Report : DIY & Hacking

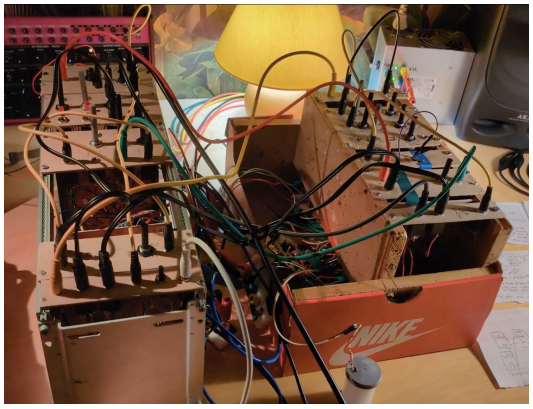
Olaf Léon[1†], Thomas Mellier[1†], Yann Rives-Hapiak[1†]

[1]UFR Sciences et Techniques, A.M.U, Pl. Victor Hugo, Marseille, 13003, France.

Contributing authors: olaf.leon.91@gmail.com; thomasmellier27@gmail.com; yann.riveshapiak@gmail.com;
[†]These authors contributed equally to this work.

Abstract

This project aims at designing and building our own experimental musical instrument. It enters into the large aesthetic and technical movement of DIY and hacking. The instrument incorporates both digital and analog technologies along with circuit bending techniques. Its design was guided by musical ideas and a composition emerged from its design. This report details the entire development cycle, including design choices, technical challenges, and areas for improvement. We hope to offer insights for future makers interested in building their own experimental instruments. Through this work, we contribute to the ongoing discourse on DIY sound synthesis and the intersection of engineering and artistic expression.



Keywords: DIY, Hacking, Modular synthesis

1 State of arts

The DIY approach of this project defends the "will to create and to share" without any constraints, whether they are market-linked, economical or ownership-linked (Benhaïm, 2019). Our circuit bending ideas are led, among other, by Reed Ghazala and his book Circuit-bending: Build your own alien instruments (2005) sharing his creative and economical resources with autonomy, autodidact and hacker ethics.

The experimental process of composition is rooted in Varèse's theory. "I do not write experimental music. My experimenting is done before I make the music. Afterwards, it is the listener who must experiment." (Varèse, as cited in Lucier, 2018, p.x). The main references and inspirations of the composition are Eliane Radigue, Suzanne Ciani, John Frusciante, Ryoji Ikeda, Floating Points, Caterina Barbieri, and Karlheinz Stockhausen (see the statement of intent in the appendix).

Modular synthesizers appeared in the 60's, R. Moog and D. Buchla were the first one to design them. The main characteristic of modular synthesis is the separation of essential modules, such as oscillators, filters, amplifiers, and envelope generators into individual units. This modular approach allows for flexible signal routing and patching, giving users greater freedom to explore and manipulate sound. Additionally, the use of Control Voltage (CV) signals to control the parameters of each module through electric signals is a fundamental feature. Nowadays, typical modules are well documented, schematics and PCBs made by different designers are often accessible in open-source. In this project, we got inspiration from multiple modular synthesizer projects (such as Mutable Instruments, YuSynth, AISynthesis [3]). The modules were also based off classical electric circuits revolving around the use of operational amplifiers (inverting gain, non inverting gain, differential operation and summing).

Digital modules in this project are inspired by Émilie Gillet's designs at Mutable Instruments [3], which pioneered the use of STM32 microcontrollers in open-source modular synthesis. Like her modules, our system employs STM32 as an embedded platform, a specialized computing system that integrates a processor, memory, and input/output peripherals within a dedicated electronic system.

The software architecture uses key embedded system protocols and file systems optimized for real-time audio processing and file management:
- I2S (Inter-IC Sound): synchronous serial communication protocol designed for digital audio transmission. Operates with separate clock and data lines, ensuring synchronization between components such as DACs and ADCs.
- DMA (Direct Memory Access): hardware feature that allows peripherals to transfer data directly to RAM without CPU intervention. Reduces latency and optimizes processing efficiency.
- SPI (Standard Peripheral Interface) : communication interface with external devices
- FATFS [3] (FAT File System): open-source file system, enables efficient file read and write operations.

