

TABULA

A Deformable Interface for Musical Expression



Figure 1: Tabula

Hannah Eikens

Department of Industrial Design, Eindhoven University of Technology, h.l.l.eikens@student.tue.nl

Word count: 7440

This work explores how deformable interfaces can be used as expressive tools used by electronic musicians. Deformable and shape-changing interfaces have become an increasingly popular area of research within design/HCI communities, yet many of this work focusses on either technical actuation or taxonomies of deformation and interaction. For this project, responsive flexible MIDI interfaces were created through combining 3D-printed compliant mechanisms with hall-effect sensing, resulting in the final design outcome Tabula. Three design iterations were developed through co-creation sessions with two expert musicians with a background in electronic music, which through making and semi-structured interviews, allowed for gathering of early stage insights into musical expressivity. The findings show that a deformable interface that entangles two or more sound parameters into a multi-axis physical deformation grants the musician expressivity through facilitating fine-motor skills. As well as enabling for surprising sonic outcomes through playful engagement. Furthermore, the investigation also highlights how an accessible fabrication method, in this case filament based 3D-printing, can be used to rapidly prototype circuit boards and compliant mechanisms. Tabula has potential as music performance tool both in the studio and live. Beyond music it can also transfer into other contexts, such as gaming and mobility.

Additional Keywords and Phrases: Deformable Interfaces, MIDI Interfaces, Entangled Control, Expressive

1 INTRODUCTION

Expressive control is a central concern in the design of digital musical instruments (DMIs). MIDI-based controllers have made electronic music production widely accessible, most commonly taking the form of keyboard controllers, drum pads, and hardware with discrete buttons, knobs, and faders. These interfaces are predominantly rigid, and that rigidity places inherent constraints on the richness of physical interaction available to a performer. The physical gestures used to play acoustic instruments carry nuance that standard MIDI interfaces struggle to capture: the continuous resistance of a string, the breath dynamics of a wind instrument, the tactile gradient between a soft and a forceful touch. As electronic music production has matured, so too has interest in expanding the physical vocabulary of performance, motivating research into alternative interface modalities that can more fully embody the expressive intent of the musician.

Deformable and non-rigid interfaces represent one such direction. By allowing the physical body of an instrument to be squeezed, bent, stretched, or twisted, deformable DMIs open up interaction spaces that rigid controllers cannot support. Commercial instruments such as the Roli Seaboard¹, the Expressive E Osmose² and Touché³, and the Intuitive Instruments Exquis⁴ demonstrate industry recognition of this potential, offering continuous pressure and surface sensitivity well beyond what conventional MIDI hardware affords. Yet systematic academic investigation of deformable DMIs remains limited: existing surveys document the sensing technologies used across research instruments [16], and practitioners in the field have explicitly called for more rigorous musical appropriation of deformable technologies [2]. Within research, the specific technical combination addressed in this project has not previously been explored: 3D-printed compliant mechanisms as the deformable body of a DMI, paired with three-dimensional hall-effect magnetic sensing to capture that deformation as continuous multi-parameter data. This implementation is investigated here through a co-design process with practicing musicians, which contributes a methodological lens for understanding how such a design is received and interpreted in musical practice. The theoretical groundwork suggests it is a productive direction: work on entangled agency in digital musical instruments [10] indicates that configurations in which a single physical action simultaneously affects multiple sonic parameters can generate expressive and playful musical behavior that neither performer nor instrument produces in isolation.

This project investigates that intersection through the design, prototyping, and evaluation of a series of deformable MIDI interfaces developed in collaboration with two expert electronic musicians. Following a Research through Design (RtD) methodology [6,19], three iterative co-design sessions were conducted with each participant, culminating in the final prototype, Tabula: a hand-operated deformable controller that uses an array of hall-effect magnetic sensors to translate continuous physical deformation into multi-parameter MIDI output. The design process was informed at each stage by participant feedback, and the insights generated through making and evaluation form the primary research contribution of this work.

The study is guided by the following research questions:

¹ <https://roli.com/eu/experience/seaboard-m>

² <https://www.expressivee.com/2-osmose>

³ <https://www.expressivee.com/1-touche>

⁴ <https://dualo.com/en/exquis/>

Main research question: How can deformable interfaces be designed to facilitate musical expressivity for practicing electronic musicians?

Sub-questions:

- How do expert electronic musicians interpret and engage with novel deformable MIDI interfaces through co-design?
- What role does multi-parameter entanglement in physical deformation play in enabling expressive and playful musical performance?
- How can accessible digital fabrication methods support the rapid prototyping of deformable musical interfaces?

The remainder of this paper is structured as follows. The related work section situates the project within existing research on deformable interfaces, magnetic sensing for musical control, multi-dimensional controllers, and the theory of entangled agency. The methodology section describes the RiD and co-design approach, the participant selection process, and the data collection and analysis methods. The design process section traces the iterative development of the prototypes across three phases. The findings section presents the two themes identified through analysis of the co-design sessions. The paper closes with a discussion of the implications for DMI design and directions for future work.

2 RELATED WORK

2.1 Deformable Interfaces in Music

Deformable and non-rigid interfaces have attracted growing interest within HCI and design research, yet their application in music remains limited. A survey of deformable instruments developed within the NIME community [16] documents the range of sensing technologies employed across existing designs. Notably, none of the instruments covered use magnetic or hall-effect sensing, pointing to an unexplored technical direction in the design of deformable DMIs. A complementary study by Boem et al. [2], which surveys eleven experts and pioneers in the field, finds that non-rigid interfaces in music remain sparse and unsystematic despite their broader adoption in contexts such as game controllers, bendable smartphones, and shape-changing displays. Based on expert input, the authors call for more systematic musical appropriation of deformable technologies. The broad range of gestures that deformable interfaces can support [11] further suggests significant untapped potential for musical interaction.

2.2 Magnetic and Hall-Effect Sensing for Musical Control

While magnetic sensing has not previously been applied to deformable musical interfaces, several instruments demonstrate its viability for expressive musical control. The Chimaera [13] is a touch-less polyphonic controller based on a hall-effect sensor array, in which neodymium magnets worn on the fingers excite the sensors to produce a continuous two-dimensional interaction space. With update rates of 2 to 4 kHz and sub-millisecond latency, the Chimaera demonstrates that hall-effect sensing can support highly responsive and expressive musical control. CubeHarmonic [9] takes a different approach, using a magnetic tracking system based on LC coils to detect the three-dimensional positions of elements in a modified Rubik's cube, translating spatial data into harmonic and timbral musical output. Although neither instrument is deformable, both establish that magnetic sensing can generate musically meaningful multi-dimensional data in performance contexts.

Beyond musical applications, eFlesh [12] introduces a low-cost magnetic tactile sensor developed for robotic manipulation, constructed from 3D-printed flexible microstructures, off-the-shelf permanent magnets, and a magnetometer circuit board. The sensor demonstrates sub-millimeter contact localization accuracy, showing that this sensing approach is both technically accessible and fabricable with hobbyist tools. Although developed for robotics, eFlesh validates the underlying sensing principle that informs the present work.

2.3 Multi-dimensional Physical Controllers

The Triplexer [8] offers a close functional comparison to the present work: a three-degree-of-freedom foot controller that maps physical pressure and position simultaneously to multiple sound effects parameters. A user study with ten musicians found the device generally expressive and playable, but identified a consistent need for visual feedback during use. All participants found a software display showing foot position helpful, and the authors subsequently incorporated a graphical screen into revised hardware designs. This reliance on visual feedback contrasts with the approach taken in this project, where the spring-back behavior of compliant mechanisms provides inherent feedback in the sense described by Wensveen et al. [17]: passive feedback arising from the physical properties of the interface itself, rather than augmented through an added display or indicator. This potentially reduces dependence on visual cues as a reference during performance. The Triplexer is also designed specifically for foot operation with the hands free, whereas Tabula is a hand-operated device intended for studio and live use alongside other controllers.

Super Size Me [20] investigates how the physical scale of a digital musical instrument affects both performer experience and audience perception. The study finds that instrument size shapes how musicians embody and identify with their instruments, and influences the legibility of performance gestures for an audience. These findings are relevant to the design of Tabula, particularly in evaluating its suitability for live versus studio contexts.

2.4 Expressivity, Unpredictability, and Entangled Agency

A key theoretical resource for the present work is Mudd's study of the no-input mixing desk [10], which examines why musicians value working with unpredictable and partially uncontrollable systems. Through thematic analysis of twenty interviews with no-input mixer practitioners, three themes are identified: the joy of working with the unknown, gestural immediacy and tactility, and favorable comparisons with acoustic instruments. Of particular relevance is the concept of entangled agency, which draws on Barad's notion of intra-action [1]: rather than agency residing in the musician or the instrument separately, it emerges through their encounter. Musicians using the no-input mixer described the instrument as a collaborator that surprises and resists the player, generating creative outcomes that neither party could produce alone.

Mudd also highlights the role of parameter cross-mappings in creating explorable and expressive digital musical interactions: configurations where a single physical input affects multiple sonic outputs, or where multiple inputs converge on a single perceptible outcome. This concept is central to understanding the expressive potential of deformable interfaces, where a single physical deformation necessarily entangles multiple sensing axes. The present work extends this idea into a physical domain: the multi-axis deformation of a compliant mechanism creates a parameter space whose entanglement is embodied in the material behavior of the interface itself, rather than configured purely in software.

2.5 Design Practice in Digital Lutherie

Renney and Gaster [14] offer insights into how digital luthiers approach the design of new musical instruments, drawing on interviews with 27 practitioners. The study finds that interaction and control are the primary focus of digital luthiers, and that they tend toward opinionated and directed design choices rather than universal solutions. This framing is relevant to the present project, which situates itself within a digital lutherie practice, using filament-based 3D printing to fabricate both the compliant mechanisms and the circuit boards, developing functional prototypes that are evaluated with musicians as part of a co-design process.

2.6 Summary

Taken together, the existing literature establishes that deformable interfaces in music remain underexplored, and that practitioners themselves have called for more systematic investigation in this area [2]. While multi-dimensional controllers exist [8,9,13], none combine a deformable physical form with magnetic sensing for musical use, and none were developed through co-design with musicians around the specific goal of musical expressivity. The theoretical work on entangled agency and parameter cross-mappings [10] suggests that the simultaneous entanglement of multiple sound parameters within a single physical gesture may be a productive design direction, but this has not been empirically investigated through a co-design process with practicing musicians. This gap motivates the research questions outlined in the introduction and the co-design methodology described in the following section.

3 METHODOLOGY

This work follows a Research through Design (RtD) methodology [6,19], in which the design and prototyping process serves as the primary mode of inquiry. RtD is well-suited to this project because the central research questions concern how deformable interfaces are interpreted and used in practice, knowledge that emerges through iterative making and evaluation with users rather than from prior theory alone [19]. The design artifacts function as research instruments, and the process of creating and refining them generates insights into musical expressivity with deformable interfaces. Existing work in this domain has largely focused on technical actuation or taxonomies of deformation and interaction; this project instead investigates how such interfaces are experienced by practicing musicians.

3.1 Participant Recruitment and Requirements

Participants were recruited from the researcher's personal network. Multiple candidates were considered; not all responded, and not all were deemed suitable. Selection prioritized the researcher's intuitive judgment over strict prescription, with several qualifications in mind: active involvement in electronic music production, experience with hardware and software synthesizers and MIDI interfaces, and a background in acoustic instruments that would inform their engagement with novel physical interfaces. The two participants who best met these criteria across background, experience, and availability were selected. A full copy of the recruitment letter is included in Appendix A.

The study involved two participants. Rather than seeking broad statistical representation, the aim was to engage deeply with expert informants across multiple extended sessions, allowing their responses to directly inform iterative design decisions. The exploratory nature of the work further justifies a small, carefully selected

sample: the goal was not to generalize findings to a population, but to generate rich design knowledge through sustained collaboration with practitioners whose expertise could meaningfully challenge and develop the designs.

As compensation for their participation, both participants will receive a fully functional version of the final prototype, Tabula.

Aniek de Rooij

A prime candidate for this study, Aniek also known as Alberta Balsam, has years of experience and is an established name in the Rotterdam scene. Through her releases at labels such as Dekmantel and Clone she shares her love of science fiction through sounds inspired by 90s IDM, Techno and Electro.⁵

She has over 15 years of experience with using Ableton Live and various MIDI controllers, but also played instruments from a young age like piano and violin. She also studied at Codarts Rotterdam.

During the interview she expressed different appreciation for instruments she plays. The piano is by far her favorite lending itself perfectly for composing, but it invites for more classical melodic composition, whereas the electronic synthesizers offer her rhythmic exploration by guiding the instrument rather than playing it.

Marko Milas

Hailing from Croatia he has been producing under the moniker Livid Husk for some time now, defining himself as a genreless composer.⁶ He has been playing string instruments such as classical guitar from a young age, as well as playing in mandolin ensembles. He also plays electric guitar and bass guitar in bands. Currently his favorite instrument is actually the piano because it is well suited for producing songs. Electronic Dance Music production is what initially motivated him to moving to the Netherlands, but stayed after pursuing degrees signal processing and electrical engineering. This knowledge of signals grants him more informed control over the sound design choices he makes in his music.

3.2 Co-Design Procedure

The study was conducted across three co-design sessions, each building on insights gathered in the previous one. This three-stage structure was informed by a comparable iterative co-design process used in a musical context [5]. Co-design, understood here as the active involvement of participants as partners in the design process rather than as passive subjects [15], was chosen because the subjective, embodied nature of musical expressivity requires insight that cannot be derived from researcher-led design alone. As the researcher is also a practicing musician, domain knowledge informed design decisions throughout the project. However, involving participants with distinct musical backgrounds and working practices was important to move beyond the researcher's own perspective and enrich the data with insights that would not otherwise have been accessible.

Each session followed the same familiarization procedure, drawn from the approach used by Zheng et al. [18], who in turn adapted their approach from the material probe method [7]. Participants were introduced to each device in three sequential steps: first through visual inspection alone, then through tactile exploration without software interaction, and finally through unrestricted use in combination with their DAW (Ableton Live). This stepwise approach was chosen because it surfaces participants' initial expectations of a novel interface before

⁵ <https://ra.co/dj/albertabalsam/biography>

⁶ <https://lividhusk.bandcamp.com/>

those expectations are revised by hands-on experience. It also allows participants to reflect on the difference between what they perceived the device to afford and what it actually affords in practice, and to contextualize both against their existing frame of reference, in this case their experience with other musical instruments and interfaces. This is particularly relevant for deformable interfaces, where the physical range of motion may suggest more or fewer expressive possibilities than the sensing system actually registers.

In Phase 1, participants were introduced to three existing deformable MIDI interfaces from the researcher's prior project, Sonable. In addition to structured feedback on each device, participants were asked about their musical background and experience with existing instruments and controllers, and were subsequently asked to generate their own speculative interface designs.

In Phase 2, two new devices were presented: one based on participants' speculative designs from Phase 1, and one incorporating their suggested improvements to the Sonable interfaces. The same session structure was followed, and feedback informed a further round of iteration.

In Phase 3, the final iterated device, Tabula, was introduced. Participants were asked to build instrument and effect racks in Ableton Live using the device, to perform with it, and to reflect on their experience. They were additionally asked to assess the device's potential as a commercial product, including willingness to pay and conditions for purchase.

3.3 Data Collection

Data was collected through semi-structured interviews conducted within each session, with questions structured around the material probe procedure described above. The full interview question guide is included in Appendix A. Sessions were also video-recorded to capture gesture and physical interaction data.

3.4 Data Analysis

Interview data was analyzed using reflexive thematic analysis[3,4], following an inductive and semantic approach in which codes were generated from the data rather than applied from a prior theoretical framework. The researcher's background as both a designer and a practicing musician is acknowledged as a factor that may have shaped how questions were asked during sessions and how participant responses were subsequently interpreted. This positionality also brought relevant domain knowledge to the coding process, enabling closer reading of musically specific accounts.

Session transcript, recorded audio and notes were used to generate preliminary codes. Rather than code the transcripts, coding was done on written session summaries. The transcripts read more like conversations and tended to go off topic from time to time, and to filter this, it was opted to code summaries. It was the researchers first time applying thematic analysis in practice. Two themes were identified as most directly relevant to the research questions. The first concerns the role of multi-parameter entanglement in enabling sonic exploration: when a single physical deformation simultaneously controls multiple sound parameters, participants described outcomes that were expressive, surprising, and difficult to reproduce with conventional controllers. The second theme concerns sensing sensitivity as a prerequisite for fine-motor expressivity: participants responded positively to the high responsiveness of the hall-effect sensing across all devices, including the Sonable interfaces, which share the same sensing principle but do not feature entangled parameter control. This distinction within the data allowed the contribution of entanglement to be isolated from the contribution of sensing technology alone. These themes are discussed in relation to the findings in the sections that follow.

It should be noted that due to analysis being conducted on researcher-compiled session summaries rather than verbatim transcripts, the analytical rigor of this study is limited.

4 DESIGN PROCESS

The design process unfolded across two parallel tracks. The first concerned technical implementation: validating the fabrication methods and sensing approach that would underpin all prototypes. The second was an iterative co-design process structured across three phases with two participants. These tracks are presented separately here, though in practice they were closely intertwined: fabrication decisions shaped what designs were feasible, and participant feedback determined what was worth pursuing in the next iteration.

4.1 Technical Implementation

4.1.1 Fabrication Approach

All prototypes produced during this project relied on filament-based 3D printing as their primary fabrication method, applied in three distinct ways: conventional rigid printing for housing and structural components, flexible printing for the compliant mechanisms that enable deformation, and an emerging technique for producing functional printed circuit boards. Each approach was validated before being incorporated into a design prototype.

4.1.2 Deformable Mechanisms

The viability of 3D-printed compliant mechanisms for deformable interfaces was established during the researcher's preceding M21 Sonable project. The key variable governing the behavior of these mechanisms is the stiffness of their flexible elements, which determines both how much force a performer must apply to deform the interface and how responsively it returns to a neutral position. Tuning this stiffness to a playable range is a critical fabrication consideration.

