ESCUELA DE MÚSICA

DESCONTROL MIDI: CREACIÓN DE UN CONTROLADOR MIDI "DO IT YOURSELF" BASADO EN UNO REALIZADO POR DANIEL JANSSON Y PRODUCCIÓN DE UNA PERFORMANCE CAPTADA EN VIDEO HACIENDO USO DEL DISPOSITIVO.

AUTOR

Samik Marcelo Zurita Londoño

AÑO

2021

ESCUELA DE MÚSICA

Descontrol MIDI: Creación de un controlador MIDI "Do it Yourself" basado en uno realizado por Daniel Jansson y producción de una performance captada en video haciendo uso del dispositivo.

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Licenciada en Música con especialización en producción.

PROFESOR GUÍA

Juan Fernando Cifuentes

AUTOR

Samik Zurita

AÑO

2021

DECLARACIÓN PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Descontrol MIDI: Creación de un controlador MIDI "Do it Yourself" basado en uno realizado por Daniel Jansson y producción de una performance captada en video haciendo uso del dispositivo, A través de reuniones periódicas con el estudiante Samik Marcelo Zurita Londoño, en el semestre 2021-20, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

M.M. Juan Fernando Cifuentes Moreta

DECLARACIÓN PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Descontrol MIDI: Creación de un controlador MIDI "Do it Yourself" basado en uno realizado por Daniel Jansson y producción de una performance captada en video haciendo uso del dispositivo, del estudiante Samik Marcelo Zurita Londoño, en el semestre 2021-20 dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

David Fernando Acosta López

17221644068

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

Samik Marcelo Zurita Londoño

1723521330

AGRADECIMIENTOS

Agradezco a mis padres, Paula y Ernest, que me han apoyado siempre y creyeron en mí desde el inicio de mi carrera, además de ayudarme con la parte técnica de este trabajo. A mi hermano Íkiam por inspirarme con su trabajo de titulación a hacer algo que se saliera de todo lo convencional.

A mis compañeros de la Universidad, en especial a Samuel y Mateo que recorrieron este camino conmigo. A mi amigo Javier con quien me inicié en la música desde muy pequeño y a quien siempre he admirado. A Seong Ju Han por ser mi amigo de toda la vida y alegrarme siempre. A Juan Fernando por ser el profesor más cercano que tuve y a quien considero un amigo. A María Emilia que hace años me dio el último empujón para arriesgarme a seguir esta carrera. A Stephy y Nicole, que me acompañaron y me dieron fuerzas cuando más las necesité y a quienes les contaba de mis ideas para este proyecto, además me hicieron mejor persona.

Por último, a todos mis amigos y a la escuela de Música de la Universidad de las Américas.

DEDICATORIA

Para los entusiastas del audio y la tecnología.

RESUMEN

El avance de la tecnología y la cultura del internet han promovido una tendencia llamada DIY (*do it yourself*) donde gracias a una infinidad de tutoriales que se pueden encontrar en Youtube, se puede llegar a construir casi cualquier cosa a mano, esto en combinación con el avance de la impresión en 3D y la existencia de las tarjetas Arduino pueden ser el inicio de una nueva cultura de autofabricación de herramientas en un futuro no muy distante. Este proyecto abarca la fabricación de un controlador MIDI y la producción de una performance captada en video haciendo uso del dispositivo.

Un controlador MIDI es un dispositivo que sirve para controlar parámetros en *Digital Audio Workstations.* Para poder construir dicho aparato, en este trabajo se parte de un tutorial creado por Daniel Jansson para su canal de Switch & Lever en Youtube, donde construye un equipo similar. Se analiza el dispositivo de Jansson, desde el diseño del dispositivo hasta la programación de los diferentes componentes y la producción de su carcasa. A partir de esto se elabora una metodología de trabajo para comenzar a construir un dispositivo similar, sin embargo, personalizado para tener un propósito distinto al del creado en el tutorial.

Se comienza por el diseño del dispositivo, en este punto surgen dos alternativas para la forma de su carcasa y se modela en 3D la alternativa principal, además se planifica qué componentes se adquirirán para realizar el controlador. Finalmente se lleva a cabo una maqueta de las conexiones, esta se utiliza para realizar la programación de cada componente conectado, lo que culmina en un dispositivo MIDI funcional.

Finalmente se realiza la performance capturada en video con el controlador MIDI hecho a mano, mapeándose a diferentes controles del DAW Ableton Live. Se concluye con varias recomendaciones para proyectos similares y un breve análisis del costo de la realización del dispositivo.

ABSTRACT

The technological development and the internet culture have promoted a trend called DIY (do it yourself) where thanks to an infinity of tutorials that can be found on YouTube, people can get to handcraft almost anything, this in combination with the development of 3D printing and the existence of Arduino boards may be the start of a new culture of self-manufacturing tools in the not-too-distant future. This project includes the manufacture of a MIDI controller and the production of a performance captured on video using the device.

A MIDI controller is a device used to control parameters in Digital Audio Workstations. In order to build this device, this work parts from a tutorial created by Daniel Jansson for his Switch & Lever channel on YouTube, where he builds a similar device. The Jansson controller is analyzed, from the design of the device to the programming of the different components and the production of its case. From this, a work methodology is developed to start building a similar device, however, customized to have a different purpose than the one created in the tutorial.

The project begins with the design of the device, at this point two alternatives arise for the shape of its case and the main alternative is modeled in 3D, in addition, different components that will be used in the device are chosen. Finally, a model of the connections is carried out and is used to carry out the programming of each connected component, which culminates in a functional MIDI device.

Finally, the performance captured on video is made with the handmade MIDI controller, mapping it’s components to different controls of the Ableton Live DAW. It concludes with several recommendations for similar projects and a brief analysis of the cost of making the device.

Introducción

La tecnología es uno de los factores más importantes que determinan el curso de la humanidad. Con el tiempo se ha vuelto cada vez más importante para un profesional conocer y estar implicado en el uso y desarrollo de nuevas tecnologías. A la par, una cantidad innumerable de avances han implicado también la reducción de costos de muchos productos y servicios que antes eran exclusivos para la élite que podía financiarlos, mientras que ahora están al alcance de gente promedio.

En el área de la producción musical se han abaratado los costos al punto de que casi cualquier persona puede llegar tener un estudio en su hogar. Algunas de las tecnologías que han permitido que esto pase son los programas de edición de audio, el estándar MIDI y el internet. Este último no solo ha aportado como medio para conseguir los productos a través de mercados en línea, sino también con tutoriales, ideas y comunidades.

Hay una tendencia en internet llamada “DIY” (*Do it Yourself*), donde gente construye todo tipo de objetos bajo sus propios medios, desde adornos hasta robots. Una de las razones por las que ha prosperado el DIY fue la creación de las impresoras 3D y la popularización de las tarjetas Arduino; con estos implementos es posible crear una gran variedad de implementos tecnológicos a un costo que suele ser menor que si se adquiriera un producto con características similares. Uno de los productos que es posible crear con estos implementos son los controladores MIDI.

Este proyecto busca construir un controlador MIDI a partir de la información obtenida en un tutorial de YouTube donde se crea este dispositivo a partir de una tarjeta Arduino, una carcasa impresa en madera e implementos electrónicos. El diseño del controlador que se realizará en este proyecto será diferente al del tutorial.

Debido a lo relevante que puede llegar a ser en el futuro la tendencia *“Do it Yourself”* cualquier aporte en esta área actualmente deja un precedente para

futuras investigaciones, ya que es posible que para las siguientes generaciones sea mucho más sencillo crear sus propios dispositivos.

Para realizar este proyecto se realizará un análisis documentado de la metodología por la cual se construye y se programa un controlador MIDI, desde el concepto y boceto del diseño del dispositivo hasta su aplicación real demostrada en una performance capturada en video.

1 Marco teórico

1.1 MIDI

1.1.1 ¿Qué es el MIDI?

Según el libro de Roland (2009) MIDI, que significa "*Musical Instrument Digital Interface*" por sus siglas en inglés, es un sistema que permite enviar información en formato digital entre computadores e instrumentos musicales, este fue presentado en 1983 y sigue vigente en la actualidad, quien desarrolló la versión 1.0 fue Dave Smith, fundador de *Sequential Circuits* y de *Dave Smith Instruments*.

Este protocolo asigna un valor digital a cada elemento del instrumento a través de valores conocidos como MIDI CC (Control Change). Hay 128 valores que van desde MIDI CC0 hasta MIDI CC127, algunos de estos tienen funciones asignadas, por ejemplo, CC64 es el pedal de *sustain* y MIDI CC1 es la rueda de modulación que se puede encontrar en una gran cantidad de sintetizadores. Hay otros MIDI CC que no tienen una función específica, pero esta puede ser asignada por la empresa que manufactura el controlador. (Roland 2009)

Dentro de cada MIDI CC hay valores entre 1 y 127 que determinan la posición del parámetro, de esta manera es posible comunicar, por ejemplo, el *velocity*, que es la fuerza con la que se tocó una nota en el controlador MIDI. Como se puede deducir, la resolución de todos los parámetros del MIDI consta de 127 puntos. En algunos casos el valor varía sólo entre 1 y 127 cuando se trata de un parámetro binario de encendido y apagado, por ejemplo, en el caso del pedal de *sustain*. (Roland 2009).

1.1.2 Controladores MIDI (definiciones, conceptos básicos, ejemplos)

Como menciona Roland (2009) un controlador MIDI es un instrumento musical que controla la señal MIDI en otro dispositivo que usualmente es una computadora, también puede ser un sintetizador u otros instrumentos MIDI.

En la era temprana de los controladores MIDI era concebido que solo los tecladistas tenían acceso a estos debido a que habían pocas opciones de controladores MIDI, además, estos dispositivos habían sido concebidos a partir de sintetizadores, por esta razón muchos músicos que tocaban otros instrumentos estaban siendo excluidos de esta nueva tecnología, sin embargo, con el tiempo se han ido añadiendo todo tipo de controladores para cada tipo de instrumentistas, desde guitarras MIDI, controladores MIDI de Instrumentos de viento, actualmente incluso existe un controlador MIDI que transforma la voz de un cantante y *beatbox* en notas MIDI.

1.1.3 MIDI y DAWs

Los DAWs (*Digital Audio Workstation*) son programas de informática que permiten la edición de audio en formato digital. Existen varios programas de este tipo, cada uno con diferentes propósitos y dirigido hacia diferentes públicos.

La gran mayoría de estos programas adoptaron el formato MIDI en algún punto de su trayectoria, el MIDI se ha convertido en un estándar, y junto a los DAWs han revolucionado la forma de crear música y dado paso a nuevos formatos de estudios musicales que requieren mucho menores presupuestos, así como a la creación de un mercado de “Plugins” digitales, que son simulaciones de componentes análogos utilizados en estudios profesionales, al igual que instrumentos virtuales que se pueden tocar gracias al controlador MIDI.

Esta revolución no solo ha afectado a la rama de producción musical, sino también al performance en vivo con el uso de secuencias, que se pueden hacer en computadores portátiles e incluso en dispositivos móviles de *Apple* y *Android*. El uso de efectos y secuencias en vivo era antes una posibilidad únicamente para los músicos de mayores ingresos y ahora está disponible para un número mucho mayor de intérpretes. (Huber, D. M., & Runstein, R. E. 2017)

1.1.4 Ableton Live

Ableton Live es un DAW que ofrece una alternativa distinta a los demás, ya que está más centrado en explotar la creatividad del artista que en ser fluído para la

edición de audio, que es la tendencia en la mayoría de los otros DAWs. Esto lo logra implementando una serie de herramientas que hacen más viable el desarrollo de ideas musicales, siendo la más importante la implementación de la llamada *session view*, que es una matriz donde se pueden compilar arreglos musicales en cuadros llamados clips, esto hace que el *looping* y el *live performance* sean mucho más fáciles de conseguir.

Al mismo tiempo Ableton ofrece la opción a sus clientes de crear sus propios dispositivos de audio a través de Max, un lenguaje de programación creado por Cycling '74 y pensado hacia la música y multimedia, Max for live ha abierto una infinidad de posibilidades a usuarios que crean y comparten sus dispositivos en comunidades que se han creado para ese fin. (Ableton, 2018)

1.2 Arduino

Arduino es una empresa de implementos de electrónica que se especializa en crear placas de desarrollo de software, con estas es posible diseñar y programar dispositivos digitales para casi cualquier propósito. Las placas Arduino están disponibles de forma comercial en solitario o como parte de un kit DIY (*do it yourself*).

Arduino está constituido principalmente por dos elementos: La placa, donde haces las conexiones físicas; y el IDE (*Integrated Development Enviroment*) que es el ecosistema donde desarrollas toda la programación de tu dispositivo. (Banzi, 2014)

1.2.1 Pro Micro 5V/16MHz

Esta es específicamente la placa que se utilizará en este proyecto, no es una placa oficial de Arduino sino un clon fabricado en china por la empresa Sparkfun con el chip ATMEGA32U4, el mismo que las placas Arduino Leonardo y Arduino Micro. Una de sus características principales es que lleva un puerto USB. (Jimblom, 2013)

Es importante conocer cada input y output que la placa tiene para saber qué tipo de elementos se puede conectar a cada pin.