2 Development and composition

2.1 Hardware modules

Each module was designed and validated on breadboards. Then, each electrical schematic was transfered onto the *Kicad* software to create the PCB print. We also used the *SPICE* additionnal module to make a transient simulation of the voltage. After untangling the rastnest and routing every component together, we used the *FlatCam* software to create a G-CODE file for the CNC engraving machine.

Every module was made on a 100x160mm, one-sided copper, non-conducting plastic plate. The CNC's reamer was .8mm. Electrical schematics and PCBs can be found in the appendix.

2.1.1 Mixer IN

The module Mixer IN is a basic input gain manager for 4 channels.

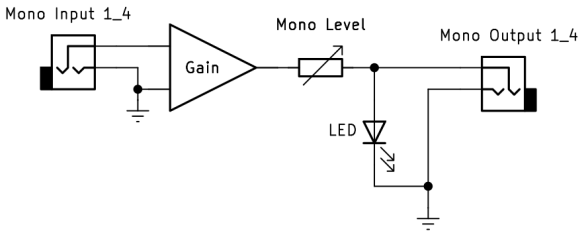


Fig. 1: Mixer IN

Each input channel is Line level and has Line impedance except for the first one which is designed for microphones. Each one has a potentiometer controlling the gain. We are using a simple opamp circuit, a non-inverting gain. The variable resistor is $10k\Omega$ and the resistor to ground is $2.5k\Omega$ hence the ratio (at max potentiometer value) is $\times 5$.

Line's voltage is $2V_{pp}$ and Eurorack norm's voltage is $10V_{pp}$. When a channel is active and correctly amplified, a red LED lights up.

2.1.2 Mixer OUT

The Mix Out module is similar to the Mix In: the gain is made by using a 4-channel opamp in the inverting gain circuit with a resistor ratio $\frac{1}{20}$, so each channel goes from $10V_{pp}$ to $0.5V_{pp}$.

All channels are summed into one, using a summing circuit. The last operation is another inverting gain with a ratio $\frac{3.3+10_{var}}{3.3}$ allowing for a $\times 1$ to $\times 4$ gain (modified by a $10k\Omega$ potentiometer). This is used to balance the line level when multiple but

not all 4 channels are active (if we were to use only one channel, the resulting voltage would be $500mV_{pp}$, and $\times 4$ makes it $2V_{pp}$, Line level).

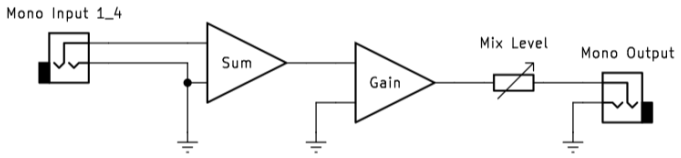


Fig. 2: Mix Out MLD

2.1.3 VCA

We chose to design two Voltage Controlled Amplifiers, VCA's, inspired by Yusynth's simple VCA (3).

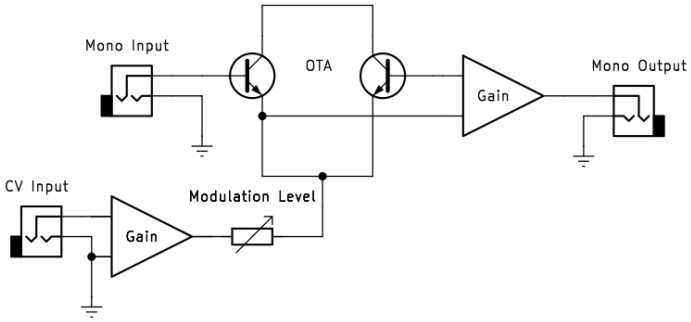


Fig. 3: VCA MLD

Both Gains are classical opamp-based circuits. The transistor-based operational amplifier is an Operational Transconductance Amplifier (OTA). By using two of the exact same transistors, symmetrically positioned, the gain applied to the input voltage is controlled by another current (CV In signal). Its output is a current, hence we route it to another classical opamp-based gain (controlled by a potentiometer) to get the wanted output voltage. OTA's are quite useful and if adjusted correctly (trimmers) can produce a very clean gain (with low distortion and a good SNR).

