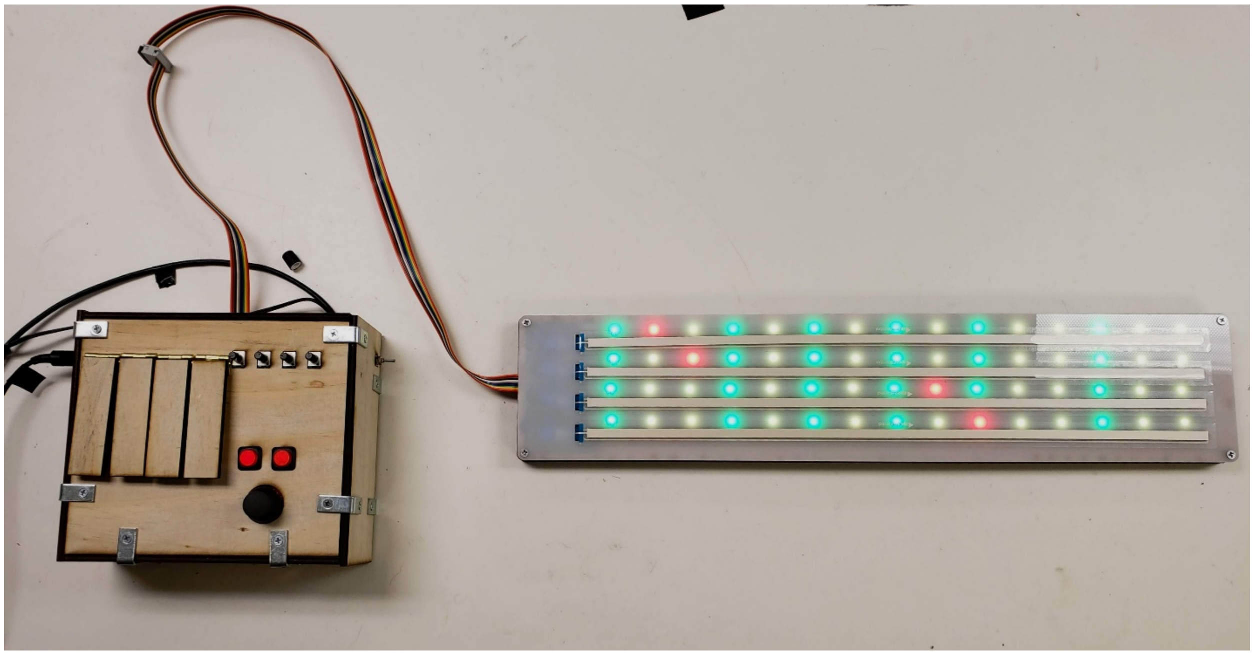
Benny Roover
5/8/19
EMID Final Report

The Glide Rule

For our final project, my team (Yekwon Park, Paige Shephard, Erick Orozco, and myself) created an instrument called the Glide Rule. Our goal for the project was to build an expressive, playable instrument in a unique layout that maintains an intuitive and familiar feel. As such, the device has four channels of pitch input configured and tuned like the fretboard of a bass guitar to be played with the right hand, as well as four accompanying velocity-sensitive keys to articulate notes with the left hand. The user plays a note by touching his or her finger to a “string” to set the pitch, then striking one or more keys. The pitches remain when the player removes their finger, which makes shaping chords much easier. Each pitch channel, or “string,” can send both discrete pitches (referred to as “fretted” notes) and continuous pitch data, allowing for precise and expressive performance. The instrument also has two octave buttons, a sostenuto pedal, and a joystick and knobs for changing parameters,