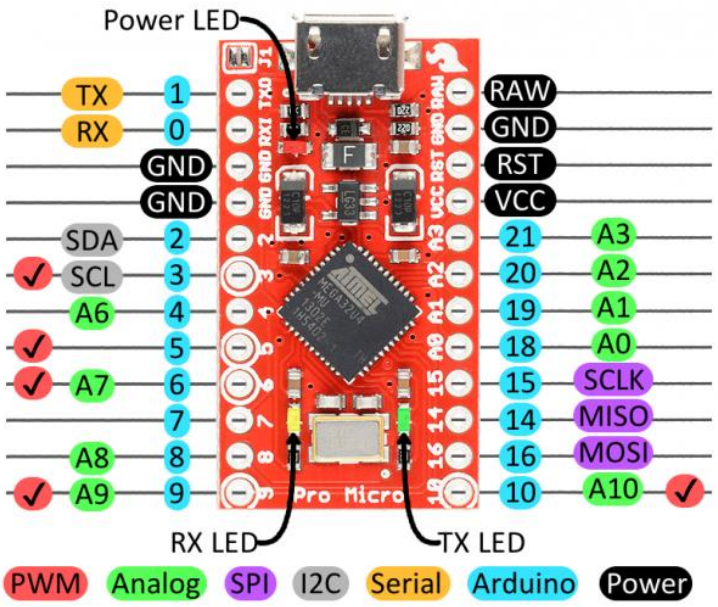


Figura 1. Esquema de placa Pro Micro. Tomada de (Jimblom, 2013)

En la figura 1 se puede apreciar que cada pin tiene diferentes capacidades, por ejemplo, los pines que son catalogados de tipo Arduino pueden ser utilizados como input para elementos que envían información digital y los pines análogos pueden recibir información de elementos analógicos, pero el número de input es distinto si se lo utiliza como input análogo o digital, por ejemplo, el input A3 análogo es el mismo que el input 20 digital, así que se tendrá que detallar en la programación cual de los dos tipos de input se está utilizando. Los pines que tienen capacidad PWM (*Pulse With Modulation*) se pueden utilizar para controlar, por ejemplo, tiras LED; y los del tipo SCL y SDA pueden servir para conectar un *display* LCD. (Jimblom, 2013)

También es posible conectar un dispositivo llamado *Multiplexer* para aumentar el número de entradas de la placa. Cada *Multiplexer* debe ir conectado a un input análogo y a uno digital, de esta forma se obtendrán 16 nuevos inputs por cada *Multiplexer.*

1.2.2 Maxuino

Maxuino es un proyecto de código abierto que permite que Max se comunique con dispositivos Arduino. De esta forma es posible crear un controlador que se comunique con Ableton Live. Maxuino comenzó como un proyecto

independiente pero finalmente se convirtió en un pack de herramientas incluido en Ableton Live Suite, y ahora es conocido como *Connection Kit*. En este pack es posible encontrar herramientas para conectar el DAW con distintos tipos de aparatos, como Arduino, Lego Mindstorms EV3 y dispositivos OSC. (Maxuino, 2014)

1.3 Diseño y programación de un controlador MIDI

Según Daniel Jansson (2019) la creación de un controlador MIDI consta de varias etapas. Lo primero será crear un boceto del producto final que se quiere construir, para esto hay que tener en cuenta cual será el propósito del aparato.

Es posible basarse en distintos parámetros en el diseño de productos, por ejemplo, la ergonomía, que optimiza la relación entre el producto y la parte del cuerpo que lo va a utilizar. También es posible guiarse por corrientes de diseño, por ejemplo, el diseño orgánico, que trata de que la estética del producto se vea más viva con bordes redondeados y curvas. (Solanas, 1981)

Comúnmente en los procesadores de audio y sintetizadores se ha visto una línea de diseño rígido, clásico y donde prima la necesidad de incluir muchos elementos en un espacio reducido dándole prioridad a la función sobre la forma.

Una vez claro el propósito habrá que conseguir los elementos que serán parte de nuestro dispositivo, por ejemplo, botones, faders, potenciómetros, pantallas LCD, etc. Cada dispositivo solo debe ser capaz de enviar información al IDE de Arduino, donde se programarán las condiciones para que la información pueda ser utilizada para controlar parámetros MIDI. (Jansson, 2019)

1.4 Daniel Jansson

Daniel Jansson es un youtuber, diseñador industrial y de interacción nacido en Suecia, en su canal Switch & Lever el crea todo tipo de aparatos y expone su trabajo a través de un video tutorial, algunos de estos videos incluyen tutoriales de cómo hacer una moneda con tu cara, cómo copiar casi cualquier objeto, 100 maneras de sacarle punta a un lápiz, etc.

En uno de sus videos titulado “Building a MIDI Controller Using Arduino” Daniel Jansson muestra como crear desde cero un controlador MIDI utilizando componentes reciclados, una placa Arduino MEGA, una carcasa de madera cortada por laser, y softwares variados para la programación. Finalmente muestra su aplicación en Ableton Live junto con los usos y aplicaciones que decidió darle. (Jansson, 2019)

Este ejemplo no solo es un referente para la creación de controladores MIDI inspirados en el dispositivo que diseñó Daniel específicamente, sino para la creación de cualquier tipo de controlador MIDI DIY, por lo que el reto es encontrar la manera de aplicar los procedimientos del video a un diseño propio.

2 Metodología

2.1 Objetivos

2.1.1 Objetivo General

Crear un controlador MIDI *"Do it Yourself"* basado en uno realizado por Daniel Jansson y producción de una performance captada en video haciendo uso del dispositivo.

2.1.2 Objetivos Específicos

1) Establecer una referencia teórica sobre la creación y programación de un controlador MIDI a través de la recopilación documental.

2) Analizar el diseño y propósito del controlador MIDI creado por Daniel Jansson en su canal *Switch And Lever*

3) Realizar el diseño y la programación de un controlador MIDI y la producción de una performance captada en video que demuestre sus aplicaciones.

2.2 Enfoque

Esta investigación estará basada en los enfoques documental y cuantitativo, ya que se registrará el proceso de construcción y programación de un controlador MIDI.

A través de la recopilación documental se detallarán los procesos por los cuales fue posible llegar al diseño, el ensamblaje y posterior programación de los parámetros. El enfoque cuantitativo estará presente en el detalle de la programación, donde se especificarán los valores asignados a distintos parámetros del controlador con el uso de matrices.

2.3 Metodología

Los tipos de métodos utilizados en este trabajo serán experimentales (prueba y error) y tecnológicos (simulación). El método experimental servirá para que el dispositivo se pueda terminar de construir sin errores de programación o en el ensamblaje, ya que primero se ensayará en una *protoboard* para verificar que las conexiones y la programación funcionen y sea posible utilizar todo dentro del DAW.

Dentro de la programación también se utilizarán herramientas tecnológicas como el modo *debug* que sirve para probar el código antes de ser enviado al procesador de la placa Arduino, esto para evitar que se envíe un código con fallos que la pueda dañar. Finalmente se utilizará Ableton Live, donde se realizará un mapeo de diferentes parámetros a los elementos del controlador.

2.4 Estrategias metodológicas

Las estrategias metodológicas seleccionadas para este proyecto han sido el análisis y el cuaderno de campo debido a que se trata de una investigación dirigida hacia el proceso de creación de un dispositivo. Se analizará un ejemplo de este tipo de proyecto y luego se describirá a través de la recopilación documental y el cuaderno de campo la construcción del dispositivo.

2.5 Plan de Trabajo

En el proyecto se realizará un controlador MIDI DIY personalizado utilizando como referencia un video del canal Switch & Lever de Daniel Jansson.

Para abordar este proyecto se analizará el video y el controlador en sí para obtener información del proceso de diseño, construcción y programación del dispositivo.

Con ayuda del cuaderno de campo se realizarán varios diseños para el dispositivo pensando en su propósito, así mismo se guardarán algunas notas para que más adelante la programación del dispositivo resulte sencilla.

Después se conseguirán los diferentes dispositivos que son necesarios para la construcción, desde la placa Arduino, los potenciómetros, botones, etc. Con las medidas de estos elementos se podrá diseñar una carcasa que será impresa en 3D.

La programación estará basada en una plantilla para controlador MIDI en IDE de Arduino creada por Gustavo Silveira para su curso online “Making Music With Arduino”. En esta plantilla se pueden encontrar organizados varios párrafos de programación que corresponden a cada tipo de elemento, por ejemplo, botones o potenciómetros; Estos párrafos se activan o no dependiendo de si ese tipo de dispositivo está presente en el controlador, además, es necesario especificar otros parámetros como el MIDI CC y los valores entre los que oscilará la señal que envía cada uno de estos elementos.

Finalmente el dispositivo se conectará a Ableton Live donde se producirá una sesión, con ella se realizará una performance que será capturada en video y se hará uso del controlador.

2.6 Análisis del dispositivo construido por Daniel Jansson

2.6.1 Diseño y Propósito

El video de Daniel Jansson (2019) comienza explicando que la razón por la que construye un controlador MIDI es debido a que su actual controlador (un Korg Microkey Air) no tiene suficientes controles para modificar varios parámetros dentro de su DAW, así que el propósito de su dispositivo es complementar a su otro controlador.

En la siguiente parte del video da una ligera explicación de Arduino, donde explica que al crear un dispositivo con esta herramienta, cualquier elemento que sea capaz de enviar información a la placa puede programarse para hacerlo en formato MIDI, así que incluso es posible reciclar elementos

electrónicos de otros aparatos para utilizarlos en su dispositivo, Daniel decide que su controlador tenga los siguientes elementos:

Tabla 1. Componentes del Controlador de Daniel Jansson.

Componente	Cantidad	Input
Ericsson Diavox Keyboard	1	Digital
Switch	3	Digital
Switch de 6 posiciones	1	Digital
Perilla potenciómetro 10kΩ	3	Análogo
Slide potenciómetro 10kΩ	1	Análogo
Joystick (*Modulation Wheel*)	1	Análogo
Rotary Encoder	2	Digital

Las posibilidades son ilimitadas en cuanto a la forma del dispositivo, por ejemplo, se pueden utilizar juguetes o incluso cajas de cartón como carcasa, sin embargo, él se decide por imprimir su propia carcasa en madera. (Jansson, 2019)

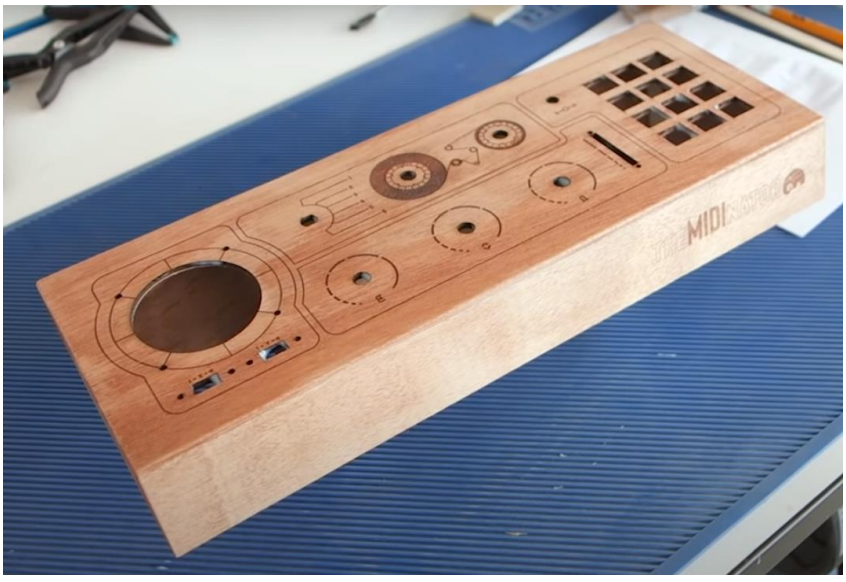


Figura 2. Carcasa impresa en madera para el dispositivo de Daniel Jansson. Tomado de (Jansson, 2019)

Para realizar la carcasa del dispositivo Jansson tuvo en cuenta el perímetro de cada uno de los elementos para que pudieran entrar y ajustarse. En cuanto a su diseño, este es parecido al de los procesadores de audio tradicionales, sin embargo, Jansson le agregó su propio “toque” imprimiendo varios gráficos en la madera, que en este caso sirven en su mayoría solo como decoración, el único gráfico con utilidad es el del *switch* con 6 posiciones, donde el gráfico muestra la posición actual del *switch*.

2.6.2 Elección de la tarjeta Arduino

En el video Jansson comienza a hacer el dispositivo con una tarjeta Arduino UNO, que cuenta con 14 *inputs* digitales y 6 *inputs* análogos, sin embargo, se da cuenta de que esta tarjeta no es suficiente para suplir la cantidad de componentes que él quiere incluir, por lo que decide que utilizará una tarjeta Arduino Mega, que tiene 54 *inputs* digitales y 16 *inputs* análogos. Este problema lo pudo haber solucionado adquiriendo uno o varios *multiplexers,* y conectándolos a la tarjeta más pequeña, ya que por el tamaño de su proyecto no es necesario tener que recurrir a una tarjeta más grande, pero es probable que se tomara esta decisión debido a que Jansson buscaba mayor comodidad al programar los componentes del dispositivo, ya que los *multiplexers* requieren una programación adicional. (Jansson, 2019)

2.6.3 Ensamble y Conexiones

Es inusual el proceso de construcción del dispositivo de Jansson ya que no realizó un prototipo del proyecto entero, sino que hizo prototipos con cada uno de los componentes por separado para comprobar si funcionarían para enviar señal MIDI con la programación en IDE de Arduino.

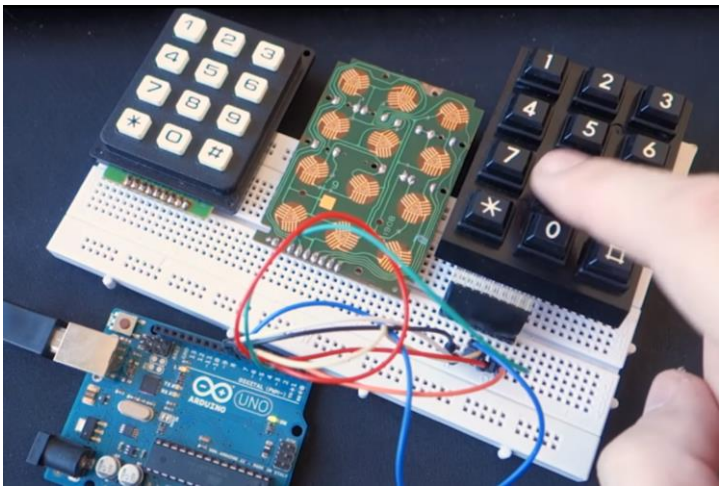


Figura 3. Módulo prototipo para el teclado de teléfono. Tomado de (Jansson, 2019)

Normalmente la creación de la carcasa sería uno de los últimos pasos en la construcción de este tipo de dispositivos, pero en el caso del dispositivo de Jansson fue lo primero que hizo. Además, ensambla todos los componentes individuales sin haber hecho antes las conexiones y finalizado la programación del dispositivo en conjunto. En el video se menciona que esto lo deja un poco “ajustado”, este método es menos seguro, ya que se podrían causar confusiones en las conexiones y luego en la programación, pero se menciona que una ventaja es que resulta más fácil colocar componentes individuales cuando aún no hay nada conectado. Esto también es una muestra de la experiencia de Jansson construyendo dispositivos con Arduino. (Jansson, 2019)

Figura 4. Jansson ajustando componentes en la carcasa. Tomado de (Jansson, 2019)

En la figura 3 se puede observar como Jansson ajusta el *switch* rotatorio de 6 posiciones con una tuerca en el agujero que había impreso para este accesorio, este procedimiento lo realiza con varios de los elementos.