The potentiometer controls the gain of the CV In signal, it is then possible to activate progressively the voltage controlled amplification.

2.1.4 VCF

Our voltage controlled filter modules are second order low pass filters. As a model we use the schematics given by Moritz Klein. In order to control RC filters with voltage,

we used vactrols. The LED intensity depends on the CV and changes the value of the photoresistance of the RC filter.

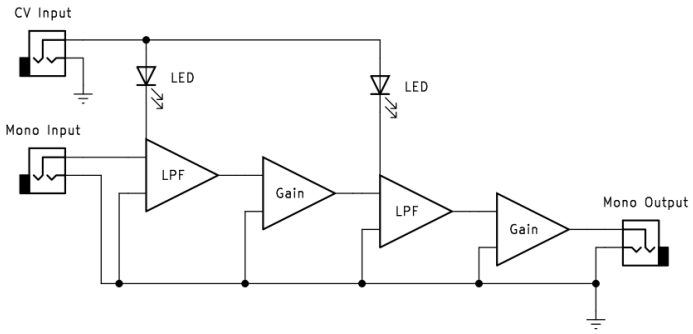


Fig. 4: VCF MLD

2.1.5 VCO

Our Voltage Controlled Oscillator module is hybrid: signal (signal generation, CV modulation etc...) is digitally processed in the STM32 microcontroller. For the analog part, it uses two four-channel op-amps for gain control: CV signals are Eurorack normed and in order to enter I2S2s, signals are converted to Line level. We do the same thing in the inverse order for the outputs. The three CV inputs and the three potentiometers control the frequencies of the three first output signals (custom shape and two sawtooth), their value is computed inside the STM32 code (2.2). The fourth output signal is white noise.

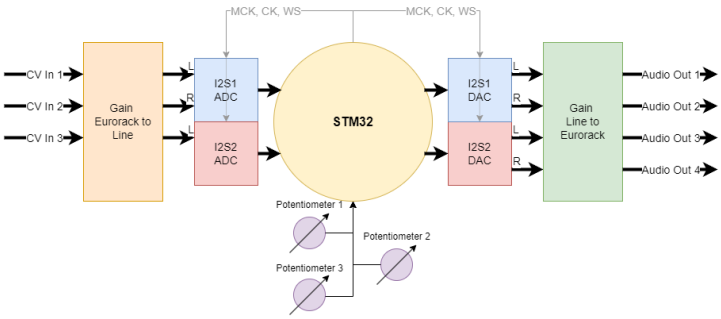


Fig. 5: VCO MLD

2.1.6 LFO

The LFO module is also an hybrid module. The two opamps (input gain/output gain) are used for eurorack to line conversion (and the other way around).

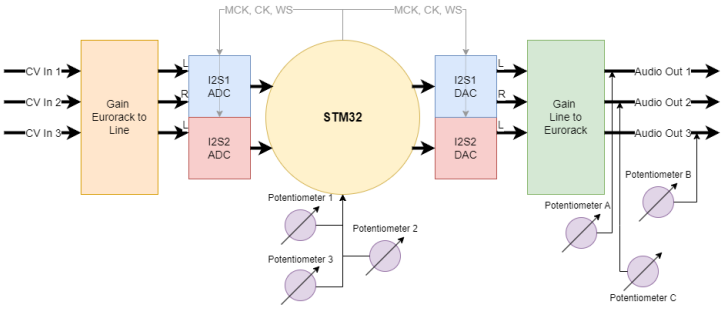


Fig. 6: LFO MLD

Three potentiometers control the gain of the output signals (sawtooth, custom and square).

