RAE

1. **TIPO DE DOCUMENTO:** Trabajo de grado para optar por el título de INGENIERO DE SONIDO.

2. **TÍTULO:** DISEÑO Y CONSTRUCCIÓN DE UN SECUENCIADOR DIGITAL EN HARDWARE CON INTERFAZ A MODO DE PEDALERA Y MATRIZ DE TONOS

3. **AUTORES:** Oscar Alford, Mauricio Barranco

4. **LUGAR:** Bogotá, D.C.

5. **FECHA:** Julio de 2012

6 **PALABRAS CLAVE:** Electrónica, Secuenciador, Programación, Secuencias, MIDI, Arduino, Pantalla Táctil, Supercollider.

7 **DESCRIPCIÓN DEL TRABAJO:** Este documento abarca el proceso de diseño y construcción del sistema SUB!, un instrumento electrónico formado por un secuenciador digital por pasos y un sintetizador virtual. El secuenciador permite la programación, almacenamiento y reproducción de secuencias que pueden ser enviadas al sintetizador donde los eventos MIDI son transformados en señales de audio. Los sistemas de entrada y reproducción de información son suplidos por una matriz de tonos y una pedalera respectivamente, elementos que hacen del SUB! una gran herramienta para músicos que deseen explorar nuevas posibilidades creativas más allá de las brindadas por interfaces convencionales como teclados o secuenciadores de una sola nota por paso. Se trataran aspectos técnicos del diseño y se presentaran diversas pruebas realizadas sobre la funcionalidad del sistema y su interacción con software y hardware externo.

8 **LÍNEAS DE INVESTIGACION:** Tecnologías actuales y sociedad – Procesamiento de señales - Diseño de sistemas de sonido.

9 **FUENTES CONSULTADAS:** Hillen. Peter; A Microprocessor Based Sequencer For Voltage Controlled Electronic Music Synttetizers; 1977. M. Kusek, David; A Microcomputer Based Polyphonic Sequencer for Music Synthesizers; 1979. Muro, Don; The Art Of Sequencing: A Step By Step Approach; Alfred Publishing; 1993. Mayer, Wetzei; A New Generation Of Musical Sequencers. Edward V. Ramirez, Melvyn Weiss; Microprocessor fundamentals: hardware and software, 1980. Mandado, Enrique; Sistemas Digitales; 2007.

10 **CONTENIDOS:** Se describen los pasos seguidos para el diseño y construcción de un secuenciador digital en hardware y un sintetizador virtual, se presentan antecedentes, elementos teóricos relevantes en el tema , pruebas realizadas para evaluar la efectividad del sistema y conclusiones obtenidas a partir de este proceso.

11 **METODOLOGÍA: E**s de carácter empirico-analitico, con un enfoque metodológico con base en el procesamiento de señales y la programación de sistemas y circuitos digitales.

12 **CONCLUSIONES:** El sistema sub evidencia, según los resultados obtenidos, una fortaleza del diseño en cuanto a la rapidez y facilidad brindada para crear, almacenar y reproducir secuencias. El comportamiento del dispositivo prueba ser estable y preciso al interactuar con diferentes tipos hardware y software. Los resultados con los obtenidos para nuestro instrumento control en las pruebas de retraso de reloj MIDI, no revelan errores significativos en el comportamiento del SUB!. El secuenciador complementa las posibilidades creativas e interpretativas del sistema. Las pruebas de satisfacción arrojaron resultados positivos en cuanto a las capacidades creativas e interpretativas del sintetizador. En el proceso evaluativo, surge como principal desventaja del dispositivo su reducida capacidad de almacenamiento y edición de secuencias.

DISEÑO Y CONSTRUCCIÓN DE UN SECUENCIADOR DIGITAL EN HARDWARE CON INTERFAZ A MODO DE PEDALERA Y MATRIZ DE TONOS

REALIZADO POR:
ALFORD JINETE OSCAR JAVIER
BARRANCO RACEDO MAURICIO JOSE

UNIVERSIDAD DE SAN BUENAVENTURA
FACULTAD: INGENIERÍA
PROGRAMA: ING DE SONIDO
BOGOTA
2012

Tabla de contenidos

Listado de tablas

Listado de figuras

Lista de Anexos

Resumen.

Este documento abarca el proceso de diseño y construcción del sistema SUB!, un instrumento electrónico formado por un secuenciador digital por pasos y un sintetizador virtual. El secuenciador permite la programación, almacenamiento y reproducción de secuencias que pueden ser enviadas al sintetizador donde los eventos MIDI son transformados en señales de audio. Los sistemas de entrada y reproducción de información son suplidos por una matriz de tonos y una pedalera respectivamente, elementos que hacen del SUB! una gran herramienta para músicos que deseen explorar nuevas posibilidades creativas más allá de las brindadas por interfaces convencionales como teclados o secuenciadores de una sola nota por paso. Se trataran aspectos técnicos del diseño y se presentaran diversas pruebas realizadas sobre la funcionalidad del sistema y su interacción con software y hardware externo.

Abstract

This document covers the design and construction process for the SUB! system, an electronic instrument formed by a digital step sequencer and a virtual synthesizer. The sequencer allows programming, storing and reproduction of sequences that can be send to the synthesizer for MIDI to audio transformation. The data input and sequence launch system, are supplied by a tone matrix and a pedalboard respectively, this makes the SUB! a great tool for musicians looking into exploring new creative possibilities beyond the conventional keyboard and single step sequencers. Technical aspects on the software and hardware design will be treated as well as tests on the functionality and interaction of the system with third-party software and hardware.

INTRODUCCIÓN

Los secuenciadores son dispositivos electrónicos diseñados para crear y manejar patrones musicales generados por medios electrónicos. Hicieron su aparición en la industria musical a finales de la década del 70 influenciando fuertemente tendencias musicales surgidas en los 80's como el techno, el post-punk y el synth pop entre otras, esto hizo que la invención de máquinas como el Moog 960 y el Akai asq-10 fuera pieza clave en el desarrollo de la música moderna.

Los secuenciadores en hardware perdieron popularidad al aparecer los DAW's, estaciones de trabajo de audio digitales (Digital Audio Workstation). Software como Protools, Logic o Ableton Live permiten hoy día la integración de diversos elementos en hardware y software dentro de un estudio, incluyendo secuenciadores, samplers, sintetizadores, etc. Sin embargo, al tiempo que es destacable la comodidad y versatilidad que brinda una configuración de este tipo se debe resaltar el hecho que los DAW más que instrumentos musicales son herramientas ingenieriles, muy útiles en el estudio pero de muy poca funcionalidad para situaciones de composición e interpretación en tiempo real, es por esto que un músico generalmente preferirá utilizar un secuenciador en hardware para componer o interpretar piezas musicales.

Algunas de las ventajas que presenta un secuenciador en hardware comparado con un secuenciador en software, al ser usado como instrumento musical, son la capacidad de acceder rápidamente a las secuencias y la flexibilidad y facilidad para secuenciar. El SUB! enriquece estos aspectos mediante la implementación de una interfaz que permite el uso del secuenciador a modo de pedal, de esta manera, el usuario puede liberar las manos en el momento que lo necesite. Este secuenciador posee una matriz de tonos que permite secuenciar en tiempo real figuras tanto rítmicas como melódicas, el sistema también permite la rápida grabación de secuencias y su organización por patrones, de esta manera el SUB! se convierte en una herramienta musical muy útil en los tres ámbitos de la producción musical: creación, edición e interpretación.

Aproximaciones basadas en sistemas analógicos fueron desechadas tempranamente debido su poca capacidad de programación en tiempo real. Cabe resaltar el hecho de que el dispositivo no se encuentra en capacidad de reemplazar el secuenciador de un DAW, por el contrario, debe funcionar como esclavo de este o alguna otra máquina de tiempo maestra. En el capítulo 5 del presente documento, se verán ejemplos de montajes comúnmente utilizados que ilustran la funcionalidad del secuenciador en conjunto con otros sistemas.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 ANTECEDENTES (ESTADO DEL ARTE)

Uno de los primeros instrumentos musicales que implementó el uso de microcomputadores a nivel comercial, fue el Synare Sequencer de la compañía Star Instruments, Inc. El Synare Sequencer es un secuenciador digital polifónico orientado a la interpretación, este era utilizado hasta con cuatro sintetizadores de baterías electrónicas. La unidad tiene cuatro bancos de memoria capaces de almacenar hasta 32 eventos, los cuales pueden ser utilizados independientemente o de manera combinada. El secuenciador no sólo almacena los ritmos tocados en las baterías sino que también almacena los acentos y dinámicas que son tan importantes en la percusión. A diferencia de los secuenciadores análogos que deben ser lentamente programados a través de un complejo arreglo de potenciómetros e interruptores, el Synare Sequencer captura los beats al tiempo que estos son interpretados y luego los reproduce según sea requerido.

Un artículo que documenta el desarrollo del dispositivo fue presentado ante la AES (*Audio Engenieering Society*) por David M. Kusek y Star instruments inc. Según presenta Kusek:

"Los métodos digitales se dividen en dos categorías, basados en dispositivos electrónicos con componentes lógicos o basados en microprocesadores. Los diseños de componentes lógicos tienden a ser complicados en términos de cantidad de circuitos integrados y tamaño en general. De igual forma no suelen ser flexibles a cambios en el diseño durante el proceso de construcción. Varios sistemas basados en microprocesadores demostraron ser más deseables en términos del número de circuitos integrados, flexibilidad a cambios, tamaño de tablero, costos, facilidad de implementación y tiempo de desarrollo" (1979)

El proyecto fue desarrollado con un microcomputador de chip sencillo Intel 8048, el primer microprocesador lanzado por Intel el cual funciona de forma muy similar a la familia de los PIC. Al respecto Kusek menciona que:

"La elección de una aproximación basada en microcomputadores probó ser apropiada, debido a que permitió que el sistema fuera desarrollado en diferentes etapas con poco cambio en el hardware usado […] La ventaja de usar un computador programable como principal controlador del sistema es que atributos adicionales pueden ser

agregados a la máquina a través de cambios en el software sin tener que hacer modificaciones en el hardware. La electrónica programable permite al diseñador concebir la complejidad del instrumento en el software en lugar que en el panel de control" (1979,12).

Peter Hillen en su artículo A Microprocessor Based Sequencer For Voltage Controlled Electronic Music Syntesizers divide el proceso de diseño de un secuenciador digital en dos partes: Diseño de software y diseño de hardware. Estos diseños, a su vez son analizados en tres modos de uso: modo de composición, modo de edición y modo de reproducción. Cada modo se relaciona, según Hillen, con una de las etapas por las cuales un músico pasa al momento de realizar una composición.

"Existen tres distintas fases en el proceso por el cual un músico atraviesa en el momento de crear una composición. Primero es la fase de composición, en la que el músico coloca en el papel nuevas ideas aún no documentadas. A continuación puede interpretar la composición y finalmente realizar cambios en lo que ha compuesto. El orden de las últimas dos operaciones puede ser por supuesto al revés. En este sistema de secuencias por computador, hay tres correspondientes modos de operación: composición, modificación e interpretación" (Hillen,1977,3).

El proceso que ha sido descrito por Hillen, puede ser aplicado para definir tres situaciones en las que el secuenciador será comúnmente utilizado. Estas situaciones, que llamaremos ámbitos de la producción musical, son: ámbito creativo, en vivo y de edición.

Con respecto al diseño del software, Hillen plantea lo siguiente: *"Los requerimientos para dicho sistema son que sea entendible por el usuario el cual en este caso es un músico y no un programador o un ingeniero, esto quiere decir que sea fácil y rápido de usar de tal manera que pueda ser utilizado en situaciones de interpretación en tiempo real y no se encuentre limitado al estudio de grabación. Algunos ítems a considerar para hacer el sistema operativo fácil de entender por el músico son utilizar nomenclatura familiar para el músico en lugar de lenguaje de computadora y ser claros al momento de guiar al usuario a través del sistema"* (Hillen,1977,3).

En el diseño de Hillen el dispositivo utiliza el modo de composición para almacenar información, el modo de edición para editarla y el modo de reproducción para reproducir. Sin embargo, el secuenciador no puede editar ni almacenar información mientras se encuentra reproduciendo una secuencia, de igual manera la interfaz es extremadamente limitante para su uso en vivo tal como se menciona en las conclusiones del artículo:

“El desempeño del secuenciador dentro del estudio es aceptable, sin embargo para su uso en vivo en deseable eliminar completamente la interfaz de computadora y reemplazarla con un teclado estilo calculadora que permita acceder a las funciones con sólo pulsar la tecla correspondiente a la función deseada” (Hillen, 1977,5).

Claramente lo mencionado por Hillen no es un problema hoy en día, cuando muchos de los secuenciadores implementan teclas o pads para acceder rápidamente a las funciones del dispositivo.

1.2 DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

La tecnología y las nuevas tendencias musicales crean nuevos tipos de músicos, es así como hoy día es común que guitarristas o bajistas deseen, además de interpretar su instrumento predilecto, hacer uso de elementos electrónicos mediante la implementación de secuenciadores.

La interacción de software y hardware con las aplicaciones requeridas por este tipo de usuarios, en términos de manejo de secuencias varían considerablemente, con respecto a las requeridas por alguien que solo se dedique a secuenciar; esto sucede tanto en vivo como en el estudio. Si se piensa en un guitarrista que desea activar una secuencia en el cuarto compás de una canción y luego desactivarla y activar otra secuencia en el octavo compás, este deberá dejar de tocar guitarra para pulsar la tecla que activa la correspondiente secuencia y así mismo cada vez que desee enviar un comando al secuenciador. Esto puede resultar impráctico en el caso de un guitarrista pero aún más en el caso de instrumentos base como el bajo.

Con la falta de una herramienta que permita a este tipo de músicos manipular secuencias en vivo, se pierde el toque humano del performance en vivo perdiendo “musicalidad”, refiriéndose a esa pérdida humana que puede presentarse en las secuencias que son realizadas por una máquina. Es claro entonces, que sobre todo para este tipo de músicos, existe la necesidad latente de un secuenciador que, de alguna manera, no demande la utilización de las manos y que permita conservar el carácter humano de la representación musical.

Teniendo en cuenta lo anterior, surge la pregunta de ¿Cómo diseñar y construir un secuenciador que permita una utilización más “musical” de secuencias por parte de músicos que requieran sus manos para interpretar otro instrumento?

1.3 JUSTIFICACIÓN

Al diseñar instrumentos electrónicos para el músico intérprete se debe considerar cuidadosamente la complejidad de la máquina. Esta debe tener un enfoque de simplicidad y su uso debe ser fácil, no solo en el estudio, sino también en situaciones en vivo. Mientras que la ubicación de un interruptor en algún punto del circuito puede ser un asunto irrelevante para un ingeniero, musicalmente esta ubicación puede ser de gran importancia, y es por esto que la interacción del hardware y software del dispositivo con las aplicaciones del usuario debe considerarse a fondo al momento de diseñar cualquier sistema de producción de audio.

Nuestro diseño se encuentra principalmente enfocado a músicos que desean hacer uso de secuencias electrónicas al tiempo que interpretan algún otro instrumento que les demande el uso de las manos (ej. bajo, guitarra, teclado). Se consideran las diferentes necesidades generadas por las aplicaciones que se requieren durante el proceso creativo y de interpretación y se responde a estas mediante dos elementos primordiales: Interfaz a modo de pedalera y matriz de tonos.

La interfaz a modo de pedalera permite al músico almacenar, disparar e interrumpir las secuencias sin necesidad de utilizar sus manos, mientras que la matriz de tonos permite programar rápidamente figuras rítmicas o armónicas.

El proyecto demanda relativamente pocos recursos económicos y físicos debido a que la complejidad de este radica principalmente en el desarrollo del software.

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 Objetivo General

- Construir un secuenciador digital por pasos con interfaz a modo de pedalera y un sistema de programación de secuencias por matriz de tonos, junto con un sintetizador en software que genere audio a partir de la información MIDI proveniente del secuenciador.

1.4.2 Objetivos Específicos

- Diseñar un sistema de hardware que permita la grabación, almacenamiento y reproducción de secuencias en tiempo real, mediante la implementación de una matriz de tonos, control de octavas y un sistema de almacenamiento de datos.

- Desarrollar un software que integre el sistema de hardware mencionado y además sea fácil y rápido de usar por parte de músicos.

- Desarrollar un software de síntesis que genere audio a partir de la información MIDI proveniente del secuenciador

- Evaluar la interacción del secuenciador con el software de síntesis diseñado, al igual que con otro hardware y software

- Evaluar el desempeño del dispositivo en los distintos ámbitos de la producción musical.

1.5 ALCANCES Y LIMITACIONES DEL PROYECTO

1.5.1 Alcances

Desarrollar un dispositivo funcional para músicos que deseen implementar la utilización de secuencias en el proceso de composición e interpretación musical al tiempo que interpretan otro instrumento que les demande la utilización de las manos.

1.5.2 Limitaciones

- Debido a la complejidad del proyecto se limitara inicialmente el secuenciador a una sola voz (monofónico).

- El número de pasos estará limitado a 8.

- Se requerirá asesoría de diseñadores para la elaboración de la interfaz física del secuenciador

2. MARCO DE REFERENCIA

2.1 MARCO TEORICO

2.1.1 Secuenciadores

Los secuenciadores son dispositivos electrónicos o aplicaciones informáticas que permiten programar y reproducir eventos musicales de forma secuencial mediante una interfaz de control conectada a uno o más instrumentos musicales electrónicos. Son herramientas de gran importancia en la programación y control de equipos de instrumentación electrónica musical (sintetizadores, samplers, cajas de ritmo, procesadores de señal, etc.). Existen dos tipos de secuenciador en hardware: Análogos y digitales.

El secuenciador análogo más común es básicamente una matriz de potenciómetros. Las columnas corresponden al período de tiempo durante el cual los potenciómetros generan el control del voltaje y las filas corresponden al número de tensiones de control independiente que se generen. Un secuenciador de 16 por 3 deberá tener 16 tiempos de llegada y 3 salidas de control de voltaje. Sumado a que cada intervalo de tiempo tiene un disparador de salida. Esto hace que el dispositivo se limite en control de parámetros como el tiempo y poco rango de interpretación.

Con respecto a los secuenciadores análogos, Alfred Mayer señala en su artículo A New Generation Of Musical Sequencers: *"En la música electrónica, hemos tenido sintetizadores análogos desde los inicios del arte. Estos eran incómodos, costosos, tomaba mucho tiempo programarlos y eran muy limitados en su largo y posibilidades de manipulación"* (1973,1).

Con la aparición de los secuenciadores digitales muchas de estas desventajas fueron suplidas, tal como lo señala Peter Hellen en su artículo A Micro Processor Based Sequencer for Voltage Controlled Electronic Music Synthesizers: *"Los secuenciadores digitales basan su funcionamiento en microcomputadores, lo cual gracias a este dispositivo simplifican gran parte de los procesos y solucionando los problemas del secuenciador análogo, ya que con una computadora, el largo de la secuencia, puede ser virtualmente ilimitada, necesitando solo memoria adicional para extender su capacidad, haciendo los problemas de programación más simples"* (1977, 2).

Existe una definida separación entre secuenciadores análogos y digitales. El secuenciador digital es la opción más común para aplicaciones que requieren el almacenamiento de grandes cantidades de información para la reproducción de secuencias. Por otro lado, el secuenciador análogo es la opción más común para aplicaciones en las que, relativamente pequeñas cantidades de distintos niveles de voltaje serán repetidos secuencialmente, como por ejemplo, ritmos o estructuras quasi-rítmicas.

2.1.2 Microcontroladores

El corazón del funcionamiento de los secuenciadores digitales esta en los Microprocesadores; *"El microprocesador es un circuito integrado que incorpora en su interior, una unidad central de proceso (CPU) y todo un conjunto compuertas lógicas que permiten enlazar otros dispositivos como memorias, y puertos de entrada y salida (I/O). Formando un sistema completo para cumplir con una aplicación específica dentro del mundo real. Actualmente las marcas más importantes son Intel y Picc….La aplicación más importante de los microprocesadores que cambió totalmente la forma de trabajar, ha sido la computadora personal o microcomputadora. La cual incorpora los circuitos de entrada y salida, al igual que de memoria, estando estas en un solo chip se denominan microcontroladores".* (Ramirez, 1986)

Esto hace del microcontrolador un computador integrado dentro de un chip, ya que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S.

La CPU es considerada el cerebro del microcontrolador, trae las instrucciones del programa almacenadas en la memoria, las interpreta (decodifica) y hace que se ejecuten. Dentro de ella se encuentran circuitos ALU que realizan las operaciones lógicas y aritméticas con los datos binarios.

2.1.3 Memorias RAM y ROM

La memoria RAM (Random Acces Memory), es una memoria de lectura y escritura, como por ejemplo, las variables utilizadas en el programa. La memoria ROM (Read Only Memory) es de solo lectura y en ella son memorizadas las instrucciones del programa a seguir de forma permanente.

2.1.4 MIDI

El modo de comunicación de un secuenciador digital con un módulo de sonido es el protocolo MIDI (Musical Instrument Digital Interface). Como su nombre lo indica, es una interfaz digital aplicada a instrumentos musicales, por lo tanto, "*es un lenguaje de comunicación que permite de la comunicación de múltiples instrumentos electrónicos de hardware y software, controladores de interpretación, computadoras y otros dispositivos relacionados a través de una conexión de red*". (Miles, 2007).

A nivel físico, la norma exige el uso de un determinado cable y determinados conectores en los aparatos. Así como cierta circuitería electrónica y una velocidad de transmisión concreta; A nivel lógico, se establecen una serie de mensajes en código digital claramente estructurados y definidos, que son los que se envían a traves del cable. La arquitectura de los mensajes está hecha en función del uso que luego van a tener, el cual es en principio y fundamentalmente, el musical.

Según Miles, el protocolo MIDI fue diseñado para trabajar en la ejecución de instrumentos musicales, su sistema de comunicación es de tipo digital y da por resultado que se pueda establecer una rápida y eficiente comunicación entre los instrumentos.

MIDI es un protocolo de comunicación serie asíncrono sin control de paridad, lo que implica la existencia de 2 bits extra: un bit Start (valor 0) y un bit Stop (valor 1) añadidos al comienzo y final, respectivamente, de cada byte MIDI propiamente dicho. Una palabra MIDI, son pues, 10 bits en total. MIDI se transmite a una velocidad de 31,25 Kbaudios (±1%) (31.250 bits/sg.) (1 bit cada 32 micro sg.) Resultando una banda pasante de 600 KHz.

MIDI usa un optoacoplador (que ha de suministrar, al menos, 5 mA. y su velocidad de respuesta debe ser mejor que 2 µs.) en la entrada de cada aparato para evitar bucles de masa y parásitos de 50 Hz provenientes de la red de alimentación, el optoacoplador también permite aislar eléctricamente un aparato de otro. El uso de un optoacoplador hace que la transmisión sea por lazo de corriente, cada receptor requiere cierto consumo para activar el led del optoacoplador que tiene en la entrada.

Los mensajes MIDI viajan en pequeños cluster llamados packets; constituidos de un byte de estado, el cual describe el tipo de información que está siendo enviada con el canal de información seguido por uno o más bytes de datos, cuando se requiere información adicional, estos proveen lo detalles de la información tales como nota o velocidad.

