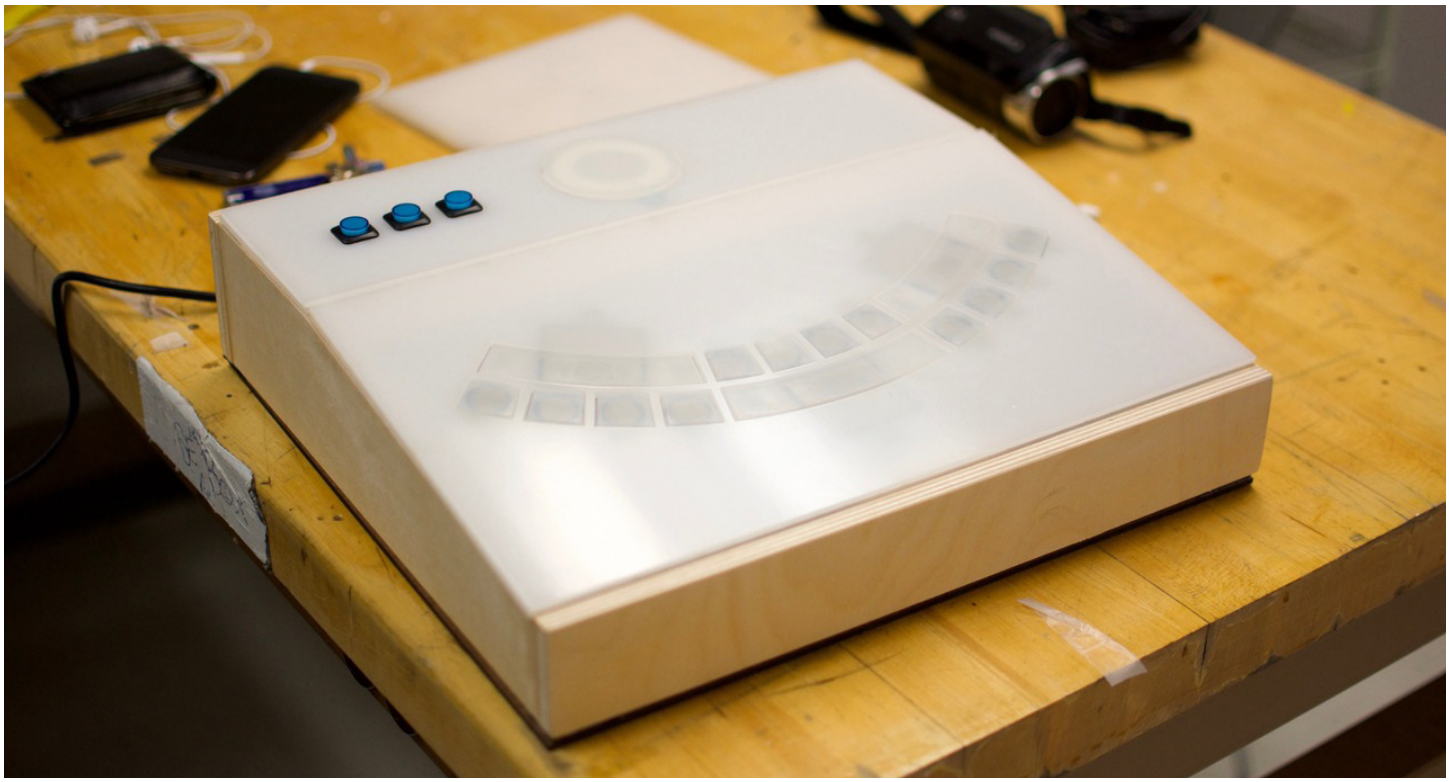
iColtrane



report by **Benjamin Averill**
in collaboration with **Liam Brady, Dylan Hudson, and Rohan Joshi**

for **Electronic Musical Instrument Design**
taught by **Professor Paul Lehrman**
presented on **10 May 2017**

Introduction and Theory

The iColtrane is an instrument that tackles the challenge of actualizing a theoretical tonal system as a physical, playable interface with expressive musical capabilities. It draws directly from a sketch of a modified version of the circle of fifths drawn by legendary jazz saxophonist John Coltrane. Though no one is sure what Coltrane's motivations for constructing the diagram were, many theorists have spent time analyzing the geometry of the diagram and the note relationships that Coltrane may have been trying to highlight. The drawing was simplified by Corey Mwamba into a graphic, shown below, as part of a blog post titled, "Way of Seeing Coltrane."

