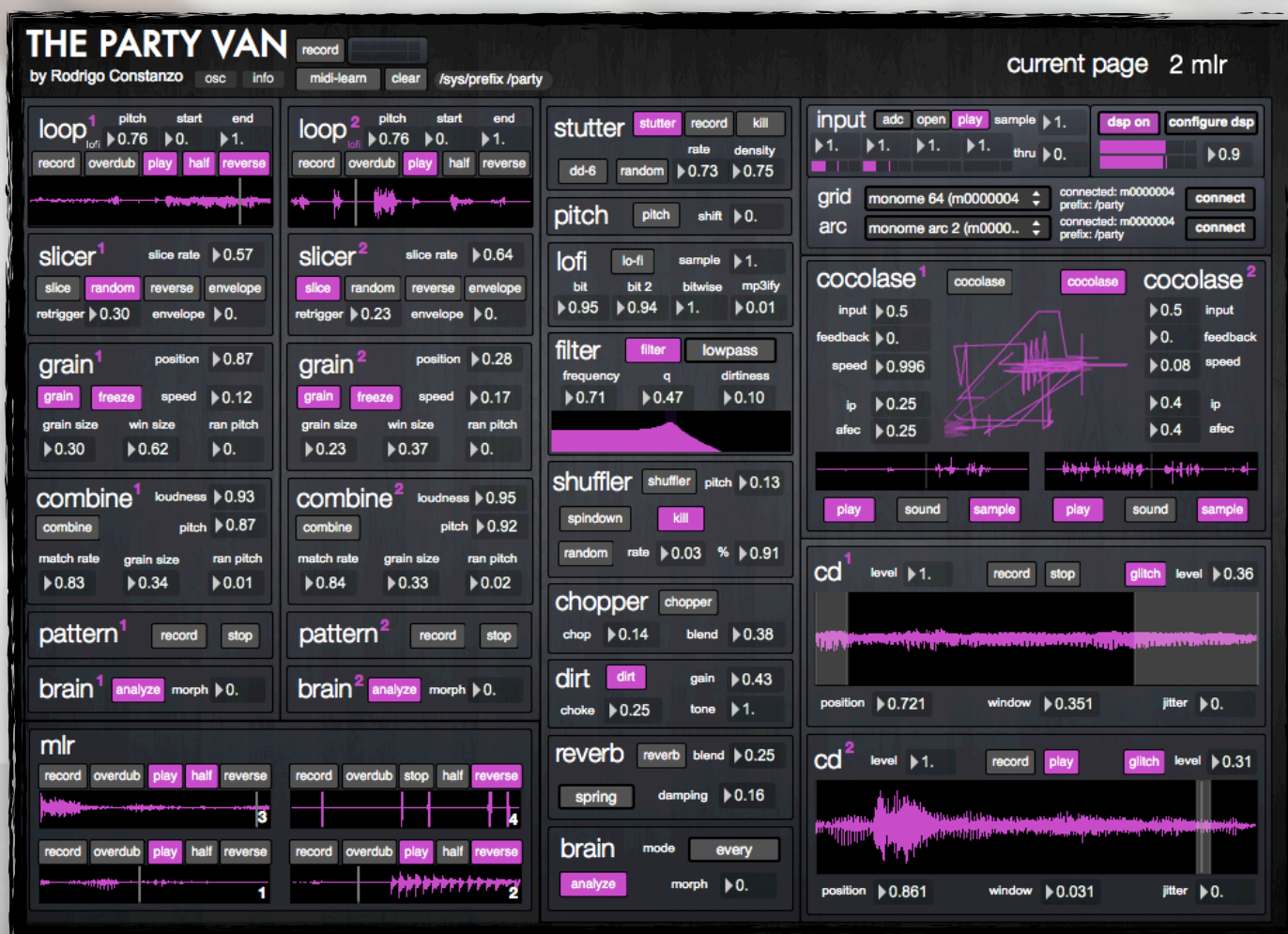


# THE PARTY VAN

by Rodrigo Constanzo

<http://www.rodrigoconstanzo.com>

V06



# INSTRUCTION MANUAL

# DESCRIPTION AND OVERVIEW

The Party Van is a live sampling and performance instrument built using Max/MSP. It was programmed around the [Monome 64](#) and [Arc 2](#) controllers but includes mapping for an iPad running [TouchOSC](#) (custom layout included) as well as the [Keith McMillen Softstep](#) foot controller. Additionally, the patch has MIDI learn functionality built in, so every parameter can be mapped to any MIDI controller\*.

\*Due to the way I programmed the GUI buttons, things that are mapped to the Monome/TouchOSC don't reflect changes made when using MIDI learn. This only really impacts things if you are planning on using a MIDI controller AND a Monome/iPad to controller the same GUI buttons. If controlling different things there are no problems.

## THE PARTY VAN INCLUDES SOME OF THE FOLLOWING:

- A variety of samples/loopers oriented towards live performance
- Granular, concatenative, and convolution based synthesis on recorded buffers
- Buffer-based and real-time audio analysis used to dynamically generate intelligent presets
- Input/output effects
- Input stage convolution and amp simulation
- 8-bit sampler/looper based on the [ciat-lonbarde Cocolase](#) instrument
- Virtual CD skipping module based on "[The Chocolate Grinder](#)", another one of my patches
- Attack-based sampling and triggering of effects
- Several soundfont and synthesis-based playback instruments

# SYSTEM REQUIREMENTS

Requires [Max6](#) and the externals listed below.

Built for Monome 64\*, Arc 2, iPad, and Softstep.

You can use any of those controllers but if you use the Monome controllers you need [serialosc](#).

\*The patch is built around the 2011 edition varibrightness Monomes but varibrightness isn't used extensively and I'm doing my best to keep it backwards compatible, but there will likely still be a couple of issues as I have nothing to test compatibility with.

## EXTERNALS USED

Alex Harker externals

<http://www.alexanderjharker.co.uk/>

Fluidsynth~ to playback soundfonts (only used on Page7)

[http://imtr.ircam.fr/imtr/FluidSynth for Max/MSP](http://imtr.ircam.fr/imtr/FluidSynth_for_Max/MSP)

FFTease library (specifically codepend~)

