James Hoder
Electronic Musical Instrument Design
5/10/19

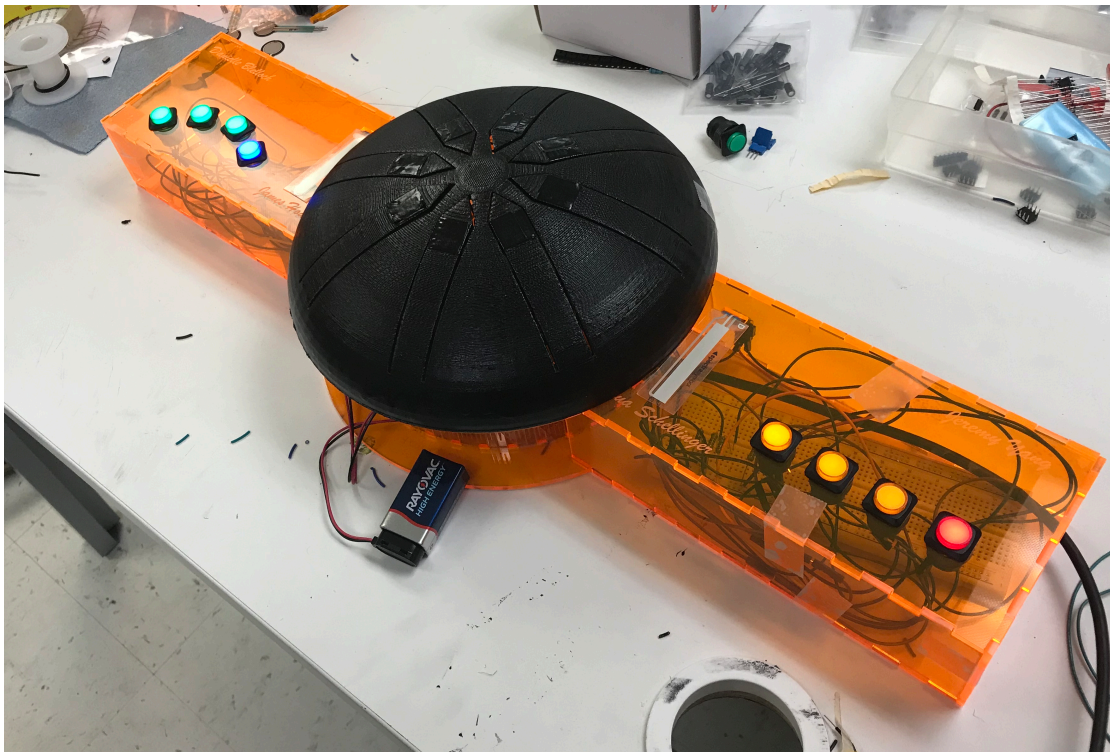
The DJJJembe – Final Design

Concept

The DJJJembe is a tonal percussive instrument, modeled after a steel tongue drum. Like a tongue drum, its main component is a dome in the middle with 8 tongues cut out. Each tongue plays a different note when hit. Since all of the tongues are connected to the same dome, when one is hit and made to vibrate, the nearby tongues are also vibrated slightly, playing their notes in addition to create a unique harmonic sound. In our design, rather than the sound being produced by the vibrations of the dome itself, our instrument will sense the dome's vibrations and electronically produce the desired sound. The dome is made to sit on a box-like base, so that it can easily sit on a table or on a person's legs while playing sitting down.

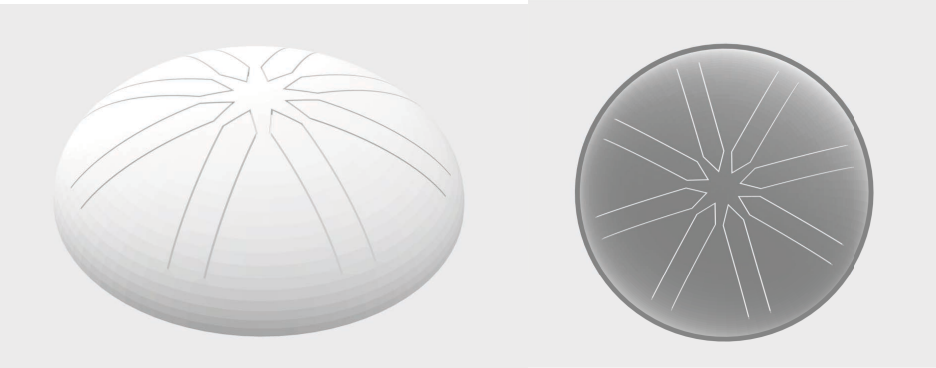
On the sides of the main dome, the DJJembe has a collection of sensors that can be used to modify the sound that is activated by hitting the dome. The dome can to produce a range of different sounds by changing the sound bank, so that it can sound like a traditional tongue drum, a full drum kit, an electronic bass, or a string synth.