2.2 Software

The software development process relies on CubeIDE for programming and FATFS for file management. The C codes have to be efficient enough to run in real time while considering that the STM32's storage is limited (512 Kbytes flash / 96 Kbytes SRAM). Each digital module uses two I2S2 modules for ADC/DAC stereo streaming granting four monophonic Ins and Outs per module. The Looper module is only presented in this section since its production stopped at software debugging stage. This report presents the actual code running on the digital modules and discusses possible improvements. Captures of the functions' code can be found in the G section of the appendix.

Looper Concept

This module allows the user to control recording via CV and a switch button, storing audio in a finite-size circular buffer. Similarly, playback is triggered using CV and a switch button, with the possibility of re-triggering before playback ends, enabling granular synthesis, where the playback frequency is determined by the CV signal frequency. Additionally, the system is designed to generate a CV signal when playback ends, allowing for automatic looping.

Real Time Audio

Each I2S interface operates with one circular input buffer and one circular output buffer, where DMA (Direct Memory Access) transfers half-word wide data samples. To ensure continuous audio streaming without data loss, a ping-pong buffering mechanism is used. In this system, DMA alternates between two buffer halves: while one half is being filled with new data from I2S, the other half is being processed by the CPU. Once DMA completes filling half of a buffer, it triggers an Interrupt Service Routine (ISR), signaling to start processing the newly filled half while DMA begins filling the next half. Another interrupt is triggered when the buffer is fully filled, allowing the CPU to process the second half while DMA returns to the first. This alternating cycle ensures that audio processing happens in parallel with data transfer (see figure [7]).

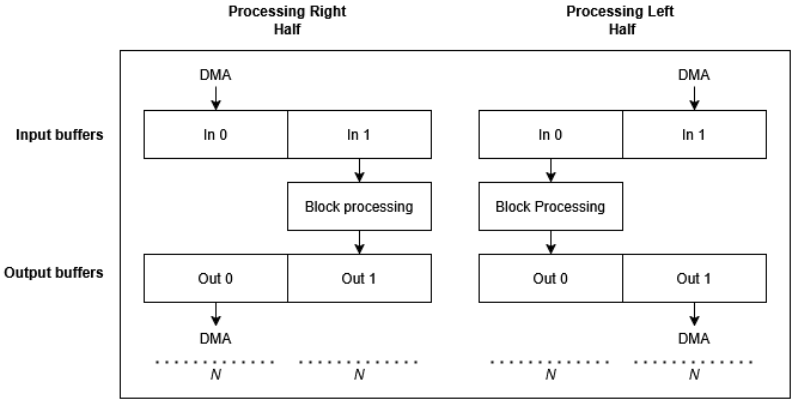


Fig. 7: DMA double buffer stream

The I2S peripheral converts input voltages into 24-bit values within a 32-bit frame at a 46.875 kHz sampling rate. Since DMA operates in circular mode with a data width set to half-word (16 bits), each sample is split into two 16-bit parts within the buffers. Each I2S buffer contains 128 integers (64 per stereo channel), but because samples are split, each buffer effectively holds 32 complete audio samples, corresponding to 0.68 ms of audio. The audio processing chain must complete its computations within 0.68 ms to avoid missing new data. The available computing time can be increased by expanding the I2S buffer size, reducing the frequency of DMA interrupts and allowing more time for signal processing.

Interfacing with STM32 modules

The digital modules are receiving and transmitting audio via Pmod I2S2 ADC/DAC (3). These converters are designed for stereo audio (Line voltage) which implies that they filter out the DC component of signals and make continuous signals impossible to use. Our modules are all using monophonic audio so we chose to use the digital modules' stereo channels as 2 monophonic audio channels.

The modules are also equipped with three powered potentiometers (LFO and VCO) for control or powered switch buttons (Looper), generating voltages between 0 and 3.3V. These voltages are converted into 12-bit integers using the STM32's integrated ADC. This ADC operates with DMA and ISR in normal (one-shot) mode.

Looper module is time based and to address the storage limitation, we connected the STM32 to an SD card for extended storage (3). Interaction with the SD card is managed via SPI and FATFS.