Two approaches to stiffness reduction were investigated alongside geometry-based tuning. The first was the use of Colorfabb Varioshore TPU⁷, a filament whose shore hardness can be reduced by increasing print temperature. Based on testing reported by CNC Kitchen⁸, a print temperature of 220 degrees Celsius with a flow rate of 60 percent yields a shore hardness of 65A, significantly lower than the 95A of conventional TPU used in the Sonable project.

The second approach was to reduce material density at a macro level by lowering sparse infill percentage in flexible parts. A set of samples was printed in Varioshore at 220 degrees and 70 percent flow rate with outer perimeters disabled, testing gyroid infill at 5, 10, 15, and 20 percent (Figure 2), and 3D cross infill at 10, 15, and 20 percent (Figure 3). Gyroid produced more uniform buckling than 3D cross, but print quality at the lowest percentages was insufficient for reliable fabrication (Figure 4), and any infill above 10 percent required significant force to compress by hand. This approach was ultimately set aside. The three primary parameters for stiffness tuning were identified as print temperature, flexor length, and flexor thickness, which together provided sufficient control without the quality trade-offs of ultra-low infill.

⁷ <https://colorfabb.com/varioshore-tpu-natural>

⁸ <https://www.cnckitchen.com/blog/testing-colorfabb-varioshore-tpu-foaming-3d-printing-filament>

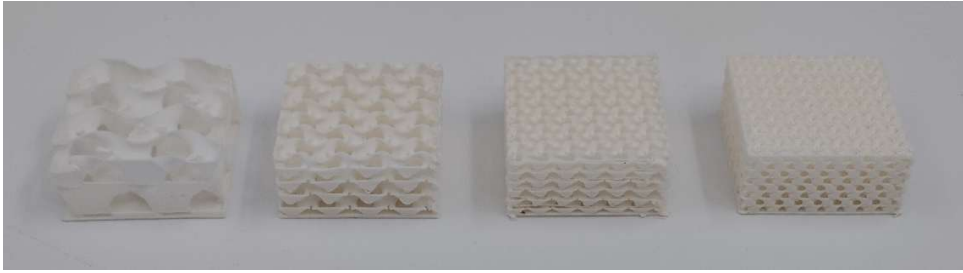


Figure 2: Gyroid infill samples at, 5, 10, 15, and 20 percent infill

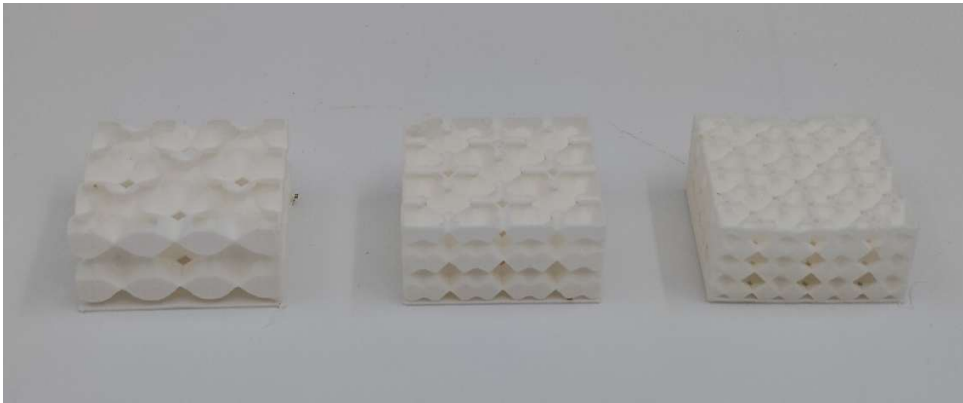


Figure 3: 3D-cross infill samples at 10, 15, and 20 percent infill



Figure 4: Low infill degrades the printing quality

The resulting lattice samples bear a structural resemblance to the flexible microstructures described in eFlesh [CITE: Pattabiraman et al. 2025], though developed independently for a different purpose.

4.1.3 Circuit Board Prototyping

Standard PCB fabrication services are too slow for rapid design iteration. An emerging technique from online maker communities (by castpixel⁹, Freche Typen¹⁰, QWZ Labs¹¹) and was therefore adopted, in which a board substrate is 3D printed with raised ridges corresponding to the non-conductive regions of the board. Copper foil is adhered to the surface, pressed into the ridges, and then sanded so that copper is removed from the raised areas while remaining in the recessed regions, leaving conductive traces that can be soldered.

To validate the technique, a simple single-sided test board was designed in KiCad (Figure 5) and imported as a STEP file into Fusion 360 (Figure 6) for substrate modelling, with ridge height set to 0.2mm to match single-layer print height. The post-printing processing steps are described in Figure 7. First the holes drilled with a 1mm drill, as these usually shrink during the printing process. Copper tape is carefully applied using a silicone brush. Then the excess copper is sanded away. After carefully inspecting the copper it is ready to be soldered. Soldering temperatures used was 300 degrees Celsius with leaded solder. The soldering time should be rather quick, otherwise the plastic will melt.

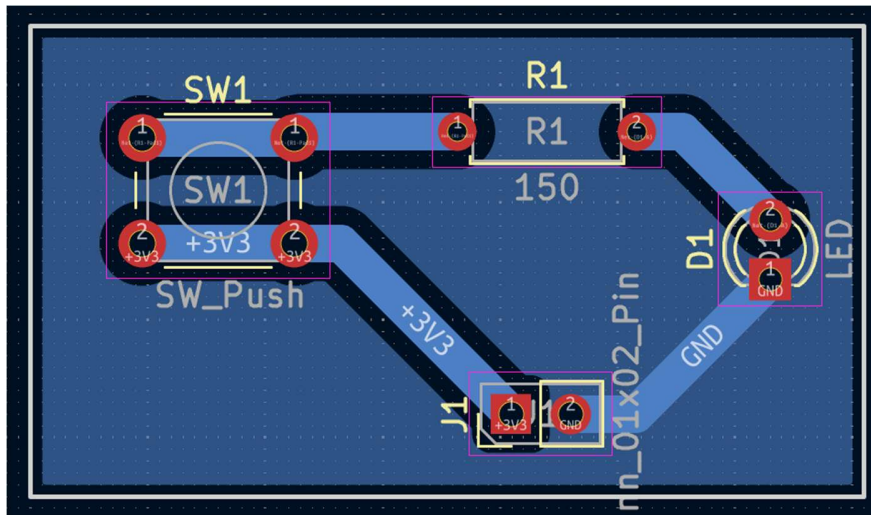


Figure 5: KiCad test PCB layout

⁹ <https://castpixel.itch.io/pcb-forge>

¹⁰ <https://www.youtube.com/watch?v=rPsZJydHG7M>

¹¹ <https://www.youtube.com/watch?v=PLliKgZKKUI>

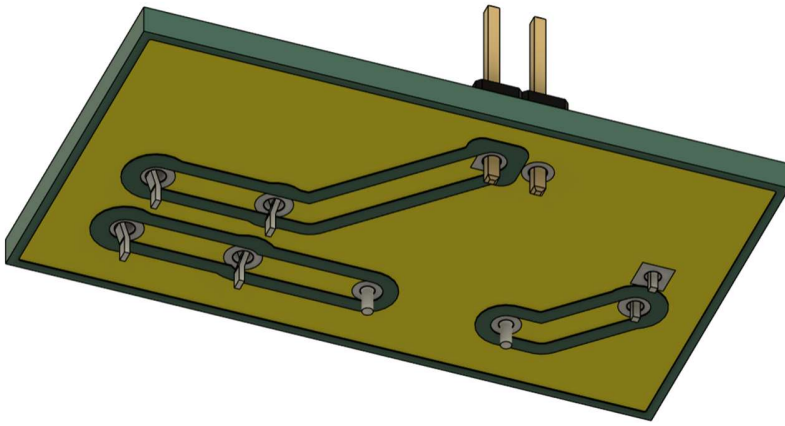


Figure 6: PCB design in Fusion 360

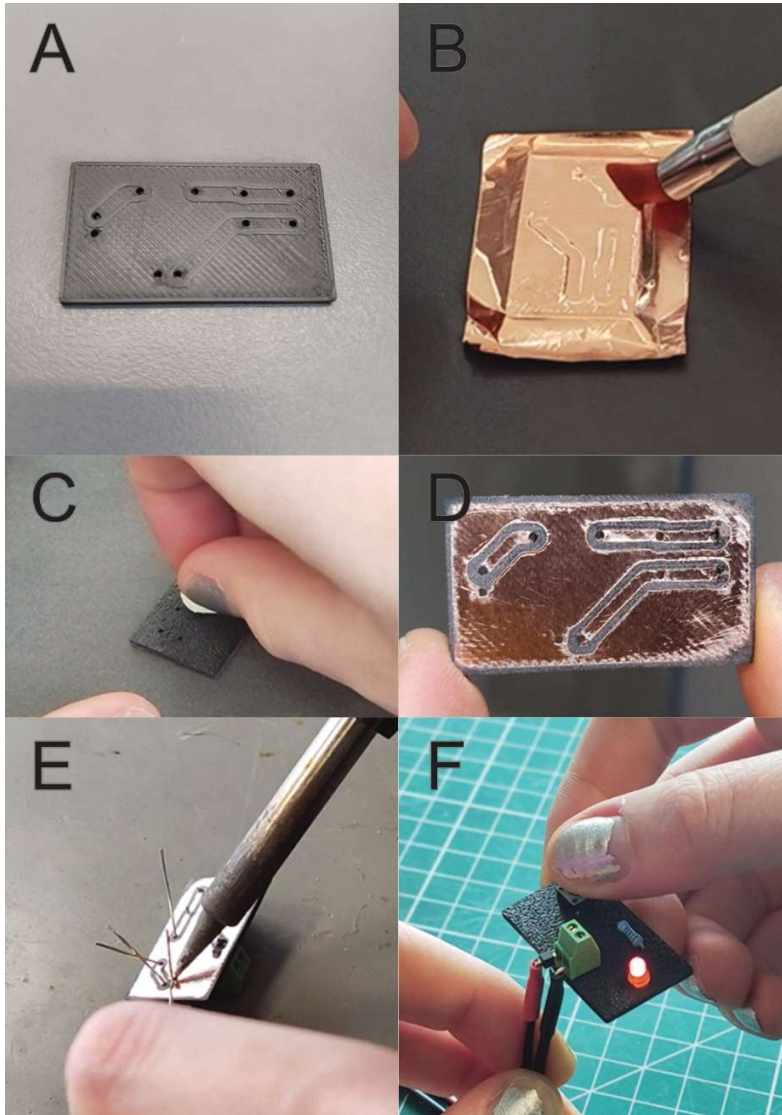


Figure 7: Manual steps for 3D-PCB; A: drill through pilot holes, B: apply copper tape, C: sand copper, D: inspect traces, E: solder, F: finished board

For the final Tabula prototype, the approach was extended to a two-sided board printed vertically rather than flat on the build plate (Figure 8). A minimum trace width of 1mm was achievable with a 0.4mm nozzle at 0.1mm layer height, and the PETG substrate withstood cleaning with isopropyl alcohol. Some reliability issues were encountered: one trace failed to adhere fully, repeated pin header connect and disconnection cycles caused trace breaks (repairable by resoldering), and a suspected short circuit in one button was attributed to the experimental fabrication process despite the PCB passing electrical continuity testing. Despite these limitations, the technique proved sufficient for prototyping purposes and shows a two sided PCB not previously demonstrated in the maker community. (Figure 9)

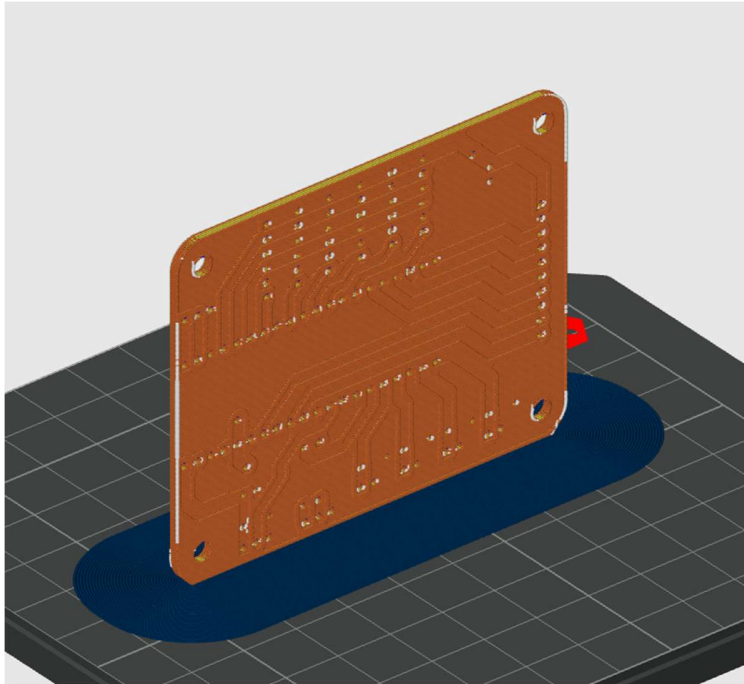


Figure 8: Print orientation of the double sided Tabula PCB

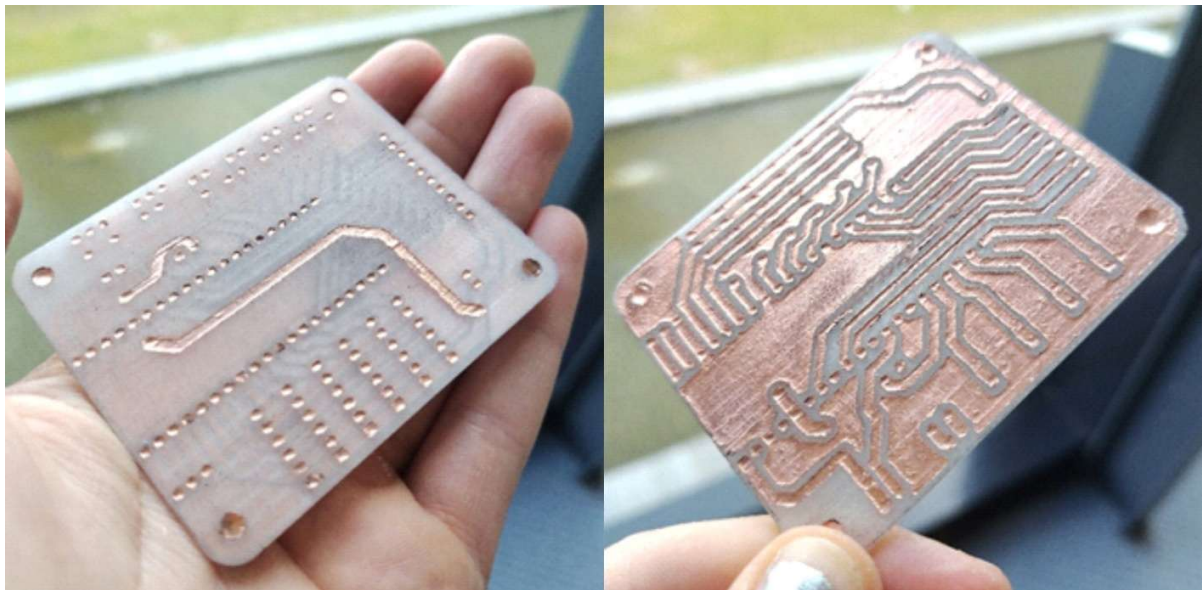


Figure 9: Finished Tabula PCB

4.2 Iterative Co-Design Process

4.2.1 Phase 1: Baseline Evaluation and Design Space Definition

The first phase had two goals: to establish an understanding of participants' musical backgrounds and instrument experience, and to introduce the concept of deformable MIDI interfaces through hands-on engagement with the Sonable prototypes. Three devices were evaluated: the Bender, the Twister, and the Jiggler. This baseline served as a reference point against which participants could form expectations and articulate what they would want from a more developed interface.



Figure 10: Sonable devices from left to right; Bender, Jiggler, Twister

Both participants read the Bender and Twister as single-parameter devices, with the Bender compared to a mod wheel. The Jiggler, with its four directional axes, was expected by both participants to output multiple independent parameters, though in its implementation it produced only one. This discrepancy between perceived affordance and actual output became an immediate design signal: a physically complex interface creates an expectation of equally complex sonic output, and a design that does not fulfill that expectation leaves expressive potential unrealized.

Aniek identified mappings to FM operators, filter cutoff, wavetable position, and detuning as appropriate use cases, and situated the devices primarily in a studio context. She noted that compared to a computer mouse, these interfaces offered more precise and expressive control for recording automation. The devices were perceived as large relative to their functional output, though she noted this could make them suitable for live performance gestures that are legible to an audience.



Figure 11: Aniek interacting with the prototypes

Marko noted the sensitivity of all three devices as a particularly appealing characteristic, comparing the responsiveness to mouse control in first-person video games. He identified the center-return behavior as a differentiating feature that enables fast flicking gestures not possible with conventional knobs or sliders, and proposed mappings to spectral and granular parameters, filter cutoff, and distortion.



Figure 12: Marko interacting with the prototypes

Both participants were asked to generate speculative interface concepts. Independently, both arrived at a similar idea: a device assigning separate controls to each finger, with individual bending or X/Y sensing per digit and a joystick at the thumb. Aniek framed this around flexibility of configuration, emphasizing that controllers typically have too many or too few features and that direct physical correspondence between input and output should be immediately legible. Marko framed his concept around compressing multi-channel control into a single hand.

