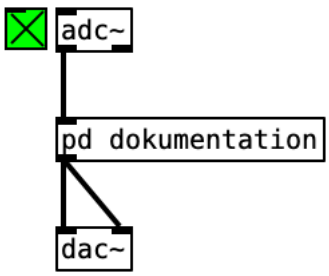
Felix Geith
Matrikelnummer 119827
2. Fachsemester
Medienkunst/Mediengestaltung BA

Richard-Wagner-Straße 15
99423 Weimar

Autonomous Collaborative

Auditory interaction on embedded systems

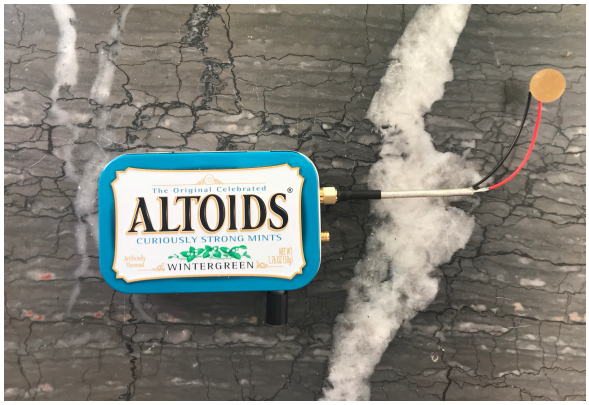
Documentation

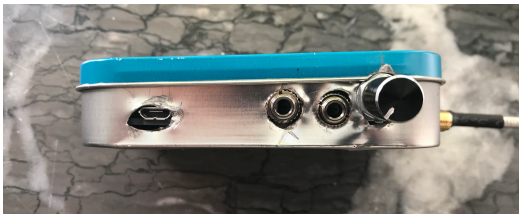


INDEX

INTRODUCTION

The main goal for the end of the semester was the construction of a modular synthesizer in pure data, controllable via midi-controlling and a piezo-microphone. The first step was to make it possible to connect the audio input of the piezo microphone to the pure data patch. Therefore the first thing was the construction of an audio interface.





Now being able to use the piezo mic, the synthesizer was made of the following modules:

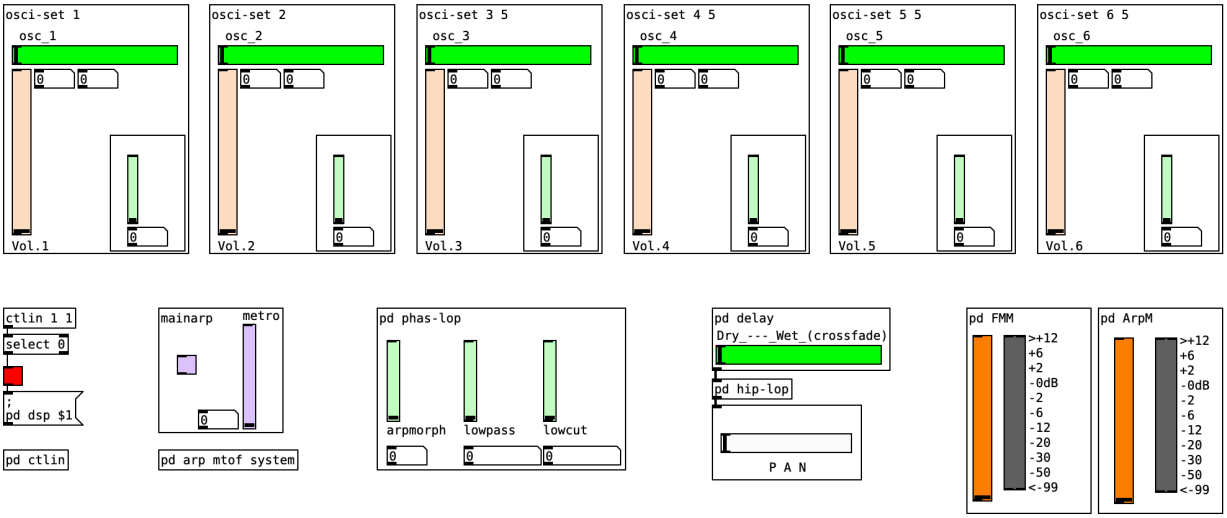
- A frequency modulation synthesis module (FM), consisting of six sinus and phasor oscillators.
- An arpeggiator, with six separately controllable steps, working with sinus and phasor oscillators.
- A spectral delay module for the arpeggiator, controllable via the midi input and the acoustical parameters of the piezo input.