<http://www.somasa.qub.ac.uk/~elyon/LyonSoftware/MaxMSP/FFTease/>

OSC-Route from the CNMAT externals

<http://cnmat.berkeley.edu/patch/4029>

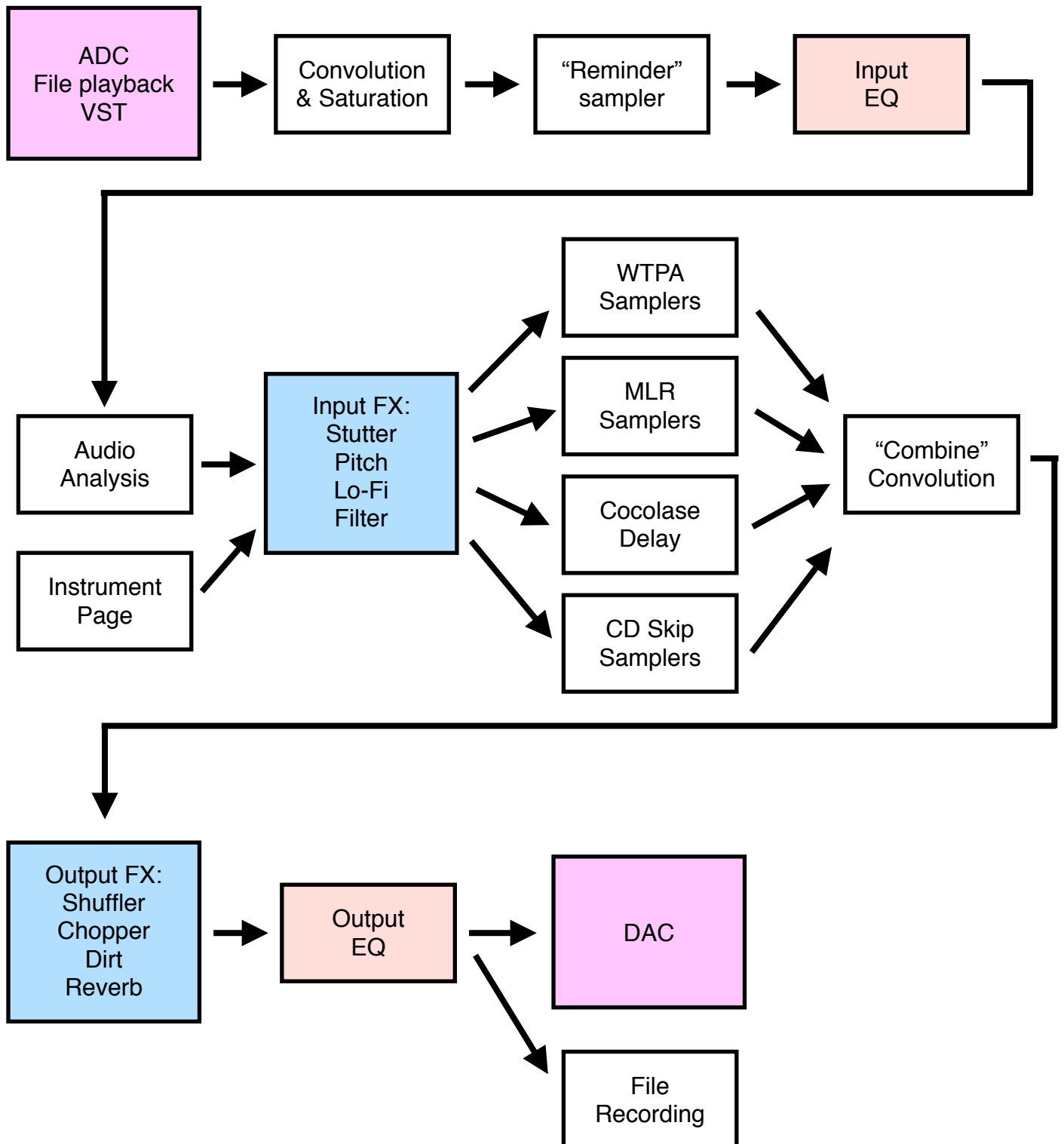
Sigmund~ for pitch tracking in "Pitch" module

[http://www.maxobjects.com/?v=objects&id\\_objet=4713](http://www.maxobjects.com/?v=objects&id_objet=4713)

## NEW IN THIS VERSION (V06)

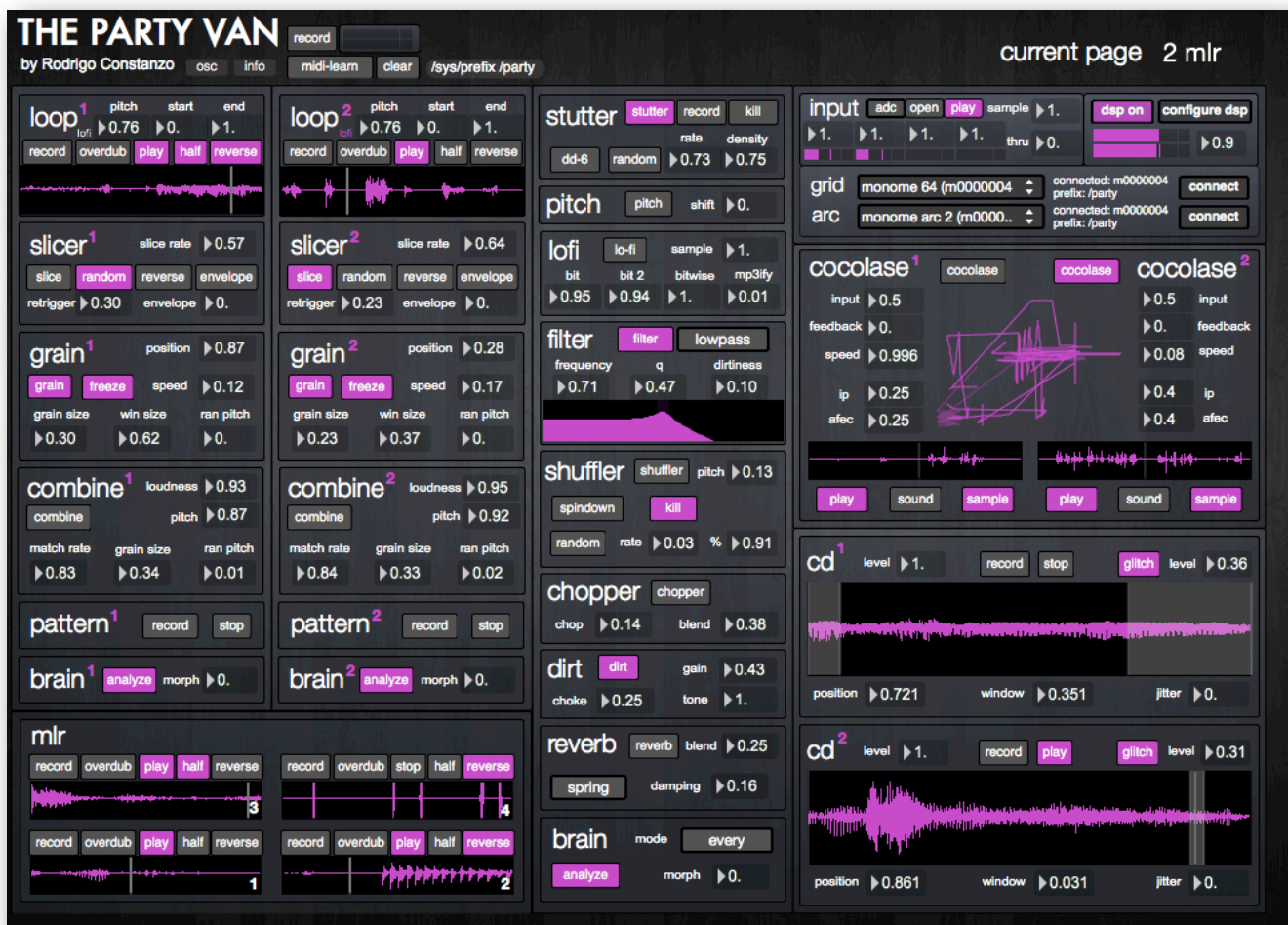
- new "Pitch" algorithm that sounds more like digitech whammy pedal (new requires sigmund~)
- made 'vst' from the input menu universally useful
- massive overhaul to how button presses work (in order to work better with MIDI/gui)
- "Filter" dirtiness turned down a bit
- fixed small click when turning off "Grain"
- made varibrightness stuff be 16 steps (for upcoming firmware update)
- adjusted the timing of the 'spindown' mode of "Stutter"
- fixed bug where resetting one "CD" module reset both
- fixed bug that prevented in/out EQ from working

# SIGNAL FLOWCHART



# MODULE & GUI BREAKDOWN

The GUI was built to be very minimal and utilitarian. It is not intended to be used while looking at the screen, and as such, you don't need to interface with things on screen while performing.