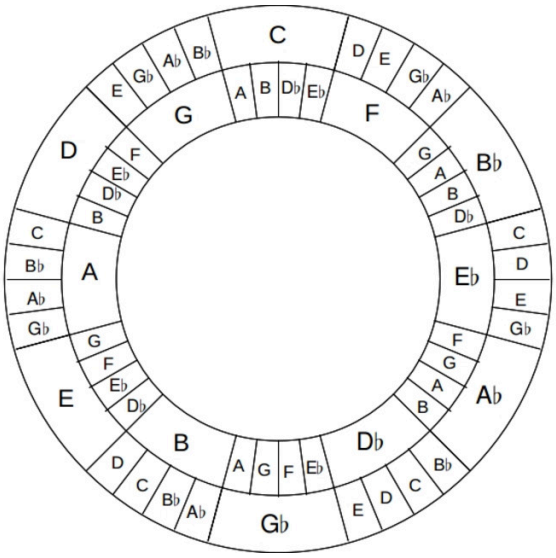


Figure 1: A digitized version of the Coltrane Circle, in which the circle of fifths that underlies the structure of the system is emphasized by enlarged sections.

The Coltrane Circle consists of two concentric circles which are each broken up into the two different whole-tone scales (the outer circle being C, D, E, G-flat, A-flat, and B-flat, and the inner circle being D-flat, E-flat, F, G, A, and B, the same as the outer but shifted up a semitone) that are repeated five times sequentially. Every fifth note in each circle takes up the same amount of physical space as the four notes between, and they are arranged such that the larger notes move through the circle of fifths as you zig-zag counter-clockwise across the two circles (and the circle of fourths clockwise).

The iColtrane allows users to interact directly with Coltrane's invention by playing 15 keys arranged in a sector comprising one quarter of the entire circle, with functionality built in to move to any of the twelve different sectors of the circle and to build custom harmonies.

Instrument Design and Playing Technique

The interface of the iColtrane is divided into two sections: the array of fifteen silicone rubber keys on the bottom, and the top section containing three buttons and a softpot rotary potentiometer.

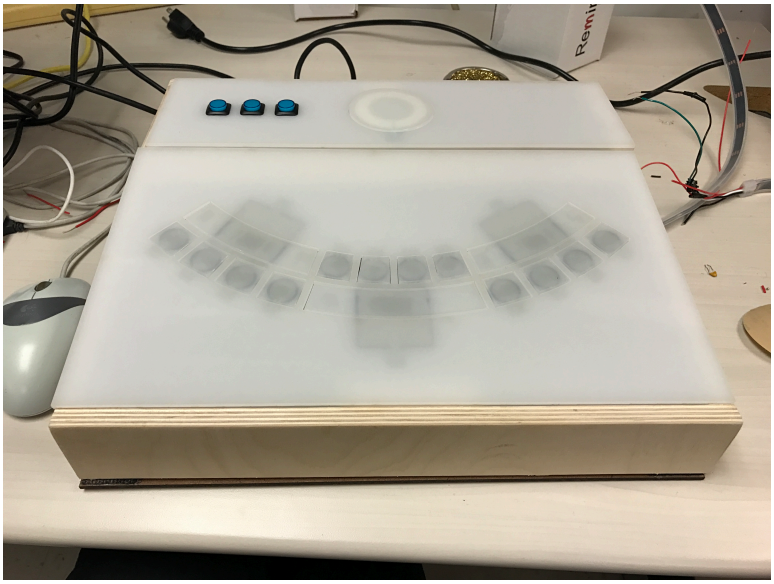


Figure 2: A view of the iColtrane from the vantage point of the player.

The silicone keys are arranged in two rows that form an arc curving away from the player towards the top of the iColtrane. The top row contains 2 large keys on the sides and 4 small keys in the middle. Conversely, the bottom row contains two sets of four small keys on the sides and one large key in the middle. Each key sits on top of one of two different sizes of force sensing resistors (FSRs): the 3 large keys are attached to 1.73” square FSRs and the 12 small keys are attached to .73” diameter circular FSRs. Each FSR is mapped to its own module in Reason so that the data read from the FSRs can be used to control parameters of each key polyphonically. In this way, the player can play any combination of keys at once and have fine control over the parameter to which force is mapped for each individual note.

The softpot on the top portion of the iColtrane is used to select any sector of the Coltrane Circle to play on the instrument. In Max, the softpot is divided into 12 equal segments that offset the values of every note in the instrument by a constant. Pressing at the bottom of the softpot sets the large key in the bottom row to C. At this point, the large left key is G, a fourth below C, and the large right key is a fourth above at F. Notice that this arrangement is the mirror image of the Coltrane Circle as it appears in Figure 1. We made this design decision to ensure that notes moved up from left to right, allowing musicians familiar with the piano to grasp the note relationships of the iColtrane more easily. Moving clockwise, the next region of the softpot selects G as the center pitch, thereby moving C up to the large right key and replacing G where it previously was on

the large left key with D, the note a fourth below. By continuing to select the next region of the softpot, the player moves the iColtrane through the circle of fifths, with the keys always maintaining the same note relationships relative to each other.