In this final iteration, the sound produced by the dome could be any of those 4 sound banks, set with any key signature, and with 5 different types of scales, as adjusted by the buttons to the right of the dome. The buttons to the left of the dome played background pad chords, to accompany the primary sound coming from the dome. The instrument also has two soft potentiometer sliders to control the pitch bend and modulation wheel of the sound produced by the dome.

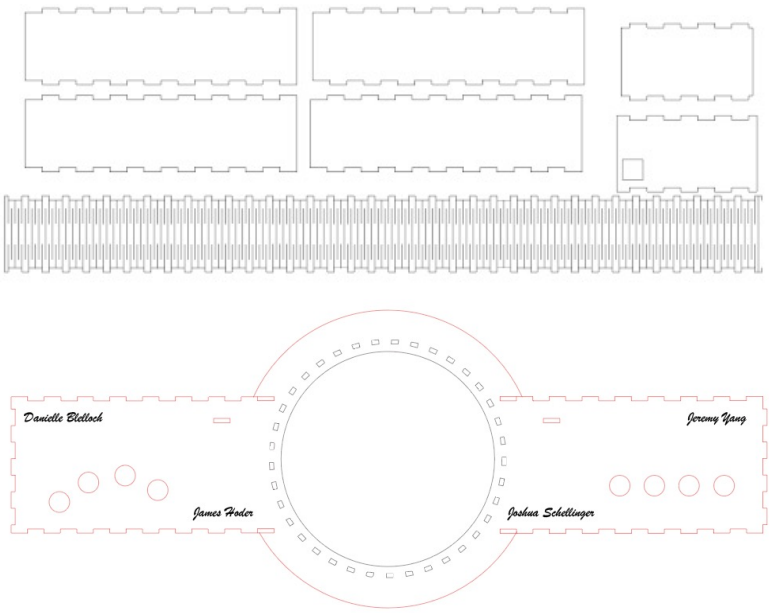


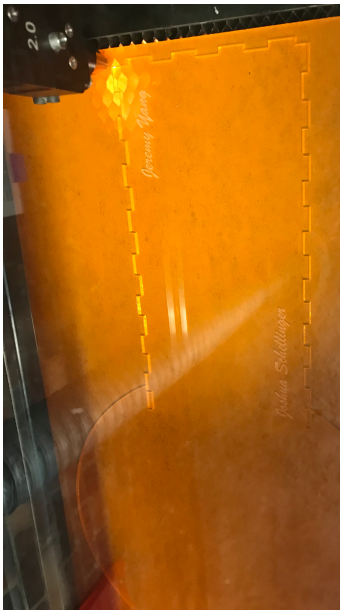
Construction

The central part of the instrument is the dome. We chose to 3-D print the dome so that it could be a single piece of hard plastic, made exactly to our specifications, so that the vibrations could most easily move through the entire piece. To print the dome, we first designed it in Solid Works by creating an oval that was wider than it was tall and revolving it, to produce a slightly flattened dome. We then sketched one of the tongues onto the dome, made 8 rotated copies of the sketch around the dome, and extrude cut the tongues out. We originally designed it to have a height of 4" and a diameter of 12", but after discovering that the largest printer we had available was 9", we scaled the dome down to fit. Bellow you can see the Solid Works file.



The dome, push buttons, and soft pots were all mounted on a box that held all of the wiring, which was laser cut from 1/8 colored acrylic. All of the parts of the box were designed in Illustrator, as shown below. The sides of the box were made with a tongue and grove system so they could press-fit together easily, and the rounded part of the box was cut using a "living hinge" design, which made many small cuts in the hard acrylic so that it could bend and curve. The inner part of the box was left hollow, and a large hole was left in the middle of the top, so that all of the electronics could fit inside the box and the Piezo sensors could come out of the box and attach to the dome. Small holes were cut in the sides of the box to allow wires to pass through. The box was 2.25" tall. The whole project was first cut out of plywood for testing, but then was cut in acrylic, as shown in the picture below.