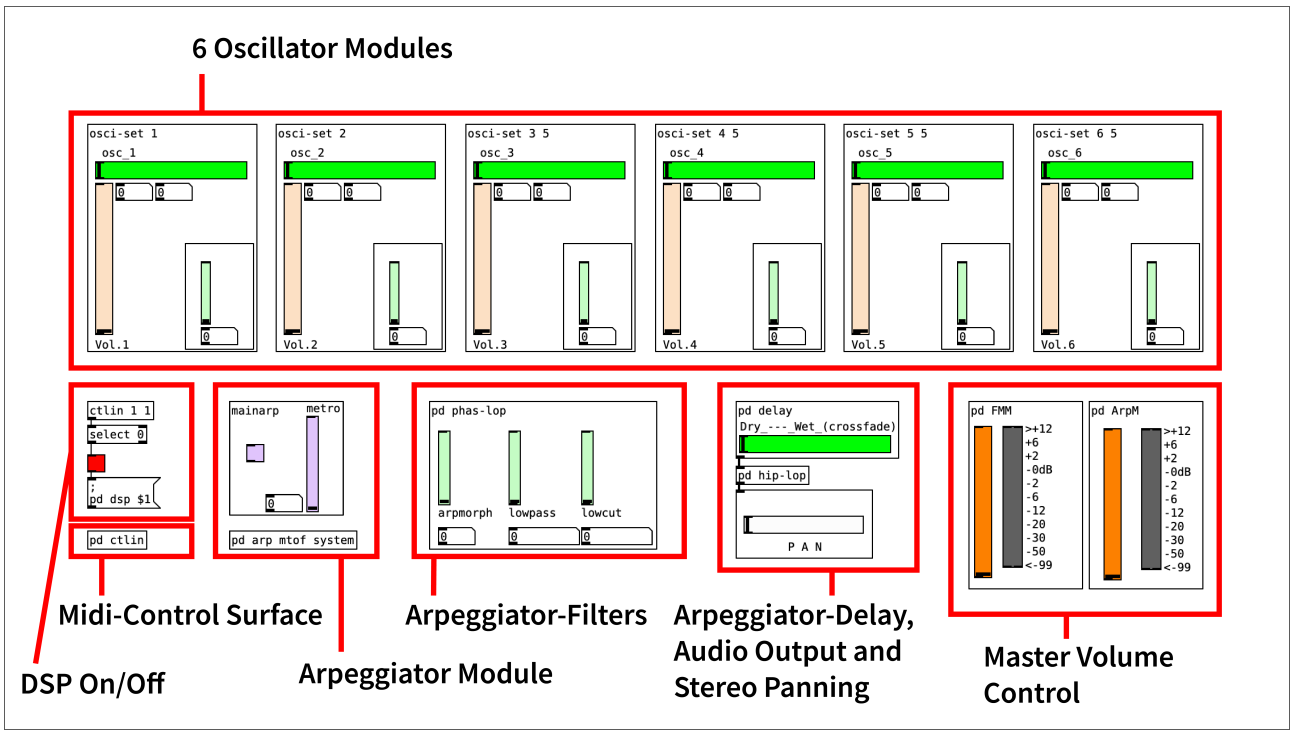
The midi input that is used comes from an Akai Midimix, shown below.

The Akai Midimix fits perfectly for the controlling of frequencies, audio filters and volume faders, because it offers 24 poti-buttons, 9 volume faders and multiple buttons.

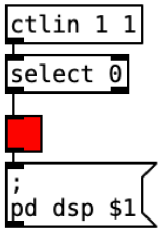
THE PATCH

The pure data patch is separated into multiple parts, fitting the multiple modules of the synthesizer.





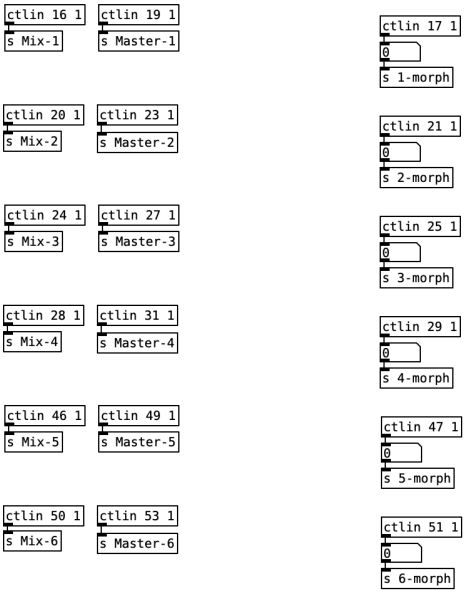
DSP On/Off



This part of the patch allows the user to switch the audio on and off via the midi controller. The midi input inject the number 127 every time the button is pressed, and a 0 every time the button is released. These two numbers are sent to the select object, that sends a bang to the toggle every time a 0 comes from the midi input, making it a simple switch.

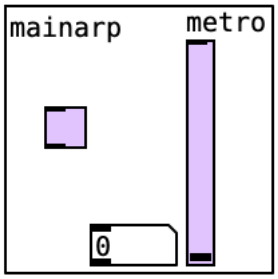
Midi-Control Surface

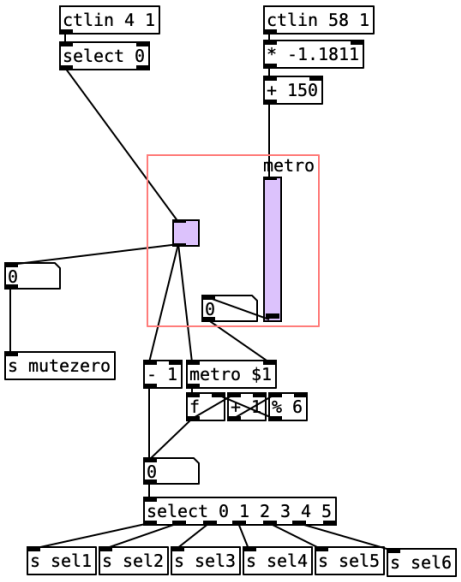
pd ctlin



The midi-control surface, which is hidden in a subpatch, is the location where most of the midi input channels (ctlin …) are routed to send objects. The receive objects fitting the send objects can be found patched to the switches and faders that are meant to be midi controlled.