Finally, the three blue buttons to the left of the softpot allow the player to add notes to the pitches that the large keys play. When the left button is pressed once and a large key is played, it sounds both the root pitch which was already selected and a major third above it. If the left button is pressed once more and a large key is played, a minor third is played. Pressing the button a third time turns off the interval and plays a root note. The middle button adds a perfect fifth above the root of each large key on the first press and a diminished fifth on the second press, and the right button similarly adds a major seventh or a minor seventh. In this way, the player is able to build a diad, triad, or seventh chord using any combination of these interval additions and play them across all of the large keys on the iColtrane.

Construction

Our construction of the iColtrane begin with an evaluation of the ergonomics of the interface. We started by drawing out potential sizes and key placements on the paper coating of a sheet of acrylic. We experimented with different widths, key spacings, and orientations of the arc itself. We opted to have the arc facing away from the player because it felt more natural to play with both hands when the fingers were angled toward each other. We also decided that the interface of the iColtrane would have two surfaces: one lying flat at the top for the softpot and chord controls, and the other angled slightly down toward the front of the instrument. This angle lets the hands rest more easily on the instrument.

After conceptualizing the iColtrane's physical design, we determined the materials we needed to use to build it. The frame of the instrument is made of wood that we carved from a sheet using a computer numerical control (CNC) router. We glued the sides together and added a support beam running horizontally across the middle. The two interactive faces of the iColtrane were built using a double-layered system of laser cut acrylic. The FSRs and softpot were directly adhered to the bottom layer of acrylic. This layer has small portholes for the tails of each sensor to run through to the wiring of the instrument within the frame. The top layer of acrylic was cut to have windows that exposed the sensors on the layer below. These were designed in Adobe Illustrator and laser cut. Additionally, we laser cut silicone rubber pieces to fit in the windows of the top layer of acrylic. We chose silicone for the keys to give players the sensation of pressing into the keys and feeling the expressive musical response that the FSRs control. Both the silicone and the acrylic are a semi-translucent white, allowing a strip of NeoPixel LEDs to run alongside the sensors on the interior of the acrylic and shine through when notes are played. The iColtrane also has a velcro detachable base plate made of medium-density fiberboard.

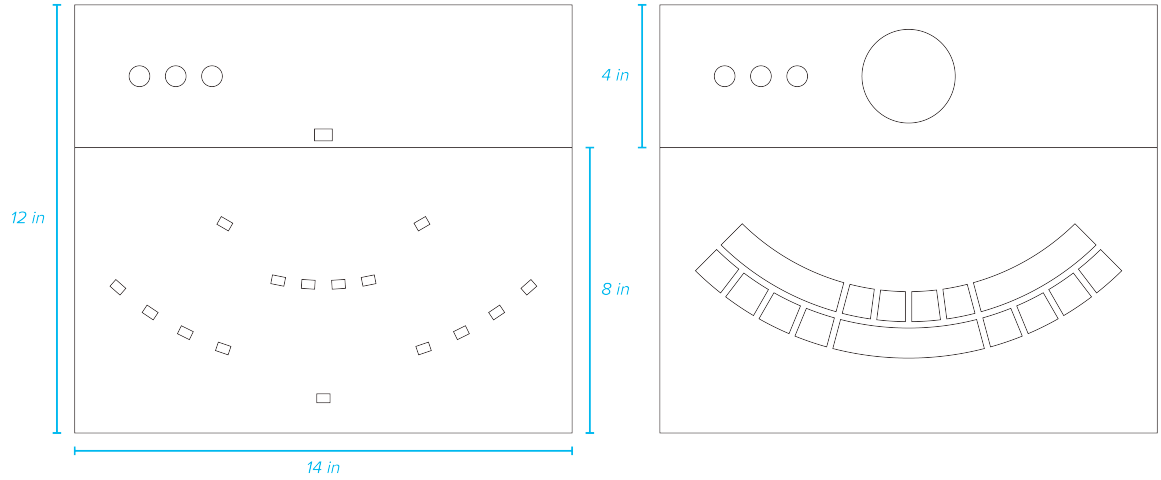


Figure 3: Designs for laser cutting the acrylic interface components. The left side shows the bottom layer to which sensors were adhered and the top side shows windowed holes for the silicone rubber keys.

