A digitally controllable analog no-input mixer

Sound and Music Computing
Department of Architecture, Design and Media Technology
Aalborg University
2024

Master Thesis

Submitted By

Gianluca Elia
gelia20@student.aau.dk
Aalborg University Copenhagen

Supervised By

Daniel Overholt
dano@create.aau.dk
Aalborg University Copenhagen

AALBORG
UNIVERSITET

Abstract

The no-input mixer is an appealing instrument for practitioners seeking a constant negotiation of their musical desires with the instrument's complex agency and affordances. Offering a simple setup, a re-invention of a traditional instrument, it resignifies an utilitaristic control space to a more obscure and surprising field of exploration, a fertile ground for artistic practices and languages to emerge.
This project proposes a new hybrid instrument: an analog no-input mixer embedding a digitally controllable interface, introducing previously impossible interaction and mapping techniques (from storing, recalling, interpolating and sequencing of control points, to semi-automatic audio-driven mapping devices and agents), to evaluate their affordances on musical practices with the instrument and to cultivate the opportunity for novel languages and practices to emerge from them.

Contents

1. Introduction

This works presents the design and creation of a digitally controllable no-input mixer. Initial motivations and research questions are presented here, often referencing terminology and context which will be expanded in more detail in sec. 2.

1.1. Motivation

No-input mixer practice demands musicians to discover the affordances of a non-trivial control space, and to constantly negotiate their musical intentions with them. Digital control is most prominently on the other side of the interaction spectrum, promising precise, efficient control and reproducibility. Furthermore, although some academic works are starting to emerge, and several artists have released artworks with the instrument, there is scarce documentation of theory and playing approaches with no-input mixers, perhaps for a structural reason: the complex nature of the instrument itself doesn't promote a standardized praxis. The present work also doesn't aim to formalize any praxis, but is rather a continuation of my personal experience with the instrument.

I have been playing no-input mixer for the past five years, without external audio effects and mostly in improvisational and harsh noise settings. Parallelly, I have been working on machine-listening, machine-learning, corpus-based concatenative synthesis, and more generally audio programming. I see these as two almost-opposite poles: one being immersed in a more or less obscure practice, the other more oriented to serviceful design. This project originates from the desire to mix them, to explore the developments this contamination can afford on my musical practice with the instrument.

Analog mixers exhibit noise and quirks often ascribed to "imperfections" in the circuitry, which are integral parts of their sounds and playing experience. When attempting to model those circuits digitally, I've encountered a fundamentally different sound and behavior, dominated by the prominence of digital sampling (e.g. the sampling rate and its harmonics being dominant resonant frequencies) and lacking key factors of the typical no-input mixer interaction, i.e. "sensitivity", "directness" and "rawness". Instead of trying to circumvent those differences, which in fact constitute characters of a different musicking domain, this works chose to re-design an analog mixer with built-in feedback routing, to keep the signal processing path analog and thus to stay closer to the musicking experience with analog electronic instruments. The resulting self-contained instrument is less prone to being modified, as circuit design and fabrication compares to digital algorithm coding. While this is a major disadvantage for prototyping, it can also be an advantage for musical practice, offering a clearer limit and distinction between instrument design and playing. Once the instrument is built, its analog part can't be easily changed, forcing to focus on the interaction with it. However, since the present work being an hybrid system, it lends to continuous development and re-design of its digital interface.

One practical reason for introducing digital control to analog no-input mixers has been the need for more precise access to resolution offered by analog potentiometers as control interfaces. No-input mixers are often sensitive to the smallest variation of parameters, which is sometimes frustrating to interact with using small potentiometer caps. The present work starts with affording a resolution that although being discrete, is much finer than what is conveniently controllable on traditional audio mixers.

Furthermore, no-input mixers afford an engaging, co-creative approach characterized by continuous learning and adaptation, possible because of the unpredictability of the instrument, but at the same time because of the non-arbitrarity of such unpredictability. Performers can learn some patterns and behaviors that are more or less reproducible, lending themselves to a deepening of the relationship with the instrument. Digital control offers tools to work with this pseudo-reproducibility, for example storing and recalling points in the control parameter space.

Finally, digital control and analytical tools allow for the introduction of techniques from machine-listening and machine learning, that are part of my milieu, but couldn't be applied to no-input mixing without resorting to sampling. Working with samples of a no-input mixer increases the distance from real-time interaction and its non-linear dynamics, constraining to playback and modification of recorded material. With a digitally controlled analog no-input mixer, computer analysis can be used to affect the parameter space only, while sound is still produced in real-time by the instrument's feedback process. This is particularly relevant when developing (semi-)autonomous agents, which could then work to explore the mixer's audio process, instead of constituting a separate one based on recordings.

1.2. Research questions

For the motivations above, the main research question of this work is an exploratory one: what happens when digital control is applied to no-input mixer musicking? What previously impossible techniques can be introduced an how do they transform practice and performance with the instrument?

As a hybrid instrument, this work lends itself to comparison to analog and digital instruments, but most importantly to their related musicking. How does a digital interface affect the experience compared to analog no-input mixers? Are the new affordances worth the added level of abstraction/mediation? Or, from the other side, how does the analog circuitry affect algorithmic design? Are its reactivity and rawness worth the extra cost and lesser flexibility? In other words: does the instrument afford an engaging musicking experience, allowing for creative development without becoming "overly analytic"?

No-input mixers offer a peculiar type of instrument agency and autonomy (e.g. some settings produce long, irregular loops, oscillating more or less stably between attractors, other settings slowly move towards a more stable attractor). This is due to circular causality, feedback loops, often metaphorized as the instrument "listening" to itself. What kinds of autonomy can be achieved by introducing computer analysis, algorithmic processes, and cybernetic mappings to this "self-listening" loop?

2. Background and related work

This section presents relevant work on no-input mixers, and recent conceptual developments about musicking with feedback instruments, a field that has found renewed interest in recent times with concepts like "unmastery" and "complexity literacy". A novel research field emerged around the "Feedback musicianship network" and "Self-resonating vibro-tactile instruments", which constitute the background and research context of the present work. Finally, a comparison is made with other feedback instruments developed within and around this context.

2.1. No-input mixers

A no-input mixer board is a traditional audio mixer wired in feedback: audio ouputs are routed back into channel inputs, making the instrument a self-resonating sound generator. This technique was brought to fame by Toshimaru Nakamura, who emerging from the japanese so-called *Onkyo* improvised music scene (David Novak 2010), started an iconic series of releases dedicated to the instrument, which he termed "nimb", acronym for "no-input mixing board". The term "no-input" refers to the fact that the mixer doesn't use any other inputs than its own outputs, and although audio effect units can be inserted in those feedback chains, it constitues a closed feedback system.

In such configuration, traditional mixer controls radically change their affect on perceived sound qualities, both by acquiring non-linear effects, i.e. a continuous variation of one parameter can result in a discontinuous variation of the percept; and by affecting sound qualities that weren't originally connected to their designed function, e.g. controlling the gain of the eq's low frequency band typically affects rhythmic qualities of the sound being produced. Furthermore, all controls become mutually dependent, with some controls affecting different perceived qualities depending on the state of some or all of the other controls, and also depending on the previous state of the whole system.

Such a complex, unpredictable, and non-linear interaction space is not unique to no-input mixers, and can be considered an *exaggeration* of the non-linear dynamic processes of playing traditional acoustic instruments (Mudd, Holland, and Mulholland 2019; Fletcher 1999). In a recent study, Mudd (Mudd 2023) collects reports of how musicians find this approach valuable, also underlying a fundamental difference with most digital music instruments, which offer simpler and more understandable interfaces, but can "*encourage an overly analytic approach to creative practice*". In the study, Mudd highlights three aspects of this appeal: (1) positive sentiment attached to the surprising and unpredictable results of working with a process (and an interface) that can't be fully understood, also after years of practice and even after attempts to systematic analysis; (2) perceived immediacy of interaction, due to its fast responsiveness, great sensitivity to parameter changes, and a sense of directness in working with "raw" materials and processes, as opposed to the experience with digital instruments; (3) a favourable comparison to the versatility and

discoverability of acoustic instruments, their chaotic elements, which also offer room for reinvention through exploiting their more or less stable states.

The present work aims at creating a hybrid instrument, where the sound processing is analog, resembling a traditional no-input mixer configuration, thus aiming to maintan the "rawness" of its sound character; but where the controls are digital, to allow novel explorations of its complex, non-trivial interaction space as afforded by the increased resolution, memorization, and interface prototyping options offered by digital control. It situates itself at a middle point in the duality between analog and digital instruments discussed by Mudd.

2.2. From co-creation to "unmastery" and "complexity literacy"

Other than its sonic qualities (and perhaps more), the most prominent value of playing no-input is the exaggeration of a "co-creative" interaction, encouraging inclusion of the instrument itself within the creative process, thus off-putting an "autocratic" commanding approach by which a performer seeks to transmit their idealized musical intention "despite" the obstacles posed by the instrument.

In an interview (Meyer and Nakamura 2003), Toshimaru Nakamura describes co-creative musicking as a determining factor for switching from playing guitar to no-input mixer: «*I think I find an equal relationship with no-input mixing board, which I didn't see with the guitar. When I played the guitar, 'I' had to play the guitar. But with the mixing board, the machine would play me and the music would play the other two, and I would do something or maybe nothing. I would think some people would play the guitar and create their music with this kind of attitude, but for me, no-input mixing board gives me this equal relationship between the music, including the space, the instrument, and me.*».

Regarding no-input mixers, Mudd identifies their "explorable unpredictability" as a condition for co-creative approach: *"When something unexpected occurs, it can usually be engaged with and explored further. For example, an unpredictable outcome may involve the discovery of a new metastable state that an artist can probe and experiment with, engaging with new sounds and behaviours, and learning about an aspect of their instrument."*. The instrument has to offer a degree of unpredictability, but it should also afford its "further exploration", an "engagement", a continuous learning experience that deepens the relationship between musician and instrument, perhaps never reaching a point where it reduces to command and execution.

In a study of contemporary approaches to feedback instruments design and performance, Magnusson et. al (Magnusson, Kiefer, and Ulfarsson 2022) centralize the notion of "playing with the instrument" instead of "playing on the instrument", signaling departure from a modernist concept of authorship towards a post-modern decentralization of the human self within distributed agency ecologies. Rejection of authorship and virtuosity are sustained by design practices aimed at increasing the instruments' agency and autonomy, as perceived by the performers, to afford active engagement in a dialogic process of continuous discovery, "re-skilling" and mutual adaptation, rather than mastery and command.

Melbye (Melbye 2022) further individuates co-creativity as being typical of feedback based musical interaction (thus including experimental music practices with acoustic instruments), where *"the apparatus affords a way of knowing this work, not as something existing prior*

to—but instead as unfolding through knowing [...] As I have described elsewhere, the precariousness of this asymmetrical relationship between performer and instrument affords a level of improvisatory music-making mostly found in social musical practices involving other human performers, and as such, raises questions of distributed agency [...] Asymmetry, manifesting as a sense of resistance from the performer's point of view (Melbye 2021; Stapleton 2008) is a way of the material making itself known: rather than the idealised interface for musical expression, matter manifests itself through irreversible processes that don't just shape, but co-constitute what is being said, in a process that may sometimes feel like a welcome complexification of performative intent and at other times a bewildering obstruction". He proceeds to define his concept of "unmastery", not as merely the rejection of mastery, but as a practice of "nurturing the asymmetry", i.e. cultivating relationship through resistance, instead of seeking to eliminate it, suggesting that *"a feedback practice that profoundly engages with the material, through a practice of unmastery and vulnerable engagement (ibid. 91), may offer ways of making music that challenges human privilege and acknowledges relational asymmetry as a condition to be explored, rather than an impediment to be neutralised. Unmastery, then, becomes a practice of nurturing that asymmetry".*

Melbye criticizes the idealization of musical instrument interfaces similarly to the divide described by Mudd when asserting that digital instrument are akin to overly analytical approaches. The present work includes a no-input mixer circuit as a conceptual ready-made, conserving its quirks, identity and resistances, thus avoiding to design a completely new sonic process from scratch. By superposing a digital interface it adds a degree of abstraction, a distance that changes the asymmetry and the resistances of the instruments, not with the intent of removing them, but rather to afford a different access to their exploration.

Eldridge (A. C. Eldridge 2022) goes even further and values the experience of designing and performing with feedback instruments as building "complexity literacy", suggesting that it's precisely the resistance to imposition fostered by such design practices, that supports a growing understanding of our contemporary complex milieau. Musicking with feedback instruments, i.e. their design, performance and experimentation, embeds a particular way of knowing systematically, performatively and intuitively, continuously through the process. The experience of agency in systems with emergent behaviors, the necessity of learning to "let others be" when working with self-determined systems (i.e. systems that are influenced but not completely controlled by their environment), are essential elements for building world views that help us understand our being-in-the-world in terms of its complexity. Such practices can be playgrounds to experimentally reverse a dominant approach to technology and the world, our misindentification as autocratic controllers of othered beings and processes, thus *"supporting us through current existential, ecological, technological and social crises".* Eldridge's analysis interprets a recent increase of activity in an emerging research area focused on feedback musicking, which also consitutes the main context of the present work.

2.3. Self-resonating vibro-tactile instruments

Feedback musicking, instrument and interface building and performance, has a long history spanning the domains of analog and digital instruments, as well as contemporary composition and popular music (Valle and Sanfilippo 2012; Sanfilippo and Valle 2013; Collins 2020; Di Scipio 2003; Bowers and Haas 2014). However, as perhaps explained by Eldridge, the field has seen a renewed interest in recent years, with the systematization

of a hybrid lutherie approach and its conceptualization under the term "Self-resonating vibro-tactile instruments".

The "Feedback Musicianship Network" ("Feedback Musicianship Network" 2021) is an international joint effort bringing together artists and researchers to support the creation of both new feedback instruments and tools for their understanding. In this context, Eldridge et. al (A. Eldridge et al. 2021) proposed the concept of "Self-resonating vibro-tactile feedback instruments" (SRIs), to *"unpack the emerging trends in lutherie and artistic practice, secondly to reflect on the experience of playing these instruments, and finally, to expound a conceptual framework to scaffold future work"*.

Neither no-input mixers nor the present work can be strictly categorized as SRIs, because they lack the vibro-tactile interface that is constitutive to them. SRIs rely on feedback between a pick-up and an actuator, connected by physical matter (e.g. strings, springs, plates) so to be directly sensitive to physical interaction. In no-input mixers such physical connection is established between electronic components in the circuit, i.e. the inputs and outputs of amplifiers, where physical interaction is mediated by potentiometers affecting their electrical characteristics. Although the previously cited work by Mudd highlights "rawness", "sensitivity" and "directness" of interaction perceived by no-input mixer players, adding digital control amounts to an extra level of mediation. However, the present work situates close to SRIs because like SRIs is "a new species of feedback instruments in which designers and luthiers consciously and deliberately incorporate feedback loops as a central design principle to create self-resonance", and because it inherits the "rawness" and sensitivity of a no-input mixer. It can be argued that although the performer can't put their hands physically in the feedback chain, the embodiement afforded by no-input mixer controls is fundamentally different but experientially kin, resulting in an immersively engaging experience due to the sensitivity, reactivity and non-linearity of its feedback process. Even though a digitally controlled no-input mixer lacks the vibro-tactile directness of a strict SRI, it can be argued that its design process is conceptually similar to SRIs, where an existing instrument is initially augmented with a feedback apparatus, and then a new instrument is built from scratch.