Code Architecture

All digital modules share the same code architecture, differing only in the processing section. The function calls follow the same order as shown in the following figure.

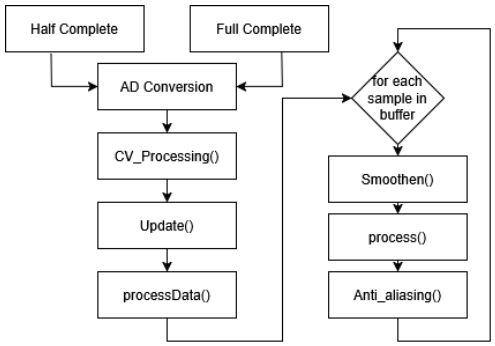


Fig. 8: Code Architecture

Half / Full Complete	HAL's callback function called when buffers are half filled, updates pointers to the buffers
AD Conversion	converts input voltages to floats (-1, +1)
CV_Processing()	computes parameters based on CV input and analog input
Update()	actualizes target parameter value
processData()	processes audio buffer within a for loop
Smoothen()	gradually adjusts parameter value based on previous parameter value and target parameter value
process() [VCO/LFO]	generates signals (saw, square, custom waveform, noise)
process() [Looper]	writes and reads values from and to buffers exchanged with the SD card
Anti_aliasing()	filters the audio outputs to prevent aliasing (2nd order lowpass IIR filter 23kHz)

Table 1: Functions Table

The processData function processes data one half buffer at a time making it the most computationally expensive function.

At very low frequencies, the I2S2s DC filter removes most of the signal, preserving only rapid changes, such as the half-cycle inversion of the square wave or the reset of the saw wave to -1 at the start of each period. As a result, the output signal degrades into a series of periodic clicks.

In the Looper code, transmit buffers are sent to the SD card after processData() with the buffer_to_SD() function and receive buffers are copied from the SD card before the processData() with the SD_to_buffer().

Audio signals are filtered by the Anti_aliasing() function. However, this filter's order is not sufficient to completely prevent aliasing. We tried to implement a fourth-order filter, but it was too computationally expensive.

Progress Status

Module	Issues	State	Solution Ideas
VCO	Anti aliasing's filter should be 4th order	Functional	Adjusting I2S Buffer Size to gain computationnal time
LFO	Lacks 1 channel	Functional	Adjusting I2S Buffer Size to gain computationnal time
LOOPER	Audio is bitcrushed because of write/read time	Writing and Reading works but audio data loss	Adjusting I2S Buffer Size to gain computationnal time + using DMA and ISR for the SD card interactions
All Modules	CV inputs can't be continuous because of the I2S2 DC filters		Bypassing I2S2 DC filter / Using STM32 analog inputs (12 bits) / Using ADC+DAC modules using I2S without DC filter

Fig. 9: Progress Status table of each module

2.2.1 Bent modules

Our instrument has two modules based on circuit bending. Both of them use a printed circuit found in old electronic toys. These toys were chosen in the sound investigation process. They should fit the composition's aesthetics : electronic, noisy, and harsh sounds and be hardly doable with other modules. They had to be short enough so that looping them would melt them into slow textures. The second part of this investigation is the deformation of these sounds by hacking the circuits of the toys[1].

In each circuit we distorted, the main goals were to be able to (1) control the pitch, (2) select the sound to play among the samples available in the chip, and (3) loop the sample so that it can play a continuous sound. These tweaks could be achieved by welding potentiometers and transistors in specific spots in the printed circuits. The process of finding the spots likely to affect one of these three goals was based on Rekoff's method for reverse engineering (1985) consisting in a hypothesis-disassemble

[1] Used toys sound examples available *here* (piano carpet) and *here* (helicopter). Every sound extract from this document is accessible in a SoundCloud playlist *here*

cycle and considering the technical object as hierarchical structures.

The first bent module is from a helicopter toy that can play three different sounds that loop as long as their wired. A potentiometer allows to change the pitch.