# INPUT/OUTPUT/MONOME SECTION



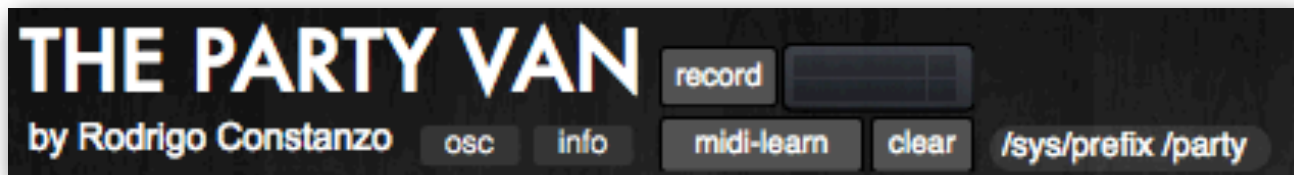
This section has three main parts. The top left is your input section. The number boxes, and corresponding meters, are the first four inputs of your soundcard. If you only have two inputs, only the first two will appear active (as in the screenshot).

The dropdown menu (currently showing “adc”) is where you select whether you want to use live input, or file playback (or test tone/vst). Open/play are for the file playback, although when in vst mode, open opens the current vst instead. “Sample” controls all sampler levels (including wtpa/mlr/cocolase/cd/etc..) and “thru” is the audio through (taken after the input FX).

The top right turns on the DSP and lets you configure your soundcard/inputs/outputs. The number box controls master output level.

The lower section are the serialosc hosts for your Monome devices. Select your device(s) from the dropdown menu and press ‘connect’. If you are not using any Monome devices you can ignore this section.

# INFO/MIDI-LEARN/OUTPUT RECORD



Here you have pop-up windows for general patch info and TouchOSC settings.

The record button lets you record the patch output. When you press it you are presented with a file dialog window. Name your file and hit 'OK' and you will see the meters going, indicating that you are recording audio.

For the MIDI-learn, first press the MIDI-learn button, then hover over the item you would like to control, and adjust the appropriate knob/button/slider on your MIDI controller. "Clear" clears your mappings. MIDI mappings persist, unless cleared or manually deleted.

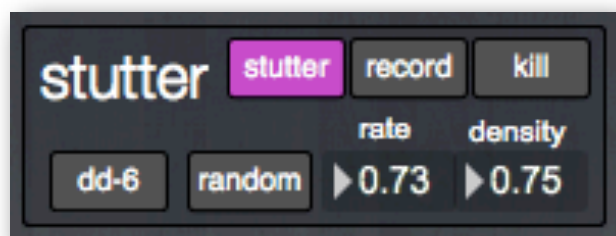
The "/sys/prefix/party" button is only for Monome usage. This sets your OSC prefix.

# INPUT FX



Here are the four input effects. Audio routing goes top down with each effect being in series with the previous.

## STUTTER



“Stutter” is a stutter/glitch module with two modes. The ‘dd-6’ mode, which does time domain stuttering and is modeled after the [Boss DD-6](#) guitar pedal in ‘hold’ mode. The other one is an fft mode that does spectral freezing. Both modes are always running so you can switch between them even after creating a stutter/glitch. To change modes press the button in the bottom left.

(Stutter cont..)

To activate the Stutter module you must first record a loop. The 'record' button records audio for the duration that it is depressed, so it is best to map it to a momentary button. Then the 'stutter' button turns on the audio.

On the Monome everything is mapped to one button so pressing and releasing the button records and plays back a glitch.

"Kill" mode mutes incoming audio when there is a glitch playing. Finally 'random' turns on/off the record function at a rate and probability set by the rate and density controls.

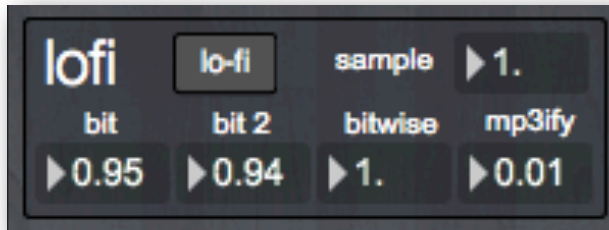
## PITCH



The "Pitch" module uses a delay-based pitch shifting technique to produce somewhat natural, somewhat glitchy sounding pitch shifting. I am aiming for a 'whammy pedal' type glitchiness, but it is difficult to emulate.

As with everything else in this patch, all parameter controls are mapped to 0. to 1., for ease of mapping. When the shift is set to 0. it is down 1 octave, when it is set to 1. it is up one octave.

# LOFI



The “Lofi” module is a combination of signal degradation modules. ‘sample’ controls traditional sample rate reduction, and ‘bit’/‘bit2’ control bit rate reduction but using different techniques that allow for fractional bit rate reduction. Both techniques sound slightly different, but combined allow for a wide range of sounds. ‘bitwise’ does bit flip/replacement processing and sounds very grungy. Even though this is controlled by a single float value, it is eight discreet steps.

The new ‘mp3ify’ control does some fft processing on the signal to allow you to dynamically control the psycho-acoustic compression very similarly to how mp3 compression works. When combined with the other parameter, this adds a very distinctive and unique twist on the lofi sound world. This parameter is also ‘inverted’ as other ones you lower, when ‘reducing’ the fidelity, this one you turn up, as in turning up the mp3 compression.