2.4. Comparison to other related instruments

Similarly to how the FAAB (Melbye and Úlfarsson 2020) is a SRI built from a double-bass, a digitally controllable no-input mixer is built on a traditional audio mixer, where the feedback mechanism originally superimposed by no-input wiring is now embedded in the design. Another similarity with the FAAB's design process is the relation between the designer and the chosen base instrument: Melbye has a long history playing the double-bass and thus choses it to become the base for his feedback instrument, in a similar way to how I relate to no-input mixers, and even more specifically no-input mixers than traditional audio mixers. Namkamura (Trapani and Namakura 2017) relates his "invention" of the no-input mixer as coming from subverting his practice from his job as an audio engineer. The present work stems from my experience playing no-input mixers for years, in experimental musical contexts such as contemporary european improvised music and harsh noise [1] [2] [3].

[1] https://nofigore.bandcamp.com/album/dance-with-my-mother
[2] https://nofigore.bandcamp.com/album/autoeater
[3] https://nofigore.bandcamp.com/album/boys-cry

Two opposite directions, closer and further away from SRIs, are represented by Shepardson and Skach's "No-input textiles" ("Victor and Sophie's No-Input Textiles | Intelligent Instruments Lab" 2023) and "Bendit.io" (Marasco, Berdahl, and Allison 2019). "No-input textiles" is a project that couples a no-input mixer with a conductive textile interface. While this can actually be considered more of a SRI because of its tactile interface, the interaction field opened by its design is more focused on the affordances of textile interfaces, while the present work aims to focus on the no-input mixer itself. On the other hand "Bendit.io" proposes a device to digitally control circuit bending (Ghazala 2005) of other devices. As such, it's closely related to the present work, as both seek to add digital control to analog sound processing devices. The present work could also have been implemented like that, i.e. a control device to be coupled with a mixer, but on one hand it would have been unpractical, due to the fragility of a setup that requires disassembling and circuit bending a mixer, especially outside of the lab environment and through travelling and performance practice; on the other hand the cost of producing a standalone digitally controlled no-input mixer is only marginally higher than producing only its control interface, digital components being the most expensive. It was thus preferred to produce a self-contained instrument, where an existing device was being reinvented with a feedback system at its core, resulting in a closer relationship to SRIs.

Last, I have previously published a work (Elia and Overholt 2021) for the Feedback Musicianship Network, proposing a feedback instrument called Squidback: a Larsen generator with an adaptive filter, implemented as a web application. Larsen effect is obtained acoustically between the host device's speakers and microphone, but no control interface is provided, thus encouraging physical interaction as the only way to affect sound, despite the audio process being completely digital and even allowing for remote collaborative performances. Squidback can thus being considered more of a SRI, because of its vibro-tactile interface, although it doesn't include any physical instrument building. It can be thought as a feedback-centered reinvention of modern telecommunication practices (web-based remote conferences), which was particularly relevant at the moment of its creation being during pandemics. It also exhibits emergent behaviours, both in "single-player" mode, between the adaptive filter and room acoustics, and in "collaborative" mode, where audio is shared among a network of remote participants. However, it resulted to be closer to a participative installation than to a musical instrument, affording more of an aesthetic experience on its own, very focused on its own character and dynamics, and lacking the versatility needed to be an engaging instrument for musical performance.

3. Hardware design

The concept for this instrument's hardware design is to imitate a simple no-input mixer setup (fig. 3.1), hardwiring feedback connections in the circuit itself, and adding digital control for each gain in the signal path. Following the approach described in Design Note 02 by THAT corp. (corp 1999), digital control is implemented combining high-resolution Digital-Analog Converters and Voltage Controlled Amplifiers, achieving more resolution and less cost than implementations based on digitally controllable potentiometers. For the prototyping stage to benefit from more modularity, which was critical to reduce costs and allow experimentation with reusable parts, the elements implementing digital control are designed and produced separately from the mixer circuit. Hardware is thus composed of two designed circuits: a "digitally controllable amplifier" module, and the mixer "motherboard".

The next sections presents these parts and their design, with simplified block diagrams, while detailed schematics are included in sec. A.2.

Figure 3.1.: A simple no-input mixer setup: channel 1 is in feedback through the AUX send, channel 2 through the FX send, channel 3 and 4 get the main outputs, while sound is sent to speakers from the headphones out. These feedback connections are embedded at the circuit level on our mixer.

3.1. Digitally controllable amplifier module

This reusable element puts together a 16-bit 4-channels DAC with a 4-channels VCA. The DAC is addressable from a microcontroller through SPI, making it easy to control more devices at the same time dedicating different pins to the Chip-Select function for different devices, and otherwise sharing the same clock and data lines. Signal from the DAC is scaled and shifted to the control range expected by the VCA, to access its full range from -100dB to +20dB. Although the chosen VCAs are current amplifiers, it was chosen to not include neither the RC networks necessary at their inputs, nor the current-to-voltage converters at their outputs, since VCAs' inputs and outputs can be connected in different ways at different stages on the mixer circuit.

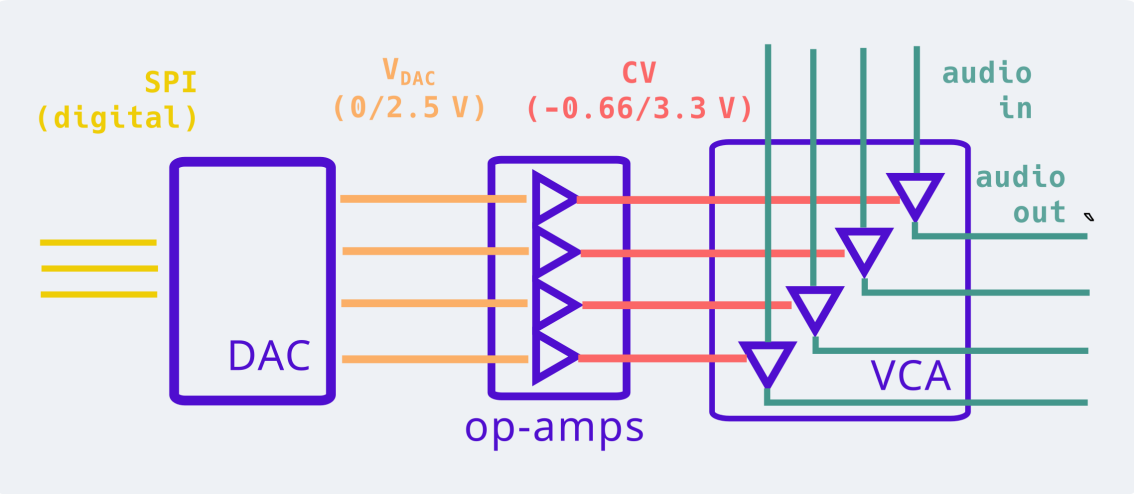

Figure 3.2.: DCA circuit block diagram

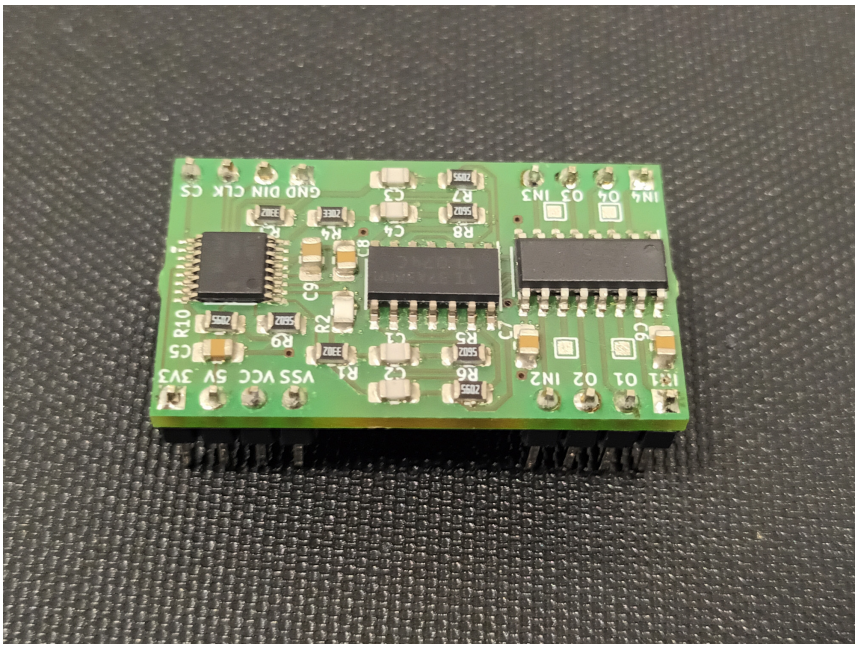

Figure 3.3.: DCA module assembled

DAC models were chosen for their high DC accuracy, and VCA models for their wide availability. Scaling and shifting DAC outputs to VCA control inputs is done through op-amps in differential amplifier configuration.

Designing a 4 channel module was a reasonable choice, since VCAs are available in maximum 4-channel packages, thus achieving an acceptable balance between cost and flexibility: a higher number of 2-channel devices would have a bigger footprint and more components, while a lower number of 8-channel devices would too often mean a waste of channels (for projects not working with a multiple of 8). The current mixer design requires 12 amplifiers, and thus uses 3 of these modules, but earlier prototype ideas were supposed to work with 16 for example, as mixers can be designed with different connection topologies and different numbers of audio channels. Even though at this stage only one mixer prototype has been entirely built (with another one still in development), having a reusable element was judged positive to allow further development and cut the costs of the most expensive components used.

3.2. Mixer circuit

The mixer circuit implements a 2-channel no-input topology, with every channel feeding back into itself and the other (with adjustable input gain), and their sum also feeding back into each of them. This "sum feedback" is akin to connecting the main output of a traditional mixer back into a channel, while the individual channels' feedback connection are normally acheived through auxialiry sends. Differently from most common practice with mixers, channels are kept separate at the output, with channel one on the left and channel two on the right, and no panning options available. Switchable analog inserts are provided before the main output, to be able to extend individual feedback chains through external devices: the mechanism uses SPDT switched jacks, to break feedback chains only when plugs are inserted.

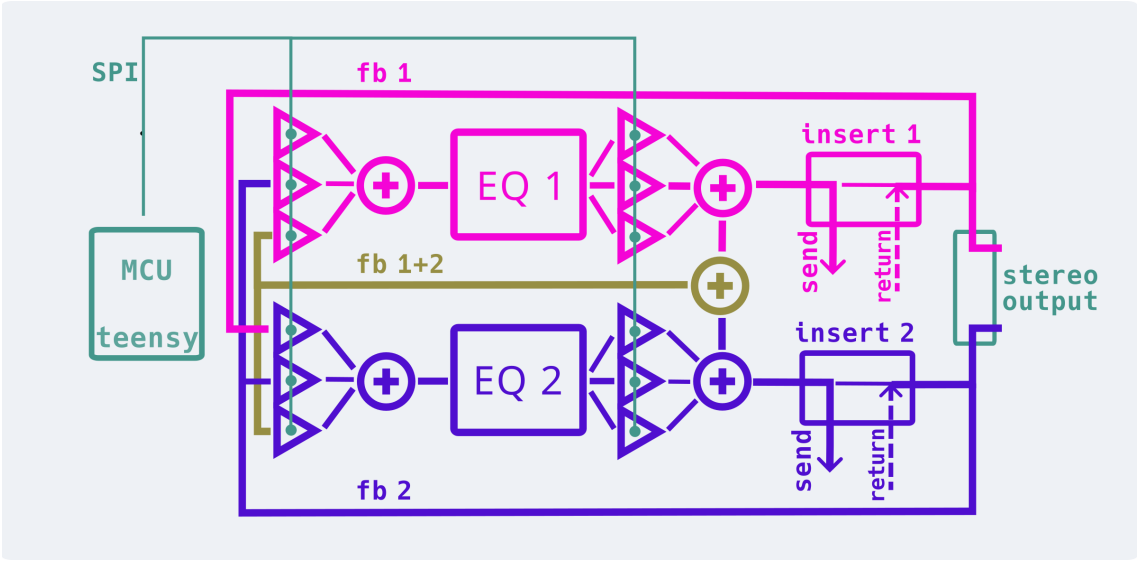

Figure 3.4.: Mixer circuit block diagram

Equalizers are designed as simple passive filters, whose output is then amplified or attenuated through VCAs. Each channel has three fixed frequency bands, mimicking the equalizers commonly available on simple audio mixers.

The circuit also includes sockets for the digitally controllable amplifier modules described above, and for the microcontroller. It was chosen to work with a Teensy 4.0, for it provides an acceptable balance between costs, availability and easeness of programming, and not less importantly for the supportive open-source community around it.

Power supply is provided through USB, and converted on the circuit to ±12V for all active components, while a voltage regulator provides 5V to DACs.

It was choosen not to include any physical control interface, to avoid fixing control paradigms to the hardware, letting a more flexible research on interface happen at software design stage first, making use of MIDI controllers via the host computer. The only control port is thus Teensy's USB, which is used to exchange OSC messages, as described in sec. 4.

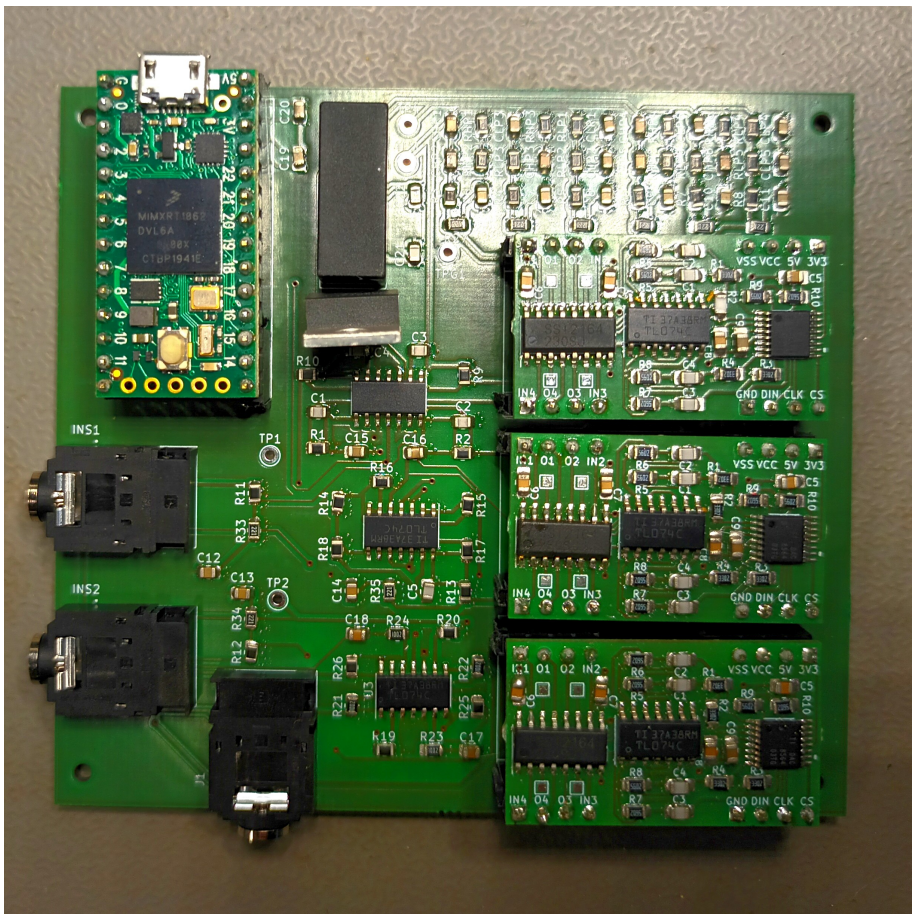

Figure 3.5.: Mixer board assembled

3.3. Prototyping and fabrication notes

All circuits were designed, simulated, tested on breadboards and then produced as PCBs and assembled. All stages except PCB production (which was commissioned to a company) were performed at the CREATE electronic lab in Aalborg University Copenhagen. A case for the first prototype was fabricated as a laser cut acrylic finger-joint box. The PCB is screwed to its base.

