Università degli Studi di Milano
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

CORSO DI LAUREA TRIENNALE IN
INFORMATICA MUSICALE

A MULTI-TOUCH CONTROL SURFACE IMPLEMENTED AS A MIDI CONTROLLER

Supervisor: Prof. Luca Andrea Ludovico
Co-supervisor: Prof. Giorgio Presti

Final essay by:
Shawn Yumha Pinciara
Matr. No. 923246

Academic Year 2023-2024

This work is dedicated to all musicians who fell in love more
with making instruments that make music
rather than making music itself,
thus becoming meta-musicians.

Thanks to

I would like to thank all the people who daily push me out of my boundaries, always making me question myself. I hope to return the favor soon, for the good and the progress.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

What musicians want is to maximize the expressiveness of the instrument while minimizing the space of playing. These pocket musical technologies, especially regarding continuous controllers, are coming forward to meet the artists' needs.

As much as a large portion of acoustic instruments are indeed discrete-controlled and have a discrete expressiveness (pianos, guitars, basses, etc.), continuous-controlled instruments allow more modulation and nuance of the sound.

Comparing it digitally, most of the music controllers (MIDI) are discrete controllers(pianos, launchpads, encoders, etc.) and many times having an "expensive" implementation: as an example, a piano must have two switches per key, launchpads must have membranes and only a few manufacturers can make them work flawlessly.

Bringing these controllers into the continuum means thinking about a surface of almost infinite possibilities, where one can hypothetically track multiple touches, their movements and their intensity (velocity). With a few trade-offs to keep the cost down (as of the number of components, their complexity, and commercial value) this project aims to describe the prototyping process of such a surface. In a world where electronic music instruments are really expensive while their electric components are cheap, this project will try to analyze and suggest the various approaches on prototyping a multi-touch surface that minimize the cost while trying to maximize the expressiveness.

The essay is structured by starting analyzing the state of the art of the technology and the presence of contact surfaces in the market, explaining what type of controllers are expressive and what technologies are used to manufacture those touch surfaces. It then follows to talk about the design of the prototype, both hardware and software, and about the different choices driven by the different use cases proposed. From that tests will follow, formalizing a testing protocol and analyzing the results. Finally the conclusions will be presented, and some thoughts about the future developments of the project will be given.

Chapter 2

State of the art

The project is based on the work of Narjes Pourjafarian, Anusha Withana, Joseph A. Paradiso and Jürgen Steimle [1], that offers a hardware and software prototype to implement an inexpensive multi-touch control surface, using commodity microcontrollers and sensors greatly diffused in the market[2].

The use of inexpensive microcontrollers with open source software and without using specialized hardware is emphasized in the source and the goal of this technique is to be able to prototype a multi-touch control surface, which then can handle several touches simultaneously, quickly and inexpensively.

This technique differs from the other ones (more expensive and less customizable) by being highly scalable, considering the hardware limitations of microcontrollers.

In fact the control surface can be crafted using different materials, such as wires, tapes or even inks, all conductive and metal based. The surface can become flexible and take the shape of a curved object. It works by arranging the probes like a nXm matrix, with the two layers (rows and columns) divided by a dielectric sheet. As an example, a few tries are presented [1]: a semirigid surface and a curved one (on the back of a 3d printed bunny).

The quality and precision of the surfaces is directly proportional to the number of traces and the distance between them: the highest number of traces has been achieved by using an Arduino Mega microcontroller, which has the most number of analog pins in the Arduino family.

This allowed the implementation of a plane with 256 contact points, thus using a 16x16 matrix. The area covered by this surface is 177x177mm, and the various metal tracks were printed using a silver-based ink printer. This surface allows tracking of up to 10 fingers and has a medium to high resolution. Another example in a very confined space in the textile domain is the test with a surface of 6x6 traces, composed of copper wires hand-sewn onto a sleeve. A battery powered Lilypad-type microcontroller has the ability to interface with mobile devices via Bluetooth that was then soldered to the wires[3].

The form-factor of this circuit is small and the multi-touch plane has quite a low resolution, but still useful for several use cases. The last example shown is about the most minimal setup of the prototype, a 9-point plane (6 strips of copper tape in a matrix 3x3) with a really low resolution, thus becoming more of a button matrix, where each button corresponds to the intersection between the two tracks. Although limited, unlike button arrays this surface can still detect useful gestures to extend the capabilities of the interface.

This surface has therefore a higher scalability and the prototyping process is greatly faster than others [4].

As an example, resistive touch surfaces (found in supermarkets, train stations, etc.) can be used with a single point of contact. Even if it can be used with a stylus (thus not a finger) it's very limiting and not prototypable, having a high cost of production [5]. Infrared technology surfaces on the other hand support multi-touch but, being much more expensive, they work only in controlled environments (e.g. not right under the sun).

One of the very robust and widely used surfaces in all ATMs are optical imaging surfaces. They triangulate the position of multiple fingers through the use of two infrared sensors and a LED strip. Although accurate and standardized in the market, they are not so suitable for fast prototyping and keeping the form factor low, as they require a large number of components and some specific construction materials.

Finally, the most widely used surfaces in the market are projected capacitance surfaces, which are used extensively on phones and mobile devices[6]. They are very precise and fast, but on the other hand very expensive, difficult to scale, and they typically support around 5 fingers maximum. Although usable as a prototyping tool (almost all have a phone in their pocket right now) it is complicated to interface directly with them, except by implementing applications directly for the OS that supports them.

There are also ulterior methods using low level electronic components to implement musical instruments surfaces. They will be now briefly described to categorize the various use cases.

The simplest method to implement a control surface is by having multiple exposed metal plates (on a planar surface) that represent one side of the buttons. The other side (ground) is usually a metal pointer or any other metal object of some sort, that closes the circuit on contact with the surface. This is the main working principle of the Stylophone [7]. It implements 2 octaves of notes by having one metal plate per key and a stylus (with a metal pointer) connected by wire to the body of the instrument.

Another way, in this case linear, can be achieved using strip resistors. They look like a strip of two layers and on pressure (following just one dimension, its length) the resistance changes. Laying them with the shape of a grid, a 2-dimensions multi touch pressure plate will be achieved. Comparing it with the presented method [1], linear resistors are way more expensive, although very cheap compared to other low level components, and crafting the prototype could be harder as a great precision is needed to glue all the

components together tightly. The prototype of the popular Linnstrument [8] uses this very own technique. This instrument is in fact considered one of the most expressive instruments commercially available. There are of course other commercially available electronic music instruments which implement a 3 axis linear layout, such as the Haken Continuum[9], the Roli Seaboard[10] and the Ableton Push[11] (which is a grid as every launchpad, but with linear capabilities for each key). These three instruments achieve a state of the art expressivity through having a linear surface, although the technologies used are not known publicly, thus making it difficult to speculate or discuss if those techniques could be easily replicated during a prototyping process.

Finally, many other instruments (even eurorack modules) rely on using metal plates by calculating the capacitance of the surface when touched by human skin [12, 13]. This technique is embedded on microcontrollers too: almost every one of the ESP32 microcontroller family contains some pins that can be used as touch buttons[14]. Implementing this technique is expensive (in terms of number of pins, as each pin can be linked to only one metal surface) although there are specific modules that use I2C just for this matter (like the MPR121).

Chapter 3

Instrument design

3.1 Instrument working

A high level description of the instrument is the following: there is a surface (planar or non planar, square or rectangle or of any other shape) that is touched using both hands. The touching points are then tracked and converted to MIDI notes and then sent to a DAW that produces sound.

Design-wise this process is divided in two sections: The hardware design thinks about how big the surface is, what shape it has and what components take place to make the instrument work. The layout, or mapping between points on the surface and the musical notes, also falls in this category. The other part is the software design, so how the microcontrollers calculate the touch points and how the raw data is processed so that it can send out ready-to-be-played MIDI notes and CCs. More particularly than the implementation itself, the different software technologies used to achieve this multi-step process will be described.

3.2 Surface design

3.2.1 Crafting techniques and materials

As reported in [1] there are three main ways in which the surface is crafted.

The first one relies on a coated copper thread, used in the textile realm. The thread is sewn on a cloth in the form of a grid, considering that the rows and columns shouldn't touch, that's why the thread used is coated. As described [1] the distance between threads should be around 1mm to achieve a precise reading; considering that the thread size is small (0.3mm of diameter) the final surface will then have a small form factor, thus making it unsuitable for both hands to cover it. Although adding more rows and columns could resolve that, the maximum achieved number of probes was with a 12x12 strips surface which won't still be large enough for this particular use case[1].