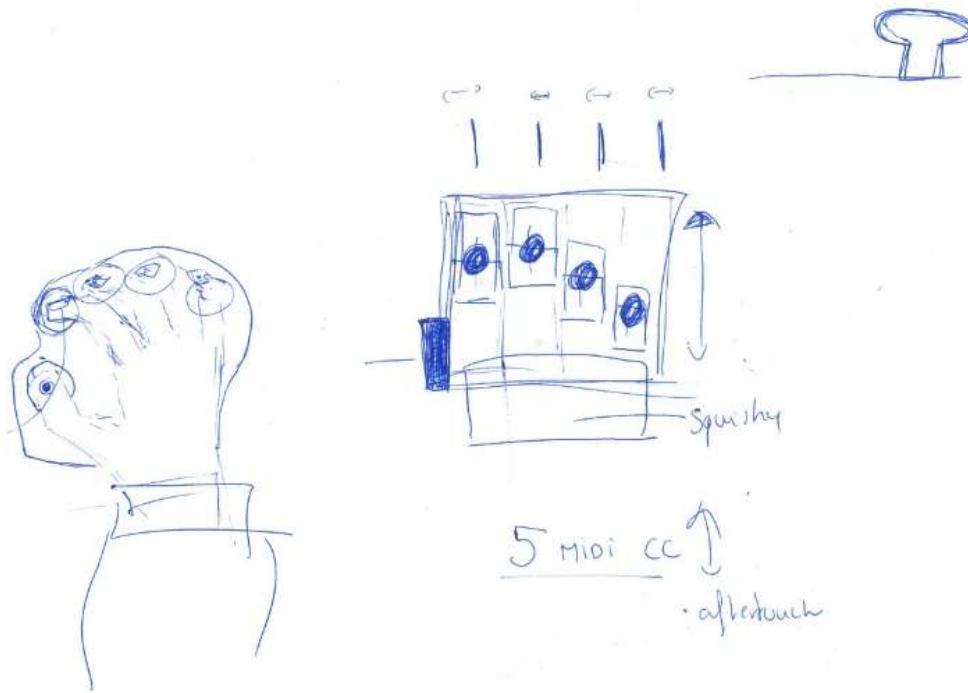


Figure 13: participant's sketch of per-finger control device

The convergence on a per-finger paradigm established one clear design direction for Phase 2. A second direction also emerged from this session: rather than distributing parameters across separate fingers, a single physically unified interface could entangle multiple parameters within a single gesture. Phase 2 would test both approaches.

4.2.2 Phase 2: Two Models of Multi-Parameter Control

Phase 2 introduced two prototype devices that embodied the two design directions identified in Phase 1. The Finger Bender assigned independently bendable controls to the index, middle, and ring fingers, with an X/Y joystick at the thumb, directly realizing participants' generative concepts. The Pressure Pad used a spring-suspended plate that registered pressure and front/back and left/right tilt as physically coupled axes, all produced by a single hand placement on a unified surface.

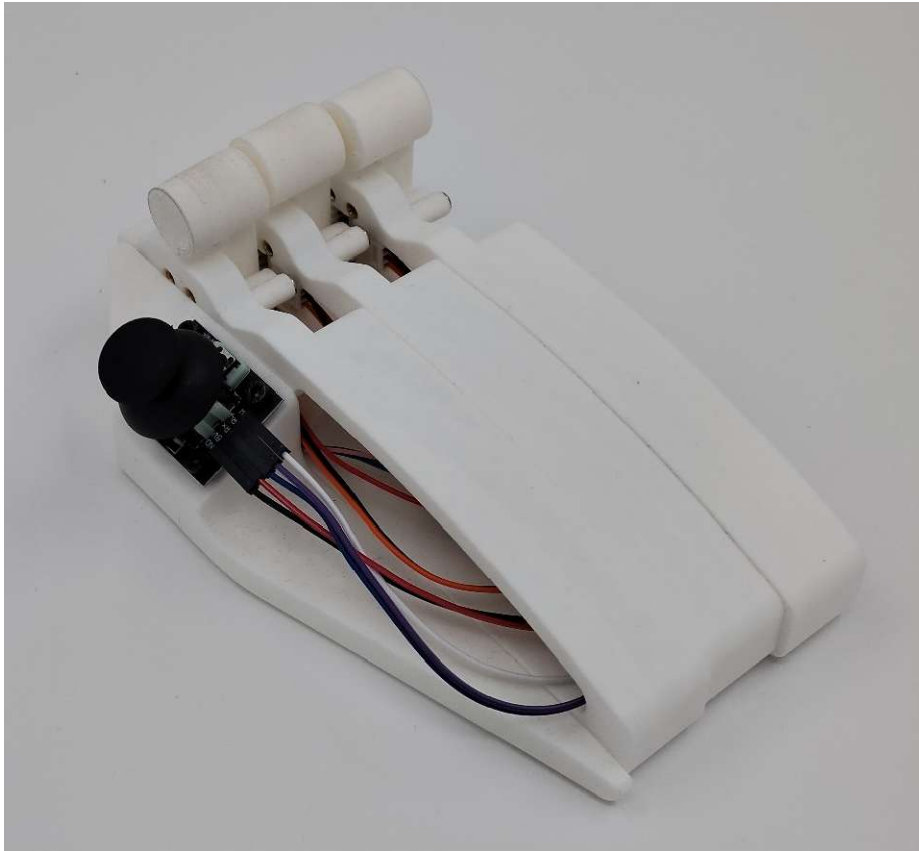


Figure 14: Finger Bender



Figure 15: Pressure Pad

A shared technical issue on both devices was addressed before evaluation: continuous MIDI output interferes with DAW parameter mapping workflows, as most DAWs reassign parameters as soon as any data is received. A hardware switch was added to both devices to suppress output to the movement endpoints only during mapping, and to disable the suppression during performance.

Aniek was unable to evaluate the Finger Bender functionally in this session due to assembly timing, but provided aesthetic feedback: she described it as mysterious in appearance and noted a similarity to her modular synthesizer due to the visible external wiring. Her evaluation focused on the Pressure Pad, which she mapped to filter cutoff (front/back tilt), reverb send (pressure), and beat repeat activation (left/right toggle). She responded positively to the simultaneous influence of multiple parameters through a single physical gesture, describing this as enabling sonic exploration through the combined action of controlled parameters. Suggested improvements included a larger pressure range, smoother tilt action, rounded plate edges, a wider plate surface, and continuity rather than toggle behavior on the left/right axis. She also proposed the ability to split the tilt axes into separately assignable parameters, drawing a comparison to the Intellijel Planar in her own setup (Figure 16).

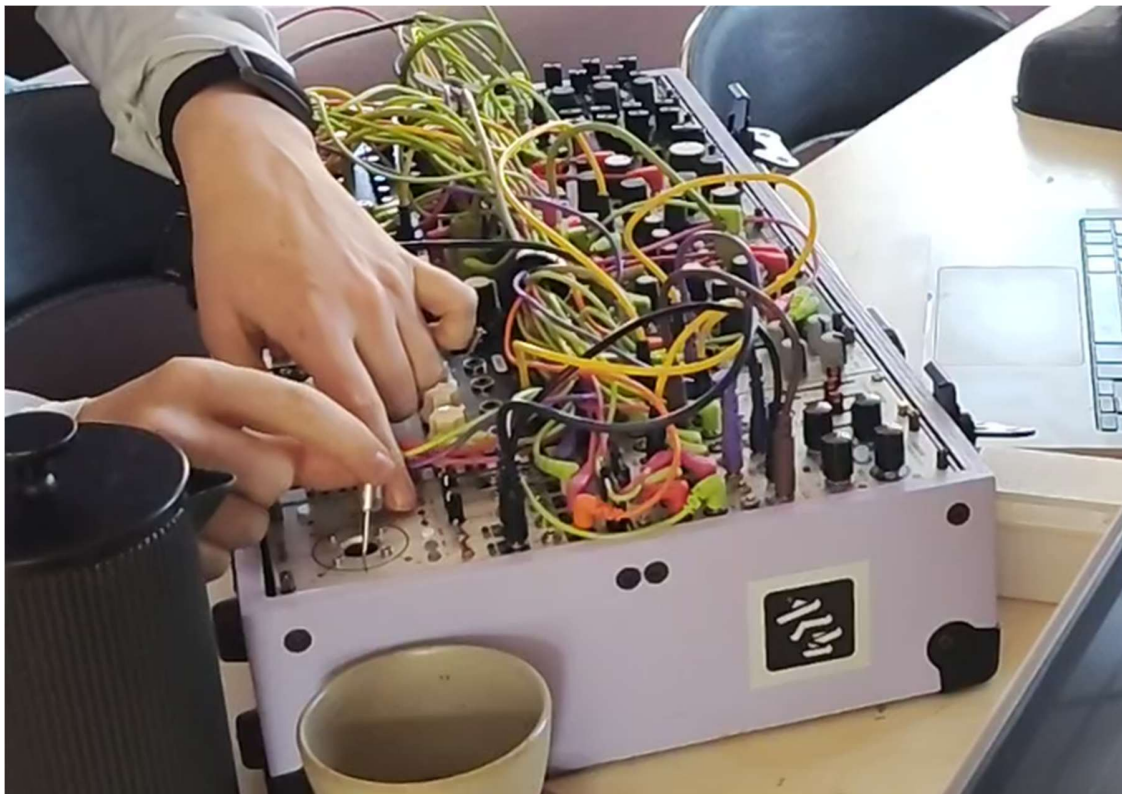


Figure 16: Aniek showing the Intellijel Planar on her modular synthesizer

Marko evaluated the Finger Bender with a full parameter mapping: playback speed on the index finger, noise modulation on the middle finger, and echo feedback and delay time on the thumb X/Y axes. He found the device physically and cognitively demanding: the limb independence required to move each finger separately is

a trained skill, and even after reducing the range of motion by half to combat fatigue, the interaction did not feel intuitive. His evaluation of the Pressure Pad was more positive. He compared it to a guitar wah pedal and identified clear applications for both studio recording and live performance. He noted that minor sensor noise on the MIDI output was perceived as adding character to the sound rather than as a flaw. As with Aniek, he preferred a continuous left/right axis, and flagged the location of the map-mode switch on the underside of the device as a significant usability problem that would interrupt workflow.



Figure 17: Marko trying to find a comfortable hand position on the Finger Bender

The Phase 2 session made the design direction clear. The Finger Bender, despite its conceptual coherence with participants' Phase 1 ideas, imposed an ergonomic and cognitive burden that worked against playful, exploratory engagement. Separating parameters across fingers required deliberate, trained control rather than intuitive experimentation. The Pressure Pad, by contrast, entangled its parameters within a single surface such that movement in one direction naturally and partially unpredictably influenced the overall sonic result. Both participants described this quality in positive terms, and Marko's observation that the parameters "control each other" pointed directly to entanglement as a design value worth pursuing. The design direction for the final prototype was the Pressure Pad, refined in response to participants' specific feedback.

4.2.3 Phase 3: Tabula

The final prototype, Tabula (Figure 18), is an iterated version of the Pressure Pad that incorporates the feedback from Phase 2. The most significant structural changes were an increased range of motion in both pressure and tilt axes; an end-stop underneath the plate (Figure 19) that allows full-depth pressing while preserving the ability to tilt; rounded plate edges; and a wider plate surface. Three top-mounted buttons were added: two to toggle between continuous and split modes on the front/back and left/right axes independently, and one to activate the mapping mode. A side button at the thumb position allows the user to freeze the plate position. The circuit board used in this prototype was the two-sided 3D-printed PCB described in section 4.1.3.



Figure 18: the final design iteration; Tabula



Figure 19: the added pressure end-stop underneath the plate, also acting as a pivot

Aniek's first impression was that the device could move in any direction without an immediately obvious primary mode of use: a surface affording tilting, pivoting, pressing, and flicking. She developed three instrument profiles during the session, mapping the device's continuous axes to parameters across a pad synthesizer, a drum effect rack, and Ableton's Granulator II plugin. During play, she described extensive crosstalk between the mapped parameters as a defining positive characteristic, comparing the quality of interaction to the Korg Kaospad and her Intellijel Planar. The entanglement of parameters through physical movement enabled playing that she noted would not be reproducible with conventional knob control, and which would typically require DAW automation to approximate. She identified the studio as the primary use context given the device's compact footprint, and noted that larger movement ranges might serve live performance better where gestural legibility to an audience is relevant. Feedback centered on the lack of mode indication for the split, map, and freeze functions: she proposed LED indicators under each button as a solution, and suggested that a logograph on each button might help communicate its function.

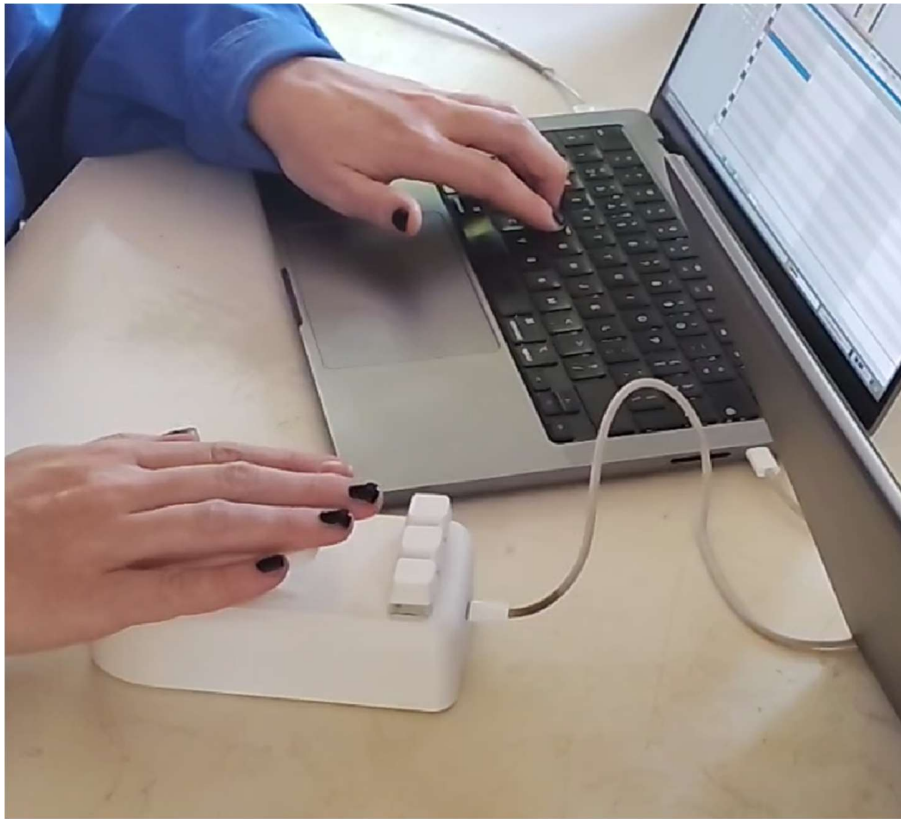


Figure 20: Aniek creating sound presets for the Tabula

Marko's first impression was that the device was inviting to touch. He noted the improvements over the Phase 2 version, particularly the pressure end-stop, which provided a physical frame of reference for tilting the plate under load. He developed three profiles mapping the device to a vocoder, a granular sampler, and an echo feedback loop. His interaction involved a wide gestural vocabulary: pressing, flicking, rapid back-and-forth movements, gradual tilting, rotating around the center pivot, vibrato-press, pressing individual corners, and sweeping arcs. Reflecting on use, he described the device as very enjoyable and noted that the physical configuration of the parameters produced a situation in which they influenced each other rather than remaining independent, enabling musical interactions that would not be possible with standard knobs. He noted that the split axis function saw little use in practice, as comparable results could be achieved through Ableton's own parameter mapping ranges. The map mode was frequently forgotten. He suggested a slimmer form factor to improve portability, while acknowledging that the current height contributes to the expressive range of pressing gestures.



Figure 21: Marko playing with the Tabula

5 FINDINGS

5.1 Theme 1: Multi-parameter Entanglement as Driver of Sonic Exploration

A pattern that was consistent in all sessions was that participants expected a physically and mechanically complex interface to produce complex sonic output accordingly. During the first session it was immediately obvious this expectation was not always met; the Jiggler in particular was thought to produce data for its X and Y axis rotation. This was identified as a missed opportunity for later iterations in the design process.

In the second session two design streams were compared that resembled different paradigms. One, inspired by both participants' designs with separated controls, was named the Finger Bender. The second, inspired by their desire for the expected complex data, was the Pressure Pad. The entangled parameters of the Pressure Pad offered the participants unique control. Aniek noted that this enabled sonic exploration. Marko noted that every movement controls the parameters simultaneously, because they are not physically separated. This contrasts with the Finger Bender, where cognitive effort is required to separately manage every aspect of sound change. Entangled control reduces cognitive load and takes away total control in favour of sonic exploration potential.

The final session presented Tabula, where the entanglement of parameters is its most prevalent feature. Aniek described the crosstalk between parameters as particularly nice, comparing it to qualities of two of her most used and favourite devices: the Korg Kaospad, and the Intellijel Planar in her modular synthesizer. She stated that the playing style of Tabula is not possible with traditional knobs and faders, and that she would typically approximate

this using DAW automation recording in the studio. Marko noted that parameters mapped to Tabula influence each other, allowing for musical interactions not possible with standard controllers.

These findings resonate with Mudd's description of entangled agency in no-input mixing [10]. It is an idea that expressive and surprising musical outcomes emerge from a configuration in which neither the performer nor the instrument is fully in control; both contribute to the sonic outcome.

5.2 Theme 2: Sensing Sensitivity as Foundation for Fine-Motor Expressivity

The sensitivity offered by the hall-effect sensors from the Sonable prototypes was quickly apparent to both participants. In the context of automation for single parameters, Aniek's studio practice is to record complex gestures using her computer mouse. During the first session she noted that the Sonable devices were more expressive and precise than her mouse. The Twister in particular was noted as suitable for subtle playing, identifying the capability to capture small movements as a relevant characteristic for studio recording. Marko drew a different comparison relating to the use of a computer mouse: the sensitivity reminded him of the sensitive control and responsiveness needed in first-person shooter games, a context in which fine motor skills are both trained and valued. Hall-effect sensing technology is also used in modern gaming controllers for this very reason.

The mechanisms exhibit a snap-back quality which allowed for certain gestures not possible with conventional controllers. According to Marko it enables fast flicking gestures, allowing for sudden changes in parameters. During the third session with Tabula, Marko remarked it was inviting to touch; likely due to its spring-loaded mechanism. This spring also gives the user continuous and inherent feedback about the position of the device. The pressure end-stop, introduced as an upgrade from the Pressure Pad to Tabula, was noted by Marko to provide a frame of reference: the user knows that full pressure has been achieved, while still being able to continue the tilting action.

This inherent feedback is a topic described by Wensveen et al. [17]: passive haptic information arising from the physical properties of the interface itself, rather than from an added display or indicator. The spring-back of the compliant mechanism and the resistance of the pressure work together provide a continuous proprioceptive signal that guides fine-motor playing without requiring the performer to monitor a visual readout.