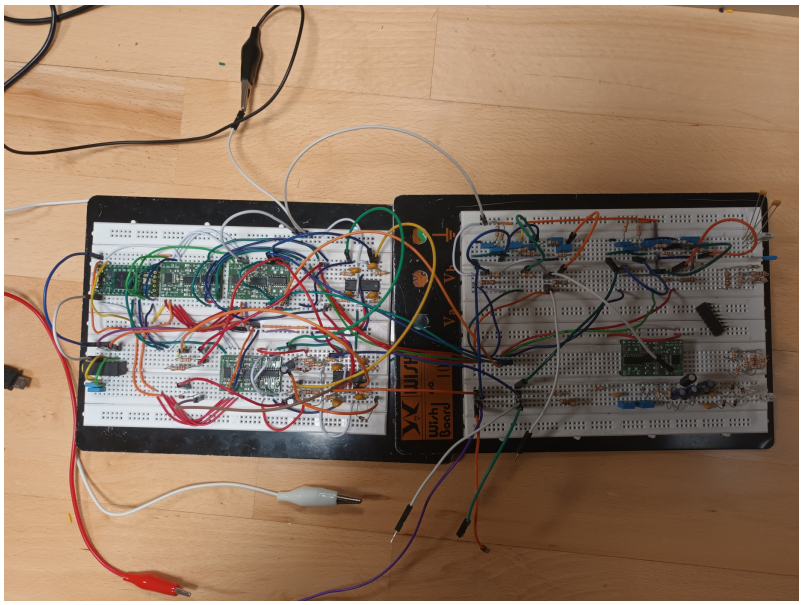

Figure 3.6.: Prototyping the mixer circuit on breadboards

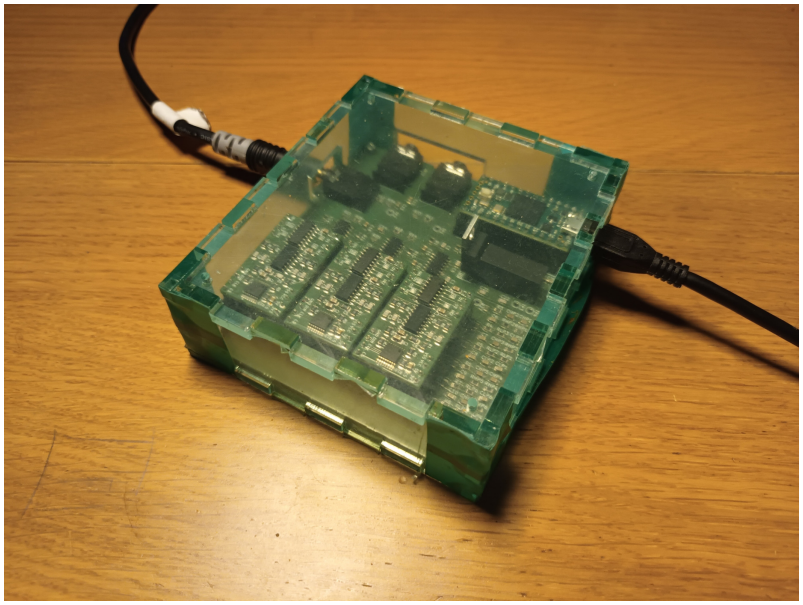

Figure 3.7.: First prototype in its case

A second prototype is currently under development, featuring USB audio interfacing for easier monitoring and switchable digital inserts, active equalizer filters, and an improved power supply system.

4. Software design

Overall software design is divided in two areas: microcontroller firmware, directly controlling DACs through SPI, and the interface on the host computer, implemented in SuperCollider, and communicating with the microcontroller through OSC. Between the two, an additional piece of software was needed to proxy OSC communication between USB and local network, not because the system is to be run over a network, but because most software accepting OSC expects in on a network device. This structure is meant to leave the maximum flexibility when developing the interface. Embedding sophisticated interfaces in the microcontroller (like embedding control devices on the hardware) was judged to be too detrimental, since interface development is supposed to be a continuous process, part of the relationship with the instrument, rather than aiming to finalize a product. This is one of the main affordances of digital instruments: to lend themselves more easily to continuous development, which at the same time comes with the risk of suggesting an "overly analytical" approach to musicking.

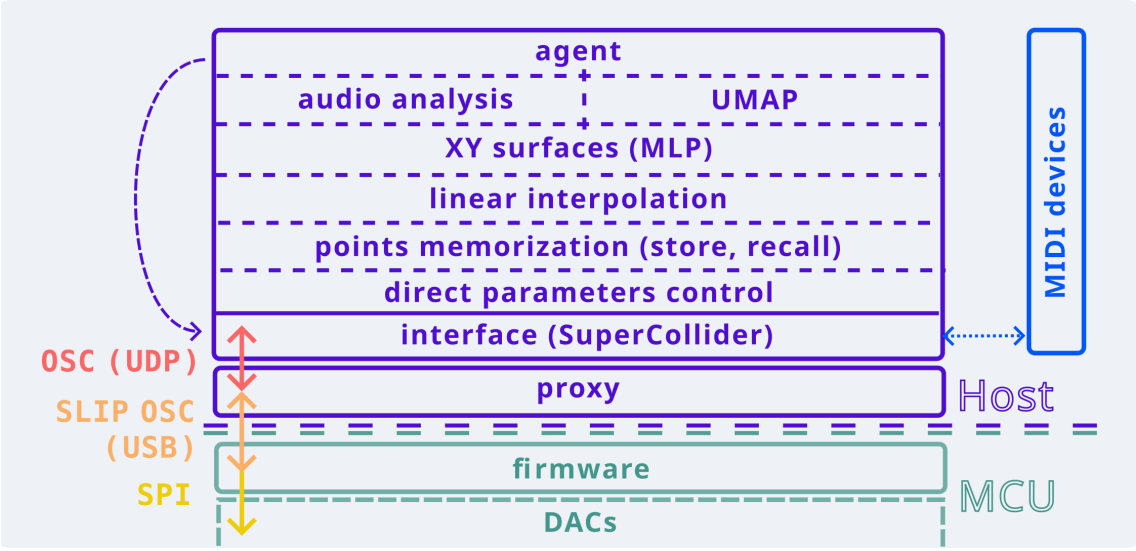

Figure 4.1.: Software interface elements, arranged to show their progressive detachment from direct parameter control: higher elements are built on top of lower ones, introducing more and more layers of mediation.

Software design is addressed in a progressive fashion: from microcontroller firmware (i.e. the closest to the analog circuit) to the digital control interface, gradually integrating elements of mediation and thus distance from direct control, up to a proof-of-concept of an autonomous agent playing the instrument.

The primary goal is to implement a basic interface to play the instrument, offering the basic affordances offered by digital control, i.e. access to variable resolution and memorization. This is intended to produce a playable instrument and an infrastracture to continue its

development. Higher degrees of mediation such as autonomous and cybernetic devices are for now implemented as proofs-of-concept, meant to suggest their feasibility, and to open research directions to be developed as the focus of future dedicated research.

4.1. Firmware

The only function of software running on the Teensy is to allow DAC output values to be controlled from a computer, via the OpenSoundControl protocol (Wright and Freed 1997). OSC is chosen because of its simplicity and flexibility for prototyping, compared to a custom serial interface. At the cost of an arguably unnecesary overhead, it allows to quickly iterate a communication protocol, testing it and implementing new functions during development. Despite this application being extremely simple, it is often necessary to have for example an echo message for testing the device, or equally often the project develops later requiring more complex communication, which is easy to develop incrementally with OSC.

Although OSC is primarily transmitted over network interfaces, it was chosen to avoid wireless networking, beause all control interfaces are on the host computer, and also because the instrument needs USB power anyways. USB networking was considered an unnecesary complication, and it was chosen to transmit OSC over serial. This comes however with its own small complication, which required developing an utility described below (sec. 4.2).

The firmware is kept simple, providing interfaces for assigning one or all dac values at one time. No interpolation is performed at this level, leaving full control over DAC settings to the host computer, and DACs provide enough resolution to perform smooth transitions directly at the digital level. The only extra commodity is that the system caches DAC values and avoids sending redundant messages if there is no difference between the cached current state and the one being assigned. It does so for every DAC channel separately, e.g. if the user is setting all values, but only two differs from cache, only those two messages will be sent over SPI to the relevant DACs. This is done to reduce noise on digital lines.

Firmware is written in C++ using the Teensyduino toolchain, and open-source libraries for OSC communication. As a final note, a second version of the firware was written for a second hardware prototype, adding support for USB audio I/O through the Teensy Audio Shield, which required a few minor adjustments to be made to both OSC and Audio libraries for Teensy (see sec. A.1). It also featured an interface for driving a relay coil, which switches digital DSP on and off (akin to the switch on audio jacks for analog inserts). Even more when developing these new features, both OSC and the Teensy ecosystem were confirmed as comfortable choices.

4.2. SLIP encoded OSC serial proxy

OSC communication is not directly possible over a serial transport such as USB. It is however supported through a simple framing protocol called SLIP, as described by CNMAT[1] and specified by the internet standard RFC 1055 (Romkey 1988). The framing consists in

[1] https://cnmat.berkeley.edu/content/slip-encoding-and-decoding

transmitting special escape sequences before and after each OSC packet, to supply for the need of a "packetized" transport required by OSC being a datagram protocol[2].

Although facilities for SLIP encoding on Arduino side are provided by CNMAT in their OSC library[3], such facilities are not part of computer music platforms such as SuperCollider. In particular, it would be ideal to have SLIP encoding facilities directly available in SuperCollider, but since such feature should not run on the audio thread, it would have to be an extension of SuperCollider's language client, *sclang*. The platform currently supports only *scsynth* extensions, i.e. audio processing plugins, so the feature could only be implemented as an official part of *sclang* itself. Even if I'm a contributor to the SuperCollider codebase, working on both sclang and scsynth in the past[4], the process of coding, testing and merging such an extension would be lenghty and outside the scope of the current work. For this reason, it was preferred to develop an external command line utility to act as a proxy between SLIP encoded OSC communication over serial interfaces, and "regular" OSC over UDP on the loopback "localhost" network interface. This allows SuperCollider (and other computer music platforms) to send and receive OSC messages to and from USB devices transparently using their native interfaces.

The program is called *oscslip-proxy*, and it features bi-directional communication, encoding outgoing messages and decoding incoming ones, automatic reconnection in case of temporary disconnection (e.g. if the USB cable is accidentally unplugged) and optionally printing all messages to the console for debugging. It's written in Python and published online as a standalone at https://github.com/elgiano/oscslip-proxy. It is also available on the Python Package Index via "pip install oscslip-proxy"[5].

A similar approach is pursued by Monome with their program *serialosc*[6]. However, serialosc translates from a proprietary, device-specific serial protocol to OSC, while oscslip-proxy just proxies OSC messages so that device firmware design can benefit of more flexible and transparent interface as offered by OSC.

4.3. Interface

The interface for interacting with the instrument is written in SuperCollider, which offers adequate facilities for quick iterations and gradual implementation through practice. Particularly relevant are the easeness of modifying software while it's running, it's coding interface which sets it apart from more visual oriented platforms, and the work contributed by a large community, e.g. facilities for interfacing with OSC and MIDI controllers[7], integration with machine listening/learning tools[8], and some of my own tools and libraries which I've been developing through the years.

Design and implementation happened as a gradual introduction of layers of mediation: first an interface for directly controlling parameters, then for saving and recalling control points, then for interpolating between them, and finally analysis: first for mapping to new

[2] https://cnmat.berkeley.edu/content/osc-over-usb-serial-transport
[3] https://github.com/CNMAT/OSC
[4] A list of merged pull-requests by the author for the SuperCollider project: https://github.com/supercollider/supercollider/pulls?q=is%3Apr+author%3Aelgiano+is%3Aclosed
[5] https://pypi.org/project/oscslip-proxy/
[6] https://monome.org/docs/serialosc/
[7] Modality Toolkit: https://github.com/ModalityTeam/Modality-toolkit
[8] FluCoMa: https://www.flucoma.org

parameter spaces for human interaction, and then for a semi-autonomos agent to navigate them.

4.3.1. MIDI controllers

From the start, to allow for a level of direct, physical interaction, the interface was built with two MIDI devices in mind:

- MIDI Fighter Twister[9]: a 4x4 grid of velocity sensitive encoders, chosen primarily to work directly with the mixer's parameters and afford variable resolution.
- Monome Grid[10]: a 8x8 buttons grid, to work with 2D space and afford button-like experience which is mostly missing on analog no-input mixers.

The choice of encoders over knobs or faders has the advantage of allowing for dynamic resolution and mirroring of an interface state that can simultaneously be affected by other agents in code. In other words, it is possible first of all to change the mapping (resolution, range, value) and to give the user a visual feedback of the current state of parameters directly on the control device, even while those parameters are changed by the program, and as such would cause a discrepancy with the position of traditional, un-motorized, knobs and faders. However, encoders lack tactile feedback of the current state of parameters, most notably at the extremes of their range, where a traditional knob would stop moving. Visual feedback becomes then more important, and it was chosen to give it both directly on the control device and on a software GUI. Mapping and color coding of the encoders is programmed to match the GUI.

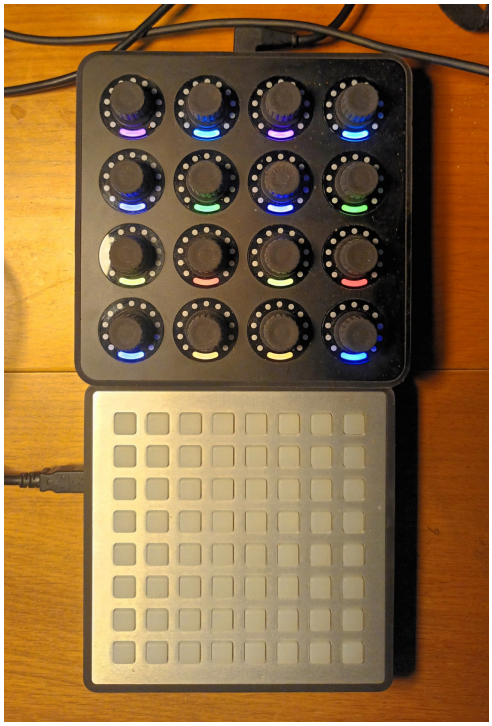

Figure 4.2.: MIDI devices: MF Twister (top), and Monome grid (bottom)

[9] https://store.djtechtools.com/products/midi-fighter-twister
[10] https://monome.org/docs/grid/

4.3.2. GUI overview

The GUI presents visualizations and control widgets for all features explained in the next sections, as summarized in tbl. 4.1. Oscilloscope views are meant to visually monitor the instrument's output signal, both in time domain and most importantly in phase space, to help forming a visual intuition of the system's complex behavior while playing. Single-channel phase space (number 8 in fig. 4.3} is displayed using a user-controllable time-delay td so that graphs are plotting $y(t-td) = f[y(t)]$. The single phase-space plot at the bottom (number 9 in fig. 4.3) is plotting the "stereo" relationship $ch2 = f(ch1)$.