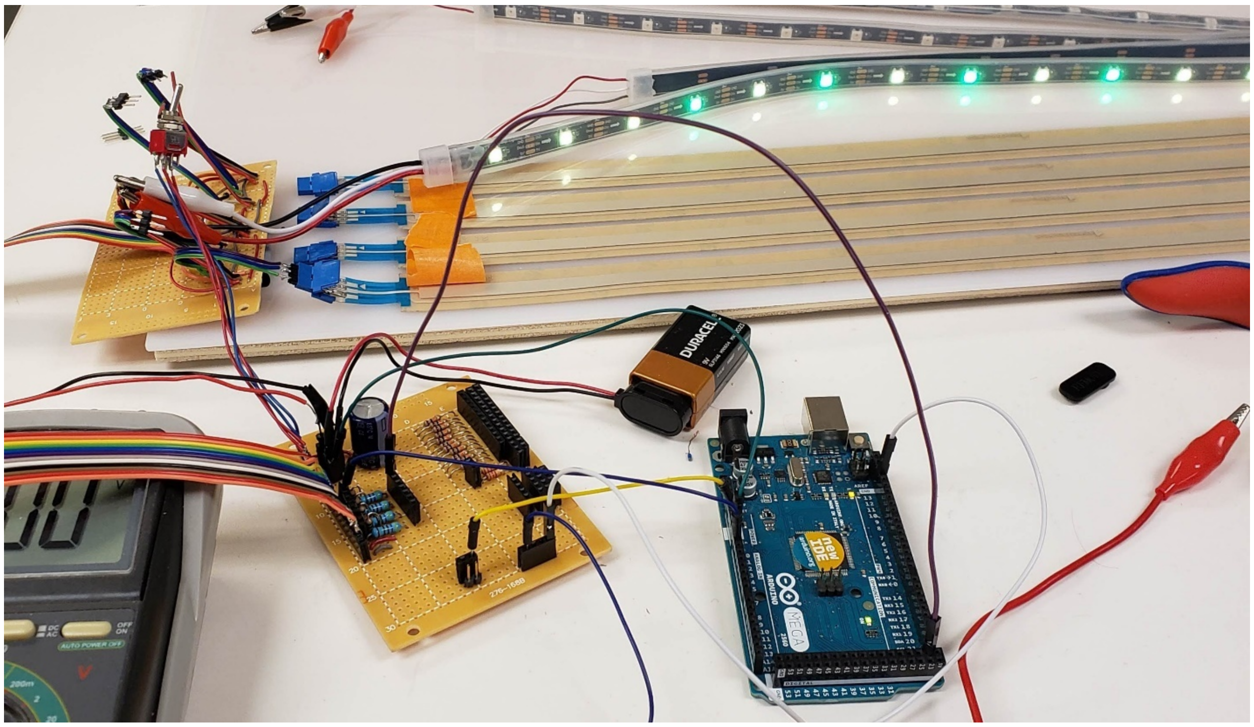
Figure 1 – The Glide Rule



We chose to construct the instrument from laser-cut wood and translucent white acrylic. The wooden “articulation” section is a simple box with a faceplate to house the buttons and knobs. Inside the box is an Arduino and an accompanying circuit board to bias the various electronics. The “fretboard” section, consisting of four channels of Linear Softpots and LED strips, gets power and input/output from the main housing via a detachable twelve-pin connector. The fretboard is constructed from four layers of laser-cut wood with cavities to house the LED strips, Softpots, and circuit board. The LED strips shine through the translucent acrylic, giving a nice effect.