“Existen básicamente dos categorías de mensaje: mensajes de canal y mensajes de sistema; Los mensajes de canal son transmitidos y recibidos en un canal MIDI específico, los mensajes de sistema no están restringidos a un canal MIDI específico, sino que son transmitidos a todo el equipo que está conectado a un sistema MIDI. Este es un modo eficiente para transmitir información pertinente a muchos instrumentos a la vez”. (Miles, 2007)

Del protocolo MIDI la mayoría de las acciones tienen lugar en los mensajes de canal de voz, estos son los mensajes que llevan la información sobre la ejecución y son los siguientes:

- **Note on:** 1 byte de estado (numero de canal) y 2 de datos (nota y velocidad).
- **Note off:** 1 byte de estado (numero de canal) y 2 de datos (nota y velocidad).
- **Aftertouch:** 1 byte de estado (numero de canal) y 1 de datos (valor de presión).
- **Pitch bend:** 1 byte de estado (numero de canal) y 2 de datos (valores de información).
- **Cambios de control:** 1 byte de estado (numero de canal) y 2 de datos (numero y valor del control).

El lenguaje MIDI va de la mano con el lenguaje musical, ya que el propósito del protocolo funciona para poder digitalizar información proveniente de la música (instrumentos electrónicos), por lo tanto, para un buen manejo musical de un secuenciador digital y por ende un buen manejo de protocolo MIDI, se deben conocer los principales fundamentos del lenguaje musical.

2.1.5 Lenguaje Musical y Escalas

En música, al emitirse dos o más sonidos simultáneos se dice que se produce un "**acorde**", este puede ser "**consonante**" o "**disonante**" según que la sensación experimentada, cuando la sensación agradable es producida por una sucesión de sonidos entonces se tiene una "melodía", la experiencia enseña que la sensación producida no depende de los valores absolutos de las frecuencias de los sonidos, sino de la relación entre ellas, es decir, del intervalo (cociente de las frecuencias, tomando siempre como numerador la mayor frecuencia), siendo esta sensación tanto más agradable cuanto más sencillo sea el intervalo entre los dos sonidos.

Como vemos, la melodía consiste en la elección y número de notas que componen un período musical, por ejemplo en las obras de tipo orquestal, la melodía es interpretada

por el solista, siendo acompañado por el resto de la orquesta que proporciona la armonía.

La escala actual (escala occidental) es el resultado de un largo proceso de aprendizaje de las notas. Los pitagóricos construyeron un aparato llamado monocordio que se componía de una tabla, una cuerda tensa y una tabla más pequeña que se iba moviendo por la grande. Los pitagóricos observaron que haciendo más o menos larga la cuerda (moviendo la tabla móvil) se producían sonidos diferentes. Entre estos sonidos escogieron algunos que eran armoniosos con el sonido original (cuerda entera).

En la música es muy importante la relación que existe entre la frecuencia de los distintos sonidos, a esta relación se le llama **intervalo**. Los intervalos musicales pueden medirse en términos de la relación de frecuencias de los sonidos, aunque en música reciben nombres propios cuya correspondencia física depende del tipo de escala utilizada.

Los más importantes, por su simplicidad y su importancia a la hora de construir la escala musical, son:

La octava*:* Cuando la cuerda medía un medio del total, el sonido se repetía, pero más agudo. La octava es lo que correspondería a un salto de ocho teclas blancas del piano; o mejor dicho, una octava es la repetición de un sonido con una cuerda con la mitad de longitud, por tanto, otra nota armoniosa. Su frecuencia es doble.

La quinta: Es otro intervalo entre notas que se obtiene con una cuerda de largura dos tercios de la inicial, su frecuencia es de tres medios del sonido inicial. Corresponde a un salto de cinco teclas blancas en un piano.

La cuarta: Es, como las anteriores, otro intervalo entre notas que se obtiene con una cuerda de largura tres cuartos de la inicial. Su frecuencia es cuatro tercios de la nota inicial.

Así, a partir de un sonido original obtenemos diferentes notas armoniosas.

Nota	*Frecuencia*	*Long. cuerda*
Original	*F*	*L*
Octava justa	*2f*	*1/2L*
Quinta mayor	*3/2f*	*2/3L*
Cuarta justa	*4/3f*	*3/4L*
Tercera mayor	5/4f	4/5f
Tercera menor	6/5f	5/6f

Tabla 1. Tabla de notas según su frecuencia y longitud de cuerda.

La nota original o donde se comienza a derivar la escala se le llama tónica y a partir de estas frecuencias se logran las escalas desde cualquier nota musical occidental (do, re, mi, fa, sol, la y si y sus respectivos bemoles y sostenidos).

2.1.5.1 Escalas

Según John Powell en su libro “How music Works”: “*Las escalas se basan en una serie de intervalos que son divisiones de un intervalo natural, la octava”* (Powell, 2010).La música occidental, divide la octava en doce intervalos iguales, normalmente se utilizan conjuntos de siete notas, estos conjuntos, componen los modos mayor y menor que constituyen casi cualquier pieza musical occidental que hayamos oído.

Los aspectos principales con respecto a las escalas son:

1. las escalas permiten memorizar melodías.
2. Las escalas dividen el intervalo de una octava en intervalos más pequeños, con lo que proporcionan un repertorio de notas.

Escala diatónica:

El modelo de escala diatónica es el más conocido y el más natural audio perceptivamente, este modelo se evidencia esquemáticamente, con el patrón que muestran las teclas blancas del piano saltando las teclas negras, por ejemplo: siguiendo la secuencia *do—re—mi~fa—sol—la—si~do*.

En la práctica común de la música clásica se simplifican los tipos de escalas diatónicas reduciéndolos a dos variantes o modos: el mayor (escala diatónica mayor) y el menor (escala diatónica menor). Los intervalos de segunda menor separados por un semitono (mi-fa y si-do) y los intervalos de segunda mayor separados por tonos completos (do-re, re-mi, fa-sol, sol-la, la-si). Esta escala tiene siete intervalos por octava, siendo la octava nota la repetición de la primera pero una octava más arriba. En un piano las

teclas blancas corresponden a la escala diatónica de "do", y en donde C-D-E-F-G-A-B es la notación alternativa a Do-Re-Mi-Fa-Sol-La-Si.

Escala pentatónica:

Está constituida por una sucesión de cinco sonidos diferentes dentro de una octava que no está separada por semitonos. La escala pentatónica se puede generar a partir de cualquiera de los doce tonos. Basándose en una nota principal (tónica) los tonos de la escala serán los siguientes:

- Tónica (1)
- Segunda mayor (2)
- Tercera mayor (3)
- Quinta justa (5)
- Sexta mayor (6)

Por ejemplo, si nos basamos en la nota *do*, los tonos de la escala serán: *do, re, mi, sol, la*.

2.1.6 Síntesis electrónica del sonido

2.1.6.1 Síntesis

Mark Russ en su libro “Síntesis y Muestreo” define Síntesis como: *“El proceso para la generación de un sonido, ya sea con el reusó y procesamiento de sonidos existentes, generándolos electrónica y mecánicamente, utilizando matemáticas, física o hasta biología; todo con el fin de traer arte y ciencia en un mezcla de destreza musical y conocimientos técnicos.”*[1]

2.1.6.2 Osciladores y formas de onda.

Un oscilador es un circuito capaz de generar a su salida una forma de onda estable, periódica y con una frecuencia determinada, por lo general, la frecuencia de oscilación es proporcional a un voltaje de entrada al circuito. Las formas de onda más comunes generadas por un oscilador son las siguientes:

[1] Russ, Mark; Síntesis y muestreo; 2009.

- **Sinusoidal:** Forma de onda simple también llamada “tono puro”, carece totalmente de contenido armónico y no se encuentra en la naturaleza, existe únicamente a través de medios electrónicos.

- **Diente de sierra:** Cuenta con un gran contenido armónico y su amplitud viene dada por la expresión 1/numero de armónicos.

- **Cuadrada:** Tiene gran riqueza en contenido armónico y de comportamiento impar; la amplitud de cada armónico es 1/numero de armónicos.
- **Triangular:** Su contenido armónico es relativamente pobre, cuenta únicamente con armónicos impares y la amplitud de estos decrece exponencialmente.
- **Ruido aleatorio:**

1. Blanco: Señal aleatoria que precisa una distribución uniforme de la energía sobre el espectro de frecuencias.
2. Rosa: Señal cuya densidad espectral es proporcional al recíproco de su frecuencia, su contenido de energía por frecuencia disminuye en 3dB por octava. Por lo anterior, cada banda de frecuencias (en octavas) contiene la misma energía total.

2.1.6.3 Envolventes y filtros

Envolvente es una señal que da forma a otra señal a lo largo del tiempo, la clase de envolvente más utilizado es el ADSR (Attack, Decay, Sustain y Release).

Un filtro es un dispositivo que amplifica ciertas frecuencias y atenúa otras, está diseñado para separar, pasar ó suprimir determinado parámetro a partir de una señal.

2.1.6.3.1 Tipos de filtros

1. **Pasa bajos:** Sólo permite el paso de las frecuencias inferiores a la frecuencia de corte (**fc)** y atenúa todas las frecuencias superiores a ella.

2. **Pasa altos**: Sólo permite el paso de las frecuencias superiores a la frecuencia de corte (**fc)** y atenúa todas las frecuencias menores a ella.

3. **Pasa banda:** Permite el paso de un rango medio de frecuencias definido entre una frecuencia de corte inferior (**fci)** y una frecuencia de corte superior (**fcs)**.

4. **Rechaza banda:** Suprime el paso de un rango medio de frecuencias definidas por la frecuencia de corte superior (**fcs)** y la frecuencia de corte (**fci).** A la resta de estas dos frecuencias se llama ancho de banda (**Bw**).

Orden de filtros:

De acuerdo al orden del filtro, después de la frecuencia de corte existirá una pendiente en dBs por octava:

•**1er Orden:** -6dB por Octava / 45°de Desfase en Fc
•**2do Orden:** -12dB por Octava / 90°de Desfase en Fc
•**3er Orden:** -18dB por Octava / 135°de Desfase en Fc
•**4to Orden:** -24dB por Octava / 180°de Desfase en Fc

2.1.6.4 Síntesis aditiva

Síntesis basada en el teorema de Fourier, que dice que cualquier señal puede ser descrita perfectamente mediante una suma de senos o cosenos de diferente fase, amplitud y frecuencia. Cada sonido musical se caracteriza por una frecuencia fundamental y una serie de frecuencias denominadas armónicos, donde la frecuencia de cada armónico se define como un múltiplo de la frecuencia fundamental, la transformada de Fourier permite calcular a partir de una señal cualquiera, la magnitud y fase de cada una de las componentes frecuenciales que posee, mediante un análisis de Fourier se estudia la evolución del espectro de cualquier sonido en el tiempo, con esta información temporal se obtiene la envolvente de cada uno de estos armónicos, a partir de esta información, se sintetiza un nuevo sonido, sumando en cada instante todos los armónicos con sus respectivas amplitudes.

2.1.6.5 Síntesis sustractiva

Técnica que simplifica la descripción de una señal compleja utilizando señales con alto contenido armónico tales como ondas cuadradas, triangulares, diente de sierra ó ruidos, de manera que con un filtrado selectivo se puedan retirar armónicos a fin de obtener el sonido deseado.

2.1.6.6 Síntesis por sampling

Técnica mediante la cual se reproducen segmentos de audio digital de un instrumento real ó sintético que han sido preliminarmente capturados. Dichos segmentos son almacenados en memoria ROM (dentro de un ámbito digital) y reproducidos a diferentes velocidades según la altura de la nota que se desee generar.
Los segmentos de audio son utilizados por un sampler, dispositivo que permite muestrear digitalmente eventos sonoros para ser reproducidos posteriormente. Algunos de ellos, tienen la posibilidad de modificar estas muestras a través de procesos en tiempo y amplitud, los registros almacenados contienen segmentos en donde se establecen *loops* debidamente editados a fin de sostener el sonido por periodos largos de tiempo.

2.1.6.7 Síntesis Modular

La modulación consiste en cambiar algún parámetro de la ecuación de la onda en función de la amplitud de una segunda onda, estos parámetros son: la amplitud, la frecuencia y la fase.

Amplitud modular:

En la síntesis AM la amplitud de una onda modula la amplitud de otra. En términos generales y para ondas senoidales:

$f(t) = (A + M * \cos(wm* t)) * \cos(wc* t)$

•M = Amplitud de la Moduladora
•wm= Frecuencia Angular de la Moduladora
•wc= Frecuencia Angular de la Portadora)
•A = Componente de continua aplicada a la señal moduladora

Si A=0 se produce una multiplicación de ambas señales (a esto se denomina modulación en anillo ó RingModulation). Si A>0 se produce una modulación de amplitud normal de la señal del oscilador.

Frecuencia modular:

En FM, la frecuencia instantánea de una onda portadora es variada de acuerdo a una onda moduladora. La cantidad de variación en la onda portadora cambia alrededor de un promedio conocido como la desviación de picos de frecuencia.

Los parámetros de una señal con características de frecuencia modulada son los siguientes:

- C: Frecuencia de la portadora o frecuencia promedio.
- M: Frecuencia moduladora.
- D: Desviación de picos de frecuencia.

Tenemos por tanto, para señales senoidales:

f(t) = A * cos(wc* t + (I * cos(wm* t)))
A = Amplitud de la Portadora
wm= Frecuencia Angular de la Moduladora
wc= Frecuencia Angular de la Portadora)
I = Amplitud de la Moduladora (Índice de Modulación)
I = Δf/ Fm
Fm: frecuencia de modulación.

El número de frecuencias laterales que ocurren en FM está relacionado al índice de modulación, al incrementar de 0 en adelante, una cantidad proporcional a la energía de la portadora es distribuida a cada una de las bandas laterales: Fj= Fc±j ·Fm, donde j = (0, I+2).

2.1.7 Arduino

Arduino se define según sus desarrolladores como: “Una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos”.[2]

Arduino cuenta con diferentes placas, cada una con diferentes características de puertos, entradas y salidas análogas y digitales, según lo requiera el usuario, y se controla mediante su propio software libre basado en C.

[2] **www.arduino.cc/es**

2.1.7.1 Arduino mega

Manejada por el microcontrolador ATmega 1280[3], la placa arduino mega posee 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset.

2.1.8 Ámbitos de la producción musical

Con base en lo enunciado por Hillen en su artículo "A Microprocessor Based Sequencer For Voltage Controlled Electronic Music Syntesizers"[4] podemos definir tres ámbitos de la producción musical en los cuales se desempeñará comúnmente el secuenciador.

2.1.8.1 Desempeño creativo

Se refiere al uso del secuenciador en la instancia de la producción musical donde surgen por primera vez las ideas. Las posibilidades creativas ofrecidas por un instrumento son complicadas de evaluar, debido a que cada usuario posee distintas preferencias creativas. Una aproximación a determinar que tan eficiente será un instrumento en este ámbito se puede lograr evaluando aspectos como: facilidad y rapidez de uso, estabilidad, grado de satisfacción tras realizar un proceso creativo con el instrumento.

2.1.8.2 Desempeño de edición

Se refiere al uso del secuenciador en la etapa de la producción musical en que se toma las ideas creadas y se les da una forma final. Los aspectos mas relevantes para evaluar el desempeño de un instrumento es este ámbito son: opciones de edición brindada, estabilidad y capacidad de almacenamiento

2.1.8.3 Desempeño en vivo

Hace referencia al uso del secuenciador en la etapa interpretativa de secuencias almacenadas, generalmente ante un público y durante un tiempo preestablecido. Los aspectos mas relevantes para evaluar el desempeño de un instrumento es este ámbito son: facilidad y rapidez de uso, estabilidad y portabilidad.

[3] http://www.atmel.com/Images/doc2549.pdf

[4] Ver lo descrito en los antecedentes sobre este tema.

2.1.9 Supercollider

Supercollider es un lenguaje de programación para síntesis de audio en tiempo real, su licencia es libre y es compatible con OS X, Windows y Linux. El software fue desarrollado 1996 por James McCartney y desde entonces se ha convertido en un sistema usado por científicos y artistas para aplicaciones que involucran sonido.

Acorde a lo enunciado en el libro "The SuperCollider Book", publicado por el instituto MIT en el 2011 "*Supercollider es un eficiente y expresivo lenguaje de programación dinámico que provee un marco de trabajo para investigaciones acústicas, música de algoritmos y programación interactiva*"(Wilson, 2011). La última versión (3.5) fue lanzada en marzo del 2012.

2.1.8.1 Arquitectura

Supercollider se divide en dos componentes: un servidor (*scsynth*) y un cliente and (*sclang)*. Estas dos partes se comunican entre sí usando el protocolo OSC[5], este sistema de comunicación permite que el servidor también pueda ser controlado por otros sistemas que utilizan protocolos similares, como por ejemplo, Pure Data.

El lenguaje de programación combina la estructura orientada a objetos de Smalltalk y características de lenguajes de programación funcional con una sintaxis derivada del lenguaje C. El servidor por su parte, implementa un plugin C API que le permite crear algoritmos llamados UGens (Unit Generators), unidades que forman los bloques constructivos para el diseño de síntesis y procesamiento de señal

El lenguaje de SuperCollider también permite a los usuarios crear interfaces graficas de usuario (GUI[6]). Mediante la instalación de extensiones llamadas "Quarks"[7], nuevos marcos de trabajo pueden ser agregados al sistema expandiendo sus posibilidades graficas, de interacción y programación.

2.2 MARCO LEGAL O NORMATIVO

Para este proyecto, no existen normas o leyes que restrinjan el diseño de un secuenciador digital.

[5] OSC (Open Sound Control) es un protocolo de comunicación entre computadores, sintetizadores y otros dispositivos multimedia. Puede considerarse la versión moderna del protocolo MIDI.
[6] Graphical User Interface
[7] Una descripción detallada del proceso de instalación puede ser encontrada en http://quarks.sourceforge.net/

3. METODOLOGIA

3.1 Enfoque de la investigación

Empírico - Analítico.

3.2 Línea de investigación de USB / Sub – línea de facultad / campo temático del programa

Tecnologías actuales y sociedad – Procesamiento de señales - Diseño de sistemas de sonido.

3.3 Hipótesis

El diseño y construcción de un secuenciador con una interfaz hardware a modo de pedal y matriz de tonos, facilitará la creación, almacenamiento y reproducción de secuencias por parte de músicos que interpreten diversos instrumentos simultáneamente.

3.4 Variables

3.4.1 Variables Independientes

- Tipo de sistema de entrada y salida de datos.
- Tipo de lenguaje de programación y plataforma electrónica.
- Cantidad de operaciones necesarias para crear, almacenar y reproducir secuencias.
- Peso y tamaño del diseño.

3.4.2 Variables Dependientes

- Precisión y estabilidad del dispositivo en el envío y recepción de mensajes MIDI.
- Capacidad de interacción del secuenciador con software y hardware externo
- Capacidad del dispositivo para ser usado en el proceso creativo y de interpretación.

4. DESARROLLO INGENIERIL.

El sistema SUB! se divide en dos componentes principales: Secuenciador y sintetizador virtual. El proceso de diseño fue abordado de manera independiente para cada uno de los elementos y finalmente unidos en una etapa de ensamble

4.1 SECUENCIADOR.

El secuenciador es el dispositivo que permite al usuario crear, almacenar y reproducir secuencias en tiempo real. Las secuencias, que son almacenadas como eventos MIDI, pueden ser enviadas a un modulo de sonido externo a través del puerto "MIDI OUT".

El funcionamiento del dispositivo puede dividirse en tres modalidades: de creación o programación, de almacenamiento y de reproducción o lectura. A continuación se explica el diseño del software y hardware para cada una de estas modalidades, los mecanismos usados por el secuenciador para establecer comunicación con módulos externos y, finalmente, los aspectos relacionados con el diseño de la carcasa del instrumento.

4.1.1 Modo creación o programación

Esta es la modalidad inicial del secuenciador, es el estado en el que el dispositivo se encuentra cuando no existe nada almacenado en su memoria y el usuario va a comenzar el proceso creativo de programar. La herramienta más importante en esta instancia es la matriz de tonos.

4.1.1.1 Matriz de tonos

La matriz de tonos es el principal sistema de entrada de datos del secuenciador, permite al usuario programar de manera rápida y sencilla secuencias rítmicas o armónicas según parámetros de tonalidad, escala, octava y tempo.

La resolución de la matriz de tonos es de 8x8 (celdas x columnas), las columnas determinan los pasos que recorre el secuenciador, y las filas representan las posibles notas que puede seleccionar el usuario por cada paso. Al activar y desactivar casillas dentro de la matriz, el músico puede programar los patrones deseados.

La versatilidad y simplicidad de programar en la matriz del SUB! se ve expandida mediante la implementación de cuatro variables controladoras: Tonalidad, escala, octava y división de tempo. La configuración apropiada[8] de estos parámetros, hace que cualquier nota programada por el usuario sea rítmica y armónicamente coherente con otros elementos dentro de una pieza musical, de esta manera, el SUB! se convierte en una gran herramienta para improvisar o componer de manera fácil y rápida. A continuación se presenta una descripción detallada de las funciones mencionadas:

Tonalidad: Permite el cambio entre las diferentes tonalidades musicales, desde DO hasta SI. El control de tonalidad se maneja entre 12 instancias:

1. Escala de DO mayor y su relativa LA menor (C - Am).
2. Escala de DO# mayor y su relativa SIb menor (C# - Bbm).
3. Escala de RE mayor y su relativa SI menor (D - Bm).
4. Escala de MIb mayor y su relativa DO menor (Eb - C).
5. Escala de MI mayor y su relativa DO# menor (E - C#m).
6. Escala de FA mayor y su relativa RE menor (F - Dm).
7. Escala de FA# mayor y su relativa MIb menor (F# - Ebm).
8. Escala de SOL mayor y su relativa MI menor (G - Em).
9. Escala de SOL# mayor y su relativa FA menor (G# - Fm).
10. Escala de LA mayor y su relativa FA# menor (A - F#m).
11. Escala de SIb mayor y su relativa SOL menor (Bb - Gm).
12. Escala de SI mayor y su relativa SOL# menor (B - G#m).

Escalas: Ofrece la posibilidad de usar 3 escalas musicales distintas dentro de cada tonalidad previamente descrita. Las posibles escalas a usar son las siguientes:

1. Escala diatónica.
2. Escala pentatónica.
3. Escala árabe.

[8] El término “apropiado” hace referencia a seleccionar una tonalidad y escala armónicamente coherentes con el resto de sonidos dentro de una composición musical.