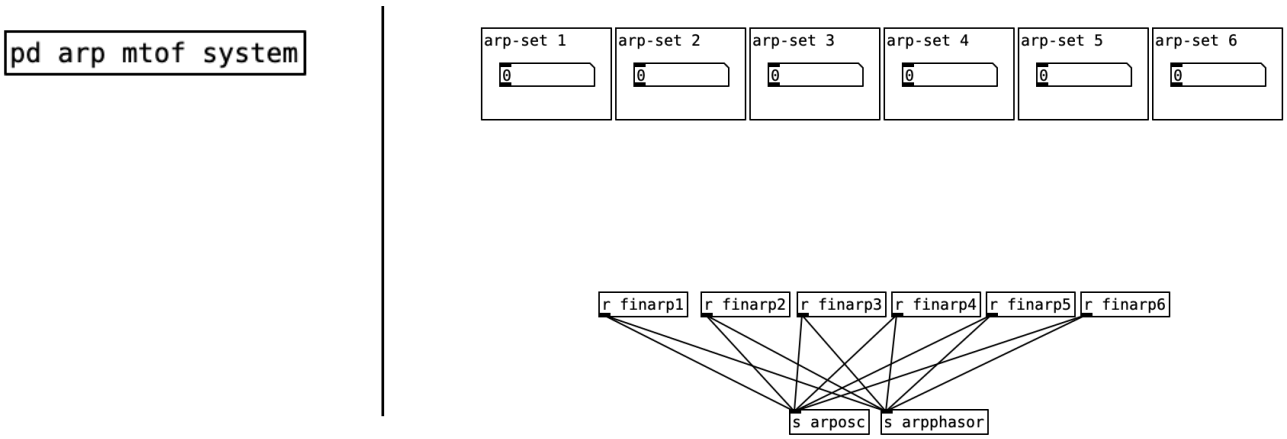
Arpeggiator Module



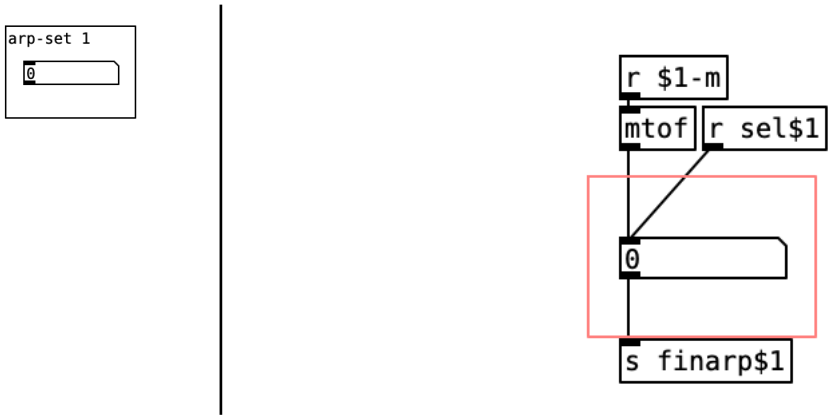


The Arpeggiator Module is one of the two sound creating parts of the patch. It basically is a select object triggering six send objects. The select module is given a metro, that can vary between 0 and 150. Also there is a toggle, that stops the select module and sets it back to -1 so it will start with „select 0“, the first variable of the select object. The toggle also sends the number 0 to the arpeggiator master volume, so it will be silenced when turned off. The toggle, which works as a mute button, can also be midi controlled.

The sends coming out of the select object give their value to receive objects, that can be found in the second part of the arpeggiator module, a subpatch called „arp mtof system“.



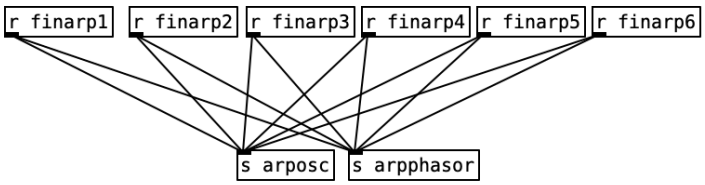
This second part of the arpeggiator module is also divided into two parts. The first part are six subpatches.



Here we can find the receivers of the select object of the arpeggiator, called „r sel$1“. The „$1“ is defined by the numbers 1 to 6, given to the six sub-patches. We can also find a receive object called „r $1-m“. This object recieves midi notes from the 6 Oscillator Modules shown above. (read about it on page 15) These midi notes are changed to a number resembling a Hertz frequency in the mtof object. (mtof = „midi to frequency“)

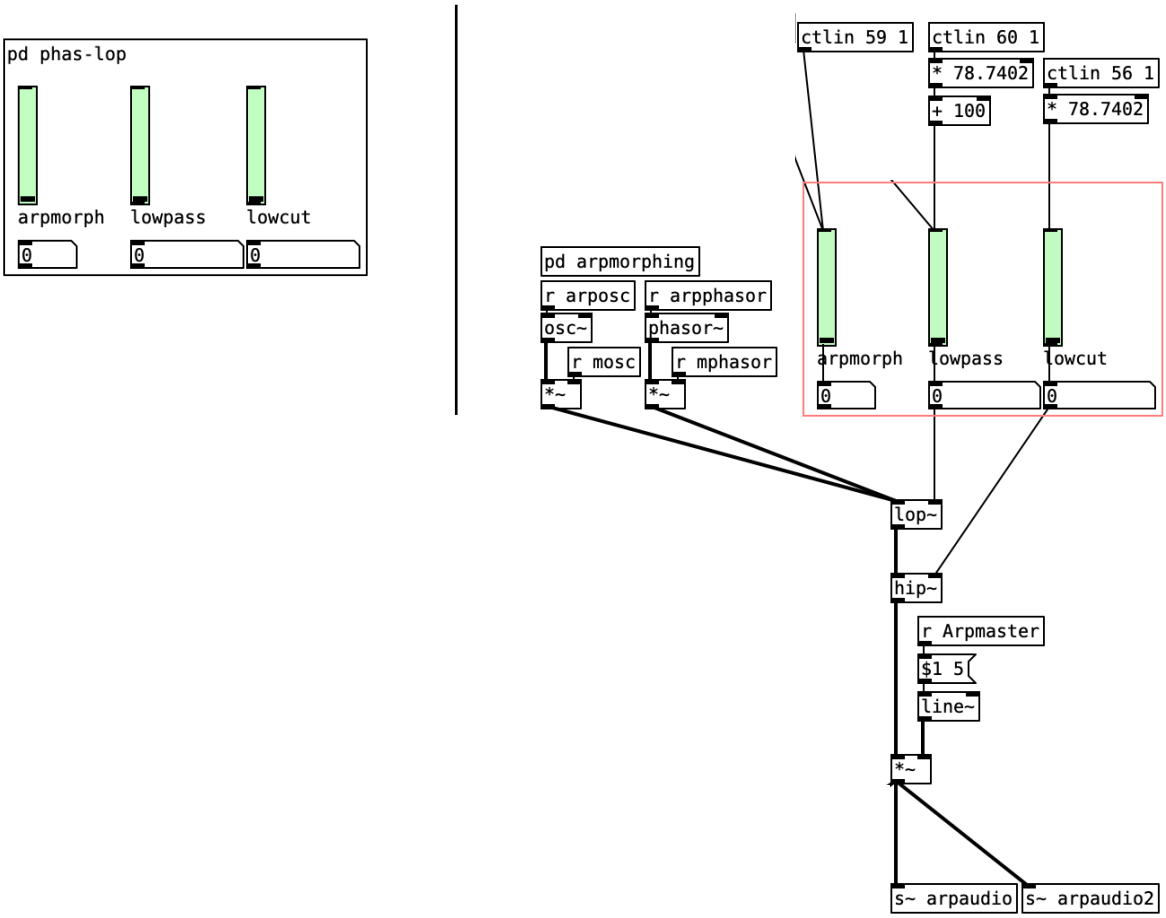
This number is given into a send object called „s finarp$1“. This means, the result of these six subpatches are six different Hertz frequency values, sent via the objects „s finarp1“ to „s finarp6“, which are triggered by the select module. (r sel1 - r sel6)

Under these six subpatches, there's another patch.