Figura 5. Lado interno de la carcaza del dispositivo de Jansson. Tomado de (Jansson, 2019)

En la figura 5 se puede notar que Jansson utiliza pedazos de madera con tornillos para sostener el teclado numérico de teléfono y dos de los potenciómetros. También ha pegado dos tiras de cinta de cobre a las cuales ha nombrado GND y 5V, mediante estas dos cintas se hará la distribución de la corriente a todos los elementos, GND es la conexión a tierra y 5V es la conexión a los cinco voltios que son proporcionados por la tarjeta. Todos los componentes análogos deberán ir conectados a tierra, a los cinco voltios y a un input análogo de la tarjeta, mientras que los componentes digitales deberán ir conectados solo a tierra y a un input digital de la tarjeta. En la descripción del video Jansson agregó un esquema en el que se puede ver detalladamente todas las conexiones que realizó.

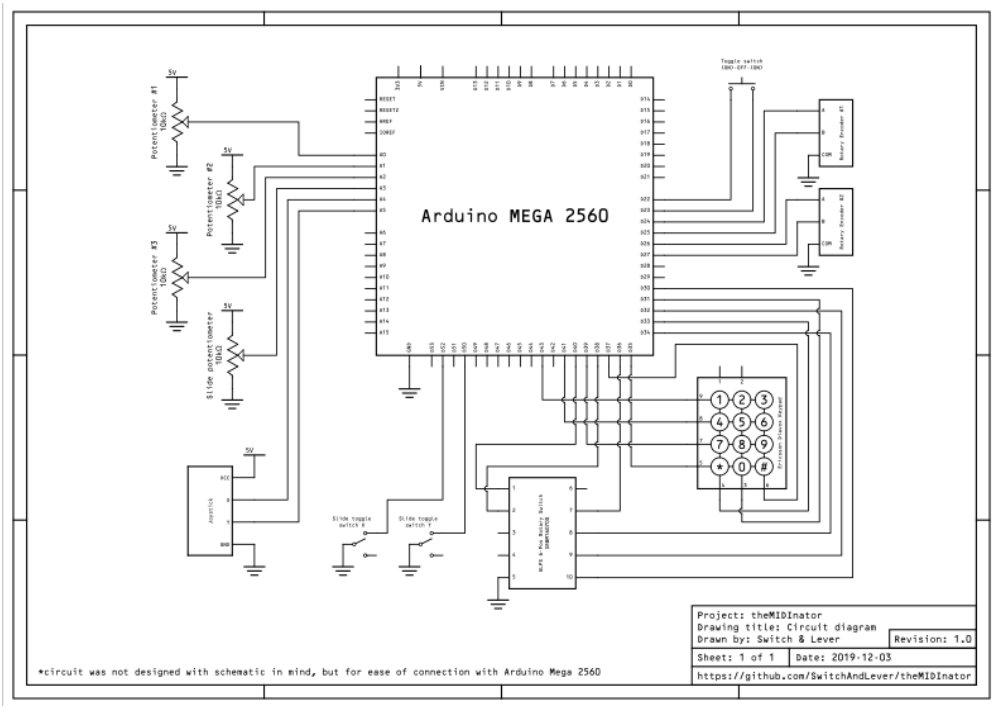


Figura 6. Esquema del dispositivo de Jansson. Tomado de (Jansson, 2019)

Generalmente se asigna el color de cable negro a todas las conexiones a tierra y rojo a las conexiones a los cinco voltios, a continuación, en la figura 7 se puede apreciar que Jansson aplica este formato. Para el resto de las conexiones se puede apreciar que Jansson utiliza cables de varios colores, y para conectar el teclado telefónico utiliza una cinta de cables arcoíris. (Jansson, 2019)

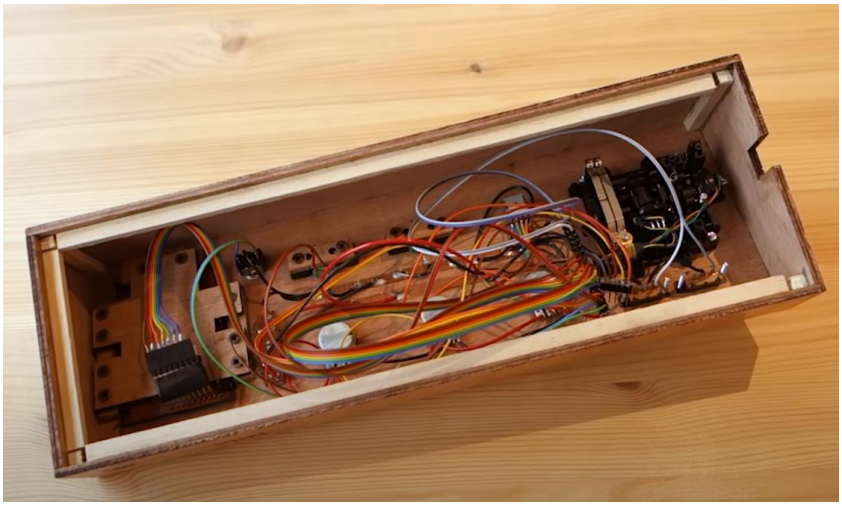


Figura 7. Conexiones internas. Tomado de (Jansson, 2019)

Como últimos pasos en el ensamble del dispositivo Jansson conecta todo a la placa Arduino Mega y sella la carcasa, finalmente coloca tapas a los potenciómetros para que tengan mejor agarre y que se vean mejor estéticamente.

Figura 8. Dispositivo terminado. Tomado de (Jansson, 2019)

2.6.4 Programación

Jansson no entra en los detalles de la programación del dispositivo ya que probablemente el video se habría vuelto demasiado extenso, sin embargo, en la descripción añade un enlace a una página en Github, que es un foro en línea donde es posible compartir códigos de , en ella se puede encontrar toda la programación del dispositivo, además, se agregaron algunas explicaciones al lado de algunas líneas del código para que sea más fácil entender su propósito, Jansson hizo esto ya que esperaba que otras personas fueran capaces de replicar fácilmente su trabajo.

El código de Jansson está dividido en secciones de líneas, cada una de estas corresponde a uno de los componentes utilizados para la construcción de su dispositivo.

El primer código es del teclado numérico de teléfono, lo programa para tocar notas musicales en MIDI. En esta misma agrupación de líneas de código añade la programación de un *switch* que utiliza para transponer estas notas.

```
// KEYPAD //

#include <Keypad.h>

const byte ROWS = 4; // four rows
const byte COLS = 3; // three columns
char keys[ROWS][COLS] = {  // keypad keys, 1-9, 0, S for star (asterisk) and P for pound (square)
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'S', '0', 'P'}
};

byte rowPins[ROWS] = {43, 41, 39, 35}; // keypad row pinouts
byte colPins[COLS] = {33, 31, 37}; // keypad column pinouts

Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

int midC = 60; // MIDI note value for middle C on a standard keyboard

const int transposePin1 = 22; // pins for the switch controlling transposing
const int transposePin2 = 23;
int transpose = 0;  // if = 0 no transposing
int transposeLeft = 0;
int transposeRight = 0;
int oldTransposeLeft = 0;
int oldTransposeRight = 0;
unsigned long transposeTimer = 0;  // for debouncing
```

Figura 9. Programación del teclado númerico de teléfono. Tomado de (Jansson, 2019)

Sin adentrarse mucho en términos de programación, es posible notar que Jansson utiliza funciones para declarar las variables y constantes que el programa utilizará para leer las señales de su componente, luego describe en inglés el rol de algunas de las líneas para que la gente que no sabe programar pueda entender un poco lo que hizo, esto lo hace después de colocar un doble slash, esto hace que la frase solo cuente como comentario sin afectar el resto del código.

En el teclado numérico lo que Jansson hizo fue declarar los pines a los que está conectado cada botón en el Arduino, declarando al inicio que hay cuatro filas y tres columnas de números. Luego declaró que la primera nota sería la nota 60 (el do central de la mayoría de teclados). Luego agregó los pines del switch que controla la transposición de notas MIDI.

```
68
69  // ROTARY ENCODER //
70
71  #define ENCODER_DO_NOT_USE_INTERRUPTS
72  #include <Encoder.h>
73
74  Encoder myEnc1(26, 27);
75  Encoder myEnc2(24, 25);
76  long position1  = -999;
77  long position2  = -999;
78  int encVals[12] = {64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64}; // set initial value of encoder to mid range of 0-127
79
```

Figura 10. Programación de los encoders de rotación. Tomado de (Jansson, 2019)

En la figura 10 se puede ver la programación de los encoders de rotación, Jansson programó estos con ayuda de una librería gratuita para ayudar a programar este tipo de dispositivos llamada “encoder”, se puede ver que en la segunda línea del código añade una función para que se incluya. En las siguientes dos líneas Jansson declara los 4 pines de Arduino Mega donde están conectados sus encoders. (Jansson, 2019)

Los encoders de rotación requieren de dos pines digitales, uno lee los valores que da la rotación en el sentido de las manecillas del reloj y el otro lee la rotación al lado contrario. Al menos uno de los pines en la tarjeta debe tener capacidad PCINT, que hace que la lectura de la tarjeta sea más rápida al recoger los valores, si ninguno de los pins tiene esta capacidad algunos de los valores que envíe el componente no serán leídos y la lectura será algo defectuosa.

```
80
81  // ROTARY SWITCH //
82
83  const int rotSwitch1 = 30;  // rotary switch pins
84  const int rotSwitch2 = 32;
85  const int rotSwitch3 = 34;
86  const int rotSwitch4 = 36;
87  const int rotSwitch5 = 38;
88  const int rotSwitch6 = 40;
89  int cVal = 1;
90
```

Figura 11. Programación del *switch* de rotación de 6 posiciones. Tomado de (Jansson, 2019)

En la programación de este *switch* se declaran los pines a los que están conectados cada una de las 6 posiciones de rotación.

```
// POTENTIOMETERS //

const int pot1 = A0;  // potentiometer pins
const int pot2 = A1;
const int pot3 = A2;
const int slidePot = A3;

int potVal1 = 0;
int potVal2 = 0;
int potVal3 = 0;
int slidePotVal = 0;

int lastPotVal1 = 0;
int lastPotVal2 = 0;
int lastPotVal3 = 0;
int lastSlidePotVal = 0;
```

Figura 12. Programación de los potenciómetros. Tomado de (Jansson, 2019)

Como se puede ver en la figura 12, los potenciómetros requieren de un pin análogo de la tarjeta, en las primeras 4 líneas se definen los pines análogos (A0, A1, A2, A3) a los que están conectados los cuatro potenciómetros. Luego se describen las variables para los valores iniciales y finales de cada potenciómetro, en este caso todas son 0.

```

// JOYSTICK //

const int joyX = A5;  // joystick pins
const int joyY = A4;

const int Xswitch = 52;  // axis switche pins
const int Yswitch = 50;

int joyXval = 0;
int joyYval = 0;
int lastJoyXval = 0;
int lastJoyYval = 0;
```

Figura 13. Programación de los potenciómetros. Tomado de (Jansson, 2019)

En la figura 13 se puede encontrar la programación del último componente del dispositivo de Jansson, que es un *joystick*. Este envía información como si se tratara de dos potenciómetros diferentes, conectados a dos pines análogos de la tarjeta, uno enviará los valores del eje X y el otro del eje Y. Además, Jansson incluye unos switches cuya función no explica en el video, en la programación solo declara los pines a los que irán conectados unos switches que están relacionados a estos ejes de alguna forma que no se explica. (Jansson, 2019)

2.7 Creación del controlador MIDI y producción de la performance.

2.7.1 Diseño del dispositivo.

Para el diseño del dispositivo se realizaron varios *sketches* en el cuaderno de campo, donde se pueden encontrar dos diseños posibles para el dispositivo. Uno de estos diseños es una alternativa en caso de que la duración de este proyecto no sea suficiente para ensamblar la alternativa principal.

Figura 14. Primer render realizado en Blender del concepto de diseño para la alternativa principal.

Como se puede ver en la figura 14, para la alternativa principal se pensó en realizar un dispositivo que tuviera una forma similar a la de un mando de

videojuegos, esto con el propósito de que fuera ergonómico para ambas manos.

En el medio del controlador están ubicados tres potenciómetros de tipo *slider*, también conocidos como *faders*, bajo estos van dos *encoders* de rotación y, a los lados izquierdo y derecho de estos respectivamente, dos potenciómetros de rotación. La función de estos siete componentes sería cualquiera que el usuario quiera asignarle en su DAW.

En el área a la derecha e izquierda de los *faders* respectivamente, se encuentran cuatro botones orientados de forma cuadrangular y tres botones orientados de forma triangular. A los extremos derecho e izquierdo de la cara superior del controlador (en la figura 14) estarían ubicados otros dos botones, finalmente, los *encoders* de rotación también cuentan con una función adicional como botones que se activa al presionarlos. Todos estos botones actúan como notas MIDI, y abarcan desde la nota 36 (el estándar de Do central o C4) hasta la nota 46, esto debido a que en total son 11 botones, y las notas se asignan por semitonos.

Sería posible asignar las notas a una escala por medio de la programación, por ejemplo, a la escala pentatónica, de esta forma se podría tocar melodías con los botones, sin embargo, este dispositivo está pensado para tocar *samples* en el instrumento virtual de Ableton Live “drum rack”, donde el estándar es que los samples estén asignados por semitonos desde C4.

Encima de los tres *faders* están ubicados dos botones, su función sería la de intercambiar el canal MIDI, de esta manera se multiplican las posibilidades de asignación de parámetros, el dispositivo podría intercambiar entre 16 canales. Los botones de cada canal pueden ser asignados a un instrumento distinto, de igual manera cada potenciómetro y slider podrían controlar un diferente parámetro en cada uno de estos canales.

Figura 15. Segundo render realizado en Blender del concepto de diseño para la alternativa principal.

En el segundo render del diseño se modificaron varios aspectos del dispositivo, pero principalmente se redujo su tamaño ya que accidentalmente el primer diseño había quedado demasiado grande. Se midió cada componente para que entrara de forma perfecta en la carcasa, y se hicieron varias modificaciones en cuanto a la orientación de los componentes:

- Se añadió una pantalla OLED de 124x64 pixeles ubicada sobre los *faders*. Su función es dar retroalimentación sobre el canal MIDI en el que se encuentra el dispositivo. Esto para que el usuario no se pierda entre los 16 canales.

- Los botones para intercambiar el canal MIDI se movieron a debajo de los botones en orientación triangular debido al reducido espacio en la parte superior a los *faders.*

- Los potenciómetros y *encoders* ya no irían debajo de los *faders* sino a los lados de su parte inferior.

