Project Report: MIDI Slide

Violet Blitz, in a group with Matthew Dacey, Natalie Nicoletti, and William Xia

Dec 17, 2024

Introduction

The MIDI Slide was designed to be an intuitive, portable, and ambidextrous electronic interpretation of a lap steel. With one hand choosing notes (akin to fingering strings) and the other triggering them (akin to plucking or bowing strings), the interface will be familiar to users with any passing knowledge of string instruments, even if they've never encountered a lap steel before. The instrument features five tuning modes inspired by different instruments so it's especially adaptable, emulating a lap steel, guitar, bass, cello, viola, or violin tuning. By being a MIDI instrument, the MIDI Slide can give players access to various sounds and effects that would otherwise require an expensive lap steel and multiple effects pedals. Finally, the instrument features a "tapping mode" for one-handed play, allowing it to easily be used in addition to other instruments.

The core feature of the MIDI Slide is the sliding itself: four softpots affixed to the main body of the instrument allow players to seamlessly slide between notes, reproducing the laid-back glissandi of the lap steel or the bluesy bends of an electric guitar. Each softpot can bend its pitch independently, putting the instrument in the MPE category alongside the ROLI Seaboard and the Linnstrument. While simpler than those instruments, the MIDI Slide has the advantage of not needing the player's hands to jump around the keyboard. Four softpots and four buttons fit easily under the player's fingers, allowing expressive play without difficult leaps–and the new wireless design allows the off-hand a greater freedom of movement.

Instrument Layout

The instrument is comprised of two unconnected components. The first, called the pad, can be placed on a tabletop, a lap, or attached by a strap around the player's neck or waist. The pad features four softpot variable resistors, meant for use by the player's pointer to pinky fingers, as well as four dials and two buttons that are used to change the instrument's tuning and volume between songs. While the knobs determine the "root note" of each of the four "strings", the softpots are used to control the pitch bend along the range of an octave to modulate the pitch of each string. The softpots are intersected by etched lines that clearly show where to press to hear each half step (12 in total) along the string.

The second component is the handheld controller, which attaches by a strap to the player's other hand. The components are both designed such that they can be used ambidexterously by whichever hand the player prefers. The handheld controller has four force-sensing resistors, meant for use by the player's pointer to pinky fingers, and a joystick positioned under the thumb. The force-sensing resistors are used to trigger notes in plucking mode and control note volume in both modes, while the joystick can be clicked to switch between plucking and tapping mode or flicked to toggle on or off four effects: chorus, phaser, overdrive, and reverb. Additionally, an accelerometer hidden within the handheld controller reads the angle it's held at and uses that information to affect the sound: the controller's pitch is tremolo amount, yaw is tremolo speed, and roll is filter frequency. By pressing buttons, changing angles, and flicking a joystick, the off-hand still has an incredible amount of musical control.

Electronic components, including an Arduino microcontroller, are hidden within the two components, out of sight from the player. The pad contains an Arduino Mega and an ESP 32-S3, while the handheld controller has an Arduino Nano.

Playing the Instrument

The instrument features two different playing modes: plucking mode and tapping mode, which can be switched by pressing the joystick on the handheld controller. While they change the instrument significantly, the tuning dials, volume dial, octave buttons, joystick controls, and accelerometer controls work the same in both modes.

The MIDI Slide features two tuning dials and two octave buttons, allowing players to choose what register, key, and tuning pattern the instrument is in. These affect the "root note" of each softpot, meaning the note they sound when the softpot is at "0", either untouched or touched at the very bottom. The softpots can then be used, regardless of which mode they're in, to bend the pitch of that root note up to an octave above.

The octave buttons are self-explanatory, changing the octave of the "root note" of each softpot. When both octave buttons are pressed at once, however, the synthesized sound of the instruments changes, rotating between a lap steel sound, cello sound, and extraterrestrial synth pad sound. The root note dial determines the pitch class of the leftmost softpot's root note. In conjunction, the octave buttons and root note dial mean the leftmost softpot's root note could be anything from C2 to B6. Then, the tuning mode dial determines the root notes of the other three softpots, based on the first. "Open", "6", and "7" are all conventional lap steel tunings. These three tunings all tune the second and third strings the same: the second is a major third above the first, while the third is a perfect fifth above the first. The only difference is the rightmost softpot, being an octave above the first in Open, a major sixth above in 6, or a minor seventh above in 7. "4ths" and "5ths" are unconventional for lap steels, but are based on other instruments. In 4ths, each softpot is a perfect fourth apart, as in a guitar, electric bass, or double bass. In 5ths, softpots are a perfect fifth apart, as in a cello, viola, or violin. Users can easily choose whichever tuning makes the most sense for them or for the piece they're playing by simply turning a couple of dials.

The Auto Tuning knob affects the behavior of the softpots: turning the knob toward “Smooth” creates a smoother glissando between notes, while going toward “Half Steps” makes it easier to play notes in tune.

The volume knob controls the overall volume of the instrument. It’s generally meant to be set once and not adjusted while playing, but can certainly be adjusted mid-song if necessary.

The joystick is used to switch between modes and turn on or off four effects. Pressing down on the joystick will toggle between plucking (default) and tapping mode. Flicking the joystick left toggles the chorus effect, right toggles the phaser effect, down toggles the overdrive effect, and up toggles the reverb effect.

In both modes, pressing the FSRs on the handheld controller hard enough will activate an aftertouch effect, increasing the volume of both oscillators on the corresponding channel. The FSR values also determine the brightness of four LEDS on the pad, so there’s visual feedback for the FSRs.

Now to the two modes. In plucking mode, the FSRs are used to trigger notes and determine their velocity. A slow press will result in a slow-attack note with a low velocity and a fast press will result in an instant-attack note with a high velocity. The notes will be held for as long as the FSR is held down, meaning that they also act as sustain controls. While one hand controls the FSRs, the other uses the softpots to change the pitch bend of the notes - each softpot and FSR are assigned to a channel, so the leftmost softpot controls the pitch of the notes triggered by the topmost FSR, and so on.

In tapping mode, the FSRs only act as aftertouch, as described above. Touching a softpot will both change the pitch bend and trigger a note, always with maximum velocity and instant attack. The note will continue to sound until the softpot is released, then it will pitch bend back to the softpot’s root note, and quickly fade out.

Instrument Construction

Physical Construction

The main pad of the instrument has one side made from a sheet of 3 mm black acrylic and its other five sides are 3 mm birch wood. All six were cut using a laser cutter, and the top acrylic sheet also had lines and labels laser-etched onto its surface. The wood sides are held together with wood glue while, for the prototype at least, the acrylic is simply held in place by finger joints. This allows the acrylic to be easily removed to access the electronic components housed within the box. Holes cut into the wood act as exit points for wires on the back face. The knobs are secured to the acrylic by a nut that holds the acrylic against the wider body of the knob, while the octave buttons are simply secured by gravity and the softpots are secured by an adhesive on the back of each. The wood was painted black to better match the acrylic top and 3D-printed components. Additionally, the entire pad was made smaller in our second iteration to be more comfortable for the player.

The handheld controller, 3D printed using black PLA, was printed in three parts, later fastened together with bolts and nuts, to have a large, accessible internal cavity to fix wires and components into. This second iteration is much larger, accommodating the Arduino Nano, 9V Battery, and power switch that was added for wireless communication. The battery holder, joystick, and accelerometer are hot-glued into the controller, while the circuit board is secured by light pressure on its sides, and the FSRs are secured by adhesives.