Las siguientes tablas muestran la relación de notas adoptadas por cada casilla de la matriz según la escala seleccionada:

Escala diatónica:

Primera	Primera	Primera	Primera	Primera	Primera	Primera	Primera
Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor
Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor
Cuarta Justa	Cuarta Justa	Cuarta Justa	Cuarta Justa	Cuarta Justa	Cuarta Justa	Cuarta Justa	Cuarta Justa
Quinta Justa	Quinta Justa	Quinta Justa	Quinta Justa	Quinta Justa	Quinta Justa	Quinta Justa	Quinta Justa
Sexta Mayor	Sexta Mayor	Sexta Mayor	Sexta Mayor	Sexta Mayor	Sexta Mayor	Sexta Mayor	Sexta Mayor
Séptima Mayor	Séptima Mayor	Séptima Mayor	Séptima Mayor	Séptima Mayor	Séptima Mayor	Séptima Mayor	Séptima Mayor
Octava Justa	Octava Justa	Octava Justa	Octava Justa	Octava Justa	Octava Justa	Octava Justa	Octava Justa

Tabla 2. Valores de notas para la escala diatónica dentro de una matriz de 8 por 8

Escala pentatónica:

Primera	Primera	Primera	Primera	Primera	Primera	Primera	Primera
Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor	Segunda Mayor
Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor	Tercera Mayor
Quinta justa	Quinta justa	Quinta justa	Quinta justa	Quinta justa	Quinta justa	Quinta justa	Quinta justa
Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor
Octava justa	Octava justa	Octava justa	Octava justa	Octava justa	Octava justa	Octava justa	Octava justa
Segunda mayor +(1oct)	Segunda mayor +(1oct)	Segunda mayor +(1oct)	Segunda mayor +(1oct)	Segunda mayor +(1oct)	Segunda mayor +(1oct)	Segunda mayor +(1oct)	Segunda mayor +(1oct)
Tercera mayor + (1oct)	Tercera mayor + (1oct)	Tercera mayor + (1oct)	Tercera mayor + (1oct)	Tercera mayor + (1oct)	Tercera mayor + (1oct)	Tercera mayor + (1oct)	Tercera mayor + (1oct)

Tabla 3. Valores de notas para la escala pentatónica dentro de una matriz de 8 por 8.

Escala árabe:

Primera	Primera	Primera	Primera	Primera	Primera	Primera	Primera
Segunda Menor	Segunda Menor	Segunda Menor	Segunda Menor	Segunda Menor	Segunda Menor	Segunda Menor	Segunda Menor
Cuarta justa	Cuarta justa	Cuarta justa	Cuarta justa	Cuarta justa	Cuarta justa	Cuarta justa	Cuarta justa
Disminuida	Quinta disminuida	Quinta disminuida	Quinta disminuida	Quinta disminuida	Quinta disminuida	Quinta disminuida	Quinta disminuida
Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor	Sexta mayor
Séptima menor	Séptima menor	Séptima menor	Séptima menor	Séptima menor	Séptima menor	Séptima menor	Séptima menor
Octava justa	Octava justa	Octava justa	Octava justa	Octava justa	Octava justa	Octava justa	Octava justa
Segunda menor + (1oct)	Segunda menor + (1oct)	Segunda menor + (1oct)	Segunda menor + (1oct)	Segunda menor + (1oct)	Segunda menor + (1oct)	Segunda menor + (1oct)	Segunda menor + (1oct)

Tabla 4. Valores de notas para la escala árabe dentro de una matriz de 8 por 8.

3. Octavas: Esta variable permite el cambio de octavas dentro de las tonalidades en un rango de 4 octavas, desde C1 (Do octava 1)) hasta C5 (Do octava 5).

4. División de tempo: Esta variable permite el cambio de tempo entre los pasos de la matriz de tonos, donde dependiendo del tempo dictaminado por el controlador maestro, cambia la velocidad de recorrido entre los 8 pasos. Este parámetro se maneja en 3 divisiones:

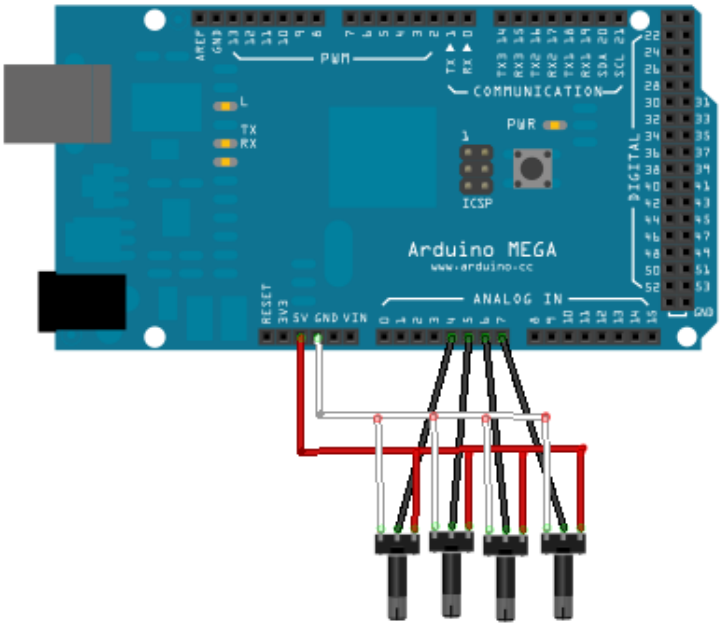
1. De ⅛ (8 pasos por compás): Donde el recorrido de los 8 pasos de la matriz transcurre dentro de un 1 compás de tempo.

2. De ¼ (4 pasos por compás): Donde el recorrido de los 8 pasos de la matriz transcurre dentro de 2 compases de tempo.

3. De 1/16 (16 pasos por compás): Donde el recorrido de los 8 pasos de la matriz transcurre dentro de medio compás de tempo, resultando así, dos recorridos por compás.

Las variables mencionadas son controladas por 4 potenciómetros conectados al Arduino Mega.

Fig. 1. Diagrama de conexión de 4 potenciómetros a la tarjeta arduino mega.



Fuente: Software Fritzing.

El Hardware de la matriz de tonos está constituido principalmente por dos elementos, una pantalla LCD grafica y un panel táctil:

Pantalla LCD grafica: Esta pantalla hace la representación grafica de la matriz de tonos. La referencia utilizada fue una LM128240T de Topway[9] comandada por un chip toshiba 6963 con un tamaño de 240x128 pixeles (esta es la LCD grafica con sistema touch mas grande en el mercado colombiano).

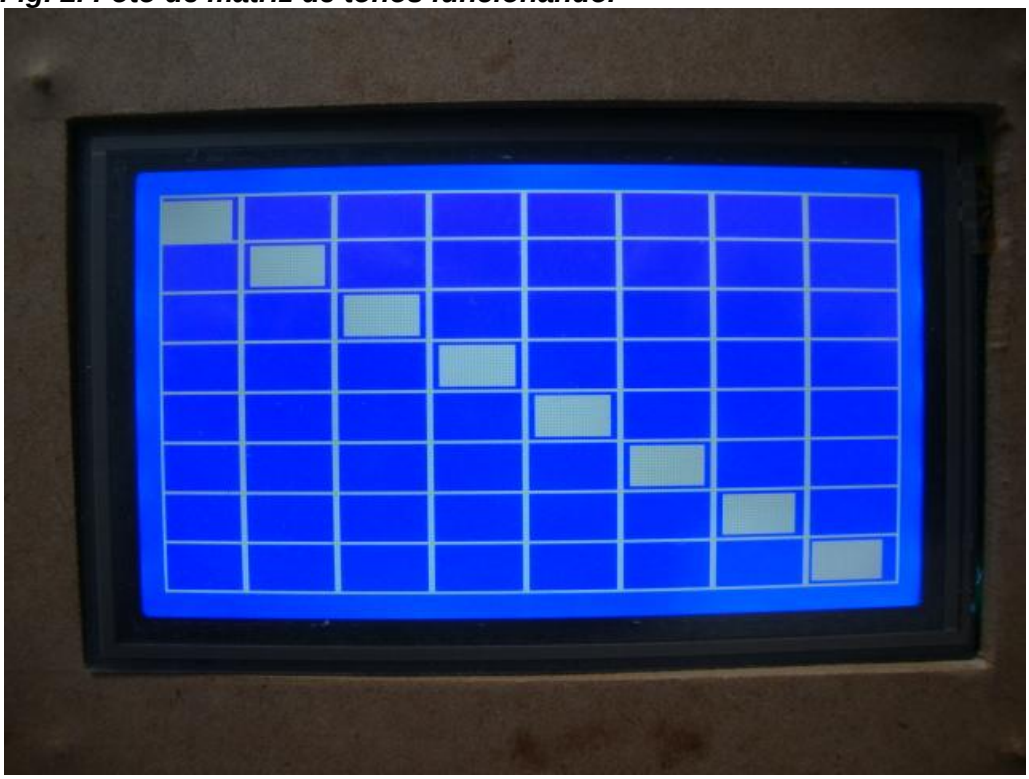
La LCD fue programada de tal manera que constantemente muestra una tabla de 8 filas y 8 columnas, al ser activada una casilla, rellena la cuadrilla seleccionada para que el usuario pueda reconocer que casillas están activadas y desactivadas. Para el control del software de la pantalla grafica en el lenguaje Arduino se utiliza la librería t6963[10], la cual optimiza la interacción entre software y hardware del sistema.

[9] http://www.ic-on-line.cn/view_online.php?id=1557632&file=0280\lm240128tfc_541881.pdf
[10] http://code.google.com/p/arduino-t6963c/

Panel Táctil o Touch: Con el mismo tamaño de la pantalla grafica, y colocado justo encima de esta, el panel táctil es un dispositivo sensitivo de dos dimensiones construido por 2 laminas ligeramente separadas por huecos de aire, cuando es presionada, las dos superficies resistivas se encuentran y se lee la posición donde estas se encontraron mediante un circuito controlador.

El panel táctil es utilizado para que el usuario pueda interactuar con la matriz de tonos, cuando el usuario presiona un punto correspondiente a alguna casilla de la cuadrilla, se envía en la programación una orden a la pantalla grafica de rellenar o vaciar ese espacio (según la condición previa en que se encuentre). Para la programación y conexionado del panel se utilizo como guía el data sheet utilizado para una pantalla de nintendo DS, fabricado por la empresa Sparkfun.[11]

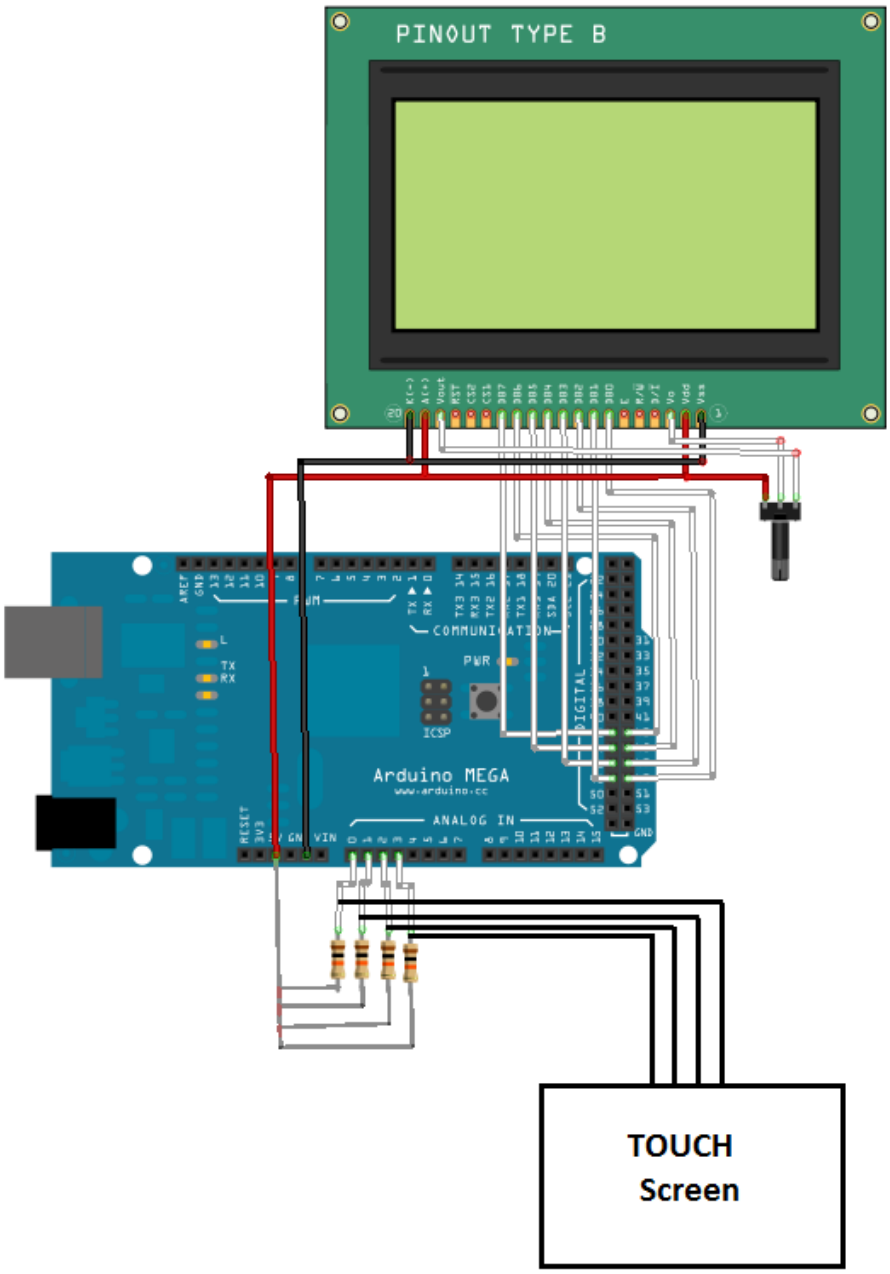
Fig. 2. Foto de matriz de tonos funcionando.



Fuente: Propia.

[11] http://www.sparkfun.com/datasheets/LCD/HOW%20DOES%20IT%20WORK.pdf

Fig. 3. Diagrama de conexión para la pantalla LCD grafica y el panel táctil a la tarjeta arduino mega.



Fuente: Software Fritzing.

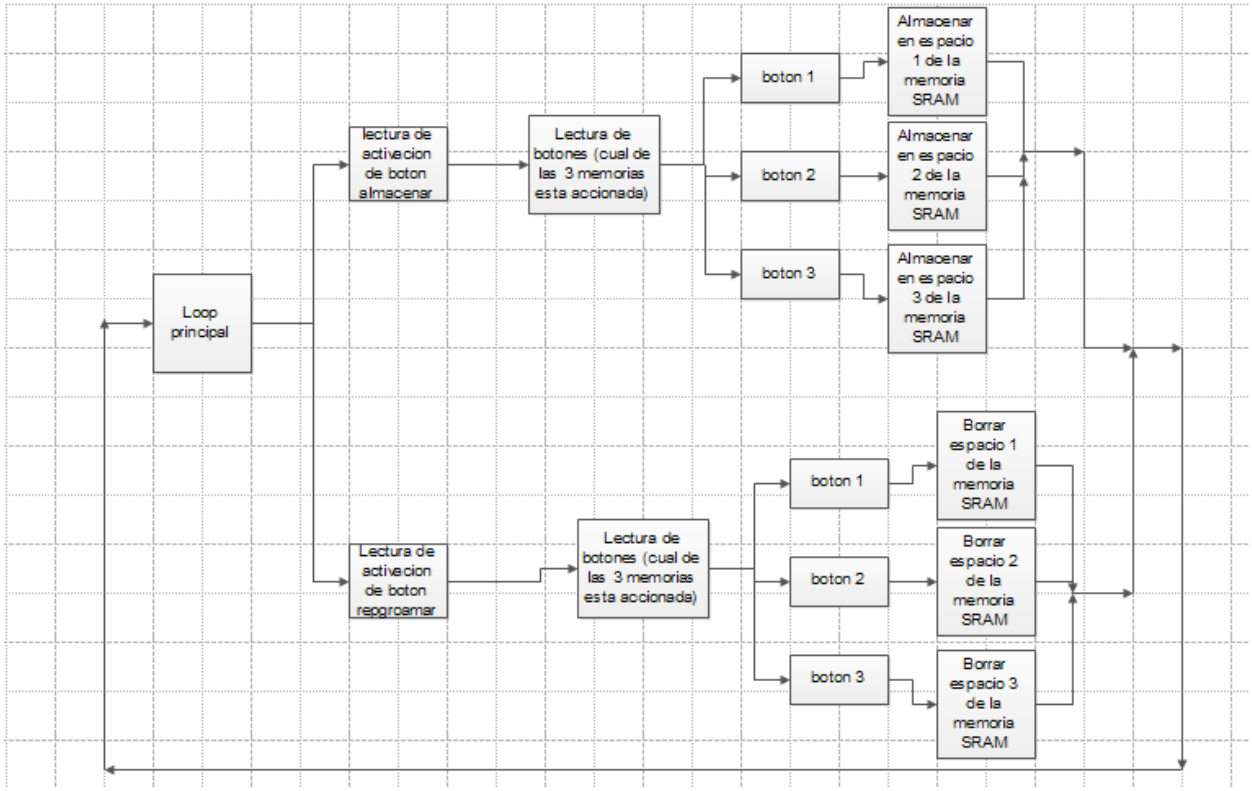
4.1.2 Modo de almacenamiento

Esta es la modalidad en la que el secuenciador maneja el almacenamiento de los patrones programados, la memoria almacena los valores activados en la matriz de tonos que luego pueden ser llamados en el modo de reproducción.

Para el modo de almacenamiento, el sistema utiliza dos simples comandos, un botón pulsador de almacenamiento y un botón pulsador de reprogramación, ambos se manejan dentro de 3 distintos espacios de memoria los cuales son accionados por 3 botones pulsadores ubicados simultáneamente en la interfaz principal del dispositivo y en la pedalera. Los patrones almacenados son guardados en la memoria SRAM de la tarjeta arduino mega, la cual cuenta con una capacidad de hasta 8kb, brindando espacio suficiente para almacenar la cantidad de información deseada.

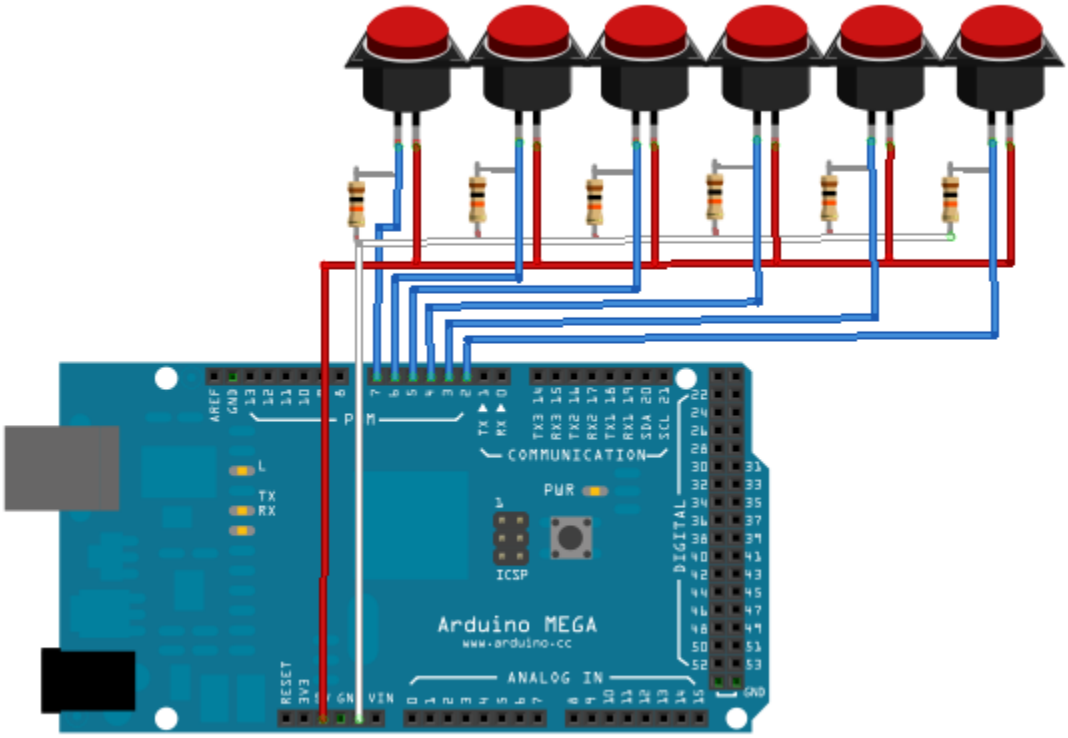
La programación para el modo de almacenamiento previamente descrito se rige por el siguiente diagrama de flujo:

Fig. 4. Diagrama de flujo para el modo de almacenamiento



Fuente: Software Edraw.

Fig. 5. Diagrama de conexión para los 6 botones a la tarjeta arduino mega.



Fuente: Software Fritzing.

4.1.3 Modo de reproducción o lectura

En esta instancia el secuenciador posee información almacenada en la memoria y el usuario se dispone a reproducirla.

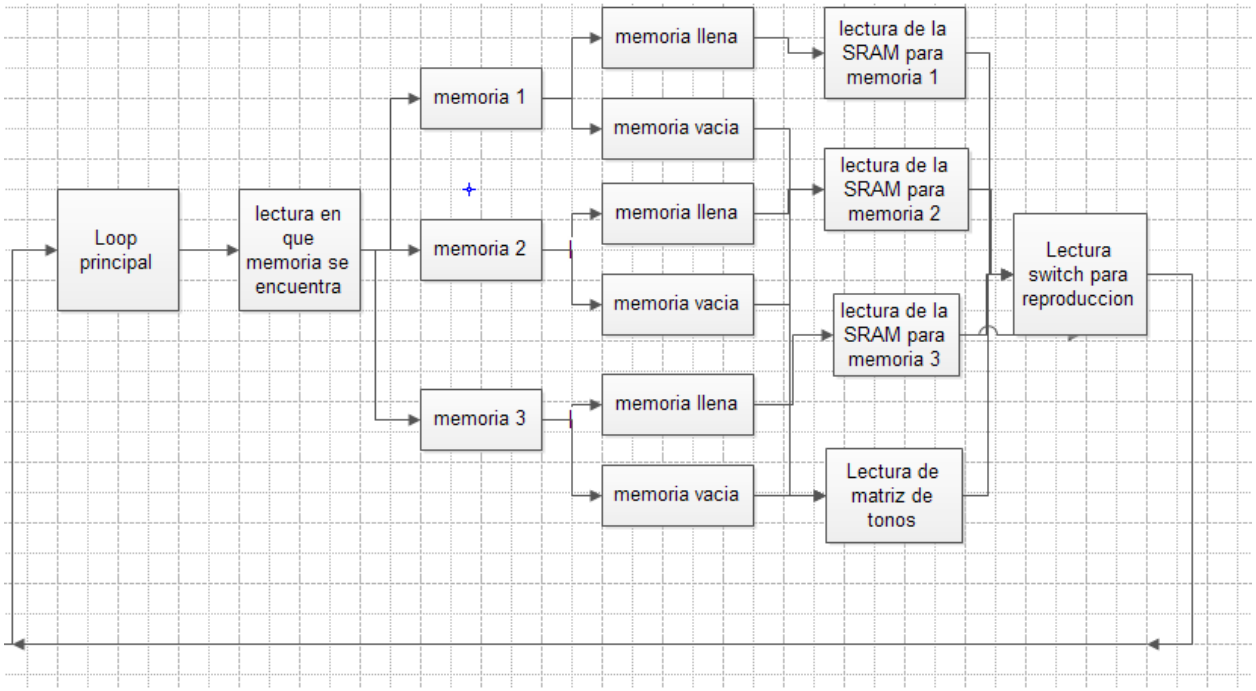
La reproducción empieza bajo el mando del controlador maestro, el cual es el encargado de dar la orden para empezar la lectura de los patrones, seguidamente, el sistema se sitúa en uno de los 3 espacios de memoria según lo seleccionado por el usuario, si en este espacio no se encuentra ningún patrón almacenado, inmediatamente empezará a reproducir el patrón que se esté programado en la matriz de tonos, en el caso de haber algo almacenado, el sistema reproducirá el patrón almacenado en dicha memoria.