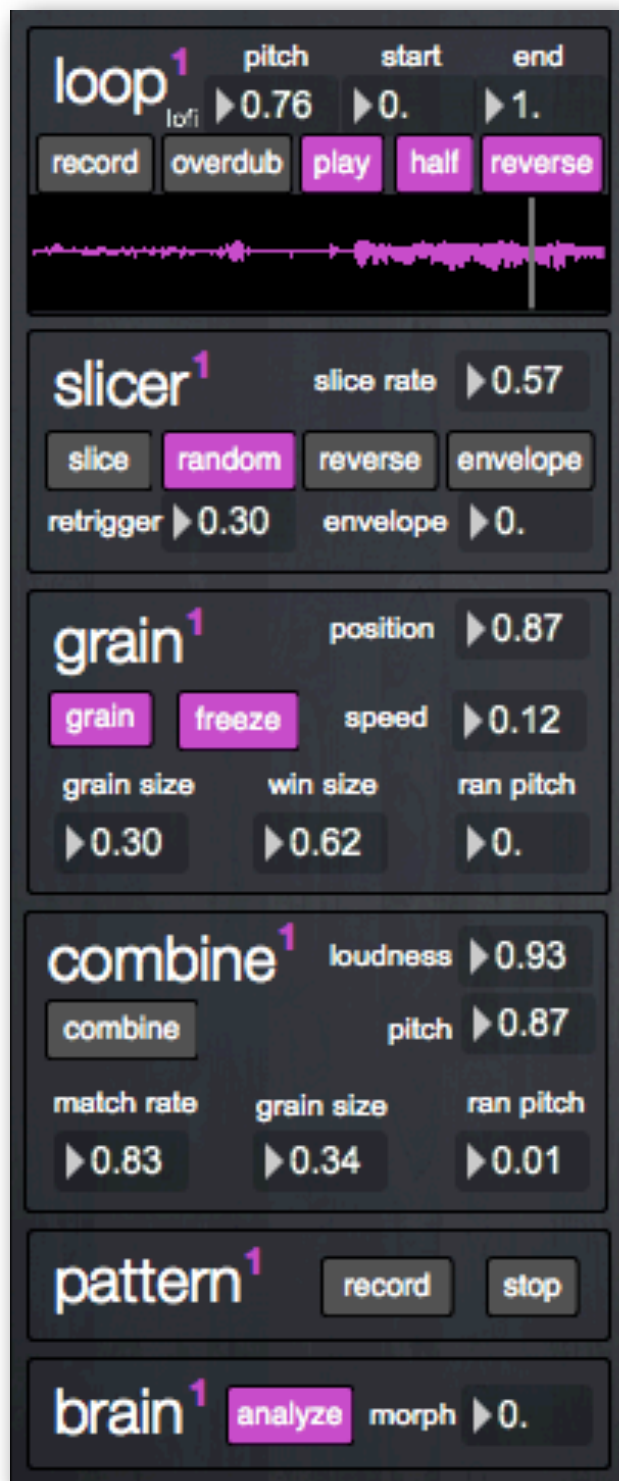
# FILTER



The “Filter” module gives you a couple of different filter modes (lowpass, highpass, & resonant) that are modeled on the moog-style ladder filter. Frequency and Q work as expected on a traditional filter, although the display is only a rough estimation of the filter shape.

The ‘dirtiness’ adds some noise to the filter coefficients resulting in a dirty, bubbly sound. Use this sparingly as it is possible to ‘blow up’ the filter if pushed too hard, and my patch has no means of resetting the filter other than a reopening the patch. The inspiration for this addition was Pierre Alexandre Tremblay’s butt~ filter.

# THE WTPA MODULES



There are two identical 'WTPA' modules which are looper/samplers with different buffer playback/processing options.

# LOOP



One of the most time consuming modules to build was the “Loop” module. Partially because I was still coming to grips with using Max/MSP with live sampling, but mainly due to the functionality of it, and it’s integration with other ‘WTPA’ modules.

It is a dynamically sized (as in you start end the loop whenever you want) varispeed looper at it’s core. It is modeled on The [“Where’s The Party At”](#) 8-bit sampler/looper, which is somewhat modeled on the [Line 6 DL-4](#) looper. Because of how it is coded, recording at speeds below 1. adds a sample rate reduction sound because it is actually recording at a lower sample rate. The pitch control works almost as a ‘clock’ control in that manner. Any changes in pitch made while recording are heard when playing back. While recording you can also engage halvespeed and reverse, including reversing ‘past zero’ which just makes the loop jump to the highest point you’ve written in the buffer up to that point.

The ‘start’/‘end’ controls control the start/end points of the loop, and these are controlled dynamically with the monome in an MLR-type fashion.

Lastly there is a ‘lofi’ mode that can be engaged by pressing the word ‘lofi’. This puts the looper into a more 8-bit sounding mode with added grunge/saturation all tied into the pitch control knob.

# SLICER



Like all the 'WTPA' modules, the "Slicer" module is linked in with other modules. You can start creating a loop with the 'slicer' engaged and it will jump around the buffer writing tiny slices into it, creating a pointillistic/glitchy loop.

The basic functionality of the "Slicer" is to jump around the buffer (either written, or while recording) at regular or random intervals (controlled by the 'random' button). 'Rate' controls the overall speed of this process. The 'reverse' toggle allows for slices to be played back in reverse as well as forward (this is independent of the global 'reverse' controlled in the "Loop" module). The 'envelope' control adds a linear fade out to each slice playback which adds an almost percussive sound to each slice. The 'envelope' parameter controls how long the duration of that envelope is. Finally 'retrigger' controls how likely the slices is going to play back the exact slice again. This can add a 'retriggery' kind of sound to the slicer module.

There is no amplitude enveloping when jumping to new slice points so the "Slicer" has lots of clicks/pops when jumping. This is intentionally, although with very bassy sounds, this can be a bit overwhelming.

# GRAIN



Like it's name would imply, “Grain” is a granular synthesis module. It is based on [sugarSynth](#) by Nobuyasa Sakonda.

The ‘position’ parameter controls the position of the grain playback engine in the sample buffer. This is tied to the playhead of the “Loop” module in such a way that you can turn on the “Grain” module at any point and it will start granulizing exactly where the playhead was.

‘Freeze’ & ‘speed’ control how “Grain” moves through a buffer. If ‘freeze’ mode is engaged “Grain” will just sit put and spit out grains wherever it is. If ‘freeze’ is turned off the granular synth will move through the buffer as controlled by the ‘speed’ control. A ‘speed’ setting of 1. would be 100% playback speed.

‘Grain size’, ‘window size’, and ‘random pitch’ control the granular synth engine and do exactly what they seem like they would do. ‘Grain size’ controls the size of each grain of audio. ‘Window size’ controls the size of the buffer window that grains are played back from and ‘random pitch’ randomizes the playback pitch of each grain.

“Grain” and “Slice” can be used in combination to jump around the buffer randomly while doing granular synthesis on that portion of the buffer.

# COMBINE



The “Combine” module is similar to the “Grain” module in that it is based on granular synthesis, but it is driven by an audio-analysis engine to handle grain replacement (concatenative synthesis). The way that this works is that as you are recording a loop in the “Loop” module, the “Combine” module is analyzing each grain of audio (20ms slices) and storing the position, loudness, and pitch of each grain in a database. When the “Combine” module is engaged it then analyzes incoming real-time audio and matches/ replaces each grain producing a very ‘granular synthesis’ sound, but controlled very differently.

There are two modes to the “Combine” module and they are selected contextually. If you engage the “Combine” module while the loop module is NOT playing, then you have concatenative synthesis as described. If you engage it while playing back audio, then it does fft convolution on your incoming audio and buffer playback. There are no controls over the convolution, and none of the displayed controls effect convolution. The convolution is effected by all other playback modes in the sense that the audio is being convolved with whatever the playback audio is (be it sliced, low pitch, granulized etc...).

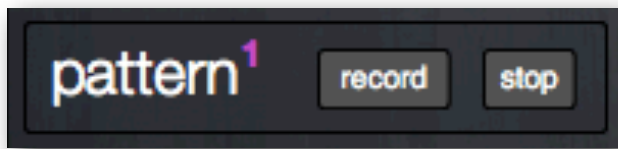
(Combine cont..)