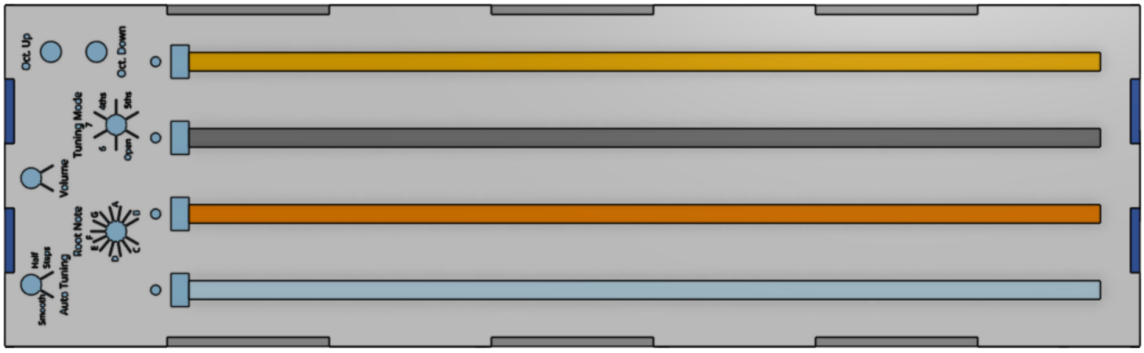


Figure 1: Tuning Labeling on the Main Pad

Parts List

Part	**Quantity**	**Link**
3 mm black acrylic		
3 mm birch wood		
Black PLA for 3D printing		
Varied bolts and nuts for fastening		
Soft pot	4	https://www.digikey.com/en/products/detail/spectra-symbol/tsp-l-0300-103-3-st/17050959
Rotating pot	4	https://www.adafruit.com/product/562

Knob	4	https://www.adafruit.com/product/2058
Button	1 (pack of 4)	https://www.digikey.ca/en/products/detail/sparkfun-electronics/PRT-14460/7915747
FSR	4	https://www.adafruit.com/product/166
Accelerometer	1	https://www.adafruit.com/product/3886
Joystick	1	https://www.adafruit.com/product/512
Elastic band	1	https://www.amazon.com/Elastic-Black-Heavy-Stretch-Elasticity/dp/B071G3J5QG/ref=sr_1_6?dib=eyJ2IjoiMSJ9.UzOLpL5VnchJA0OfhSlHzjVzHgHov7frPtV-W65VGIXNbLNt_sgI5Hprwb1NC5g2djU9mXTZjAByZT10QFNz9NGqRE-dK89IMcCGhM8aXmVfx6JGvBkdEU01S4EwxlpvvUxWHAI96x7Ilar5eDWn0RJLOG38Ft7KwuA1-8qwdtDszHajvhj7jHAoOfIQnYFLI3m0DgN9K2L0-Ws2B6eqdiVgryvPShkPen02pxDLoSL9iLkXQcRHb7n63pTw4untY13HXwgfSao5xifu-RwGjaR7FS0sIOooraVmrjNpC8U.vnFhVb474kiohFfrC9hksqGp3JTcqg9d2MlfmREzGp4&dib_tag=se&keywords=heavy%2Bduty%2Belastic%2Bband&qid=1728860066&sr=8-6&th=1=
Guitar strap	1	https://www.amazon.com/Ernie-Ball-Black-Polypro-Guitar/dp/B0002D0E92/ref=sr_1_1?crid=1X9KGI5FJEVDR&dib=eyJ2IjoiMSJ9.si-xNzyUrbH2cySvY36gPVuAodO_GAIjr689t2L3FK4W0Wm_-MDUxHXafTUzljWMBYPLGWk8F0982syeTIJZ8-jHls6rETz8bRaSKdCST-BqAqGkhkB6L8CBIXR5cdUF8ZqoBjDWHF6wEQaJcBiq1RK1iKceKVTYGU5ufm7zsV_gjZdUvWw65AbELAjyNw0lf-euKxxSj6yTXvUjHePupnhIAaIHy7Ufo42T8cUdIPiHtOoWhUL0z9NZIX-2qqOAI5FeVysbiSKfxCvLhcws6IgrmMmj9ZkG0x-ciKFQlCQ.EQ1zWCfP5BKhZsyeidyRSkxeZIbFzurM8q8Xk70kI7Y&dib_tag=se&keywords=ernie+ball+black+guitar+str

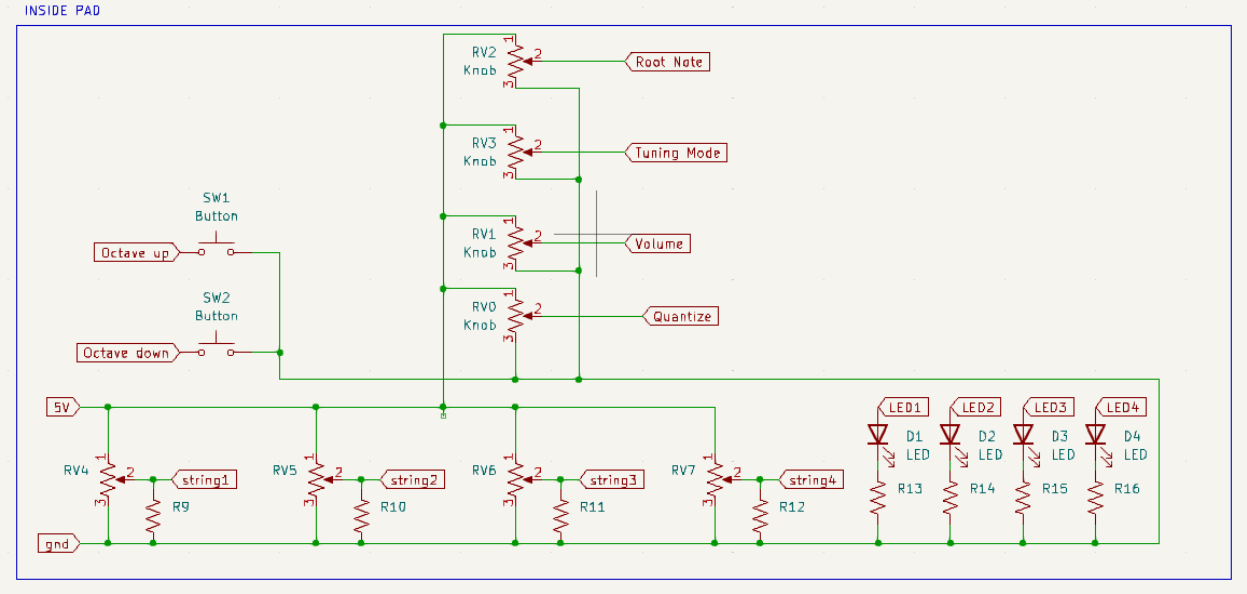
		ap&qid=1731515280&sprefix=ernie+ball+black+gu%2Caps%2C168&sr=8-1
Guitar Strap holder- silver	1 (pack of 4)	https://www.amazon.com/Giantree-Security-Mounting-Acoustic-Electric/dp/B0DCNCLVBR
On switch	1 of each	https://www.adafruit.com/product/1443?gad_source=1&gclid=CjwKCAiAudG5BhAREiwAWMlSjNDyBAZ3cNBjZnExu8rtNBTBt6m_9kz8heWABNdFpMf0FT_fVn2-KBoCuQ8QAvD_BwE AND https://www.digikey.com/en/products/detail/e-switch/RR511D1121/2116256
Battery holder	1	https://www.digikey.com/en/products/detail/bud-industries/HH-3634/3681242?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMax%20Shopping_Product_Low%20ROAS%20Categories&utm_term=&utm_content=&utm_id=go_cmp-20243063506_adg-_ad-__dev-c_ext-_prd-3681242_sig-CjwKCAiA3Na5BhAZEiwAzrfagOdKg5HhH9TtDOIYPGcf1sbCOC3R5pG71NUKlGLCyOKkXeP2x39sERoCc30QAvD_BwE&gad_source=1&gclid=CjwKCAiA3Na5BhAZEiwAzrfagOdKg5HhH9TtDOIYPGcf1sbCOC3R5pG71NUKlGLCyOKkXeP2x39sERoCc30QAvD_BwE
Arduino Nano	1	https://www.amazon.com/Arduino-ABX00083-Bluetooth-MicroPython-Compatible/dp/B0C947BHK5/ref=asc_df_B0C947BHK5?mcid=5a30f294e1e6319b9f1c315bf04bf5e6&tag=hyprod-20&linkCode=df0&hvadid=693535620612&hvpos=&hvnetw=g&hvrand=12973110678850364259&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9002012&hvtargid=pla-2187047704270&psc=1
ESP 32	1	https://www.espressif.com/en/products/socs/esp32-s3

