for), and some python scripts that interact with MAX using the ‘udpreceive’ object. Each of the touch sensors act like a kind of key on a keyboard. Each of the slide potentiometers controls either the pitch of their respective notes or the level of modulation on each note. When 5th touch sensor is tapped, the potentiometers are switched from controlling the pitch to controlling the modulation index of the frequency modulation.

The data flow of this project may have been a little more complicated than it had to be, but it was the easiest way I could think of. It starts at the Arduino, where the Arduino simply broadcasts it’s formatted sensor data to my computer at a set polling rate. Python then picks up the string and sends out the data odot variables in an OSC bundle through UDP. MAX then uses the udpreceive object to get the osc bundle and routes it to the given MAX objects. This was probably the most complicated part of the project and I was helped significantly by my friend who has significant Arduino experience, and a previous project involving communication between Arduino and MAX.

In MAX, the flow is fairly simple. The capacitive touch sensors send either a 1 or 0 depending on whether or not they’ve been touched. This is received as “/dN/on” where “N” is the number corresponding to the capacitive touch sensor. This is then passed on to the right inlet of an o.var object that pushes the osc bundle with the amplitude curve data into the click/phase/buffer combination. Note that it also receives information about duration, which is controlled by the right most potentiometer on the wood board. This value is scaled by a factor of 10 to make it more dynamic as the potentiometer only has a range of 0 to 1023. The mechanism to control pitch utilizes frequency modulation by setting the carrier frequency and frequency input of the biquad object to the scaled value of the left 4 potentiometers. Modulation is controlled by the scaled value of the middle potentiometer.

While I’m pretty satisfied with where this project has gone, I believe there’s significant room for growth. Given that the information flows from 3 different blocks of code (Arduino, Python, and MAX), there’s room for substantial amounts of processing of the signal. While the project currently only demonstrates how to use the sensors in a way similar to a synth, they can be easily configured to do almost anything else. Other hurdles I faced were more out of my control. Amazon messed up the shipment of many of my sensors: many shipments arrived with either the wrong amount of sensors or just completely different sensors than I ordered. I originally intended on having 10 capacitive touch sensors, 10 potentiometers, a .3 megapixel image sensor. I received only 5 capacitive touch sensors, 4 potentiometers, no camera, and a hodgepodge of random bits whose function I cannot even figure out. But perhaps it’s a good thing that these sensors did not come, because even if they had, I would have (and did even with my cut down resources) run into another problem: the Arduino I have only has 6 analog inputs, meaning that I can only use a max of 6 potentiometers. While there are methods of using multiplexors or analog to digital conversion circuitry, I barely passed EE16A and I couldn’t quite get that circuitry to work. In the future, with all the sensors in the world and no Arduino/EE limitations, I’d like to include the camera sensors so that I can manipulate image data and turn it into something meaningful in the sound space. I’d also love to use other kinds of sensors, maybe take in EEG data to create music straight from the mind (I’ve used EEG headsets for other, similar things). In any case, I hope this project demonstrates some understanding of the class material, thanks for the semester!