In this patch, the receivers for the Hertz frequency values can be found. (r finap1 - r finarp6) Every one of the receivers are linked to two send objects called „s arposc“ and „s arpphasor“. These send objects send the values into the Arpeggiator Filter Module.

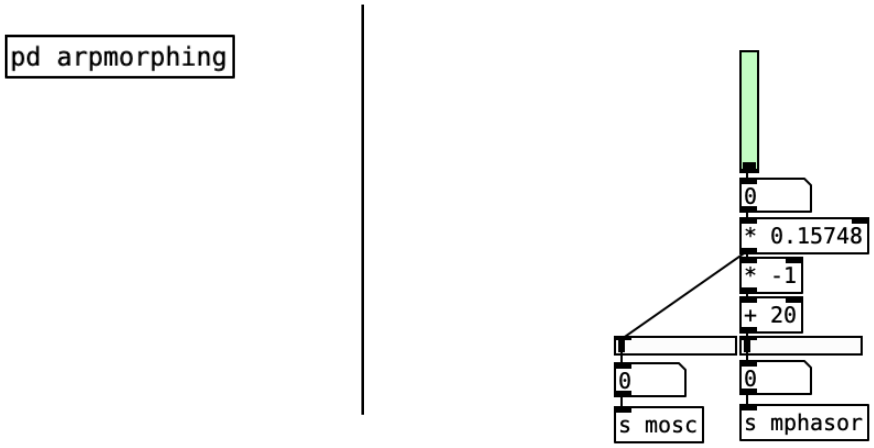
Arpeggiator-Filters



The Arpeggiator-Filter Module is the first part of the arpeggiator system, where audio comes into play. But the module does not only contain the oscillators of the arpeggiator, it also contains the most parameters and filters that are shaping the sound of the arpeggiator.

The two signal inputs „r arposc“ and „r arpphasor“, that receive the Hertz frequency values from the Arpeggiator Module, give these values to an osc~ object and a phasor~ object. This is the exact part, where all the values and midi information from the parts above become audio signals. These two audio signals, the sine wave and the saw tooth wave, can be regulated via a crossfader. The crossfader system is hidden inside the subpatch called

„pd arpmorphing“ and outputs its information the receive objects „r mosc“ and „r mphasor“.



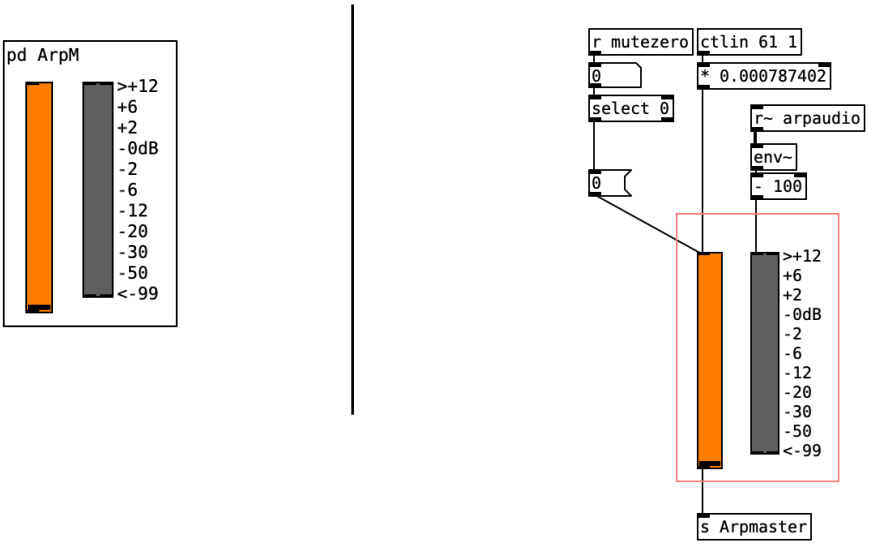
The crossfader (coloured green) is receives the midi control information from the Midi-Control Surface and outputs values from 0 to 127. Both volume faders (coloured white), that are connected to the oscillators via the send modules work with values from 0 to 20, so the first step is to send the values through the „* 0.15748“ object, that translates them to values from 0 to 20.

The next step is to get the two oscillator volume faders to move in the opposite direction. That means if one goes up, the other one has to go down parallel. This is archived by the multiplication with „-1“ and the addition of „20“ on only one side of the crossfader.

The values gained in the crossfader system then reach the send symbols.

(*Continuing in the Arpeggiator-Filters patch*)
After both audio signals, that come from the osc~ and the phasor~ object are regulated by the crossfader, the audio signals join each other in a lowpass filter object („lop~“). And after that, run though a lowcut filter („hip~“). Both filters get values from 0 to 127 by the midi control input, but work with frequency values from 0 to 10000, so the values need to be translated by the multiplication object „* 78.7402“. If the lowpass filter cuts at the value 0, all the sound is cut off and the whole arpeggiator is muted. To avoid that muting of all sound, the values coming from the midi controls for the lowpass filter get an addition of 100, making 100 the deepest possible cut frequency.

After these two EQ-filters, the audio signal runs though a master volume system, receiving values from the Master Volume Control.