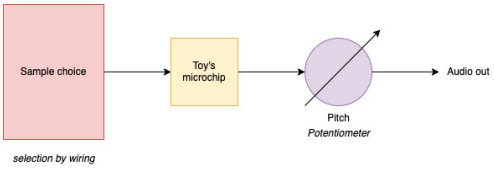
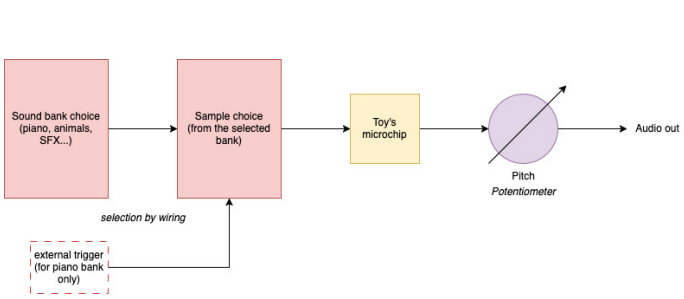
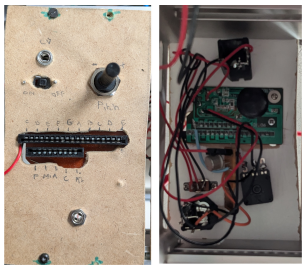


Fig. 10: Bent toy helicopter MLD

The second module is from an electronic piano-carpet toy. Forty-six different samples can be played (animals, sfx, piano notes) but only the piano keys can be triggered by the CV input. To select a sample, one must link the bank slot (on the bottom) to a sample slot (on the top). To chose the piano key controlled by CV one must plug the wire coming from the panel to the sample slot.



(a) MLD



(b) Module circuit and panel

Fig. 11: Bent toy piano module

2.3 Composition

2.3.1 Evolution of an intent

In order to plan and fix aesthetic ideas, we put together a statement of intent. First, in the form of a mindmap (Fig [12]). The first idea was to compose a piece in two main parts : a drone section slowly intensifying leading to a more active and rhythmic part. This step also permitted us to gather sound influences and textures inspiration.

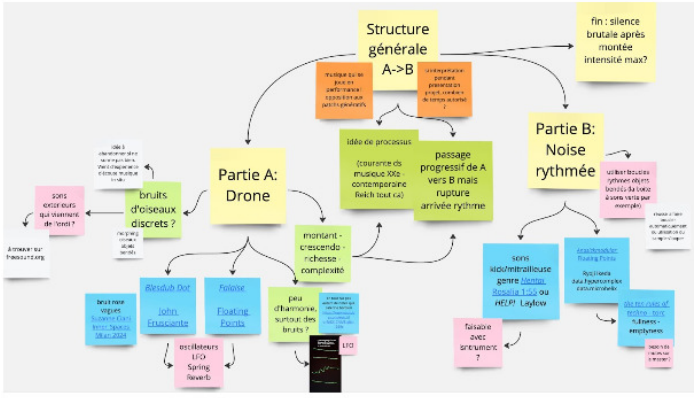


Fig. 12: Aesthetics research : a mind map

Then, the intent took the form of a written argued document focusing on the work on the sound itself that characterizes modular synthesis music. Its structure follows Richard's methodology (2017). A major inspiration of the composition is Inner Spaces by Suzanne Cianni : the beginning of the piece inspired the second approach of this composition making the connection between the way one listens to this kind of piece and the way one looks at the ocean. A first model of the composition was made to illustrate this intent. *[sound extract]*

This first draft of the composition was made using a prototype on VCV Rack replicating our instrument virtually. This prototype aimed to try the first musical ideas using the modules we were designing. It helped to chose custom wavetables for the VCO and LFO and to set priorities in the development of modules and highlighting the importance of filters. This step of the project reinforced our team workflow : the connection between the technical and aesthetic researches (setting priorities, custom waveforms...). A second model of the composition was recorded with theses new features and using samples from the bent toys. *[sound extract]*