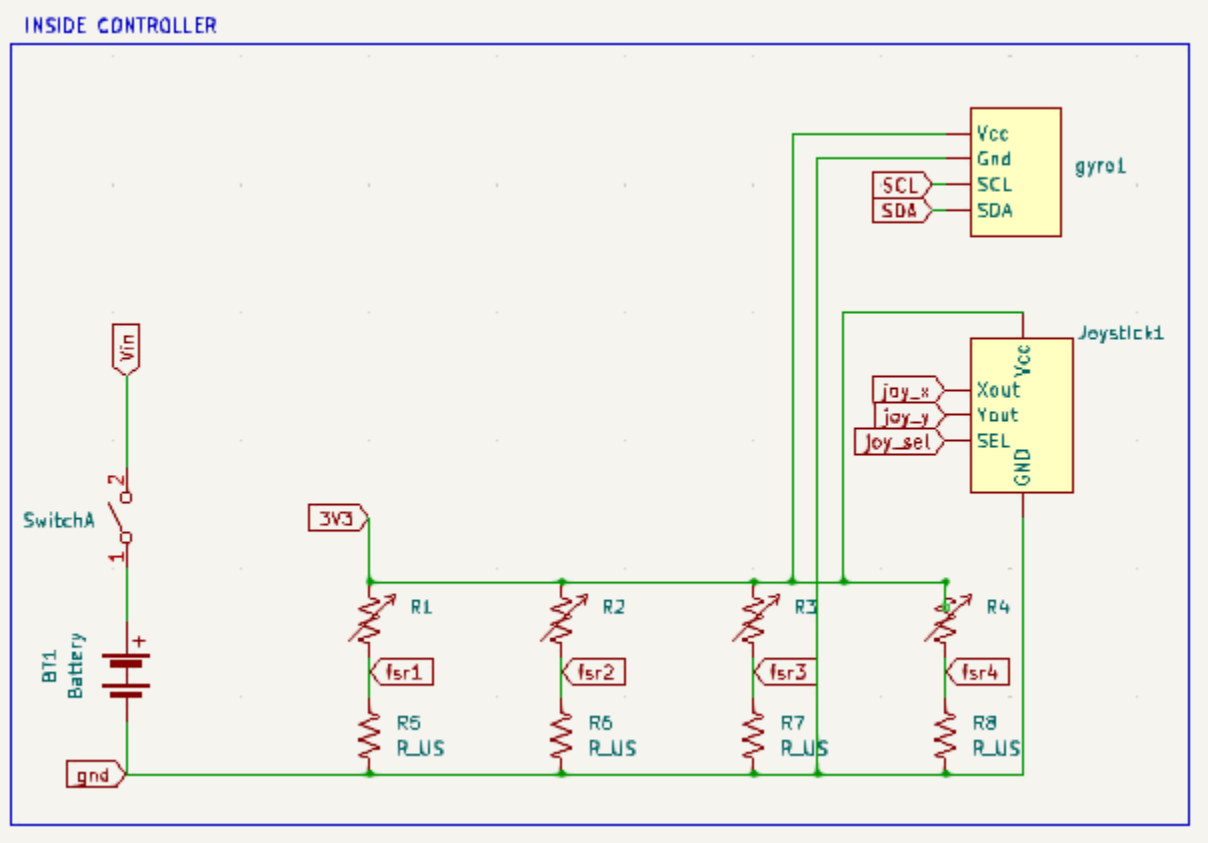
Circuitry

Inside the pad, each sensor except the octave buttons is wired to have +5V power and ground from the Arduino Mega and to send analog or digital signals back to the Arduino through its output pin. The softpots are also connected to ground with pull-down resistors, so they default to an output of 0 when not being pressed. Finally, the LEDs are connected to receive a PWM signal from the digital output pins of the Arduino and are wired in series with voltage-dividing resistors. The octave buttons are simple switches so they provide a digital output to the Arduino and require fewer wires.

The handheld controller packs a lot of components into a smaller space, but the wiring is fairly simple. A rocker switch connects a 9V battery to the Arduino Nano's V_{in} port, and the battery's ground serves as the common ground for all of the components. The sensors receive 3.3V from the Nano since a 9V input would overload them. The sensors are all wired to the Nano's pins via a small proto-board, and the FSRs have additional voltage-regulating resistors as well.

Refer to these schematics (made by Matthew Dacey) of the main pad and handheld controller's internal wirings.





Matt also put together these charts of how each component connects to the microcontrollers:

Main Pad (Arduino MEGA 2560)

COMPONENT	**PIN**	**INPUT TYPE**
SoftPot 1	A0	Analog
SoftPot 2	A1	Analog
SoftPot 3	A2	Analog
SoftPot 4	A3	Analog
Root Note Knob	A11	Analog
Tuning Mode Knob	A10	Analog
Volume Knob	A9	Analog
Gravity well Knob	A12	Analog
Octave (+) Button	22	Digital

Octave (-) Button	23	Digital

*Additionally, the Main Pad contains 4 output pins for the 4 LEDs used to indicate each “string” being played. The output pins used are 2, 4, 5, and 6.

Handheld Controller (Arduino NANO ESP32)

COMPONENT	**PIN**	**INPUT TYPE**
FSR 1	A1	Analog
FSR 2	A0	Analog
FSR 3	A2	Analog
FSR 4	A3	Analog
Joystick X value	A6	Analog
Joystick Y value	A7	Analog
Joystick Button	D12	Digital
Gyro SDA	SDA (A4)	Digital
Gyro SCL	SCL (A5)	Digital

Programming

Wireless Connectivity

The wireless connection between the handheld controller and the main pad is made possible by an Arduino Nano in the controller and an ESP32-S3 in the pad. They communicate using the ESP_NOW protocol, which sends data from the Nano to the ESP32’s local wireless network. The data Arduino Nano reads the inputs of the handheld controller’s sensors: the joystick, FSRs, and accelerometer and formats them for use in Max. The joystick is
As stated earlier in the circuitry section, our instrument uses 3 separate microcontrollers to facilitate the wireless communication between the controller and the main pad. For the joystick, each direction of movement is used as an on/off toggle for 4 different settings, and pressing down the joystick similarly toggles tapping mode. As a result, there is a function in the controller that determines the current cardinal direction of the joystick and then changes the value stored in a local variable for the corresponding setting (chorus, phaser, reverb, overdrive) from true to false (1 or 2, we use 1 or 2 instead of 0 or 1 because that’s the convention in max) or vice versa.

The FSRs are simply scaled to 0-127, as are the pitch, yaw, and roll of the accelerometer. This data is then sent via ESP-NOW to the ESP32, which is connected directly to one of the serial ports of the main pad's Arduino Mega. All the Mega has to do is take the incoming data and append it to its printing line, so both the pad's sensors and controller's sensors are sent to Max at the same time.