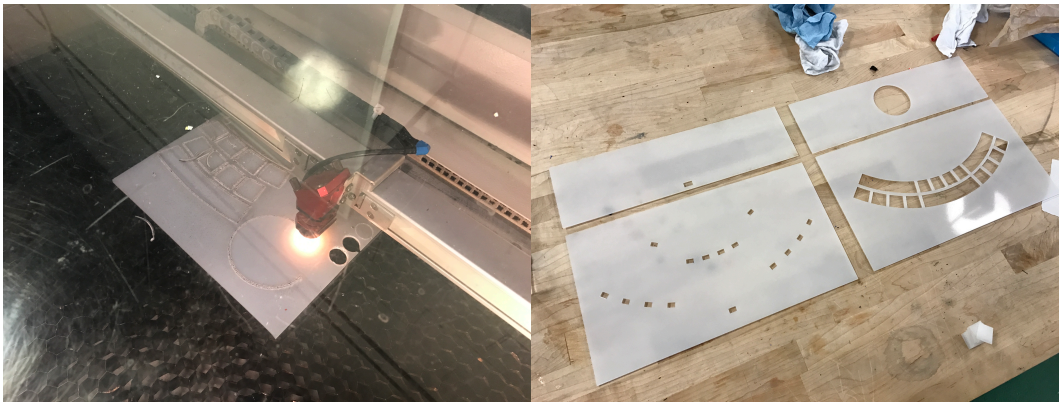


Figure 4: *Left:* Laser cutting silicone keys. *Right:* Cut layers of acrylic, with tails for sensor holes on the left and windows for silicone buttons on the right.

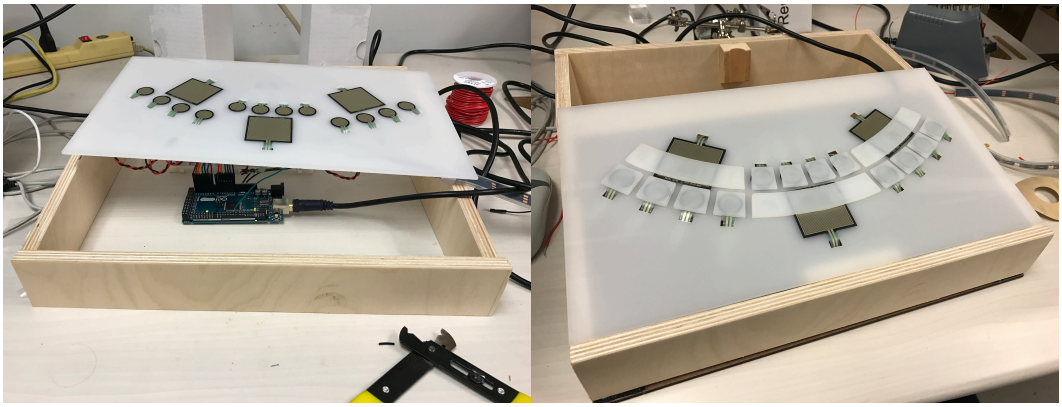


Figure 5: *Left:* FSRs attached to the top layer of acrylic. *Right:* Silicone keys attached to each FSR.

Finally, every sensor and button on the iColtrane is wired up to an Arduino Mega via a breadboard. The FSRs and softpot are connected to the 16 analog pins on the Arduino, each with an intermediary resistor on the breadboard. The buttons and LED strip are connected to the digital pins. The Arduino and breadboard sit inside the frame of the iColtrane.

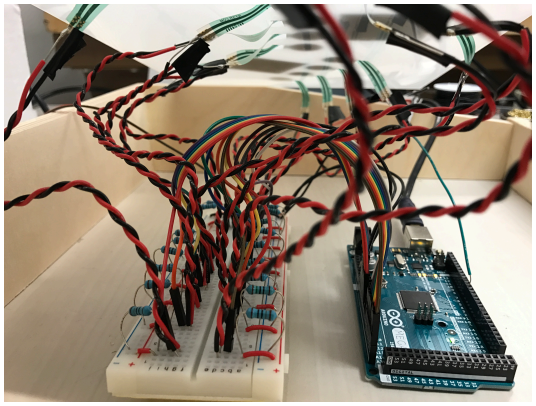


Figure 6: Sensors are wired to the breadboard and routed to the analog pins.

Programming

The iColtrane uses custom Arduino code to be able to read serial data from every input on the Arduino into Max and to send Max data back to the Arduino to control the LED strip. The Arduino code conditions the data received by the sensors to make sure that there is no cross talk between the sensors, buttons, and LEDs. The data from the Arduino is fed into Max as a list of numbers which is unpacked in the main iColtrane patch and directly accessible for processing.

The Max patch is divided into 4 main sections:

1. The main window with serial port settings, patchers for every sensor, initialization and note offset math, button control, and a monitor for displaying the current position in the circle.
2. The softpot patcher, called *p softpot*.
3. 12 patchers for the small FSRs, *p note*.
4. 3 patchers for the large FSRs, *p bignote*.