The parameters 'loudness' and 'pitch' control the weight that is given to each in terms of playback analysis. If you have a high 'pitch' setting, it will only match grains that match in pitch very closely, same goes for loudness. If you record an audio loop into the "Loop" module and play back that audio loop with loudness/pitch set to high settings you will hear, more or less, exactly the same audio. It will be a little glitchy sounding, but it will match and replace each grain with it's real-time counterpart.

'Match rate' controls how often a grain is selected and played back. This only effects the start point of the grain, that is to say it won't just play that 'grain' of audio. It jumps to the start point and will play as for as long as is is set by the 'grain size' parameter.

Finally 'random pitch' will, like in the "Grain" module, randomize the playback pitch of each grain.

# PATTERN

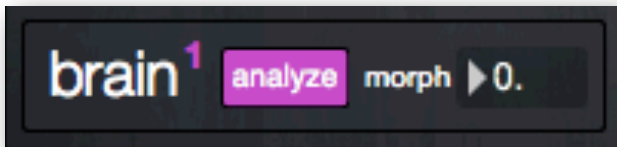


The “Pattern” module lets you record, and loop, anything in the ‘WTPA’ section. This effects button presses, parameter controls, and Monome MLR-style presses. It is coded in a way that allows you to make a pattern out of absolutely anything in this section, including creating loops.

The “Pattern” mode will allow you to start/stop a pattern for as long as you want but once you press ‘record’ it creates a new one. Patterns are not saved so when you load the patch, there is no pattern loaded.

The Monome mapping for this control creates a new pattern when engaged. It is not possible to stop, and restart the same pattern from the Monome. This is intentional, although sometimes not ideal.

# BRAIN



The “Brain” module is the ‘backseat driver’ of the whole ‘WTPA’ section of the patch. It uses complex\* buffer analysis to change all the parameters of the “Slice”, “Grain”, & “Combine” modules. When the ‘analyze’ button is engaged, every time you create a new loop, the “Brain” module will change the above mentioned parameters to something musically/sonically relevant. You can still manually control things, but this is meant as an intelligent preset generator so that when I turn on “Grain” it doesn’t always sound the same, but it isn’t random or unrelated to the contents of the buffer.

I spent a tremendous amount of time coding, tweaking, testing, and updating the analysis algorithms to something I find musically useful.

The ‘morph’ control lets you morph between the current analysis preset and the previous one.

\*Fifteen different analysis parameters are used and weighed against each other different for each control parameter. Each parameter is also weighted against what it was set to previously.

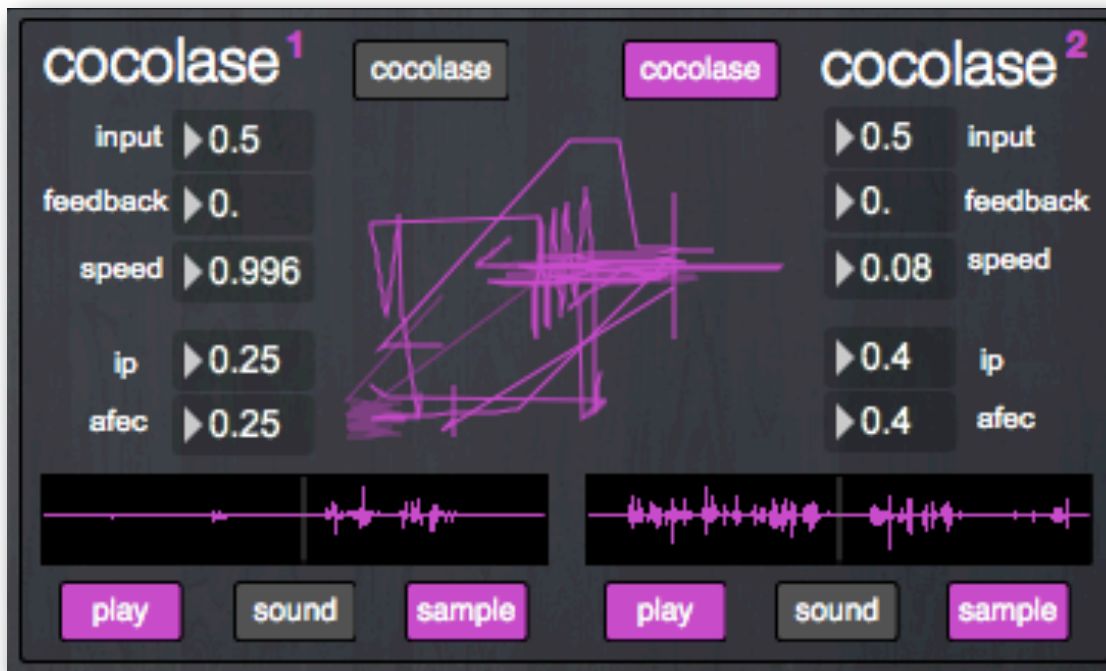
# MLR MODULES



The “MLR” modules are built using the same core sampler as the “Loop” modules in the ‘WTPA’ section of the patch. They are mapped onto a separate page on the Monome GUI and are simply here to allow for more complex/layered MLR-style playing\*.

\*MLR is a very popular and powerful core Monome patch that lets you trigger and loop sections of the buffer using the Monome buttons, with LED indication of playhead.

# MONOLASE



The “Monolase” modules from my program of the same name, which are in turn modeled on the [ciat-lonbarde Cocolase](#), an 8-bit sampler and modulated delay. Most of the routing/modulation control of this module is only accessible via a Monome, although what is in the GUI makes for a great sounding lofi delay/sampler. The controls mirror that of the Cocolase in terms of control and names, so they can be a bit confusing.

The Cocolase modules are always recording into a very smaller circular buffer. When you engage the ‘cocolase’ buttons, we can hear the output of the delay. ‘Play’ starts/stops the playhead. The ‘sound’ button mutes/erases the buffer as you go over that section. Useful for creating silences in the loop. The last button ‘delay/sample’ selects between the two modes. ‘Delay’ is a traditional circular delay whereas ‘sample’ freezes the buffer in it’s current state with no new audio being written.

(Monolase cont..)

‘Input’ controls the input level to that particular module.  
‘Feedback’ works as it would on a traditional delay module.  
‘Speed’ also works as in a traditional delay module, but since the buffer is of a fixed length (1 second) ‘speed’ also controls loop length. The longer your loop the lower the sample rate.

There are several LFOs working behind the scene that are randomized and repatched to create complex control signals for three playback parameters.

The parameters that can be controlled are:

Afec = Pitch/speed modulation

Skip = Jumping position in delay

Flip = Change direction (forward/reverse)

The ‘ip’ works like a threshold for each trigger.

The Cocolase is a complex instrument and system onto itself so for more information you can read the manual for the hardware unit here:

<http://www.ciat-lonbarde.net/cocolase/man/index.html>

# THE CHOCOLATE GRINDER



“The Chocolate Grinder” comes from another patch I wrote of the same name. It is a virtual skipping CD player, with authentic glitch samples and complex/weighted randomization for glitch playback. In addition to the ‘glitch’ sounds of a skipping CD player, this module also emulates the fastforward/rewind sound of digital CD players.