The process of manufacturing such a surface would be as fast as one can saw, or by using a sewing machine that supports a metallic thread (and using the coated one) the process would be sped up. Another aspect to keep in mind is dealing with the cloth itself; the textile material should not be too stretchy nor sturdy, easy to saw (not like leather) and thin enough so that the thread stays linear to the surface and don't make vertical zig-zags through the surface (which might be a big fault of using sewing machines).

The second method uses silver ink pens to draw lines on any sort of surface that can be stacked to form a grid layout. An example of using this method is the birthday card with two pieces of paper stacked upon each other [1]. Paper is a great material for ink to attach to but is not as reliable as other harder thicker surfaces (such as plastic sheets). The shapes can be traced on one piece of paper only, by making use of the two sides of it (Figure 1-c). The paper sheet should be thick enough so that the ink doesn't penetrate to the other side of it, effectively shorting the circuit. The conductive ink pen (Figure 1-a) might be hard to use as it's hard to cover a surface uniformly without leaving spaces between lines, and drawing many shapes could results in a time consuming and imprecise process. A plotter could be used to draw the shapes precisely, but it requires specific hardware not usually easy to find in common homes. Other types of materials could be used but ink is hard to handle if not with specific techniques; Another method proposed relies on using a printer to print on plastic with some special silver conductive inks, which can't be easily found in normal homes and thus categorized as "DIY"[1].

While it is really fast to prototype, because just two layers and some fast drawn lines are needed, the process of soldering some wires to it (in order of being able to attach the other components such as the microcontroller) or by using crocodile clips might not be as easy and firm, resulting in an unstable and messy setup (Figure 1-b). The other problem that arose previously affects this method too: the thickness of an ink line is around 0.1-0.3mm which is again very small to properly cover the entire surface considering that there should be a small gap between traces.

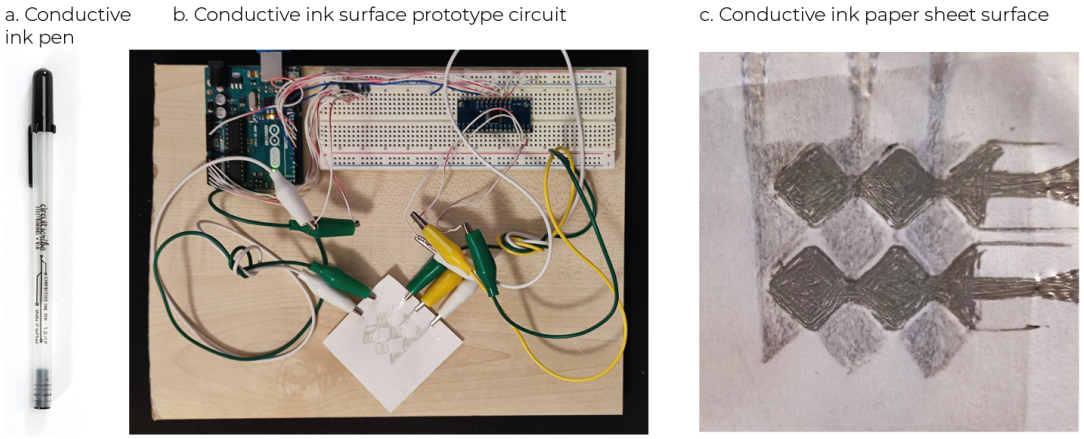

Figure 1: Conductive ink based surface.

3.2.2 Copper based surface

The third method implements the surface by using the commercially available copper tape, that can be found in almost every hardware store. The tape is glued to any surface. As the row and columns should be electrically separated, a dielectric layer is added in between. This in-between layer could be a plastic sheet or a plastic tape for planar surfaces, or even a coat of spray paint, very convenient to cover non planar probes. A test was performed with the spray paint but the difference in heights between the copper tape and the surface in which the tape has been glued made the spray paint cover the surface non uniformly, leaving some copper uncovered, thus shorting the circuit when adding the second row of probes. As the copper tape exceeds 5mm in length, it has to be cut in smaller strips or other shapes. The wideness of the strips can both determine how big or dense the surface can be. As of an example, having a fixed area: Using bigger strips will effectively reduce the number of microcontroller pins needed to cover that area, but at the same time reducing the precision to calculate the point of contact. Vice versa, using smaller strips will increase the granularity and precision but consuming more resources (pins) of the microcontrollers.

The balance between resources used and size of the surface should be considered for every use case. For instance, implementing a simple grid-like sampler would require a few trips and a medium sized surface. Or implementing a melodic instrument would require having to calculate a much more precise finger's positioning on the surface, by increasing the granularity and the number of small strips needed.

Another way to see the problem would be thinking about how many intersections of

rows and columns should be found under the tip of the finger at any given time. Having more intersections near each other (with a higher density) allows to calculate the position if just the tip or the full finger is in contact with the surface. This is actually a great example to explain how the velocity of the music instrument could be implemented on the surface. By pressing more, more surface of the finger will get in contact with the plane, so by calculating the area of pressure for a single finger, the intensity of touch can be approximated.

If the area touched represents the velocity, having a denser surface (more strips with a smaller size) will surely better model the velocity. Opposite to that, if only one intersection is found under a finger, it will be harder to determine if just the tip or the full finger has been laid.

What before was described as "strips" meaning long rectangles of copper, may be conceived in a different way. In fact the shape proposed is referred to as being "diamond shaped"[1]. That means that each strip is a series of 90 degrees rotated squares (diamonds) all connected through a line. Using this disposition each diamond never overlaps with another one; the only intersection between the two layers is the small line connecting each diamond.

This reason might be one of the key points to have clear in mind while designing the surface: the intersection between rows and columns should be as small as possible, so that a little part of the finger can be used to track its position, without the need of fully pressing the finger's area on the surface. It is easy to assume that as thin wires as possible are needed. Using thin wires would leave most of the inside squares (part of the grid made of wires) empty. Expanding the wires only in the empty spaces, diamond shapes will be formed (Figure 2). The surface is designed to minimize the intersections between rows and columns, while maximizing the space covered by rows or columns.

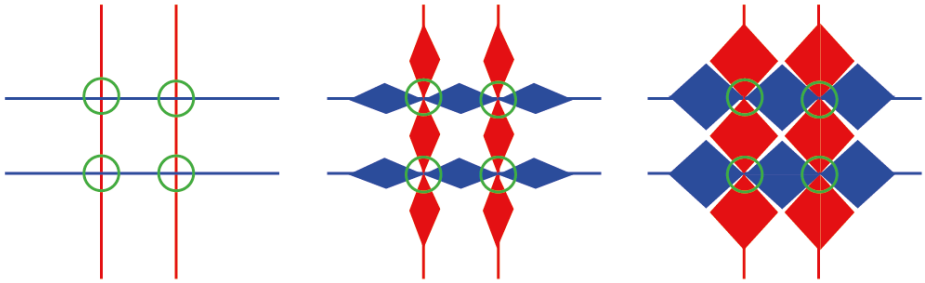

Figure 2: Lines expanding into the diamond shape

Now that the surface has been described in detail, The preferred shape chosen to implement the MIDI controller can be obtained.

Summing it up a surface used as a MIDI controller should:

1. Be used by both ends
2. Have no more than 12x12 strips
3. Have enough granularity to support velocity

The two chosen shapes are a square and a rectangle, made up of 4x4 and 5x8 probes. The size of the first touchable surface is 45x45mm, which gives enough space for up to 3 fingers to push onto. The 8 probes are built from a copper tape, with each probe having a diamond shaped layout, where each diamond has a diagonal of 1cm. Each row/column has roughly 2mm of space before and after, ensuring that two adjacent probes can not touch and shorten the circuit. The rectangular surface is 50x95mm in size and uses the same 1cm square diamonds' layout. It is longer to be able to perform more melodies base actions, so that an octave of a musical scale can fit nicely on the length of the surface.

The surface should be grounded correctly not to add any additional interference, that is why it is preferred to keep all the components on a plastic surface.

3.3 Electric components

All the components used in the project are easily findable in hardware stores and online. They are inexpensive as well, and might be found already in maker's houses. Starting

from the commodity microcontroller, the Arduino Mega has the most number of analog pins among the Arduino family. Any other Atmel AVR microcontroller can be used too, in fact there is a guide on how to make the multi-touch control surface with the Arduino Uno [15].