El sistema implementa un interruptor que controla la reproducción en general, si este se encuentra activado, el sistema enviará los mensajes MIDI de activación de nota que

llevará a la reproducción, si se encuentra desactivado, hará todo el proceso de lectura pero no hará envío de mensajes MIDI y por lo tanto no habrá reproducción.

La programación para el modo de reproducción se rige por el siguiente diagrama de flujo:

Fig. 6. Diagrama de flujo para el modo de reproducción.



Fuente: Software Edraw.

4.1.4 Recepción y envío de mensajes MIDI

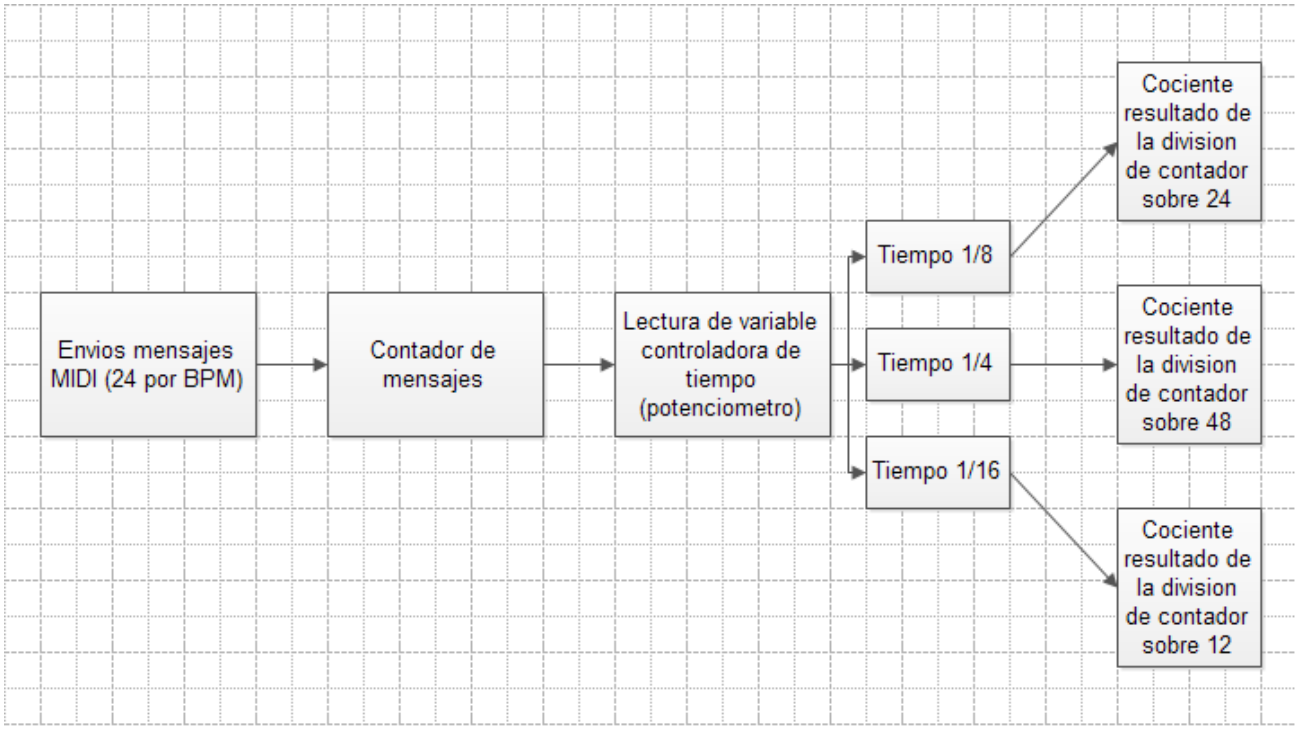
4.1.4.1 Recepción de mensajes (MIDI in).

La recepción MIDI es una de las partes más importantes en el funcionamiento del SUB!, los mensajes de reloj externos dan el pie de partida para la reproducción de sonidos generada por el sistema. Mediante el puerto de entrada serial de la tarjeta arduino RX, el secuenciador recibe los mensajes MIDI de arranque (START), parada (STOP), continuación (CONTINUE) y de reloj provenientes del controlador maestro.

Durante la sincronía, el reloj del controlador maestro envía 24 mensajes por beat. Para la reproducción de notas, se utiliza en la programación una variable contadora la cual es dividida a razón de la variable controladora de tiempo de la matriz de tonos, utilizando los cocientes resultantes de dichas divisiones, se controla la división bpm de acuerdo a los 24 mensajes enviados por el reloj del controlador maestro.

La programación para sincronía de reloj se rige por el siguiente flujo:

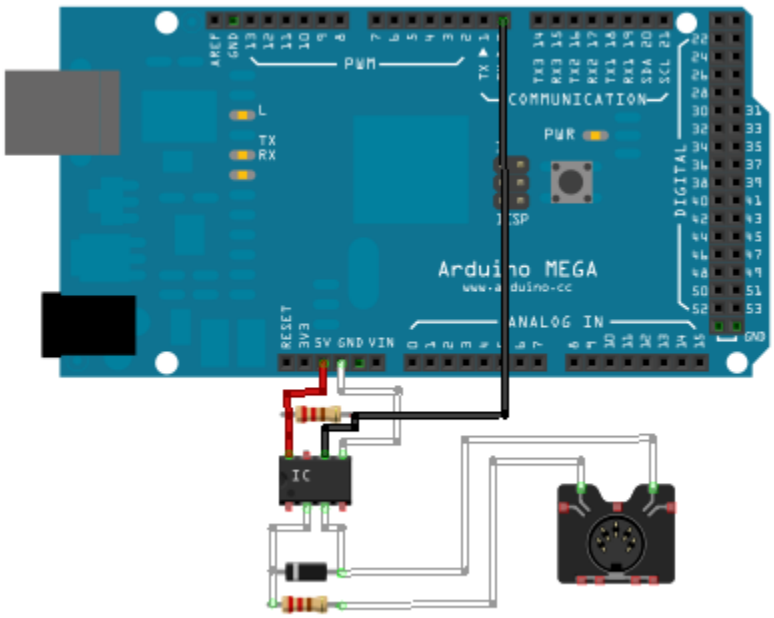
Fig. 7. Diagrama de flujo para la sincronía de tempo Midi.



Fuente: Software Edraw.

Siguiendo las indicaciones provistas por la página de Arduino[12] , se utiliza el siguiente conexionado para la entrada de datos MIDI utilizando un octocoplador 6n138 y un diodo 1n914 para las salidas al MIDI jack:

Fig. 8. Diagrama de conexión para la recepción de mensajes midi a la tarjeta arduino mega.



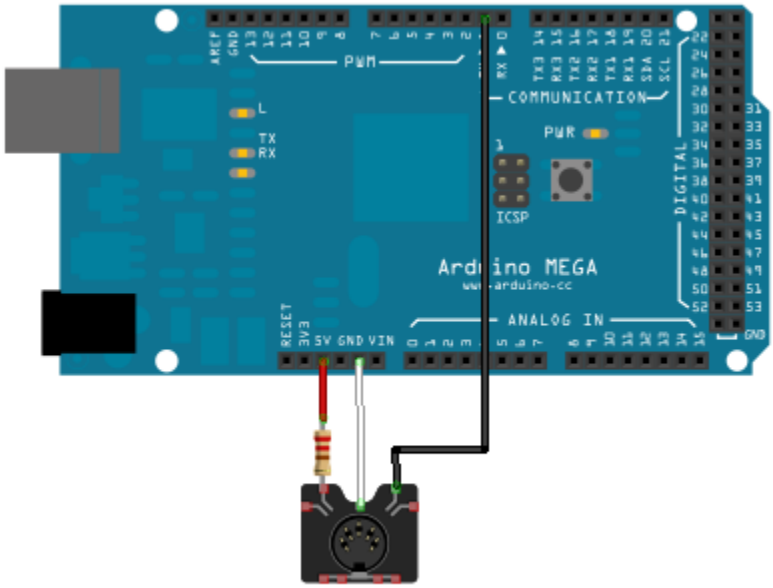
Fuente: Software Fritzing.

4.1.4.2 Envió de mensajes (MIDI OUT):

La salida MIDI es parte esencial del SUB!, por medio del puerto serial TX se envían los mensajes de activación de nota al modulo externo encargado de convertir la información MIDI en sonidos.

[12] http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1187962258

Fig. 9. Diagrama de conexión para el envío de señales midi a la tarjeta arduino mega.



Fuente: Software Fritzing.

4.1.5 Pedalera e Interfaz.

4.1.5.1 Pedalera.

La pedalera es un dispositivo externo al secuenciador diseñado de manera auxiliar para que el usuario pueda interactuar con el sistema sin necesidad de utilizar las manos. Contiene 3 botones pulsadores que permiten acceder a las 3 memorias del SUB!, además contiene un interruptor que activa o desactiva el envío de mensajes MIDI para la reproducción. La pedalera contiene también 3 leds que encienden y apagan de acuerdo al espacio de memoria seleccionado, permitiendo al usuario saber en qué memoria se está trabajando sin tener que ver a la caja principal.

4.1.5.2. Pantalla LCD alfanumérica.

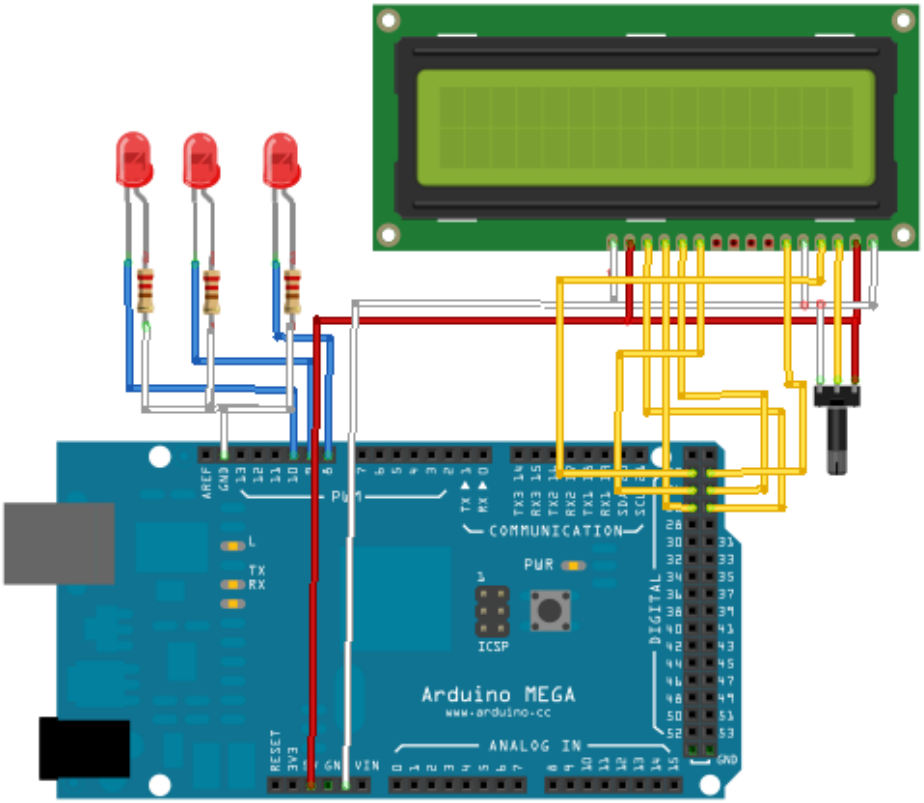
El sistema implementa una pantalla LCD alfanumérica donde se muestra los valores de las variables controladoras de la matriz de tonos y en qué espacio de memoria se encuentra el usuario. Los valores mostrados por la pantalla son los siguientes:
Memo: E un diminutivo de memoria, su valor (entre 1-3) nos indica en que memoria se encuentra ubicado el sistema.

Esc: Es un diminutivo de escala, su valor (entre 1-3) indica en qué escala se encuentra el sistema.

Time: Time significa tiempo en inglés, su valor (1/8, 1/16 o 1/4) indica la división de tiempo seleccionada.

Ton: Es un diminutivo de tonalidad, su valor son notas en lenguaje de cifrado mostrando la nota de tonalidad sobre la cual se está trabajando.

Fig. 10. Diagrama de conexión de pantalla LCD alfanumérica y 3 leds a la tarjeta arduino mega.



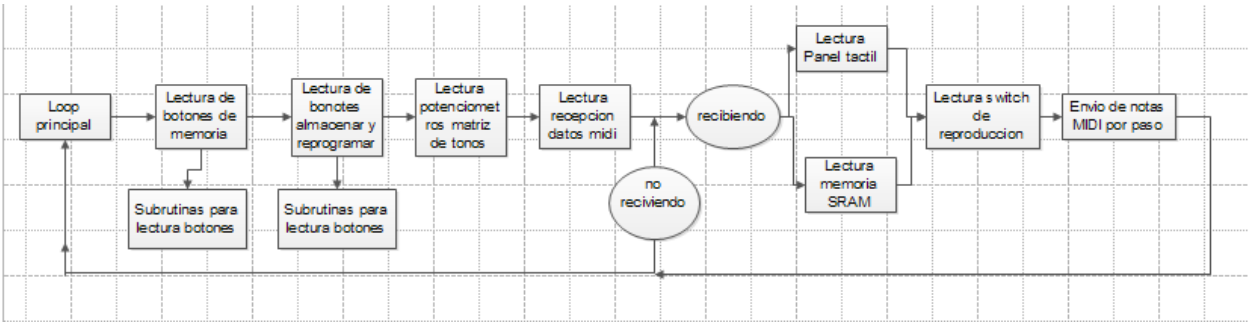
Fuente: Software Fritzing.

.

4.1.6 Flujo global de programación

A continuación se presenta el diagrama de flujo global para la programación del SUB! integrando todos los diagramas de flujo previamente mostrados.

Fig. 11. Diagrama de flujo global de la programación para el secuenciador SUB!



Fuente: Software Edraw.

4.1.7 Diseño de interfaz física

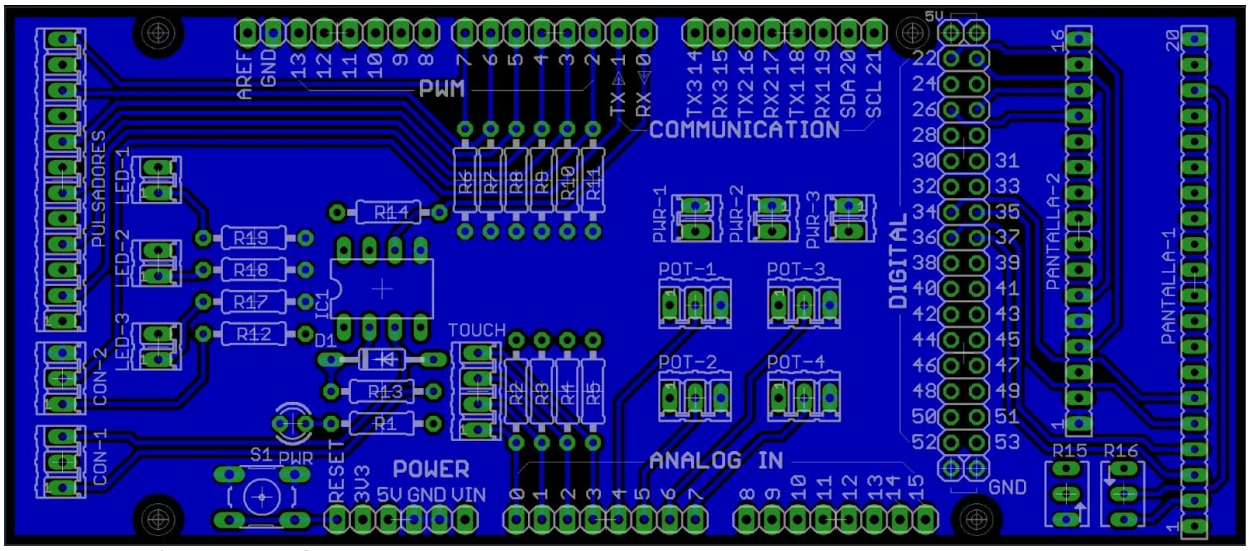
4.1.4.1 Circuito impreso

Se diseñó una placa cuyas medidas se integran con las de la tarjeta Arduino, para la optimización de espacio, se utilizó una doble capa para enviar la alimentación a la placa.

Para el diseño del circuito se utilizó el software Eagle[13] .

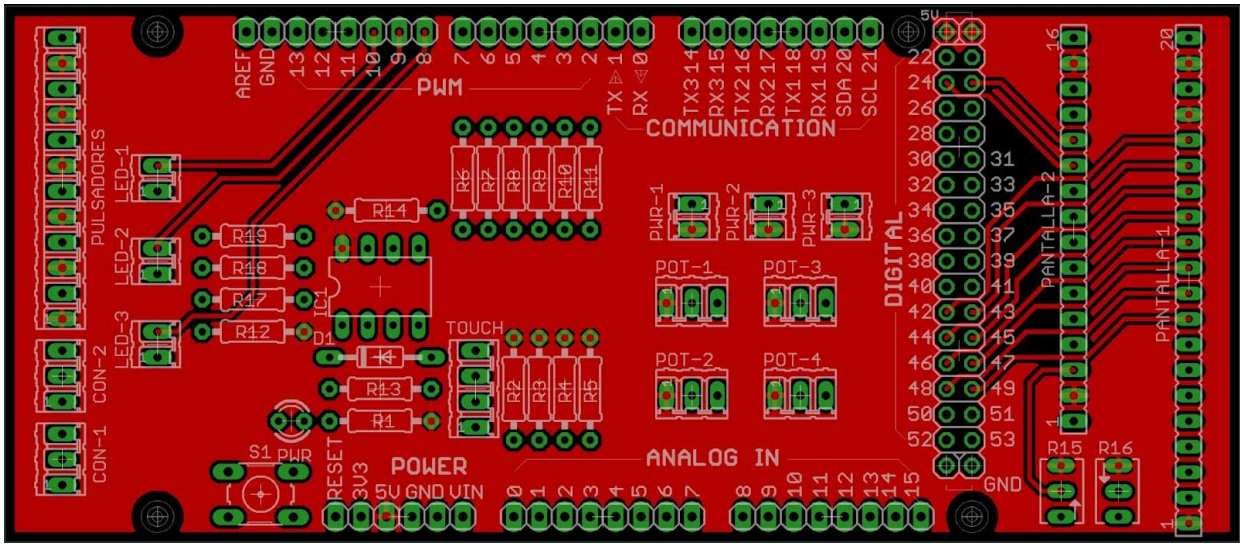
[13] http://www.cadsoftusa.com/eagle-pcb-design-software/?language=en

Fig. 12. Capa superior del circuito impreso.



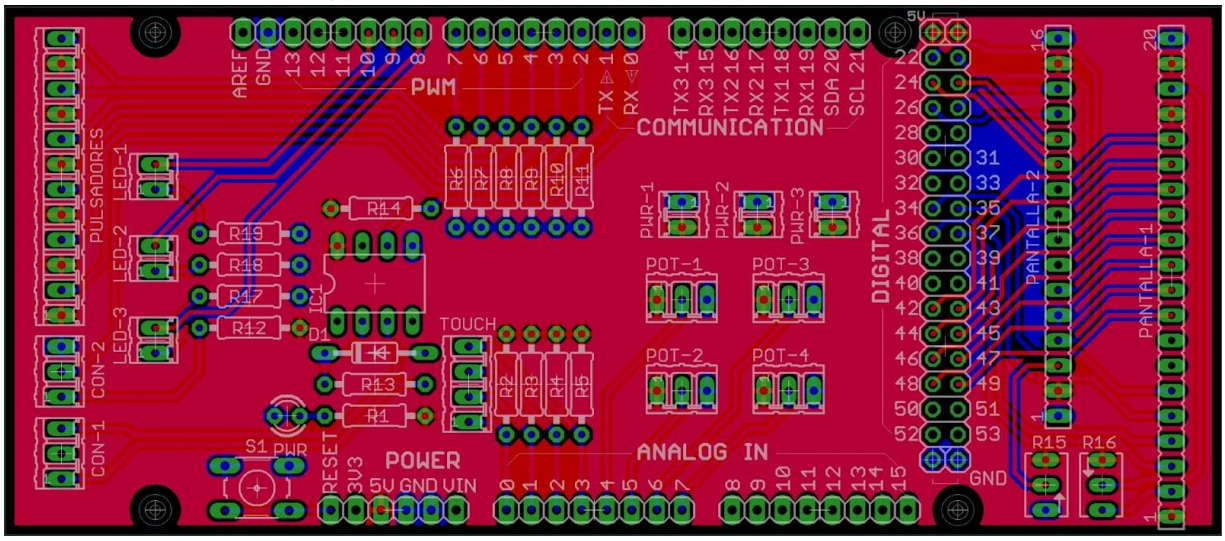
Fuente: Software Eagle.

Fig. 13. Capa posterior del circuito impreso.



Fuente: Software Eagle.

Fig. 14. Capas superior y posterior juntas del circuito impreso.



Fuente: Software Eagle.

4.1.4.2 Diseño parte física del secuenciador

4.1.4.2.1 Objetivo

El diseño tiene como objetivo establecer las medidas adecuadas para que el uso de la consola y el pedal brinden al usuario comodidad y total capacidad de control.

4.1.4.2.2 Requerimientos.

De uso: El sistema SUB! y el pedal se deben adaptar al usuario de tal forma que sea cómodo realizar la actividad de uso.

Funcionales: Los dispositivos deben ´poder ser manipulados en los diferentes tiempos que el músico requiera.