The ‘level’, ‘record’ and ‘stop/play’ controls are all self explanatory. One thing worth mentioning is that there is no overdubbing or pitch/speed control. To make it true to a CD player, those features are not available. The ‘glitch’ control turns on the actual glitch sounds, which are made up of actual samples as well as some synthesized glitches. The ‘level’ control next to ‘glitch’ controls the level of the glitches.

(The Chocolate Grinder cont..)

Along the bottom row we have 'position', which controls the position of the START of the playback window. Then 'window' controls the SIZE of that window. Those two parameters determine when a glitch/skip happens. When the playhead reaches the end of the window, it jumps to the start of the window. This happens with sample accuracy, so that you can 'play' this module by moving the window around the playhead. It will only glitch/skip when you let it. The 'jitter' control adds some noise/randomization to the 'position' and 'window' controls, giving it a more realistic glitch (rather than a predictable rhythmic one).

An important control, that is not exposed, is the 'lockout' control. This sets the sound/speed of the fastforward/rewind. Essentially, the 'lockout' only lets a glitch/skip happen every 100ms or so (the 'lockout' is randomized on each patch load, to simulate different brand/era CD players).

In addition to the 'lockout' randomization, the glitch randomization weighting and levels are also randomized on patch loading. I did this so as to allow for different 'feeling' CD players each time you load it. The changes aren't radical, but it won't always sound the same (like a real skipping CD player).

# PRESET MORPHING/HANDLING

There is complex, behind the scenes, preset generation in this patch for both buffer-based samples as well as incoming audio.

There are three preset sections:

**WTPA1** : Based on buffer analysis of loop1, this generates an intelligent preset for all WTPA1 modules (slice1, grain1, combine1)

**WTPA2** : Same as above but for WTPA2 modules

**EFFECT** : Based on incoming audio analysis this generates an intelligent preset for all input/output effects.

Whenever a new preset is made, it is saved to a preset slot. There are two more presets which are not editable. One has relatively low values and the other has relatively high values. When adjusting the appropriate interpolation parameter (wtpa1, wtpa2, effect) each value is individually interpolated between where it currently is, and either a higher or lower value. Additionally, hand-drawn lookup tables were created to add non-linearities to the interpolation.

This setup creates a very powerful, macro-like “more” and “less” type control, allowing you to control every parameter but without having to micromanage.

# MONOME MAPPING

THE TOP ROW REPRESENTS DIFFERENT "PAGES". EACH ONE BEING IT'S OWN APPLICATION/SETUP

[illegible]

# PAGE DESCRIPTIONS

## PAGE1 - WTPA MODULE/GREATEST HITS

Two sampler/loopers with mlr-style display/control, granular synthesis, grain replacement, sample slicing, pattern recording, input/output fx, audio input based triggering of effects, and circular buffer “reminder”.

## PAGE2 - MLR ADAPTATION

Four sampler/loopers with mlr-style display/control along with four pattern recorders and mute/kills.

## PAGE3 - MOCOLASE

An 8-bit modulated delay/sampler based on the Cocolase by ciat-lonbarde.

## PAGE4 - THE CHOCOLATE GRINDER

A virtual skipping CD sampler.

## PAGE5 - NOT IN USE

## PAGE6 - NOT IN USE

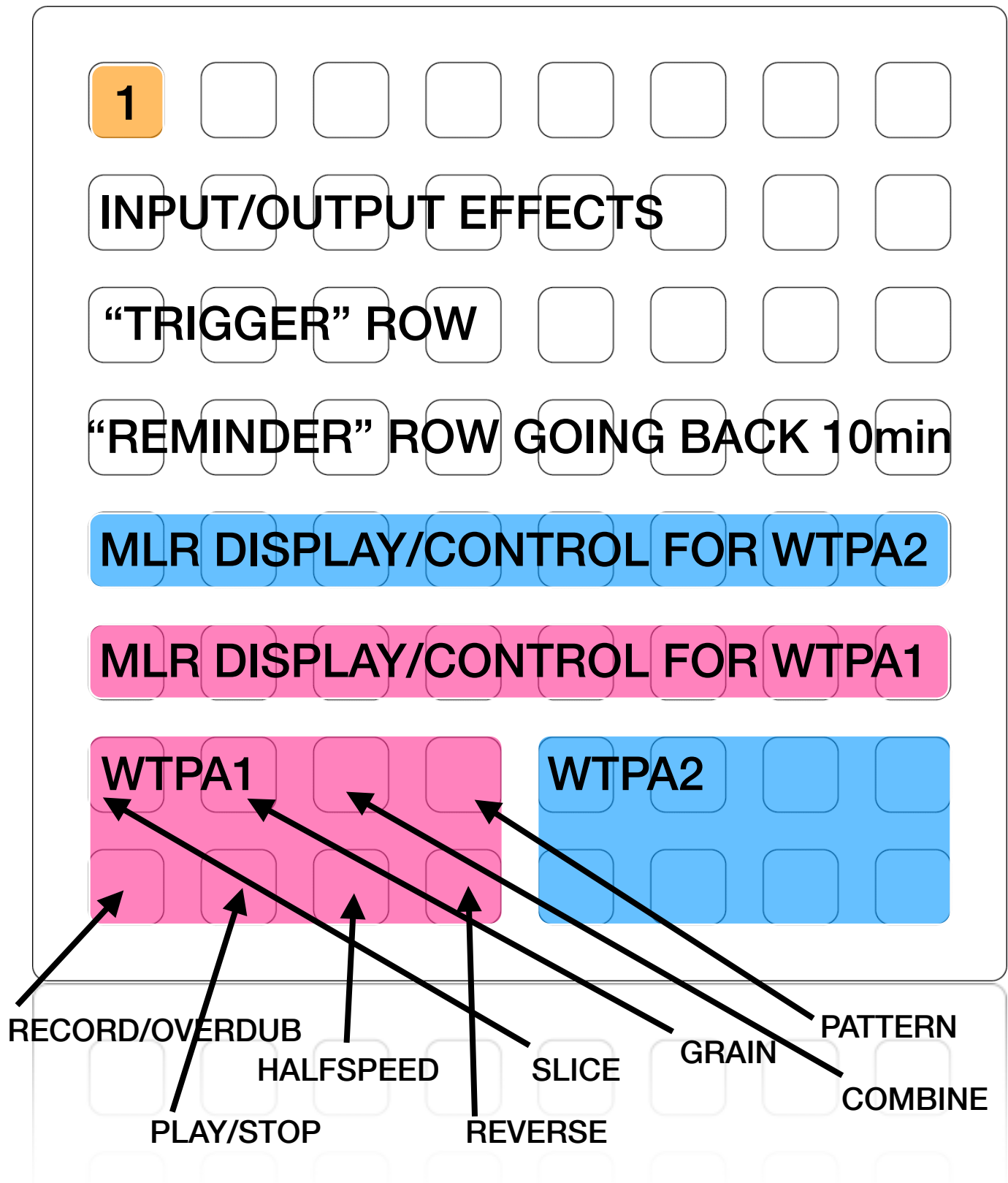
## PAGE7 - INSTRUMENT

A collection of different instruments on subpages including piano/rhodes and adaptations of Fourths and Straw

## PAGE8 - LEVELS

“Fader” style control over four input levels, overall sample levels, overall instrument levels, thru level, and master level.