6 DISCUSSION

6.1 Limitations of the Thematic Analysis

6.1.1 Coding from session summaries rather than transcripts

A methodological limitation of this analysis concerns what was actually coded: rather than working from verbatim session transcripts, codes were generated from session summaries. This introduces an additional layer of interpretation before formal coding even began, since writing a summary already involves selecting what is noteworthy and condensing participants' talk. Nuance such as hesitation, tone, or short-lived contradictions within a participant's account may be lost because of this. Reflexive thematic analysis is particularly well suited to semantic coding, which identifies explicit, surface-level content [4]. Working from summaries likely reinforced a semantic rather than a more latent, in-depth reading of the data. The contextual cues needed to interpret underlying assumptions are harder to recover once the data has been summarized.

At the same time, Braun and Clarke's approach does not treat the researcher's interpretive involvement as a threat to validity in the way more positivist approaches to qualitative coding do. Their original framework already

treats coding as an active, theoretically informed process rather than a mechanical search for themes that simply exist in the data [3], and their later reflexive thematic analysis work makes this more explicit still: the researcher's judgement is treated as part of the analysis, provided it is made transparent [4]. Where individual codes felt uncertain on review, the original session notes were consulted directly rather than relying on the summary alone, which partially mitigates the risk of the summary stage having silently dropped relevant data.

The decision to summarize rather than code directly from transcripts was itself a practical response to the nature of the interview data: sessions were conversational by design, and discussion frequently drifted away from the questions originally posed. Summarizing functioned as a filtering step, distilling each session down to the content most relevant to the research questions before formal coding began. A more direct way to address this in future iterations would be to shorten interview duration and rely on more tightly focused questioning, reducing the amount of off-topic material generated in the first place. This would make it more feasible to code directly from full transcripts, allowing latent-level codes to be developed with greater confidence.

6.1.2 Reflection on Fabrication: Novel Use of 3D Printing

Tabula demonstrates two novel applications of 3D printing: a 3D-printed circuit board, and 3D-printed compliant mechanisms. The 3D-printed PCB proved adequate for prototyping purposes, though it carries quality concerns already discussed in section 4.1.3. The compliant mechanisms performed well functionally, but raise a concern around material fatigue: with repeated use, the printed plastic risks losing its spring-back quality over time. Should future iterations remain committed to 3D printing as a fabrication method, more resilient filaments such as polypropylene could help mitigate this fatigue. Alternatively, silicone rubber cast in 3D-printed molds offers another route to fabricating compliant mechanisms, trading the convenience of printing the mechanism directly for a material better suited to repeated elastic deformation.

6.2 Future Work

6.2.1 Gestural vocabulary as a direction for future work

During the third session, Marko's interaction with Tabula encompassed a notably wide range of distinct gesture types: pressing, flicking, rapid back-and-forth movement, gradual tilting, rotation around the center pivot, vibrato-press, and sweeping arcs. While these gestures cannot be attributed to a particular element of Tabula, this does suggest that the combination of a deformable, spring-loaded surface with multi-parameter sensing creates interaction affordances that invite gestural exploration. This aligns with findings from related work on deformable interfaces in music [2,16]. A more systematic investigation of which gestural forms emerge represents a promising direction for future work, investigating what conditions help to facilitate rich and abundant gestures. Inspired by Mudd's findings [10], feedback synthesis would provide a promising case study to explore gesture possibilities with Tabula, as it is an exploratory sound synthesis method by nature. To gain more in-depth insight into these gestures, the data analysis should adopt more latent coding, rather than the summary-based approach used in this work.

6.2.2 Mathematically engineered parameter entanglement

Beyond entanglement that emerges incidentally from a device's physical construction - as is the case with Tabula's mechanical crosstalk - entanglement can also be understood, and designed, in purely geometric terms. Two control axes behave as fully independent parameters only when they are positioned at a right angle (90°) to

one another: a movement along one axis then has no effect on the value read from the other. As soon as the angle between the two axes departs from 90°, this independence breaks down; a movement along one axis necessarily produces some change along the other as well, because the two axes can no longer represent a point in space without their contributions overlapping. This gives entanglement a precise mathematical origin that is independent of the sensing technology used: by choosing the angle between two physical control axes, a designer can decide in advance, and tune by degree, how strongly one parameter bleeds into another. Future iterations of the Sonable devices could explore non-orthogonal axis arrangements as a deliberate design strategy for engineering entanglement, rather than treating it purely as a by-product of a particular mechanism or sensor.

7 CONCLUSION

This project explored how deformable interfaces can be designed to support musical expressivity, through an iterative co-design process that culminated in Tabula, a 3D-printed deformable MIDI controller built on hall-effect sensing. Entangling multiple sound parameters within a single physical deformation drove the unpredictable, exploratory sound-making participants valued, while separating parameters across discrete controls worked against it. High sensing sensitivity was a precondition for this expressivity, independent of whether parameters were entangled, allowing for subtle playing and manipulation of the interface. The findings suggest that for a deformable interface to be expressive, multiple sensitive sensing axis should be physically unified, entangling control, rather than presenting them separately. The project further demonstrated that 3D-printed compliant mechanisms and circuit boards are robust enough to support this kind of rapid, hands-on co-design. As this was explored with two expert musicians across three sessions, these findings are necessarily exploratory, and point toward engineered entanglement and the breadth of gesture a deformable surface invites as productive directions for future work.

ACKNOWLEDGMENTS

First and foremost I would like to thank both Aniek and Marko for their participation in this study. I would like to thank Marrit and Geert for their help with fabrication.

This deliverable was created with help from Claude Sonnet 4.6, to help guide writing, restructure the text, summarize, and fortify the academic tone of the report. Section that was excluded from processing by Claude is particularly the one containing background information of the participants (Section 3.1). Claude was also used to inspire narrowing of identified codes for the thematic analysis, as well as fortifying the thematic write-up. To allow for a close reading of the interviews, a local instance of Vibe (<https://github.com/thewh1teagle/vibe>) was used for transcription only for the researcher to be read.

REFERENCES

1. Karen Barad. 2007. *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*. Duke University Press. <https://doi.org/10.2307/j.ctv12101zq>
2. Alberto Boem, Giovanni M Troiano, Giacomo and Lepri, and Victor Zappi. 2020. Non-Rigid Musical Interfaces: Exploring Practices, Takes, and Future Perspective. *Proceedings of the International Conference on New Interfaces for Musical Expression*, 17--22. <https://doi.org/10.5281/zenodo.4813288>
3. Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2: 77–101. <https://doi.org/10.1191/1478088706qp063oa>
4. Virginia Braun and Victoria Clarke. 2019. Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health* 11, 4: 589–597. <https://doi.org/10.1080/2159676X.2019.1628806>

5. Yuan-Ling Feng, Zhaoguo Wang, Yuan Yao, Hanxuan Li, Yuting Diao, Yu Peng, and Haipeng Mi. 2024. Co-designing the Collaborative Digital Musical Instruments for Group Music Therapy. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (CHI '24), 1–18. <https://doi.org/10.1145/3613904.3642649>
6. William Gaver. 2012. What should we expect from research through design? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 937–946. <https://doi.org/10.1145/2207676.2208538>
7. Heekyoung Jung and Erik Stolterman. 2010. Material probe: exploring materiality of digital artifacts. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction* (TEI '11), 153–156. <https://doi.org/10.1145/1935701.1935731>
8. Steven Leib and Anil Çamcı. 2018. Triplexer: An Expression Pedal with New Degrees of Freedom. 263–268. <https://doi.org/10.5281/zenodo.1302577>
9. Maria C. Mannone, Eri Kitamura, Jiawei Huang, Ryo Sugawara, and Yoshifumi Kitamura. 2018. CubeHarmonic: A New Interface from a Magnetic 3D Motion Tracking System to Music Performance. 350–351. <https://doi.org/10.5281/zenodo.1302615>
10. Tom Mudd. 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* (CHI '23), 1–11. <https://doi.org/10.1145/3544548.3580662>
11. Anke van Oosterhout and Eve Hoggan. 2021. Deformation Techniques for Shape Changing Interfaces. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (CHI EA '21), 1–7. <https://doi.org/10.1145/3411763.3451622>
12. Venkatesh Pattabiraman, Zizhou Huang, Daniele Panozzo, Denis Zorin, Lerrel Pinto, and Raunaq Bhiringi. 2025. eFlesh: Highly customizable Magnetic Touch Sensing using Cut-Cell Microstructures. <https://doi.org/10.48550/arXiv.2506.09994>
13. Hanspeter Portner. 2014. CHIMAERA The Poly-Magneto-Phonic Theremin An Expressive Touch-Less Hall-Effect Sensor Array. 501–504. <https://doi.org/10.5281/zenodo.1178909>
14. Nathan Renney and Benedict Gaster. 2024. Insights Into How Digital Luthiers Approach Design. 505–515. <https://doi.org/10.5281/zenodo.13904935>
15. Elizabeth B.-N. Sanders and Pieter Jan Stappers. 2008. Co-creation and the new landscapes of design. *CoDesign* 4, 1: 5–18. <https://doi.org/10.1080/15710880701875068>
16. Giovanni Maria Troiano, Esben Warming Pedersen, and Kasper Hornbæk. 2015. Deformable Interfaces for Performing Music. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15), 377–386. <https://doi.org/10.1145/2702123.2702492>
17. S. A. G. Wensveen, J. P. Djajadiningrat, and C. J. Overbeeke. 2004. Interaction frogger: a design framework to couple action and function through feedback and feedforward. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques* (DIS '04), 177–184. <https://doi.org/10.1145/1013115.1013140>
18. Jianing Zheng, Nick Bryan-Kinns, and Andrew P. McPherson. 2022. Material Matters: Exploring Materiality in Digital Musical Instruments Design. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference* (DIS '22), 976–986. <https://doi.org/10.1145/3532106.3533523>
19. John Zimmerman, Jodi Forlizzi, and Shelley Evenson. 2007. Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07), 493–502. <https://doi.org/10.1145/1240624.1240704>
20. Super Size Me: Interface Size, Identity and Embodiment in Digital Musical Instrument Design | Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems. Retrieved June 15, 2026 from <https://dl.acm.org/doi/10.1145/3491102.3517626>

A INTERVIEW AND THEMATIC ANALYSIS APPENDICES

In the appendix section, three levels of Appendix headings are available.

A.1 Recruitment Email

Dear reader,

I would like to invite you to participate in a user study for my graduation project. It concerns my master thesis for the Industrial Design Master at the Eindhoven University of Technology. In my thesis, I am developing soft midi interfaces for musical expressivity. You are being recruited because of your experience in electronic music production and recording.

The user study will consist of three sessions. The first session is a co-design session where together, we will come up with a hypothetical design for an interface. Second session will be a feedback session prototype version of the hypothetical interface, so it can be revised (if needed). The third session will involve you interacting with a functional prototype of the interface.

Each session should last 2 hours and can be done at your preferred (home-)recording studio. I will visit you at your preferred location.

Possible dates for each session (whatever suits you best):

1st session: 16th of March - 27th of March

2nd session: 13th of April - 17th of April

3rd session: 28th of April - 8th of May

During the sessions, I will interview you about various things, like your background in music, but most importantly your design choices during the co-creation session. These interviews will be audio recorded and transcribed. During the 3rd session, video will also be recorded to investigate how you interact with the device. This study will be conducted on a voluntary basis. You can withdraw at any time.

As a reward for participating in the study, you will receive a functional copy of the interface we co-designed during the study.

Let me know if you are willing to participate. Don't hesitate to ask any questions!

Best regards,

Hannah Eikens

A.2 Interview Questions Session 1

What is your musical background?

(how many years)

(what genres)

(bands? or alone?)

(releases)

(Music Technology)

What are your main musical influences (genres and sounds, gear)?

What instruments do you play?

(for how long?)

(any favorite?)

(what characteristics about it do you like)

(what characteristics don't you like)

What is your favorite audio hardware? (synth, fx processor)

(what about it)

(how does it match your sound)

Familiarisation

Interact with Sonable M21 devices:

participant observes prototypes

Ask participants to order the devices from most liked to least liked

Why this order?

What are your ideas about these devices in a musical context? What sounds would they make? How are they manipulated?

Interaction and Interpretation

Play with prototypes in musical context (connect to their laptop) (use with ableton to map them to sound controls)

What sound do these devices make?

How did you map the sound and why? (based on material qualities, what sound qualities)

Compare and Contrast

How do the materials compare to other DMIs?

How can these devices be used in your own musical workflow

Have reference images of different kinds of deformation

Co-creation:

pick a specific sound synthesis or manipulation technique to build your interface for.

Using crafting materials, create a speculative interface for this synthesis technique.

interview:

explain what you've made

Why the shape? Why texture?

How are the sound parameters mapped to the deformation? (can use colored stickers to indicate which part is which)

A.3 Interview Questions Session 2

Any thoughts since last session?

Introduction to the new device:

Inspect the device visually. What do you think it does, how does it work?

Inspect the device tacitly. How does this change your view its functionality?

Introduce Pressure Pad

Explain the device

You are now instructed to make as many instrument/effect racks that use the device as you can (1 hour)

You can explain your thought process as you go if you want. You can also explain it at the end.

Perform using the device as you go, and make it inform your next iterations.

Now I would like you to explain what you made and explain how and/or if the device made you pick specific effects and mappings.

Can you demonstrate playing it? What are the different gestures that you use for each instrument rack.

Reflect on the device. What's your experience of using it?

What do you like? What do you dislike.

What does the device do that other devices you use cannot.

Introduce Finger Bender

Explain the device

You are now instructed to make as many instrument/effect racks that use the device as you can (1 hour)

You can explain your thought process as you go if you want. You can also explain it at the end.

Perform using the device as you go, and make it inform your next iterations.

Now I would like you to explain what you made and explain how and/or if the device made you pick specific effects and mappings.

Can you demonstrate playing it? What are the different gestures that you use for each instrument rack.

Reflect on the device. What's your experience of using it?

What do you like? What do you dislike.

What does the device do that other devices you use cannot.

A.4 Interview Questions Session 3

Introduction to the new device:

Inspect the device visually. What do you think it does, how does it work?

Inspect the device tacitly. How does this change your view its functionality?

Explain the device

You are now instructed to make as many instrument/effect racks that use the device as you can (1 hour)

You can explain your thought process as you go if you want. You can also explain it at the end.

Perform using the device as you go, and make it inform your next iterations.

Now I would like you to explain what you made and explain how and/or if the device made you pick specific effects and mappings.

Can you demonstrate playing it? What are the different gestures that you use for each instrument rack.

Reflect on the device. What's your experience of using it?

What do you like? What do you dislike.

What does the device do that other devices you use cannot.

In the current form it is aimed at use in the studio. If you were to use this for a live performance, would you want to change anything about it?

What do you think the device is worth in it's current state? Would you be willing to buy this with your own money? If yes, why? If not, what would need to be improved? Does the fact that it is a 3d-printed device bother you at all?

What are features that the device doesn't have, that would drastically improve the value for you? If it has these features, what would you be willing to pay for it?