Figure 4.3.: Main GUI, annotated in red

Table 4.1.: GUI picture annotations

Num	description
1	Point store/recall view (sec. 4.3.4)
2	Linear interpolation controls (sec. 4.3.7)
3	Direct control (displayed like points)
4	Direct control knobs (sec. 4.3.3)
5	UMAP analyzer (sec. 4.3.9)
6	MLP XY surface controls (sec. 4.3.8)
7	Oscilloscope, time-domain signals
8	Oscilloscope, time-delay phase plots for each channel
9	XY phase plot (stereo)

4.3.3. Direct control of parameters: affording resolution

The closest level of interaction with the instrument is afforded by directly changing its parameters. Here the goal is to facilitate access to the resolution available through digital control, where the problem is to map physical action to a big range of values (16-bits), compromising between fine-grained control and mobility throughout the range. An example mapping is given in eq. 4.1, where ν is a velocity sensitive increment provided by the encoder device (i.e. 17 steps between 0 and 1 according to how fast the encoder is being turned), and δ is the resulting increment as an integer value for DACs. This exponential curve, which was tuned empirically, affords a maximum resolution of 2 (a fine-grained ~0.004 dB) when the encoder is turned slowly, up to an increment of 200 (a faster moving ~0.4dB) when turned fast.

$$\delta = sign(\nu) * [\frac{1 - e^{4*abs(\nu)}}{1 - e^4}(200 - 2) + 2] \tag{4.1}$$

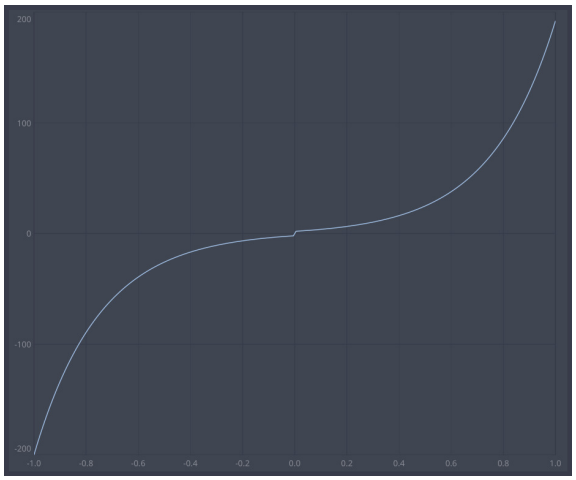

Figure 4.4.: Velocity sensitive mapping: plot of eq. 4.1 which is a simplification of SuperCollider's "lincurve" function for specific parameters.

Resulting increment δ is then added to the current parameter value. Absolute values are constrained to an adjustable control range, so that DACs full range $[0, 65535]$ ($\sim[-100dB, +25dB]$) can be tuned to focus on a more specific region of interest, e.g. $[-20dB, +20dB]$. Visual feedback about the relative value of parameters mapped to their control range is provided both on the MIDI device and on the GUI. Velocity curve, sensitivity and control range can be changed while playing, although currently only by writing and evaluating code.

To provide even more mobility, e.g. to quickly turn a knob from its minimum to its maximum, a faster and more coarse mapping can be accessed by pushing the knob while turning it. This is ment to allow fast and extreme action, as can be performed on analog knobs.

4.3.4. Points: saving and recalling parameters' state

When playing a traditional no-input mixer, it's increasingly hard to manipulate more parameters at the same time. Furthermore, other than for any buttons eventually on

the mixer, manipulation is always a continuous transition between values, as afforded by analog potentiometers. Saving and recalling state allow to quickly jump between points far apart in parameter space, changing all parameter values at once, aside from the obvious memorization options.

Points in parameter space are the first abstraction poised by the interface: playing no-input becomes a matter of visiting points in its parameter space. This abstraction has a fundamental flaw as it doesn't take into account the non-triviality of the instrument, i.e. the dependency of the output on the previous state of the system, so that jumping to the same destination from different starting points can produce different results. However, as a manifestation of the instrument's resistance to be framed in this paradigm, it can be considered positive from the co-creative point of view, as it adds unpredictability that can be engaged with in its non-arbitrarity, as long as a relationship to the lowest-level of abstraction and to sound is still observed.

The interface offers to store and recall banks of 16 points, which can also be written or read from disk. Keyboard shortcuts (and code interface) are provided to quickly save and recall them. It was attempted to use the encoder switches for this, but it was preferred to reserve them to direct parameter manipulation, to always have it as a reference on the encoders, even when travelling across saved points using another interface.

Two visualizations were implemented to represent the 12-dimensional value of a point: one as a bar graph; the other as a set of 6 points in 2-dimensional space, where x and y coordinate of each point are the values of the same parameter for the 2 mixer channels. Color coding for each parameter matches the rest of the interface.

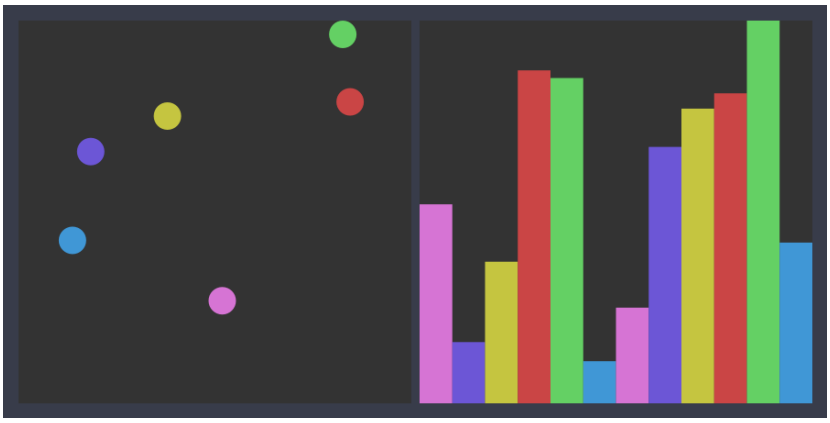

Figure 4.5.: 2d and bar graph visualizations of a point in parameter space

The 2d view (on the left in fig. 4.5), is considered a more identifiable representation, while also offering a more intuitive intuition of the distribution of values between the two channels. When used as a control interface, it has less movable elements and thus offers a significantly different approach from adjusting every single knob.

4.3.5. Random variations

Random variations around the current parameter state are accessible by a *vary* function, which adds a random increment limited by a variation amount parameter. The function normally doesn't allow wrapping, i.e. dimensions which after variations would result in

values outside the control range (explained in sec. 4.3.3) are clipped to it. However, wrapping is allowed when the *amount* parameter exceeds a certain value (hardcoded to 50 dB). This is intended to make the function able to output random points in parameter space, not related to the current state.

"Vary" is mapped to the rightmost knob of the bottom row of encoder: the encoder value controls the variation amount and switch applies it. The dynamic wrapping mechanism makes an extra "random" button unnecessary since the function is already available through *vary*.

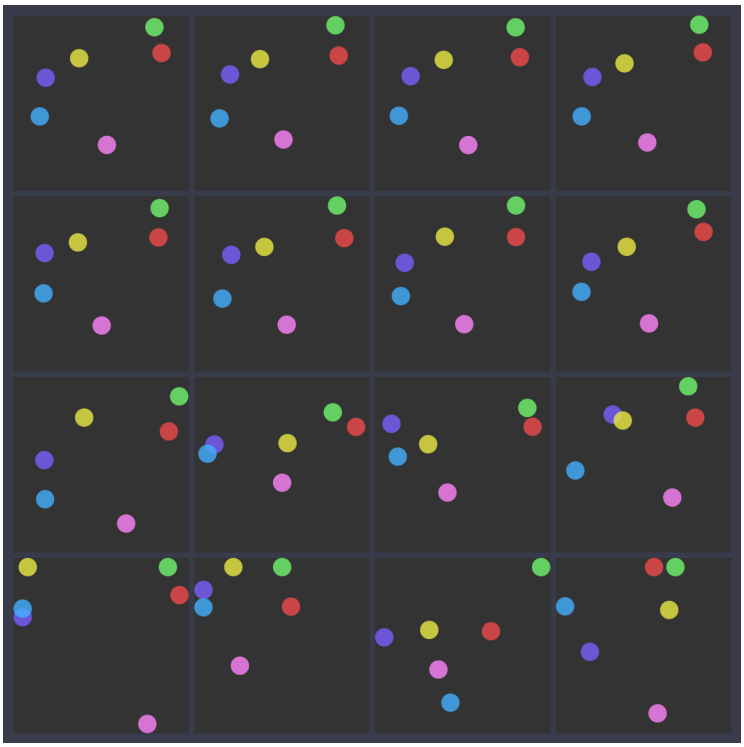

Figure 4.6.: Examples of variations obtained by *vary* from the point displayed in fig. 4.5. Each row has a different *amount* parameter, from top to bottom: 1dB, 2dB, 10dB, 20dB. Note that variations as small as 0.01dB are usually perceivable in sound, but would not be distinguishable on the visualization.

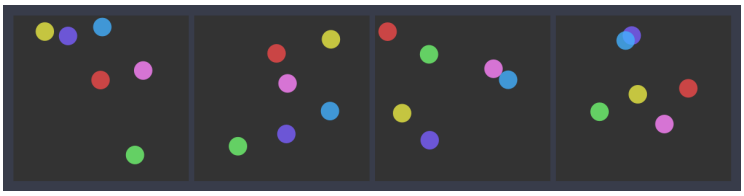

Figure 4.7.: Four random points obtained by *vary* with amount = 100dB (wrapping enabled)

4.3.6. Navigation history: no-input mixer "undo" feature

A history mechanism is provided to navigate back to previously visited points, like an "undo" function. The list is maintained simple, with no bifurcations, meaning that if a point P at position in history i_P ($P = H[i_P]$) is recalled, and then from there a new point

is saved, all the history points after P will be discarded before the new point is added. Navigating history is not itself recorded in history, but other than this, current state is saved to history every time a new value is set, with configurable debounce and throttle times.

Current position in history and its length can be monitored on the GUI, and back and forward functions are mapped to side-keys of the encoders device.

4.3.7. Linear interpolation: from points to lines

A simple linear interpolation mechanism provided to navigate between points is shown in lst. 4.3.7. Note that this function accepts a destination point, but sets as starting point the current parameter state at each iteration. This is intended to keep it sensitive to manual perturbations, which would automatically change the interpolation path, since only a small step is calculated at every iteration. The maximum displacement allowed at every iteration is controlled by *lerpSpeed*, which is mapped to the leftmost knob on the bottom row of encoders. The switch on that encoder simply stops the interpolation, leaving the system at whatever state it is at that moment, useful to stop a navigation halfways in case an interesting sound is reached.

```
~lerp = { |self, destination, saveToHistory = true|
    if (saveToHistory) { self.debounceHistoryAdd };
    Tdef(\lerp) {
        var diff, current;
        while {
            current = self.rawValues;
            diff = destination - current;
            diff.abs.sum > 0
        } {
            // convert from dB/s to increment
            var maxIncrement = self.incDbInt(self.lerpSpeed) * dt;
            // don't save to history during interpolation
            self.setValues(current + diff.clip2(maxIncrement), false);
            0.01.wait;
        }
    }.play
}
```

Since points are saved disregarding any control range mapping, linear interpolation also operates on *rawValues* (i.e. integers in DACs range). Also note that current state is saved to history only when interpolation starts, and never along the path, not to pollute the history list with too many points. However, if the user is traveling towards a point and applies any perturbation (e.g. changing any parameter or triggering interpolation towards another destination), the current state prior to such perturbation is saved.

Interpolation becomes then a tool to trigger multi-parameter gestures, by using saved points as directions towards which to move, and being able to stop or change direction at any moment (see fig. 4.8 for examples).

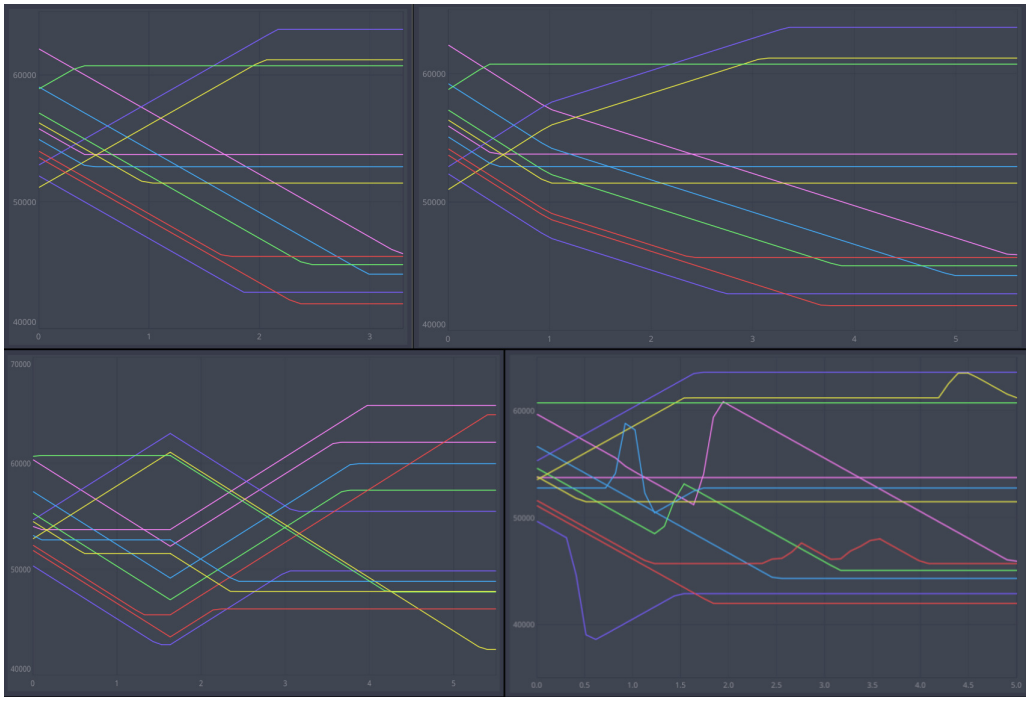

Figure 4.8.: Four examples of linear interpolation between two saved points. Top-left: constant speed (10db/s); top-right: speed is changed from 10db/s to 5db/s after 1s; bottom-left: constant speed, but destination is changed halfways, bottom-right: user applied perturbations by directly changing mixer parameters

4.3.8. XY Maps: organizing points as surfaces with MLP

The multiplicity of trajectories in 12-d spaces afforded by linear interpolation suggests a possibility for further organization. Similarly to how points define directions for multi-dimensional gestures, we can define a surface of interpolated trajectories between a number of points by distributing them on a lower dimensional projection and derive a non-linear mapping between the two.

This interpolation technique, intended to afford lower-dimensional data-specific control of higher-dimensional parameter spaces (e.g. xy controls for synthesizers with tens of parameters), was recently popularized within computer music communities by the FluCoMa project (Moore 2022; Green, Tremblay, and Roma 2018). It works by manually assigning 2d coordinates to a number of interesting points from parameter space, and then training a Multilayer Perceptron to learn a non-linear mapping from 2d coordinates to parameter values in the instrument's higher-dimensional parameter space. The network can then be fed with 2d coordinates to output instrument parameters using the learned non-linear mapping.