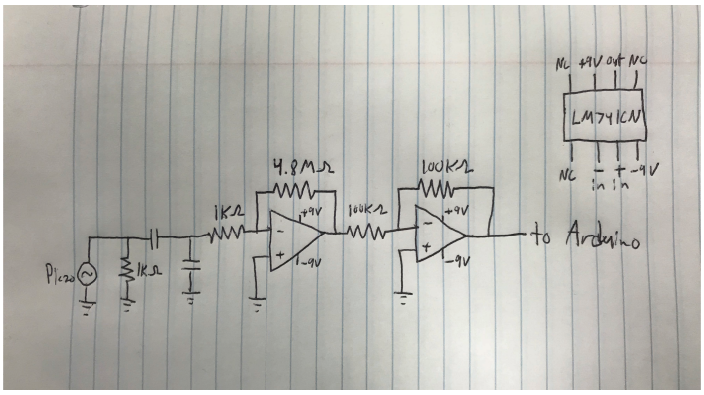


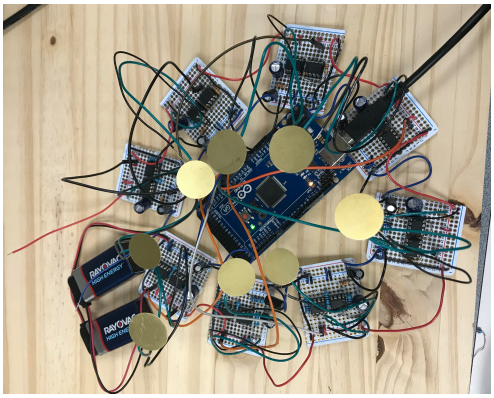
Sensors and Electronics

Inside the dome, 8 Piezo vibration sensors were implemented in order to individually read the vibration of each of the tongues. The Piezo send out a voltage spike whenever they are hit, proportional to the strength of the vibration. Therefore, whenever one of the tongues is struck, it will vibrate the Piezo, allowing note-on and velocity data to be detected based on the maximum amplitude of the Piezo's output. The data from each of the Piezos is sent to an analog input pin on an Arduino, so that the data can be brought into the computer for use in Max.

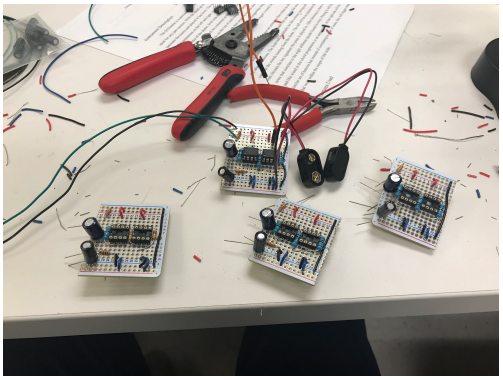
To utilize the Piezo properly, it was put in parallel with a resistor and tied to ground. Thereby, the output of the Piezo was always a positive voltage spike. The output of the Piezo was very low voltage, had a lot of high frequency noise, and had a DC offset, so it was filtered and amplified before being sent to the Arduino, which expected to receive clean data ranging from 0V to 5V. To block the DC offset, a 100uF capacitor was put in series with the Piezo to create a high pass filter. To reduce high frequency noise, a 47uF capacitor was put in parallel with the Piezo to create a low pass filter. Thereby, only voltage spikes made it through. To amplify the data, it was sent through two LS741 inverting op-amps, configured to have a combined gain of 4800V/V. The op-amps were powered by +9V and -9V rails, implemented using 9V batteries. The output of the amplifiers was then sent to the Arduino.

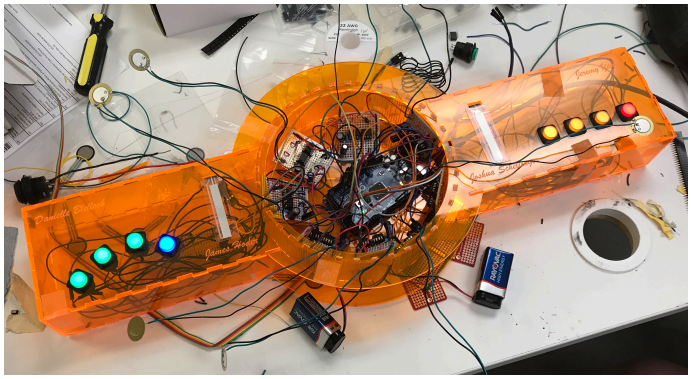
This circuit was first tested on a breadboard, and then 8 of them were soldered individually onto protoboards. All of the protoboards had their power and ground lines tied together, so they could all be powered by only 2 batteries. The schematic, breadboard, and protoboards are shown below. The protoboards were put into the acrylic box, and the piezos were positioned outside the top hole, so they could be attached to the dome.