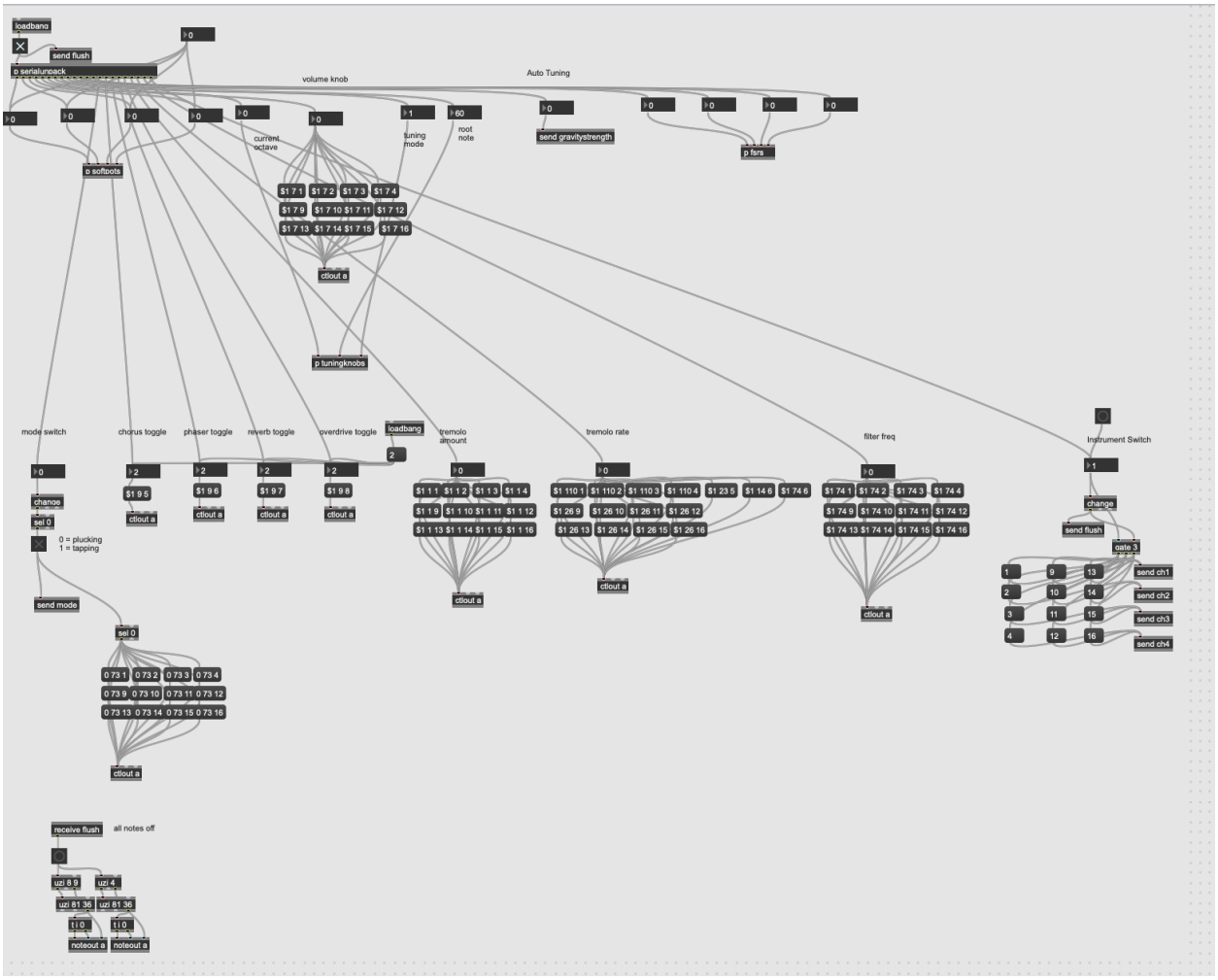
Arduino Code

Our Arduino is coded via Arduino IDE to interpret the analog and digital signals of the pad's sensors into data formatted for Max. It takes raw data from the sensors every 10 milliseconds. It was initially every 50 milliseconds, but that caused some noticeable latency, and we were luckily able to speed it up without evident drawbacks. Raw data from analog ports, ranging from 0-1023, it reinterpreted into ranges used by Max: 0-1200 for the softpots, 0-127 for the volume knob, 60-71 (analogous to MIDI notes C4-B4) for the root note knob, 1-5 (for the five different tuning modes) of the tuning knob, and 0-50 for the auto-tuning knob. The octave buttons required more complicated interpretations. The octave buttons increment an "octave" variable up or down 1, with a range of -2 to 2. Pressing both octave buttons at once increments the instrument switch variable between 1, 2, and 3. The Arduino Mega formats all this sensor data, appends the data it receives from the ESP-32, and prints these output values in a single line, each sensor or variable's output separated by a space, every 10 milliseconds. Finally, the Mega is also coded to light up four LEDs on the pad in proportion to how hard the FSRs on the handheld are being pressed.

Max Patch

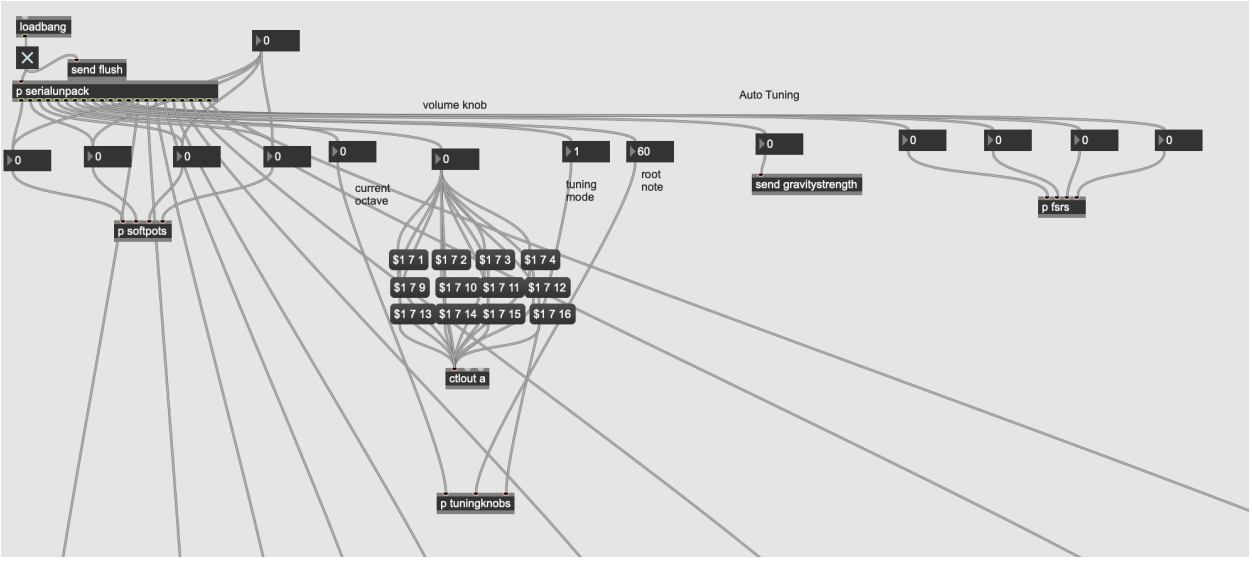
The Max patch functions as a modular system that interprets each output of the Arduino to send notes and control information to Reason.

Overall Max Patch:

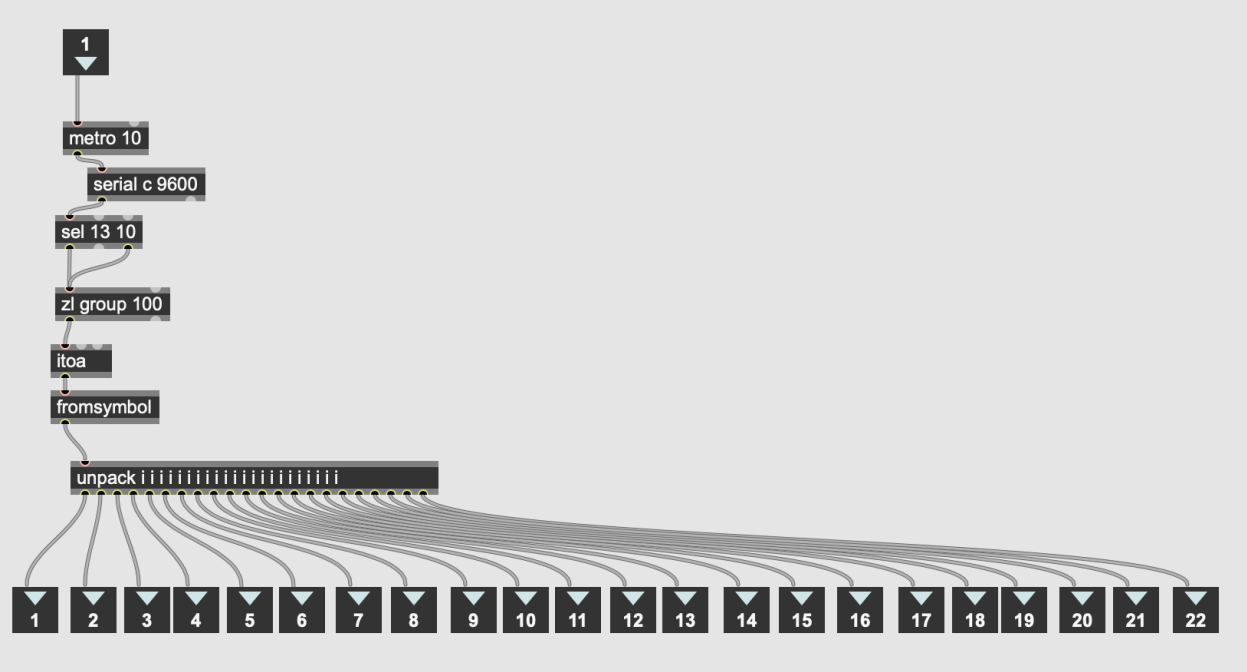


The patcher has many sections and sub-patches, so we'll start from the top left and move down towards the bottom right. The function to admit and interpret the output of the Arduino, "p serialunpack" was adapted from a YouTube video, "Arduino To Max/MSP (Tutorial)" by Sound Simulator.