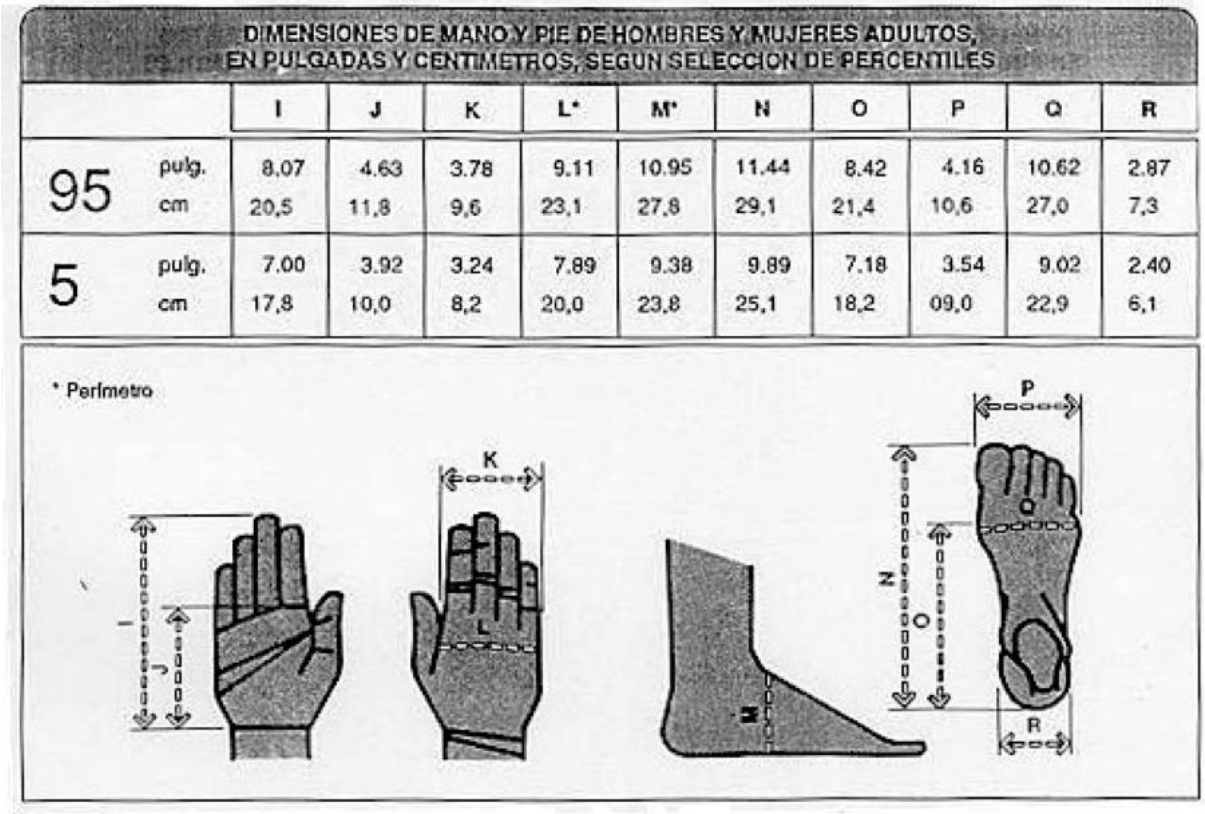
Estructurales: El pedal debe tener una inclinación máxima de 30 grados.

.

Las medidas de la consola deberán ser de acuerdo a los siguientes percentiles[14]

[14] Obtenido de: Panero,Julius, Dimensiones humanas para los espacios interiores; Gustavo Gili, 2007.

Fig. 15. Diagrama de percentiles de las medidas antropométricas del ser humano por Panero.



DIMENSIONES DE MANO Y PIE DE HOMBRES Y MUJERES ADULTOS, EN PULGADAS Y CENTIMETROS, SEGUN SELECCION DE PERCENTILES

		I	J	K	L*	M*	N	O	P	Q	R
95	pulg.	8.07	4.63	3.78	9.11	10.95	11.44	8.42	4.16	10.62	2.87
	cm	20,5	11,8	9,6	23,1	27,8	29,1	21,4	10,6	27,0	7,3
5	pulg.	7.00	3.92	3.24	7.89	9.38	9.89	7.18	3.54	9.02	2.40
	cm	17,8	10,0	8,2	20,0	23,8	25,1	18,2	09,0	22,9	6,1

Fuente: ***Panero,Julius, Dimensiones humanas para los espacios interiores; Gustavo Gili, 2007.***

4.1.4.2.4 Propuestas y conclusiones.

.

Propuesta 1: Partiendo del percentil mas grande, el 95, se arrojaron unas posibles dimensiones para el diseño de la caja principal, estas medidas dieron como resultado una caja de grandes proporciones que no se adapta al concepto del dispositivo, este percentil fue descartado.

Para la caja principal se obtuvieron las siguientes medidas:

Ancho: .30cm
Largo: 30 cm
Profundidad: 6cm.

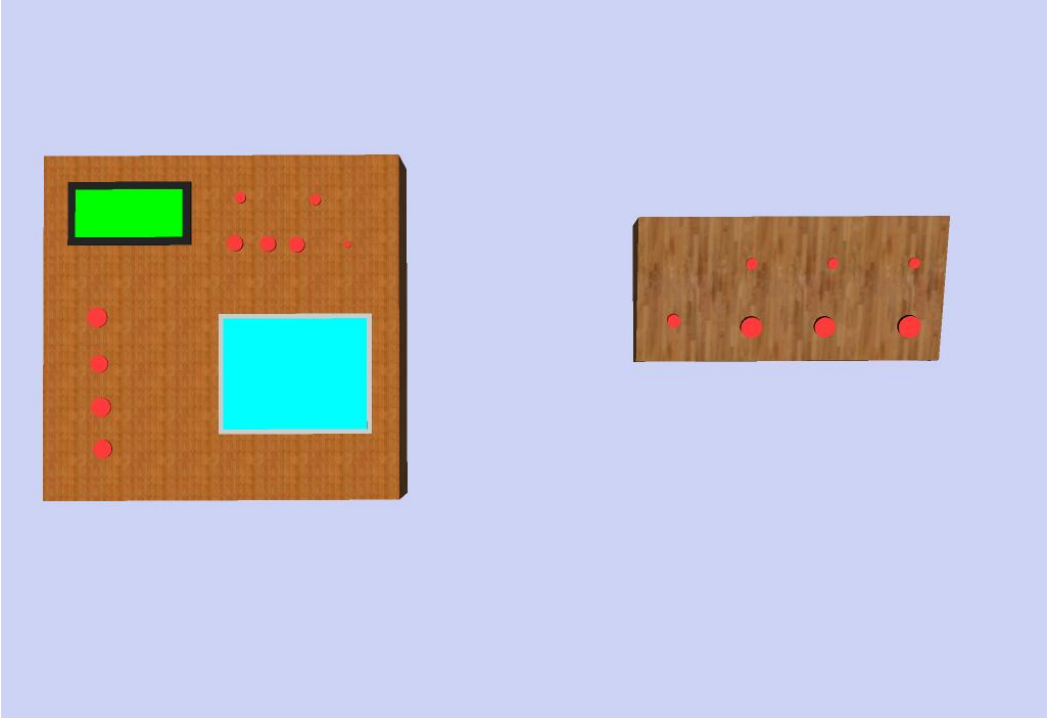
Propuesta 2: Utilizando el percentil 5 se obtuvieron las siguientes medidas:

Caja:
Ancho: 21.4 cm
Largo: 22.1 cm
Profundidad: 6 cm.

Pedalera:
Largo: 27.5cm.
Ancho: 9.9 cm.
Inclinación: 25 grados.

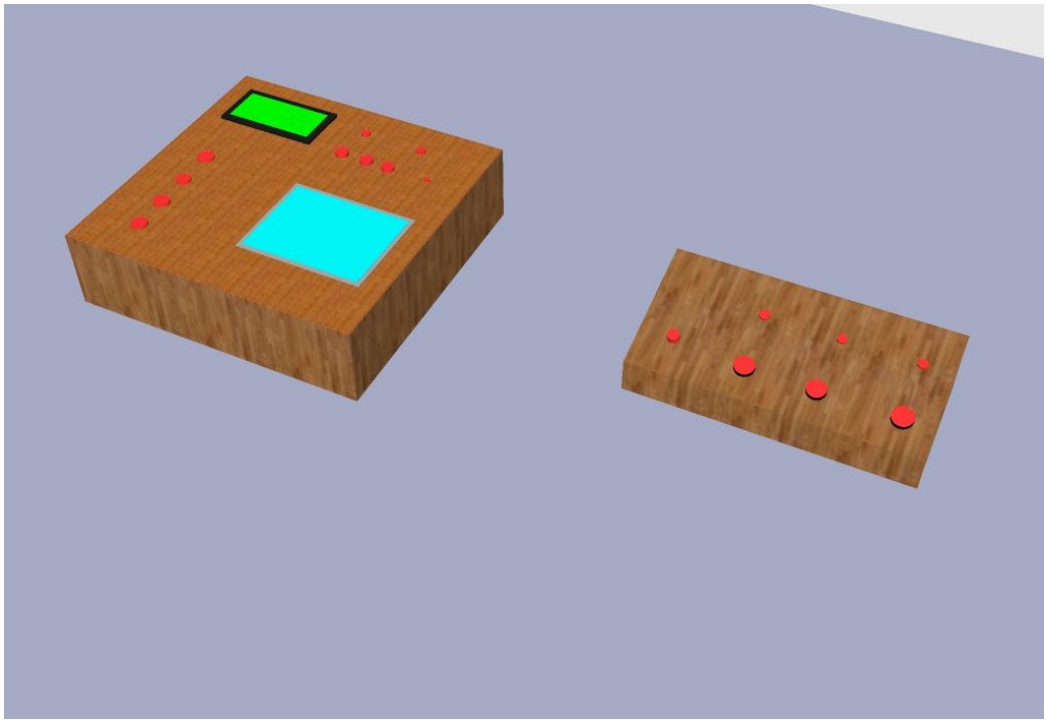
El resultado de este percentil se considero acorde con el concepto del dispositivo.

Fig. 16. Vista frontal caja principal y pedalera.



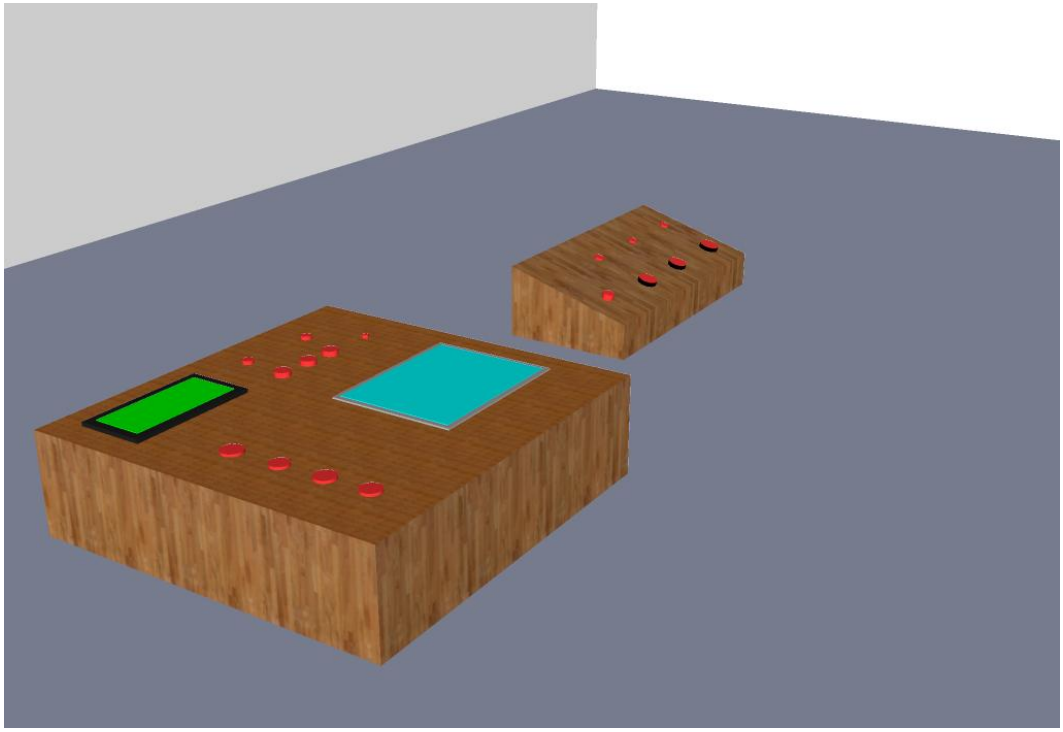
Fuente: Software Google Sketchup.

Fig. 17. Vista lateral derecha caja principal y pedalera.



Fuente: Software Google Sketchup.

Fig. 18. Vista lateral izquierda caja principal y pedalera.



Fuente: Software Google Sketchup.

4.2 SINTETIZADOR VIRTUAL

El sintetizador virtual “SUB! Syn” es un patch de Supercollider[15] (SC) creado como complemento del Secuenciador SUB!. Recibe información MIDI proveniente del secuenciador y la transforma en señales de audio, la señal puede ser modificada por el usuario variando parámetros de síntesis accesibles mediante una interfaz grafica o directamente desde el código.

A pesar de las limitaciones gráficas que exhibe Supercollider en comparación con software similar como Pure Data o Max/Msp, este posee características que lo hacen de particular utilidad para el proyecto:

1. En primer lugar podemos mencionar las posibilidades de interacción relativas a la asignación del mouse a parámetros de síntesis, como ejemplo de esto, el sintetizador SUB! brinda la posibilidad de controlar la frecuencia de corte de un filtro pasa bajos mediante el movimiento del cursor en el eje X y controlar la frecuencia de un LFO con los movimientos del cursor en el eje Y.

2. Supercollider es un software gratis y de licencias abiertas, lo cual reduce los costos del SUB!, y lo convierte en parte de una comunidad mundial de proyectos que se retroalimentan sin ánimo de lucro.

3. La particularidad de la forma en que se aborda la generación de sonidos permite explorar tareas que resultarían demasiado complicadas con otro software[16]

La programación del sintetizador está basada en dos Quarks que potencializan las opciones de control y recepción de mensajes MIDI por parte de SuperCollider. El primer Quark, llamado “ddwVoicer”, es un reproductor de notas MIDI con GUI y asignación de controles incluido, el segundo Quark, “ddwMIDI”, es un marco de trabajo para manejo de entrada y salida de mensajes MIDI.

[15] Supercollider es un lenguaje de programación para síntesis de audio en tiempo real, su licencia es libre y es compatible con OS X, Windows y Linux. http://supercollider.sourceforge.net/

[16] Como ejemplo de esto invitamos a escuchar algunos patchs de http://sccode.org/

La síntesis utilizada por el SUB! es de tipo sustractivo; la relativa simplicidad de este tipo de síntesis permite la minimalización de los parámetros controladores, conservando la sencillez del diseño al tiempo que evoca un sonido vintage característico de los sintetizadores sustractivos populares en los años 60's y 70's.

La programación del SUB! Syn se divide en tres módulos:

1. Sintetizador
2. Controles Sintetizador y MIDI
3. GUI

4.2.1 Módulo 1: Sintetizador.

Esta sección define el funcionamiento y características del sintetizador en términos de sus componentes o UGens, parámetros o argumentos controladores y flujo de señal.

4.2.1.1 Componentes o UGens.

Osciladores: Todos los sonidos son logrados a partir del procesamiento de una señal tipo diente de sierra generada por un solo oscilador, esta forma de onda resulta particularmente útil para la síntesis sustractiva debido a su rico contenido armónico.

Envolvente de amplitud: Se utiliza una envolvente tipo ADSR para controlar la forma en que se comporta la amplitud del sonido ante los mensajes MIDI recibidos. Este elemento, presente en la mayoría de sintetizadores, complementa el sistema SUB! al brindar control sobre el comportamiento de la amplitud de la señal proveniente del secuenciador.

Filtro: Se utiliza un filtro pasa bajos resonante con envolvente como principal filtro del sistema, alternativamente, existe un segundo filtro pasa bajos cuya frecuencia de corte varia según la posición del cursor en la pantalla, este filtro solo puede ser activado o desactivado directamente desde el código.

Envolvente de Filtro: Se utiliza una envolvente tipo ASR para controlar el comportamiento en el tiempo del filtro pasa bajos. Este tipo de envolvente, comúnmente usada en los sintetizadores Moog, brinda una sensación análoga en el sonido y control del filtro.

Oscilador de Baja Frecuencia: Un oscilador de baja frecuencia, aplicado sobre la amplitud del sintetizador, permite modular y distorsionar la señal. Este elemento es particularmente útil para crear efectos y sonidos atonales.

Reverberación: Un efecto de reverberación aplicado sobre la salida del sintetizador brinda la posibilidad de jugar con la espacialidad del sonido sin necesidad de usar procesadores externos.

4.2.1.2 Argumentos controladores.

4.2.1.2.1 Argumentos Oscilador.

Frecuencia (freq): Determina la frecuencia (o nota) del oscilador. El valor de este argumento cambia según el mensaje MIDI de nota on que sea recibido.

Paneo (pan): Permite variar la localización estero del sonido. Se encuentra predeterminado a 0
.

4.2.1.2.2 Argumentos de la envolvente de amplitud.

Ataque (ampenv_attack_time): Determina el tiempo en segundos que toma la señal en variar su amplitud desde 0 hasta el Nivel Pico o Peak Level. Con un rango de 0 a 30, permite lograr sonidos con un carácter percusivo (ataque rápido) y sonidos con un carácter atmosférico (ataque lento).

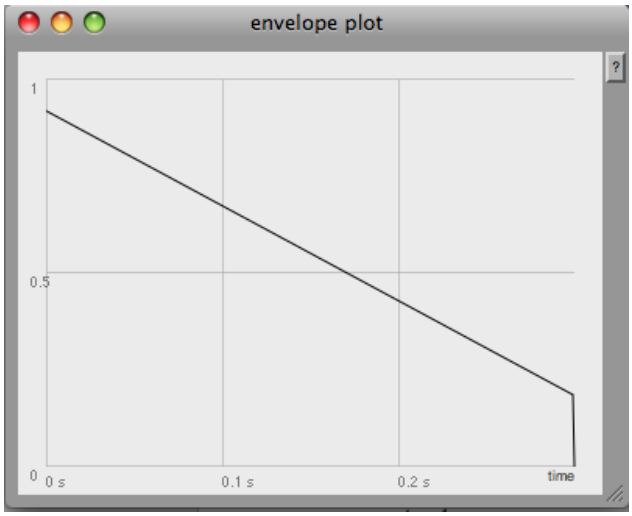
Nivel Pico (ampenv_peak_level): Amplitud máxima lograda por la envolvente.

Decaimiento (ampenv_decay_time): Tiempo en segundos que tarda la señal en pasar del nivel pico al nivel de sostenimiento.

Nivel de Sostenimiento (ampenv_sustain_level): Nivel que alcanza la envolvente al llegar al punto de sostenimiento.

Relajación (ampenv_release_time): Define el tiempo que toma la amplitud de la señal en decaer hasta 0 db una vez ha terminado el tiempo de sostenimiento.

Fig. 19.Amp. env (ataq=0, nivel pico=0.92, dec=0.3, nivel sust=0.2, Rel=0).



Fuente: Supercollider plot.

4.2.1.2.3 Argumentos de la envolvente de filtro.

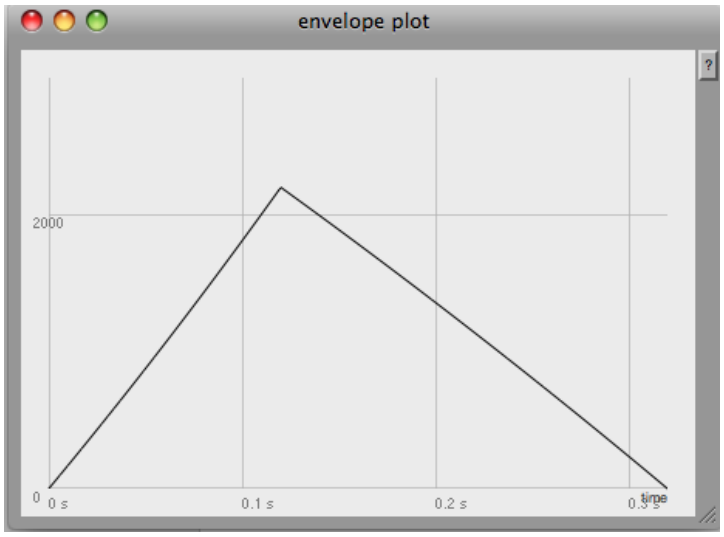
Ataque (filterenv1_attack_time): Tiempo que tarda el filtro en cerrarse o alcanzar la frecuencia de corte especificada.

Nivel de Sostenimiento (filterenv1_sustain_level): Nivel de la envolvente en la frecuencia de corte.

Relajación (filterenv1_release_time): Tiempo que tarda el filtro en pasar de la frecuencia de corte deseada a una frecuencia neutral.

Rq (rq1): Nivel de resonancia del filtro.

Fig. 20 .Filt. env (ataq=0.12, nivel sust=2200, Rel=0.2, rq=0.15).



Fuente: Supercollider plot.

4.2.1.2.4 Argumentos oscilador de baja frecuencia (LFO)

Frecuencia de oscilación (lfo1Speed): Valor en segundos para la frecuencia a la que oscila el LFO. Con un rango de 0 a 30, permite crear efectos ondulatorios (tiempo largos) o efectos de distorsión (tiempos muy cortos).

4.2.1.2.5 Argumentos de la reverberación

Damp (verbDamp): Define la frecuencia de corte de un filtro pasa bajos ubicado al final de la reverberación. Controla la forma en que se percibe el material de las paredes del cuarto simulado.

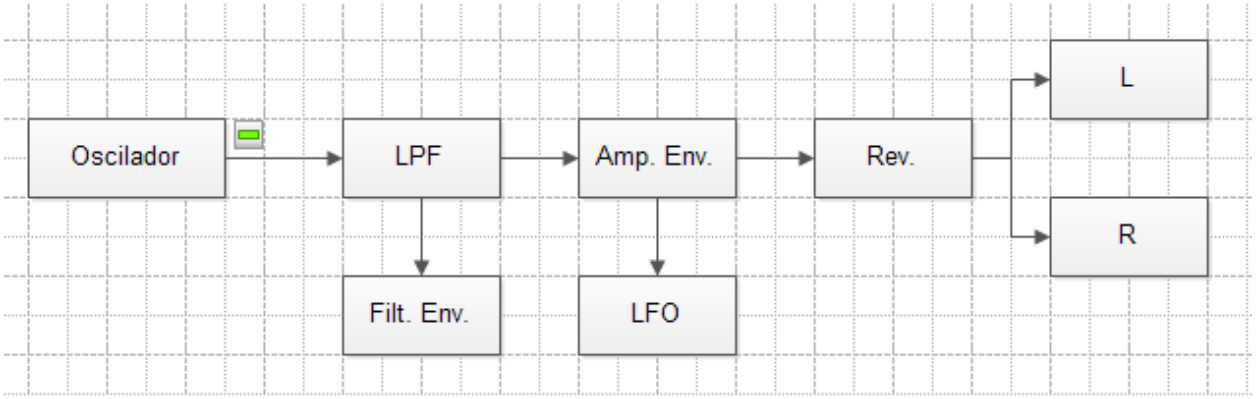
Room (verbRoom): Define el tamaño del cuarto simulado por la reverberación.

Mezcla (mix): Define la cantidad de señal reverberada o sin reverberación.

4.2.1.3 Flujo de Señal

El flujo de señal para el sintetizador es representado en el siguiente diagrama.

Fig. 21. Diagrama de bloques de flujo de señal para el sintetizador virtual.



Fuente: Software Edraw.