After initially building the circuit and soldering the 8 copies, extensive debugging had to be done. On the protoboard, many different gain values and filter capacitor values had to be tested before the output was consistent and within the correct range. Once soldered, not all of the protoboards worked, and each one had to be individually debugged. Due to manufacturing error, some boards had power lines switched, some had noise from accidental shorts, some had broken op-amps, and some had loose connections. All of these problems had to be fixed using the oscilloscope and multimeter. Furthermore, there is still some noise that we haven't been able to eliminate, and when initially powering up, the circuits take a few minutes before they can operate properly because the capacitors need to settle.



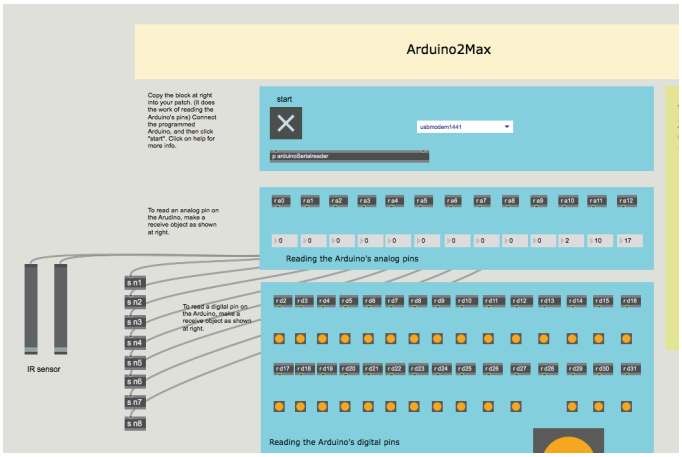


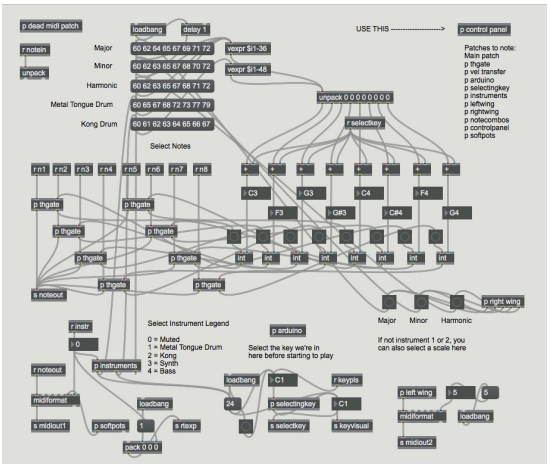
All 8 push buttons had their LEDs tied to power through a 330-ohm resistor, and to ground. The push button's switch leads were attached to ground and digital pins on the Arduino, since the Arduino was set to active low. The two ends of the soft potentiometers were attached to power and ground, and the center pin of the soft pots were attachced to analog inputs of the Arduino. All power for the buttons and soft pots was pulled from the 5V source on the Arduino.

Max

A Max patch was created to take the input data spikes from the Arduino, read the maximum value, and convert that into a MIDI note-on with a proportional velocity for the proper tongue note. To get the data into Max, the ArduinoToMax_Mega patch was used. If the spike from the Piezo goes above a specified threshold value, it will activate a note-on. A delay was put in place so that the maximum value of the spike is read, and the note-on is only sent once per spike.