Top row of the overall max patch:



P serialunpack:

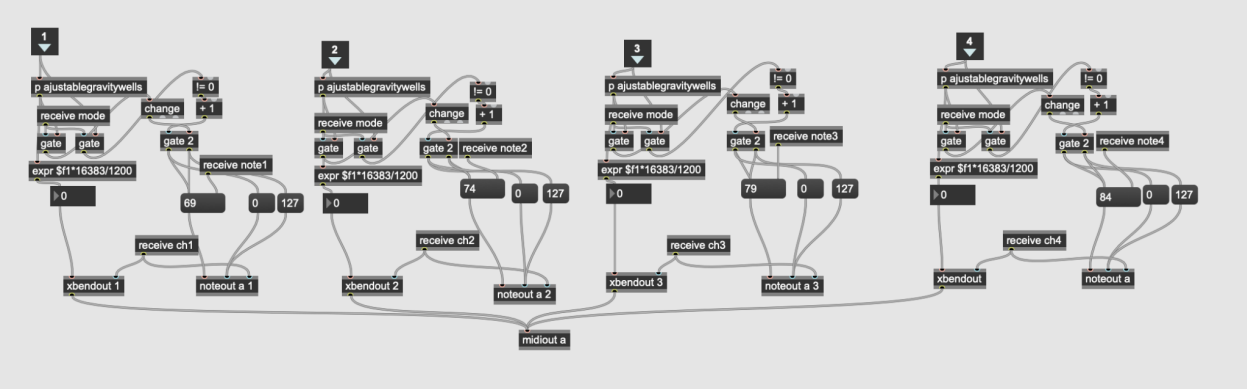


Once the initial toggle is turned on by the loadbang, the “p serialunpack” function triggers an output from the serial port every 10 milliseconds, groups the incoming values, and converts them to integers, resulting in a list of 20 values. The values are then sent out of 22 ports to the 22 different number boxes in the main patch.

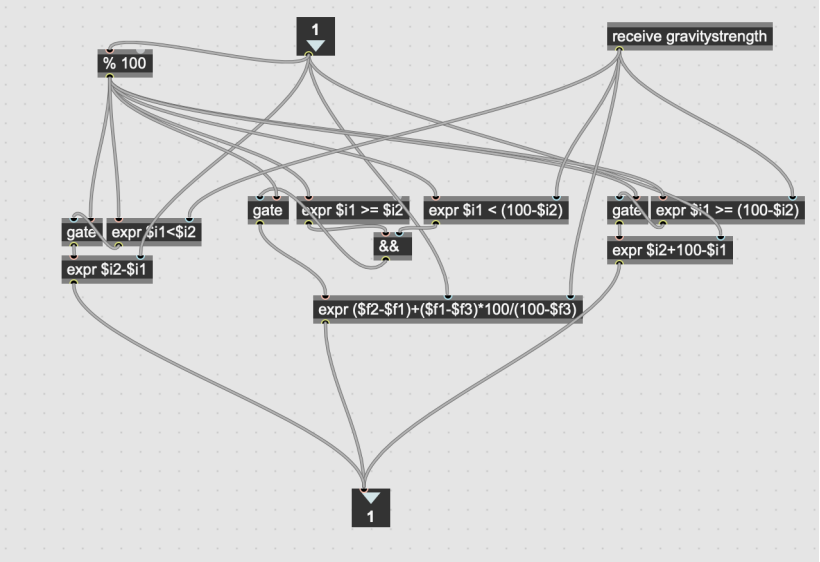
The first four inputs from Arduino are the current position of the finger on the linear softpot, expressed from 0 (bottom) to 1200 (top). In the “p softpots” sub-patch, each softpot does two things: sends a “fudged” pitch bend value to the corresponding channel in Reason, and, when in tapping mode, triggers note-ons for that same channel. Each softpot is assigned its own MIDI channel so each can have independent pitch bend values. The leftmost softpot, acting as

the “lowest string” of the lap steel is channel 1, 9, or 13, depending on the instrument switch variable, while the rightmost “highest string” softpot is channel 4, 12, or 16..

P softpots:



P adjustablegravitywells:



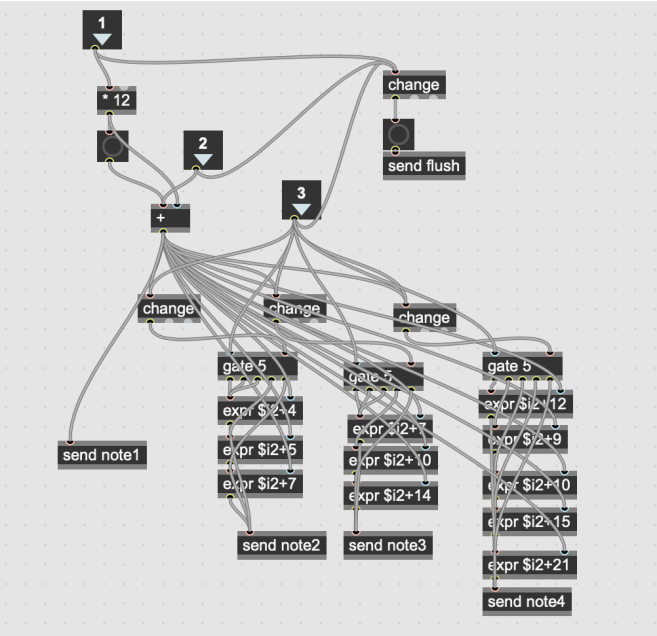
The incoming 0-1200 value of each softpot is sent through a “p adjustablegravitywells” function that fudges the pitch, making it easier to play in tune by increasing the “target” range of inputs that result in the in-tune semitone outputs 0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, and 1200. The size of the target zones for these numbers depends on the “gravitystrength” variable, which is determined by the “Auto Tuning” knob, from 0 (no fudging) to 50 (acting as a fretted instrument). The output of “p adjustablegravitywells” is sent through the xbendout function to affect the pitch of the corresponding channel.

When in tapping mode, a change in the output of “p adjustablegravitywells” also triggers a note to be played on the corresponding channel. Each channel is in monophonic legato mode, so the continuous noteon commands do not sound individually. The pitch of the note is determined by the tuning knobs and adjusted by the softpot’s pitch bend value, and its velocity is always 127, so the note will sound regardless of the FSRs (as opposed to plucking mode). Pressing down on the FSRs will still increase volume, as explained later.

One special case is the softpot input of 0. Since the softpot is wired using pull-down resistors, its output defaults to 0 when it isn't being touched. When a 0 is received while in tapping mode, the "p softpots" module will trigger a noteoff command for all notes on the channel. This function takes input from the Arduino's output itself instead of from "p adjustablegravitywells" because "p adjustablegravitywells" also interprets values 1-20 as 0, as mentioned earlier. But input values of 1-20 would mean the softpot is being pressed, so this function must be limited to only values of 0 from the Arduino. In theory, only a noteoff command for the currently playing note should be necessary, but edge cases arose in testing and we determined it was better to ensure all notes on the channel were off.

The next module to discuss is "p tuningknobs." This function takes the octave buttons, root note dial, and tuning mode dial as its inputs. The Arduino code has already manipulated the sensor's values, so the octave buttons combine to change a single integer between -2 and +2, the root note dial has outputs 60-71, and the tuning mode knob has outputs 1-5.