A.5 Code Overview

Code	Keywords	Description	Session	Participant	Theme
Resting position / snap-back	center, middle, snap-back	The mechanism's return to a neutral state after deformation	S1	Aniek	Theme 2
Application outside music	film, flight controller, car	Possible application areas beyond music control	S1	Aniek	Contextual
Sweet spot of sounds	sweet spot, range	The range of parameter values that sound musically appropriate or intentional	S1	Aniek	Theme 1
More expressive than mouse	mouse, automation	The Sonable devices offer more precise and expressive control than recording automation with a mouse	S1	Aniek	Theme 2
Unexpected outcome from mapping	creative outcome, accident	A wrong or unintended mapping produces an unexpectedly pleasing sonic result	S1	Marko	Theme 1
High sensitivity / learning curve	max sensitivity, mastery	High sensitivity rewards trained fine motor control; mastery unlocks greater expressive freedom	S1	Marko	Theme 2
Novelty	unique, surprise	What distinguishes the device from existing controllers	S1	Marko	Contextual
Use in music	automation, live, studio, desk	Contexts in which the device could be used in musical practice	S1	Marko	Contextual
Aesthetics of the device	textured finish, surface	The visual and tactile appearance of the device	S1	Marko	Contextual
Parameter split request	split axis, separate parameters, Intellijel Planar	Desire to assign tilt directions as independent parameters for greater mapping flexibility	S2	Aniek	Theme 1
Haptic comfort of pressure	pressure feel, force, distance	The tactile quality of pressing the plate; force and depth feel appropriate or need adjustment	S2	Aniek	Theme 2
Open-ended physical affordance	any direction, tilt, pivot, flick, press	The device does not prescribe a single mode of interaction; multiple gesture types are possible	S3	Aniek	Theme 1
Not reproducible with knobs	automation, conventional control, unique	The playing style enabled by the device could not be achieved with standard knob or slider control; DAW automation would be needed to approximate it	S3	Aniek	Theme 1
Interface legibility	logograph, button function, label	The function of controls is not immediately legible from their appearance	S3	Aniek	Contextual
Tactile invitation	inviting to touch, first impression	The physical form of the device invites handling and exploration on first contact	S3	Marko	Theme 2
Haptic reference from end-stop	frame of reference, stop, pressure	The pressure end-stop provides a tactile anchor that makes tilting under pressure feel controlled and predictable	S3	Marko	Theme 2
Gestural vocabulary	pressing, flicking, vibrato, arcs, pivot	The device supports a wide range of distinct gesture types, each producing different sonic outcomes	S3	Marko	Contextual
Redundant feature (split axis)	split, Ableton mapping, unused	The split axis function duplicates what can be achieved in Ableton and was rarely used in practice	S3	Marko	Contextual
Entangled parameter control valued	combined parameters, simultaneous, crosstalk, entangled, control each other, not physically separated, sonic exploration	A single physical gesture simultaneously and partly unpredictably affects multiple sound parameters; both participants described this entanglement as a positive, exploratory quality rather than a defect, on both the Pressure Pad (S2) and Tabula (S3). Replaces: Multi-parameter sonic exploration, Entangled parameters preferred, Parameter crosstalk as positive, Parameters influence each other.	S2, S3	Aniek, Marko	Theme 1
Mode feedback friction	map mode, mode switch, workflow, forgotten, LED, feedback absent, split, freeze	Insufficient feedback about the device's current mode caused workflow disruption or being forgotten; persisted across Pressure Pad (S2) and Tabula (S3). Replaces: Mapping mode friction, Mode indication absent, Mode awareness problem.	S2, S3	Aniek, Marko	Contextual
Comparison to known/owned gear	Intellijel Planar, Korg Kaospad, Arturia Prophet-VS, wah pedal, throttle, modular synthesizer, vector synthesis	Device compared to instrument/controller participant already owns; Theme 1 when comparison is to multi-parameter/vector gear (S1, S3). Contextual when purely aesthetic/use-case (S2). Replaces: Resemblance to own gear, Aesthetic resemblance to own gear, Familiar interface comparison.	S1, S2, S3	Aniek, Marko	Theme 1 (S1,S3) / Contextual (S2)
Sensor noise as expressive quality	unstable, fluctuations, controlled chaos, charm, noise, expressive imperfection	Slight hall-effect sensor noise was first noted in S1 as a device characteristic, then explicitly reframed in S2 as a positive, characterful quality ("charm") rather than a defect; concerns the haptic/sensing quality of the device, not parameter entanglement. Replaces: Sensor noise (S1), Sensor noise as expressive quality (S2 — name retained).	S1, S2	Marko	Theme 2
Portability / form factor concern	carry, size, desk space, slim, portable, backpack, thin, footprint, studio	Device size/thickness limits portability for touring or live use: the compact footprint makes a studio desk the most natural primary use context, while a slimmer profile would improve carry without sacrificing expressive range. Replaces: Portability concern (S1 Aniek), Studio as primary context (S3 Aniek), Form factor concern (S3 Marko).	S1, S3	Aniek, Marko	Contextual
Live performance gesture legibility	big gestures, expressive, watch, crowd, big movement, legible	Larger gesture/movement ranges during live performance make the relationship between gesture and sound change visible and legible to an audience. Replaces: Size legible for live performance (S1), Live performance gesture legibility (S3 — name retained).	S1, S3	Aniek	Contextual
Ergonomic and motor-control limitations	comfort, fit, fatigue, palm rest, plate width, rounded edges, ambidextrous, finger independence, range of motion, trained skill	Physical comfort, fit, and motor-control demands affecting sustained/comfortable use of the devices; describes physical/bodily strain rather than evidence about sensing sensitivity itself. Replaces: Ergonomic improvements needed (S2 Aniek, S2 Marko, S3 Aniek, S3 Marko), Motor independence barrier (S2 Marko), Fatigue from range of motion (S2 Marko).	S2, S3	Aniek, Marko	Contextual
Sound mapping	FM, filter, filter cutoff, reverb, delay, wavetable, tuning, oscillator, spectral scan	Which synthesizer or effect parameter can be mapped to the physical controller — spans FM operators, filters, reverb/delay sends, wavetable position, and detuning (Aniek's framing) and filter cutoff, spectral/granular scan position, and reverb (Marko's framing). Consolidates two duplicate per-participant S1 definitions of the same code; reused via inline tagging in S2 and S3 without a new table row.	S1, S2, S3	Aniek, Marko	Theme 1
Controller axis/gesture	parameter, axis, multi-parameter, bending, rotation, translation, XY grid, three-dimensional	How a movement axis relates to one or more control parameters. Consolidates two duplicate per-participant S1 definitions of the same code.	S1	Aniek, Marko	Theme 1
Expected control/gesture	assumed, affordance, visual inspection, expected, more	Expected gesture affordances based on visual inspection before use. Consolidates two duplicate per-participant S1 definitions of the same code. On audit, tagged Theme 1 for the Jiggler (S1) and Finger Bender (S2, Aniek) instances, where multiple/entangled parameters were expected; tagged Contextual for the Bender/Twister (S1, both participants) instances, where only a single parameter was expected.	S1, S2	Aniek, Marko	Theme 1 (S1,S2) / Contextual (S1)
Small gestures and movements	fine, expressive, small, sensitive, slight movement, flick, snap	The device's ability to capture subtle, fine-grained movements, including the kinetic/haptic qualities (e.g., the Jiggler's center-return enabling fast flicking) that support small-gesture, expressive play. Consolidates two duplicate per-participant S1 definitions plus Marko's separate "Gesture characteristics" code.	S1, S3	Aniek, Marko	Theme 2
Continuous vs. discrete control	toggle, continuous, on/off, binary switching	Preference for continuously variable control over binary toggling on a given control axis (the Pressure Pad's left/right beat-repeat switch). Consolidates two duplicate per-participant S2 definitions of the same code.	S2	Aniek, Marko	Theme 1

A.5 Code Theme Mapping

Theme 1: Multi-parameter Entanglement as Driver of Sonic Exploration		
Code	Session(s)	Participant(s)
Controller axis/gesture	S1	Aniek, Marko
Expected control/gesture (Theme 1 instances — Jiggler/Finger Bender)	S1, S2	Aniek, Marko
Sound mapping	S1, S2, S3	Aniek, Marko
Sweet spot of sounds	S1	Aniek
Unexpected outcome from mapping (unconfirmed instance — verify)	S1	Marko
Comparison to known/owned gear (Theme 1 instances)	S1, S3	Aniek
Entangled parameter control valued	S2, S3	Aniek, Marko
Continuous vs. discrete control	S2	Aniek, Marko
Parameter split request	S2	Aniek
Not reproducible with knobs	S3	Aniek, Marko
Open-ended physical affordance	S3	Aniek
Theme 2: Sensing Sensitivity as Foundation for Fine-Motor Expressivity		
Code	Session(s)	Participant(s)
Small gestures and movements	S1, S3	Aniek, Marko
Resting position / snap-back	S1, S2	Aniek, Marko
High sensitivity / learning curve	S1	Marko
Sensor noise as expressive quality	S1, S2	Marko
Haptic comfort of pressure	S2	Aniek
More expressive than mouse	S1	Aniek
Tactile invitation	S3	Marko
Haptic reference from end-stop	S3	Marko
Contextual Codes (background / design quality / use context)		
Code	Session(s)	
Application outside music	S1	
Novelty	S1	
Aesthetics of the device (unconfirmed instance — verify)	S1	
Use in music	S1, S2, S3	
Expected control/gesture (Contextual instances — Bender/Twister)	S1	
Comparison to known/owned gear (Contextual instances)	S2	
Mode feedback friction	S2, S3	
Portability / form factor concern	S1, S3	
Live performance gesture legibility	S1, S3	
Ergonomic and motor-control limitations	S2, S3	
Interface legibility	S3	
Redundant feature (split axis)	S3	
Gestural vocabulary	S3	

A.6 Coded Session Summaries

Annotated Session Summaries

Code labels appear in **[brackets]** immediately after the passage they apply to. Codes contributing to Theme 1 (multi-parameter entanglement) are marked **[T1]**, codes contributing to Theme 2 (sensing sensitivity) are marked **[T2]**, and contextual codes are marked **[C]**.

Session 1

Aniek

For two of the devices, the Bender and the Twister, the participant expected each device to correspond to a single sound parameter. **[Expected control/gesture - C]** The Bender was deemed similar to a mod wheel on a piano and seemed logical for adjusting parameters; it also looked like something used in a car or an aeroplane. **[Application outside music - C]** The Twister was perceived to have a limited range of motion, making it more suitable for playing subtle gestures. **[Small gestures and movements - T2]**

The Jiggler was expected to produce two mappable parameters, given its four distinct movement directions. **[Expected control/gesture - T1]** **[Controller axis/gesture - T1]** It was compared to existing gear the participant owned: the Intellijel Planar, the Korg Kaospad, and the Arturia Prophet-VS. **[Comparison to known/owned gear - T1]**

Sound parameters identified as suitable for mapping included FM operators, filters, generic FX controllers, wavetable position, and detuning. **[Sound mapping - T1]** The range of each mapped parameter was adjusted so it could be played within a sweet spot, ensuring sounds produced were always audible and musically appropriate. **[Sweet spot of sounds - T1]**

The devices were found most useful in a studio setting. **[Use in music - C]** Typically, the participant recorded automation using the computer mouse; these devices were described as more expressive and precise than the mouse. **[More expressive than mouse - T2]** The size of the devices was deemed quite large relative to their functional output **[Portability / form factor concern - C]**, though this was noted as potentially making them interesting for live performance, where bigger gestures can make sound changes legible to an audience. **[Live performance gesture legibility - C]**

The participant generated two speculative concepts. The human-centered concept described a device registering input from all five fingers: bendable controls at each fingertip with a joystick at the thumb, adjustable for individual anatomy, and supporting side-to-side motion and pressure detection. **[Controller axis/gesture - T1]** **[Small gestures and movements - T2]** A common concern raised was that MIDI controllers typically have too many or too few features. A related concern was that controllers are often too large or too small. **[Portability / form factor concern - C]**

Marko

Two devices, the Bender and the Twister, were expected to correspond to a single parameter each. **[Expected control/gesture - C]** The Jiggler was expected to output more than one parameter: at minimum X (left/right) and Y (front/back), and possibly also Z (pulling). **[Expected control/gesture - T1]** **[Controller axis/gesture - T1]**

All devices exhibited slight noisy fluctuations in their measured parameters. **[Sensor noise as expressive quality - T2]** The devices were described as very sensitive, which the participant appreciated - comparing the responsiveness to mouse control in a first-person shooter video game. **[Small gestures and movements - T2]** **[High sensitivity / learning curve - T2]** The center-return behavior of the Jiggler was noted as enabling fast flicking gestures, producing sudden parameter changes not possible with conventional knobs or sliders. **[Resting position / snap-back - T2]** **[Small gestures and movements - T2]**

Sound parameters proposed for mapping included spectral and granular scan position, filter cutoff, and distortion level. **[Sound mapping - T1]** The devices were considered unique compared to existing controllers, though the Bender was noted as similar to a mod wheel. **[Novelty - C]** Use in both the studio and during live performance was identified. **[Use in music - C]**

The participant's own concept described a device with separate controls for each finger: individual X/Y control and pressure sensing per finger, with a very small range of motion - the idea being to compress multi-channel control into a single hand. **[Controller axis/gesture - T1]** **[Small gestures and movements - T2]**

Session 2

Aniek

Reflecting on the prior session, the participant noted that usability of their own gear should be immediate and self-explanatory, with minimal menu-diving. Sound possibilities were described as the primary concern, ahead of interface ergonomics.

Finger Bender: The device was not yet functional, so evaluation was limited to aesthetics and feel. The participant described it as mysterious in appearance, and noted a similarity to her own modular synthesizer due to the visible external wiring. **[Comparison to known/owned gear - C]** The three finger controls clearly suggested three distinct sound parameters. **[Expected control/gesture - T1]** The range of motion appeared good. The visible internal structure (holes in the sides) was found visually appealing.

Pressure Pad: The device was mapped to multiple parameters simultaneously: front/back tilt to filter cutoff, pressure to reverb send, and left/right toggle to activate beat repeat. **[Sound mapping - T1]** The fact that the device controlled multiple parameters at once was described as particularly appreciated, enabling sonic exploration through the combined effect of controlled parameters. **[Entangled parameter control valued - T1]** The spring-loaded return to a center position was noted as similar to a mod wheel. **[Resting position / snap-back - T2]**

The pressure tactility was described as pleasant, with appropriate force required. **[Haptic comfort of pressure - T2]** Suggested improvements included increasing the pressure depth **[Haptic comfort of pressure - T2]**, rounding the sharp plate edges **[Ergonomic and motor-control limitations - C]**, widening the plate **[Ergonomic and motor-control limitations - C]**, and making the left/right axis continuously variable rather than a binary toggle. **[Continuous vs. discrete control - T1]** The participant also proposed the option to split the tilt axes into separately assignable parameters, drawing a comparison to the Intellijel Planar. **[Parameter split request - T1]** The map mode switch, located on the underside of the device, was flagged as disrupting workflow. **[Mode feedback friction - C]**

Marko

Reflecting on the prior session, the participant was curious whether the per-finger concept would work as well in practice as it had seemed in theory.

Finger Bender: Aesthetically, the device resembled a throttle control. **[Comparison to known/owned gear - C]** The ergonomics were identified as needing improvement: the palm rest was too low and not cupped enough. **[Ergonomic and motor-control limitations - C]** The device was mapped to playback speed on the index finger, noise modulation on the middle finger, and echo feedback and delay time on the thumb X/Y axes. **[Sound mapping - T1]** Playing the thumb and fingers together simultaneously was found non-intuitive and required practice. **[Ergonomic and motor-control limitations - C]** The device caused hand fatigue; halving the range of motion during the session helped reduce this **[Ergonomic and motor-control limitations - C]**, but did not resolve the intuitiveness problem. Moving each finger independently was identified as a trained skill akin to limb independence. **[Ergonomic and motor-control limitations - C]**

Pressure Pad: Visually, the device resembled a guitar wah pedal. **[Comparison to known/owned gear - C]** The expected parameter outputs were X and Y control. Slight noise was present on the MIDI parameters, which the participant described as a positive quality - adding a kind of charm to the sound. **[Sensor noise as expressive quality - T2]** The left/right control was preferred as a continuous parameter rather than a toggle. **[Continuous vs. discrete control - T1]** The use case was identified as similar to a wah pedal: adding expressive options for both studio recording and live performance. **[Use in music - C]** The plate was noted as needing to be wider to fit all fingers. **[Ergonomic and motor-control limitations - C]** The map mode switch on the underside of the device was flagged as a significant usability problem. **[Mode feedback friction - C]**

The participant reflected that with the Pressure Pad, one can create control configurations where every movement controls the other parameters simultaneously, because the parameters are not physically separated. **[Entangled parameter control valued - T1]**

Session 3

Aniek

The first impression of Tabula was that it could move in any direction: a surface affording tilting, pivoting, pressing, and flicking, without an immediately obvious primary mode of interaction. **[Open-ended physical affordance - T1]**

Three instrument profiles were developed during the session. The first mapped front tilt to reverb/delay send, back tilt to distorted delay send, right tilt to shimmer reverb/delay, and left tilt to the coarse frequency of FM operators. The second profile mapped beat repeat to pressure, beat division to front tilt, DJ filter to left/right tilt, and delay to back tilt. The third profile used Granulator II: scan position mapped to left/right, pressure to granular scan, and grain size to front/back. **[Sound mapping - T1]**

During play, the participant noted extensive crosstalk between the mapped parameters, which she described as particularly enjoyable - comparing this quality to the Korg Kaospad and the Intellijel Planar. **[Entangled parameter control valued - T1]** **[Comparison to known/owned gear - T1]** The device was noted as allowing both very subtle playing and the searching out of extremes in sound. **[Small gestures and movements - T2]** **[Open-ended physical affordance - T1]** The way parameters were played through the Tabula was described as not possible using conventional knob control; this type of expressive control would typically be programmed in the DAW as automation. **[Not reproducible with knobs - T1]**

The studio was identified as the primary use context given the device's compact footprint. **[Portability / form factor concern - C]** For live performance, larger movement ranges would be preferable to make the relationship between gesture and sound change legible to an audience. **[Live performance gesture legibility - C]**

Several improvements were identified: the absence of feedback indicating the current mode (split, map, or freeze) was a clear issue, with LED indicators under each button proposed as a solution. **[Mode feedback friction - C]** The button functions did not communicate their purpose; a logograph on each button was suggested. **[Interface legibility - C]** The device was not ambidextrous, which was noted as a limitation for switching between right- and left-hand use. **[Ergonomic and motor-control limitations - C]**

Marko

The first impression of Tabula was that it was inviting to touch. **[Tactile invitation - T2]** Compared to the previous version, the added buttons were easily reachable. **[Ergonomic and motor-control limitations - C (positive)]** The improved size, range of motion, and rounded corners were all noted positively. The pressure end-stop was highlighted as particularly helpful: it provided a physical frame of reference, making it easier to tilt the plate while pressing it. **[Haptic reference from end-stop - T2]**

Three profiles were developed: the first mapped vocoder depth to front/back tilt, vocoder sensitivity to left tilt, and formant to pressure; the second mapped sample position to left/right tilt, grain size to front/back tilt, and random to pressure; the third mapped delay time to left/right tilt and feedback to front/back tilt. **[Sound mapping - T1]**

The gestural vocabulary observed during the session was wide: pressing, flicking, rapid back-and-forth movement, gradual tilting, rotation around the center pivot, vibrato-press, pressing individual corners, and sweeping arcs from side to side. **[Gestural vocabulary - C]**

Reflecting on use, the participant described the device as very enjoyable. He noted that one can create control configurations where every movement controls other parameters simultaneously, because the parameters are not physically separated from each other. **[Entangled parameter control valued - T1]** This enables musical interactions not possible with standard knob control. **[Not reproducible with knobs - T1]**

The split axis function was noted as rarely used in practice, as comparable effects can be achieved using Ableton's parameter mapping ranges. **[Redundant feature (split axis) - C]** The map mode was frequently forgotten, indicating insufficient mode feedback. **[Mode feedback friction - C]** The participant suggested a slimmer body to improve portability, while acknowledging that the current height contributes to the expressive range of pressing gestures. **[Portability / form factor concern - C]** He also noted potential for use in audio-visual performance contexts. **[Use in music - C]**

A.7 ERB Approval



Date
2026-03-04

Our reference
ERB-769

Ethical review board TU/e

T +31 (0)40 247 6259
ethics@tue.nl

intranet.tue.nl/ethics

Ethical review research proposal with reference ERB-769

Dear ,

It is a pleasure to inform you that the Ethical Review Board (ERB) has discussed and preliminarily approved your application with reference ERB-769.

Furthermore, the Board wants to draw your attention to the terms and conditions in the appendix.

Success with your research!

Sincerely,

A handwritten signature in black ink, appearing to read 'D. Lakens'.

Dr. D. Lakens
Chair Ethical Review Board TU/e

The ERB retains the right to revise its decision regarding the implementation and the WMO1/WMH2 status of any research study in response to changing regulations, research activities, or other unforeseen circumstances that are relevant to reviewing any such study. The ERB shall notify the principal researcher of its revised decision and of the reasons for having revised its decision.
1WMO: Law on Medical Scientific Research Involving Human Beings (in Dutch: Wet medisch-wetenschappelijk onderzoek met mensen)
2WMH: Medical Device Directive (in Dutch: Wet op de medische hulpmiddelen)

Page
1 of 2

Date
2026-03-04

Our reference
ERB-769



APPENDIX 1

Terms and conditions

Amendments

When considerable amendments are made to the design of the study or educational activity, or when the time period between ERB approval and start of the study is longer than one year, please consult the ERB.