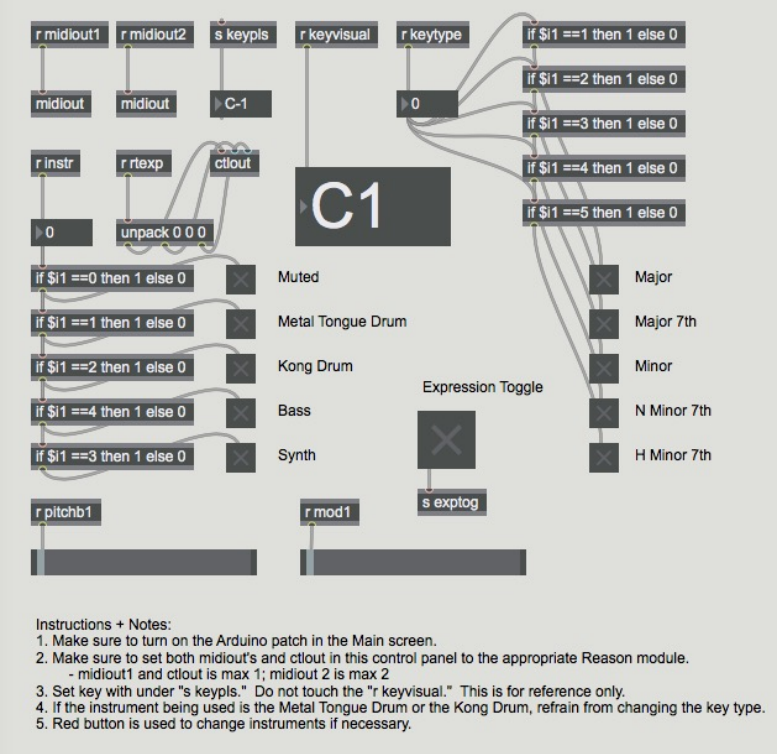
In addition, the Max patch allows for the selection of different instruments in Reason, and for the use of different key signatures with the drum. The 8 tongues can be mapped to play different scales in different keys.





A logic gate system was implemented so that different combinations of the buttons on the right of the dome can change the sound bank of the instrument, and the scale of the instrument. The red button was made to be a mute button, and the green and blue buttons were made to play synth pad chords. The 4 synth pad buttons were programmed to play 7 different chords, based on pressing different combinations of the buttons at the same time.

A control panel was also made, in order to easily see all of the current settings based on the button inputs, and to allow the user to easily change the key. The control panel is pictured below.



Audio Synthesis in Reason

In Reason, 4 different sound banks were created on 4 different channels, so that they can be used in different modes of operation. First, the NNXT was used to create a sound bank for a traditional tongue drum, using samples found online. Each tongue had a set of samples that were assigned to it, which were chosen and activated based on velocity. The online sample set included greater attack, higher volume, and more harmonic resonance of nearby tongues in the samples that we mapped to higher velocities. The notes used in this sound bank are within an F minor scale, so the notes are C2, F2, G2, Ab2, C3, Db3, F3, and G3. In addition, mod wheel was set to control LFO rate, which is mapped to vibrato, to be implemented physically in the next iteration.

The second sound bank was implemented using the Kong Drum in Reason, using the Rubadub patch, giving 8 unique samples that were modified based on velocity. Mod wheel was mapped to pitch for this bank.

The third bank was implemented using the Woodcutter patch on the Subtractor, which sounded like a string synth with a short attach and long decay. Reverb was added to this bank in order to give a more pronounced sound after the initial hit.

The fourth bank was implemented using a standard synth bass in the Subtractor. It used a single sine wave with a short attack, and was made monophonic, so that the bass notes would not interfere with one another.

Mistakes and Additional Improvements

The biggest mistakes made included the printing of the dome, debugging of the piezos, and the complexity of the Max patch. We made the mistake of assuming that 3-D printing would be our easiest option when, in reality, it was very difficult for such a large part. Next time, if we need to produce another dome, we will likely print it in smaller parts and glue them together to ensure that the 3-D printer does not fail.

We also struggled lot in debugging the piezos, because we tied power and ground of all of the piezos together. Therefore, noise or bugs in any one of the circuits resulted in issues in all of the circuits. As a result, we had to spend much longer trying to debug, and we still weren't able to fully get rid of the noise. In another iteration, we might want to isolate each of the Piezo circuits, so that they cannot interfere with one another.

The Max patch had to be very complex in order to properly read data from the piezos as we hoped. However, it eventually became so complex, it was hard to edit and debug. We should have waited to create the Max patch until we had the Piezos fully functional, then we could have built the patch slowly and with less bugs and need for difficult tweaking.

The piezos were also not amplified enough, so a very low threshold had to be used, which increased noise and reduced the range in velocity control. In the future, the instrument could be improved by using more sensitive piezos that would not require as much amplification.