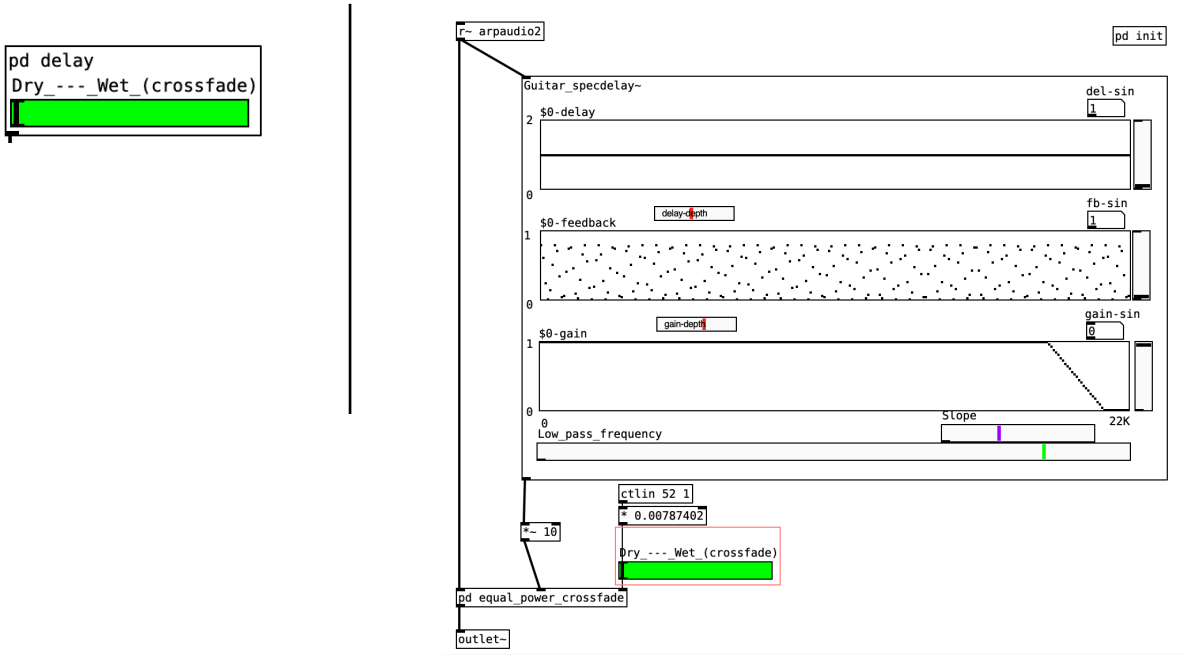
The master volume fader has multiple value inputs. One of the inputs is the receiver of the on/off toggle of the Arpeggiator Module. It sends the value 0 to the master volume fader, as already mentioned above. The other value input is the midi control input. This volume fader works with the values 0 to 0.1, so the this time the multiplication object „* 0.000787402“ is needed to translate the values given by the midi input.

The values of the masterfade are sent to the audio signal processing via „s Arpmaster“. After the master control, the audio signal ouputs in two send objects. One („s~ arpaudio“) sends the signal to a db-meter, that can be found besides the master volume fade. An accurate visualisation of the audio information on the db-meter is achieved with an „env~“ object and the subtraction of 100.

The other send object („s~ arpaudio2“) sends the audio signal to the Arpeggiator-Delay Module.

Arpeggiator-Delay Module

(spectral delay)