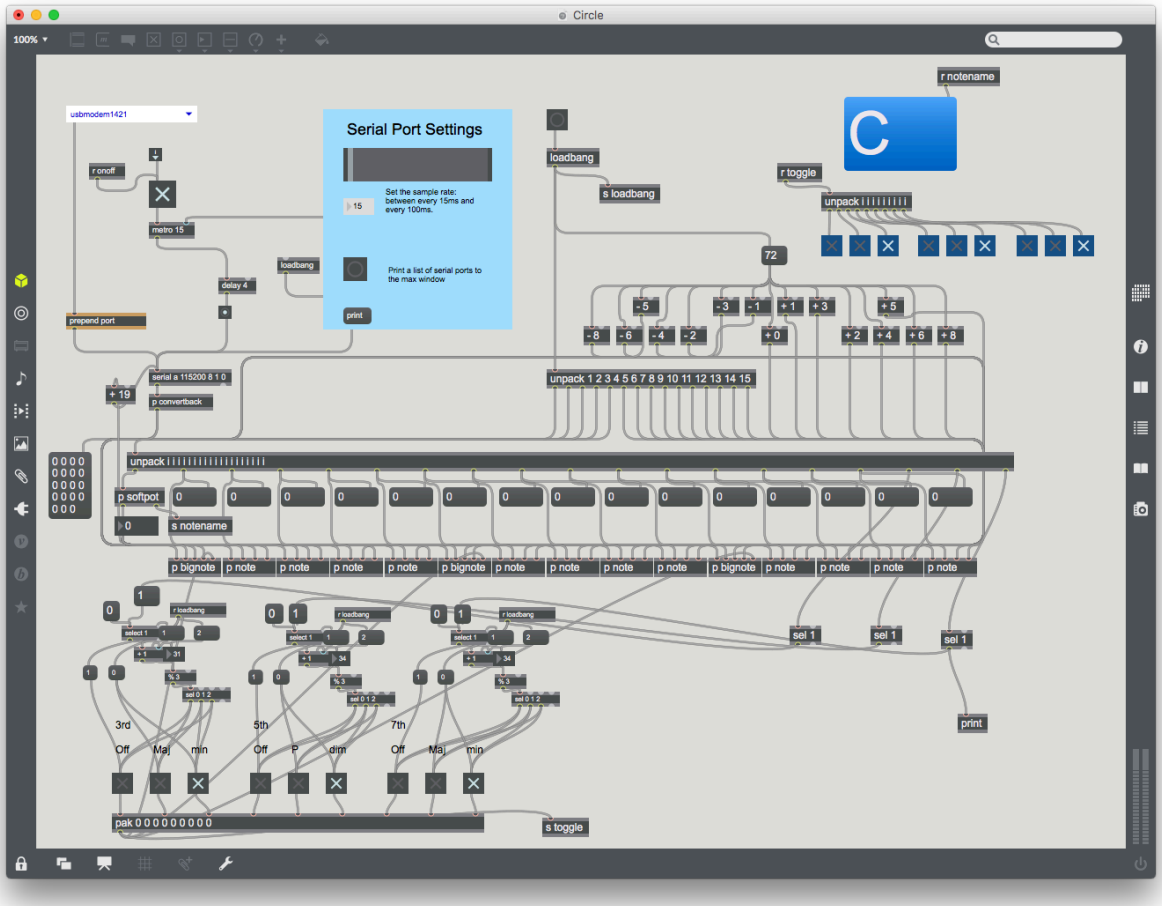


Figure 7: The main window of the Max patch.

The top left corner of the main patch window contains code that reads in data from the Arduino that is sent to the long unpack object with nineteen outlets in the center of the window. The first outlet is connected to the *p softpot* patcher, which reads in the position of the finger on the softpot and outputs a positive or negative integer constant that offsets every FSR note value. The next fifteen outlets are devoted to the FSRs, which are numbered from left to right and top to bottom on the interface of the instrument itself. The last three outlets are for the three buttons that control intervals. Below the row of note patchers is the logic for moving between the three toggle states of the buttons.

Above and to the right of the unpack object is the math for calculating each note's relative position away from each other. The arithmetic objects are arranged in the same design as the keys of the iColtrane. It shows that the large FSR in the middle of the bottom row is the "home" key, and every other note is some integer value away from it. The bottom row is offset by even numbers and the top row is offset by odd numbers. Notice the relationship between each large key and the small keys across from it; the two middle small keys are always a semitone down and up from the large key, and the outer keys are 3 semitones. The design of the Coltrane Circle is incredibly modular, so no matter which softpot position was last read, the relationship between the individual keys always stays the same.