Such dimensionality reduced parameter spaces don't claim to be universally valid: at the opposite, they are specifically defined by the input points provided and their distribution. They can be used to explore spaces-in-between two or more points, in a new control space defined by a 2d spatialization of intuitive distance relationship between those points. These maps can be used to expand or compress the trajectory between two points, to suit the need for a bigger or smaller interaction sensitivity and range. Although mapping is learned

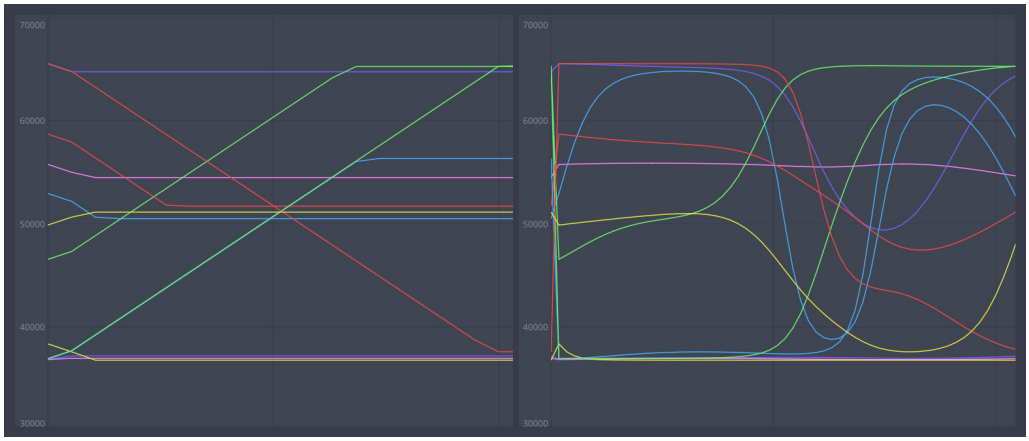

Figure 4.9.: Two different trajectories between the same two points. On the left: linear interpolation in parameter space; on the right: linear interpolation in MLP 2d space, resulting in non-linear interpolation in parameter space.

automatically, its conditions have to be decided by the musician playing the instrument, identifying points, and distributing them.

Once a surface is obtained, the user can move a cursor in its 2d space to perform trajectories. The MIDI buttons grid device maps to a coarse quantization of coordinates in such 2d space, which can then be fine-tuned using the central knobs of the last row of encoders, mapped to adjustments within a quantization unit. Linear interpolation performed on 2d coordinates result in non-linear interpolations in parameter space, as shown in fig. 4.9.

4.3.9. Analyzing audio descriptors to generate audio-driven maps

MLP mapping, and particularly its reliance on a 2d distribution, is reminiscent of corpus-based concatenative synthesis approaches (Schwarz 2012), where a 2d projection of samples' audio descriptors becomes an intuitive control interface for their triggering, as similarity of audio descriptors translates to proximity on the interface. With the MLP approach presented above, similarity is measured in parameter space, and thus it doesn't necessarily apply to the resulting sounds. To explore the affordances of an audio-descriptor driven mapping, audio analysis and dimensionality reduction can be leveraged to automatically generate 2d coordinates for each point of interest, which will be then fed to the MLP to generate a map, aiming to produce a more perceptually linear XY surface.

This approach can produce very different results according to the choice of descriptors, dimensionality reduction technique and their parameters. Not being yet concerned with extensive research on the subject, the present work limits itself to one example choice, and provides an interface for further experimentation. Regardless of specific choices, a generic pipeline for this process is:

1. record produced sound of each point
2. analyze each recording with a variable choice of parameters
3. scale and project the dataset to 2D space
4. send results to MLP to learn the mapping

As its currently working pipeline, the system records each saved point for one second, analyzes MFCCs means and their derivatives as descriptors, and projects the dataset to two dimensions using UMAP(McInnes and Healy 2018; Hart and Harker 2022). UMAP's *minDistance* parameter can be tuned to get more or less uniform distribution. Once the coordinates are sent to MLP, the user can still adjust them manually before (or after, or during) learning the mapping.

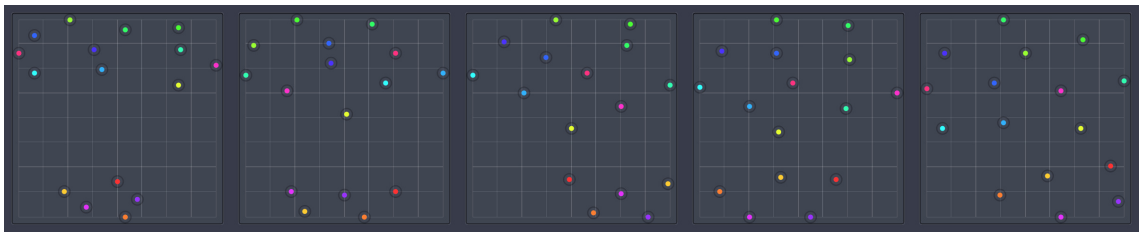

Figure 4.10.: UMAP: projections of the same set of points, with increasing *minDistance*, resulting in more and more uniform distributions. Values from left to right: 0.01, 0.05, 0.1, 0.5. Note also that while left-most projections display similar clusters, the position of individual points changes across plots: this is an effect of the non-deterministic behavior of UMAP algorithm.

4.3.10. Autonomous agent

The highest level of abstraction and mediation developed during this project is a proof-of-concept of an autonomous agent playing the instrument. Based on Kiefer's work on complexity measures (Kiefer 2023) and their use to affect feedback systems' behavior (Kiefer, Overholt, and Eldridge 2020), our agent performs a random walk in a given XY parametric space, either generated with audio descriptors or not, using a temporal complexity metric of the mixer's output sound to adjust its own speed of movement. The intuition is a simple cybernetic mapping: since moving in XY space will change the produced sound, more movement will generate higher values of complexity. This relation is counterbalanced by an inverse mapping between complexity and speed, causing the agent to slow down when visiting more complex sonic regions. A second inverse mapping, between speed and complexity's absolute deviation from its moving average, is meant to avoid the agent getting stuck in some points of the parameter space. Seen as a form of homeostatic machine, the agents works to keep both measured complexity and its variation within a certain range, only by adjusting its speed.

A variety of complexity metrics is reviewed in (Lau et al. 2022). Differently from (Kiefer 2023), we chose to use the Higuchi Fractal Dimension (Higuchi 1988) metric, because of its well defined range for this application: between 1 and 2. I coded an implementation following (Wanliss and Wanliss 2022) and proposed it to be included in Kiefer's *libcccrt*[11].

Even though further research is needed to experiment and design with the agent's behavior, tuning its mapping and parameters, developing a proof-of-concept is relevant to the present work because it represents, at the highest level of indirection, a reintroduction of feedback at the self-regulation level, and thus a transition to a cybernetic instrument.

[11] https://github.com/chriskiefer/libcccrt

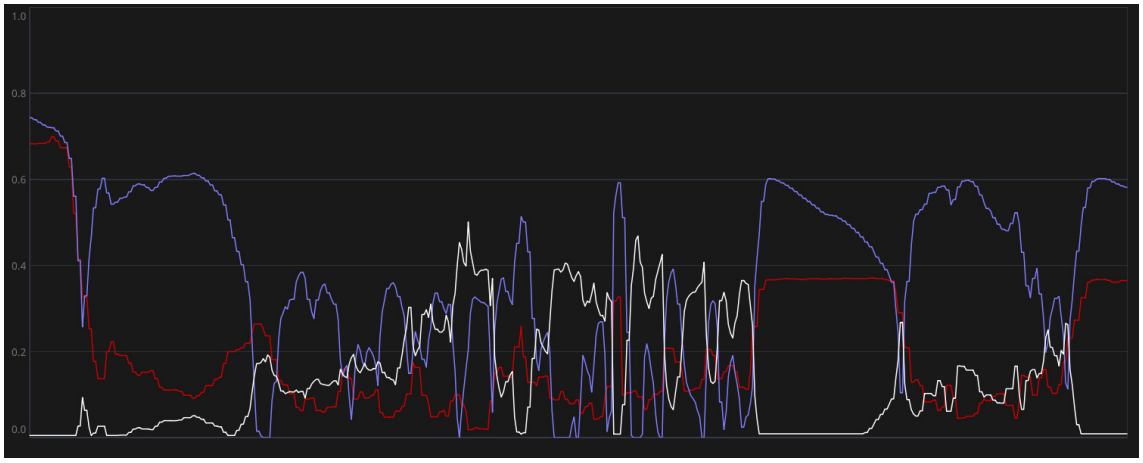

Figure 4.11.: Agent: example recording illustrating how speed (white line) is inversely mapped to measured complexity (red) and its variation (purple). At around 2/3 of the recording (where it becomes less noisy), it's visible how the variation mapping works: complexity is relatively high but stable, leading to a decreasing purple line, which causes speed to rise again.

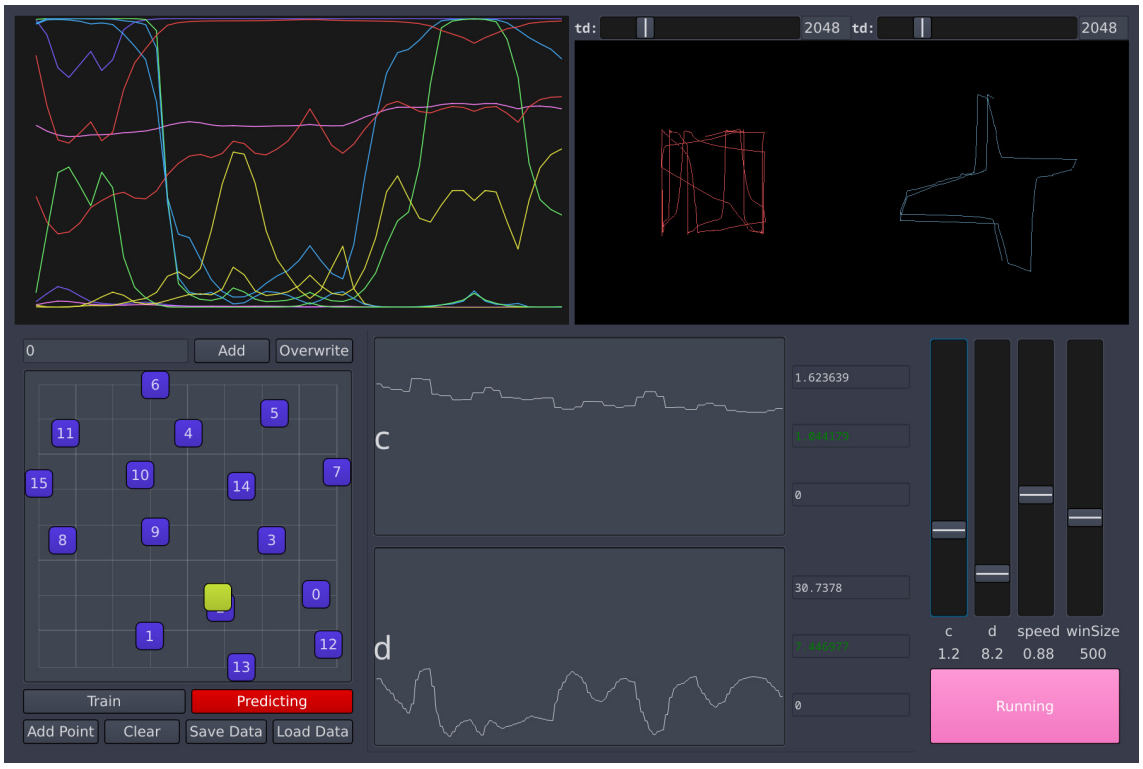

Figure 4.12.: Agent GUI: allows to adjust target complexity and variation, and to monitor the agent's movement and the mixer's state. The plot at the top-left displays mixer parameters variation through time, at the top-right are the usual output phase-space plots, plot "c" is complexity and "d" its deviation.

5. Evaluation

After building the instrument and its software interface, the project went into an evaluation phase, to assess how the project is addressing its research questions and initial motivations. Motivations and questions discussed in sec. 1 can be summarized as assessing the consequences of introducing digital control to analog no-input mixer practice, in terms of what new possibilities are afforded, and what performative attitudes are suggested by the instrument and its interface. Evaluation is performed qualitatively, through a "feature freeze" period during which I formulated the following reflections, integrating feedback from a restricted pool of expert peers.

5.1. Process

Choices made about the evaluation process itself are already a way to address issues in the development process. The first issue that I had after building the instrument was that its realization took most of the energy and focus, and as much as design happened while constantly playing the instrument, my performative practice on it was mostly directed to implementing features and making adjustments to the system. This is one way I can relate to Mudd's claim (see sec. 2), that digital instruments can lead to an "overly analytic" approach: in my experience, their proness to continuous development can take too much focus away from performing. Even if design was guided by listening, imagining and playing the instrument, I didn't have any confidence performing with it, and I hadn't tried to play it enough yet. Comparatively, it took me years of practice to discover and develop techniques for playing traditional no-input mixers, without ever significantly changing the setup, and as much as this process is characterized precisely by never reaching an end, for this new instrument it hadn't started yet.

I thus decided to run an evaluation phase for a week, where I wouldn't modify any code (not only by avoiding to implement new features, but also not fixing eventual bugs), focusing instead on playing the instrument as it was, keeping notes of findings and desires for changes, to assess the current state of the instrument and inform the next development cycle. As part of this evaluation phase I also showed the instrument to a number of peers, listening to them playing it and to their thoughts and impressions about it. The goal of this process is not to study the instrument's performance quantitatively, but rather to gather first impressions on how the instrument suggests ways of playing and composing, informing and transforming performance and composition practices with different aesthetic and poietic starting points than my own, which had been too involved in developing the instrument. For this reason I decided to choose musicians from the local scene who already have experience with both no-input mixers and experimental computer music.

To give an idea about the composition of this pool of experts, it is a group of four people, aged between 27 and 34 years old, two men and two women, each with more than ten years of experience playing music and higher music education, having already worked

with no-input mixers as part of their practice. An overview of how their areas of interest and expertise is given in fig. 5.1 as a word cloud made from publicly available textual descriptions of themselves as artists and their work. Even though impressions from a more diverse group would be valuable to continue the instrument design, I considered more important at this stage to focus on a pool of experts who could more readily contribute to the work's current direction, understanding specific implications of the instrument in relation to no-input mixer practices.

Figure 5.1.: Word cloud illustrating how participants describe themselves and their work, made from their publicly available autobiographies and work descriptions

Feedback sessions with experts lasted between one and three hours each, during which I would typically first show the instrument, demonstrating its features, then listen to them playing it while eventually guiding them through the interface, all along keeping notes of their observations and mine. At various points through the session we would have brief discussions about ideas, impressions and inspirations. Some of the participants also showed me their own related works during these sessions. Notes collected are not reported directly, but were crucial contributions to elaborate the following observations and reflections.

5.2. Interaction affordances and suggested attitudes

Playing the instrument reflects the progressively increasing mediation hierarchy of its interface. This overall organization, and each of its elements, suggests different performative and compositional practices, characterized by more or less mediation in players' involment with the instrument. The following sections examine each layer from bottom to top. To summarize, the instrument suggests two main performative attitudes: one centered around exploring micro-variations and searching for metastable states, as afforded by high resolution controls and interpolation; and the other directed to form sequences perhaps more compositionally, by compiling dictionaries of sounds and manually triggering either saved points or positions in XY space, afforded by save/recall and points organization features. As much as each layer of mediation introduces new possibilities, it also entails further distance from the underlying feedback process, which the interface tries to mitigate by providing acces to its most direct layer of interaction at all times, and offering visualization tools to help relating to its current parametric and sonic state.