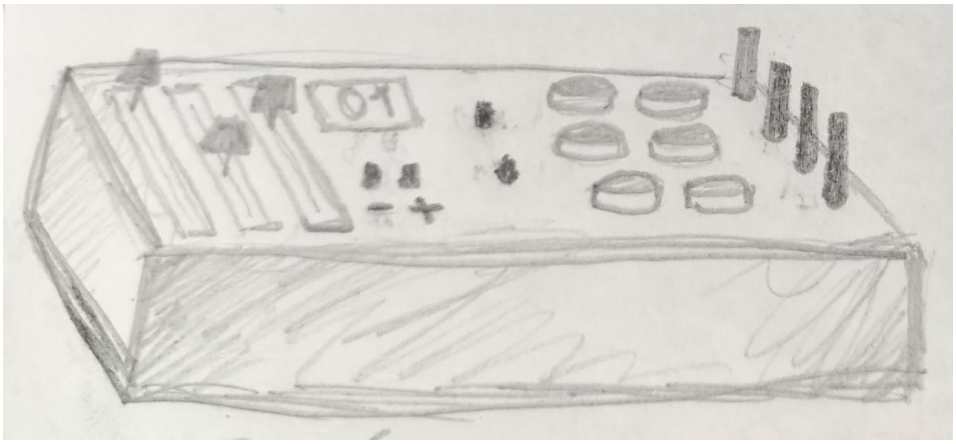


Figura 16. Boceto en el cuaderno de campo del diseño alternativo del dispositivo.

Se planificó también un diseño alternativo para el dispositivo, que es más sencillo y puede ser construído en menor tiempo, este similar en algunos aspectos al de Jansson, por estar ensamblado en una caja de madera, sin embargo, incorpora componentes distintos.

En este diseño todos los botones se ubican juntos al lado derecho de la cara frontal, a su derecha se ubican los dos potenciómetros y los dos *encoders* de rotación. Al extremo izquierdo se encuentran los tres *faders* y a su lado derecho la pantalla OLED y los botones para elegir el canal MIDI.

2.7.2 Prototipo de las conexiones.

Debido a que ambos diseños se pueden hacer con los mismos componentes, se realizó el prototipo de las conexiones a los pines con ayuda de dos *protoboards* y cables dupont, de esta manera se buscó realizar un modelo funcional del dispositivo que pudiera ser programado antes de estar ensamblado. Para ello se hicieron varios modelos en Fritzing, un programa de código abierto en el que es posible encontrar la mayoría de componentes utilizados en este proyecto y realizar un modelo virtual con ellos. Los componentes que no se pudieron encontrar en el programa fueron descargados luego desde Github, donde una gran comunidad de entusiastas sube sus propios modelos de componentes que no encontraron en la aplicación.

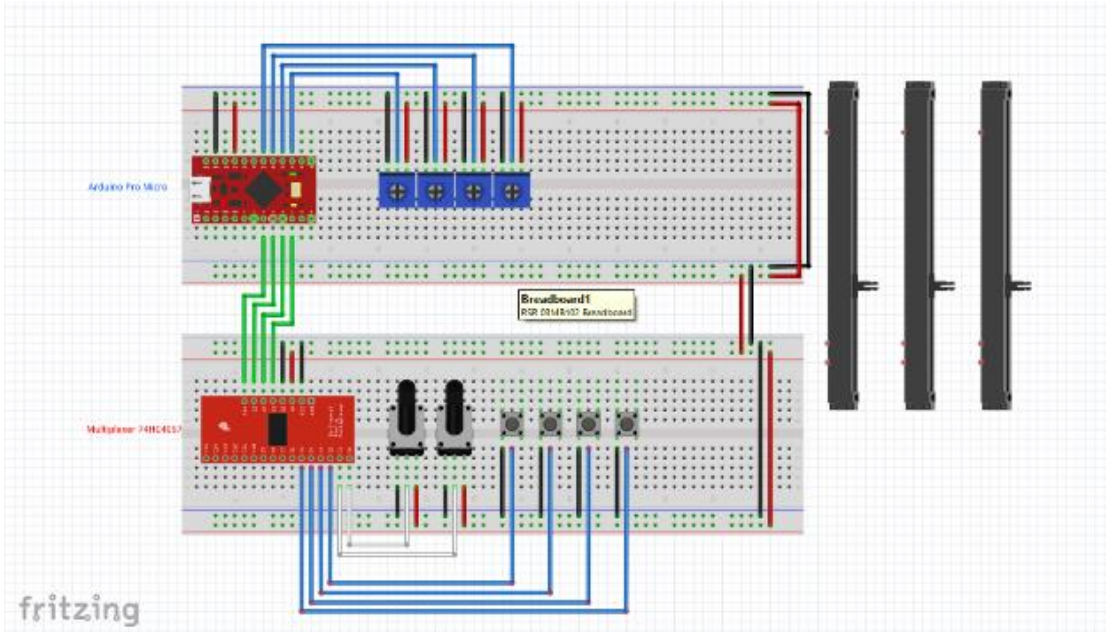


Figura 17. Primer *sketch* de las conecciones realizado en Fritzing.

En la figura 17 se puede ver en esencia los dos componentes principales del dispositivo y sus conexiones; la tarjeta Pro Micro y un *multiplexer* de 16 canales para incrementar la cantidad de pines analógicos de la tarjeta.

En esta etapa no habían sido añadidos todos los componentes, y algunas de las conexiones estaban mal hechas: la de los cuatro botones, ya que habían sido conectados de una manera que no habrían tenido contacto con los cables; los tres *faders* no habían sido conectados; todos los pines que conectaban el *multiplexer* con la tarjeta Pro Micro estaban desplazados un pin, por lo que no habrían funcionado correctamente.

En cuanto al prototipo físico. En las primeras instancias, cuando no se tenía claro cómo se iba a realizar el dispositivo, se habían adquirido varios componentes que no serían utilizados finalmente y que solo sirvieron para realizar las primeras pruebas: tres potenciómetros de resistencia B100K, una tarjeta Arduino Leonardo, cuatro botones y cables.

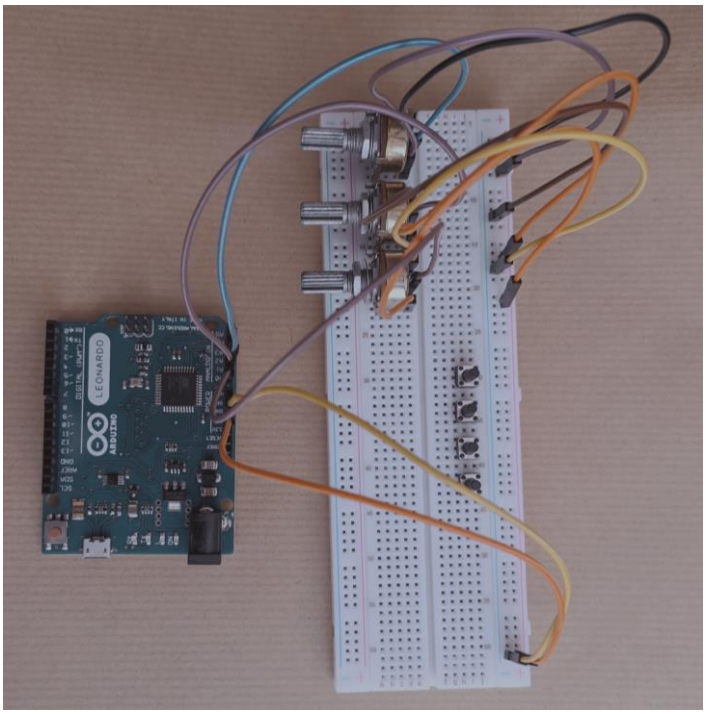


Figura 18. Primer prototipo de las conexiones.

Más adelante la tarjeta Arduino Leonardo fue reemplazada por la Pro Micro debido a que el tamaño de la Pro Micro es bastante más reducido, lo que es favorable para un ensamblaje pequeño como el que se tiene en mente. Por su parte los cables no seguían ningún estándar de colores y no eran suficientes para poder conectar todos los componentes, así que se adquirieron 2 nuevos paquetes de cables. Los potenciómetros B100K fueron reemplazados por potenciómetros B10K que eran más amigables para la programación del proyecto. Finalmente los botones fueron reemplazados por unos más grandes y fáciles de oprimir.

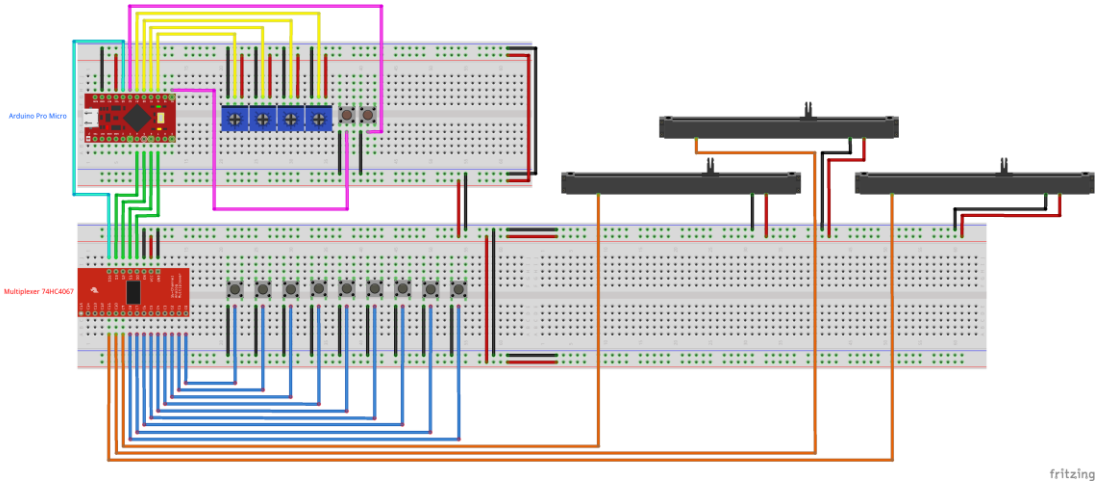


Figura 19. Segundo *sketch* de las conecciones realizado en Fritzing.

En el *sketch* de la figura 18 se pueden notar varios cambios, por ejemplo, se incrementa el número de botones, se añaden los dos botones para cambiar de banco midi, se conectan los tres faders de forma correcta.

Aún así tenía algunos errores, por ejemplo, los nueve botones de la parte inferior siguen sin estar conectados, esto es porque al momento de realizar este *sketch* no se tenía en cuenta que la linea horizontal que divide por la mitad las dos zonas de pines en la protoboard también divide sus conexiones, por lo que entre la zona de pines superior e inferior no existe contacto, así que si los botones están conectados sobre esta línea y sus cables por debajo de ella, no están conectados.

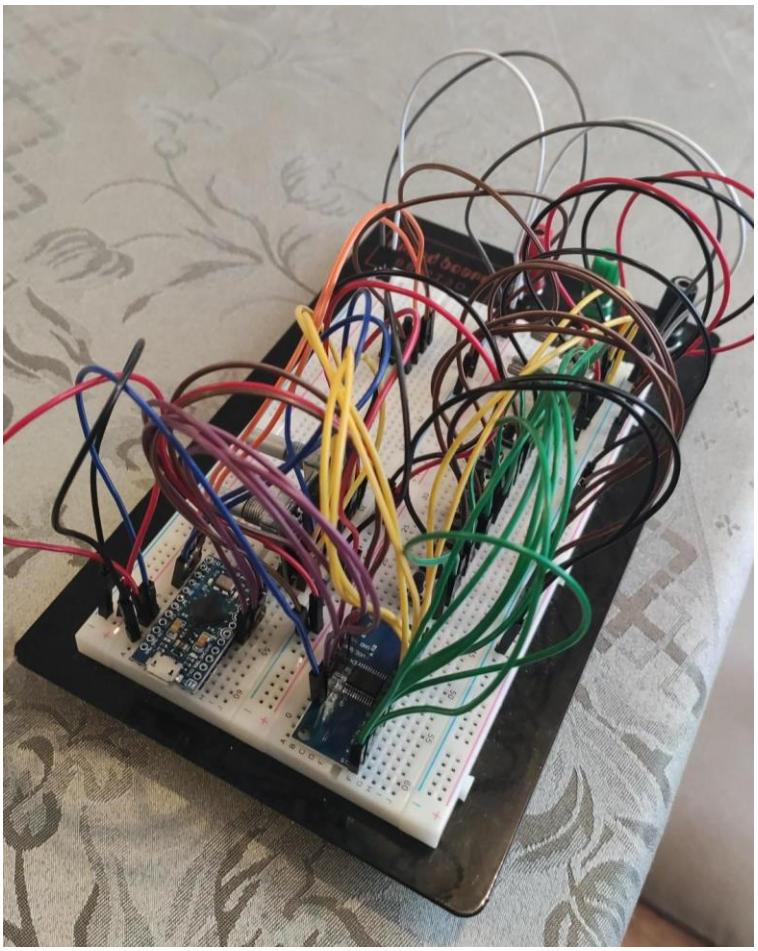


Figura 20. Segundo prototipo físico de las conexiones.

Este prototipo aún no era funcional ya que hacía falta soldar la placa Pro Micro y el *multiplexer* a unos pines para poder conectarlos a la protoboard, además, los botones habían sido mal conectados ya que su posición había sido copiada de la posición que tenían en la figura 18, sin embargo, ya se habían hecho algunas conexiones de forma correcta, como la de los encoders y la de los potenciómetros, los cables estaban ordenados por colores de forma que todos los componentes similares tuvieran el mismo color de cable, después, los cables que fueran a cinco voltios serían rojos o blancos, y los cables que fueran a tierra serían negros, cafés y grises, esto debido a que no habían suficientes cables rojos y negros para todas las conexiones.

Figura 21. Soldando la placa Pro Micro a los pines.

Una vez soldados la placa y el *multiplexer* se procedió a ensamblar de forma correcta todos los componentes en el protoboard.