Finally, there is a small section in the top right that displays a note name in a large blue message box. This note is the note that is currently assigned to the large key in the center of the instrument. Below it is another iteration of the toggle keys for the chord buttons so at a quick glance you can see both what note is the home key and what chord is currently selected.

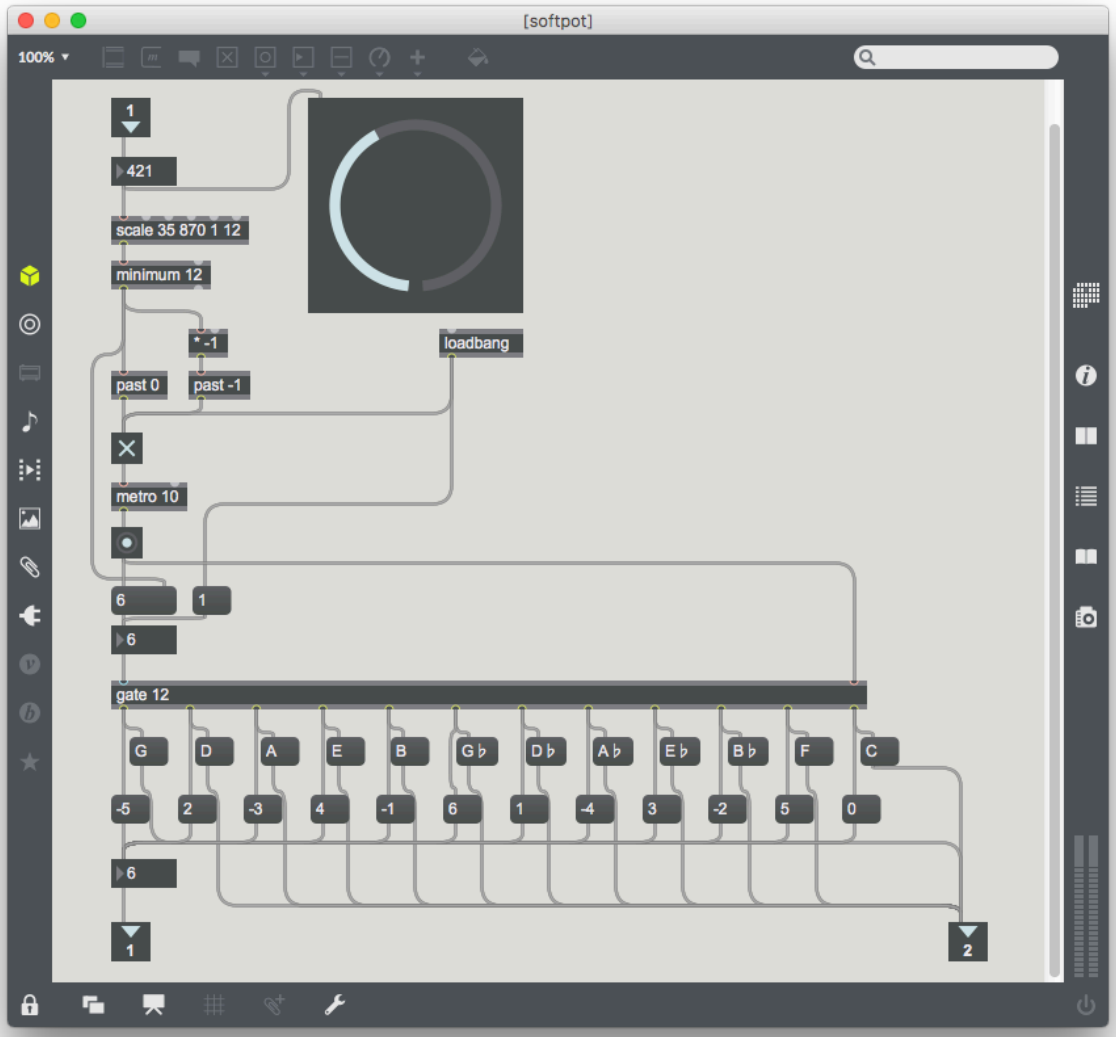


Figure 8: The softpot patcher.

In short, the softpot patcher reads in the analog data of the softpot from the Arduino and converts it to an integer offset for every MIDI note number in the note patchers. The softpot value is scaled down from a large range of numbers to an integer between one and twelve. Position twelve is at the very bottom of the softpot, and moving clockwise from there moves up starting at 1. Then, the position integer is converted to a note offset value for that portion of the Coltrane Circle. These values are seen under the gate object, along with their corresponding pitch class names. The offset is read out of outlet one, and the note name is sent to outlet two. While the softpot data value always reads 0 when it is not being pressed, the Max code retains the last section that was selected until the player next interacts with the softpot.