5.2.1. Direct parameter manipulation

At the least mediated level, i.e. direct control, the instrument is prone to be played like a traditional no-input mixer, with a few differences. Its sound is generally less noisy, notably lacking white noise hisses that are commonly accessible on no-input mixers by amplifying inherent noise of disconnected channels' amplifiers at high gain. Our EQ design also responds differently than typical analog mixers, most notably in the high frequency range, eventually contributing to less harsh results. However, sounds produced by our instrument are generally recognizable as "no-input mixer sounds", and most knowledge about interacting with a traditional no-input mixer can be applied to our instrument's direct parameter manipulation. Despite the differences, at this level of interaction the instrument is prone to a "continuous learning" approach, to get to know what sounds it can produce and how they can be affected, with non-arbitrary unpredictability , similarly to a traditional no-input mixer (see sec. 2.2). Already at this stage however, partly because of the encoders' action feeling, but most importantly for the presence of a digital interface yet to be explored, the instrument feels different from a no-input mixer. Its discoverability is not only about the complex sonic field of a no-input feedback process, but also about what functions were designed as part of a digital interface, what are their affordances, and perhaps what else could they be designed to do.

Another important point about direct manipulation is that our mixer doesn't expose any interface to control output volume. Although this can be achieved further down in the processing chain, e.g. by using a volume pedal, it makes the instrument feel less dynamic. Loosing control of mix and individual channels output volumes, regardless of feedback gain, is loosing a key semantic capability of mixers, which becomes completely dedicated to feedback, at the cost of feeling even less dynamically controllable. Dynamics have to be searched for in a parametric space that doesn't directly account for them. This makes it harder to access silence, which is however available via the main mute button or by saving and recalling silent states.

High resolution parameter adjustment suggests a "slow" performance mode, focused on searching for unstable states and micro-variations around a parameter state. This process is supported by phase-space visualizations, offering an intuitive and stimulating guide for discerning the instrument's possible states. On the other hand, more coarse encoder mappings and parameter randomization functions permit a more abrupt, physical interaction, which is however suggested more weakly, as most of the tools offered by the interface seem to point to other directions.

5.2.2. Saving and recalling points

Saving and recalling points in parameter space is contributing to facilitating the search for metastable states, at the same time suggesting a more compositional attitude, inviting players to compile a "dictionary" of sounds. Saving those sounds "for later use" is already a premise for composition, suggesting to use the instrument as a reservoir of sounds to be found first, and composed later. However these two phases can influence each other or even happen more or less simultaneously, they characterize a shift of performative attitude from an earlier "searching" phase to a later "triggering" phase. This affords a continuous performative axis between micro- and macro-structure, focusing on one end on sound morphology and micro-variations, and on the other on phrasing morphology and sequences.

When it comes to sequencing, the instrument only affords a manual approach: not offering any automated sequencing facility, it requires the user to play sequences by triggering one sound at a time, maintaining a direct physical connection despite the increased mediation. Even though a triggering approach has a risk of making each recalled sound more static at each repetition (as if they were samples), the availability at all times of direct parameter manipulation allows for introducing variations. Working along this duality is however a critical skill to be cultivated by playing the instrument, which by its interface doesn't immediately suggest it, encouraging instead a static recall of saved points. For each automation added to a creative process, there is a risk for the performer to loose the skill being automated, which has to be weighted against the new possibilities offered by the automation. In this case, the instrument affords to automate gestures that would be nearly impossible on a no-input mixer, at the cost of risking to impoverish direct connection to its parametric state. The following layer of mediation, i.e. interpolation, further automates movements in parameter space, but also offers tools to relate those movements to their underlying parameter states at all times, keeping a connection with lower levels of abstraction.

5.2.3. Interpolation

Point interpolation suggests a second performative axis between abrupt jumps (no interpolation) and continuous transitions (slow interpolation). Interpolated trajectories are gestures involving all parameters at the same time, which can be extremely difficult to perform on a traditional no-input mixer. When they are performed by the interface, they also help the user reaching new points in parameter space, acting as a generative tool to help finding metastable states, if the player is interested in them, or more generally to compile a sonic vocabulary for composition.

Although the 2d view of current parameter state (number 3 in fig. 4.3) is not particularly representative of the sound being produced, it is a valuable tool to follow parameter state changes through interpolation. It gives an intuitive overview of the state of all parameters in relation to each other, in a way that is easier to follow while playing than looking at knobs (which afford greater detail on the state of each parameter by itself) or time domain plots. It helps the user following interpolated trajectories, to eventually interrupt them or alter them by directly affecting mixer parameters. As an example, fig. 5.2 shows a transition between two points, with snapshots of relative 2d and phase-space views. Together with phase-space plots, and of course listening, these visualizations can help maintaining connection between higher levels of mediation, such as interpolated movements in parameter space, and the basic underlying feedback process as influenced directly by its parameters.

Another mediation compromise is due to trajectories being performed automatically. Even if the user can affect them by modulating speed and changing destinations, it's still a less direct influence than on traditional no-input mixers, where each transition has to be performed manually for each parameter. Another approach to grant more direct control could be to introduce a "scrubbing" mode: the user sets up a trajectory by choosing a destination, but no movement is performed automatically, and progress along the trajectory (i.e. position between starting point and destination) has to be controlled manually, for example through an encoder. This would also facilitate reversing the direction of a trajectory, and extrapolating trajectories beyond their extreme points.

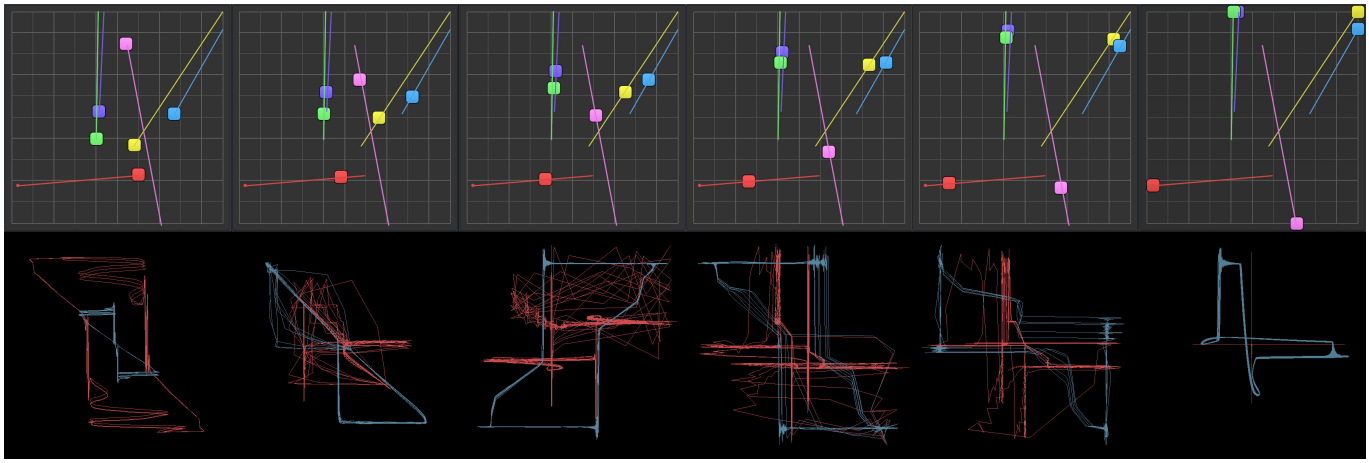

Figure 5.2.: Snapshots at equal time intervals of a linear interpolated trajectory between two points, with relative phase-space plots. Note how more complex figures are encountered between the two extremes.

5.2.4. XY maps

XY MLP maps also contribute to a compositional attitude, acting as a bigger and more nuanced sonic dictionary, that can be played performatively working on the same two axes discussed above (micro-macro vs macro-structure, jump vs. transition). However, since points have to be distributed in XY space manually, and even using UMAP requires recording sound for all saved points, setting up MLP maps entails a break in performance flow. Furthermore, displaying spaces for 16 recallable points suggests that all of them have to be filled before generating a map, resulting in an even stronger formal prescription. A possible improvement would be to generate maps automatically, periodically through a performance, to integrate map navigation more into performance flow.

Then, navigating the map likely shows potential for storing more points, but accessing them in relation to the previous ones is now less immediate, because they get stored in different "sets", and switching between those requires pressing more buttons. The XY space should already be a way to access all points used to generate it, but due to compromises learned during training, it's not precise at recalling them. One strategy could be to generate maps with less than 16 points, for example with sets of 4, so that the process would be repeatable a number of times before a "set" is filled.

Finally, when navigating XY maps there is an inconsistency in how the system works, that can lead to a behavior not intended by design. This discovery aligns with a constitutive principle of no-input mixers, the unintended use of mixers, and even if it could now be integrated into the design as a feature, its nature as unintended behavior is a welcome finding of cultivating a relationship with the instrument. This "bug" and its performative affordance is explained below.

and when This not by a use is of feature, behavior maps works, welcome into and the navigating Finally, intended that XY as the discovery unintended how it design. below. now instrument. with explained aligns mixers, a its is nature to cultivating design affordance a no-input relationship could there unintended can lead the integrated constitutive This even a mixers, of its "bug" principle with of be the is as a in if inconsistency finding performative an system behavior.

There can be a discrepancy between the two representations of the instrument's current state, i.e. its 12d state in parameter space, and its XY coordinates in MLP space. When parameter state is modified directly, current XY state can't be updated accordingly, because inverse mapping is not currently implemented (i.e. calculating XY coordinates from parameter values, which would require building a second MLP). Since interpolation is designed to work by iteratively applying a displacement from the current state of the system, interpolating in XY space could use a different current state than interpolating in parameter space. It was found that when performing a linear transition towards a parameter state, while also training an MLP, the training process would repeatedly make the system jump to its current XY coordinates, unaffected by the ongoing parameter interpolation, resulting in a loop transition between the current XY position and the target parameter state. This process is a primitive form of automated sequencing between two points, and it produces sonic results which are not easily achievable in any other way with the current system. The user can affect it by changing the extremes of this transition setting current XY coordinates or target parameter state, and by modulating interpolation speed.

5.2.5. Agent

At the highest level of mediation, practice with the autonomous agent is characterized by the least level of user interaction, and thus more as listening experiences, built on top of the maps previously generated by the user. Although this is interesting to produce artworks, for example long-durational radio streams, it leaves with a desire for semi-automatic agents that could be more integrated and contribute to performance practice as it's being outlined above. One idea could be an algorithm to find a set of points, either arbitrarly or starting from a provided set, perhaps using complexity measures or perhaps being trained as a classifier, to either populate sets of different sound classes or to identify metastable states.

5.3. Hybrid design compromises

The instrument is designed as a hybrid system, an requires a more complicated setup than either typical analog (which are most often self-contained devices) or digital instruments (which only require a host computer to run and in the best cases can be shared seamlessly as data on the internet). Our instrument not only requires a custom hardware device, which is expensive to build and was realized so far in only one copy, but it's also useless without a computer with custom software and external controllers to play it. This makes it harder to give other musicians chances to work with the instrument for longer times, but on the other hand makes it possible for me to continue its experimental development. Future implementations could aim at resolving this compromise differently, by embedding a more powerful computer and perhaps a display, to make the device playable without a computer and at the same time offering a "developer mode" to modify software interfaces in real-time. This would affect performance practice, not only by making setup easier, but also by removing the laptop from performative settings, a presence that is often frowned upon among electronic music practicioners.

On a final note, there is a semantic difference between hacking a mixer by turning it into a complex synthesizer, and building a hybrid instrument introducing digital control on a no-input mixer. The latter presents itself as a newly designed instrument, rather than

an adaptation of a ready-made one, perhaps loosing the charm of re-signifying a common device. Although for me as a developer these processes are perceived in continuity, the same doesn't necessarily apply to users and audience. The presence of a computer in the setup also influences this difference in conception, as computers can encourage an idea of a technological territory of endless possibilities, to be unlocked by a commanding attitude. In contrast, a self-contained device is perceived as a more strictly defined individual, with a finite set of features, which although can be explored and re-signified through relationship, practice and unmastery, don't suggests as strongly a proness to essential, structural modifications and re-definitions. As a hybrid instrument, our mixer situates critically in between these extremes, offering designers and performers to play across this spectrum. While the analogy to no-input mixers (and the hardware device) can inspire a self-contained approach, the digital interface is prone to continuous re-implementation. These two extremes can inspire each other, but their balance is up to performers and designers, to which the instrument's nature implicitly poses the question and a great part of responsibility.

Table 5.1.: Fixes and feature requests noted during evaluation

Component	description
Presets	autosave periodically to /tmp to resume previous session
Presets	initial state could be silent (all params to -3dB)
Presets	default filename with timestamp
Presets	undo preset save in case of unintentional overwrite
Presets	phase-space plots as point representation
Encoders	map side switch to toggle fine/coarse sensitivity
Encoders	save/recall on encoder switches
Lerp	negative speed
Lerp	map keyboard "space" to toggle interpolation
Lerp	scrubbing mode: control progress manually with encoder
Lerp	plot transition lines on 2d view
History	generate trajectory from history
UMAP	fix crash when analyzing with less than 16 points
MLP	print training loss on GUI
MLP	current state discrepancy between params and XY spaces
MLP	loop paths: implement current state confusion as feature?
Agent	adapt analysis window to detected pitch
Agent	metastability classifier
sclang	make quark for easier code sharing