La variable “sig” contiene la información de la señal que va siendo procesada, esta va adquiriendo un nuevo valor a medida que pasa a través de los UGens.

```
sig    =     Saw.ar(freq);
lfo1 = LFSaw.ar(lfo1Speed, 0, 0.5, 0.5);
sig    =     RLPF.ar(sig, filterenv1, rq1);
 /////////sig    =    LPF.ar(sig,(MouseX.kr(40, 10000, 1, 0.2)));
sig    =    sig * ampenv * lfo1 * 0.24;
sig = FreeVerb.ar(sig, mix, verbRoom, verbDamp, 1,0);
```

La sección “/////////sig = LPF.ar(sig,(MouseX.kr(40, 10000, 1, 0.2)));” declara un filtro pasa bajos controlado por la posición del mouse que puede ser activado (predeterminadamente desactivado), removiendo los slash(///) antes de la línea de código en mención.

El ruteo final de la señal, hacia los canales 1 y 2 de nuestra interfaz, se hace mediante la siguiente fracción de código:

```
Out.ar (0,sig);
Out.ar (1,sig);
```

4.2.2 Módulo 2: Controles Sintetizador y MIDI

En este módulo de la programación se utilizan los quarks ddwVoicer y ddwMIDI para definir los argumentos que serán mostrados en la interfaz gráfica, la forma en que el usuario podrá controlarlos y la configuración de nuestro sistema MIDI. Los argumentos se definen de la siguiente manera:

v.mapGlobal (\"nombre del control", nil, "valor inicial", \"especificación de control");

La especificación de control (o control spec) determina el tipo de unidad y rango que se utilizará para el control, las especificaciones utilizadas se describen en la siguiente tabla:

Nombre	Unidad	Rango	Incremento
Time	Seg	0 - 30	0,1
Nivel	No	0 - 3	0,1
Amp	No	0-1	0,1
Freq	Hz	20 - 20000	0,1

Tabla 5. Especificación de controles utilizados.

Nótese que algunas especificaciones de control no tienen unidades, estas especificaciones funcionan en base a convenciones particulares de SC.

Los argumentos definidos para la interfaz, su valor inicial y su respectiva especificación de control son los siguientes:

Argumento	Valor Inicial	Especificación
Ataque (amplitud)	0	time
Relajación (amplitud)	0	time
Decaimiento (amplitud)	0	time

Nivel de sostenimiento(amplitud)	0	amp
Nivel pico (amplitud)	1.5	nivel
Ataque (filtro)	0	time
Nivel de sostenimiento (filtro)	2200	freq
Relajación (filtro)	0	time
rq filtro	0,15	amp
Velocidad LFO	0	time
Damp Reverb	0	amp
Room Reverb	0	amp
Mix Reverb	0	amp

Tabla 6. Lista de argumentos definidos para la interfaz.

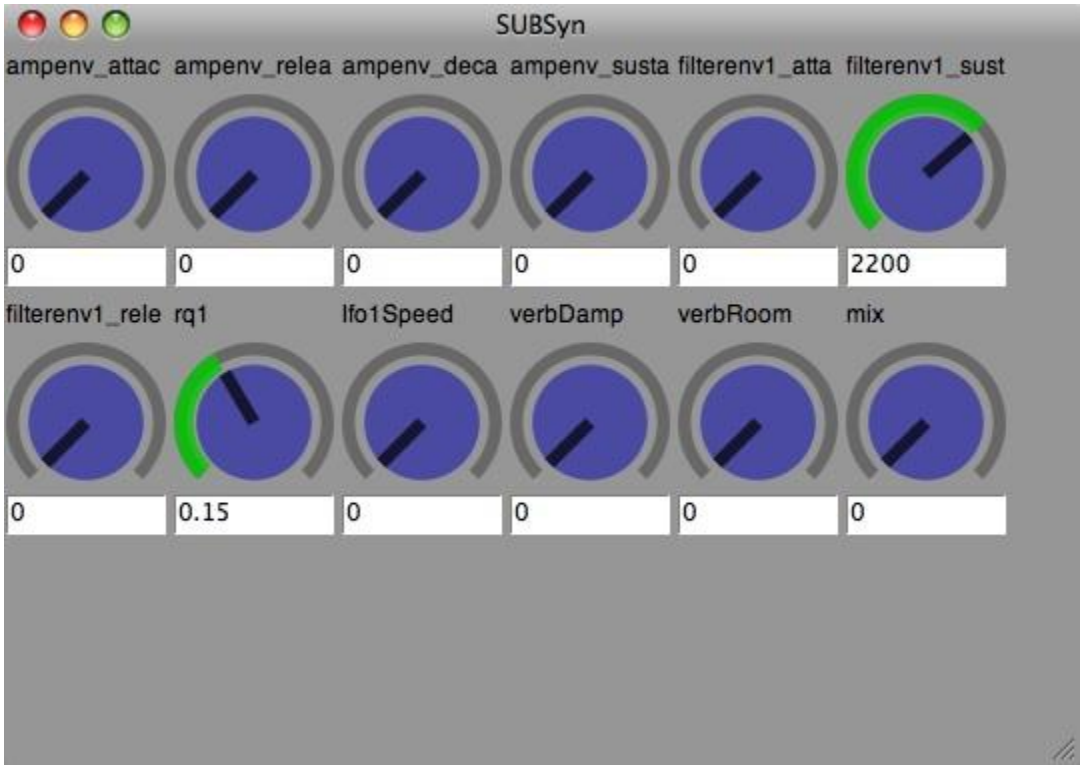
Se utiliza el comando **MIDIPort**.**init**; del ddwMIDI para establecer conexión entre Supercollider y los dispositivos MIDI conectados al computador, finalmente se usa el comando "VoicerMIDISocket ([dispositivo, canal], voicer)" para configurar que dispositivo y canal MIDI se desea utilizar.

4.2.3 Módulo 3: GUI

Este módulo es el encargado de generar la interfaz gráfica. El programa realiza un recorrido a través de las definiciones de control realizadas y asigna un knob para cada control. Se seleccionaron knobs tipo EZ[17] para facilitar la interacción del usuario con el sintetizador sin necesidad de un controlador MIDI. La interfaz resultante puede observarse en la Fig. 22.

[17] Tipo de knob con color y tamaño extra grande ofrecido por la interfaz gráfica de Supercollider.

Fig. 22. Interfaz grafica del sintetizador SUB!.



Fuente: Propia.

Fig. 23. Comparación interfaz grafica sintetizadores substractivos virtuales. Minimoog y SUB! Syn.

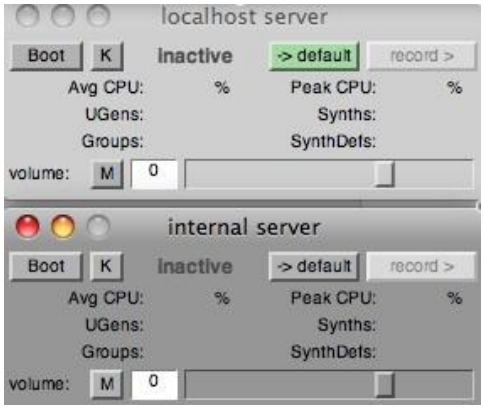
Fuente: Propia.

Los tres módulos de la programación funcionan de manera conjunta, pero también pueden ser modificados independientemente para permitir al usuario operar el sintetizador desde el GUI o directamente desde el código, de esta manera, se puede personalizar el sintetizador o realizar híbridos con otros patchs de Supercollider sin necesidad de tener conocimientos avanzados de programación.

Debido a que inicialmente la operación del sintetizador puede parecer compleja para usuarios ajenos a Supercollider, se presenta a continuación un listado detallado de los pasos necesarios a seguir para correr el código. Esta sección fue mostrada a los usuarios al momento de realizar las pruebas de satisfacción. Como valor agregado, se espera que esta información sirva para que los usuarios puedan desarrollar sus propios códigos utilizando las bases de interacción gráfica y MIDI proporcionadas por el patch.

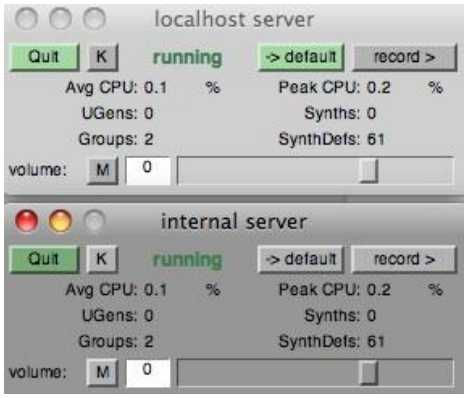
Primer paso: Prender los servidores utilizando el botón “Boot”

Fig. 24. Servidores del supercollider apagados.



Fuente: Propia.

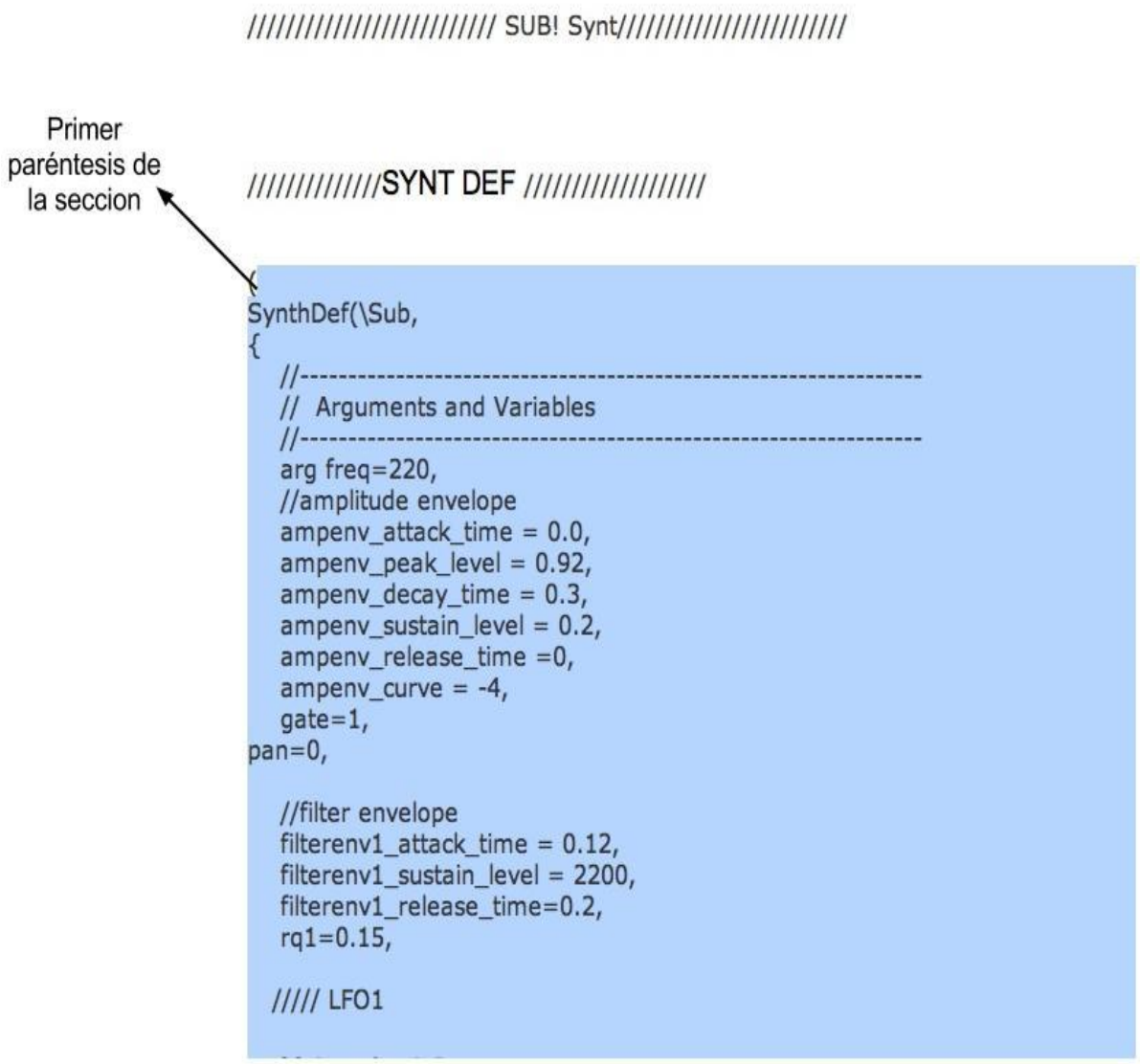
Fig. 25. Servidores del supercollider prendidos.



Fuente: Propia.

Segundo paso: Ejecutar la sección del código “SYNT DEF”. Para ejecutar esta parte del código se debe hacer doble click sobre el primer paréntesis de la sección y luego presionar CTRL + ENTER (Mac).

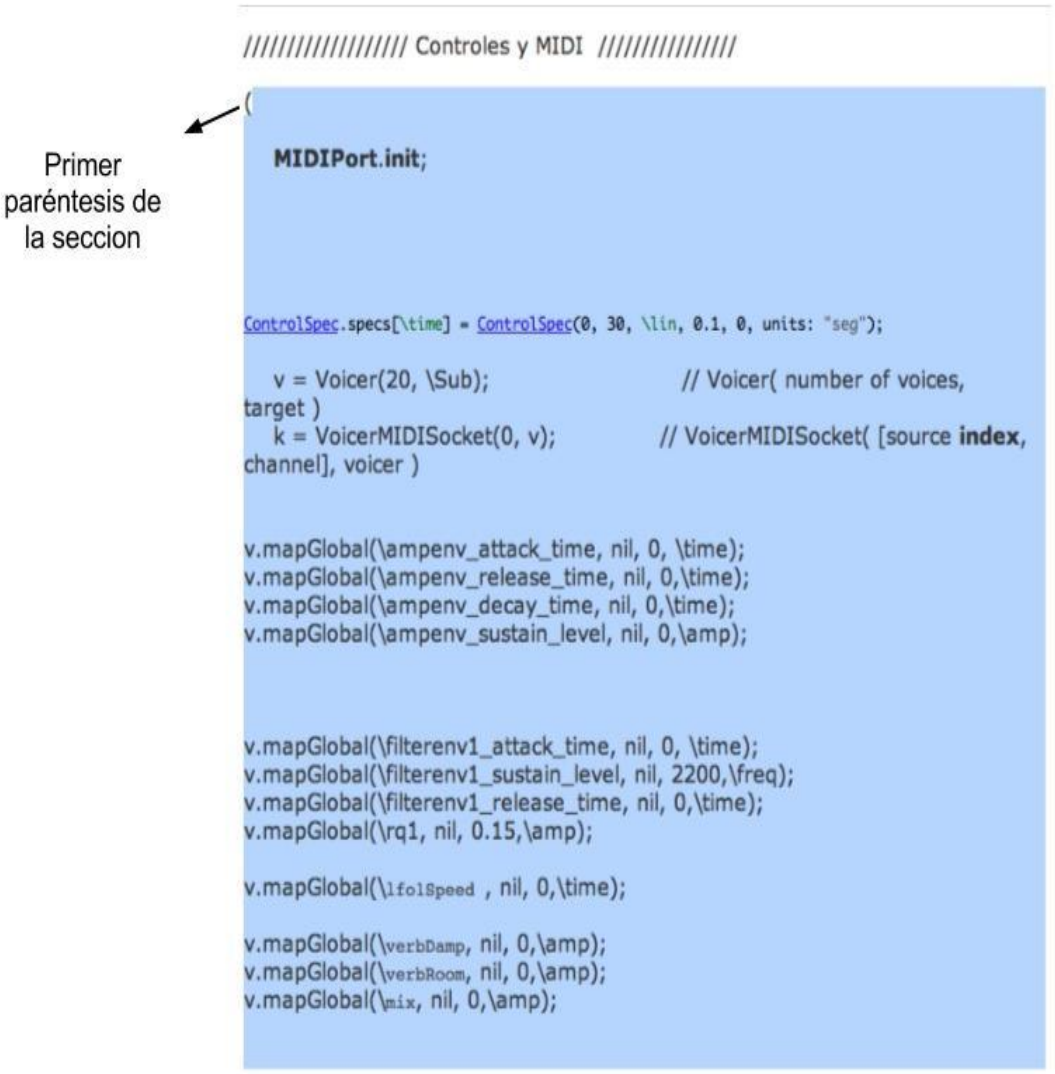
Fig. 26. Primera parte del código Synt def.



Fuente: Propia.

Tercer paso: Ejecutar la sección del código “Controles MIDI”.

Fig. 27. Segunda parte del código Controles y MIDI.




```
////////////////// Controles y MIDI ////////////////
(
   MIDIPort.init;



ControlSpec.specs[\time] = ControlSpec(0, 30, \lin, 0.1, 0, units: "seg");

   v = Voicer(20, \Sub);                    // Voicer( number of voices,
target )
   k = VoicerMIDISocket(0, v);            // VoicerMIDISocket( [source index,
channel], voicer )


v.mapGlobal(\ampenv_attack_time, nil, 0, \time);
v.mapGlobal(\ampenv_release_time, nil, 0,\time);
v.mapGlobal(\ampenv_decay_time, nil, 0,\time);
v.mapGlobal(\ampenv_sustain_level, nil, 0,\amp);



v.mapGlobal(\filterenv1_attack_time, nil, 0, \time);
v.mapGlobal(\filterenv1_sustain_level, nil, 2200,\freq);
v.mapGlobal(\filterenv1_release_time, nil, 0,\time);
v.mapGlobal(\rq1, nil, 0.15,\amp);

v.mapGlobal(\lfoSpeed , nil, 0,\time);

v.mapGlobal(\verbDamp, nil, 0,\amp);
v.mapGlobal(\verbRoom, nil, 0,\amp);
v.mapGlobal(\mix, nil, 0,\amp);
```

Fuente: Propia.

Cuarto paso: Ejecutar la sección del código “GUI”.

El GUI aparecerá automáticamente con los controles asignados en la sección “controles y MIDI”.

Fig. 28. Tercera parte del código GUI.

```
)


////////////////////////////// GUI ////////////////////////////////////

(
var flow, knobs, updaters;
w = Window(\SUBSyn, Rect(500, 50, 540, 360));
flow = FlowView(w, w.view.bounds);
knobs = v.globalControlsByCreation.collect { |gc|
	EZKnob(flow, Rect(0, 0, 80, 120), label: gc.name, controlSpec: gc.spec,
action: { |view|
		gc.set(view.value)
	}, initVal: gc.value);
};
updaters = v.globalControlsByCreation.collect { |gc, i|
	Updater(gc, { |obj, event|
		if(event[\what] == \value) {
			defer { knobs[i].value = gc.value };
		}
	});
};
w.onClose = { updaters.do { |upd| upd.remove } };
flow.resizeToFit(reflow: false, tryParent: true);
w.front;
)


	v.gui;
)
```

Fuente: Propia.

5. PRESENTACION DE RESULTADOS

Se realizaron dos tipos de mediciones para evaluar la funcionalidad y eficiencia del sistema: Comparativas y de satisfacción.

Pruebas comparativas: Buscan identificar diferencias puntuales entre el sistema SUB! y uno de los secuenciadores más usados en la industria musical desde los años 80's, las MPC de akai. Acorde a lo mencionado por D'Errico en su tesis "Behind the Beat: Technical and Practical Aspects of Instrumental Hip-Hop Composition": **"***La Akai MPC ha sido considerada por mucho tiempo tecnología estándar de sampleo en cualquier estudio de producción de hip-hop. Cubriendo las diversas técnicas desarrolladas por Djs pioneros de hip-hop - incluyendo creación, edición y mezcla de beats- la MPC introdujo un rango más amplio de posibilidades no solo en el aspecto de manipulación de samples individuales, sino también su ensamble dentro de una composición musical. Además de esto, la expansión de la maquina ha coincidido con el desarrollo musical de la tradición del hip-hop, al tiempo que los productores han respondido y reaccionado a la tendencias cambiantes de la tecnología con nuevas tendencias en el performance y la práctica"* (2011).

Las mediciones hechas sobre la MPC 1000 servirán como pruebas de control, cuyos valores serán tomados como marco de referencia.

Pruebas de satisfacción: Son encuestas en las que una muestra de la población usuaria del SUB! evalúa diversos aspectos relativos a la funcionalidad del sistema. A partir de estos datos se podrá determinar la eficiencia del diseño.

A continuación se presenta una descripción detallada de las pruebas realizadas y sus resultados:

5.1 HERRAMIENTAS UTILIZADAS EN LAS PRUEBAS

5.1.1 Hardware

Akai MPC 1000:

La MPC 1000 es un sampler de 32 voces con un secuenciador de 64 tracks que puede manejar hasta 32 canales MIDI. Puede ser sincronizado con un reloj MIDI y leer secuencias MIDI. Ha marcado el sonido de artistas como J Dilla, Flying Lotus, Richard Blair, Dj Shadow, *Grandmaster Flash*, Dj Premier y *Kanye West* entre otros.

Korg Electribe ES-1:

El ES-1 es un sampler rítmico y un secuenciador, hace parte de la primera ola de Electribes junto con la máquina de ritmos ER-1 y el sintetizador EA-1. El ES1 puede grabar y guardar hasta 150 samples (100 mono, 50 estéreo). Su frecuencia de sampleo es de 32 kHz y puede ser sincronizado vía MIDI in con un reloj externo.

Dave Smith Mopho (modulo):

Sintetizador análogo monofónico. Ofrece una salida estéreo y una entrada de audio para procesar sonidos externos atreves de de sus filtros análogos. Posee dos osciladores con un sub cada uno, entrada y salida MIDI así como también un puerto de MIDI Thru.

M-Audio Fast track Pro:

Interfaz de audio de 24-bit/96kHz con dos preamplificadores de mic/line. 2x4 I/O balanceadas y no balanceadas, S/PDIF coaxial, MIDI I/O y salida de audífonos. Compatible con Mac y PC.

Digidesign Digi 003 (Rack):

Interfaz de audio de 24-bit/96kHz con hasta 18 canales de I/O, un puerto de entrada MIDI y dos puertos de salida MIDI. Conectores ópticos y RCA con soporte para (ADAT) I/O y S/PDIF. Word Clock in y out.

5.1.2 Software

Plataforma: OS X 10.5

Digidesign Pro Tools LE 8:

Pro Tools es una estación de trabajo de audio digital desarrollada por Avid con soporte para Microsoft Windows y Mac OS X. Es ampliamente utilizada a nivel mundial por profesionales de la industria musical y audiovisual en las áreas de grabación, edición, producción y post producción de audio[18].

Ableton Live 8:

Es un secuenciador y estación de trabajo digital desarrollado por Ableton para OS X y Windows. En contraste con muchas otras estaciones de trabajo, Live está diseñado para ser un instrumento usado para interpretación en vivo y como herramienta de composición y arreglo. La última versión del software, Live 8, fue lanzada en el 2009. En el 2011 fue reconocido uno de los 10 DAW's mas usados en el mundo según una encuesta realizada por la revista Music Radar[19]

5.2 PRUEBAS Y MEDICIONES COMPARATIVAS.

5.2.1 Determinación de retraso de reloj MIDI

Los dispositivos de hardware MIDI comúnmente toman cierta cantidad de milisegundos en responder tras recibir un mensaje MIDI de nota o sincronía[20]. Para trabajar con mensajes de reloj, software como Pro Tools y Ableton Live poseen la opción de minimizar esta latencia mediante un parámetro llamado MIDI Beat Clock Offset (Pro Tools) o MIDI Clock Sync Delay (Live).