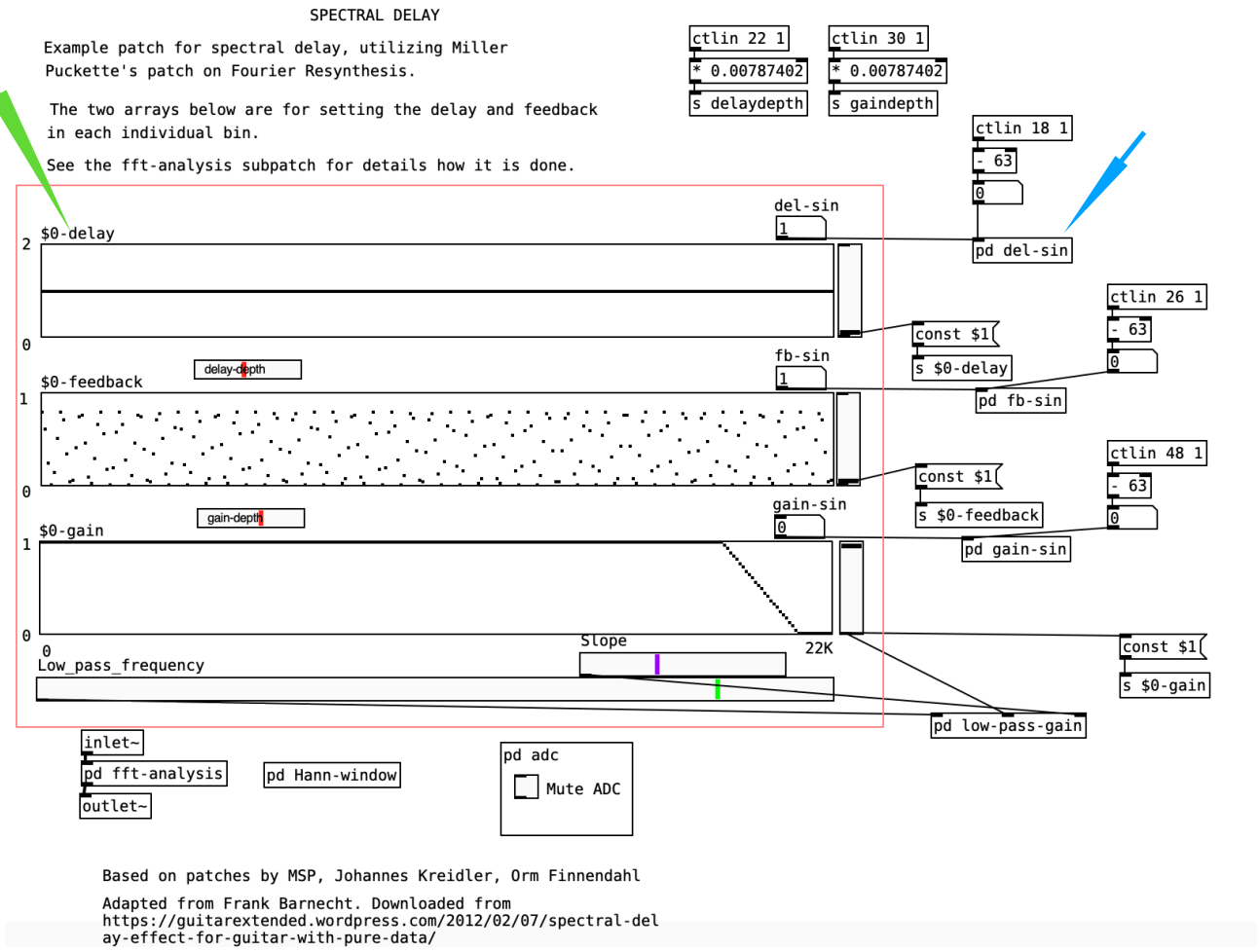


The delay module is the only part of the patch that's not self-made. It is based on a patch that can be found on the internet[1] and was originally meant to be a spectral delay effect for and electric guitar. It has multiple parameters, and the following parameters can be controlled:

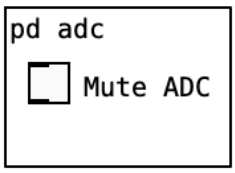
- Delay wave length
- Delay depth
- Feedback wave length
- Gain depth
- Gain
- Dry - Wet

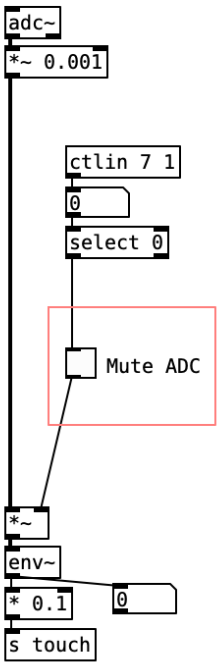
All the controllable parameters have different values, so some value multiplication are needed in the delay subpatch to translate the midi values.

[1] https://guitarextended.wordpress.com/2012/02/07/spectral-delay-effect-for-guitar-with-pure-data/

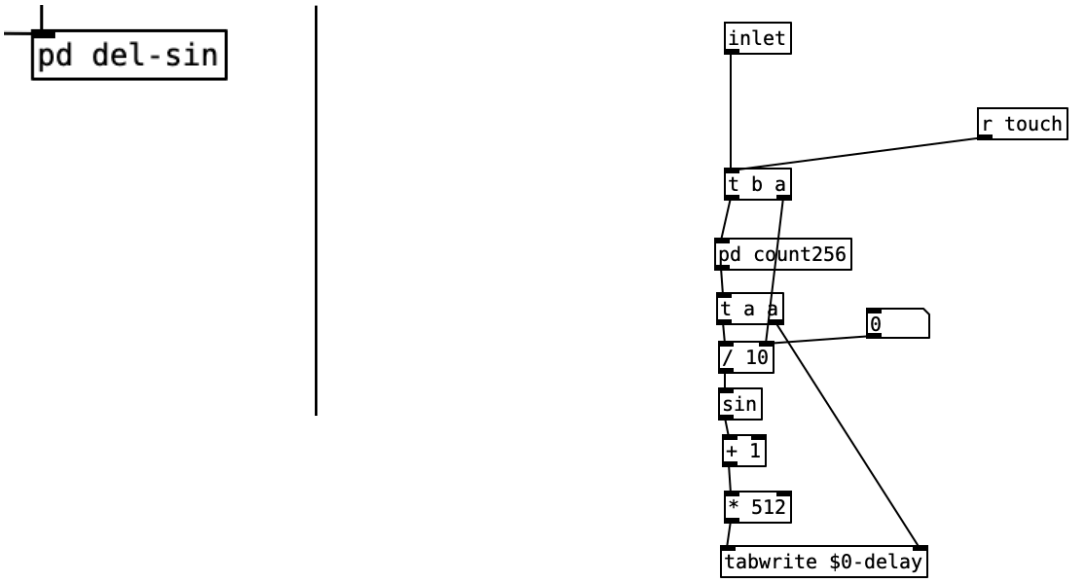


In this subpatch another subpatch called „pd adc“ can be found, which contains the audio input (adc~) of the piezo mic.





The audio input („adc~“) receives the audio signal from the piezo microphone. It is amplified with the value 0.001 and can be muted with a midi controlled toggle, working the same way as the on/off toggle of the arpeggiator. The audio signal is translated into a number signal via an „env~“ object and sent („s touch“) to the subpatch „pd del-sin“ that can be found in the delay subpatch. (marked with a blue arrow)



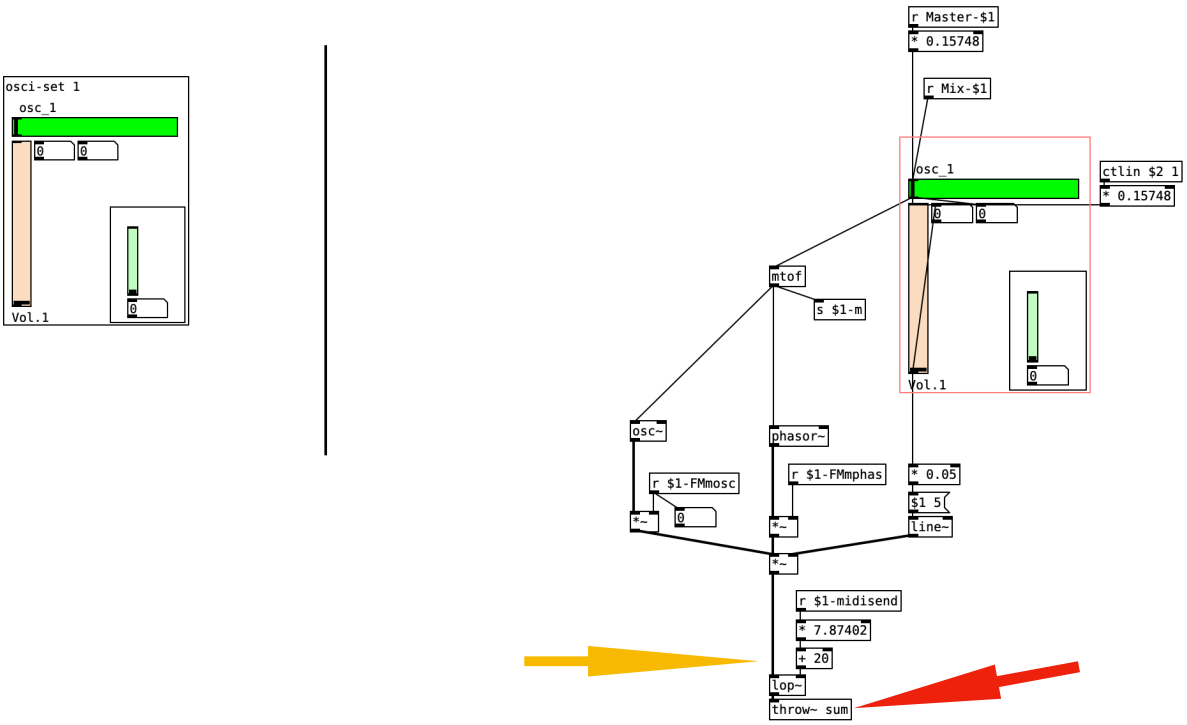
Here the number signal of the piezo input is received by „r touch“ and intervenes in the „t b a“ object, which is a part of the system, that writes the sine frequency in the spectral delay array. (marked with a green arrow in the delay subpatch)