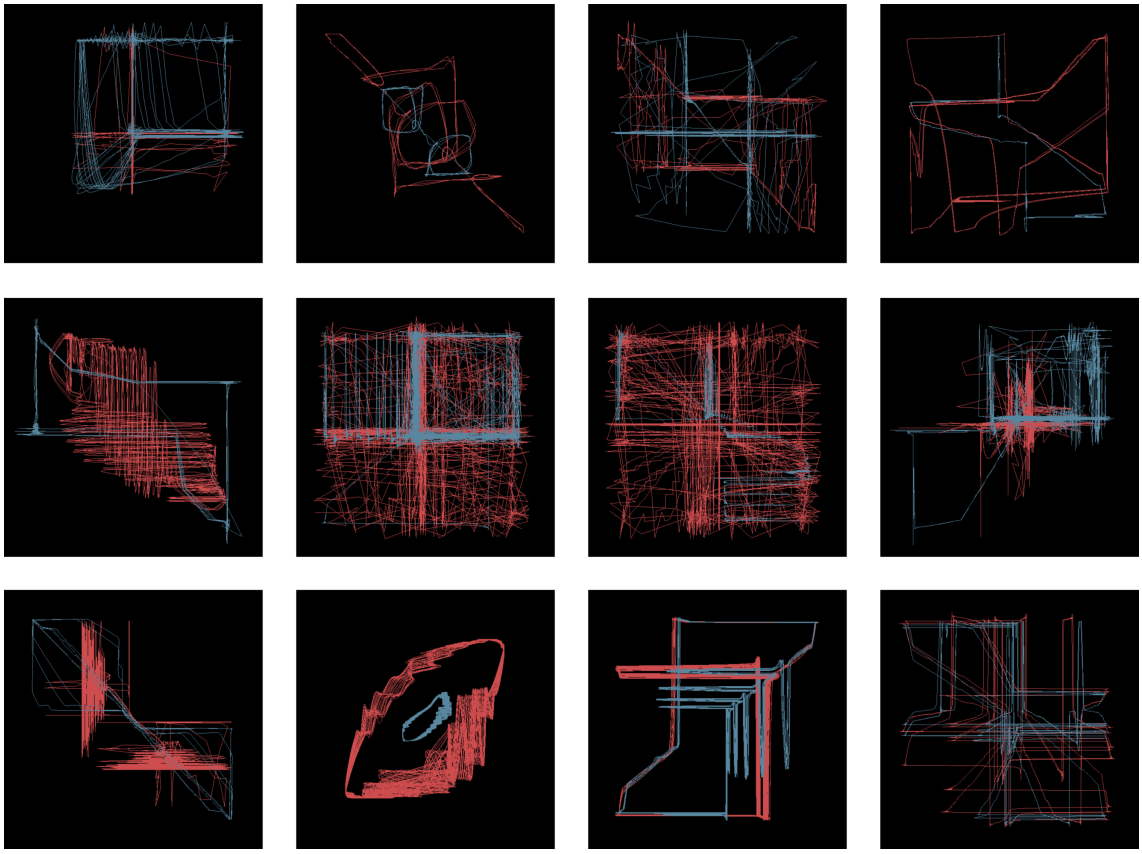

Figure 5.3.: Phase-space plots for a selection of metastable states

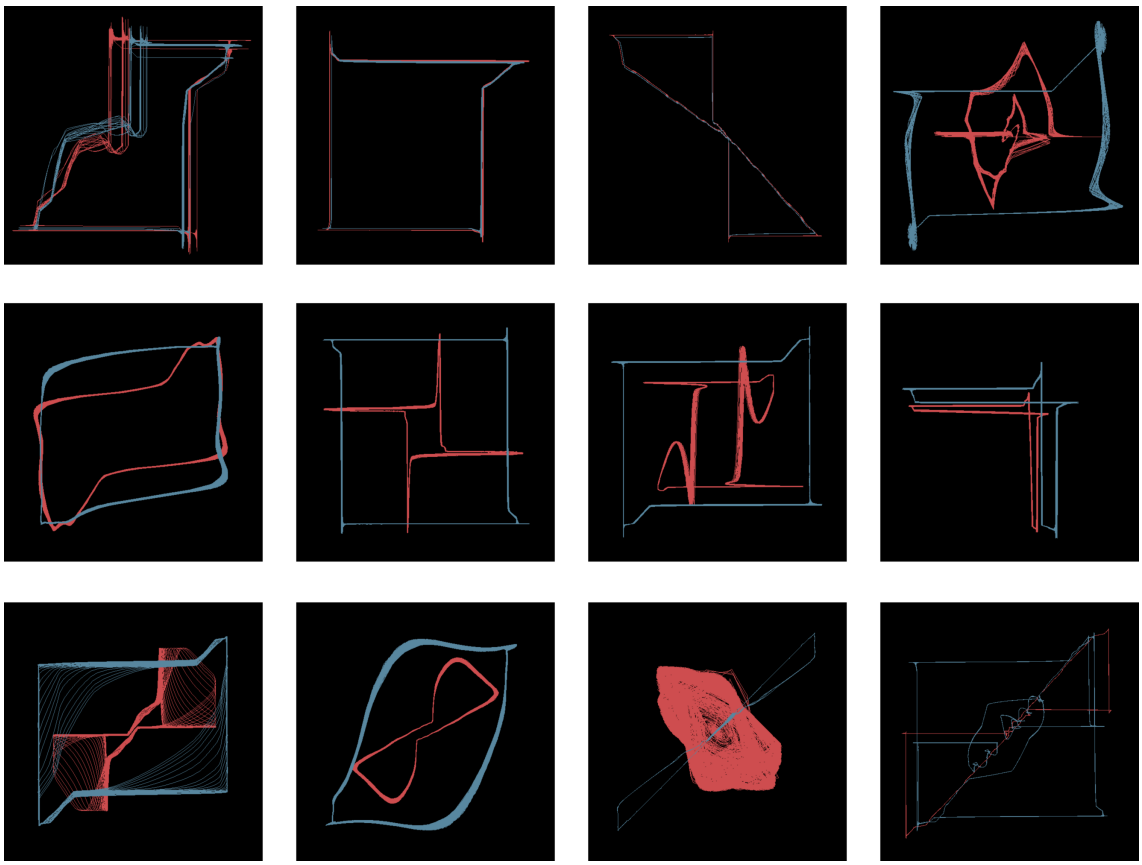

Figure 5.4.: Phase-space plots for a selection of stable states

6. Conclusion

This work presents a new experimental feedback musical instrument, built by re-designing an analog mixer with hardwired feedback connections and a digital control interface to programmatically affect its parameters. Digital controls introduce new approaches to practice with no-input mixers, facilitating multi-parameter gestures and finding metastable states within the underlying feedback process. The interface suggests two main performative/compositional attitudes. On one hand, high resolution controls afford focusing on micro-variations around given parameter states, often inspiring slower performative flows. On the other hand, saving/recalling and organizing points in parameter space suggest a direction towards forming macro-structures, by composing sonic dictionaries and using them as elements to be triggered, sequenced and modified through time.

Although the interface hierarchy entails increasing degrees of mediation, visualization tools help forming intuitive understandings of the system's parametric and sonic state, to follow automated trajectories keeping a relationship to the underlying feedback process at all times. Performers' participation is kept central, having to search for sounds, affect them directly or trigger them manually. At the other extreme of the interaction spectrum, a digital interface also allows for integrating analytic techniques for automated state-space search, organization and autonomous playing, as demonstrated by proofs-of-concept of control surfaces informed by audio descriptors analysis, and an autonomous playing agent. Such techniques can be integrated in performance practice, but generally suggest a contemplative attitude that can be used to produce artworks of their own.

A small pool of experts involved in evaluation agreed on the instrument opening new exciting direction for exploring no-input mixer sonic possibilities perfomatively, compositionally and in production work.

6.1. Future work

This writing documented the first development phase of research around a new instrument. A second prototype is already under development, integrating findings from the present evaluation phase, and implementing new hardware features aimed to simplify its setup. Further development of analytic and autonomous features will be the subject of a specific research period, already planned for the near future, after which the focus will be dedicated to produce artworks and perform with other musicians in the experimental music scene. A first concert featuring the instrument is scheduled in Copenhagen for February 2025.

References

Bowers, John M., and Annika Haas. 2014. "Hybrid Resonant Assemblages: Rethinking Instruments, Touch and Performance in New Interfaces for Musical Expression." In *New Interfaces for Musical Expression*. https://api.semanticscholar.org/CorpusID:9703283.

Collins, Nicolas. 2020. "Improvising with Architecture – Pea Soup and Related Work with Audio Feedback." https://www.nicolascollins.com/texts/peasouphistory.pdf.

corp, THAT. 1999. "Digital Gain Control with Analog VCAs." https://www.thatcorp.com/datashts/dn02.pdf.

David Novak. 2010. "Playing Off Site: The Untranslation of *Onkyô*." *Asian Music* 41 (1): 36–59. https://doi.org/10.1353/amu.0.0054.

Di Scipio, Agostino. 2003. "'Sound Is the Interface': From *Interactive* to *Ecosystemic* Signal Processing." *Organised Sound* 8 (3): 269–77. https://doi.org/10.1017/S1355771803000244.

Eldridge, Alice C. 2022. "Computer Musicking as Onto-Epistemic Playground On the Joy of Developing Complexity Literacy and Learning to Let Others Be." *Journal of Creative Music Systems*. https://api.semanticscholar.org/CorpusID:247808054.

Eldridge, Alice, Chris Kiefer, Dan Overholt, and Halldor Ulfarsson. 2021. "Self-Resonating Vibrotactile Feedback Instruments ||: Making, Playing, Conceptualising :||" In *NIME 2021*. Shanghai, China: PubPub. https://doi.org/10.21428/92fbeb44.1f29a09e.

Elia, Gianluca, and Dan Overholt. 2021. "Squidback: A Decentralized Generative Experience, Based on Audio Feedback from a Web Application Distributed to the Audience." In *Proceedings of the 18th Sound and Music Computing Conference*. Vol. 2021–June. https://vbn.aau.dk/en/publications/squidback-a-decentralized-generative-experience-based-on-audio-fe.

"Feedback Musicianship Network." 2021. https://feedback-musicianship.pubpub.org/.

Fletcher, N. H. 1999. "The Nonlinear Physics of Musical Instruments." *Reports on Progress in Physics* 62 (5): 723. https://doi.org/10.1088/0034-4885/62/5/202.

Ghazala, Reed. 2005. "Circuit-Bending: Build Your Own Alien Instruments." In. https://api.semanticscholar.org/CorpusID:60065252.

Green, Owen, Pierre Alexandre Tremblay, and Gerard Roma. 2018. "Interdisciplinary Research as Musical Experimentation: A Case Study in Musicianly Approaches to Sound Corpora." In *Proceedings of the Electroacoustic Music Studies Network Conference*. Florence. http://www.ems-network.org/IMG/pdf_GREEN_TREMBLAY_ROMA_EMS18.pdf.

Hart, Jacob, and Alex Harker. 2022. "Exploring the Oboe with FluCoMa." *Learn FluCoMa*. https://learn.flucoma.org/explore/harker/.

Higuchi, T. 1988. "Approach to an Irregular Time Series on the Basis of the Fractal Theory." *Physica D: Nonlinear Phenomena* 31 (2): 277–83. https://doi.org/10.1016/0167-2789(88)90081-4.

Kiefer, Chris. 2023. "Dynamical Complexity Measurement with Random Projection: A Metric Optimised for Realtime Signal Processing." In *Proceedings of the Sound and Music Computing Conference 2023*. Stockholm. http://sro.sussex.ac.uk/id/eprint/111991/1/smc2023%20RPC_final.pdf.

Kiefer, Chris, Dan Overholt, and Alice Eldridge. 2020. "Shaping the Behaviour of Feedback Instruments with Complexity-Controlled Gain Dynamics." In *Proceedings of the New Interfaces for Musical Expression 2020 Conference*, 343–48. Copenhagen. https://www.nime.org/proceedings/2020/nime2020_paper66.pdf.

Lau, Zen J., Tam Pham, S. H. Annabel Chen, and Dominique Makowski. 2022. "Brain Entropy, Fractal Dimensions and Predictability: A Review of Complexity Measures for EEG in Healthy and Neuropsychiatric Populations." *European Journal of Neuroscience* 56 (7): 5047–69. https://doi.org/10.1111/ejn.15800.

Magnusson, Thor, Chris Kiefer, and Halldor Ulfarsson. 2022. "Reflexions Upon Feedback." In *NIME 2022*.

Marasco, Anthony T., Edgar Berdahl, and Jesse T. Allison. 2019. "Bendit_i/O: A System for Networked Performance of Circuit-Bent Devices." In *New Interfaces for Musical Expression*. https://api.semanticscholar.org/CorpusID:195888140.

McInnes, Leland, and John Healy. 2018. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." *ArXiv* abs/1802.03426. https://api.semanticscholar.org/CorpusID:3641284.

Melbye, Adam Pultz. 2022. "A Continuously Receding Horizon." *ECHO, a Journal of Music, Thought and Technology*, no. 3 (January). https://doi.org/doi.org/10.47041/DXGG9867.

Melbye, Adam Pultz, and Halldór Úlfarsson. 2020. "Sculpting the Behaviour of the Feedback-Actuated Augmented Bass: Design Strategies for Subtle Manipulations of String Feedback Using Simple Adaptive Algorithms." In *New Interfaces for Musical Expression*. https://api.semanticscholar.org/CorpusID:221669229.

Meyer, William, and Toshimaru Nakamura. 2003. "Toshimaru Nakamura Sound Student." *Interview for Perfect Sound Forever, Online Music Magazine*. http://www.furious.com/perfect/toshimarunakamura.html.

Moore. 2022. "Controlling a Synth Using a Neural Network." *Learn FluCoMa*. https://learn.flucoma.org/learn/regression-neural-network/.

Mudd, Tom. 2023. "Playing with Feedback: Unpredictability, Immediacy, and Entangled Agency in the No-Input Mixing Desk." In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–11. Hamburg Germany: ACM. https://doi.org/10.1145/3544548.3580662.

Mudd, Tom, Simon Holland, and Paul Mulholland. 2019. "Nonlinear Dynamical Processes in Musical Interactions: Investigating the Role of Nonlinear Dynamics in Supporting Surprise and Exploration in Interactions with Digital Musical Instruments." *International Journal of Human-Computer Studies* 128 (August): 27–40. https://doi.org/10.1016/j.ijhcs.2019.02.008.

Romkey, J. 1988. "A Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP." {RFC} 1055. IETF. https://datatracker.ietf.org/doc/html/rfc1055.

Sanfilippo, Dario, and Andrea Valle. 2013. "Feedback Systems: An Analytical Framework." *Computer Music Journal* 37 (2): 12–27. https://doi.org/10.1162/COMJ_a_00176.

Schwarz, Diemo. 2012. "The Sound Space as Musical Instrument: Playing Corpus-Based Concatenative Synthesis." In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. University of Michigan. http://articles.ircam.fr/textes/Schwarz12a/index.pdf.

Trapani, Dario, and Toshimaru Namakura. 2017. "The Art of Noise: Toshimaru Nakamura Interview." *MusicOff, Online Music Magazine*, November. https://www.musicoff.com/musica-e-cultura/interviste/the-art-of-noise-toshimaru-nakamura-interview/.

Valle, Andrea, and Dario Sanfilippo. 2012. "Towards a Typology of Feedback Systems." In

International Conference on Mathematics and Computing.

"Victor and Sophie's No-Input Textiles | Intelligent Instruments Lab." 2023. https://iil.is/openlab/61.

Wanliss, J. A., and Grace E. Wanliss. 2022. "Efficient Calculation of Fractal Properties via the Higuchi Method." *Nonlinear Dynamics* 109 (4): 2893–2904. https://doi.org/10.1007/s11071-022-07353-2.

Wright, Matthew James, and Adrian Freed. 1997. "Open SoundControl: A New Protocol for Communicating with Sound Synthesizers." In *International Conference on Mathematics and Computing*. https://api.semanticscholar.org/CorpusID:27393683.

A. Appendix

A.1. Contributions to open-source projects

The present work is built on top of open-source software, and generated contributions to such ecosystem in form of bug fixes, reusable code, and publicly available software. The following is a list of the relevant contributions:

PaulStoffregen/Audio

- #468: correct headphoneSelect bits.

chriskiefer/libcccrt

- #2: avoids scsynth crash when windowSize >= maxWindowSize
- elgiano/libccrt/feat/higuchi: implements real-time calculation of Higuchi Fractal Dimension. Not yet merged upstream.

supercollider/supercollider

- #6212: ScopeView: allow lissajou for more than 2 chs as overlaid pairs.

elgiano/oscslip-proxy

- Since it's not uncommon to design devices (especially Arduino, ESP32 and similar, for audio related projects) to transmit OSC over serial, I released oscslip-proxy (sec. 4.2) on both GitHub and Python Package Index (pip).

elgiano/handjoints-osc

- Utility to track hand finger joints using machine learning on a live camera feed (e.g. laptop webcam) and broadcasting their coordinates via OSC. Also available on PyPi (via *pip install handjoints-osc*).

A.2. Schematics

The following pages presents schematics and PCB designs for both DCA and mixer circuits.