PAGE1 IS A "GREATEST HITS" TYPE  
PAGE SO IT CONTAINS A LOT OF  
DIFFERENT COMPONENTS



# BREAKDOWN OF TOP THREE ROWS

1

INPUT/OUTPUT EFFECTS

“TRIGGER” ROW

“REMINDER” ROW GOING BACK 10min

EFFECTS:

- 1 - STUTTER (PRE)
- 2 - PITCH SHIFT (PRE)
- 3 - LOFI (PRE)
- 4 - FILTER (PRE)
- 5 - SHUFFLER (POST)
- 6 - CHOPPER (POST)
- 7 - DIRT (POST)
- 8 - REVERB (POST)

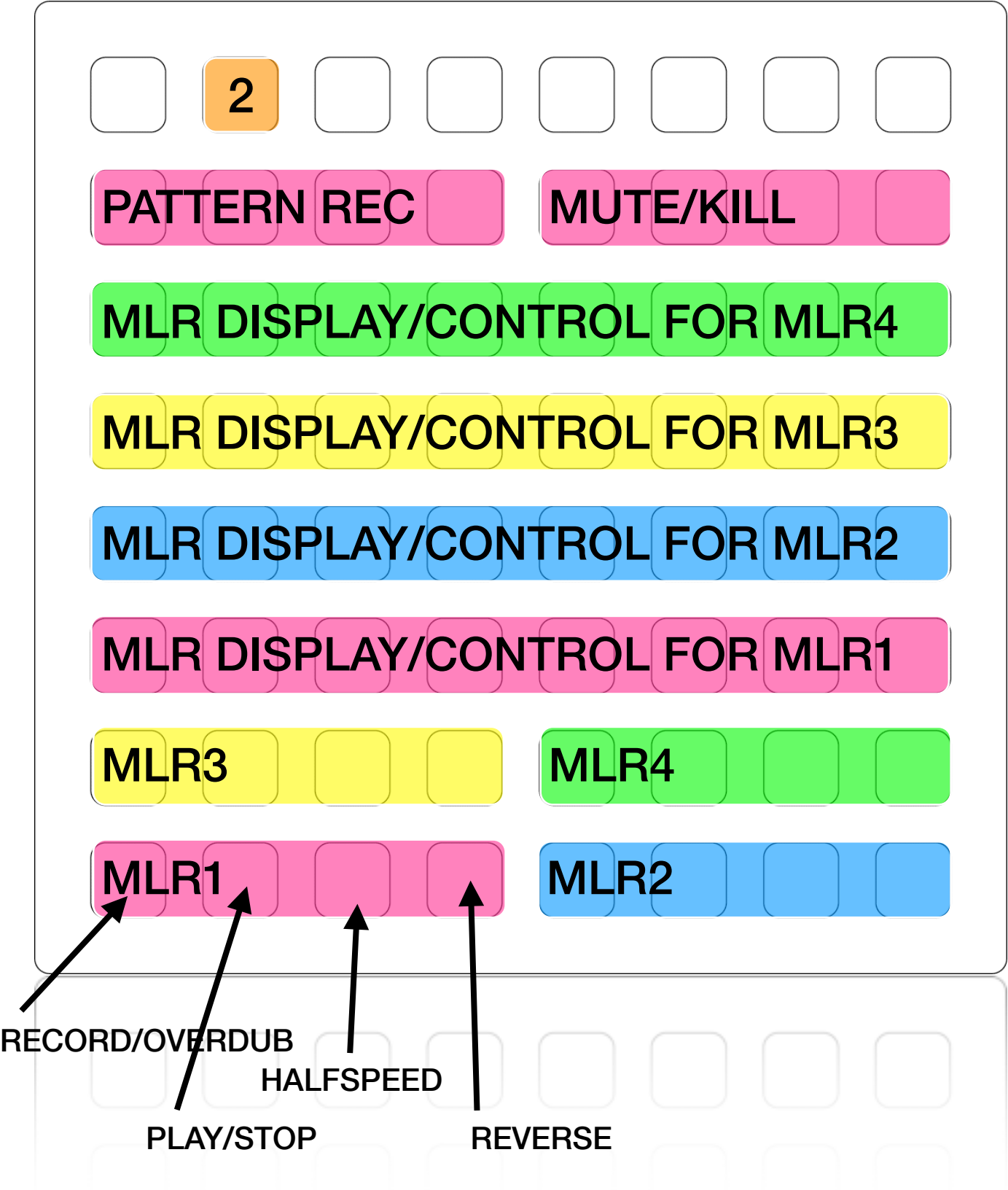
TRIGGERS:

- 1 - RECORD/OVERDUB
- 2 - STOP/PLAY
- 3 - HALFSPEED
- 4 - REVERSE
- 5 - SLICER
- 6 - GRAIN
- 7 - COMBINE
- 8 - POSITION JUMP

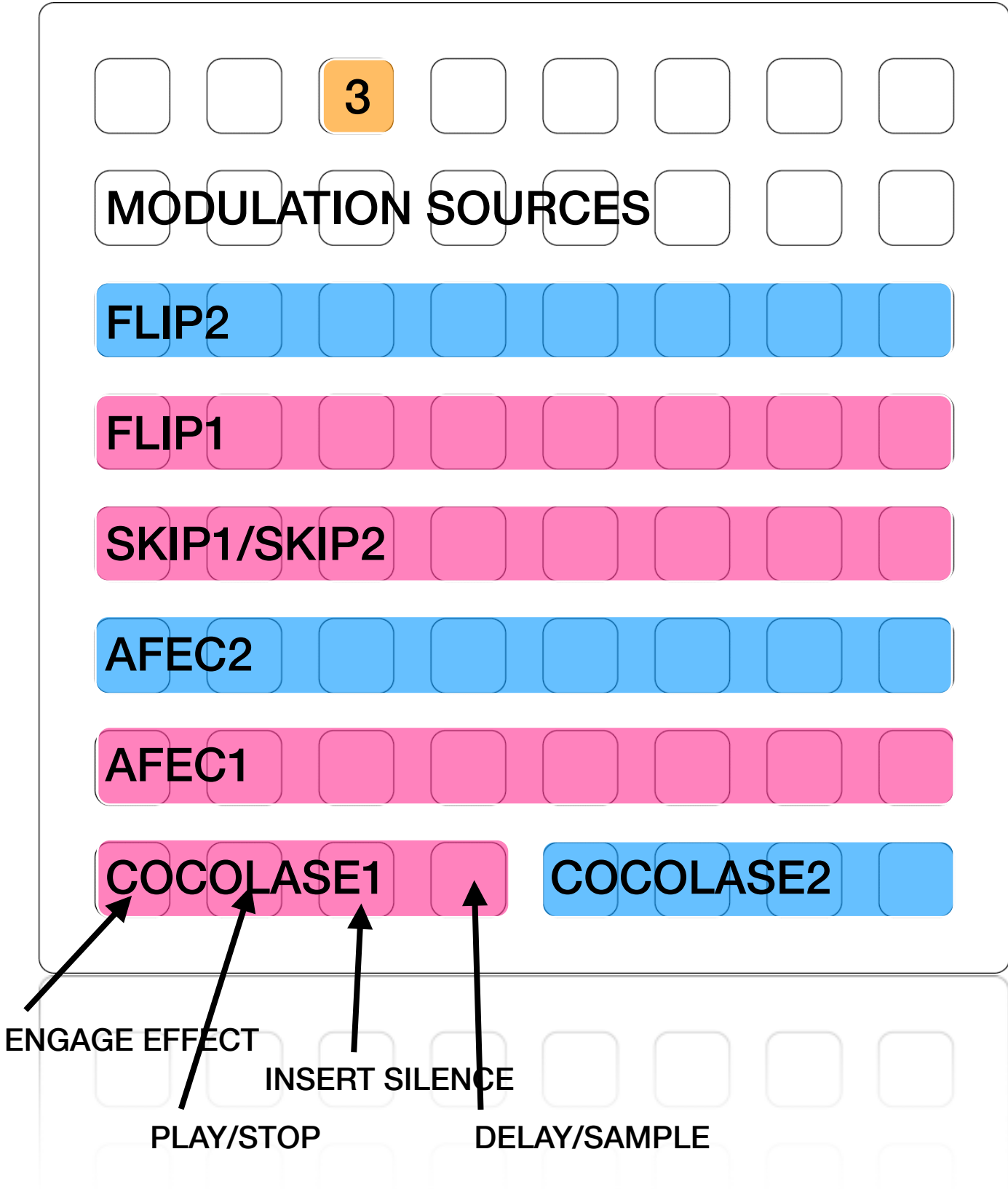
Trigger row works by analyzing incoming audio for attack onsets. When an attack is detected it triggers the designated function. There are two modes. The first mode toggles the function on/off alternately with each attack. The second mode turns the function on, then off automatically.