P tuningknobs:

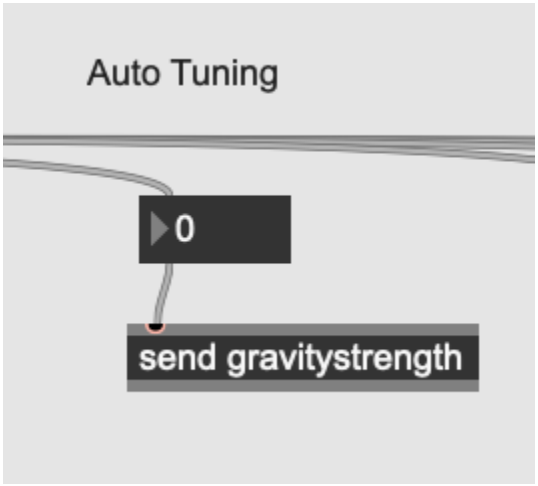


This module interprets the three inputs to send variables "note1," "note2," "note3," and "note4," which will become the pitch component of noteout commands in the "p softpots" and "p fsrs" patchers. The root note, with a value of 60 (C3)-71 (B3) is added to the octave integer multiplied by 12, becoming the bottommost note of the leftmost softpot (and the first channel). So, an incoming root note value of 60 and an octave value of -1 would result in the leftmost softpot's lowest value being a C2 (48). As written, C2 should be the leftmost softpot's middle value, not the lowest, since a pitchbend value of 8192 would have no pitch bend and reproduce the pitch determined by the MIDI note command. This is resolved in the Reason patch, where each oscillator is tuned up by 6 semitones.

Based on the gate selected by the tuning mode knob, the other three channels have a root note that relates in some way to the first channel's root.

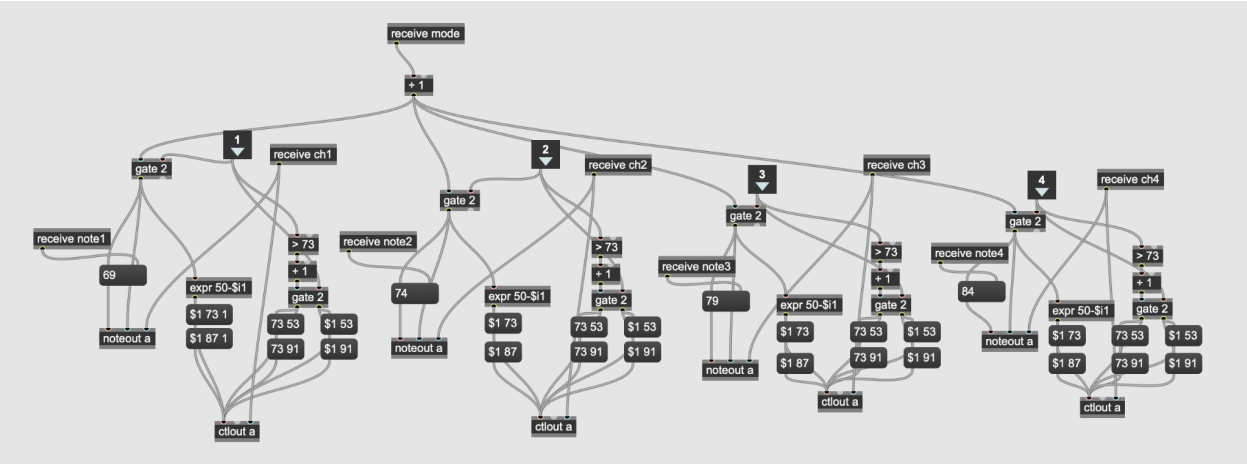
- In mode 1, labeled “Open” on the knob, the second channel is 4 half steps (major third) above the first, the third channel is 7 half steps (perfect fifth) above the first, and the fourth channel is 12 half steps (octave) above the first.
- In mode 2, labeled “6” on the knob, the second channel is 4 half steps (major third) above the first, the third channel is 7 half steps (perfect fifth) above the first, and the fourth channel is 9 half steps (major sixth) above the first.
- In mode 3, labeled “7” on the knob, the second channel is 4 half steps (major third) above the first, the third channel is 7 half steps (perfect fifth) above the first, and the fourth channel is 10 half steps (minor seventh) above the first.
- In mode 4, labeled “4ths” on the knob, the second channel is 5 half steps (perfect fourth) above the first, the third channel is 10 half steps (minor seventh) above the first, and the fourth channel is 15 half steps (minor tenth) above the first.
- In mode 5, labeled “5ths” on the knob, the second channel is 7 half steps (perfect fifth) above the first, the third channel is 14 half steps (major ninth) above the first, and the fourth channel is 21 half steps (major thirteenth) above the first.

Auto Tuning:



The Auto Tuning knob, as aforementioned, sends its output to “gravitystrength”, impacting the “adjustablegravitywells” patcher.

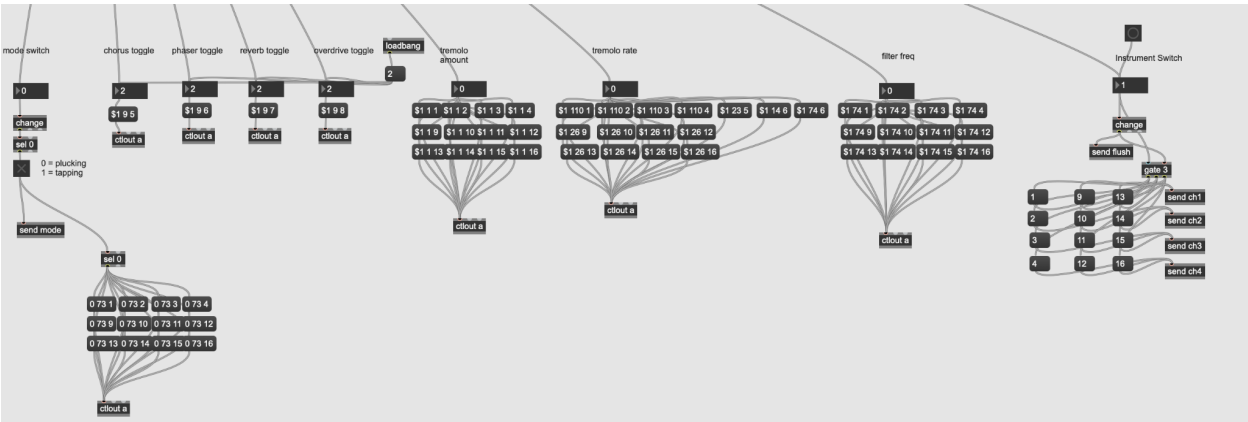
Next is the control module for the FSRs, taking the four FSR values 0-127 as its inputs.
P fsrs:



Like “p softpots,” this module functions differently based on which mode it’s in. In both modes, FSR values between 73 and 127 work an aftertouch volume control for each channel, giving notes a little more volume by turning up Oscillator A Gain and Oscillator B Gain when pressed hard. In plucking mode, the FSRs also control velocity for their own noteouts, triggering a new one every time they are updated, so every 10 milliseconds. Again, the legato mode of the Subtractor instruments, plus the fact that many of these noteon commands have a velocity of 0, make the high density of noteon commands more reasonable. Additionally, when the FSR values are under 50, they’ll also affect the attack speed of the synthesized sounds. Therefore, an FSR value of 10 will result in a quiet note with a slow attack, while a value of 60 or above will result in a louder note with an instantaneous attack.

As “p fsrs” is the last of the subpatches, we can finally move on to the simpler parts of the Max patch.

Second row of the overall Max patch:

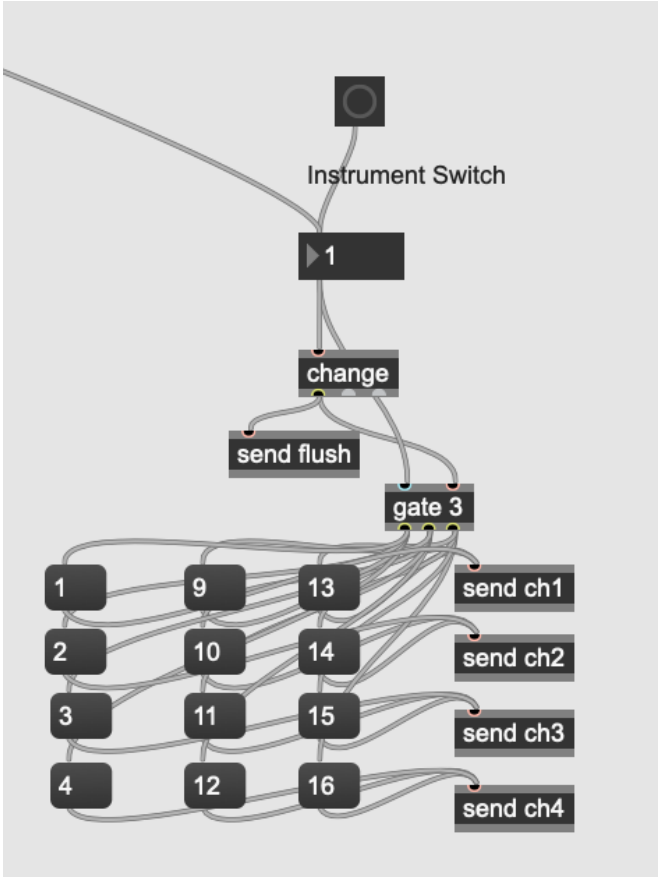


The mode switch, toggled by clicking the joystick, switches between plucking mode and tapping mode. It sends the mode variable out to “p softpots” and “p fsrs”, adjusts the target zone size of the gravity wells as mentioned above, and ensures that the attack speed is instantaneous when in tapping mode.