The other IC (wrapped around a module) used in the project is a multiplexer, it can be found in various models on online stores, although the recommended one is the CD74HC4067[1]. The multiplexers, along with some 100k resistors connected both to ground and analog pins, form the rows and columns of the surface.

Apart from these components, just a few wires are required to connect all components together, particularly by attaching to ground (with 100k resistors) all the analog pins of the microcontroller, while exposing the pins to the rows or columns of the surface grid. The module CD74HC4067 is then attached to Arduino by some digital pins 3 and 5v/ground are connected as well. Starting from the pin C0 of the CD74HC4067 each pin is connected to a row or column of the grid. The number of rows/columns on one side is limited to how many analog pins the microcontroller has, and to the other side the number of pins of the multiplexer.

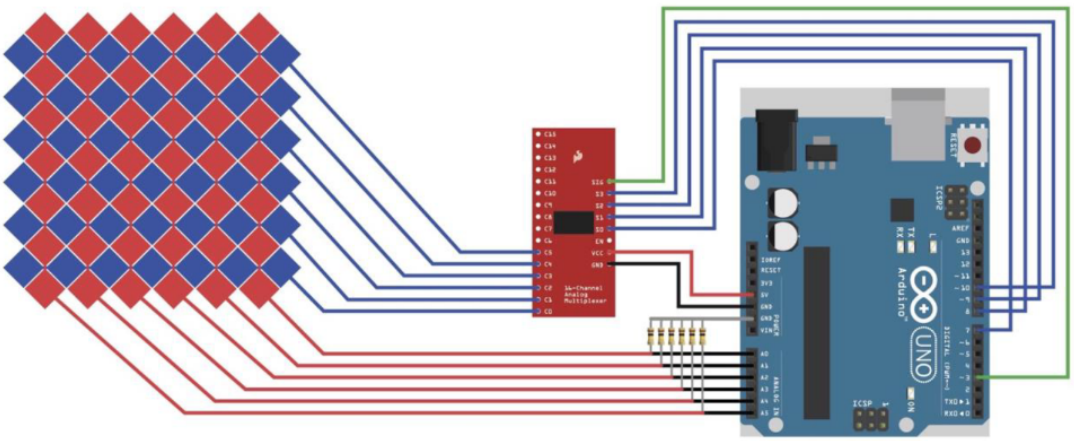

Figure 3: Basic connections with the Arduino Uno

As the multiplexer has more pins than the analog pins on the microcontroller, they will be used to connect the columns. Viceversa, all the rows will be attached to the analog pins. The receivers (RX) probes should always stay on top, so accordingly both the TX and RX probes can be used both as rows or columns. According to the two types of dispositions (TX as rows, TX as columns) the order of pins connected to the rows and columns might be

reversed, to map them according to the MultiTouchKit library specifications. Other than that, the Processing's MultiTouchKitUI library will always put the RX probes as columns with the most left probes attached to the A0 pin. Viceversa, the rows will be considered as TX and connected to the multiplexer, with the far most top row connected to the MUX 0 pin.

Chapter 4

Implementation

This chapter describes how the raw data calculated by the microcontroller gets translated to MIDI events, and eventually to sound. This process can be achieved in many different ways, encapsulating the computation more or less to the microcontroller. Here's one way to do it, trying to clearly separate all the subprocesses.

4.1 Technologies flow

Both the processes of calculating the position of the fingers on the surface and the translation of these positions to MIDI could theoretically be done on the microcontroller, although the process of blob detection is not as implemented on commodity microcontrollers as in specific ones (as the Arduino Portenta, which retails for 99 dollars)[16]. It was then chosen to separate the two processes clearly: the Arduino handles the translation from touch events to raw data, and the computer handles the blob detection and blob to MIDI events translation.

The first process is about calculating through the capacitance of the finger, his position on the copper matrix. This is achieved by differentiating the probes (rows and columns) as transmitters and receivers. The multiplexer sends some signals in the MHz range to the transmitters, that are then received from the receivers and go to the analog pins available to the microcontroller. As the signal in the selected range (<40 MHz) propagates more passing through the finger than the air, a higher amplitude of the signal is received when the finger is near the surface, thus detecting the touch. The finger acts as a conductor from the transmitters to the receivers. Sending and receiving these signals is handled by the MultiTouchKit library [17], that exposes the pins where the components are connected to, the size of the grid, and two modes: the first will return only the touched and not-touched state of the intersections (as push buttons) while the second one will return the raw data calculated. The row data, needed to calculate the blobs, are sent to the serial port as comma separated values, one line per row intersections. Each line will start with the number of

the row, and its length will be as the number of columns set. The Processing script will then complete the second process: blob detection and blob to MIDI events translation[18].

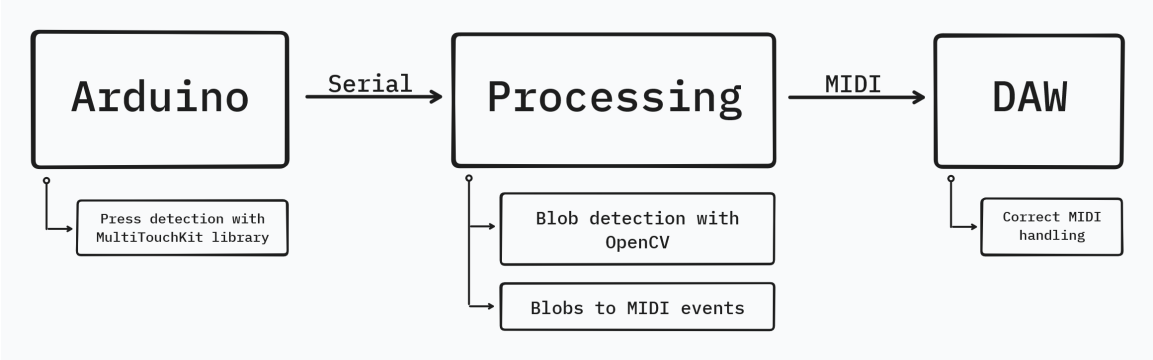

Figure 4: The technologies flow

The Processing script needs the MultiTouchKitUI library [19] to work: this library create the right structure to expose the blob objects while hiding the actual computation needed for their detection, which is handled by the OpenCV library [20] and the BlobDetection dependency [21]. Once the number of transmitter and receiver lines is set (tx and rx), the threshold fot the blob detection and the MultiTouchKit object is created, the library will start reading from the serial port the incoming data. A calibration process will be executed at the script startup: it ensures that a baseline is set to perform all the calculations according to that. The baseline is the state of the surface not being touched by anything. After the calibration, a visual representation of the grid will be given. By setting true the interpolationCubic, the library will perform a gaussian blur to the grid before performing the blob detection. Finally the blobs will be graphically shown in real time on screen. By fine tuning the threshold and the maxInputRange variable, a better detection can be performed, according to the physical surface materials and dimensions. The blob objects exposed by the library remain null until a touch is detected, and then assigned by the order in which the fingers press onto the surface. The first time the surface is touched will always be assigned to the blob with id 0. The blobs are catalogued by an id (i) and they have a position (X,Y), a width W and height H. The width and height will help calculate their area and therefore the velocity on attack and quantity of modulation during the press. These objects have to be translated then to some MIDI events. One of the key objectives is how the objects represent sounds, and that use case directly translates on what's the appropriate way to handle the objects to midi events translation.

There are three different use cases that will be presented for the last step. The first is about emulating a classical midi controller, by mapping the X value of the surface to a key of the piano (with one octave per line) and vertically extending the octaves in rows 5.

Figure 5: Layout of a piano keyboard on the surface

The second use case is about using the surface to control a virtual instrument that has a more contemporary approach to synthesis, which is a granular virtual synth 6. The X axis on the surface represents the position of the grain selected at touch (which can be then moved around left and right) and the Y axis represents the speed of playing of the grain. There is a Z axis too (which is the velocity, so the size of the blob) that can be mapped to the size of the grain, so by pressing more on the surface the grain of the granular synth will increase.

Figure 6: Ableton's granular synth (Granulator)

The third use case, designed for a small surface area and a low number of probes, is a launchpad. Dividing the surface in N even squares, each one can be considered a pad, where a note will be sent on touch. One example could be a 2x2 squared layout (Figure 7-a) or a 3x3 one (Figure 7-b). The velocity of the played note is represented by the area

of the pad covered by the finger. By playing harder or purposely covering the space more, a fuller sound will be played.