The Reminder row goes back in time with the left most button being 5seconds into the past, and the rightmost button being 10minutes into the past. The buttons in between are between those values. The 10 minute button does concatenative synthesis on the last 10 minutes of audio (using the same analysis/playback engine as the “Combine” module).

PAGE2 IS AN MLR-ADAPTATION WITH  
FOUR MLR LOOPERS AND PATTERN  
RECORDERS



PAGE3 IS AN 8-BIT MODULATED  
DELAY/SAMPLER BASED ON THE  
COCOLASE



# DETAILED EXPLANATION OF COCOLASE PAGE

3

The modulation sources can be altered by pressing any button on that row. Each source has a rate (random at startup) and can be either square or triangle, and LFO or audio rate. Pressing the buttons toggles between square/triangle and lfo/audio rate.

The parameters that can be controlled are:

Afec = Pitch/speed modulation

Skip = Jumping position in delay

Flip = Change direction (forward/reverse)

Any one of the parameters can be controlled by multiple sources, and combining modulation sources will create complex waveforms.

The bottom row of buttons work as follows.

Cocolase = Engages playback of effect. The delay is constantly running/capturing audio

Play = Stops/start the audio right where it left off

Silence = Inserts silence into the loop.

Delay/Sample = Freezes the buffer

More information can be found here:

<http://www.ciat-lonbarde.net/cocolase/man/index.html>



# PAGE7 IS AN INSTRUMENT-BASED PAGE WITH EACH INSTRUMENT BEING A SUBPAGE



## SUBPAGE DESCRIPTIONS:

1 - Adaptation of “Fourths” using a Rhodes

2 - Adaptation of “Fourths” using a Piano

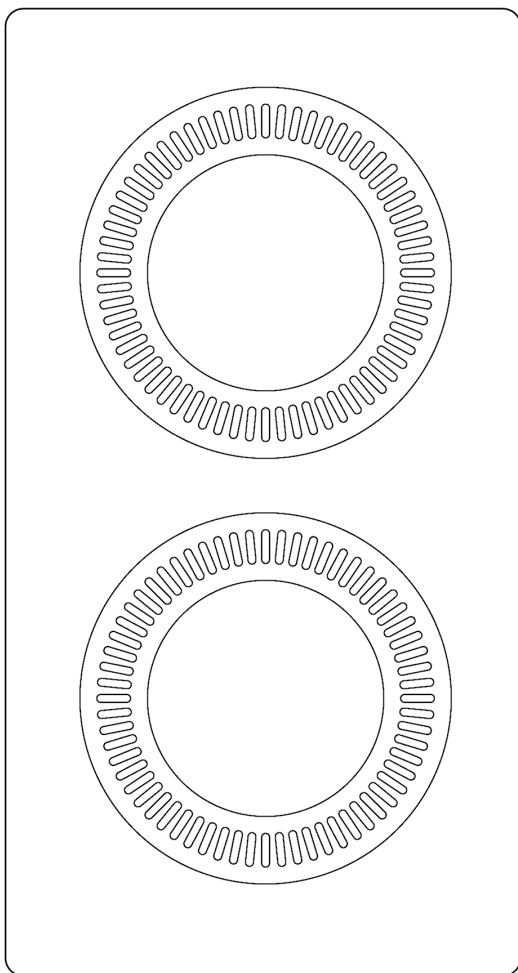
3-8 Adaptations of “Straw”

Straw gets its pitch choices by analyzing incoming audio. The list of pitches is updated every time you revisit Page7. While on the instrument page, the pitches remain the same.

INPUT 1								
INPUT 2								
INPUT 3								
INPUT 4								
MASTER SAMPLE LEVEL								
MASTER INSTRUMENT LEVEL								
DRY THRU LEVEL								
MASTER OUT LEVEL								8


# ARC MAPPING

## EACH MONOME 'PAGE' HAS IT'S OWN ARC MAPPING



**Page1 :** Each encoder cycles through the following controls : loop1 pitch, loop2 pitch, wtpa1 fx control, wtpa2 fx control, loop1 start/end, loop2 start/end, in/out effects

**Page2 :** The top encoder is mapped to mlr1/mlr2 pitch and the bottom encoder is mlr3/mlr4 pitch.

**Page3 :** The top encoder is cocolase1 pitch, and the bottom one is cocolase2 pitch. Pressing the top one randomizes the 'sidrassi brain' modulation 'petals' and pressing the bottom one randomly repatches the modulation points.

**Page4 :** Exactly mapping from "The Chocolate Grinder". Each encoder is it's own CD sampler. The first press starts recording a loop and the second press stops it. Pressing it a third time starts sample playback and any subsequent presses toggle playback on/off. Turning the encoder changes the sample position. Press +turn changes the window. Press+hold clears/resets that sampler to begin recording again.

**Page8 :** The top encoder controls master volume. Pressing the top encoder mutes the patch. The bottom encoder cycles through the following controls : in1, in2, in3, in4, sample level, thru level.

# STUFF

For any questions, comments, bugs, or feature requests contact me at [rodrigo.constanzo@gmail.com](mailto:rodrigo.constanzo@gmail.com)

Special thanks to Alex Harker, Pierre Alexandre Tremblay, Dominic Thibault, Bejnamin Van Esser, and countless people on the cycling74 forum.

Patch can be downloaded at my webpage:  
<http://www.rodrigoconstanzo.com>