That means, if the piezo input is activated, the sine wave, that normally is responsible for the sound of the spectral delay gets replaced by the wave coming from the piezo microphone. Like this it is possible to change the sound of the delay by haptic audio interaction, as long as the piezo input keeps getting a signal.

After the Delay Module, the audio signal of the arpeggiator is proceeded to the subpatch „pd hip-lop“, where it joins with the audio signal of the six oscillator modules.

6 Oscillator Modules

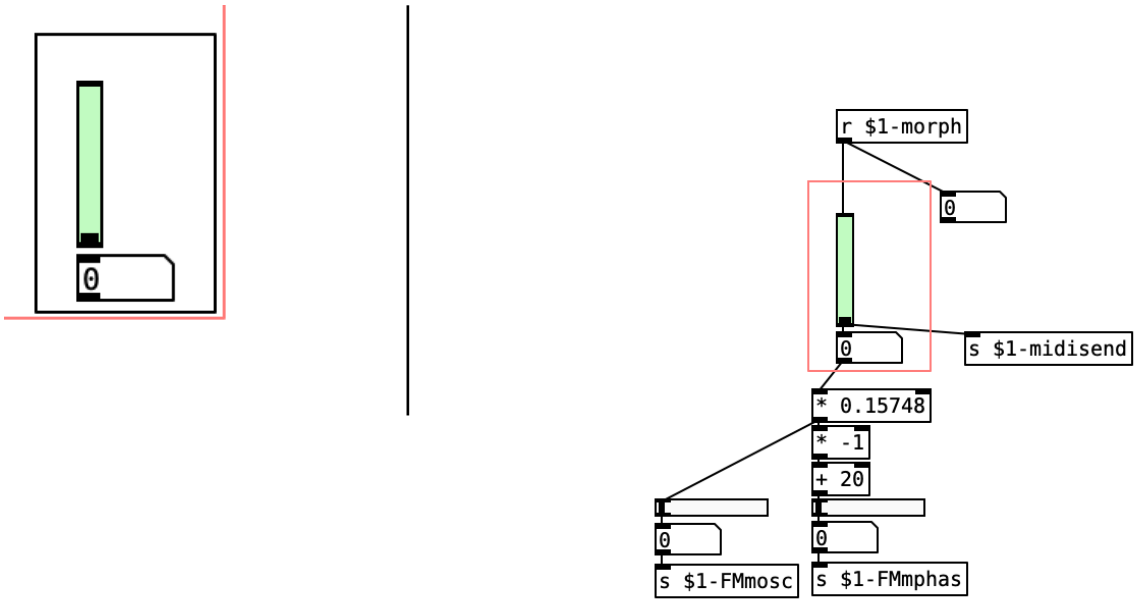
(FM synthesis)



The 6 oscillator modules, that can be used as for frequency modulation synthesis, are basically 6 oscillators with 3 different parameters that can be controlled: Hertz frequency of the oscillating wave, volume/amplitude of the wave, and a crossfade between sine wave and sawtooth wave.

The Hertz frequency can move between ~ 8 and ~ 10000 Hertz and receives it's midi values via the object „r Mix-$1“, that receives the midi control from the Midi-Control Surface. These midi values are translated to Hertz values by am „mtof“ object and reach into both an osc~ and a phasor~ object. The Hertz frequency values are sent to the six different arpeggiator steps via the send object „s $1-m“.

The crossfade system that regulates the relation between the sine wave and the sawtooth wave works the same as the crossfade system that can be found among the Arpeggiator Filters. The crossfader is found in a subpatch in the oscillator module and sends its values to the receivers „r $1-FMmosc“ and „r $1-FMmphas“.

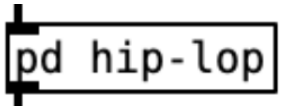


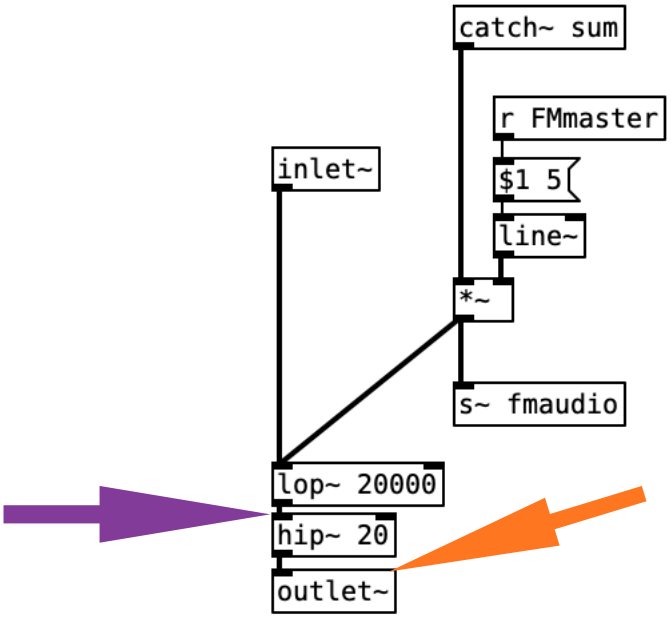
The midi control comes from the Midi-Control Surface and is received through „r $1-morph“. These midi notes are translated to the correct crossfader values, and they are also transmitted to a lowpass filter via „s $1-midisend“.