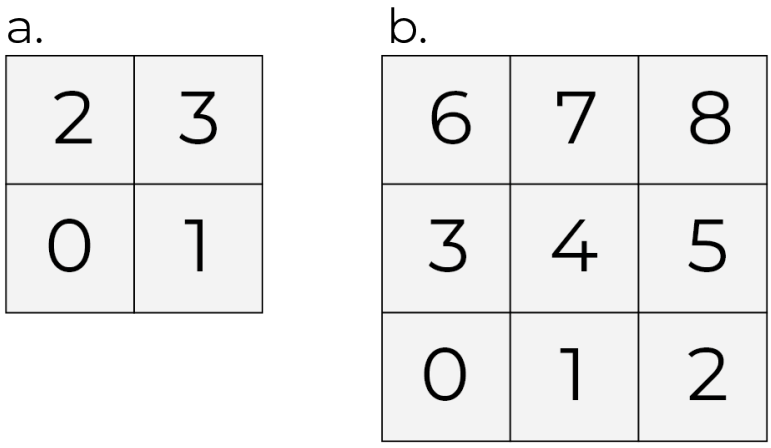

Figure 7: Launchpad's layout

Moving forward, these MIDI events are then sent to a DAW that will assign them accordingly as notes or CCs, effectively controlling virtual instruments or the DAW interface as well. According to what type of midi structured information has to be sent, different types of MIDI protocols can be used.

4.2 MIDI

MIDI is the protocol that will be used to send information to the DAW effectively controlling the virtual instruments. There are different types of specific MIDI protocols that can be used with different levels of trade-offs to the functionality of the surface.

Starting from MIDI 1.0, each blob can correspond to a midi note having the pitch related to the X axis of the surface and the velocity linked to the Z (or area) of the blob. Moving the finger up or down (in the Y direction) would bend or modulate the note. The main limitation of using MIDI 1.0 like this is that the modulation would affect all notes touched in that moment. A solution to that problem would be to use different CCs and assign them to each blob, and then DAW wise each CC would need to be assigned to that type of modulation.

A great way to overcome using different CCs and mapping them right is to use MIDI 1.0 multi-channel. By assigning one channel per blob, all the standard modulation CCs can be maintained and polyphonic aftertouch can be achieved. The big tradeoff of this method is using multiple channels for the instruments, considering that the full polyphony could be at a maximum of ten fingers (tested on the biggest surface of 16x16 probes [1]), than means using 10 of the 16 channels available in MIDI 1.0. A great benefit is still achieved by controlling each blob in a different way. As an example the first blob could have a bass sound while the others could emulate strings in unison, with every channel having its own characteristic and different modulation.

Midi 1.0 multichannel could be configured as MIDI MPE, so by standardizing the use of different channels and CCs. The MIDI Polyphonic Expression standard divides the 16 MIDI channels into zones, where a zone is a group of contiguous MIDI channels comprising a Master Channel and one or more Member Channels [22]. Two different types of zones can be defined, although only one can be used. The Lower Zone has a Master channel which is MIDI channel number 1, and the Members Channel going from 2 to n. The second zone that can be defined is the Upper Zone, having as the Master channel the MIDI channel 16, and as the Members Channel the MIDI channels 15 through n. If just the lower zone is used, the MIDI channels 2 to 15 can be used as Members Channels, leaving out MIDI channel 1 as the Master Channel of this lower zone and MIDI channel 16 as the unused Master Channel of the higher zone. The differentiation of channels as Master and Members implies that a modulation sent to the Master channel will affect all its Members too, effectively using the master-slave paradigm. Although seems a natural evolution of the MIDI 1.0 multichannel, the MPE standard's overhead data would in this particular project outweigh the plain MIDI 1.0 messages, as the load of modulation data is very low (2 CCs maximum per note), thus making the use of MPE impractical.

The Midi 1.0 multichannel has been then chosen to clearly handle the blobs, where each blob has its own channel. The main class that handles the conversion from OpenCV's blob to a MIDI packet is called BlobMidi, it is instantiated at the Setup of the Processing script, and updated during the Loop. Each OpenCV blob object is null when no fingers touch the surface, so that no new objects are created on run time. If the object becomes non-null during the Loop, the BlobMidi.update() is called, where these passages happen:

1. If the object has been set to active, a new note or a modulation CC is sent.

2. If the object turns to non-active, a NoteOff is sent, considering a debounce on release so that the note off message doesn't get triggered multiple times.

At each press on the surface, a new note is sent for the first time, with the note corresponding to its X position of the surface, and the overall size of the blob as his velocity. As the finger is being kept on the surface, a CC is sent to modulate that note. Most of the customization relies here, on how to modulate the note considering both his

changeable coordinates and size. Wherever the user removes a finger from the surface, the blob becomes inactive and a NoteOff event is sent, having the same note as the original NoteOn. The debounce on release is set and the debounce on attack reset, so that no multiple NoteOff are sent and the BlobMidi is ready to receive another NoteOn. A rapid fire of pushing and releasing caused by the surface not being able to sense the touch properly could send many notes on messages but not as many notes off, keeping the notes from releasing properly. This inconvenience can be resolved by sending the control change number 123 instead of the NoteOff, which corresponds to the message AllNotesOff. As each channel is monophonic, the CC123 will affect only the note corresponding to one finger. Various types of normalization are performed to the coordinates of the blob as they are always a float between 0.0 and 1.0 . The X coordinate is normalized to fall under 0 to 11, which represents the MIDI first octave, 0 being C0 and 11 being B0. The Y coordinate is normalized as the number of octaves available for the height of the surface.

Chapter 5

Tests

5.1 Protocol

5.1.1 Latency

Before performing any type of tests a few technical information will be displayed. That includes what type of microcontroller is used, based on what IC, and how many analog pins are available. Then how many of those analog pins are used for the surface and how many pins of the multiplexer are being used. Regarding the surface, how many rows and columns it has, what type of layout has been chosen (strips, diamonds) and any technical measurements about the layout, such as the size of diamonds and the size of the intersection between rows and columns. Finally, what type of material the probes are built from and the material of the dielectric layer in between them. The tests performed fall under two distinct categories:

1. Instrument functioning tests
2. Usability tests

The instrument functioning tests take care of analyzing how well the surface responds to touch, the area covered by the touch surface and the latency of the instrument. Latency is a big part of instruments' design as it shows how responsive and reactive an instrument is. From when the user starts to touch the surface, until a sound is heard out of the DAW, the latency should fall under the 20ms threshold set not to have any delay perception on the playing. The latency will be calculated by comparing using a DAW both the sound made by pushing the finger on the surface and the MIDI event. A microphone will be attached to a commodity audio interface and pointed directly to the surface, where the sound will be recorded. Viceversa, the MIDI impulse will be directly recorded to the DAW. By routing the two signals through the DAW, its buffer latency will be bypassed, as it insists on both signals (Figure 8).

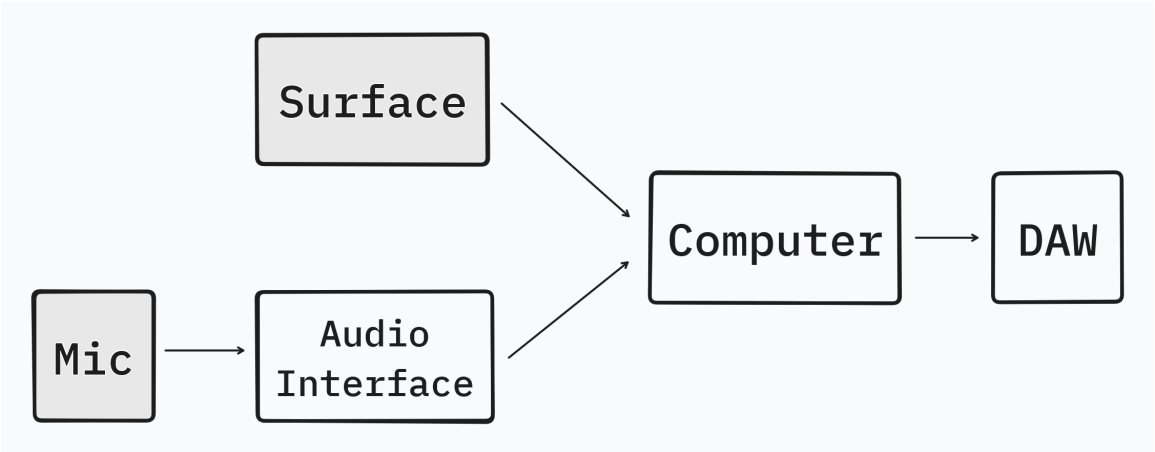

Figure 8: Latency test flow.