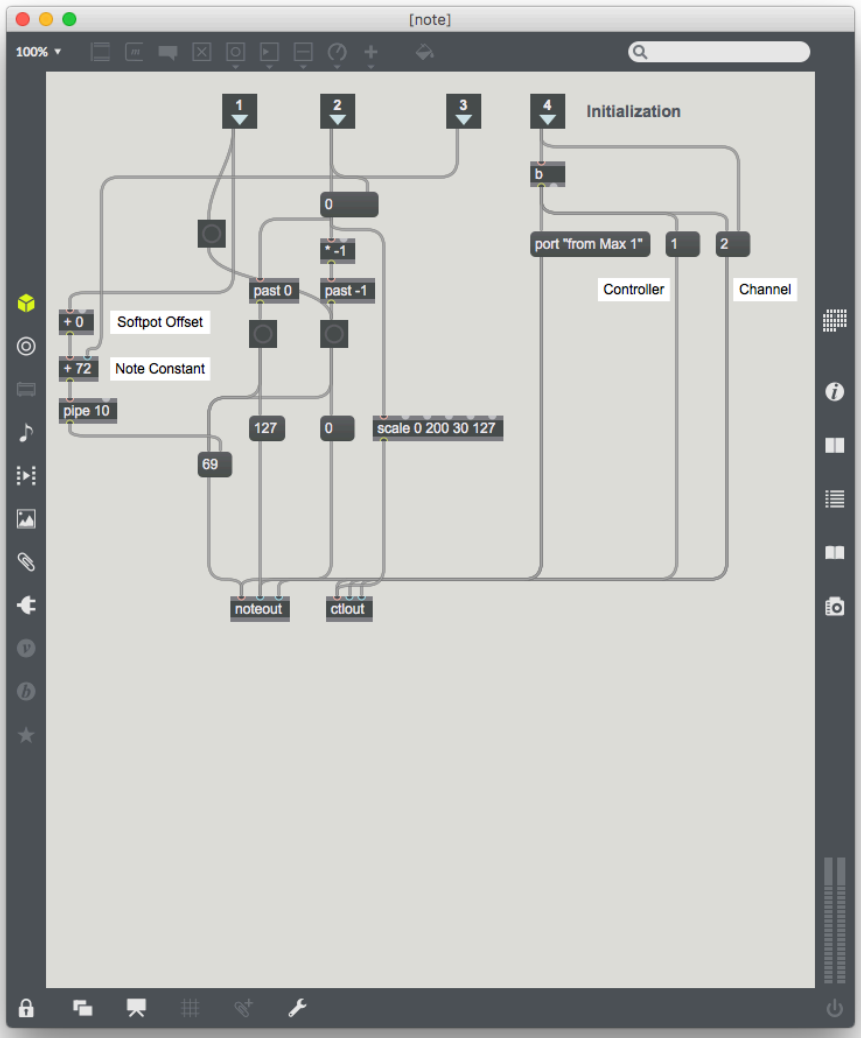


Figure 9: The note patcher for the small FSRs.

The note patcher is relatively simple, but very powerful. The first inlet receives the offset value set by the softpot patcher, and in combination with the third outlet which receives the key value offset from the array of integers in the main window, these numbers add to determine the current note number. The fourth outlet receives a number from one through twelve that corresponds with the ordering of the keys, and this is used to set the channel number, port information, and the controller number to the noteout and ctlout objects at the bottom. Finally, the second outlet reads in the data from the FSR attached to this individual note patcher. When the value exceeds 0, a bang is sent to the note number message box and the message box containing the note on max velocity of 127. These messages are received by the noteout object and a note is turned on. The ctlout object receives a scaled integer that is used to control any parameter of the synth module in Reason. Then, whenever the number received in outlet two returns to zero, a bang is sent to the note number again and the velocity message box of 0, which sends a note off command for the currently playing note.