The next four toggle controls are activated by quickly flicking the joystick in four cardinal directions. They control the Off/On/Bypass switch on four insert effects in the Reason rack. Since the mixer's audio is being sent through the four effects, they should never be in the Off position, only On or Bypass, which allows input audio to pass through to the output unaltered. Therefore, the default position of these four toggles is 2, meaning Bypass, and the value will be switched to 1, turning the effect on when the joystick is flicked. A second flick will turn it back to Bypass. These flicks are interpreted into toggled values of 1 and 2 in the Arduino code.

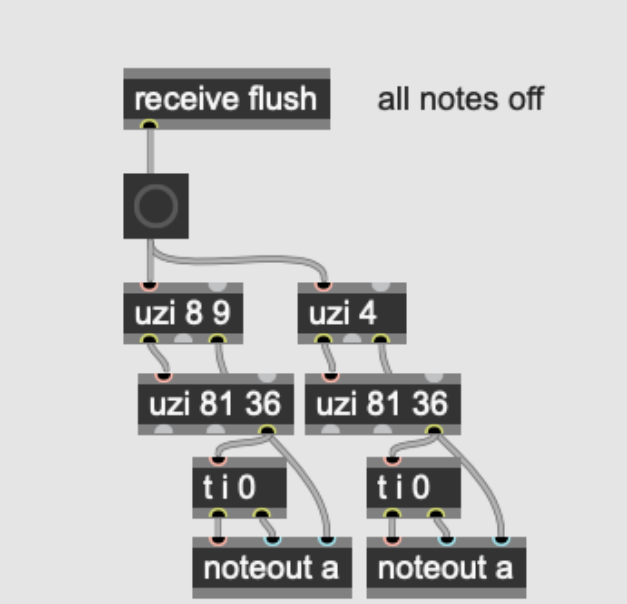
The next three inputs are the pitch, yaw, and roll outputs of the accelerometer in the handheld controller. Pitch determines the tremolo amount, yaw controls the tremolo rate, and roll controls the filter frequency. While the tremolo amount defaults to zero in the resting position, the tremolo rate and filter frequency default to 64.

Instrument switch:



The last input from the Arduino is the "instrument switch" variable, a number 1-3 that increments when both octave buttons are pressed simultaneously. Through a gate, this variable activates a set of channel numbers - 1-4, 9-12, or 13-16, which are then sent to the softpots and fsrs' noteout and midiout commands to switch which instrument is being played. A value of 1 corresponds to the lap steel instrument (Mälstrom) on MIDI channels 1-4, 2 is the cello sound (NN-19) on channels 9-12, and 3 is the pad sound (Thor) on channels 13-16.

The final part of the Max patch is the “all notes off section.”
Bottom-left corner of the overall Max patch:



A “flush” command is sent whenever the octave switch or tuning knobs change and whenever the Max patch as a whole is turned on or off via the topmost toggle button. This function receives that flush and turns it into a series of 972 bangs that turn off every note from 36 (C2) to 117 (A7) on the twelve instrument channels. In testing, we encountered many problems with hanging MIDI notes, so this was an effective way to ensure all notes were stopped.

Reason Rack

The Reason rack takes the MIDI messages from Max and turns them into audio. The Reason Rack consists of a Line Mixer 14x4, 4 insert effects: A Quartet Chorus Ensemble, PH-90 Phaser, Scream 4 Sound Destruction Unit, RV7000 MkII Reverb, four identical Malström Graintable Synthesizers, four NN-19 Digital Samplers, and four Thor Polysonic Synthesizers. The synthesizers each receive input on a set of four channels, each corresponding to a softpot and FSR. Each of the 12 units outputs audio to a different channel on the 14x4 mixer as well. The audio output from the mixer is then sent through the Quartet, PH-90, Scream 4, and RV7000, and then to Reason’s audio output 1-2. These effects were originally used as sends instead of inserts (meaning the instruments’ outputs would be mixed with various amounts of each effect’s output), but we wanted the effects to be able to interact with each other, so they had to be in sequence. The settings used on each effects module can be seen in the screenshot below.
Top of Reason rack:

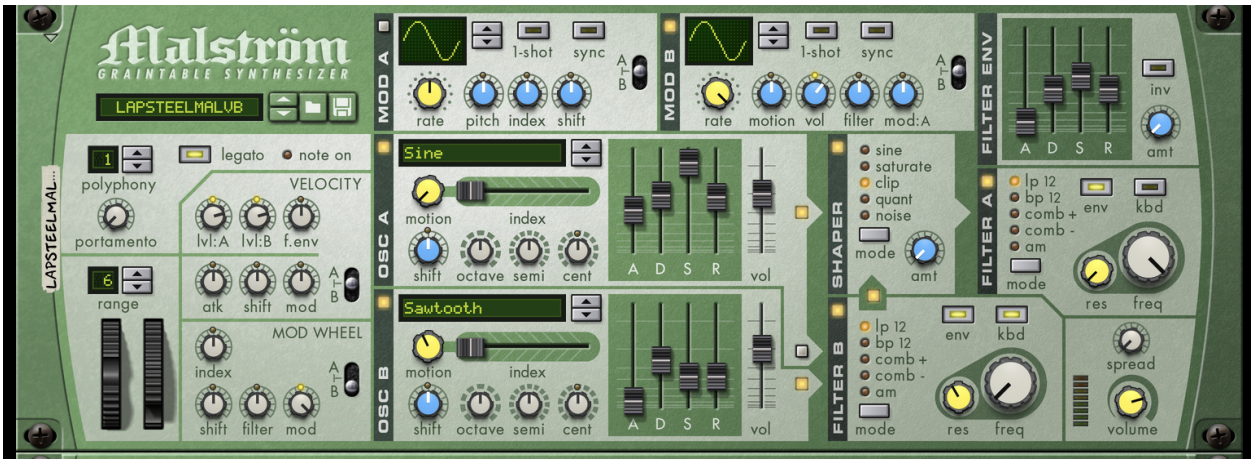
The synthesizers, however, warrant a closer look. Each channel has a polyphony of 1 and a legato trigger pattern, meaning that it can only play one note at a time (like a string), and once it's being played, the note won't "strike" again until the note is turned off. Instead, additional

note-on commands will change the pitch of the note without restarting the Amp and Filter envelopes.

In every synthesizer, the pitch bend wheel's range is 6 semitones. This is very important because it's what defines the softpot's range as one octave: 6 semitones down and 6 up from the central note. By changing the pitch bend range, the function of the softpot would change drastically. This iteration of the MIDI Slide is designed specifically for a range of one octave.

The velocity of note-ons, 127 in tapping mode and determined by the FSRs in plucking mode, impacts the loudness of the two oscillators in the Mälstrom. Oscillator A is a loud, consistent, sine wave, while B is a sawtooth wave, tuned an octave higher, that acts as the initial "pluck" sound of the string. Oscillator B is also sent through Filter B, which attenuates the higher frequencies so it blends more with Oscillator A. Oscillator B dies down quickly, while Oscillator A has a longer decay and higher sustain level. As mentioned earlier, both are also tuned up 6 semitones, so that the MIDI pitch value they receive from Max will be heard as the bottom end of the pitch bend range instead of the middle. Oscillator A moves through a passive Shaper that doesn't do anything to Filter A, which is controlled by the joystick's accelerometer. The filter envelope also doesn't do anything. Modulator B is how the tremolo effect is created. Its amount (via the mod wheel) and rate are both determined by the accelerometer.