5.1.2 Spatial Accuracy

The next test will cover the touch of the surface, and how well the surface can sense a gesture performed by finger, precisely analyzing these motions:

- Horizontal movement
- Vertical movement
- Diagonal movement

The vertical motion corresponds to passing the finger both right over the column probes and in between them, and the same action will be performed for the rows. The diagonal motion covers the passage from both the rows, the columns and the space in between them[23]. All these movements will be recorded by moving the finger on a straight line. To ensure the linear motion of the finger, a rectangular hole was laser-cut from a thick cardboard that will perform as a track for the finger to move through (Figure 9). The data will be then saved as a CSV and the error distance from the theoretical path calculated using Python with NumPy. For every test, 10 tries will be recorded, with a mean of 100 samples for each try. On horizontal and vertical gestures, the root mean square error is calculated respectively on the average X and Y of the chosen row and column of the test. On the diagonal gesture the points sampled (X,Y) will be used to calculate the root mean square error from the theoretical line passing through the corners of the interface, from coordinates (0.0,0.0) to (1.0,1.0).

a. Diagonal rail

b. Vertical rail

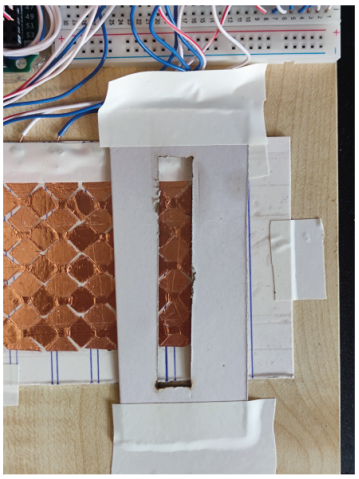

Figure 9: Thick cardboard's laser cut rails.

The usability test comprehends the ways the users interact and explore the surface. For that mean, the user will have a period of 10 minutes to freely explore the surface, play with the sounds and discover its capabilities. The user will be only instructed on what use case the surface is set on: launchpad mode, piano mode (with a pentatonic scale), granular synth mode.

Some information about the user will be asked, such as the age and any connection to the music field (music lover/listener, hobby musician, professional musician, music technician). After this period of time a feedback will be asked and reported. It will be particularly interesting to know:

- If the sound feedback from the surface best represent the positioning of the fingers
- If no physical feedback (as the surface is flat and without buttons) compromise the usability of the instrument
- If the area of the finger is a good representation of the notes' velocities
- If the area of the surface is too small, too big or enough for the use case.

5.2 Results

5.2.1 Arduino Uno based surface

The first group of tests will be conducted on a surface that relies on the Arduino UNO microcontroller, based on the ATmega328P IC (Figure 10).

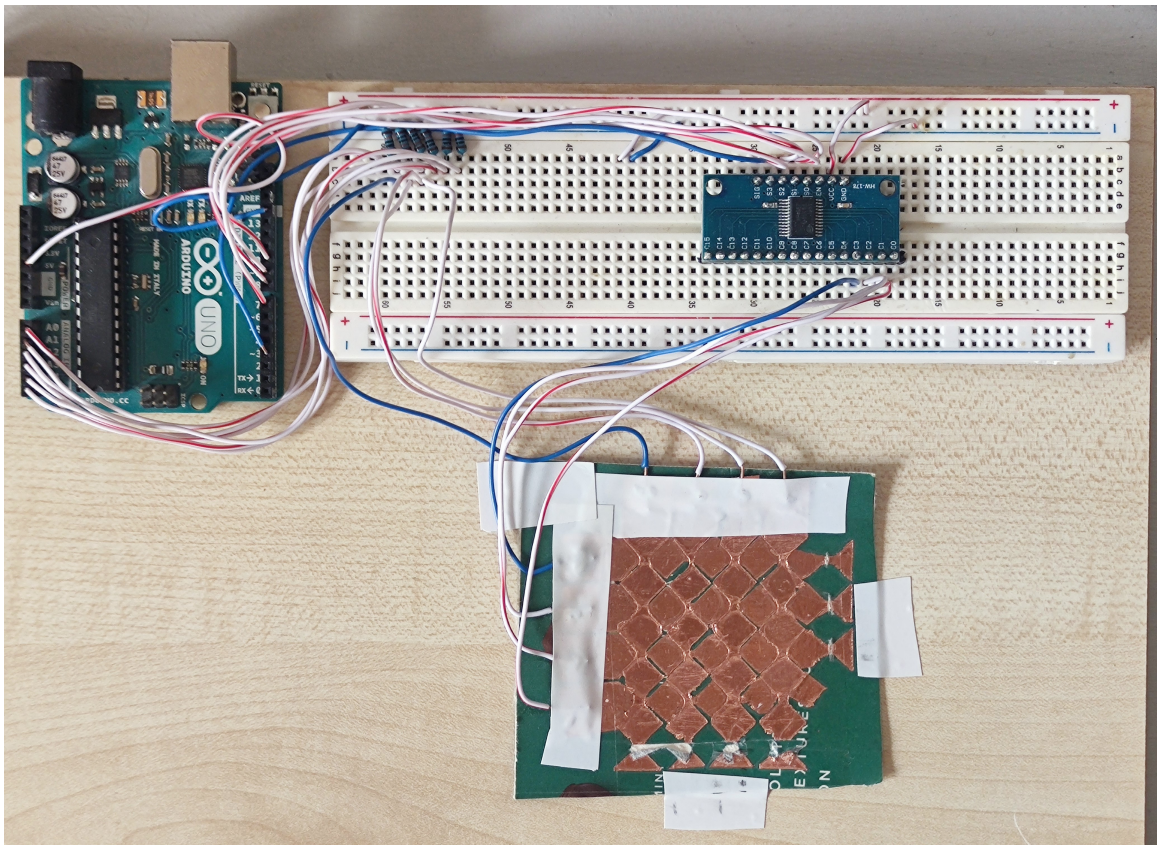

Figure 10: Arduino Uno based surface top view photo.

The analog pins available to the board are six, and four of them are used for the surface. The multiplexer has eleven pins available, in which four of them are being used. The surface is a square grid of 4x4 probes ,45x45mm in size and the overall area is $4.5cm^2$. Each probe is formed by four diamonds, each one 1x1cm in size (diagonally). The diamonds are connected by a strip passing through their center, 1.5mm wide. The probes are built from a copper tape, and the layer between them is a clear plastic tape. The columns (RX) are on top of the rows (TX) (Figure 11).

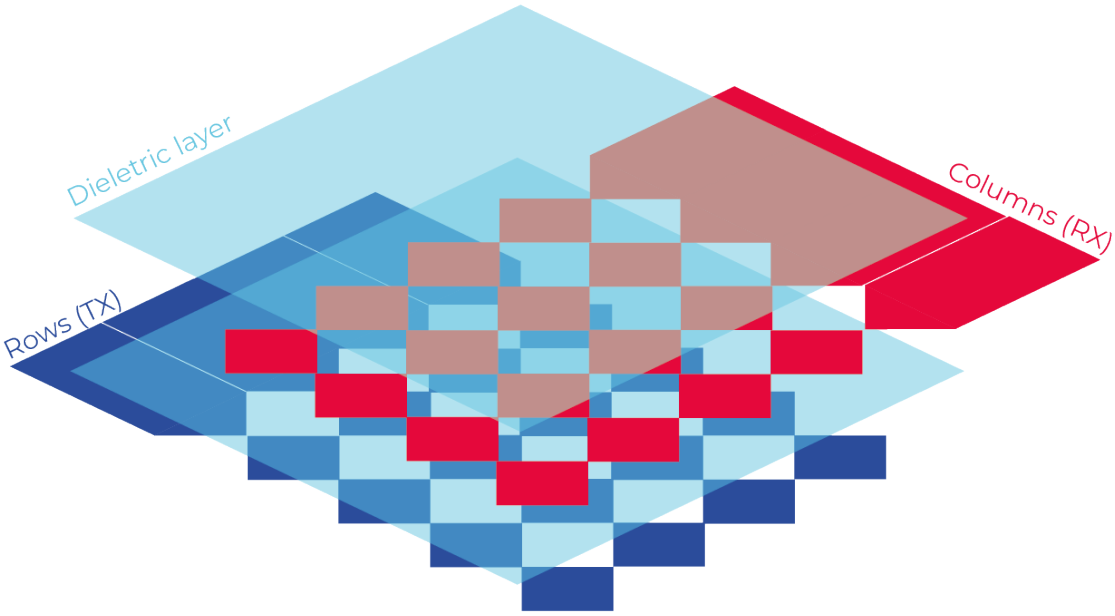

Figure 11: Isometric view of the surface.