Privacy and research data management

The ERB would like to point out that collecting, handling and storing personal information is subject to the General Data Protection Regulation. Please visit TU/e intranet for the latest information and regulations on www.tue.nl/rdm

B T&R MDC ARTIFACTS, CODE AND ELECTRONICS APPENDICES

In the appendix section, three levels of Appendix headings are available.

B.1 Forked Adafruit MLX90395 Library (implemented SPI communication)

```
#ifndef ADAFRUIT_MLX90395_H
#define ADAFRUIT_MLX90395_H

#include "Arduino.h"
#include <Adafruit_BusIO_Register.h>
#include <Adafruit_I2CDevice.h>
#include <Adafruit_SPIDevice.h>
#include <Adafruit_Sensor.h>

/** Status Byte values */
#define MLX90395_STATUS_OK 0x00      /**< OK value for status response. */
#define MLX90395_STATUS_RESET 0x02
#define MLX90395_STATUS_SMMODE 0x20
#define MLX90395_STATUS_DRDY 0x01
#define MLX90395_STATUS_WOCMODE 0x40
#define MLX90395_STATUS_BURSTMODE
#define MLX90395_STATUS_BURSTMODE_MEASUREMENT_COUNTER_MASK 0x70
#define BURSTMODE_MEASUREMENT_COUNT(status) (status>>4 & 0x07) /* get
measurement count from status byte for burst mode */
#define MLX90395_STATUS_COMMUNICATION_ERROR 0x08
#define MLX90395_STATUS_OVF_ERROR 0x04

#define MLX90395_LIB_ERROR 0xFF /* transfer function return value when I2C
communication fails */

/** Register map. */
#define MLX90395_AXIS_ALL (0x0E)      /**< X+Y+Z axis bits for commands.
Drops T */
#define MLX90395_REG_0 (0x00)          /**< Gain */
#define MLX90395_REG_1 (0x01)          /**< Burst, comm mode */
#define MLX90395_REG_2 (0x02)          /**< Oversampling, filter, res. */
// #define MLX90395_CONF4 (0x03)        /**< Sensitivity drift. */
#define MLX90395_GAIN_SHIFT (4)        /**< Left-shift for gain bits. */
#define MLX90395_HALL_CONF (0x0C)      /**< Hall plate spinning rate adj. */
```

```

enum MLX90395_COMMAND_REGISTERS {
    MLX90395_REG_SB = (0x10),  /**< Start burst mode. */
    MLX90395_REG_SWOC = (0x20), /**< Start wakeup on change mode. */
    MLX90395_REG_SM = (0x30),  /**> Start single-meas mode. */
    MLX90395_REG_RM = (0x40),  /**> Read measurement. */
    MLX90395_REG_RR = (0x50),  /**< Read register. */
    MLX90395_REG_WR = (0x60),  /**< Write register. */
    MLX90395_REG_EX = (0x80),  /**> Exit mode. */
    MLX90395_REG_RV = (0xC0),  /**< Read voltage. */
    MLX90395_REG_HR = (0xD0),  /**< Memory recall. */
    MLX90395_REG_HS = (0xE0),  /**< Memory store. */
    MLX90395_REG_RT = (0xF0),  /**< Reset. */
};

typedef enum mlx90393_osr {
    MLX90395_OSR_1,
    MLX90395_OSR_2,
    MLX90395_OSR_4,
    MLX90395_OSR_8,
} mlx90393_osr_t;

typedef enum mlx90393_res {
    MLX90395_RES_16,
    MLX90395_RES_17,
    MLX90395_RES_18,
    MLX90395_RES_19,
} mlx90393_res_t;

static const float gainMultipliers[16] = {
    0.2, 0.25, 0.3333, 0.4, 0.5, 0.6, 0.75, 1,
    0.1, 0.125, 0.1667, 0.2, 0.25, 0.3, 0.375, 0.5};

#define MLX90395_DEFAULT_ADDR (0x0C) /** Can also be 0x18, depending on IC
 */

/** Class for interfacing with MLX90395 magnetometer */
class Adafruit_MLX90395 : public Adafruit_Sensor {
public:
    Adafruit_MLX90395();

```

```

bool begin_I2C(uint8_t i2c_addr = MLX90395_DEFAULT_ADDR,
               TwoWire *wire = &Wire);
bool begin_SPI(uint8_t cs_pin, SPIClass *theSPI = &SPI);

void reset(void);
bool exitMode(void);
bool startBurstMode(void);
bool startWOCMode(void);
bool startSingleMeasurementMode(void);
bool readMeasurement(float *x, float *y, float *z);
bool readData(float *x, float *y, float *z);

mlx90393_osr_t getOSR(void);
bool setOSR(mlx90393_osr_t osrval);
mlx90393_res_t getResolution(void);
bool setResolution(mlx90393_res_t resval);
uint8_t getGain(void);
bool setGain(uint8_t gainval);

void getSensor(sensor_t *sensor);
bool getEvent(sensors_event_t *event);

uint16_t uniqueID[3]; ///< 48 bits of unique identifier, read during init

private:
Adafruit_I2CDevice *i2c_dev = NULL;
Adafruit_SPIDevice *spi_dev = NULL;

bool _init(void);
uint8_t command(uint8_t cmd);
bool readRegister(uint8_t reg, uint16_t *data);
bool writeRegister(uint8_t reg, uint16_t *data);

mlx90393_res_t _resolution = MLX90395_RES_17;
uint8_t _gain = 0;
float _uTLSB = 0;
int32_t _sensorID = 90395;
int _cspin;
};

```

```
#endif
```

```
/******  
***  
This is a library for the MLX90395 magnetometer.  
  
Designed specifically to work with the MLX90395 breakout from Adafruit:  
  
----> https://www.adafruit.com/products/4022  
  
These sensors use I2C to communicate, 2 pins are required to interface.  
  
Adafruit invests time and resources providing this open source code,  
please  
support Adafruit and open-source hardware by purchasing products from  
Adafruit!  
  
Written by Kevin Townsend/ktown for Adafruit Industries.  
SPI implementation added by Hannah Eikens  
  
MIT license, all text above must be included in any redistribution  
*****  
***/  
  
#include "Adafruit_MLX90395.h"  
  
Adafruit_MLX90395::Adafruit_MLX90395(void) {}  
  
/*!  
*   @brief Sets up the hardware and initializes I2C  
*   @param i2c_addr  
*           The I2C address to be used.  
*   @param wire  
*           The Wire object to be used for I2C connections.  
*   @return True if initialization was successful, otherwise false.  
*/
```

```

bool Adafruit_MLX90395::begin_I2C(uint8_t i2c_addr, TwoWire *wire) {
    if (!i2c_dev) {
        i2c_dev = new Adafruit_I2CDevice(i2c_addr, wire);
    }
    spi_dev = NULL;

    if (!i2c_dev->begin()) {
        return false;
    }

    return _init();
}

bool Adafruit_MLX90395::begin_SPI(uint8_t cs_pin, SPIClass *theSPI) {
    i2c_dev = NULL;
    if (!spi_dev) {
        _cspin = cs_pin;
        spi_dev = new Adafruit_SPIDevice(cs_pin,
                                         1000000,           // frequency
                                         SPI_BITORDER_MSBFIRST, // bit order
                                         SPI_MODE3,           // data mode
                                         theSPI);
    }
    if (!spi_dev->begin()) {
        return false;
    }
    return _init();
}

bool Adafruit_MLX90395::_init(void) {
    exitMode();
    delay(2);
    reset();
    if (!exitMode())
        return false;

    delay(10);

    _gain = getGain();
}

```

```

    if (_gain == 8) { // default high field gain
        _uTLSB = 7.14;
    } else {
        _uTLSB = 2.5; // medium field gain
    }

    _resolution = getResolution();

    if (!readRegister(0x26, &uniqueID[0]) || !readRegister(0x27,
&uniqueID[1]) ||
        !readRegister(0x28, &uniqueID[2])) {
        return false;
    }

    return true;
}

/**
 * Performs a single X/Y/Z conversion and returns the results.
 *
 * @param x    Pointer to where the 'x' value should be stored.
 * @param y    Pointer to where the 'y' value should be stored.
 * @param z    Pointer to where the 'z' value should be stored.
 *
 * @return True if the operation succeeded, otherwise false.
 */
bool Adafruit_MLX90395::readData(float *x, float *y, float *z) {
    if (!startSingleMeasurementMode())
        return false;
    while (!readMeasurement(x, y, z))
        delay(1);

    return true;
}

/**
 * Reads data from data register & returns the results.
 *
 * @param x    Pointer to where the 'x' value should be stored.

```

```

* @param y      Pointer to where the 'y' value should be stored.
* @param z      Pointer to where the 'z' value should be stored.
*
* @return True on command success
*/
bool Adafruit_MLX90395::readMeasurement(float *x, float *y, float *z) {
    uint8_t tx[1] = {0x80}; // Read memory command in I2c
    uint8_t rx[12] = {0};    // status, crc, X16, Y16, Z16, T16, V16

    /* Perform the transaction. */
    if (i2c_dev) {
        if (!i2c_dev->write_then_read(tx, 1, rx, 12)) {
            return false;
        }
    }

    if (spi_dev) {
        tx[0] = MLX90395_REG_RM; // read measurement command
        if (!spi_dev->write_then_read(tx, 1, rx, 12)) {
            return false;
        }
    }

    // check status for Data Ready
    // Serial.print("Status: "); Serial.println(rx[0], HEX);
    if ((rx[0] & MLX90395_STATUS_DRDY) != MLX90395_STATUS_DRDY) {
        return false;
    }

    int16_t xi, yi, zi;

    // Convert data to uT and float.
    xi = (rx[2] << 8) | rx[3];
    yi = (rx[4] << 8) | rx[5];
    zi = (rx[6] << 8) | rx[7];

    *x = xi;
    *y = yi;
    *z = zi;

```

```

    // multiply by gain & LSB
    *x *= gainMultipliers[_gain] * _uTLSB;
    *y *= gainMultipliers[_gain] * _uTLSB;
    *z *= gainMultipliers[_gain] * _uTLSB;

    return true;
}

/**
 * Perform a soft reset, status byte of next command will indicate if reset
took place
 */
void Adafruit_MLX90395::reset(void) {
    uint8_t cmd = MLX90395_REG_RT;

    if (spi_dev) {
        spi_dev->write(&cmd, 1);
    }
}

/**
 * Perform a mode exit
 * @return True if the operation succeeded, otherwise false.
 */
bool Adafruit_MLX90395::exitMode(void) {
    // do once and ignore status
    command(MLX90395_REG_EX); // first call returns RT status
    return command(MLX90395_REG_EX) == MLX90395_STATUS_OK;
}

/**
 * Begin a single measurement on all axes
 *
 * @return True on command success
 */
bool Adafruit_MLX90395::startSingleMeasurementMode(void) {
    return (command(MLX90395_REG_SM | 0x0F) == MLX90395_STATUS_SMMODE);
}

```

```

/**
 * Begin burst mode on all axis
 *
 * @return True on command success
 */
bool Adafruit_MLX90395::startBurstMode(void) {
    return command(MLX90395_REG_SB);
}

/**
 * Begin wake-up on command mode on all axis
 *
 * @return True on command success
 */
bool Adafruit_MLX90395::startWOCMode(void) {
    return command(MLX90395_REG_SWOC);
}

/**
 * Get the current oversampling setting
 * @return MLX90395_OSR_1, MLX90395_OSR_2, MLX90395_OSR_4, or
MLX90395_OSR_8
 */
mlx90393_osr_t Adafruit_MLX90395::getOSR(void) {
    if (i2c_dev) {
        Adafruit_BusIO_Register reg2 =
            Adafruit_BusIO_Register(i2c_dev, MLX90395_REG_2 << 1, 2, MSBFIRST);
        Adafruit_BusIO_RegisterBits osr_bits =
            Adafruit_BusIO_RegisterBits(&reg2, 2, 0);
        return (mlx90393_osr_t)osr_bits.read();
    }

    if (spi_dev) {
        uint16_t data;
        readRegister(MLX90395_REG_2, &data);
        return (mlx90393_osr_t) (data&0x3);
    }
}

```

```

    return (mlx90393_osr_t) 0xFF;
}

/**
 * Set the current oversampling setting
 * @param osrval MLX90395_OSR_1, MLX90395_OSR_2, MLX90395_OSR_4,
 * or MLX90395_OSR_8
 * @return True on command success
 */
bool Adafruit_MLX90395::setOSR(mlx90393_osr_t osrval) {
    if (i2c_dev) {
        Adafruit_BusIO_Register reg2 =
            Adafruit_BusIO_Register(i2c_dev, MLX90395_REG_2 << 1, 2, MSBFIRST);
        Adafruit_BusIO_RegisterBits osr_bits =
            Adafruit_BusIO_RegisterBits(&reg2, 2, 0);
        return osr_bits.write(osrval);
    }

    if (spi_dev) {
        uint16_t data = osrval & 0x3;
        return writeRegister(MLX90395_REG_2, &data);
    }

    return false;
}

/**
 * Get the current gainsel value, see DS 18.6. Sensitivity for values
 * @return 0-15 gain selection offset
 */
uint8_t Adafruit_MLX90395::getGain(void) {
    if (i2c_dev) {
        Adafruit_BusIO_Register reg0 =
            Adafruit_BusIO_Register(i2c_dev, MLX90395_REG_0, 2, MSBFIRST); //
no need to shift I2C register << 1, because it is zero
        Adafruit_BusIO_RegisterBits gain_bits =
            Adafruit_BusIO_RegisterBits(&reg0, 4, MLX90395_GAIN_SHIFT);
        return gain_bits.read();
    }
}

```

```

    if (spi_dev) {
        uint16_t data;
        readRegister(MLX90395_REG_0, &data);
        return data >> MLX90395_GAIN_SHIFT & 0x0F; // mask off gain from
register
    }

    else return 0xFF;
}

/**
 * Get the current gainsel value, see DS 18.6. Sensitivity for values
 * @param gainval 0-15 gain selection offset
 * @return True on command success
 */
bool Adafruit_MLX90395::setGain(uint8_t gainval) {
    gainval; //&= 0xF;
    _gain = gainval; // cache it

    if (i2c_dev) {
        Adafruit_BusIO_Register reg0 =
            Adafruit_BusIO_Register(i2c_dev, MLX90395_REG_0, 2, MSBFIRST); //
no need to shift I2C register << 1, because it is zero
        Adafruit_BusIO_RegisterBits gain_bits =
            Adafruit_BusIO_RegisterBits(&reg0, 4, MLX90395_GAIN_SHIFT);
        return gain_bits.write(gainval);
    }

    if (spi_dev) {
        uint16_t data = gainval<MLX90395_GAIN_SHIFT;
        return writeRegister(MLX90395_REG_0, &data); // writes to all of
register 0, will override other setting. Need function to write all
registers in order to remembersettings.
    }

    return false;
}

/**

```

```

* Get the resolution, note that its till 16 bits just offset within the 19
* bit ADC output.
* @return MLX90395_RES_16, MLX90395_RES_17, MLX90395_RES_18 or
* MLX90395_RES_19
*/
mlx90393_res_t Adafruit_MLX90395::getResolution(void) {
    if (i2c_dev) {
        Adafruit_BusIO_Register reg2 =
            Adafruit_BusIO_Register(i2c_dev, MLX90395_REG_2 << 1, 2, MSBFIRST);
        Adafruit_BusIO_RegisterBits resX_bits =
            Adafruit_BusIO_RegisterBits(&reg2, 2, 5);
        return (mlx90393_res_t)resX_bits.read();
    }
    if (spi_dev) {
        uint16_t data;
        readRegister(MLX90395_REG_2, &data);

        return (mlx90393_res_t) (data>>5 & 0x03);
    }
    return (mlx90393_res_t) 0xFF;
}

/**
* Set the resolution, note that its till 16 bits just offset within the 19
* bit ADC output.
* @param resval MLX90395_RES_16, MLX90395_RES_17, MLX90395_RES_18 or
* MLX90395_RES_19
* @return True on command success
*/
bool Adafruit_MLX90395::setResolution(mlx90393_res_t resval) {
    _resolution = resval; // cache it

    if (i2c_dev) {
        Adafruit_BusIO_Register reg2 =
            Adafruit_BusIO_Register(i2c_dev, MLX90395_REG_2 << 1, 2, MSBFIRST);
        Adafruit_BusIO_RegisterBits resX_bits =
            Adafruit_BusIO_RegisterBits(&reg2, 2, 5);
        resX_bits.write(resval);
        Adafruit_BusIO_RegisterBits resY_bits =

```

```

        Adafruit_BusIO_RegisterBits(&reg2, 2, 7);
        resY_bits.write(resval);
        Adafruit_BusIO_RegisterBits resZ_bits =
            Adafruit_BusIO_RegisterBits(&reg2, 2, 9);
        return resZ_bits.write(resval);
    }

    if (spi_dev) {
        uint16_t data = resval<<5 | resval<<7 | resval<<9; // X, Y, Z   TODO:
global constants for each register offset and their set methods
        return writeRegister(MLX90395_REG_2, &data);
    }

    return false;
}

/*****

/**
 * Send command byte over I2C or SPI
 * @return status byte
 */
uint8_t Adafruit_MLX90395::command(uint8_t cmd) {
    uint8_t tx[2] = {0x80, cmd};
    uint8_t status = 0xFF;

    /* Perform the transaction. */
    if (i2c_dev) {
        if (!i2c_dev->write_then_read(tx, 2, &status, 1)) {
            // Serial.println("Failed command");
            return 0xFF;
        }
    }
    if (spi_dev) {
        if (!spi_dev->write_then_read(&cmd, 1, &status, 1, 0x00)) {
            return 0xFF;
        }
    }
    return status;
}