[18] Una reseña sobre la historia de Pro Tools puede ser encontrada esta pagina web http://www.musicradar.com/tuition/tech/a-brief-history-of-pro-tools-452963/3

[19] http://www.musicradar.com/tuition/tech/the-15-best-daw-software-apps-in-the-world-today-238905/15)/1

[20] **Collins, Mike. Working with MIDI. Focal Press, 2011: 136. (Traduccion del autor)**

5.2.1.1 Diseño de la prueba

Según lo sugerido en la guía de referencia de Pro Tools el tiempo de retraso puede ser calculado de la siguiente manera:

Primer Paso: Establecer Pro Tools como secuenciador maestro y el dispositivo hardware como esclavo

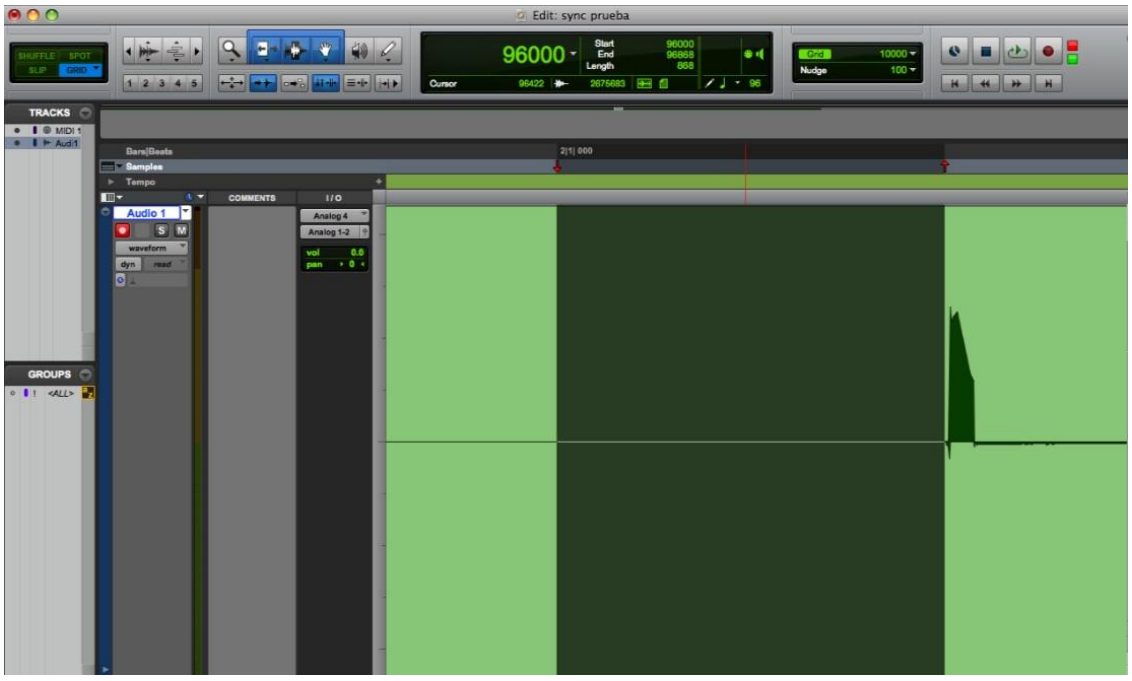
Segundo paso: Grabar una señal de audio o MIDI cuantizada proveniente del secuenciador en hardware

Tercer Paso: Setear la escala de tiempo principal (Main Time Scale) a Bars|Beats. Habilitar las opciones Snap to Grid y Show Grid.

Cuarto Paso: Utilizando la función Tab to Trasient, seleccionar el lapsus de tiempo entre la barra donde debería encontrarse la transiente y el punto donde actualmente se encuentra.

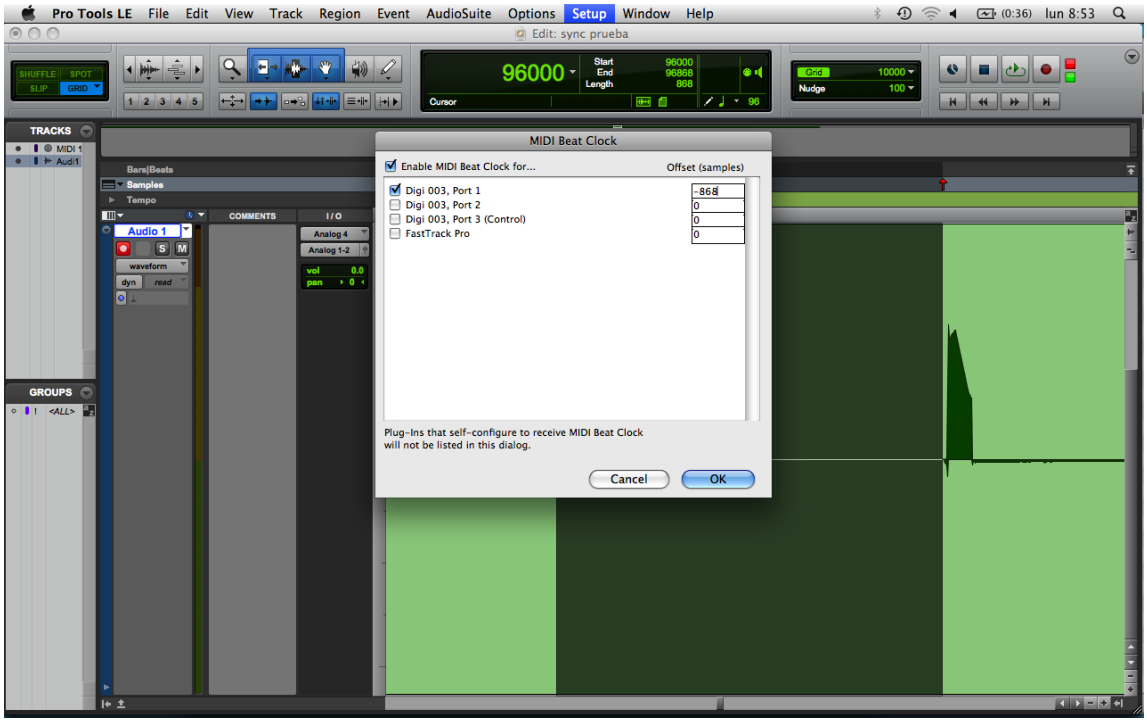
Quinto Paso: Cambiar de modo Bars|Beats a Samples y observar el valor indicado por Pro Tools en el cuadro Lenght. Para el caso de ableton el valor puede ser observado en la parte de abajo en el indicador de Lenght.

Fig. 29. Prueba de retraso Pro Tools.



Fuente: Propia.

Fig. 30. Configuración retraso de reloj, Pro Tools.



Fuente: Propia.

Fig. 31. Configuración retraso de reloj Ableton Live.

Fuente: Propia.

5.2.1.2 Conexiones

	MIDI in	MIDI out	Audio in	Audio out
MPC o SUB!	**Cable 1**	**Cable 2 (opción 1)**		**Cable 3 (opción 2)**
Interfaz MIDI	**Cable 2 (opción 1)**	**Cable 1**	**Cable 3 (opción 2)**	

Tabla 7. Tabla de conexionado para prueba de retraso.

5.2.1.3 Resultados obtenidos

Vale la pena mencionar que este parámetro no se encuentra estrictamente ligado al dispositivo (secuenciador en este caso) sino al sistema MIDI en general, la variación del retardo según interfaz y DAW se puede observar en la siguiente tabla:

	Avid Pro tools	Ableton Live
Software / **Interfaz MIDI**		
Digi 003	**MPC: 870 samples** **SUB!: 950 samples**	**MPC: 1500 samples** **SUB!: 1200**
Fast TR	**N/A**	**MPC: 870 samples** **SUB!: 950 samples**

Tabla 8. Tabla de resultados para prueba de retraso.

5.2.2 Prueba de estabilidad y precisión de la sincronía MIDI con software y hardware externo

Busca medir la precisión y estabilidad del secuenciador al funcionar como esclavo de un dispositivo que envía una señal MIDI de reloj[21].

5.2.2.1 Prueba con software externo

5.2.2.1.1 Descripción de la prueba:

Primer paso: Una señal de percusión cuantizada (señal guía) fue generada en el secuenciador maestro (DAW).

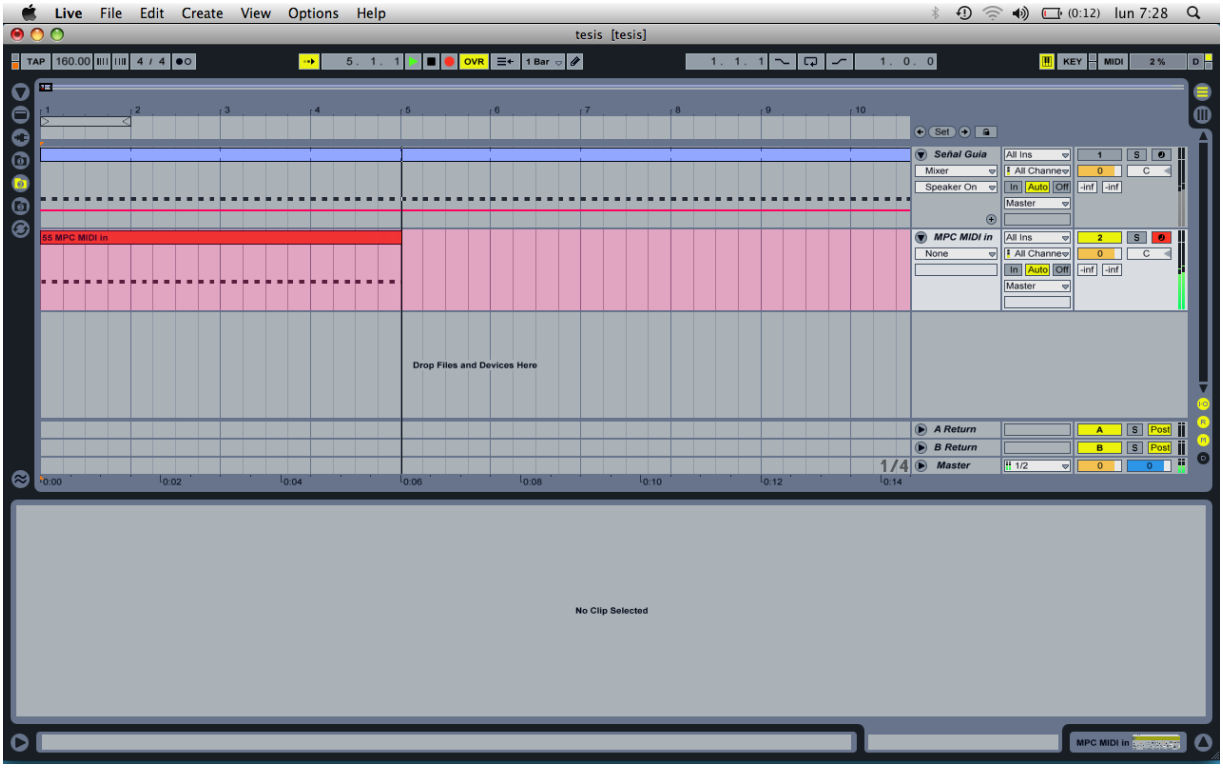
Segundo paso: Una secuencia de igual resolución fue creada en nuestro secuenciador de prueba (MPC y SUB!)

Tercer paso: Se inicio la reproducción y grabación simultanea de la señal guía y la secuencia durante un tiempo de 15 minutos.

Cuarto paso: Utilizando las opciones Snap to Grid y Snap to Note podemos obtener los tiempos de fluctuación entre la señal cuantizada y la señal proveniente del secuenciador.

[21] El diseño de esta prueba se hizo con base en el documento “Checking Tempo Stability of MIDI Sequencers” presentado por Marius Perron en la convención No 97 de la AES.

Fig. 32. Pantallazo prueba de estabilidad Ableton Live.



Fuente: Propia.

5.2.2.1.2 Conexiones:

Dispositivo	MIDI IN	MIDI OUT
MPC o SUB!	Cable 1	Cable 2
Digi 003	Cable 2	Cable 1

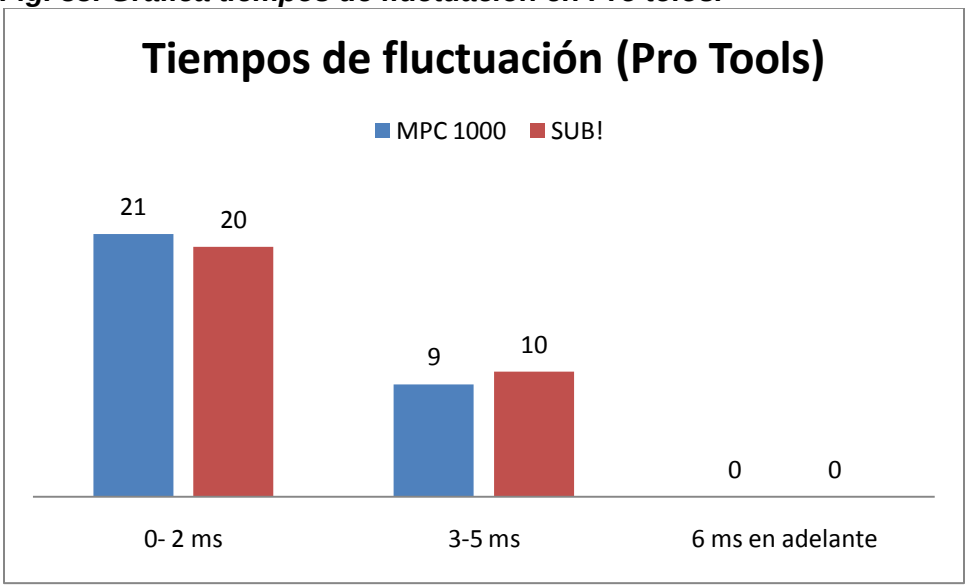
Tabla 9. Tabla de conexionado para prueba de estabilidad.

5.2.2.1.3 Resultados Obtenidos:

Se tabulo el tiempo máximo de fluctuación por repetición en 10 repeticiones por valor de división de tempo, para un total de 30 repeticiones por cada combinación de software y hardware. Las tablas de datos pueden ser encontradas en el documento “secuenciador sub! final” adjunto en el cd del proyecto.

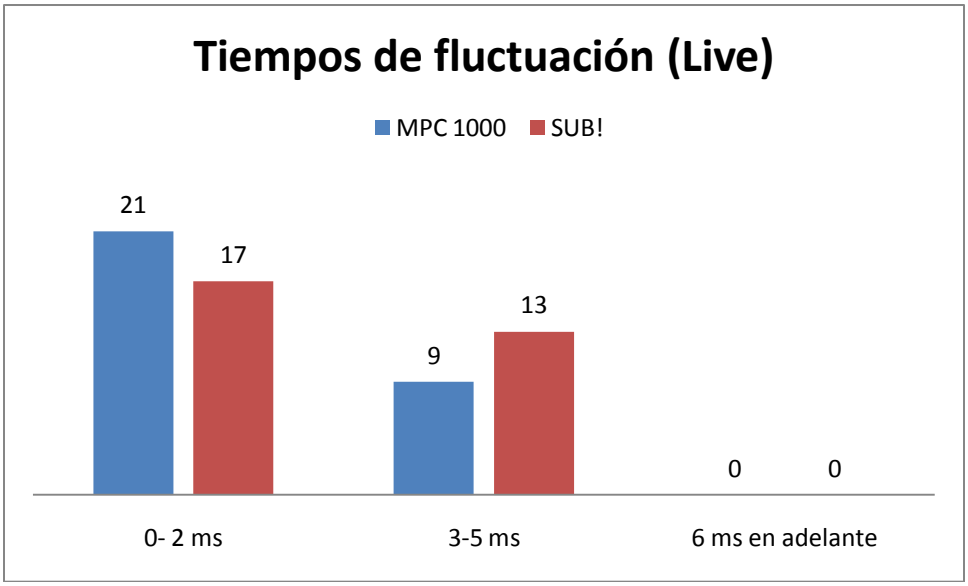
La primera nota enviada no fue tabulada ya que esta exhibe un retraso constante para todos los dispositivos, este retraso puede atribuirse a un tiempo muerto en que él no se ha establecido aun la compensación por atraso de sincronía de reloj.

Fig. 33. Grafica tiempos de fluctuación en Pro tolos.



Fuente: Software Microsoft Excel .

Fig. 34. Grafica tiempos de fluctuación Ableton Live.



Fuente: Software Microsoft Excel.

Los resultados revelan la existencia de un margen de imprecisión en la MPC y en el SUB!, esta imprecisión, cuyo valor oscila mayoritariamente entre ±0 y ±2 ms, alcanza valores de hasta ±5 ms según la resolución y complejidad de la secuencia.

5.2.2.2 Prueba con hardware externo

Para esta prueba la señal de reloj es enviada por el Electribe ES-1.

5.2.2.2.1 Descripción de la prueba:

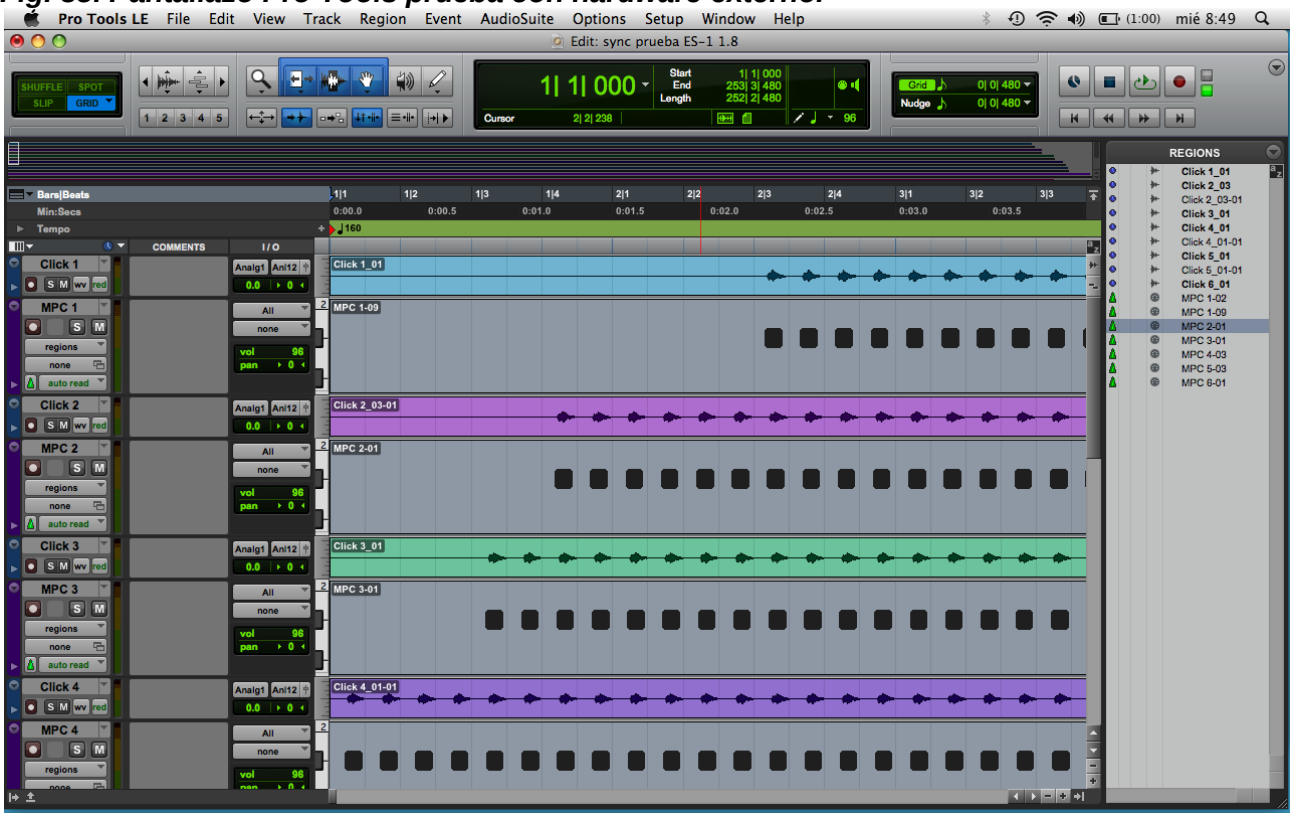
Primer Paso: Programamos una señal de percusión cuantizada (señal guía) en nuestro secuenciador maestro (Electribe ES-1).

Segundo paso: Creamos una secuencia de igual resolución en nuestro secuenciador prueba (MPC o SUB!).

Tercer paso: Iniciamos la reproducción y grabación simultanea de la señal guía proveniente del Electribe y la secuencia proveniente del secuenciador prueba.

Cuarto paso: Comparamos las fluctuaciones de tiempo entre las dos señales.

Fig. 35. Pantallazo Pro Tools prueba con hardware externo.



Fuente: Propia.

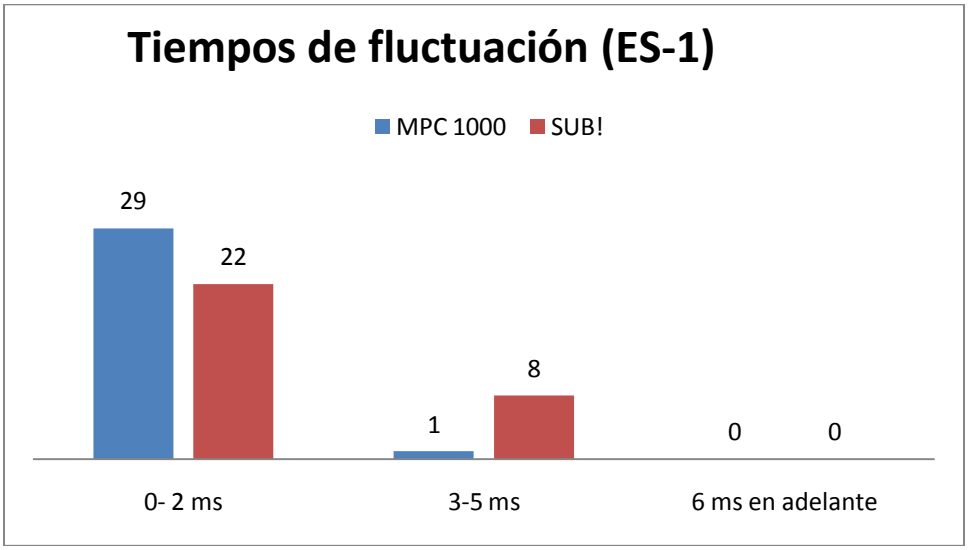
5.2.2.2.2 Conexiones

Dispositivo	MIDI IN	MIDI OUT	AUDIO IN	AUDIO OUT
MPC o SUB	Cable 1	Cable 2		
Electribe ES-1		Cable 1		Cable 3
Digi 003	Cable 2		Cable 3	

Tabla 10. Tabla de conexionado para prueba con hardware externo.

5.2.2.2.3 Resultados Obtenidos

Fig. 36. Tiempos de fluctuación prueba con hardware externo.



Fuente: Software Microsoft Excel.

La utilización de hardware externo como maestro de sincronización muestra un efecto positivo en el margen de imprecisión de ambos secuenciadores.