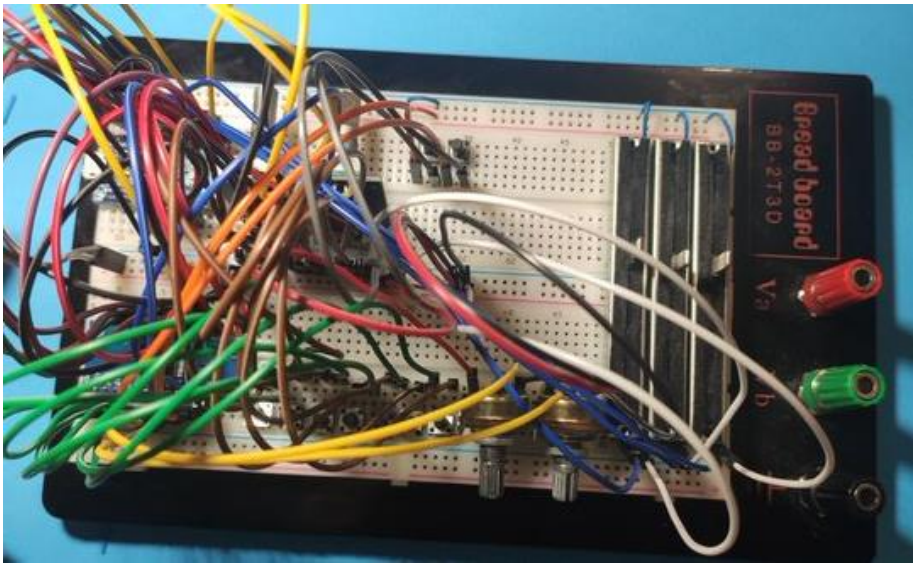


Figura 22. Tercer prototipo físico de las conexiones y primer prototipo funcional.

Al obtener el primer prototipo funcional se comenzó a realizar la programación del dispositivo, lo que aportó con retroalimentación sobre los siguientes pasos a cumplir para obtener un mejor dispositivo.

Principalmente se cambió la ubicación de conexión de los dos botones para elegir el canal MIDI, ya que la plantilla del código que se utilizó no permitía que estos botones estuvieran conectados al multiplexer, debían ir conectados directamente a la tarjeta, por lo que fueron trasladados a los pines 8 y 9 de la Pro Micro. Luego, al hacer las primeras pruebas con el intercambio de canales MIDI se decidió añadir la pantalla OLED para poder visualizar el canal MIDI.

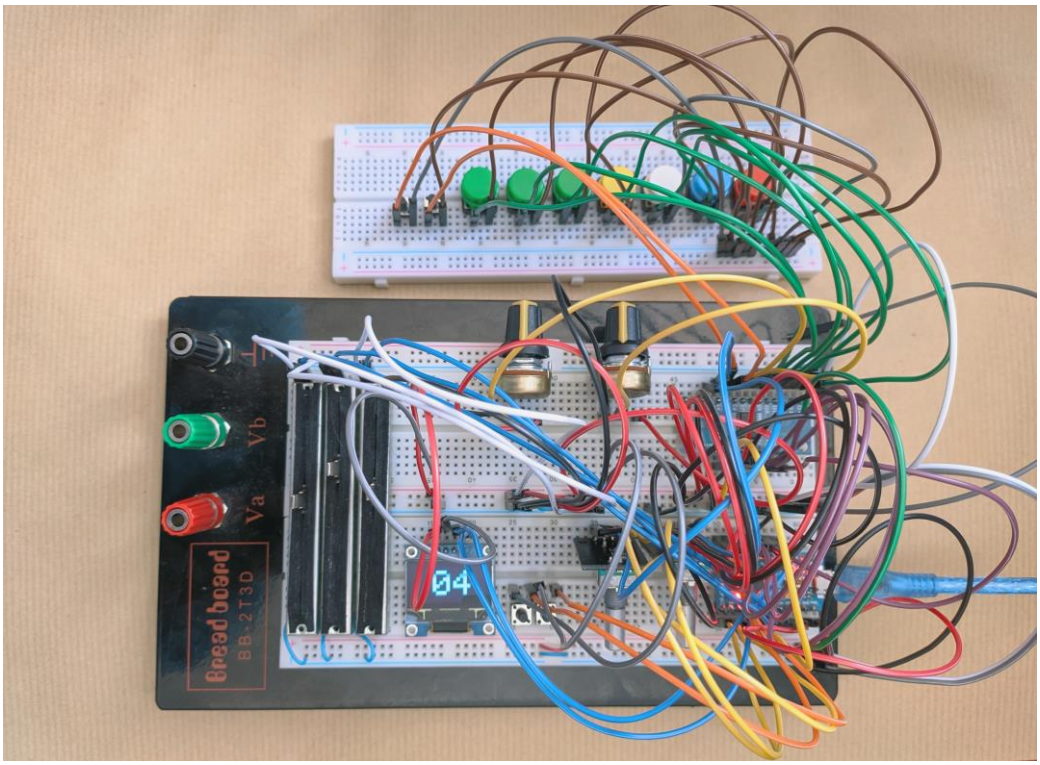


Figura 23. Prototipo final de las conexiones.

El dispositivo finalmente quedó ensamblado de forma que todos sus componentes funcionaran de forma óptima y listo para terminar la programación. En la figura 23 se puede observar el prototipo final con los componentes que se pueden observar en la siguiente tabla:

Tabla 2. Componentes e inputs del dispositivo.

Componente	Cantidad	Inputs
Perilla potenciómetro 10kΩ	2	*Multiplexer* (C7, C8)
Fader 10kΩ	3	Pro Micro (A0, A1, A2)
Botones	13	*Multiplexer* (C3, C4, C5, C6, C9, C10, C11, C12, C13, C14, C15) Pro Micro (8, 9)
Rotary Encoder	2	Pro Micro (15, 14, 16, 10)
Pantalla OLED	1	Pro Micro (2, 3)

Estas conexiones también se pueden observar en la vista esquemática del dispositivo que se realizó en Fritzing:

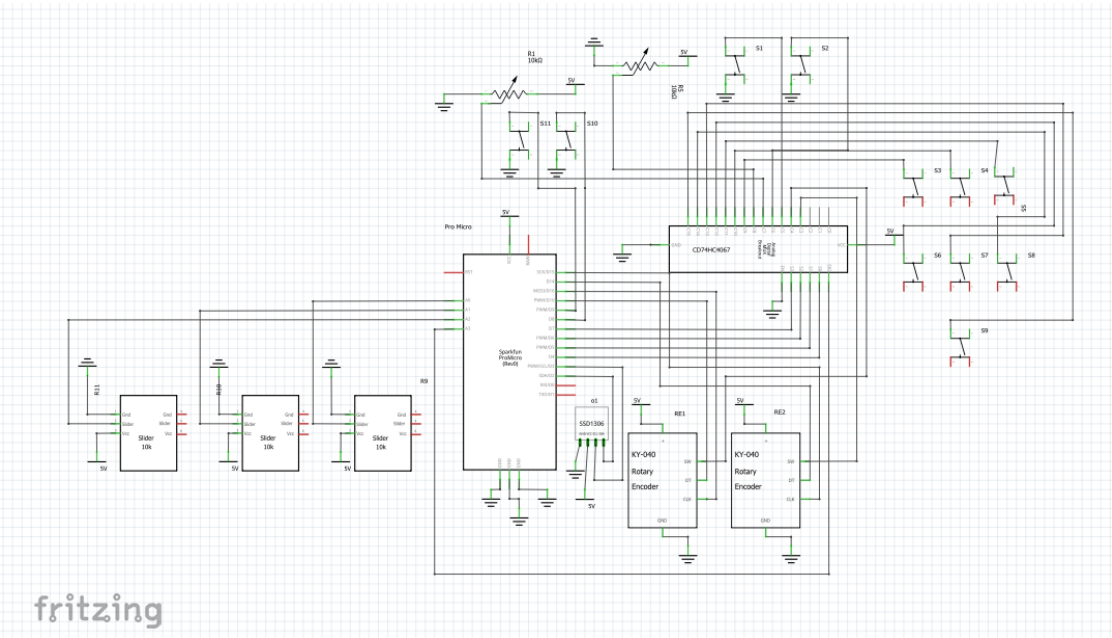


Figura 24. Vista esquemática del dispositivo en Fritzing.

Finalmente se intentó crear la tarjeta PCB que utilizaría el dispositivo, esta se realizó para el diseño alterno del dispositivo, sin embargo, algunos errores no permiten que los caminos de las conexiones se realicen automáticamente.

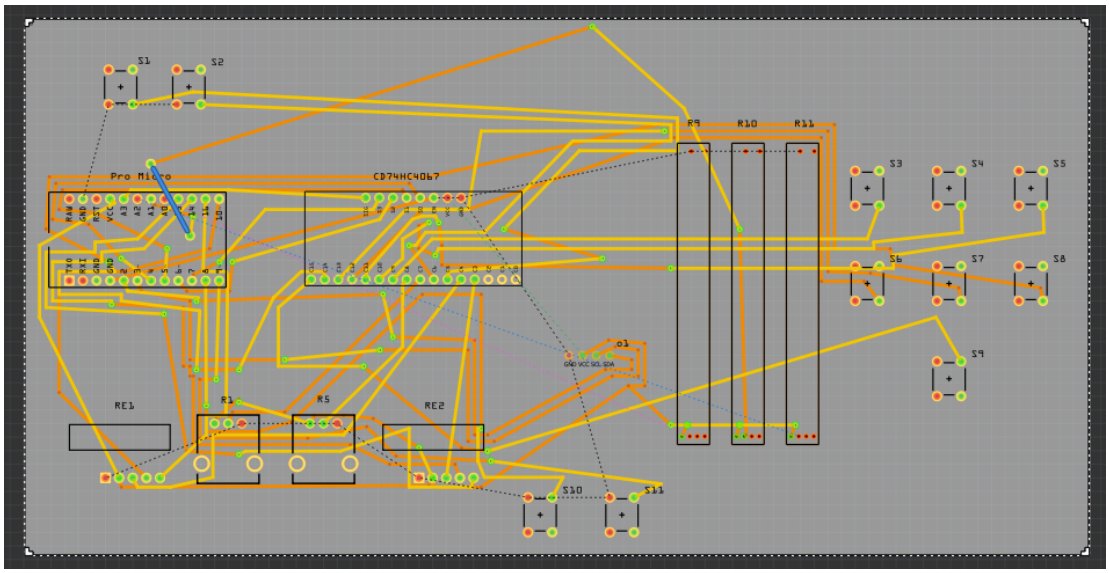


Figura 25. Intento de autoruteo para la placa PCB

Una vez finalizada la placa PCB se imprime, y los componentes se añaden a ella.

2.7.3 Programación del controlador MIDI.

Para realizar la programación se utilizó una plantilla realizada por Gustavo Silveira para su curso “Making Music With Arduino”, y fue creada para que personas sin mucho conocimiento en programación pudieran programar sus propios dispositivos.

Al igual que el código de Jansson, el código de Silveira está dividido en párrafos dedicados a componentes individuales. El código se abre en el IDE de Arduino y contiene varias pestañas, cada pestaña corresponde a un componente individual, Gustavo utiliza estas pestañas para proteger la parte del código de cada componente que no necesita ser modificada, a diferencia de la pestaña principal, que es una hoja extensa donde el programador debe ir cambiando las variables de los párrafos de cada componente bajo su propia tutela para que el código funcione.

Al inicio del código se debe declarar qué tipo de procesador tiene la tarjeta que se está utilizando, en el caso de la Pro Micro se trata de un ATMEGA32U4, al definir este procesador en esta línea se activa solo la sección de la

programación dedicada a este procesador y se desactivan las demás opciones. Esta plantilla solo admite cuatro opciones: ATMEGA32U4, que también está presente en la tarjeta Arduino Leonardo y en la Arduino Micro; ATMEGA328, presente en tarjetas como la Arduino Uno y Arduino Mega; TEENSY, es la opción para el procesador de las placas de la marca Teensy, estas placas son más caras pero tienen mayores capacidades; DEBUG, que permite entrar en un modo donde una herramienta llamada serial monitor del IDE de Arduino lee la información que envían los componentes conectados a la tarjeta, lo que permite monitorear si están conectados y programados de forma correcta. (Silveira, 2019)

```
// Choosing your board
// Define your board, choose:
// "ATMEGA328" if using ATmega328 - Uno, Mega, Nano...
// "ATMEGA32U4" if using with ATmega32U4 - Micro, Pro Micro, Leonardo...
// "TEENSY" if using a Teensy board
// "DEBUG" if you just want to debug the code in the serial monitor

#define ATMEGA32U4 1//* put here the uC you are using, like in the lines above followed by "1", like "ATMEGA328 1", "DEBUG 1", etc.

////////////////////////////////////////////
// Are you using buttons?
#define USING_BUTTONS 1 //* comment if not using buttons

////////////////////////////////////////////
// Are you using potentiometers?
#define USING_POTENTIOMETERS 1 //* comment if not using potentiometers

////////////////////////////////////////////
// Are you using a multiplexer?
#define USING_MUX 1 //* comment if not using a multiplexer, uncomment if using it.

////////////////////////////////////////////
// Are you using encoders?
#define USING_ENCODER 1 //* comment if not using encoders, uncomment if using it.

////////////////////////////////////////////
// Are you using neopixels (any addressable strips)?
//#define USING_NEOPIXEL 1 //* comment if not using neopixels, uncomment if using it.

////////////////////////////////////////////
// Are you using an I2C Oled Display?
#define USING_DISPLAY 1 //* comment if not using an I2C Oled Display.
```

Figura 26. Parte inicial del código de la plantilla. Tomado de: (Silveira, 2019)

En la figura 26 se puede observar el inicio del código de esta plantilla. Como se mencionó anteriormente, utilizar doble *slash* "//" sirve para convertir una línea en un comentario que no afecte la funcionalidad del código. A través de estos comentarios el programador del código da indicaciones de varios tipos; también así se pueden utilizar para prender o apagar la programación de algunos componentes.

El código comienza preguntando qué componentes se van a utilizar en el proyecto. Por defecto la programación de cada componente está apagada, para encenderse se debe encontrar la línea relacionada a este, por ejemplo:

“//#define USING_DISPLAY” (relacionada a la pantalla OLED), y se debe eliminar el doble slash antes del comando *#define*, que sirve para “definir” (en este lenguaje de programación estar “definido” es un estado) la palabra clave que se encuentra a su lado, en este caso “USING_DISPLAY”.