Recording ten samples of the latency of the instrument, the mean is 21.9ms and its standard deviation 6.22ms (Table 1).

Latency (ms)
24
21
33
16
28
24
23
10
16
24

Table 1: Latency between surface's on touch sound and midi impulse.

The second group of tests performed is about calculating how precise the surface is when dragging a finger on a straight line. As previously explained, the finger was

dragged on a straight rail hole to ensure no errors were introduced from the movement of the finger. 10 horizontal,vertical and diagonal gestures were recorded, with each passage lasting 3 seconds, and the mean of samples logged per passage is around 100 coordinate samples. For each type of gesture (horizontal/vertical/diagonal) two variations were recorded: the first one is about dragging the finger directly on the row/column's probe (the copper diamond) and the second is between two adjacent probes. Figure 12-a shows the recorded samples of a diagonal gesture on the virtual strip composed of both vertical and horizontal diamonds, while Figure 12-b shows the performed gesture in between two adjacent diagonal rows. For every group of samples the root mean square error is calculated, and finally the average of the root mean square errors derived (Table 2).

	RMSE (mm)
Horizontal	0.596
Horizontal in between	1.040
Vertical	0.068
Vertical in between	0.458
Diagonal	3.278
Diagonal in between	1.072

Table 2: Straight gesture precision.

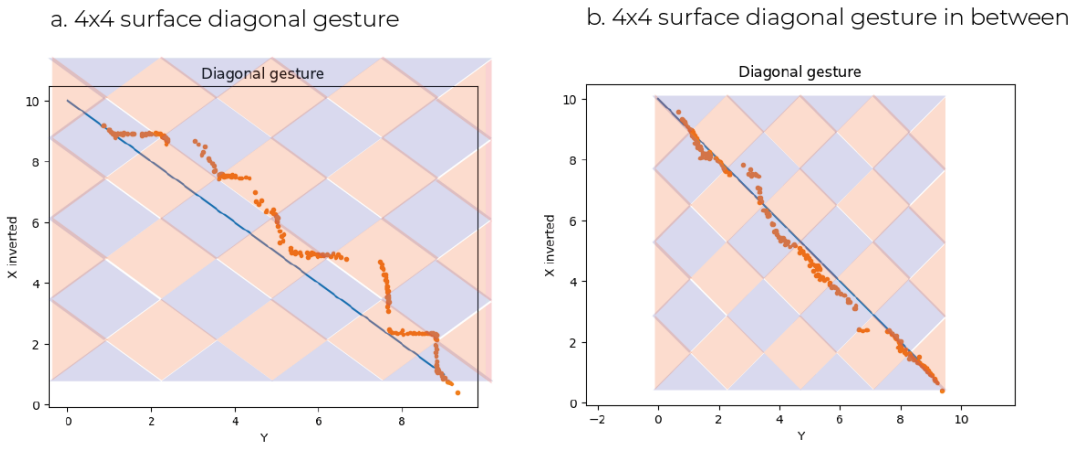

Figure 12: Recorded samples of a diagonal gesture.

5.2.2 Arduino Mega based surface

The first group of tests will be conducted on a surface that relies on the Arduino MEGA microcontroller, based on the ATmega2560 IC (Figure 13).

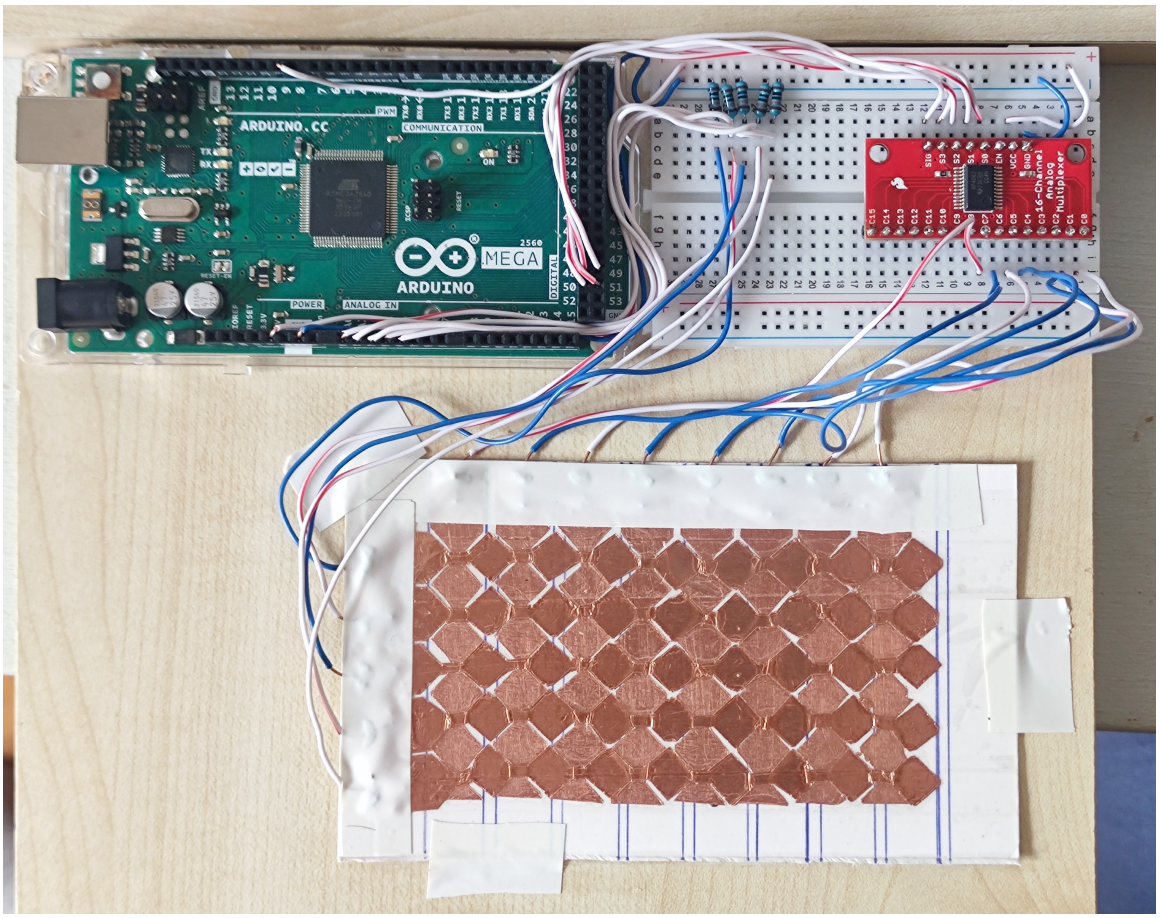

Figure 13: Arduino Mega based surface top view photo.

The analog pins available to the board are 16, and 5 of them are used for the surface. The multiplexer has eleven pins available, in which 8 of them are being used. The surface is a rectangle grid of 5x8 probes ,50x95mm in size and the overall area is $4.75cm^2$. Each probe is formed by four diamonds, each one 1x1cm in size (diagonally). The diamonds are connected by a strip passing through their center, 1.5mm wide. The probes are built from a copper tape, and the layer between them is a clear plastic tape. The columns (RX) are under the rows (TX) (Figure 14).