```

```

}

bool Adafruit_MLX90395::readRegister(uint8_t reg, uint16_t *data) {
    /* Perform the transaction. */
    if (i2c_dev) {
        Adafruit_BusIO_Register regis =
            Adafruit_BusIO_Register(i2c_dev, reg << 1, 2);
        *data = regis.read();
    }

    if (spi_dev) {
        uint8_t tx[2] = {MLX90395_REG_RR, reg << 2};
        uint8_t rx[3] = {0};

        spi_dev->write_then_read(tx, 2, rx, 3, 0x00);
        *data = (rx[1] << 8) | rx[2];
        if (rx[0] & MLX90395_STATUS_COMMUNICATION_ERROR) {
            return false;
        }
    }

    return true;
}

bool Adafruit_MLX90395::writeRegister(uint8_t reg, uint16_t *data) {
    if (spi_dev) {
        uint8_t tx[4] = {MLX90395_REG_WR, *data>>8, *data&0xFF, reg<<2}; // WR
command, data MSB, data LSB, reg
        uint8_t rx = 0xFF; // status byte

        spi_dev->write_then_read(tx, 4, &rx, 1, 0x00);

        if (!rx & MLX90395_STATUS_COMMUNICATION_ERROR) {
            return true;
        }
    }

    return false;
}

```

```

/*****
/
/
/*!
    @brief Gets the sensor_t device data, Adafruit Unified Sensor format
    @param sensor Pointer to an Adafruit Unified sensor_t object that
we'll
    fill in
*/
/*****
/
void Adafruit_MLX90395::getSensor(sensor_t *sensor) {
    /* Clear the sensor_t object */
    memset(sensor, 0, sizeof(sensor_t));

    /* Insert the sensor name in the fixed length char array */
    strncpy(sensor->name, "MLX90395", sizeof(sensor->name) - 1);
    sensor->name[sizeof(sensor->name) - 1] = 0;
    sensor->version = 1;
    sensor->sensor_id = _sensorID;
    sensor->type = SENSOR_TYPE_MAGNETIC_FIELD;
    sensor->min_delay = 0;
    sensor->min_value = -120000; // -120 gauss in uTesla
    sensor->max_value = 120000; // +120 gauss in uTesla
    sensor->resolution = _uTLsb; // 240/16-bit uTesla per LSB
}

/*****
/
/
/*!
    @brief Gets the most recent sensor event, Adafruit Unified Sensor
format
    @param event Pointer to an Adafruit Unified sensor_event_t object that
we'll fill in
    @returns True on successful read
*/
/*****
/
bool Adafruit_MLX90395::getEvent(sensor_event_t *event) {

```

```

/* Clear the event */
memset(event, 0, sizeof(sensors_event_t));

event->version = 1;
event->sensor_id = _sensorID;
event->type = SENSOR_TYPE_MAGNETIC_FIELD;
event->timestamp = millis();

return readData(&event->magnetic.x, &event->magnetic.y, &event-
>magnetic.z);
}

```

B.2 Finger Bender main.cpp and platformio.ini

```

#include <Arduino.h>
#include <Adafruit_MLX90395.h>
#include <Adafruit_TinyUSB.h>
#include <MIDI.h>
#include <insertionSort.h>
#include <mapf.h>

#define DEBUG
#define SENSOR_MAX_VALUE 23395.6370

#define MAX_MAGNET_DISTANCE 16.407 // mm
#define MIN_MAGNET_DISTANCE 5.5 // mm
#define SENSOR_DISTANCE 15.0 // from sensor origin to origin
#define TOGGLE_PIN 14
#define STICK_SW 22
#define STICK_X_PIN A0
#define STICK_Y_PIN A1

Adafruit_MLX90395 sensor_a = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_b = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_c = Adafruit_MLX90395();

// MIDI
Adafruit_USBD_MIDI usb_midi;

```

```

// Create a new instance of the Arduino MIDI Library,
// and attach usb_midi as the transport.
MIDI_CREATE_INSTANCE(Adafruit_USBD_MIDI, usb_midi, MIDI);

void setup() {
  #ifdef DEBUG
    Serial1.begin(115200);
  #endif

  // Manual begin() is required on core without built-in support e.g. mbed
  // rp2040
  if (!TinyUSBDevice.isInitialized()) {
    TinyUSBDevice.begin(0);
  }
  // Initialize MIDI, and listen to all MIDI channels
  // This will also call usb_midi's begin()
  MIDI.begin(MIDI_CHANNEL_OMNI);

  // If already enumerated, additional class drivers begin() e.g msc, hid,
  // midi won't take effect until re-enumeration
  if (TinyUSBDevice.mounted()) {
    TinyUSBDevice.detach();
    delay(10);
    TinyUSBDevice.attach();
  }

  // Setup SPI
  SPI.setRX(4);
  //SPI.setCS(5);
  SPI.setSCK(2);
  SPI.setTX(3);

  // init sensor a
  sensor_a.begin_SPI(5, &SPI);
  delay(100);
  //sensor.setResolution(MLX90395_RES_17);
  sensor_a.setOSR(MLX90395_OSR_8);

  // init sensor b

```

```

sensor_b.begin_SPI(6, &SPI);
delay(100);
//sensor.setResolution(MLX90395_RES_17);
sensor_b.setOSR(MLX90395_OSR_8);

// init sensor c
sensor_c.begin_SPI(7, &SPI);
delay(100);
//sensor.setResolution(MLX90395_RES_17);
sensor_c.setOSR(MLX90395_OSR_8);

delay(100);
pinMode(TOGGLE_PIN, INPUT_PULLUP);
pinMode(STICK_SW, INPUT_PULLUP);
pinMode(STICK_X_PIN, INPUT);
pinMode(STICK_Y_PIN, INPUT);
analogReadResolution(12);
}

void loop() {
    float ax, ay, az;
    float bx, by, bz;
    float cx, cy, cz;

    static uint8_t a_angle_last_cc_value;
    static uint8_t b_angle_last_cc_value;
    static uint8_t c_angle_last_cc_value;
    static uint8_t stick_x_last_cc_value;
    static uint8_t stick_y_last_cc_value;
    static uint8_t stick_sw_last_cc_value;

    bool toggle_state = digitalRead(TOGGLE_PIN);
    bool stick_sw_state = digitalRead(STICK_SW);
    static bool last_stick_sw_state;

    sensor_a.readData(&ax, &ay, &az);
    sensor_b.readData(&bx, &by, &bz);
    sensor_c.readData(&cx, &cy, &cz);
    //delay(10);

```

```

// determine CC values

float a_angle = -degrees(atan2(ay, ax));
float b_angle = -degrees(atan2(by, bx));
float c_angle = -degrees(atan2(cy, cx));

uint8_t a_angle_cc_value = constrain(map(a_angle, -10.0f, 10.0f, 0, 127),
0, 127);
uint8_t b_angle_cc_value = constrain(map(b_angle, -10.0f, 10.0f, 0, 127),
0, 127);
uint8_t c_angle_cc_value = constrain(map(c_angle, -10.0f, 10.0f, 0, 127),
0, 127);
uint8_t stick_x_cc_value = analogRead(STICK_X_PIN)>>5;
uint8_t stick_y_cc_value = analogRead(STICK_Y_PIN)>>5;
static uint8_t stick_sw_cc_value = 0;

if (!stick_sw_state && stick_sw_state != last_stick_sw_state) {
    if (stick_sw_cc_value == 0) stick_sw_cc_value = 127;
    else stick_sw_cc_value = 0;
}

// send cc messages
if (toggle_state) {
    if (a_angle_last_cc_value != a_angle_cc_value)
MIDI.sendControlChange(102, a_angle_cc_value, 1);

    if (b_angle_last_cc_value != b_angle_cc_value)
MIDI.sendControlChange(103, b_angle_cc_value, 1);

    if (c_angle_last_cc_value != c_angle_cc_value)
MIDI.sendControlChange(104, c_angle_cc_value, 1);

    if (stick_x_last_cc_value != stick_x_cc_value)
MIDI.sendControlChange(105, stick_x_cc_value, 1);

    if (stick_y_last_cc_value != stick_y_cc_value)
MIDI.sendControlChange(106, stick_y_cc_value, 1);
}

```

```

    if (stick_sw_last_cc_value != stick_sw_cc_value)
MIDI.sendControlChange(107, stick_sw_cc_value, 1);
}
// if toggle switch is on, send cc messages in mapping mode
else {
    if (a_angle_cc_value == 0 || a_angle_cc_value == 127)
MIDI.sendControlChange(102, 64, 1);
    if (b_angle_cc_value == 0 || b_angle_cc_value == 127)
MIDI.sendControlChange(103, 64, 1);
    if (c_angle_cc_value == 0 || c_angle_cc_value == 127)
MIDI.sendControlChange(104, 64, 1);
    if (stick_x_cc_value == 0 || stick_x_cc_value == 127)
MIDI.sendControlChange(105, 64, 1);
    if (stick_y_cc_value == 0 || stick_y_cc_value == 127)
MIDI.sendControlChange(106, 64, 1);
    if (!stick_sw_state && stick_sw_state != last_stick_sw_state)
MIDI.sendControlChange(107, 0, 1);
}

a_angle_last_cc_value = a_angle_cc_value;
b_angle_last_cc_value = b_angle_cc_value;
c_angle_last_cc_value = c_angle_cc_value;
stick_x_last_cc_value = stick_x_cc_value;
stick_y_last_cc_value = stick_y_cc_value;
stick_sw_last_cc_value = stick_sw_cc_value;

last_stick_sw_state = stick_sw_state;

//delay(1);

#ifdef DEBUG
static uint32_t last_micros;
uint32_t loop_time = micros() - last_micros;
Serial1.printf("ax: %4.3f, ay: %4.3f, az: %4.3f, bx: %4.3f, by: %4.3f,
bz: %4.3f, cx: %4.3f, cy: %4.3f, cz: %4.3f, loop time: %d, a_angle: %4.3f,
b_angle: %4.3f, c_angle: %4.3f, cc102: %u, cc103: %u, cc104: %u, cc105: %u,
cc106: %u, cc107: %u\n",

```

```

        ax, ay, az, bx, by, bz, cx, cy, cz, loop_time, a_angle,
b_angle, c_angle, a_angle_cc_value, b_angle_cc_value, c_angle_cc_value,
stick_x_cc_value, stick_y_cc_value, stick_sw_cc_value);
    last_micros = micros();
#endif
}

```

```

; PlatformIO Project Configuration File
;
; Build options: build flags, source filter
; Upload options: custom upload port, speed and extra flags
; Library options: dependencies, extra library storages
; Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:rpipico2]
platform = https://github.com/maxgerhardt/platform-raspberrypi.git
board = rpicico2
framework = arduino
board_build.core = earlephilhower

build_flags =
    -DUSE_TINYUSB
board_build.arduino.earlephilhower.usb_manufacturer = FrostiFish
Enterprises
board_build.arduino.earlephilhower.usb_product = Sonable P1 Finger Bender
board_build.arduino.earlephilhower.usb_pid = 0x4269

upload_protocol = cmsis-dap
debug_tool = cmsis-dap

lib_deps =
    https://github.com/FrostiFish/Adafruit_MLX90395.git
    fortyseveneffects/MIDI Library@5.0.2

```

B.3 Pressure Pad main.cpp and platformio.ini

```
#include <Arduino.h>
#include <Adafruit_MLX90395.h>
#include <Adafruit_TinyUSB.h>
#include <MIDI.h>
#include <insertionSort.h>
#include <mapf.h>

#define DEBUG
#define SENSOR_MAX_VALUE 23395.6370

#define MAX_MAGNET_DISTANCE 16.407 // mm
#define MIN_MAGNET_DISTANCE 5.5 // mm
#define SENSOR_DISTANCE 15.0 // from sensor origin to origin
#define TOGGLE_PIN 7
#define GPIO_GND 0

Adafruit_MLX90395 sensor_a = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_b = Adafruit_MLX90395();

float az_min = 3000.0f;
float bz_min = 3000.0f;

float a_mag_min = 4200.0f;
float b_mag_min = 4200.0f;
float a_mag_max = 14900.0f;
float b_mag_max = 14900.0f;

float mag_min = 7900.0f;
float mag_max = 25850.0f;

float min_pressure = 10500.0f;
float max_pressure = 21000.0f;

float y_axis_median = 0.0f;
float y_axis_min = -2250.0f;
float y_axis_max = 2100.0f;
float y_axis_deadzone = 1500.0f;
```

```

// MIDI
Adafruit_USBD_MIDI usb_midi;
// Create a new instance of the Arduino MIDI Library,
// and attach usb_midi as the transport.
MIDI_CREATE_INSTANCE(Adafruit_USBD_MIDI, usb_midi, MIDI);

void setup() {
  #ifdef DEBUG
    Serial.begin(115200);
  #endif

  // Manual begin() is required on core without built-in support e.g. mbed
  // rp2040
  if (!TinyUSBDevice.isInitialized()) {
    TinyUSBDevice.begin(0);
  }
  // Initialize MIDI, and listen to all MIDI channels
  // This will also call usb_midi's begin()
  MIDI.begin(MIDI_CHANNEL_OMNI);

  // If already enumerated, additional class drivers begin() e.g msc, hid,
  // midi won't take effect until re-enumeration
  if (TinyUSBDevice.mounted()) {
    TinyUSBDevice.detach();
    delay(10);
    TinyUSBDevice.attach();
  }

  // Setup SPI
  SPI.setRX(4);
  //SPI.setCS(5);
  SPI.setSCK(2);
  SPI.setTX(3);

  // init sensor a
  sensor_a.begin_SPI(5, &SPI);
  delay(100);
  //sensor.setResolution(MLX90395_RES_17);
  sensor_a.setOSR(MLX90395_OSR_8);

```

```

// init sensor b
sensor_b.begin_SPI(6, &SPI);
delay(100);
//sensor.setResolution(MLX90395_RES_17);
sensor_b.setOSR(MLX90395_OSR_8);

delay(100);
pinMode(TOGGLE_PIN, INPUT_PULLUP);
pinMode(GPIO_GND, OUTPUT);
digitalWrite(GPIO_GND, LOW);
}

void loop() {
    float ax, ay, az;
    float bx, by, bz;

    static uint8_t xangle_last_cc_value;
    static uint8_t yangle_left_last_cc_value;
    static uint8_t yangle_right_last_cc_value;
    static uint8_t pressure_last_cc_value;

    bool toggle_state = digitalRead(TOGGLE_PIN);

    sensor_a.readData(&ax, &ay, &az);
    sensor_b.readData(&bx, &by, &bz);
    //delay(10);

    float amag = sqrt(sq(ax)+sq(ay)+sq(az));
    float bmag = sqrt(sq(bx)+sq(by)+sq(bz));
    float magnitude = amag + bmag;

    float a_normalized = mapf(amag, a_mag_min, a_mag_max, 0, PI/2);
    float b_normalized = mapf(bmag, b_mag_min, b_mag_max, 0, PI/2);
    float xangle = degrees(atan(a_normalized-b_normalized));

    // if (angle < min_angle) min_angle = angle;
    // if (angle > max_angle) max_angle = angle;

```

```

    uint8_t xangle_cc_value = constrain((uint8_t)mapf(xangle, -16.0f, 16.0f,
0.0f, 127.0f), 0, 127);
    uint8_t pressure_cc_value = constrain((uint8_t)mapf(magnitude,
min_pressure, max_pressure, 0.0f, 127.0f), 0, 127);
    float dy = by-ay;
    uint8_t yangle_left_cc_value = dy > y_axis_median+y_axis_deadzone ? 127 :
0;
    uint8_t yangle_right_cc_value = dy < y_axis_median-y_axis_deadzone ? 127
: 0;

    if (toggle_state) {
        if (xangle_last_cc_value != xangle_cc_value) MIDI.sendControlChange(80,
xangle_cc_value, 1);

        if (pressure_last_cc_value != pressure_cc_value)
MIDI.sendControlChange(81, pressure_cc_value, 1);

        if (yangle_left_last_cc_value != yangle_left_cc_value)
MIDI.sendControlChange(82, yangle_left_cc_value, 1);

        if (yangle_right_last_cc_value != yangle_right_cc_value)
MIDI.sendControlChange(83, yangle_right_cc_value, 1);
    }
    else {
        if (xangle_cc_value == 127) MIDI.sendControlChange(80, 64, 1);
        if (pressure_cc_value == 127) MIDI.sendControlChange(81, 64, 1);
        if (yangle_left_cc_value == 127) MIDI.sendControlChange(82, 0, 1);
        if (yangle_right_cc_value == 127) MIDI.sendControlChange(83, 0, 1);
    }

    MIDI.sendNrpnValue();

    xangle_last_cc_value = xangle_cc_value;
    pressure_last_cc_value = pressure_cc_value;
    yangle_left_last_cc_value = yangle_left_cc_value;
    yangle_right_last_cc_value = yangle_right_cc_value;

    //delay(1);

```

```

#ifdef DEBUG
    static uint32_t last_micros;
    uint32_t loop_time = micros() - last_micros;
    Serial.printf("ax: %4.3f, ay: %4.3f, az: %4.3f, bx: %4.3f, by: %4.3f, bz:
%4.3f, angle: %4.3f, loop time: %d, amag: %4.3f, bmag: %4.3f, mag: %4.3f,
dy: %4.3f, cc80: %u, cc81: %u, cc82: %u, cc83: %u\n",
                ax, ay, az, bx, by, bz, xangle, loop_time, amag, bmag,
magnitude, dy, xangle_cc_value, pressure_cc_value, yangle_left_cc_value,
yangle_right_cc_value);
    last_micros = micros();
#endif
}

```

```

; PlatformIO Project Configuration File
;
; Build options: build flags, source filter
; Upload options: custom upload port, speed and extra flags
; Library options: dependencies, extra library storages
; Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:rpipico2]
platform = https://github.com/maxgerhardt/platform-raspberrypi.git
board = rpicico2
framework = arduino
board_build.core = earlephilhower

build_flags =
    -DUSE_TINYUSB
board_build.arduino.earlephilhower.usb_manufacturer = FrostiFish
Enterprises
board_build.arduino.earlephilhower.usb_product = Sonable P1 Touch Pad
board_build.arduino.earlephilhower.usb_pid = 0x4268

; upload_protocol = cmsis-dap
; debug_tool = cmsis-dap

```

```
lib_deps =  
  https://github.com/FrostiFish/Adafruit_MLX90395.git  
  fortyseveneffects/MIDI Library@^5.0.2
```