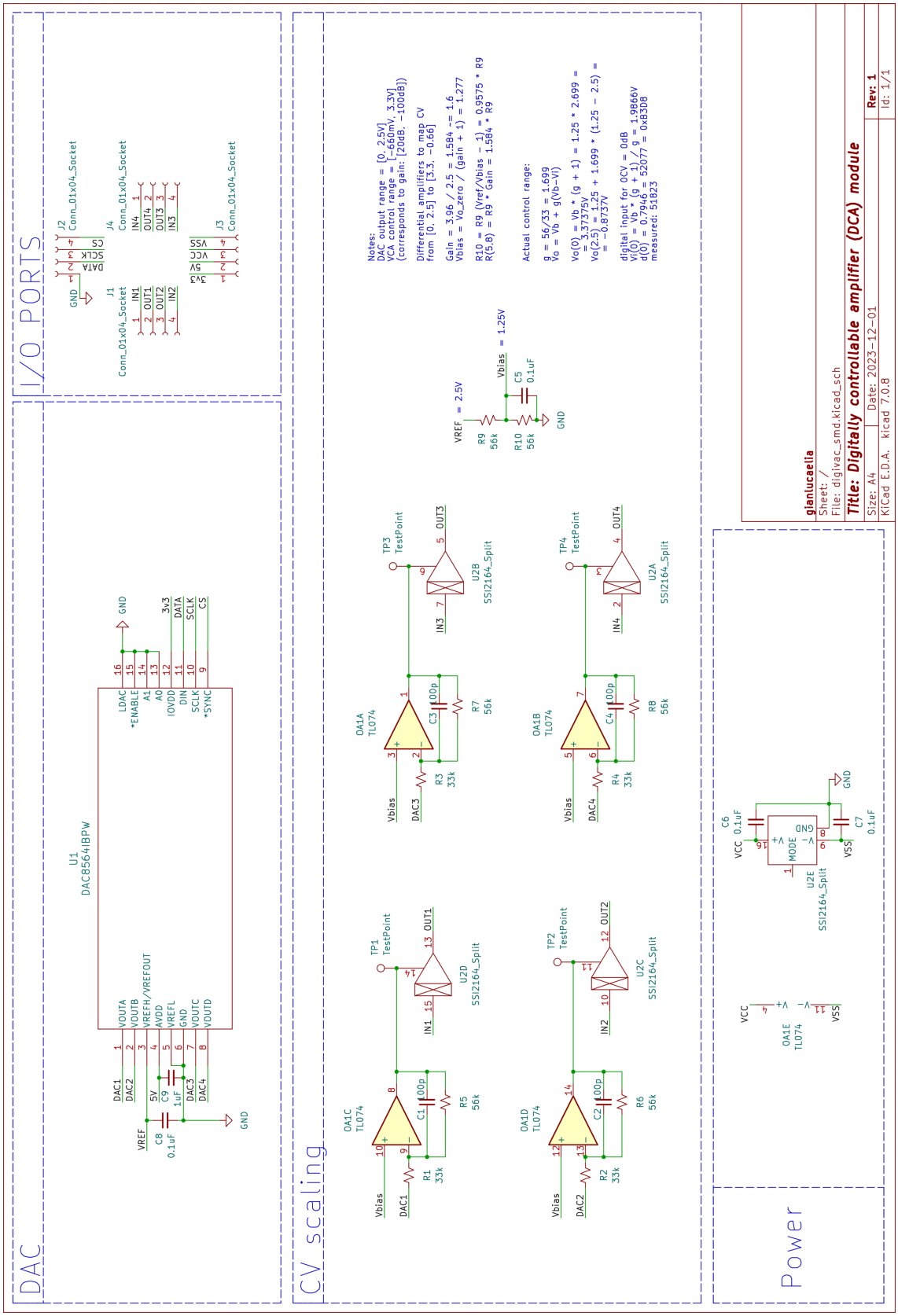

Figure A.1.: DCA schematics

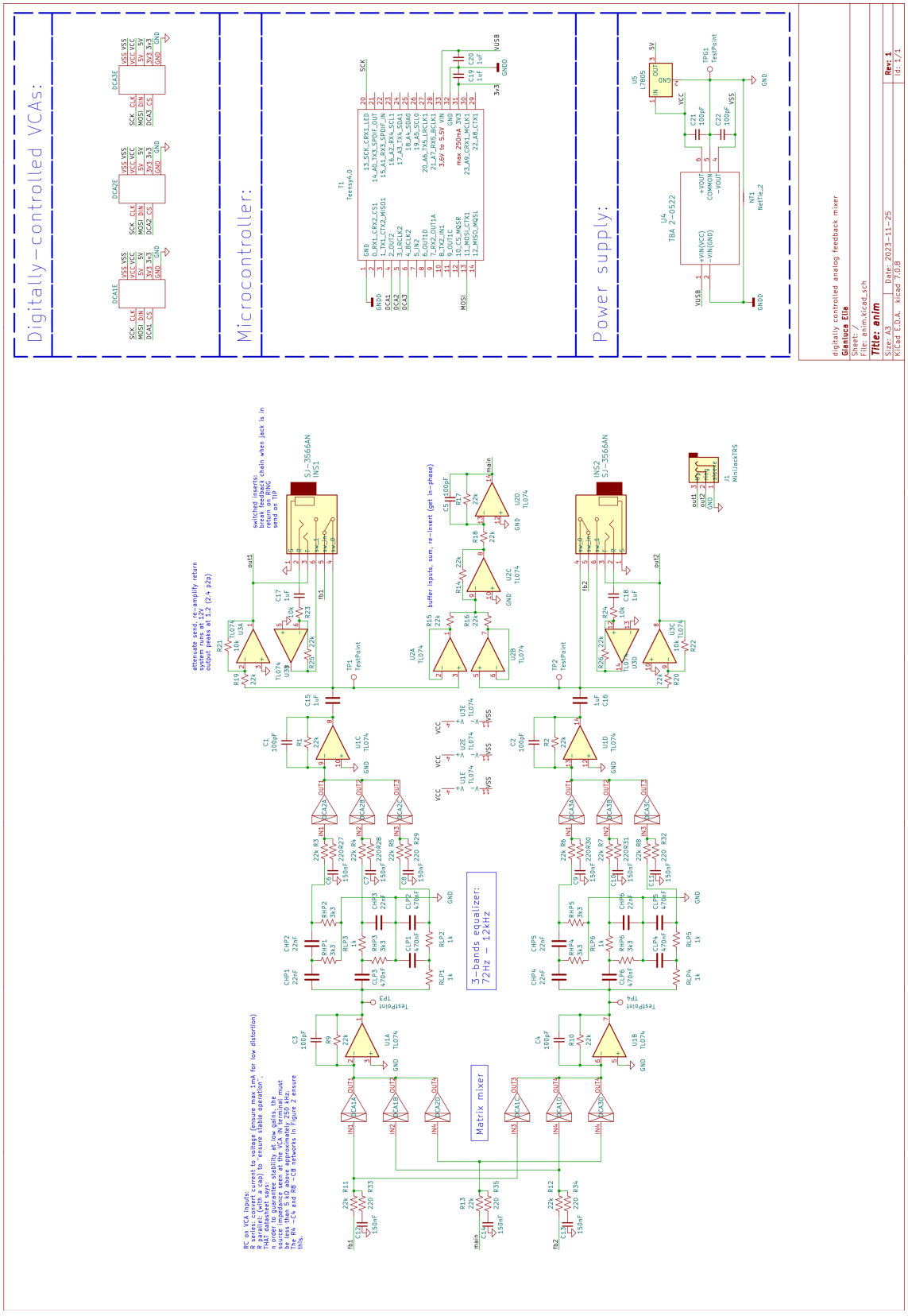

Figure A.2.: Mixer “motherboard” schematics

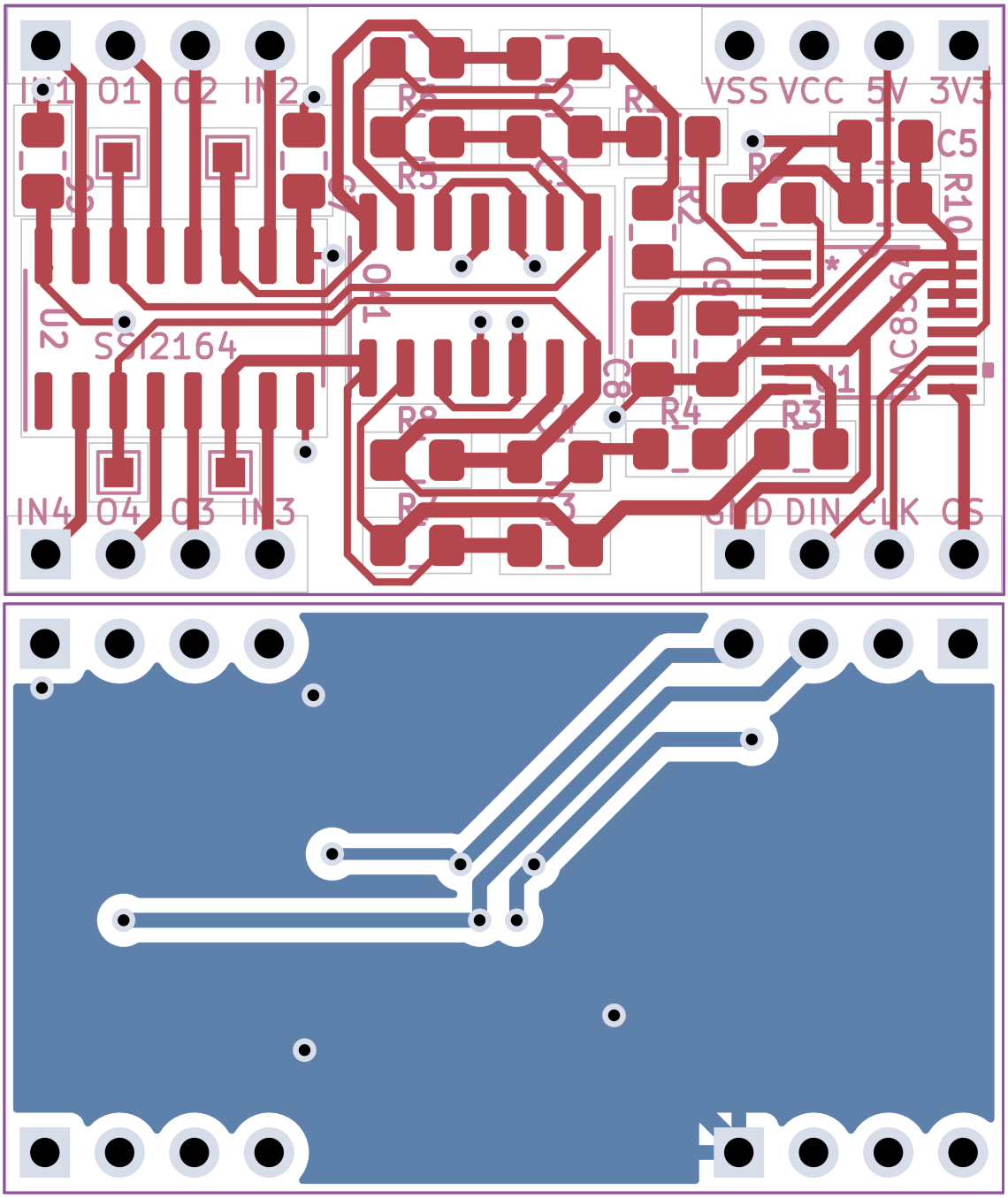

Figure A.3.: DCA PCB design front and back

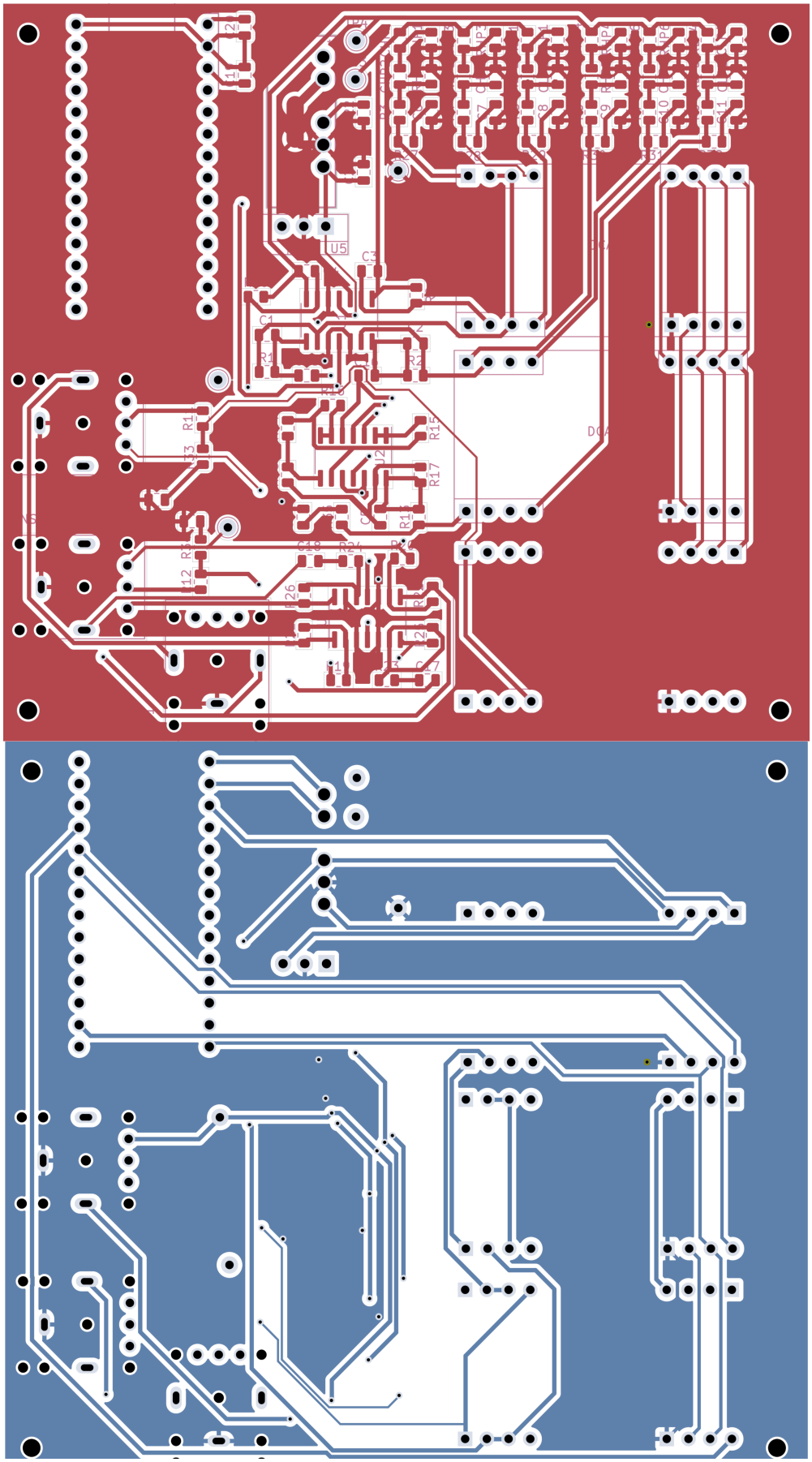

Figure A.4.: Mixer “motherboard” PCB design front and back

Acknowledgements

I had this work in mind for a long time, through my experience as an electronic musician playing no-input mixer and developing digital musical instruments. I first want to thank Jonatan Uranes, we've been talking about automating no-input mixers for years, and now a first step is done. Every single person in the noise scene has made my life in Copenhagen more bearable culturally in recent years, and gave me an artistic context to output my work, and that's also thanks to Jonatan, Lorenzo Colocci, and other people whose artist name is not appropriate for this text, for their immense musicianship and organization efforts, keeping the scene alive and exciting.
I would like to thank Marcela Lucatelli, Søren Kjærgaard and Torben Snekkestad for believing in my skills at an early stage and giving me opportunities to develop further by working on their projects, accompaining my transformation from a saxophone player to whatever I'm doing now.
More practically, I need to thank Archelaos Vasileiou for initial suggestions about hardware design, especially components selection. The Teensy online community on PJRC forum has also been a nice place to find guidance and feedback through the hardware design phase. Finally, Dan Overholt, for connecting me to a relevant research community and being there following the making of this thesis.

Gianluca Elia, 2024