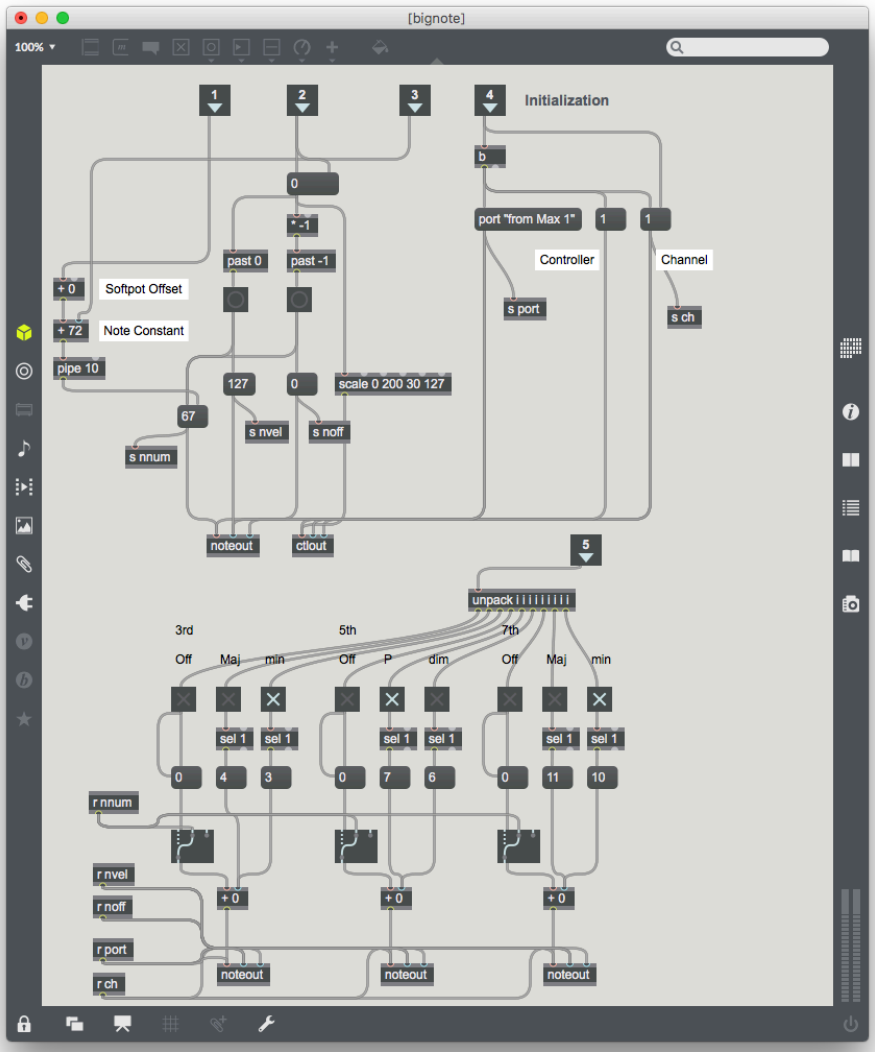


Figure 10: The bignote patcher for the large FSRs.

The bignote patcher is the same as the note patcher except it contains logic for reading the current toggle state of the chord buttons and adding the necessary notes. This is shown at the bottom of the window. The toggle states are read into inlet five as a list, which is unpacked and relinked to new toggle buttons. There are three sets of three buttons, with two on states and one of state. When a button is in the off state, no information is sent to the corresponding noteout object below it. However, when it is in one of the two on states, a switch is turned on which receives the current note number from the upper portion of the patch. A fixed number, representing an interval, is added to the note number and sent to the noteout object with the same velocity, note off, port, and channel information as the main noteout object above. There is one additional noteout for each button, so intervals can be added individually and turned off easily.

Reason

Figure 11: The primary Reason rack for the iColtrane.

All of the sounds made by the iColtrane are produced using Reason. The rack contains 15 individual Subtractor synthesizer modules which are routed through a compressor and a reverb module and each given its own output channel in the Advanced MIDI device section. Each note patcher in Max had a channel assigned to it, and this setup reflects those channel assignments. Also in Max, the ctlout objects were assigned to controlled number 1, which is linked to the modulation well in the Subtractor modules. Then, we decided to have the modulation wheel control filter frequency. Thus, when a key on the iColtrane is pressed, the modulation wheel moves up, increasing the filter frequency and giving the player expressive control over the note. Of course, the controller number can be changed to control any parameter of the sound the player wants, making the use of FSRs a powerful tool for expression.

Conclusion

While the iColtrane is already a powerful tool for exploring the theory behind John Coltrane's circle diagram through direct musical expression, there are many enhancements that could be implemented in the future. Most importantly, we would focus on improving the amount of feedback the instrument provides to the player as they interact with it. There is currently no indication of button states or softpot position on the instrument, and this could be improved by adding LEDs and labels indicating these states above and below the buttons and around the softpot. While we enjoy the transparency of the acrylic and silicone that allows for the light to shine through, there is low visual contrast between the keys and the acrylic itself. Other desired features include individual chord building for each large FSR, more robust coding of LED colors to pitch classes, and the ability to record and play back loops to practice soloing over chord progressions.

We were very excited about the musical possibilities that the Coltrane Circle inspires, and we did what we set out to do in constructing a physical interface for what was once an abstract idea. We hope that the iColtrane delights all musicians, both studied jazz theorists and novice players alike.