2.3.2 Composition process : an instrumental approach

Once most of the modules were functional, we could start to get familiar with the instrument. As intended, the patches of the digital prototype sounded very different on the real instrument. Thus, we started to investigate what is possible do to with the instrument and its limits by trying new patches that would keep the intent of the first drafts. This experimental approach, based on the instrument, led to the find of interesting textures, and thus, patches. A large part of these investigations were recorded (sound and video) so we could listen and watch it again to find the wiring back. From this phase, we got three patches[2] that could co-exist on the instrument and led to the form of the composition : a succession and-or superposition of these

[2]Extract from patch A *here*, patch B *here*, and patch C *here*.

patches slowly evolving from a texture to another. The scheme of these sub-patches (Fig. [13]) is the first part of the score for the music piece.

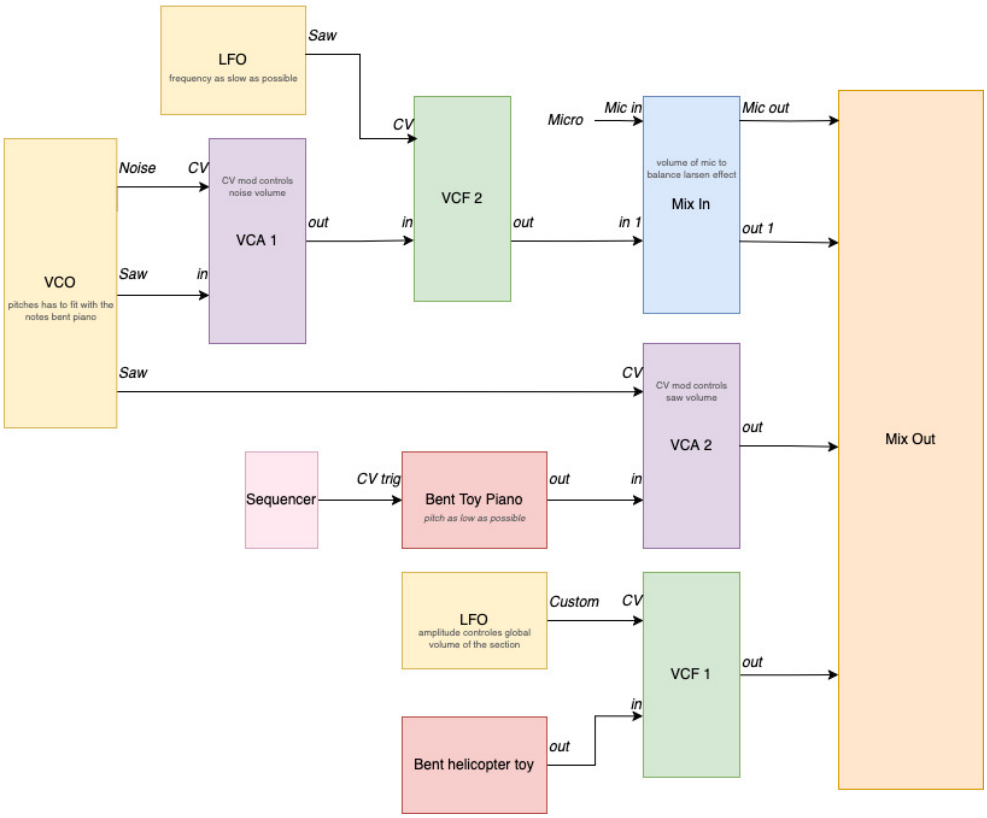


Fig. 13: Global patch schematic

We set the order of the 10 min succession of sound-patch ideas by trying transitions between textures in different orders. Once the patches and their connections in time is established, most of the interpretation consists of fading in and out texture and adjusting pitches. Thus, the composition score consists of the patch schematic and the list of instructions stating what is intended and what potentiometer should move to do it [14]. A performance of the piece is accessible *here* (audio only) and *here* (video on YouTube).