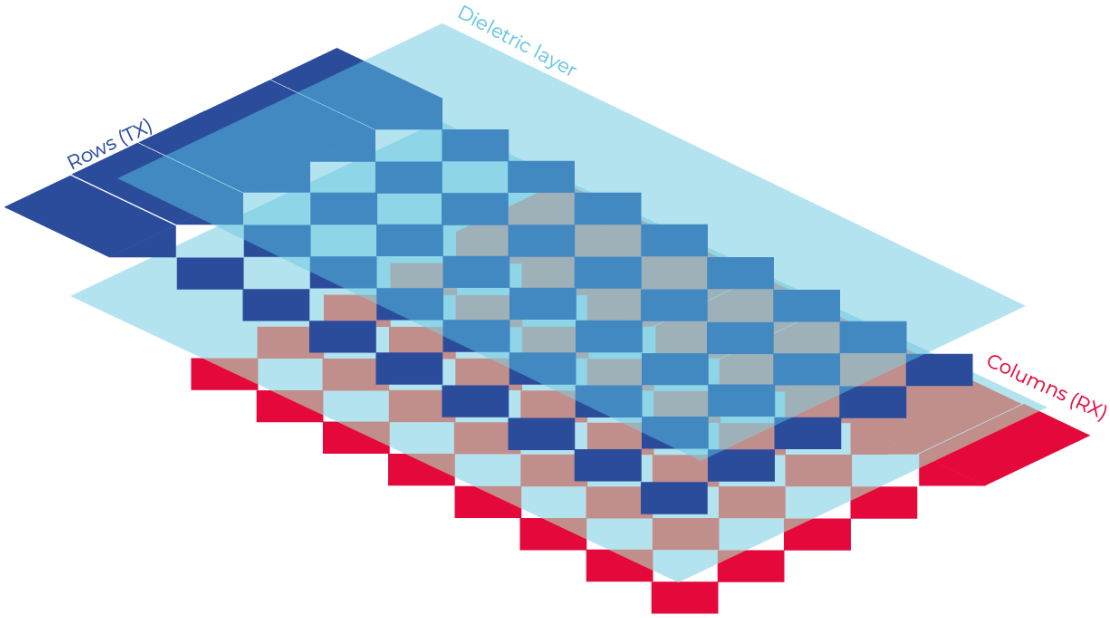

Figure 14: Isometric view of the surface.

Recording ten samples of the latency of the instrument, the mean is 26.6ms and its standard deviation 4.54ms (Table 3).

Latency (ms)
31
26
21
20
26
34
25
27
23
33

Table 3: Latency between surface's on touch sound and midi impulse.

The second test conducted is about the precision of the surface when performing a gesture, where the root mean square error is calculated from the theoretical line of the

gesture (Table 4). As previously described , a straight rectangle rail was used to ensure the straight finger's action on the surface (Figure 9. As the surface has not an even number of rows and columns (being 5x8) the only diagonal gesture recorded crossing the whole surface was the "diagonal in between" gesture

	RMSE (mm)
Horizontal	0.028
Horizontal in between	0.088
Vertical	0.008
Vertical in between	0.052
Diagonal in between	4.949

Table 4: Straight gesture precision.

The first user test was conducted on a male R, 25 years old, musician and synthesizers' player. R used the surface in two use cases, both the pentatonic scale piano roll (3 octaves) and the 2x2 launchpad with drums sounds, for 10 minutes each. The user was uninstructed on how to play the surface and on how the surface was subdivided, except knowing the use case of the test. After the 20 minutes a feedback was asked, and the previously cited questions arose. R started to talk about how the feel of the surface was very smooth, and how it expressed to be used with a light touch, so he started using the tip of the finger, but as the surface didn't sense that, he continued by using almost all the fingers' surface. Without any visual reference about the subdivisions of the surface and where are the limits to play one sound or the other, R seemed disorientated. He also tried to use the surface changing its orientation and using it as a console controller, with thumbs touching directly the surface. Other than that, the user found it interesting that the touch was coherent on all places of the surface being planar, and liked the idea of having a multi functional surface where the layout can be customized according to the instrument played or the use case. Finally R found the surface small to use in the piano roll use case, feeling the space between notes undefined, but with right dimensions for the launchpad use case.

The second user tested on the pentatonic scale use case is G, 28 years old, music listener and lover. The surface was visually subdivided (with some white lines) in two octaves of a pentatonic scale, and the user played the instrument for 10 minutes while listening to a backing track (with the same musical key as the pentatonic scale used). G never played music before, but found the experience fun and very enjoyable. He had technical difficulties at the start while touching the surface with the tip of the finger, but understood within minutes the correct position in order to hear a stable musical feedback. The visual grid subdividing the surface was found useful to understand where to play. He found the dimensions of the surface to be perfect for the use case, with his fingers covering more than half of each note's space. The user finally reported that it would be a nice experience to tap on the body or other curved surfaces instead of a flat one.

The first user tested on the launchpad use case is M, 60 years old, non musician nor particular music listener. The user tested the surface on the 2x2 launchpad use case, playing under a backing track. She found the surface fun to use as there was a drum kit on the palm of her hand, but thought the surface to be too small compared to the use case proposed. The instrument was said to be intuitive and great to use without knowing anything at all about music. The user found it strange to use the entire finger and not only the tip of it (such as using a smartphone) but with little time she got used to the difference in touch. She also thought it could be a fun experience to have the drums stuck on different parts of the body.

The second user is A, 19 years old, music listener and lover but non musician. The user tested the surface on the 2x2 launchpad use case, playing under a backing track. She described the surface as smooth and good to touch, and interested to notice that such a surface can play music. During the playing the user reported not to hear sound feedback correctly but to appreciate the velocity while applying different forces, thus becoming very organic as a real drum. The surface was reported to be at the right size for the use case, and the user said she was not used to using the entire finger to play it, but reported becoming easy after some minutes of practice. Finally she reported that a tool like this could spark in her an interest to learn music.

The final user that tested the 2x2 launchpad use case is S, 28 years old, hobbyist drummer and music lover. He reported the surface to be better than any entry level electronic drum kits, although with not too precise taps and sound feedback not entirely right all the time. The use of the full fingers was found to be personally slow compared to using just the tips. He would have preferred to have a rougher surface with a texture to touch on, and thought the surface to be a bit small for the use case proposed. Finally he reported the doubt of the surface to be that usable with dirty or oily hands.

5.3 Observations

The first data found was about the latency of the two surfaces. With a mean of respectively 21.9ms and 26.6ms, the instrument can be played by almost unnoticeable latency. It was found that the latency was stable throughout the tests with a respective RMSE of 6.22ms and 4.54ms. It was also noticed that the latency could change depending on the number of blobs detected on the surface at any given time, although no tests were performed to calculate the latency of simultaneously touching the surface with more than one finger. Then the precision on gestures was calculated, where the RMSE of horizontal and vertical motion were almost all under 1mm, meaning that the horizontal or vertical drift (on vertical and horizontal motion) is almost nonexistent for the use cases of the surface. Differently enough were the RMSE of the diagonal gestures, having a peak of 4.9mm on the 5x8 surface, whereas the surface is not squared, the finger drag through the surface passing unevenly between rows and columns probes. A bit different were in fact the RMSE

on the squared surface, where a difference in drift is noticeable while passing the fingers between on one row and column at the same time (Figure 12-a) or in between two probes (Figure 12). These RMSE values could not be so alarming, considering that the minimum area of the finger touching the surface being $1cm^2$ and the blurred version of it used for the blob detection is bigger than that.

It might be right to then consider having the most important gestures for the surface being horizontal and vertical (having a higher precision) and the less important ones, the diagonal gestures. In fact, the horizontal axis was chosen to represent the different notes, and the vertical axis the octaves.

The second observation is about the size of the probes' diamonds. Having smaller diamonds could greatly help representing the various motions and the velocity more accurately, although considering that augmenting the number of probes would augment the time of probes scanning, and the latency as well. Although the number and granularity of probes having a great impact on the usability of the surface, it is well recommended to perform a fine tuning (on the software side) between the two variables used for the blob detection process: threshold and maxInputRange (the brightness of the blob). The right balance between the two values is the one that allows no signal detected when the finger almost touches the surface, and the area of the finger's touching the surface is well represented with the size of the blob detected. It is also crucial (for the two use cases presented) that two fingers near each other (on two different columns) are represented by two separated blobs, and they do not merge into one. The maxInputRange variable, that represents the blob's brightness, can be changed in real time by the user interface made available through the Processing's MultiTouchKitUI library [19].

Regarding the user tests, the first observation is about the disorientation that the users felt because of not having any visual reference on where to play the notes and the space for each one. After putting some lines to subdivide the surface into smaller rectangles, every user had a more clear view on where to touch, even if the inside space linked to each sound was notably bigger than the finger area. The use of multiple fingers seems to be correlated to the space available on the surface, and all the users spread a maximum of 4 fingers on the surface. As there was no right orientation to the surface itself, it was interesting to see the users experiment on different ways to hold the instrument, to better accommodate the playing. As the finger pressed on the surface is vertical, some users rotated the surface so that the rectangles were vertically orientated as well. The test conducted on surfaces with lines on the top were found to be more usable, as there was a visual representation on where to put the fingers. Furthermore, if these subdivisions' rectangles were vertical (as vertical is the finger) the surface was reported to be more usable as well. Given these visual subdivisions, the users seem to consider them more like buttons, so more in a discrete form than linear. It seems likely that as the notes played were in a scale (so not linear, but discrete) the discrete division of the surface was more coherent with the audio feedback. The most talked about subject (cited by every

user) is about the use of the whole finger instead of just the tip. Both musicians and non musicians found it difficult at first to use the surface differently from music instruments or smartphones, so not by using only the tip of the finger to touch the instrument. After learning the new touch type, all the users used the surface correctly and even tried to implement different types of touches to modulate the velocity of notes played. Although able to adapt, all the musicians still found it hard to switch to the new finger positioning technique, not used in any other instrument played.