En otra pestaña llamada SETUP, se utiliza un comando llamado *#ifdef*, al lado del cual se escribe la palabra clave, en este caso sería “#ifdef USING_DISPLAY”, esto sirve para condicionar la programación que se encuentra en las líneas posteriores a este comando. En otras palabras:

“#ifdef USING_DISPLAY” se traduciría a: “si, USING_DISPLAY se encuentra definido, hacer lo siguiente”. Estas programaciones condicionadas se terminan con el comando *endif*, que actúa como si cerrara un paréntesis donde se encuentra la programación. (Silveira, 2019)

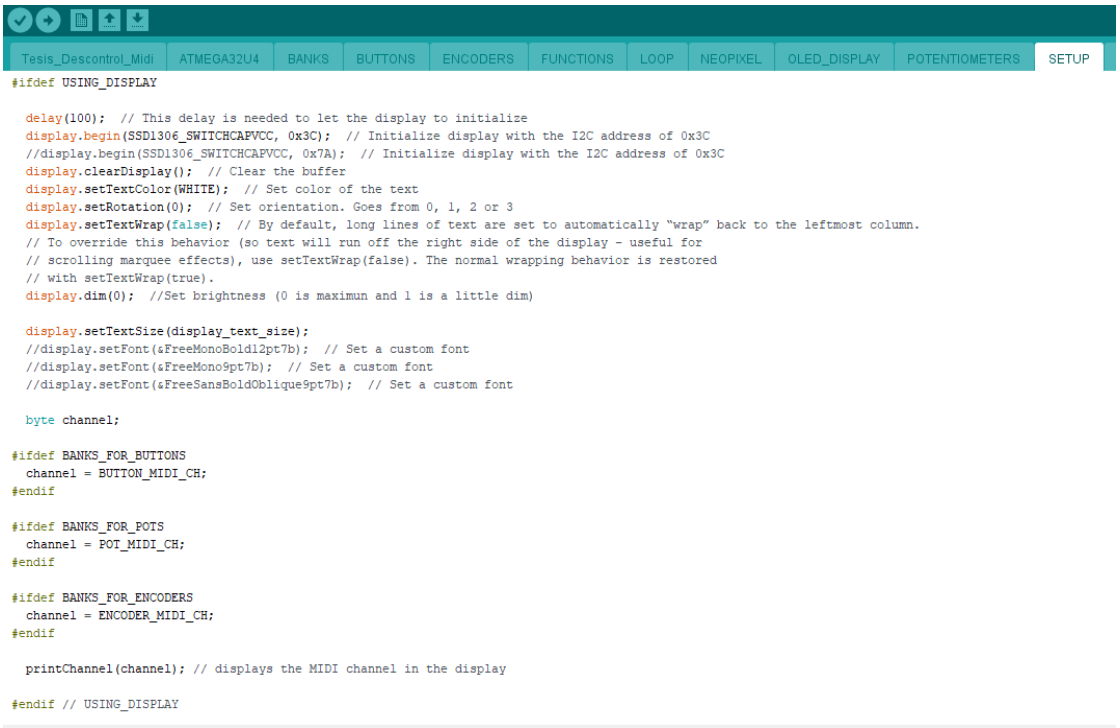


Figura 27. Pestaña SETUP con la programación del display OLED. Tomado de (Silveira, 2019)

De esta manera, la programación de cada componente solo se activará si su palabra clave se encuentra definida en la pestaña principal. Como se puede ver en la figura 26, en este proyecto la única línea apagada es "//#define USING_NEOPIXEL", que serviría para programar tiras de leds automatizables, pero en este proyecto no se utilizarán.

En esta plantilla la pestaña SETUP se utiliza para escribir todos los valores que serán constantes en la programación, es decir que, para funcionar bien no deben ser cambiados, mientras que en la pestaña principal se encuentran todos los valores que serán variables y que para funcionar bien necesitan ser cambiados. (Silveira, 2019)

La siguiente parte del código pide añadir varias librerías de programación al IDE de Arduino, las cuales son códigos realizados por terceros para facilitar la programación de los componentes y que fueron utilizadas en esta plantilla. Estas se añaden con el comando *#include* seguido del nombre de la librería entre los signos de menor que "<" y mayor que ">" como se puede apreciar en la figura 28.

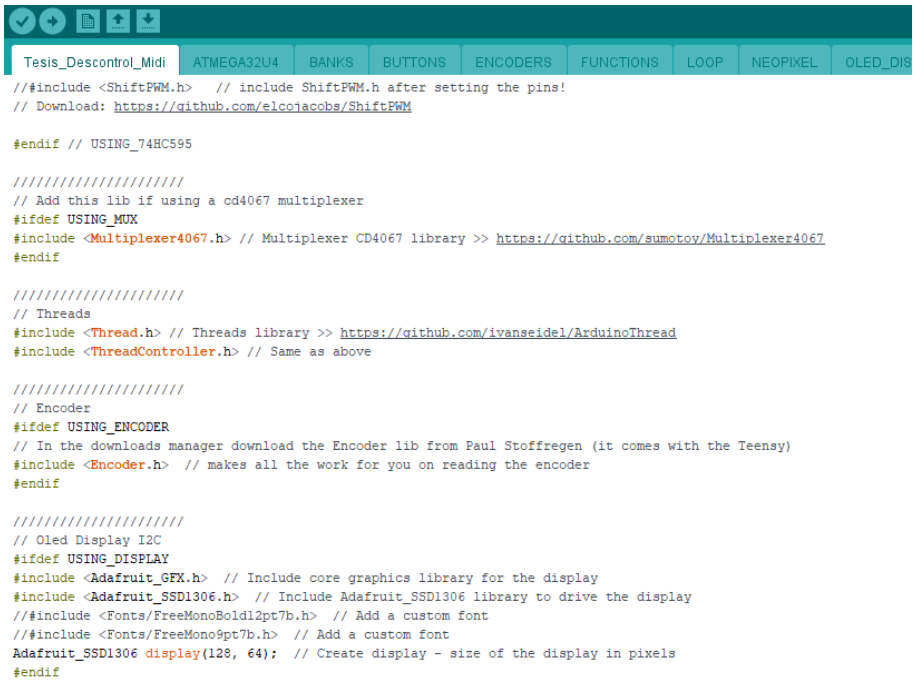


Figura 28. Sección del código donde se includen las librerías. Tomado de (Silveira, 2019)

Luego de añadir las librerías se procede a la parte del código donde se encuentran las variables que se debe cambiar según las especificaciones del proyecto.

```
//////////////////////////////////////////
// BUTTONS
#ifdef USING_BUTTONS

const byte N_BUTTONS = 11; //*  total numbers of buttons. Number of buttons in the Arduino + number of buttons on multiplexer 1 + number of buttons on multiplexer 2...
const byte N_BUTTONS_ARDUINO = 0; //* number of buttons connected straight to the Arduino
const byte BUTTON_ARDUINO_PIN[N_BUTTONS] = {}; //* pins of each button connected straight to the Arduino

#ifdef USING_MUX // Fill if you are using mux, otherwise just leave it
const byte N_BUTTONS_PER_MUX[N_MUX] = {11}; //* number of buttons in each mux (in order)
const byte BUTTON_MUX_PIN[N_MUX][16] = { //* pin of each button of each mux in order
{3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15}, //* pins of the first mux
// ...
};
#endif
```

Figura 29. Programación de los botones. Tomado de (Silveira, 2019)

Para la programación de los botones, lo primero que habrá que hacer será declarar el número total de botones, sin contar los botones que serán utilizados para seleccionar el canal MIDI, en este proyecto fueron once.

Lo siguiente será declarar cuantos botones están conectados directamente a la tarjeta Pro Micro, nuevamente sin contar con los botones que eligen el canal MIDI, en este caso no hubo ningún botón conectado directamente a la tarjeta. En el siguiente párrafo se debe declarar cuántos botones fueron conectados al *multiplexer,* y enumerar cada pin del *multiplexer* al que está conectado un botón, en el proyecto actual el total de botones conectados al *multiplexer* fue de once y los pines a los que fueron conectados se pueden revisar en la tabla 2.

```
#ifdef USING_BANKS_WITH_BUTTONS

const byte BANK_BUTTON_PIN[2] = {9, 8}; //* first will decrease MIDI chennel and second will increase
int buttonBankCState[2] = {0};
int buttonBankPState[2] = {0};
#endif
```

Figura 30. Programación de la selección de canales MIDI. Tomado de (Silveira, 2019)

Para la selección de canales MIDI fue necesario declarar cuáles eran los pines de la tarjeta a los que estaban conectados los botones designados, la plantilla no permitía que estos botones estuvieran conectados a un *multiplexer*, por lo tanto estos botones fueron conectados en los pines 8 y 9 de la Pro Micro.

```
//////////////////////////////////////////////
// POTENTIOMETERS

#ifdef USING_POTENTIOMETERS

const byte N_POTS = 5; //* total numbers of pots (slide & rotary). Number of pots in the Arduino + number of pots on multiplexer 1 + number of pots on multiplexer 2..

const byte N_POTS_ARDUINO = 3; //* number of pots connected straight to the Arduino
const byte POT_ARDUINO_PIN[N_POTS_ARDUINO] = {A0, A1, A2}; //* pins of each pot connected straight to the Arduino

#define USING_CUSTOM_CC_N 1 //* comment if not using CUSTOM CC NUMBERS, uncomment if using it.
#ifdef USING_CUSTOM_CC_N
byte POT_CC_N[N_POTS] = {11, 13, 14, 15, 16}; // Add the CC NUMBER of each pot you want
#endif

#ifdef USING_MUX
const byte N_POTS_PER_MUX[N_MUX] = {2}; //* number of pots in each multiplexer (in order)
const byte POT_MUX_PIN[N_MUX][16] = { //* pins of each pot of each mux in the order you want them to be
{7, 8} //* pins of the first mux
// ...
};
#endif
```

Figura 31. Programación de los potenciómetros. Tomado de (Silveira, 2019)

En la figura 31 se puede encontrar la sección del código correspondiente a la programación de los potenciómetros, que es similar a la de los botones, ya que primero se declaró que se utilizaron cinco potenciómetros en total, luego se determinó que tres potenciómetros fueron conectados directamente a la tarjeta, se especificó sus pines, y se realizó el mismo procedimiento pero para los potenciómetros que estaban conectados al *multiplexer*.

```
//////////////////////////////////////////////
// ENCODERS
#ifdef USING_ENCODER

//#define TRAKTOR 1 // uncomment if using with traktor, comment if not

const byte N_ENCODERS = 2; //* number of encoders
const byte N_ENCODER_PINS = N_ENCODERS * 2; //number of pins used by the encoders
const byte N_ENCODER_MIDI_CHANNELS = 16; // number of ENCODER_MIDI_CHs

byte ENCODER_CC_N[N_ENCODERS] = {17, 18}; // Add the CC NUMBER of each encoder you want

Encoder encoder[N_ENCODERS] = {{10, 16}, {14, 15}}; // the two pins of each encoder -  Use pins with Interrupts!

byte encoderMinVal = 2; //* encoder minimum value
byte encoderMaxVal = 127; //* encoder max value

byte preset[N_ENCODER_MIDI_CHANNELS][N_ENCODERS] = { //* stores presets to start your encoders
  //  {64, 64}, // ch 1
```

Figura 32. Programación de los *encoders* de rotación. Tomado de (Silveira, 2019)

Los *encoders* de rotación requieren ir conectados a *inputs* digitales con capacidad PCINT, que tienen tiempos de lectura más rápidos que los puertos normales, en el caso de la Pro Micro los puertos que tienen estas capacidades son los *inputs* 10, 16, 14 y 15. En la programación se especificaron estas entradas, así como el número de encoders, número de pins usados por los *encoders* (este valor es de dos en la figura 32 pero fue cambiado a cuatro). Luego se limita un rango de valores entre uno y ciento veintisiete entre los

cuales podrá variar el encoder. En este proyecto los encoders que se pudieron adquirir no funcionaban de forma adecuada, ya que a veces al manipularse se reducían dos números de un solo salto, lo que hacía que desde el valor uno se saltara al ciento veintisiete, por lo cual se tuvo que programar que el valor mínimo fuera dos, de esta forma ya no ocurría lo antes descrito.

```
////////////////////////////////////////////
// DISPLAY
byte display_pos_x = 27; // pos x
byte display_pos_y = 7; // pos y
byte display_text_size = 7; // text font size

////////////////////////////////////////////
```

Figura 33. Programación de la pantalla OLED. Tomado de (Silveira, 2019)

En la plantilla, la programación de la pantalla OLED solo requería de la asignación de los valores de las posiciones en el eje X y Y de de la pantalla de 128x64 pixeles y del tamaño de la fuente.

2.8 Producción de la performance.

2.8.1 Incorporación en Ableton Live.

La plantilla de programación del dispositivo permitió una conexión inmediata *plug and play* con cualquier ordenador al que se conectara, en cualquier sistema operativo y DAW.

Para realizar la performance se realizaron varios mapeos previamente.

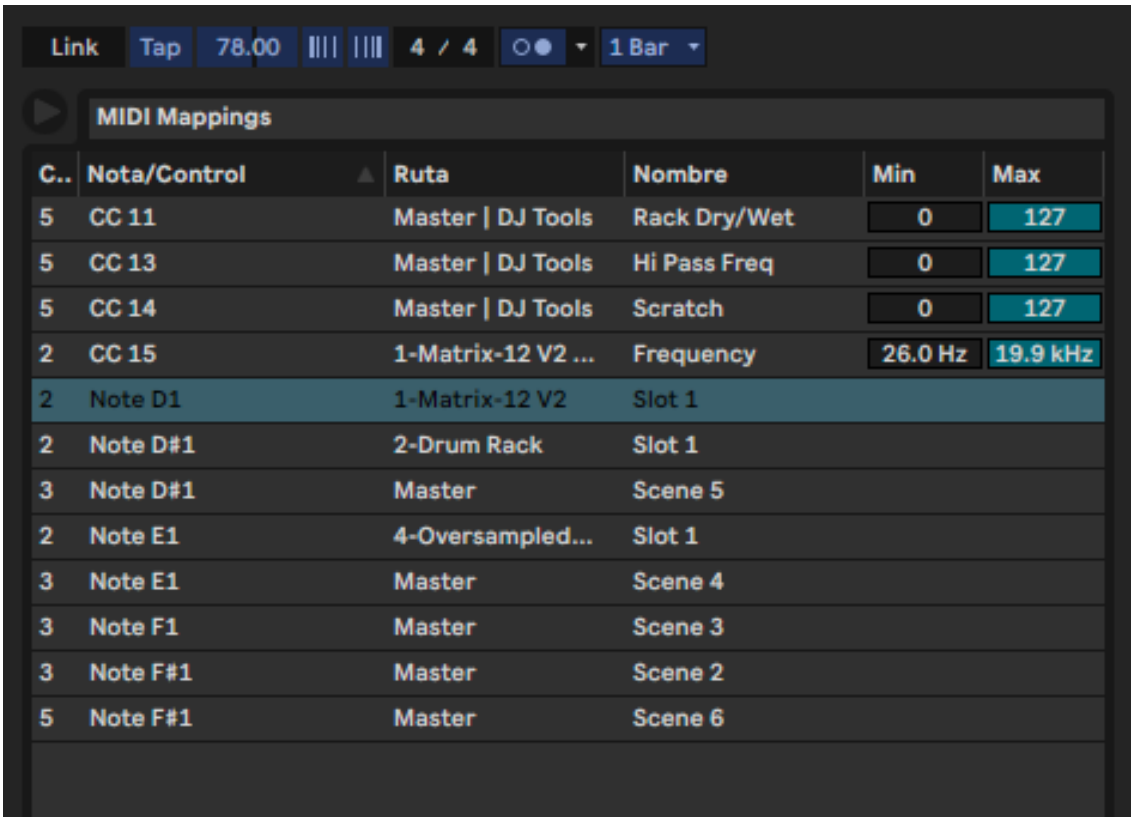


Figura 34. Mapeo de componentes en Ableton Live

De esta manera se utilizaron varios de los componentes del controlador MIDI para controlar diferentes secciones de la performance.