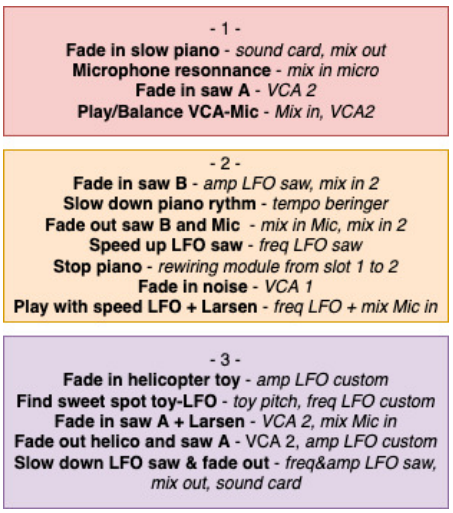


Fig. 14: Instructions for interpretation

An interesting aspect of this process is the high constraints that the composition is under. The modules sometimes behave unpredictably (VCO may be clean or noisier, its pitch may slightly vary, LFO shapes are not what we predicted when frequencies are too low, filters need CV input to allow the signal to pass, sounds from the bent modules may change depending on which module its output is wired to...). Some of these constraints became features of the composition as its unpredictability may create valuable surprises during improvisation sessions. For instance, the VCF can be used to control volume, VCAs became signal mergers with a control on the volume of one of the two signals, the pitch variation of the VCO around a set frequency was used to tune-untune with a sound from a bent module. Another example is the scratch some of the potentiometers make when one rotates them. Once we spotted which one glitches, we can use it as a new sound for our textures : one of the improvisation that led to the composition largely used this phenomenon. *[sound extract]*

3 Critical Review

This project successfully combined DIY electronics, DSP, and experimental music to create a modular synthesizer blending digital and analog techniques. The STM32-based modules enabled unique sonic textures despite the challenges that appeared with real-time processing and interfacing, particularly in SD card read/write speed and I2S2 filtering. The quirks of the instrument inspired the creative process, reinforcing the link between technical limitations and artistic expression. The project stands as a contribution to DIY modular synthesis, offering insights for future refinement and innovation[3].

[3]This project's resources are available on Google Drive *here* and on Github *here*.

References

Benhaïm, S. (2019). DIY et hacking dans la musique noise. Une experimentation bricoleuse du dispositif de jeu. Volume!. La revue des musiques populaires, 16 : 1,17–35. https://doi.org/10.4000/volume.7230

Ghazala, R. (2005). Circuit-bending: Build your own alien instruments. Wiley Pub.

Lucier, A. (Ed.). (2018). Eight lectures on experimental music. Wesleyan University Press.

Rekoff, M. G. (1985). On reverse engineering. IEEE Transactions On Systems Man And Cybernetics, SMC-15(2), 244-252. https://doi.org/10.1109/tsmc.1985.6313354

Richard, M., Théberge, M.-P., Majeau, C. (2017). LE DISPOSITIF DE CRÉATION/MÉDIATION AMALGAME : CROISER LES POSTURES ET TRANSGRESSER LES FRONTIÈRES. Revue de recherches en littératie médiatique multimodale, 6, 1043752ar. https://doi.org/10.7202/1043752ar

Stack Overflow. https://stackoverflow.com/questions (Consulted 18.02.2025)

STMicroelectronics community. https://community.st.com/ (Consulted 13.02.2025)

Elm Chan. FatFs Generic Fat Filesystem Modules. http://elm-chan.org/fsw/ff/ (Consulted 12.02.2025)

Usson, Y. (s. d.). Synth DIY. https://yusynth.net/index_en.php

AI Synthesis. (2024, June 11). Home - AI Synthesis. AI Synthesis - Tools and Guides for the Aspiring Modular Synth Builder. https://aisynthesis.com/

Gillet, É. (n.d.-b). Mutable Instruments documentation. https://pichenettes.github.io/mutable-instruments-documentation/

PMOD I2S2 Reference Manual - Digilent Reference. (n.d.). https://digilent.com/reference/pmod/pmodi2s2/reference-manual

PMOD MicroSD - Digilent Reference. (n.d.). https://digilent.com/reference/pmod/pmodmicrosd/start