Chapter 6

Conclusions and future developments

6.1 Conclusions

The finished surface, made up of copper probes, allows to sense multiple fingers' touches at once, by carefully using capacitance as the technique to sense the touch gestures. Electric signals in the MHz range are sent to the probes by a multiplexer, and when a finger is placed on the surface, effectively acting as a conductor, the signal comes back to the analog ins of a commodity microcontroller. Here, via the microcontroller, the raw data is calculated and sent to the serial port, to the PC.

The machine, through the programming environment Processing and a few libraries (OpenCV, blobDetection) analyze the data and detect the touch points as blobs. These blobs are then converted into MIDI messages with parameters that depend on the different use cases proposed such as a piano keyboard, a launchpad or a granular synth. These MIDI messages are sent through a virtual MIDI port to a DAW, where the sounds are effectively generated and the playing is effectively performed. The X and Y axis of the surface are used to control different parameters of the virtual instrument or the performance.

The strength of the surface relies firstable on the cost, as the components used are low level and globally easely commercially available. The part list for the finished prototype is small too, as it includes only one microcontroller, a multiplexer, some resistors and cables and the surface made up by hand cut copper tape probes. It can be all prototyped fast on a breadboard using jumper wires, and the microcontroller runs an almost stock piece of software that remains unchanged after the first setup. After the initial setup is done, all the changes can be performed easily just on the software that handles the conversion, where the Processing environment offers a lot of flexibility and integration to many audiovisual applications.

The weakness of the prototype stands as a good amount of time has to be spent around the surface itself. Firstable by finding the right combination of materials to build it, as copper was chosen for its availability and speed of cutting/handling, but might not be the

best fit for some specific applications different from the here presented one. The process involves a lot of labour if no specific tools are available, as every probe has to be manually cut and the space between them calculated correctly in order not to have any distortion of the matrix lines. The probes should be carefully designed too; the ones presented were taken from [1] but could not be the best option to maximize the coverage of the surface while minimizing the space of crossing between probes.

The other main weak point is about the number of various pieces of software that have to be linked together for the surface to actually perform. Every piece should connect right for the instrument to play. On the software side the surface has to be greatly fine tuned depending on the environment, as wet hands or surroundings could change the sensitivity, thus the playing. Lastly, this surface has to be considered a mere prototype surface to test and implement the correlation between planes interaction and playing, more than a commercial product, as it would appear to be difficult to scale on a commercial level due to the great variability of materials and software pieces.

All the source code and tutorials on making the surface work are available on the official Github page[1] of the Laboratorio di Informatica Musicale (Music Informatics Laboratory) of the Department of Computer Science, Università degli Studi di Milano.

6.2 Future developments

Here presented was a planar surface around which the user interacts to play music. The first proposition of future developments is about rethinking the surface as not planar. As in [1] a test was performed with the probes stacked on a 3d printed bunny figure. How would a non planar surface shape the human-instrument interaction, and how would it affect the design of the probes?

The surface could be re-thought of having a bigger area (such a wall) or a really dense structure, and how big surfaces could be utilized into music (or pedagogic) research. As most of the tests performed were about a surface made by copper tape probes, the other two main types of techniques (copper thread sewing, ink pen drawing) could be tested, to well compare all the different types of crafting, seeing if each has its own different use case and precision of sensing. The use of different materials could potentially lead to discovering new applications or structure of such a surface, where probes could be clustered without needing a constant spacing or density. That leads to studying the tessellation of the probes, as the ones tested were only based around the concept of expanding lines, thus using only diamond shapes (Figure 2). Design changes to the shape of probes could lead to optimizing the shapes for the various applications and to discover more about how to best utilize the sensing approach proposed. Recursive shapes tessellation could lead to analyzing how more or less dynamic dense structure would affect both the

[1] `https://github.com/LIMUNIMI/Multi-touch-midi-surface`

sensing and playing[23].

On the software side research on new ways to handle both the blobs and the blob detection could be performed, by customizing the MultiTouchKitUI library [19]. Blobs are detected by performing a Gaussian blur on the raw values of the probes, but what if the horizontal and vertical blurs were different? would it improve jumping more precisely from a note to the other horizontally (with a lower blur) and maintain the smooth slide action vertically (with a higher blur)?.

About the hardware side, the only microcontrollers tested were of the Arduino family using the Atmega IC; other types of microcontrollers could be tested, such as the ESP family, to further expand the capabilities of the surface.

Lastly, all the applications revolved around music making, but having a multi-touch planar surface could mean thinking about all the other types of uses, from software shortcut and gestures managers, to other types of communications standards used as OSC and DMX.

Bibliography

[1] Joseph A. Paradiso3 Jürgen Steimle Narjes Pourjafarian1, Anusha Withana2. Multi-Touch Kit: A Do-It-Yourself Technique for Capacitive Multi-Touch Sensing Using a Commodity Microcontroller. 2019.

[2] Abid Rafique Muhammad Samiullah, Muhammad Zohaib Irfan. Microcontrollers: A Comprehensive Overview and Comparative Analysis of Diverse Types. 2023.

[3] Leah Buechley and Michael Eisenberg. The LilyPad Arduino: Toward Wearable Engineering for Everyone. *IEEE Pervasive Computing*, 7(2):12–15, 2008.

[4] 4 Touch Panel Types – Explained. `https://www.viewsonic.com/library/business/touch-panel-types-explained/` [Accessed: (2024-04-03)].

[5] Rick Downs. Using resistive touch screens for human/machine interface. *Analog Applications Journal Q*, 3:5–10, 2005.

[6] Frank Beck and Bent Stumpe. Two devices for operator interaction in the central control of the new CERN accelerator. Technical Report. CERN. 1973.

[7] Stylophone. `https://stylophone.com/genx1/` [Accessed: (2024-04-03)].

[8] The Linnstrument working. `https://www.viewsonic.com/library/business/touch-panel-types-explained/` [Accessed: (2024-04-03)].

[9] Haken Continuum. `https://www.hakenaudio.com/` [Accessed: (2024-04-03)].

[10] Roli Seabord. `https://roli.com/products/seaboard/rise2` [Accessed: (2024-04-03)].

[11] Ableton Push. `https://www.ableton.com/en/push/` [Accessed: (2024-04-03)].

[12] Asta Roseway Andres Calvo Hsin-Liu (Cindy) Kao, Christian Holz and Chris Schmandt. Rapidly Prototyping On-skin User Interfaces Using Skin-friendly Materials. 2016.

[13] David Holman, Nicholas Fellion, and Roel Vertegaal. Sensing touch using resistive graphs. 06 2014.

[14] Neil Cameron. *ESP32 microcontroller features*, pages 641–682. Apress, Berkeley, CA, 2021.

[15] Multi-touch Kit with Arduino Uno tutorial.

[16] Arduino Portenta. `https://store.arduino.cc/products/portenta-h7` [Accessed: (2024-04-03)].

[17] MultiTouchKit Arduino library. `https://github.com/HCI-Lab-Saarland/MultiTouchKit` [Accessed: (2024-04-03)].

[18] Processing. `https://processing.org/` [Accessed: (2024-04-03)].

[19] MultiTouchKitUI Processing library. `https://github.com/HCI-Lab-Saarland/MultiTouchKitUI/tree/master` [Accessed: (2024-04-03)].

[20] OpenCV library. `https://opencv.org/` [Accessed: (2024-04-03)].

[21] Blobdetection library. `https://www.v3ga.net/processing/BlobDetection/` [Accessed: (2024-04-03)].

[22] MPE specifications. `https://d30pueezughrda.cloudfront.net/campaigns/mpe/mpespec.pdf` [Accessed: (2024-04-03)].

[23] Humza Akhtar and Ramakrishna Kakarala. A Methodology for Evaluating Accuracy of Capacitive Touch Sensing Grid Patterns. *Journal of Display Technology*, 10(8):672–682, 2014.

Project developed at the Laboratory of Music Informatics (LIM)
https://www.lim.di.unimi.it