Los botones tenían asignadas notas por defecto en la programación, sin embargo, gracias a la capacidad del controlador de cambiar de canales MIDI, dependiendo del canal MIDI seleccionado estos pueden actuar como notas o mapearse a un control del DAW.

En este caso los botones se utilizaron de las dos formas, inicialmente utilizándose para tocar notas en la *drum machine* genérica de Ableton Live que se llama Drum Rack, creando así un *loop* del primer *beat* presentado esto en el canal dos del controlador.

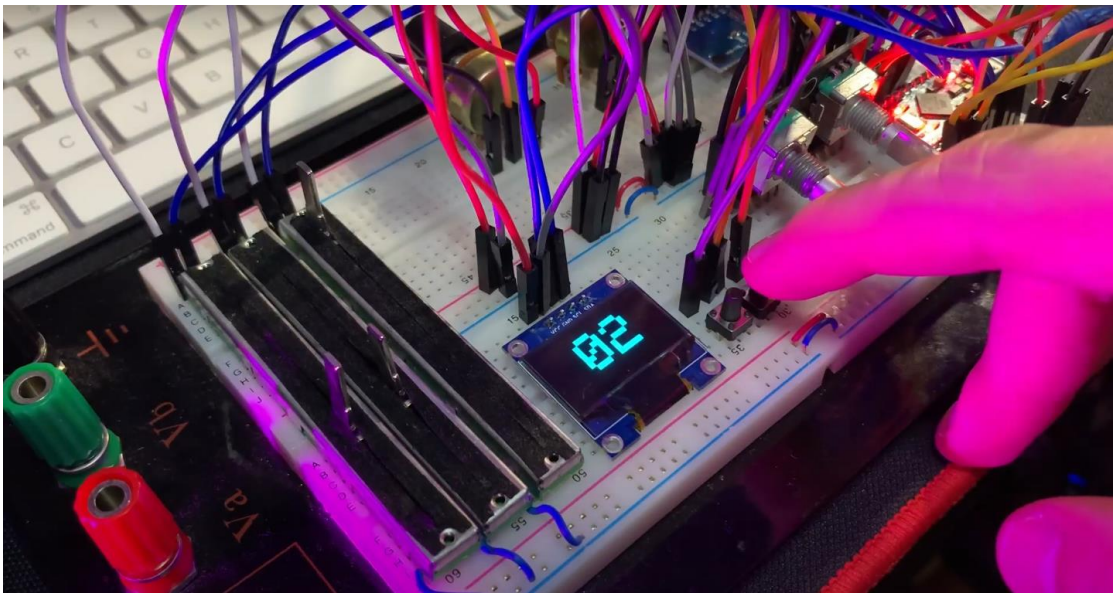


Figura 35. Controlador siendo utilizado en la performance en el canal dos. El canal es elegido mediante los botones que se encuentran a su derecha.

Luego, se cambió al canal tres, que estaba mapeado a las llamadas “escenas” de la sesión de Ableton Live, que son controles que se encuentran solo en la *sesión view* en el canal de *master*. Estos sirven para lanzar todas las secuencias que se encuentren en la escena, de esta forma se controla la duración de cada sección.

Figura 36. Mapeo de *triggers* de escenas a botones del controlador MIDI.

Para finalizar con los botones, en el canal cuatro fueron utilizados para realizar una improvisación con *samples* recortados de un audio mediante la herramienta “Simpler” de Ableton Live, que sirve para procesar y recortar muestras de audio de forma rápida e inteligente, y mediante una opción de la misma herramienta fueron trasladados al instrumento Drum Rack mencionado anteriormente, donde a través de los botones funcionando nuevamente como notas, era posible lanzar distintos recortes del audio de una forma musical.

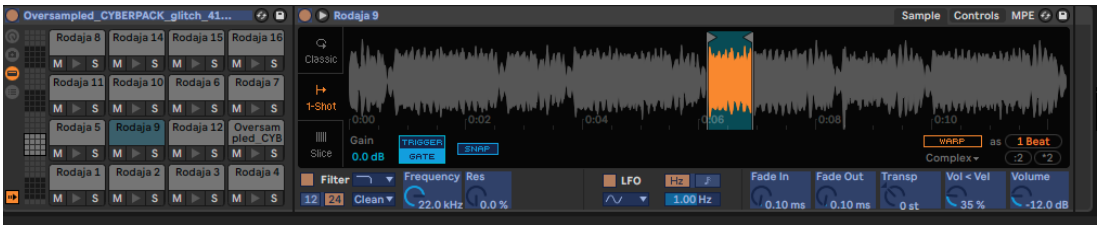


Figura 37. Drum Rack con las rodajas de audio recortadas a través de la herramienta Simpler.

En cuanto a los demás componentes, se utilizaron los potenciómetros para controlar distintos efectos dentro de la performance.

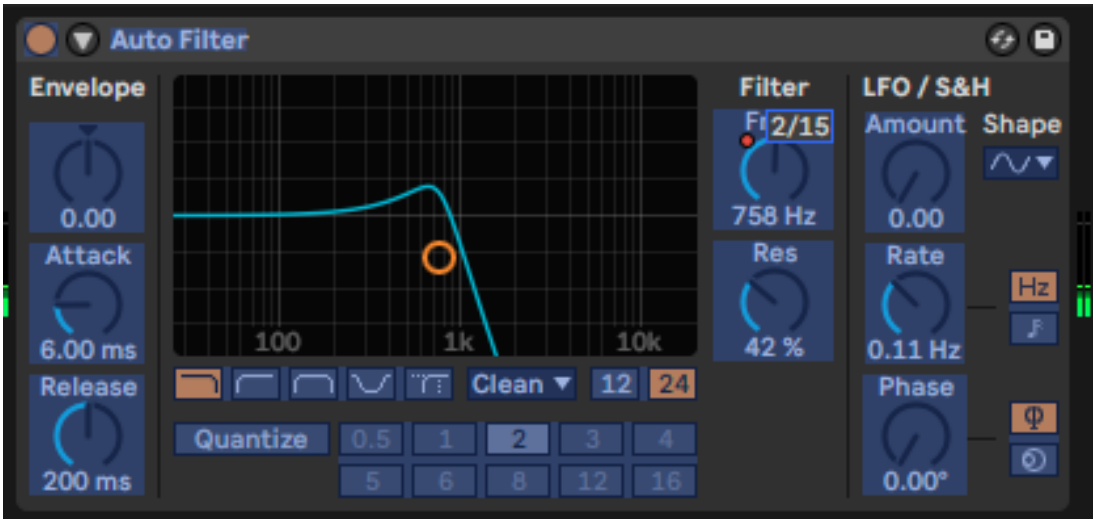


Figura 38. Auto Filter con un mapeo de la frecuencia al potenciómetro CC15, en el canal dos.

Uno de los potenciómetros fue mapeado a la frecuencia del filtro para controlar la apertura de un bajo sintetizado al inicio de la performance.

Luego se utilizan los tres *faders* del controlador para controlar una cadena de efectos al final de la performance en el canal del máster, para que tenga efecto sobre todos los canales, uno de los faders controla el parámetro *dry/wet* de esta cadena, otro abre y cierra un filtro de tipo high pass, y finalmente otro fader controla un efecto de *scratch* mientras la secuencia llega a su fin para terminar la performance.

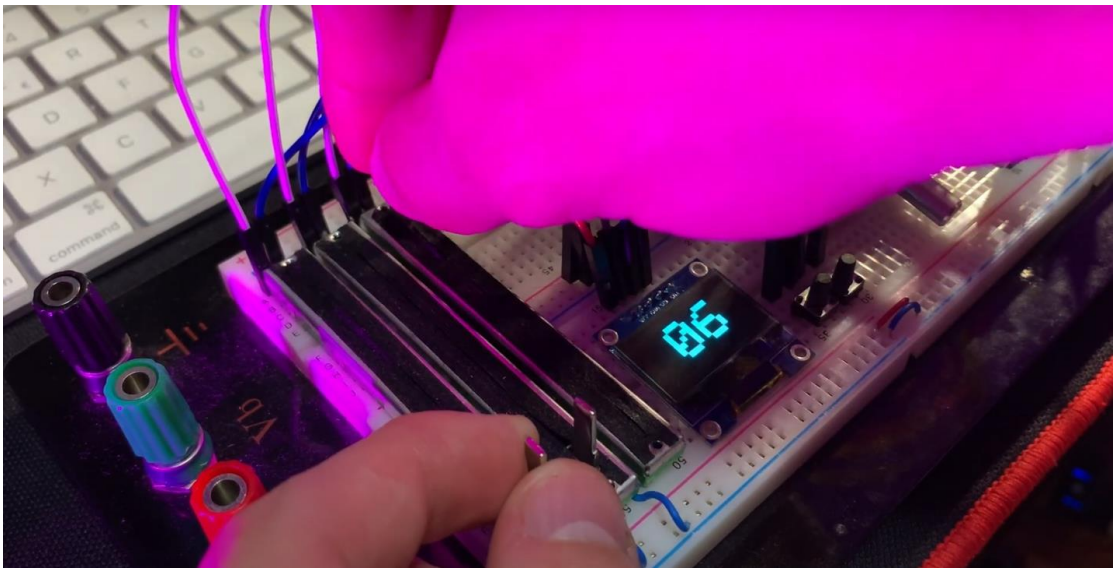


Figura 39. Los tres faders del controlador siendo utilizados para controlar los efectos al final de la performance en el canal seis.

Adicional a todos estos controles se utilizó un Ableton Push 2, un controlador MIDI muy famoso en el mercado mundialmente y que tiene bastante sinergia con el DAW utilizado, este sirvió para recibir un feedback de las secciones del tema, moverse por distintos canales, y tocar algunas notas al inicio de la performance. Con los 7 botones del controlador MIDI realizado en este trabajo la cantidad de notas era una gran limitación para tocar melodías, ya que los botones no están pensados para tocar notas musicales sino para tocar samples en una *drum machine.*

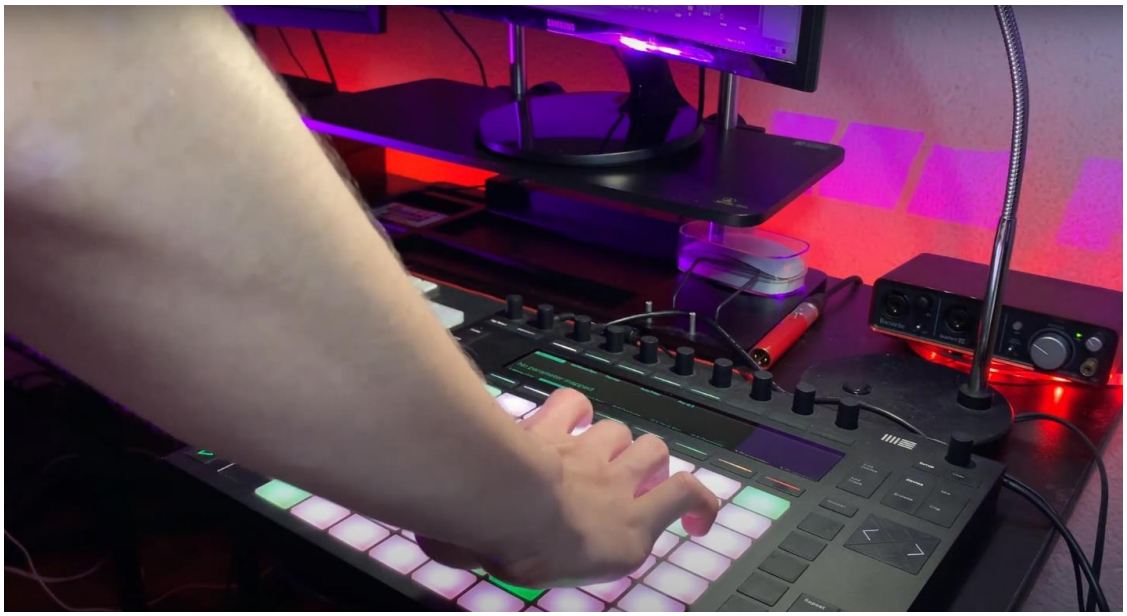


Figura 40. Ableton Push 2 siendo utilizado para grabar una secuencia de bajo sintetizado.

Todo el audio de la performance fue grabado en vivo, pero en el video se añadieron distintas tomas grabadas posteriormente para resaltar cada componente al ser utilizado. Posteriormente se mezcló y masterizó el audio para obtener un resultado profesional.

La performance se puede encontrar en el siguiente enlace: https://youtu.be/nWADH6IPFNU

También se realizó un video explicativo sobre la realización de la performance: https://www.youtube.com/watch?v=D3O4jw1S04k

3 Conclusiones y Recomendaciones

El presente trabajo concluyó con la realización de un controlador MIDI funcional, que puede ser utilizado en cualquier DAW y sistema operativo, el cual se utilizó para realizar una performance captada en video. Sin embargo, hubo varias limitaciones en su realización.

Debido al tiempo de duración del proyecto, el producto final fue el prototipo de las conexiones del controlador MIDI, que es en sí un controlador funcional, pero que todavía no cumple con las expectativas de diseño y de calidad que se preveía al inicio del trabajo, esto pasó debido a que ninguna de las alternativas de diseño para la carcasa se alcanzó a concretar, sin embargo, el dispositivo seguirá en desarrollo por lo que se espera completar su diseño en el futuro.