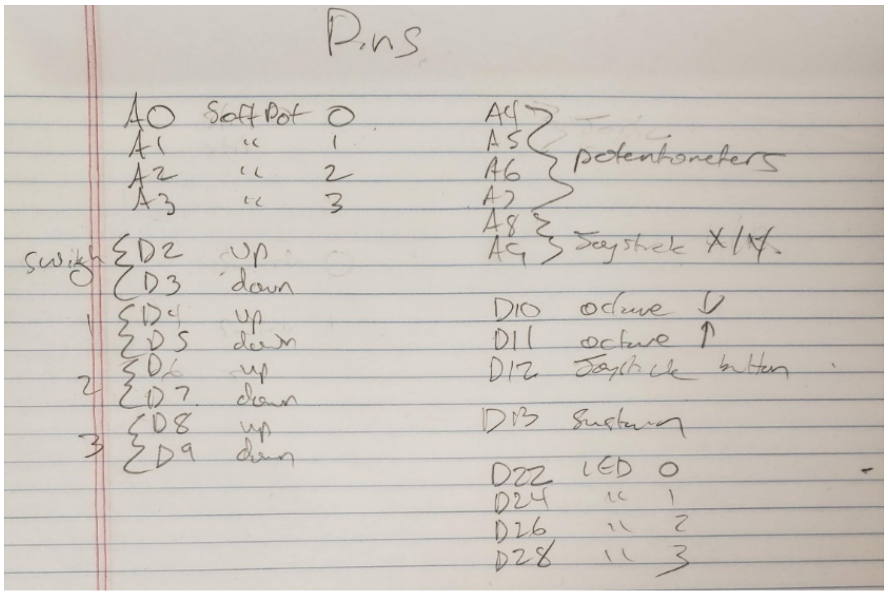
To implement our strings, we chose to use Linear Softpots, which produce a changing, continuous voltage as the user drags his or her finger across the device. Each of the Softpots has a designated Adafruit LED strip that provides visual feedback by indicating the active note with a red LED. The LED strips also outline the shape of a fretboard, with green lights indicating frets 0, 3, 5, 7, 9, and 12. In the articulation section, each key sits atop a single-pole, double-throw, break-before-make pushbutton that allows us to measure the velocity of a note press. We also implemented four rotary potentiometers and an analog joystick for further parameter control, and two momentary single-pole single-throw pushbuttons with embedded LEDs to act as octave buttons. We also have a sustain pedal, which is another single-pole, momentary switch. All electronics are powered from f the Arduino's 5 Volt regulator except the LED strips, which have their own 5V DC Power supply.

Figure 2 – The Arduino, main electronics board (bottom center) and unassembled fretboard (top)



All the above devices are fed as inputs or outputs to an Arduino Mega microcontroller. We have ten analog inputs from the Softpots, potentiometers, and joystick, as well as twelve digital inputs from each pole of the pushbuttons and sustain pedal. We also have six digital outputs – one for each data channel for the LEDs, and two for the embedded LEDs in the octave buttons. Our Arduino software controls the LED strips and illuminates the currently active fret on each channel. To do this, we linearize the Softpot data to translate the analog reading into a fret value from 0 to 14. The rest of the inputs are completely unprocessed and are sent straight to our Max patch.

Figure 3 – Pin list for the Arduino (minus octave button LEDs)



Pins

A0 SoftPot 0
A1 " 1
A2 " 2
A3 " 3

Switch { D2 up
D3 down
1 { D4 up
D5 down
2 { D6 up
D7 down
3 { D8 up
D9 down

A4, A5, A6, A7 } potentiometers
A8, A9 } Joystick X/Y.