Malström Graintable Synth. All four copies are identical:



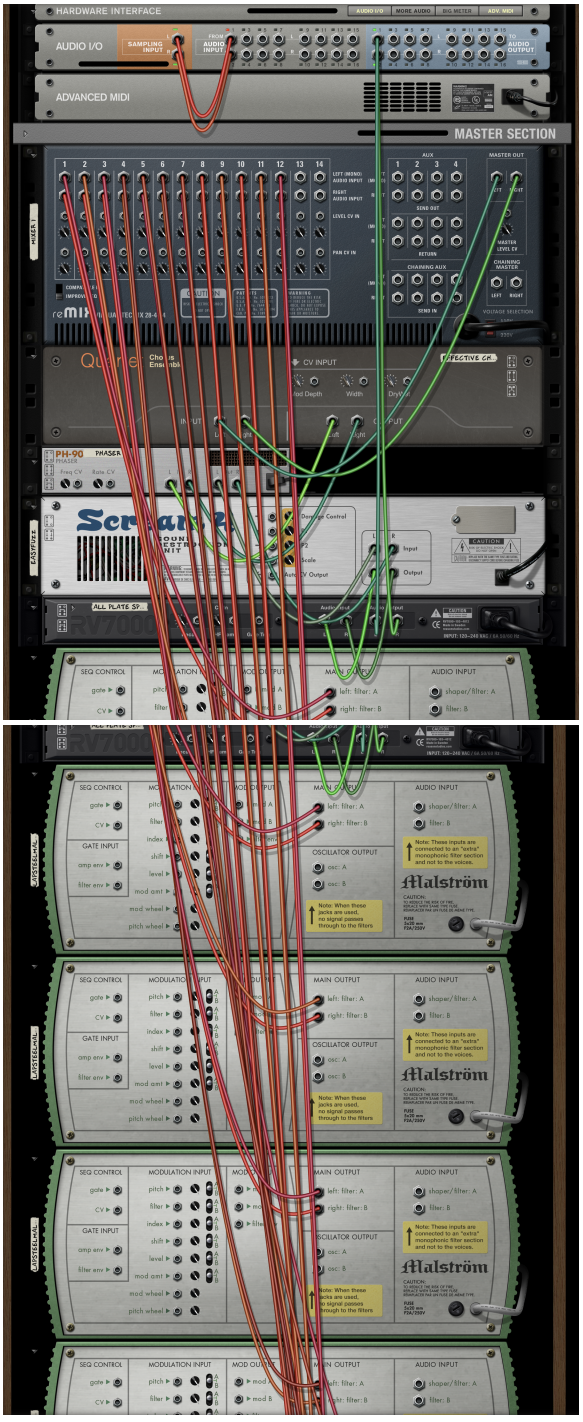
The NN-19 sampler is used with a pre-loaded "Cello" sound, with minor tweaks to Polyphony, Amplitude Envelope, and LFO settings.

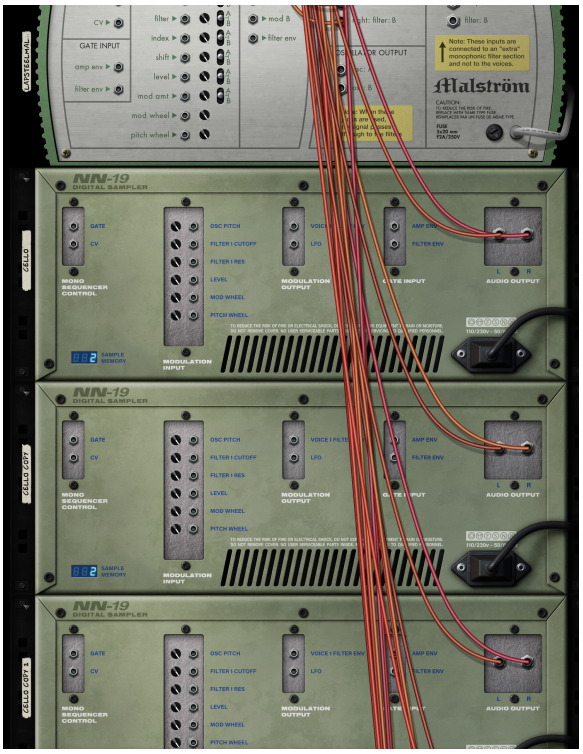
NN-19 Digital Sampler. All four copies are identical:

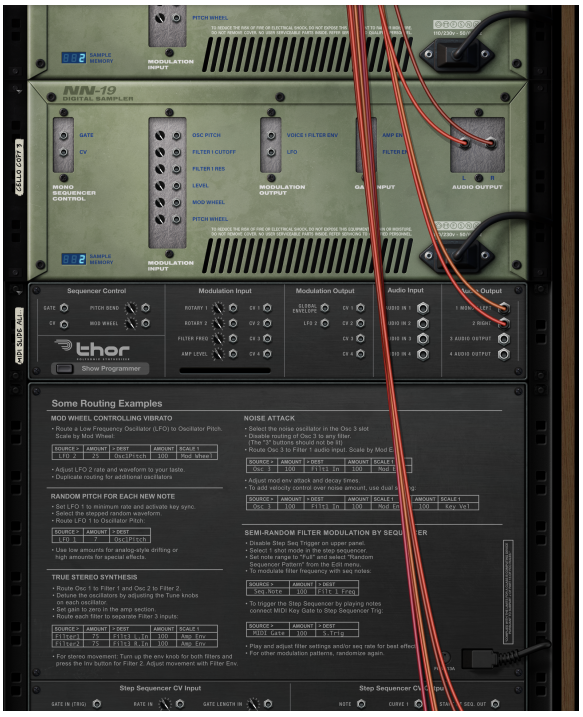
The Thor Polysonic Synthesizer is loaded with an adjusted version of the "Alan Turing's Dream" patch. Settings were changed similarly to the NN-19 to map the modulation wheel to the LFO, as well as the detune of the oscillators. The three oscillators, two with random detune and one with an organ sound, are fed through filters, envelopes, and LFOs, and finally a delay module, to create a unique spacey electronic sound.

Thor Polysonic Synthesizer. All four copies are identical:

Routing of Reason Rack:




CV
GATE INPUT
filter
index
shift
level
amp env
filter env
mod amt
mod wheel
pitch wheel
mod B
filter env
Note: These inputs are connected to an "extra" monophonic filter section and not to the voices.
Malström
NN-19
DIGITAL SAMPLER
GATE
CV
MONO SEQUENCER CONTROL
OSC PITCH
FILTER 1 CUTOFF
FILTER 1 RES
LEVEL
MOD WHEEL
PITCH WHEEL
MODULATION INPUT
VOICE I FILTER ENV
LFO
MODULATION OUTPUT
AMP ENV
FILTER ENV
AUDIO OUTPUT
SAMPLE MEMORY
CELLO
CELLO COPY
CELLO COPY 2


NN-19
DIGITAL SAMPLER
MODULATION INPUT
SAMPLE MEMORY
CELLO COPY 2
Sequencer Control
Modulation Input
Modulation Output
Audio Input
Audio Output
thor
Show Programmer
ADB SNDC ALL
Some Routing Examples
MOD WHEEL CONTROLLING VIBRATO
RANDOM PITCH FOR EACH NEW NOTE
TRUE STEREO SYNTHESIS
NOISE ATTACK
SEMI-RANDOM FILTER MODULATION BY SEQUENCER
Step Sequencer CV Input
Step Sequencer CV Output

Evaluations and Future Plans

As presented on December 17th, 2024, our instrument worked very well. I'm glad we were able to revisit and improve upon our instrument as our second project! I was pleasantly surprised at how low latency we were able to get our wireless system, even though we couldn't figure out the Bluetooth. If we had more time to work on this in the future, I think we should revisit the design of the pad to make it more ergonomic and have fewer sharp edges. It would also be great if the pad could connect wirelessly to the computer as well, or process data (i.e. use Max) internally and have a MIDI Out and USB Out instead of needing Max to run in a computer. That setup would better emulate commercially available MIDI instruments, so it would be easier for anyone with a passing knowledge to pick up and play.