The lowpass filter can be found in the oscillator subpatch (marked with a yellow arrow) and shapes the sound of the audio signal dependent of the crossfader values. The lowpass filter works with values between 0 and 1000, so the midi values are translated with „* 7.87402“. These values also get an addition of 20, to prevent the lowpass filter from cutting off all the sound, when the values come to the minimum.

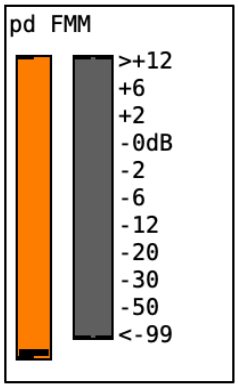
After the lowpass filter system, the audio signals get transmitted to the subpatch „pd hip-lop“ via a throw~ object called „throw~ sum“ (marked with red arrow).

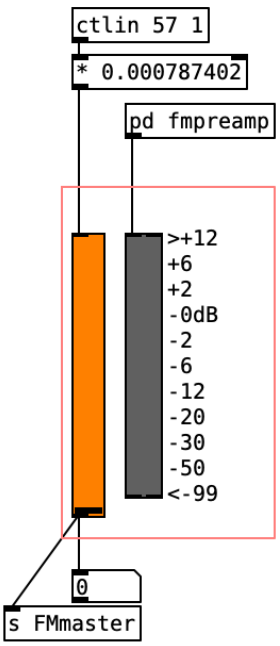
Subpatch „pd hip-lop“





This subpatch is the exact place, where the audio signal of the arpeggiator and the 6 Oscillator Modules join together. The audio of the oscillator modules is volume controlled by „r FMmaster“, which receives values from the Master Volume Control.





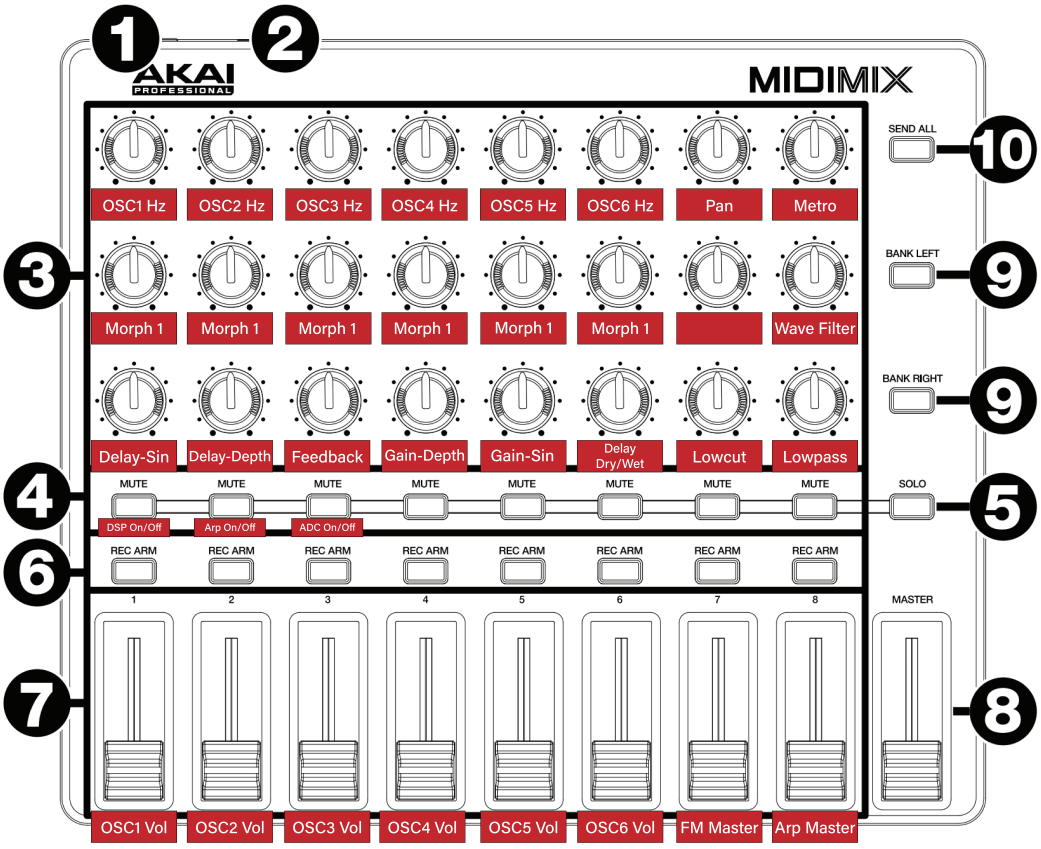
It works exactly like the master fader of the arpeggiator, but receives its own midi control values, so the volume of the arpeggiator and the oscillator modules do not depend from one another.

After the oscillator modules audio signal is regulated by its master volume control, both audio signals join together and are proceeded through a lowpass and a highpass filter, cutting all frequencies above 20000 Hertz and under 20 Hertz. (marked with purple arrow)

After these two filters, the audio signal is proceeded in the audio output, the dac~ object. (marked with orange arrow)

HARDWARE

The midi controller that controls all the parameters which receive midi values is an Akai Midimix.



ADDITIONS

Where can we go from here? Which features can be added to improve the users experience?

One idea would be placing the patch on a raspberry pi to make it a „stand alone“ modular synth, so it can be used without connection to other hard- or software

Another feature would be presets, some to be already premade, some to be configurable by the user.