D10 octave ↓
D11 octave ↑
D12 Joystick button
D13 Sustain

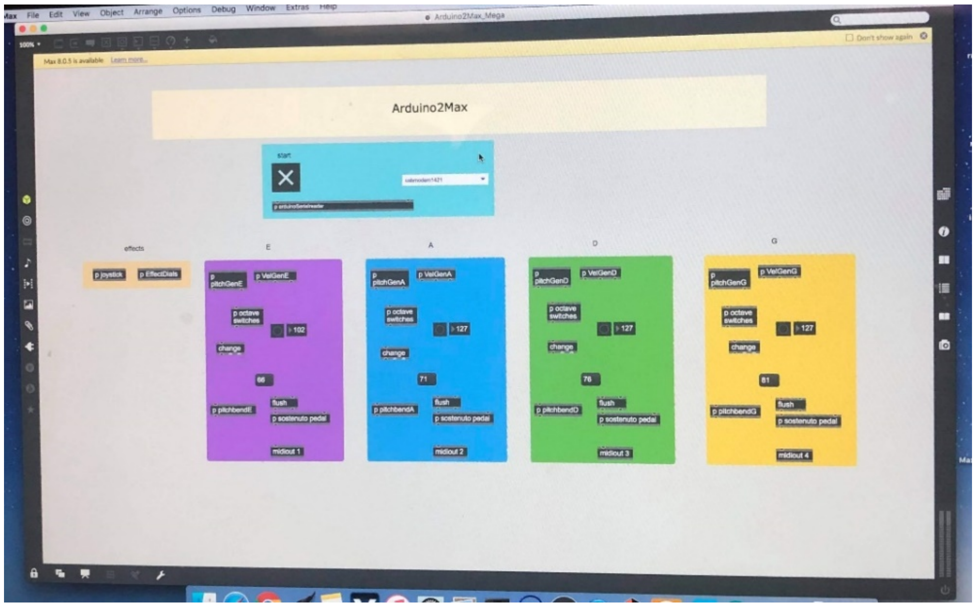
D22 LED 0
D24 " 1
D26 " 2
D28 " 3

Our Max patch consists of a few modules. For each channel, we have a sub-patch that extracts the velocity of a key press based on the duration between when the first conductor breaks and the second conductor makes. To extract pitch, we have a module that linearizes the raw Softpot data into a continuous value representing the number of semitones above the "root note" of the channel. Flooring this value and adding the channel's root note gives us a note number for that channel. Finally, each channel has a pitch bend module that computes the number of continuous semitones above or below the start of a bend and converts this to a midi message via the *xbendout* module. The rest of the data are sent as global parameters: The rightmost potentiometer controls master volume level, and the other three are portamento time, LFO rate, and LFO delay time. The joystick's two axes affect the X and Y position of a formant filter in our Reason patch. Each channel also feeds through a patch that processes the sustain pedal input as a sostenuto pedal, so that the currently held notes sustain while new notes do not. Finally, the two octave buttons control the octave of every channel collectively.

In Reason, we set up four channels of an identical Thor patch. We created this patch from scratch, and it has a mellow triangle-wave oscillator with a high sustain level. We mapped velocity to amp envelope attack, and LFO to oscillator frequency, and an X/Y formant filter alters the sound (and can be used for expression via the joystick as mentioned above). Each instance of the Thor patch is monophonic and thus has its own associated pitch bend, allowing the player to bend one or more notes at once without affecting the others.

For tasks, Erick was mainly responsible for the Max patch, and Yekwon dealt with the Arduino code, including linearizing the Softpots and programming the LED strips. Paige handled most of the physical design, including all drawings and cuts, but our whole team helped with making measurements and deciding on the dimensions and overall look. Finally, I designed and soldered the electronics and connectors.

Figure 4 - The Max interface



There are many improvements we could make to this project were we to continue with this version, particularly on the software side. First, there were many bugs that we did not have time to fix before our deadline. On the day of our presentation, Reason only recognized midi data on one channel due to some unsolved communication error between Max and Reason, rendering the multiple-monophonic nature of our instrument useless. We would have liked to spend time going over our Max and Reason patches to fix this issue. In addition, we could put more time into our Thor patch to showcase the various features of our instrument, mainly by adding longer decay times and higher sustain, and mapping velocity and the LFO to more parameters. Finally, our velocity generation in Max could have been tweaked to offer a smoother velocity curve. Personally, I would have liked to ditch the Max patch altogether and do all processing and midi generation directly from the Arduino, but that would require extra hardware.

Though the development of this project had its ups and downs, overall, I am relatively happy with the results. I think the physical design of the Glide Rule is beautiful and intuitive to play, and it theoretically offers a wide variety of playstyles (if only our software worked!). I would say our biggest mistakes during this project were in time management, evidenced by the fact that the physical design wasn't complete until the morning of the presentation. If I were to build this instrument again, I might consider removing the keys and instead triggering notes when the user presses on the "strings," but then we'd be left with what is essentially a Linnstrument. I would also consider using capacitive sensors instead of Softpots because they provide more accurate position data and require no linearization (this is also how the Linnstrument operates), and as I mentioned before, I would almost certainly prefer to generate MIDI information directly off the Arduino to eliminate Max as the middleman. This likely would be much easier to code, test, and debug, and our Softpots and LEDs could communicate with each other more easily.