B.4 Tabula main.cpp and platformio.ini

```
#include <Arduino.h>  
#include <Adafruit_MLX90395.h>  
#include <Adafruit_TinyUSB.h>  
#include <MIDI.h>  
#include <insertionSort.h>  
#include <mapf.h>  
  
#define PRINT_INFO  
#define SENSOR_MAX_VALUE 23395.6370  
  
#define MAX_MAGNET_DISTANCE 16.407 // mm  
#define MIN_MAGNET_DISTANCE 5.5 // mm  
#define SENSOR_DISTANCE 15.0 // from sensor origin to origin  
  
#define MIDI_OUT_PIN 20 // midi current spec is 5 mA, increase current for  
pinMode  
  
#define MAPMODE_BUTTON_PIN 17  
#define LEFT_BUTTON_PIN MIDI_OUT_PIN //16  
#define RIGHT_BUTTON_PIN 18  
#define FREEZE_BUTTON_PIN 19  
  
#define SCK_PIN 2  
#define MOSI_PIN 3  
#define MISO_PIN 4  
#define CS_A_PIN 5  
#define CS_B_PIN 6  
#define CS_C_PIN 7  
#define CS_D_PIN 8  
#define CS_E_PIN 9  
#define CS_F_PIN 10
```

```

#define LED_DATA_OUT_PIN 22

void mapmode_button_callback();
void freeze_button_callback();
void left_button_callback();
void right_button_callback();

enum OpMode {
    RUNNING,
    MAPPING
} OP_MODE;

enum AxisMode {
    CONTINUOUS,
    SPLIT
};

AxisMode X_AXIS_MODE = CONTINUOUS;
AxisMode Y_AXIS_MODE = CONTINUOUS;

bool freeze = false;

Adafruit_MLX90395 sensor_a = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_b = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_c = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_d = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_e = Adafruit_MLX90395();
Adafruit_MLX90395 sensor_f = Adafruit_MLX90395();

float front_min = 1200.f;
float front_max = 11000.f;
float back_min = 1100.f;
float back_max = 11000.f;
float left_min = 1400.f;
float left_max = 8300.f;
float right_min = 1500.f;
float right_max = 7500.f;
float pressure_min = 1000.f;

```

```

float pressure_max = 4000.f;

float x_angle_min = -58.f;
float x_angle_max = 55.f;
float y_angle_min = -50.f;
float y_angle_max = 53.f;
float deadzone = 8.f;
float deadzone_y = 12.f;

#define MIDI_CHANNEL 1
#define X_AXIS_CC1 102
#define X_AXIS_CC2 103
#define Y_AXIS_CC1 104
#define Y_AXIS_CC2 105
#define PRESSURE_CC 106

// MIDI
Adafruit_USBD_MIDI usb_midi;
// Create a new instance of the Arduino MIDI Library,
// and attach usb_midi as the transport.
MIDI_CREATE_INSTANCE(Adafruit_USBD_MIDI, usb_midi, MIDI);

void setup() {
    #ifdef PRINT_INFO
        //Serial1.setTX(0);
        //Serial1.setRX(1);
        Serial.begin(115200);
    #endif

    // Manual begin() is required on core without built-in support e.g. mbed
    rp2040
    if (!TinyUSBDevice.isInitialized()) {
        TinyUSBDevice.begin(0);
    }
    // Initialize MIDI, and listen to all MIDI channels
    // This will also call usb_midi's begin()
    MIDI.begin(MIDI_CHANNEL_OMNI);

```

```

    // If already enumerated, additional class drivers begin() e.g msc, hid,
    midi won't take effect until re-enumeration
    if (TinyUSBDevice.mounted()) {
        TinyUSBDevice.detach();
        delay(10);
        TinyUSBDevice.attach();
    }

    pinMode(MAPMODE_BUTTON_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(MAPMODE_BUTTON_PIN),
mapmode_button_callback, FALLING);
    pinMode(FREEZE_BUTTON_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(FREEZE_BUTTON_PIN),
freeze_button_callback, FALLING);
    pinMode(LEFT_BUTTON_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(LEFT_BUTTON_PIN),
left_button_callback, FALLING);
    pinMode(RIGHT_BUTTON_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(RIGHT_BUTTON_PIN),
right_button_callback, FALLING);

    pinMode(CS_A_PIN, OUTPUT);
    digitalWrite(CS_A_PIN, HIGH);
    pinMode(CS_B_PIN, OUTPUT);
    digitalWrite(CS_B_PIN, HIGH);
    pinMode(CS_C_PIN, OUTPUT);
    digitalWrite(CS_C_PIN, HIGH);
    pinMode(CS_D_PIN, OUTPUT);
    digitalWrite(CS_D_PIN, HIGH);
    pinMode(CS_E_PIN, OUTPUT);
    digitalWrite(CS_E_PIN, HIGH);
    pinMode(CS_F_PIN, OUTPUT);
    digitalWrite(CS_F_PIN, HIGH);

    // Setup SPI
    SPI.setRX(MISO_PIN);
    //SPI.setCS(5);
    SPI.setSCK(SCK_PIN);
    SPI.setTX(MOSI_PIN);

```

```

delay(5000);

// init sensor a
sensor_a.begin_SPI(CS_A_PIN, &SPI);
//sensor.setResolution(MLX90395_RES_17);
sensor_a.setOSR(MLX90395_OSR_8);

delay(1);

// init sensor b
sensor_b.begin_SPI(CS_B_PIN, &SPI);
//sensor.setResolution(MLX90395_RES_17);
sensor_b.setOSR(MLX90395_OSR_8);

delay(1);

// init sensor c
sensor_c.begin_SPI(CS_C_PIN, &SPI);
//sensor.setResolution(MLX90395_RES_17);
sensor_c.setOSR(MLX90395_OSR_8);

delay(1);

// init sensor d
sensor_d.begin_SPI(CS_D_PIN, &SPI);
//sensor.setResolution(MLX90395_RES_17);
sensor_d.setOSR(MLX90395_OSR_8);

delay(1);

// init sensor e
sensor_e.begin_SPI(CS_E_PIN, &SPI);
//sensor.setResolution(MLX90395_RES_17);
sensor_e.setOSR(MLX90395_OSR_8);

delay(1);

// init sensor f

```

```

sensor_f.begin_SPI(CS_F_PIN, &SPI);
//sensor.setResolution(MLX90395_RES_17);
sensor_f.setOSR(MLX90395_OSR_8);

delay(100);

sensor_a.startWOCMode();
sensor_b.startWOCMode();
sensor_c.startWOCMode();
sensor_d.startWOCMode();
sensor_e.startWOCMode();
sensor_f.startWOCMode();
}

void loop() {
    static float ax = 0.f, ay = 0.f, az = 0.f;
    sensor_a.readMeasurement(&ax, &ay, &az);
    float a_mag = sqrt(sq(ax)+sq(ay)+sq(az));

    static float bx = 0.f, by = 0.f, bz = 0.f;
    sensor_b.readMeasurement(&bx, &by, &bz);
    float b_mag = sqrt(sq(bx)+sq(by)+sq(bz));

    static float cx = 0.f, cy = 0.f, cz = 0.f;
    sensor_c.readMeasurement(&cx, &cy, &cz);
    float c_mag = sqrt(sq(cx)+sq(cy)+sq(cz));

    static float dx = 0.f, dy = 0.f, dz = 0.f;
    sensor_d.readMeasurement(&dx, &dy, &dz);
    float d_mag = sqrt(sq(dx)+sq(dy)+sq(dz));

    static float ex = 0.f, ey = 0.f, ez = 0.f;
    sensor_e.readMeasurement(&ex, &ey, &ez);
    float e_mag = sqrt(sq(ex)+sq(ey)+sq(ez));

    static float fx = 0.f, fy = 0.f, fz = 0.f;
    sensor_f.readMeasurement(&fx, &fy, &fz);
    float f_mag = sqrt(sq(fx)+sq(fy)+sq(fz));

```

```

float front_mag = e_mag + f_mag;
float back_mag = a_mag + b_mag;
float left_mag = a_mag + c_mag + e_mag;
float right_mag = b_mag + d_mag + f_mag;
float pressure = c_mag + d_mag;

float x_angle = degrees(atan(mapf(back_mag, back_min, back_max, 0,
HALF_PI) - mapf(front_mag, front_min, front_max, 0, HALF_PI)));
float y_angle = degrees(atan(mapf(right_mag, right_min, right_max, 0,
HALF_PI) - mapf(left_mag, left_min, left_max, 0, HALF_PI)));

static uint8_t x_axis_cc1 = 0;
static uint8_t x_axis_cc2 = 0;
static uint8_t y_axis_cc1 = 0;
static uint8_t y_axis_cc2 = 0;
static uint8_t pressure_cc = 0;
static uint8_t last_x_axis_cc1 = 0;
static uint8_t last_x_axis_cc2 = 0;
static uint8_t last_y_axis_cc1 = 0;
static uint8_t last_y_axis_cc2 = 0;
static uint8_t last_pressure_cc = 0;

if(OP_MODE == RUNNING) {
    if (!freeze){
        if(X_AXIS_MODE == CONTINUOUS) {
            x_axis_cc1 = (uint8_t)constrain(mapf(x_angle, x_angle_min,
x_angle_max, 0.f, 127.f), 0.f, 127.f);
            x_axis_cc2 = (uint8_t)constrain(mapf(x_angle, x_angle_min,
x_angle_max, 127, 0.f), 0.f, 127.f);
        }
        else if (X_AXIS_MODE == SPLIT) {
            x_axis_cc1 = (uint8_t)constrain(mapf(x_angle,
(x_angle_max+x_angle_min)+deadzone, x_angle_max, 0.f, 127.f), 0.f, 127.f);
            x_axis_cc2 = (uint8_t)constrain(mapf(x_angle,
(x_angle_max+x_angle_min)-deadzone, x_angle_min, 0.f, 127.f), 0.f, 127.f);
        }

        if(Y_AXIS_MODE == CONTINUOUS) {

```

```

        y_axis_cc1 = (uint8_t)constrain(mapf(y_angle, y_angle_min,
y_angle_max, 0.f, 127.f), 0.f, 127.f);
        y_axis_cc2 = (uint8_t)constrain(mapf(y_angle, y_angle_min,
y_angle_max, 127, 0.f), 0.f, 127.f);
    }
    else if (Y_AXIS_MODE == SPLIT) {
        y_axis_cc1 = (uint8_t)constrain(mapf(y_angle,
(y_angle_max+y_angle_min)+deadzone_y, y_angle_max, 0.f, 127.f), 0.f,
127.f);
        y_axis_cc2 = (uint8_t)constrain(mapf(y_angle,
(y_angle_max+y_angle_min)-deadzone_y, y_angle_min, 0.f, 127.f), 0.f,
127.f);
    }
    pressure_cc = (uint8_t)constrain(mapf(pressure, pressure_min,
pressure_max, 0.f, 127.f), 0.f, 127.f);
}

    if (x_axis_cc1 != last_x_axis_cc1) MIDI.sendControlChange(X_AXIS_CC1,
x_axis_cc1, MIDI_CHANNEL);
    if (x_axis_cc2 != last_x_axis_cc2) MIDI.sendControlChange(X_AXIS_CC2,
x_axis_cc2, MIDI_CHANNEL);
    if (y_axis_cc1 != last_y_axis_cc1) MIDI.sendControlChange(Y_AXIS_CC1,
y_axis_cc1, MIDI_CHANNEL);
    if (y_axis_cc2 != last_y_axis_cc2) MIDI.sendControlChange(Y_AXIS_CC2,
y_axis_cc2, MIDI_CHANNEL);
    if (pressure_cc != last_pressure_cc)
MIDI.sendControlChange(PRESSURE_CC, pressure_cc, MIDI_CHANNEL);

    last_x_axis_cc1 = x_axis_cc1;
    last_x_axis_cc2 = x_axis_cc2;
    last_y_axis_cc1 = y_axis_cc1;
    last_y_axis_cc2 = y_axis_cc2;
    last_pressure_cc = pressure_cc;
}
else if (OP_MODE == MAPPING) {
    if (x_angle > x_angle_max-deadzone) MIDI.sendControlChange(X_AXIS_CC1,
64, MIDI_CHANNEL);
    if (x_angle < x_angle_min+deadzone) MIDI.sendControlChange(X_AXIS_CC2,
64, MIDI_CHANNEL);

```

```

    if (y_angle > y_angle_max-deadzone) MIDI.sendControlChange(Y_AXIS_CC1,
64, MIDI_CHANNEL);
    if (y_angle < y_angle_min+deadzone) MIDI.sendControlChange(Y_AXIS_CC2,
64, MIDI_CHANNEL);
    if (pressure > pressure_max) MIDI.sendControlChange(PRESSURE_CC, 64,
MIDI_CHANNEL);
    delay(100);
}

#ifdef PRINT_INFO
static uint32_t last_micros;
uint32_t loop_time = micros() - last_micros;
Serial.printf("x angle: %f, y angle: %f, pressure: %f, OP_MODE: %u,
freeze: %u, x axis split: %u, y axis split: %u, a mag: %f, b mag: %f, c
mag: %f, d mag: %f, e mag: %f, f mag: %f, ", x_angle, y_angle, pressure,
OP_MODE, freeze, X_AXIS_MODE, Y_AXIS_MODE, a_mag, b_mag, c_mag, d_mag,
e_mag, f_mag);
Serial.printf("front mag: %f, back mag: %f, left mag: %f, right mag: %f
\n", front_mag, back_mag, left_mag, right_mag);
last_micros = micros();
#endif
//delay(1);
delayMicroseconds(500);
}

void mapmode_button_callback() {
    delay(1);
    if (!digitalRead(MAPMODE_BUTTON_PIN)) {
        if (OP_MODE == RUNNING) OP_MODE = MAPPING;
        else OP_MODE = RUNNING;
    }
}

void freeze_button_callback() {
    delay(1);
    if (!digitalRead(FREEZE_BUTTON_PIN)) {
        if (!freeze) freeze = true;
        else freeze = false;
    }
}

```

```

}

void left_button_callback() {
    delay(1);
    if (!digitalRead(LEFT_BUTTON_PIN)) {
        if (X_AXIS_MODE == CONTINUOUS) X_AXIS_MODE = SPLIT;
        else X_AXIS_MODE = CONTINUOUS;
    }
}

void right_button_callback() {
    delay(1);
    if (!digitalRead(RIGHT_BUTTON_PIN)) {
        if (Y_AXIS_MODE == CONTINUOUS) Y_AXIS_MODE = SPLIT;
        else Y_AXIS_MODE = CONTINUOUS;
    }
}

```

```

; PlatformIO Project Configuration File
;
; Build options: build flags, source filter
; Upload options: custom upload port, speed and extra flags
; Library options: dependencies, extra library storages
; Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:rpipico2]
platform = https://github.com/maxgerhardt/platform-raspberrypi.git
board = waveshare_rp2350_plus
framework = arduino
board_build.core = earlephilhower

build_flags =
    -DUSE_TINYUSB
board_build.arduino.earlephilhower.usb_manufacturer = FrostiFish
Enterprises

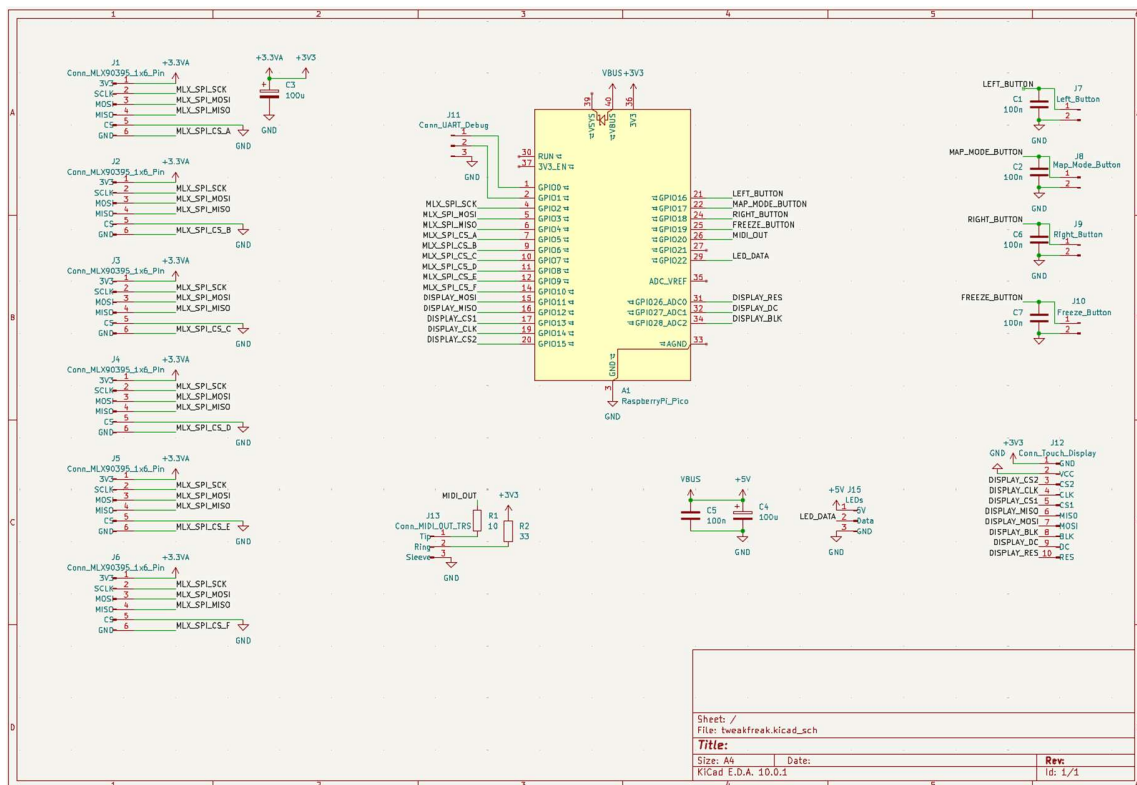
```

```
board_build.arduino.earlephilhower.usb_product = TweakFreak
board_build.arduino.earlephilhower.usb_pid = 0x4272
```

```
upload_protocol = cmsis-dap
debug_tool = cmsis-dap
```

```
lib_deps =
  https://github.com/FrostiFish/Adafruit_MLX90395.git
  fortyseveneffects/MIDI Library@^5.0.2
```

B.5 Tabula KiCad Schematic



B.6 Tabula KiCad Board Layout