A través de la fabricación de este dispositivo se adquirieron conocimientos y habilidades relacionadas a las áreas de la producción musical, la tecnología y la programación.

La investigación fue un proceso clave para el desarrollo del presente trabajo, a través de ella se obtuvo la información suficiente de las distintas disciplinas que implican la producción de un dispositivo de este tipo, y el resultado fue una comprensión mucho más profunda de los controladores MIDI, que son las herramientas del futuro para la producción musical.

Incluso se obtuvo conocimientos en otras materias, por ejemplo, se llegó a comprender la estructura de los aparatos electrónicos y lo que implica su programación, además, cómo se fabrican, donde se pueden encontrar los componentes necesarios para su realización, cuánto cuesta hacerlos a mano, etc.

Como recomendación, para la realización de trabajos de este tipo es importante tener la supervisión de un tecnólogo o de una persona con experiencia y habilidad para las cuestiones técnicas, al menos si es la primera vez realizando un proyecto de este tipo, ya que existen algunas cuestiones que requieren habilidad, como la soldadura de los componentes, ya que si no se

cuenta con conocimiento en esta área es posible dañar los puertos de la tarjeta y el *multiplexer*, y el trabajo podría aplazarse de manera indefinida hasta comprar nuevos componentes. Además, es posible equivocarse en cosas básicas, por ejemplo, en confundir la distribución de los contactos en una *protoboard*, o conectar un componente análogo en un puerto exclusivamente digital.

Un aspecto importante en la realización de este controlador fue el costo total que tuvo finalmente. En general los costos de los controladores MIDI que se encuentran actualmente pueden variar mucho, hay distintas gamas y propósitos. El costo actual en promedio de un controlador MIDI de gama baja ronda los cien dólares. El costo los componentes de este dispositivo se encuentra en la siguiente tabla.

Tabla 3. Costo de los componentes del dispositivo.

Componente	Precio
11 Botones	$3
Pro Micro	$10
Multiplexer	$5
2 Potenciómetros de Rotación	$2.50
3 *Faders*	$12
Pantalla OLED	$5
2 *Encoders* rotativos	$3
Costo tentativo de la carcasa	$15
Costo tentativo de los cables	$4
Costo total	$59.5

Como se puede apreciar en la tabla 3, el costo final del producto sería de casi sesenta dólares, lo que no es especialmente una cantidad baja ni alta, se encuentra dentro de lo normal de lo que costaría un dispositivo similar manufacturado industrialmente. Sin embargo, se debe tomar en cuenta que todos los componentes fueron comprados en el país y que por ello sus precios suelen duplicarse o hasta triplicarse en comparación a los precios que tienen los mismos componentes en otros países, donde es mucho más viable hacer proyectos de este tipo. De todas maneras, este trabajo es un precedente para futuras investigaciones y trabajos similares en el país y en toda la región, ya que no son comunes este tipo de proyectos en Latinoamérica debido quizás al retraso tecnológico y cultural que existe en la zona.

Es importante para la música y para el avance tecnológico de toda la región que se sigan realizando proyectos de esta índole para que la región en algún punto sea capaz de competir con el resto del mundo en esta y otras industrias.

Referencias

Ableton. (2018). Ableton Reference Manual Version 10 (10ma ed.) [Libro electrónico]. Ableton AG. https://cdn-resources.ableton.com/resources/0b/c1/0bc1007e-bd0b-4d6f-b52d-ee0054f3a6f8/l10manual_en.pdf

Banzi, M., & Shiloh, M. (2014). Getting Started with Arduino: The Open Source Electronics Prototyping Platform (3ra ed.). Make Community, LLC.

D. Huber (2007) The MIDI Manual: A Practical Guide to MIDI in the Project Studio. FocalPress.

Gurevich, M., & von Muehlen, S. (2020). The Accordiatron: A MIDI controller for interactive music. arXiv

Huber, D. M., & Runstein, R. E. (2017). Modern Recording Techniques (9na ed.). Routledge.

Jansson, D. [Switch & Lever]. (2019). Building a MIDI Controller Using Arduino [mp4]. Recuperado de https://www.youtube.com/watch?v=JZ5yPdoPooU&t=263s

Jensenius, A. et al (2020). Vrengt: A Shared Body-Machine Instrument for Music-Dance Performance. Recuperado de: https://www.researchgate.net/publication/344551909_Vrengt_A_Shared_Body-Machine_Instrument_for_Music-Dance_Performance

Jimblom (2013) Placa Pro Micro [Esquema]. Recuperado de: https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide?_ga=2.214658260.2127936738.1620667735-1755365861.1620667735

Jimblom (2013) Pro Micro & Fio V3 Hookup Guide. Recuperado el 10 de Mayo de 2021, de https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide?_ga=2.214658260.2127936738.1620667735-1755365861.1620667735.

Maxuino. (2014). Maxuino » Getting Started. http://www.maxuino.org/getting-started

Miranda, E. R., & Wanderley, M. M. (2006). New digital musical instruments: control and interaction beyond the keyboard (Vol. 21). AR Editions, Inc.

Pieter, P. (2017). Arduino MIDI. https://tttapa.github.io/PDF/Arduino-MIDI.pdf

Roland. (2009). An Introduction to MIDI (Vol. 1). Roland Corporation US. https://www.midi.org/images/easyblog_articles/43/intromidi.pdf

Rothstein, J. (1995). Midi: A Comprehensive Introduction (2 Sub ed.). A-R Editions.

Schloss, W. A. (2003). Using Contemporary Technology in Live Performance: The Dilemma of the Performer*. Journal of New Music Research. https://doi.org/10.1076/jnmr.32.3.239.16866

Silveira, G. (2019). DIY-Midi-Controller-full. *Making Music with Arduino.* https://mmwa.musiconerd.com/courses/546297/lectures/31147165

Solanas Donoso, J. (1981). Diseño, arte y función (1st ed., pp. 14-19). Salvat.

Velte, M. (2012). A MIDI Controller based on Human Motion Capture. University of Applied Sciences. https://doi.org/10.13140/2.1.4438.3366

ANEXOS

Anexo 1. Cuaderno de campo.

Programación:

1) Elegir la librería según el modelo de chip de la tarjeta

```
# define ATMEGA32U4 1 //
```

Chip de la Pro Micro.

Para corregir errores cambiar por

```
# define DEBUG 1 //
```

La primera página de la plantilla es donde se pueden prender y apagar las partes más generales. Para apagar una sección (Por ejemplo, el multiplexer) comentar

//

antes del código en el que se define dicho componente.

Librería Para multiplexer
Multiplexer4067.h

Librería para MIDI en ATMEGA32U4
USBMIDI.h

Librería para Encoders
Encoder.h

Para usar el multiplexer

Descomentar // en "are you using a multiplexer" luego buscar USING_MUX con control+F

El primer resultado será para añadir la librería que utiliza el multiplexer.

Primero hay que elegir el número de multiplexers

#define N_Mux 1 → Número de multiplexers

Luego hay que definir que pines de arduino corresponden a los pines del multiplexer

En este caso:

```
#define S0 7
#define S1 6
#define S2 5
#define S3 4
#define x1 A3
```

Luego de esto cargué el programa en el Pro Micro y automáticamente se convirtió en un plug and play.
Sin embargo, solo están programados los botones por ahora y no he asignado el Midi CC específico a cada botón

Multiplexer potentiometer:

Definir qué potenciómetros están conectados directamente a la tarjeta

```
#ifdef USING MUX
const int N_POTS_PER_MUX] = 2
```

y en que pines están conectados.

(C7/C8)

```
const int POT_MUX_PIN [N-MUX][16]
= → {solo si Hay varios multiplexers}
= {7, 8} → En caso de ser un único multiplex
    ↳ Aquí están conectados los míos.
```

En el inicio del código recordar descomentar el párrafo de potenciómetros para encenderlos, ajustar el threshold para que se sienta más agradable la manipulación del potenciómetro.

Rotary Encoder: Hay que asegurarse de que estos vayan conectados a un input con capacidad "PCint" ya que necesitan una mayor cantidad y velocidad de procesamiento. En este proyecto se utilizaron los pines 15, 14, 16 y 10 de la pro Micro, todos con PCINT.

Programación: Primero hay que descomentar la sección de encoders para prenderla. Hay que descargar la librería de encoders. <Encoder.h>

N_ENCODERS = 2; → Para definir el número de encoders

N_ENCODER_CHANNELS = 1 → Para usar diferentes canales.

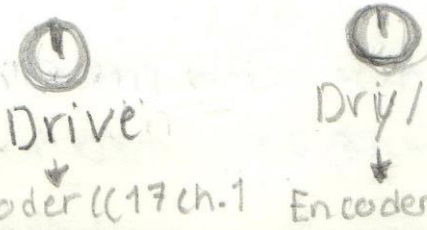
Luego definir el punto desde el que comenzará el encoder {64}

Cambio de banco MIDI con 2 botones:

- Prender los bancos para botones
#Define Banks_FOR_BUTTONS 1

Luego ir a Ifdef USING BANKS WITH BUTTONS
cambiar el número total de Botones que habían antes y especificar cuáles son los pines que se van a usar ahora (solo pueden ser pines de la Pro Micro, no pueden ser del multiplexer)

Con los bancos, y con una programación aparte, es posible controlar muchos parámetros asignados a distintos canales midi.
Para ello solo hay que activar los encoders en la línea using encoders with Banks. y se puede mapear por ejemplo:



Usando MIDI CC: Con esto se puede hacer que cada nota de los botones sea la nota que especifiquemos en la programación, evitando así confusiones como que el botón del encoder rotatorio justo sea el que controla el kick en una batería, siendo uno de los botones menos alcanzables.

Hay que designar a cada botón en orden qué nota de MIDI será o qué nota de CC será (Para asignarse a otros parámetros)

El orden en este caso será:

CC de los Potenciómetros:

{11, 13, 15, 16, 19}

CC de los Encoders

{17, 18}

Orden de las notas en los botones

{0, 1, 42, 41, 40, 39, 38, 37, 36}

Estos son los botones del encoder

etc ← D#4 D4 C#4 C4

Para que el primer botón sea el kick

Pantalla OLED I2C: inicialmente no estaba previsto utilizar este componente en el proyecto, pero resultó necesario ya que es primordial tener un feedback visual del canal midi que se está utilizando para poder mapear todo de forma correcta y que el performance no se vea afectado por estar desorientado sobre cuál es el canal midi en el que nos encontramos.
En el código la programación de este display es complicada ya que el código vino con algunos problemas como:

- No había nada definido para el modo DEBUG
- La tarjeta se crashea cuando se intenta programar este componente erróneamente
- Para revivir la tarjeta: Se debe conectar un cable al pin "RST" (Reset) de la tarjeta, a este se le mapeará un botón conectado a tierra, se debe hacer doble click y por 8 segundos la tarjeta volverá a su forma pre-

determinada, estos 8 segundos hay que aprovecharlos para subir cualquier código estable desde el IDE de Arduino, de esta manera se saca al procesador del Loop de errores.

Una vez que se saca a la tarjeta de este error se puede intentar una vez más de programar la pantalla.

En el código principal se debe simplemente prender la opción del OLED:

```
//USING_OLED_DISPLAY
```

Abajo se debe aclarar la posición del texto en la pantalla de 128x64,

```
byte display_posx = 34;
byte display_posy = 8;
byte display_text_size = 6;
```

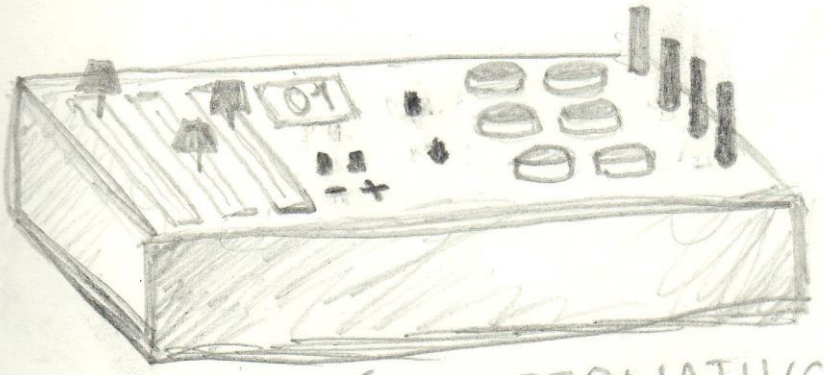
En las sub pestañas se ve las constantes de la programación para que se imprima todo en la pantalla.

Aquí es donde tuve que agregar una corrección al código ya que necesitaba que hubiera algún tipo de feedback en el modo DEBUG,

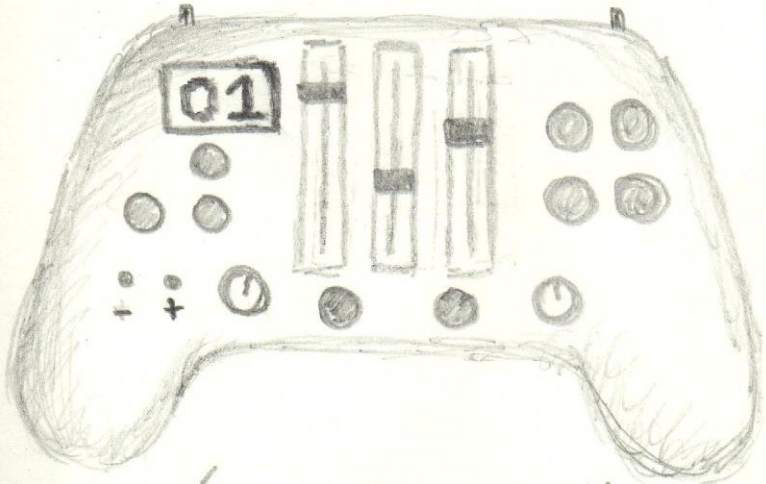
```
// Print text
# elif ATMEGA32U4
  display.print(n+1);
  display.display();
# elif DEBUG
  display.print(n+1);
  display.display();
# endif
}
# endif.
```

En la siguiente sub pestaña se eligen distintas características de la librería, elegir el color de la letra, aunque en este caso el display es de un solo color, se puede elegir fuentes personalizadas, también la intensidad de la iluminación etc.

Con el avance y terminación del prototipo se plantean más alternativas para el diseño ya que por el tiempo puede ser más beneficioso un diseño alternativa más sencillo de construir por el tiempo con el que se cuenta.



OPCIÓN ALTERNATIVA



OPCIÓN ORIGINAL