5.2.3 Prueba de estabilidad y precisión en el almacenamiento y envío de mensajes MIDI con software y hardware externo

La primera parte de esta prueba busca evaluar la estabilidad y precisión de los mensajes almacenados y enviados por el secuenciador en relación a lo programado por el usuario, es decir, que la información guardada y enviada por el dispositivo sea realmente la información que el usuario secuenció inicialmente, la segunda parte, buscar evaluar la interacción del secuenciador con el sintetizador virtual SUB!

5.2.3.1 Primera Parte

5.2.3.1.1 Descripción de la prueba

Primer paso: Iniciamos la reproducción de nuestro secuenciador maestro (Electribe ES-1) y la grabación de mensajes MIDI en nuestro DAW

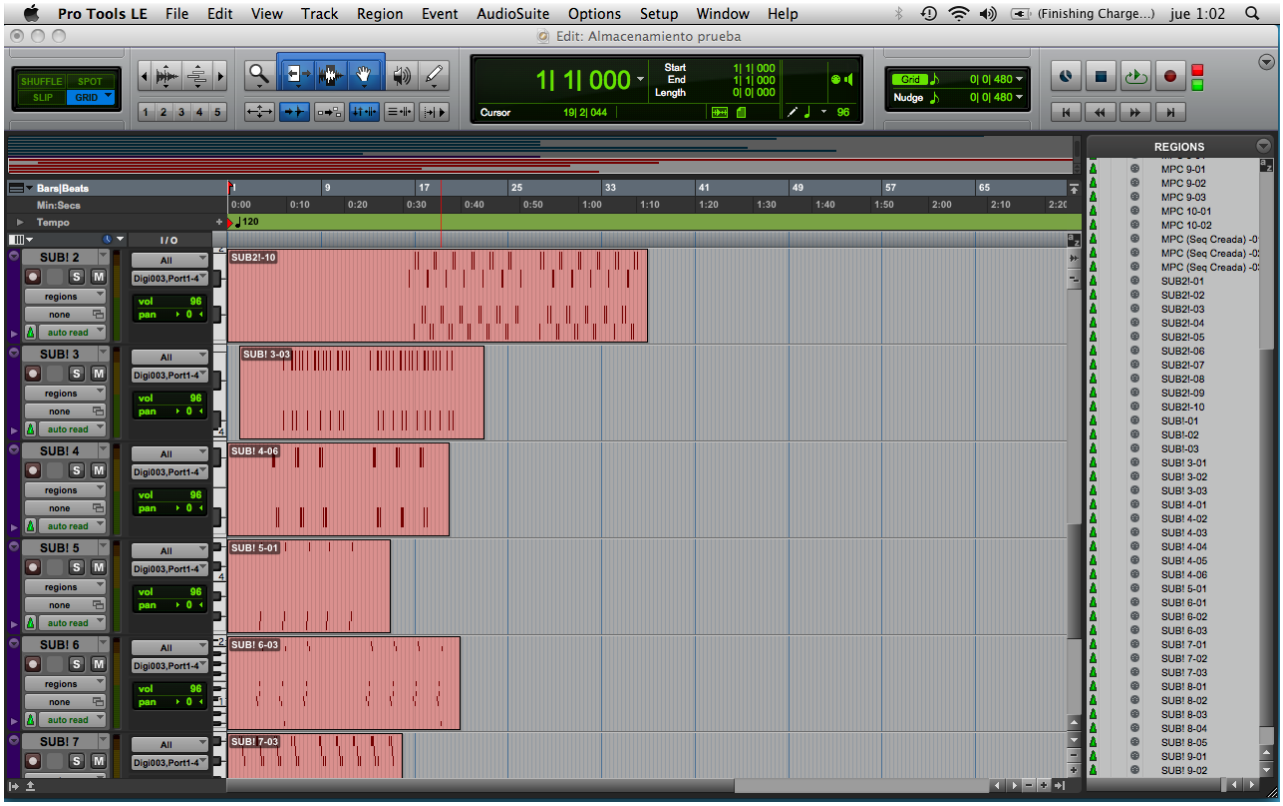
Segundo paso: Programamos una secuencia en tiempo real en nuestro secuenciador prueba (MPC o SUB!)

Tercer paso: Detenemos la reproducción y almacenamos la secuencia haciendo uso del sistema de almacenamiento del secuenciador.

Cuarto paso: Reproducimos la secuencia almacenada

Quinto paso: Comparación. La secuencia del segundo paso debe ser idéntica a la secuencia del cuarto paso.

Fig. 37. Pantallazo para primera parte de prueba de estabilidad en Pro tools.



Fuente: Propia.

5.2.3.1.2 Conexiones

	MIDI in	MIDI out
Electribe ES-1		Cable 1
MPC o SUB!	Cable 1	Cable 2
Digi 003	Cable 2	

Tabla 11. Tabla de conexionado para primera parte de prueba de estabilidad.

5.2.3.1.3 Resultados obtenidos:

Para ambos secuenciadores todas las secuencias reproducidas fueron idénticas a las secuencias almacenadas.

5.2.3.2 Segunda parte

5.2.3.2.1 Descripción de la prueba

Se configuro el SUB! Synt para recibir señal desde el secuenciador, la señal de audio producida por el sintetizador virtual fue enviada al DAW utilizando SoundFlower[22]. Simultáneamente, la señal MIDI proveniente del secuenciador fue grabada en un canal del DAW al que se le asigno el FM7, un sintetizador virtual de Native Instuments. Las dos señales de audio fueron comparadas para determinar posibles errores en la interacción de la MPC o el secuenciador SUB! con el sintetizador SUB!.

[22] Aplicación gratuita para mac que permite el ruteo interno de señal en el computador. http://cycling74.com/soundflower-landing-page/

Fig. 38. Pantallazo evaluación de la interaccion del secuenciador con el SUB! Y el FM7.

Fuente: Propia.

5.2.3.2.2 Resultados obtenidos

No se observo ningún error en la interacción de ninguno de los dos secuenciadores con ninguno de los dos sintetizadores usados.

5.2.4 Medición de la Cantidad de operaciones mínimas necesarias para crear, almacenar y reproducir secuencias.

Dispositivo	Crear	Almacenar	Reproducir
MPC 1000	0	4	4
SUB!	0	1	1

Tabla 12. Comparación cantidad de operaciones mínimas necesarias para crear, almacenar y reproducir secuencias MPC y SUB!

5.2.5 Medición de Peso y tamaño del diseño

Dispositivo	Ancho	Largo	Alto	Peso
MPC	33,1 cm	21,3 cm	5 cm	3.45 kg
SUB!	21,4 cm	22,1 cm	6 cm	1.3 kg

Tabla 13. Comparación peso y tamaño diseños SUB! y MPC 1000.

5.2.6 Capacidad de almacenamiento

Dispositivo	Capacidad de almacenamiento
MPC	6336 secuencias
SUB!	3 secuencias

Tabla 14. Comparación capacidad de almacenamiento MPC y SUB!

5.3 PRUEBAS DE SATISFACCIÓN

5.3.1 Diseño de la prueba

La población objetivo sobre la cual se aplicó esta prueba está definida como los músicos graduados de las facultades de música de la ciudad de Barranquilla entre los años 2009, 2010 y 2011. Actualmente existen tres programas de música a nivel profesional funcionando en la ciudad de Barranquilla: Programa de Música de la Universidad del Norte, Programa de Música de la Universidad del Atlántico y Programa de Música de la Universidad Reformada. De estas facultades solo la Universidad del Atlántico ha tenido graduados en los últimos tres años. Nuestra población se limito a este grupo de egresados, el cual según datos oficiales, corresponde a 65 personas.

A partir del listado de egresados proporcionado por la universidad[23], se realizó una encuesta sobre una muestra de 53 personas. El tamaño de la muestra fue calculado a partir de la formula estándar de muestreo aleatorio simple[24]:

- N = 65
- Za2 =1.645 (seguridad del 90%)
- p = Proporción esperada (en este caso 50% = 0.5)
- q = 1 – p (0.5)
- d = Margen de error de 5%

.

Se hicieron preguntas orientadas a identificar el nivel de satisfacción de los usuarios frente a: Aspectos físicos del secuenciador, funcionalidad creativa del secuenciador, funcionalidad interpretativa del secuenciador y aspectos generales del sintetizador. Estos aspectos fueron evaluados sobre una escala de 1 a 5 en la que uno es el valor más bajo y cinco el valor más alto. Los resultados fueron los siguientes:

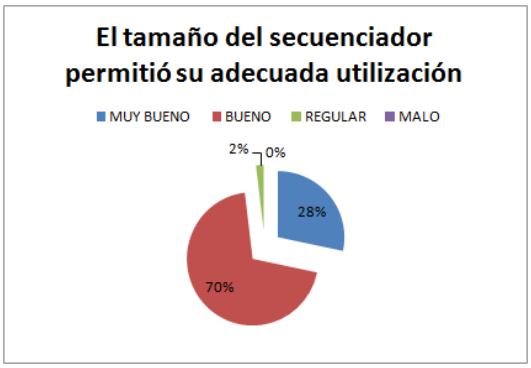
[23] Anexo 2

[24] Bouza, Carlos. Estadística Teoría Básica y Ejercicios. Editorial Felix Varela. La Habana. 2004

5.3.2 Resultados obtenidos

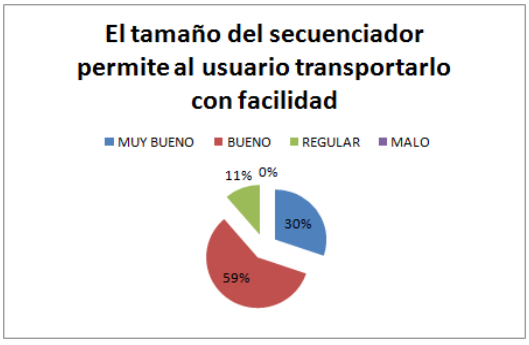
5.3.2.1 Aspectos físicos del secuenciador.

Fig. 39. Pregunta 1, aspectos físicos del secuenciador.



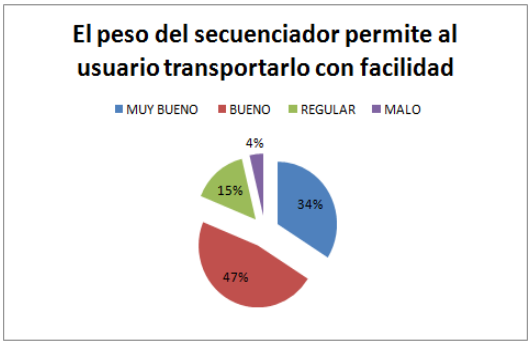
Fuente: Software Microsoft Excel.

Fig. 40. Pregunta 2, aspectos físicos del secuenciador.



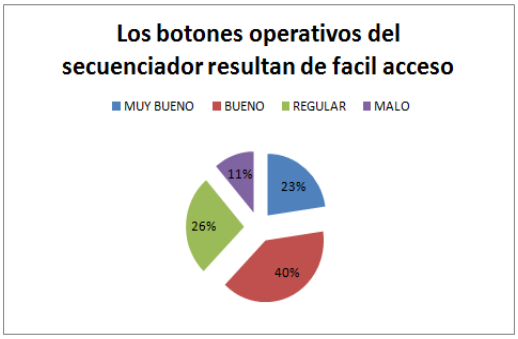
Fuente: Software Microsoft Excel.

Fig. 41. Pregunta 3, aspectos físicos del secuenciador.



Fuente: Software Microsoft Excel.

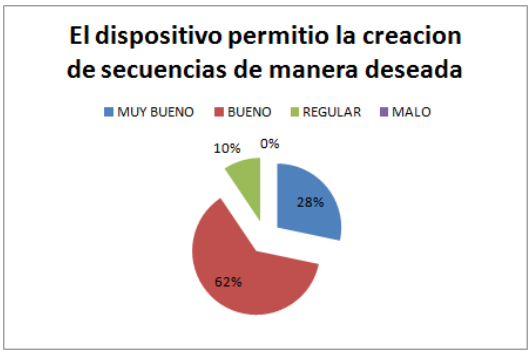
Fig. 42. Pregunta 4, aspectos físicos del secuenciador.



Fuente: Software Microsoft Excel.

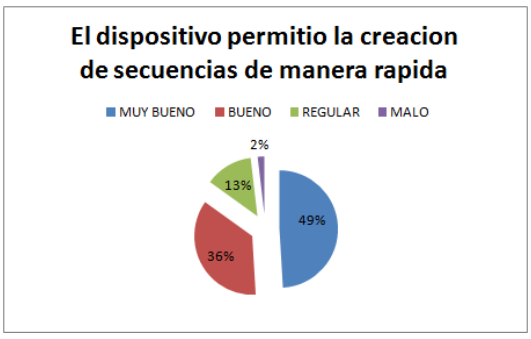
5.3.2.2 Funcionalidad creativa del secuenciador.

Fig. 43. Pregunta 1, funcionalidad creativa del secuenciador.



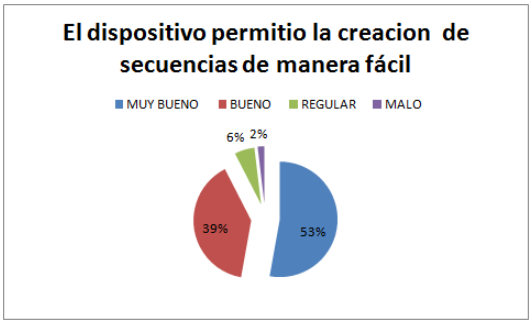
Fuente: Software Microsoft Excel.

Fig. 44. Pregunta 2, funcionalidad creativa del secuenciador.



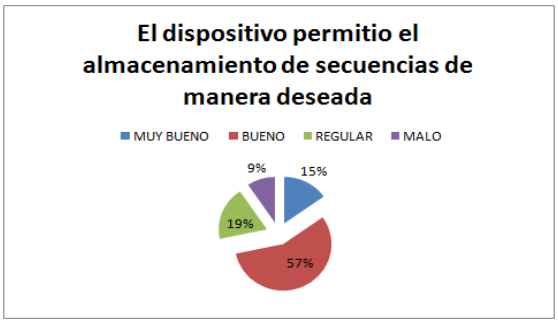
Fuente: Software Microsoft Excel.

Fig. 45. Pregunta 3, funcionalidad creativa del secuenciador.



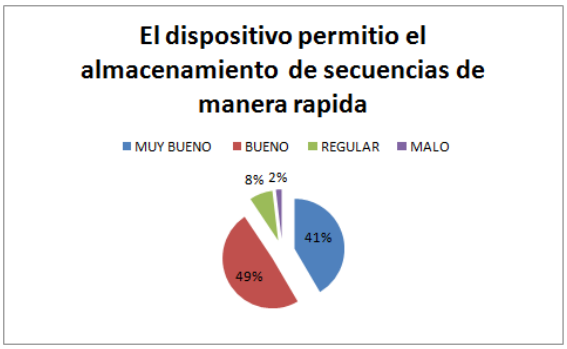
Fuente: Software Microsoft Excel.

Fig. 46. Pregunta 4, funcionalidad creativa del secuenciador.



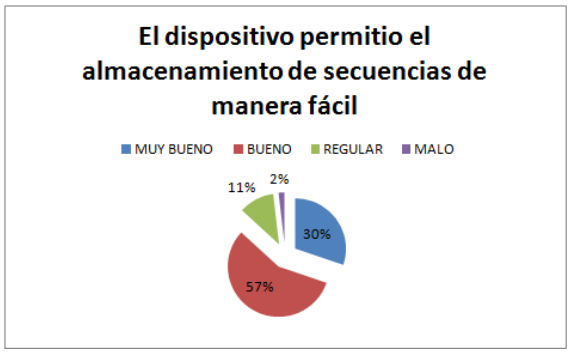
Fuente: Software Microsoft Excel.

Fig. 47. Pregunta 5, funcionalidad creativa del secuenciador.



Fuente: Software Microsoft Excel.

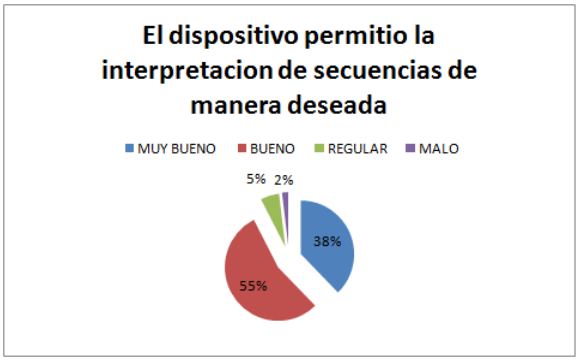
Fig. 48. Pregunta 6, funcionalidad creativa del secuenciador.



Fuente: Software Microsoft Excel.

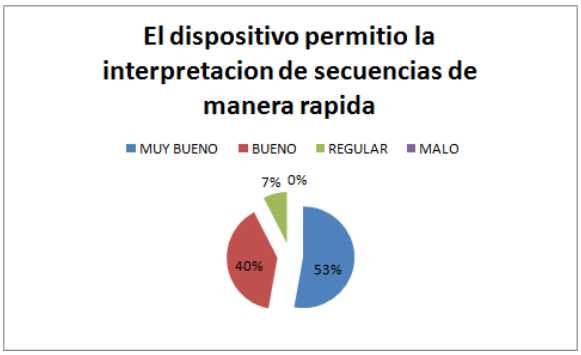
5.3.2.3 Funcionalidad interpretativa del secuenciador

Fig. 49. Pregunta 1, Funcionalidad interpretativa del secuenciador.



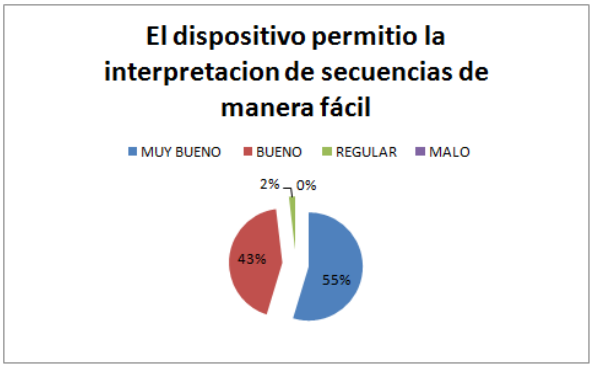
Fuente: Software Microsoft Excel.

Fig. 50. Pregunta 2, Funcionalidad interpretativa del secuenciador.



Fuente: Software Microsoft Excel.

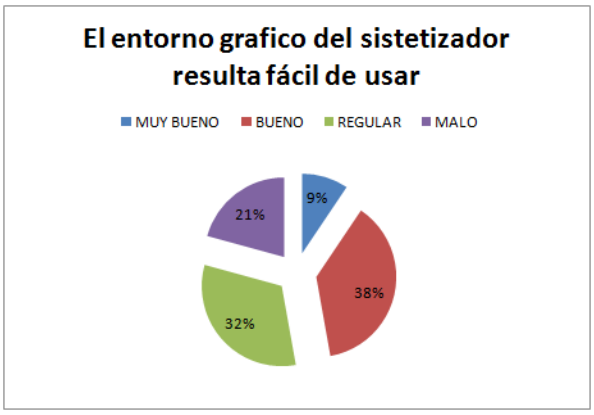
Fig. 51. Pregunta 3, Funcionalidad interpretativa del secuenciador.



Fuente: Software Microsoft Excel.

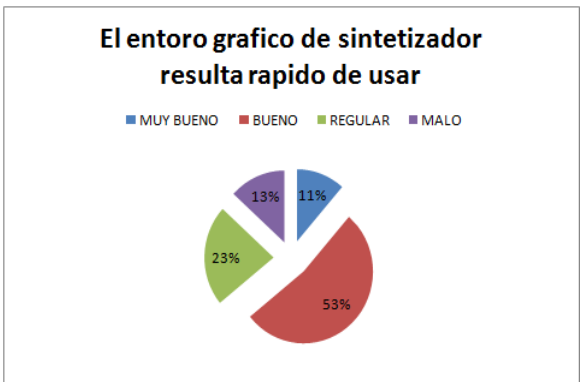
5.3.2.4 Aspectos generales del sintetizador

Fig. 52. Pregunta 1, Aspectos generales del sintetizador.



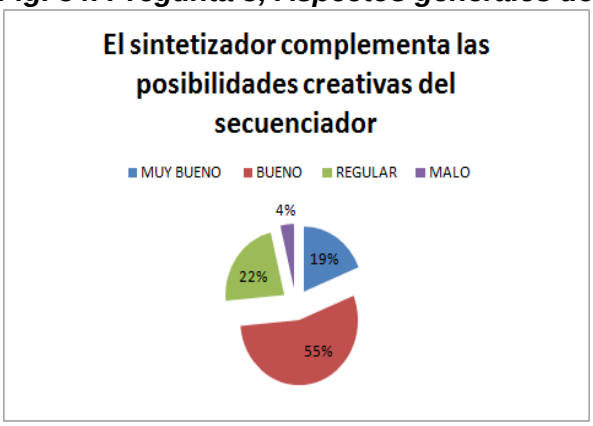
Fuente: Software Microsoft Excel.

Fig. 53. Pregunta 2, Aspectos generales del sintetizador.



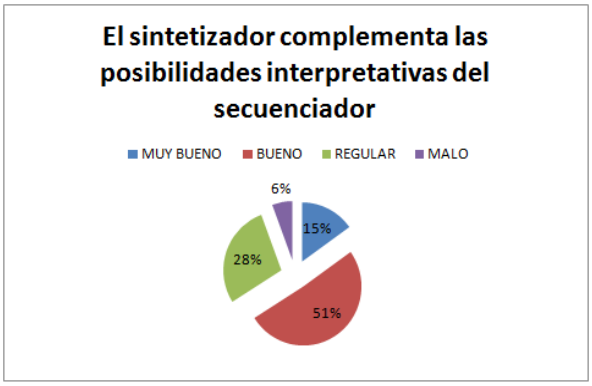
Fuente: Software Microsoft Excel.

Fig. 54. Pregunta 3, Aspectos generales del sintetizador.



Fuente: Software Microsoft Excel.

Fig. 55. Pregunta 4, Aspectos generales del sintetizador.



Fuente: Software Microsoft Excel.

6. CONCLUSIONES

El sistema SUB! es un instrumento electrónico que implementa la utilización de una matriz de tonos y una interfaz a modo de pedalera para permitir la creación, almacenamiento y reproducción de patrones rítmicos o armónicos en tiempo real. Los resultados obtenidos en la comparación de la cantidad de operaciones necesarias para trabajar con secuencias (tabla 12) y los resultados de la encuesta de satisfacción con respecto a la funcionalidad creativa e interpretativa del secuenciador (figuras de la 43 a 51) evidencian una fortaleza del diseño en cuanto a la rapidez y facilidad brindada para crear, almacenar y reproducir secuencias.

El comportamiento del dispositivo prueba ser estable y preciso al interactuar con diferentes tipos hardware y software, los resultados de las pruebas comparativas realizadas para evaluar la estabilidad del tempo y sincronía (fig. 33, 34, 36) muestran que durante un tiempo de reproducción continuo de hasta 15 minutos, nuestro secuenciador no evidencia fluctuaciones de tiempo superiores a ± 5 ms, este valor se considera adecuado teniendo en cuenta los resultados obtenidos para nuestro instrumento control y los márgenes de imprecisión encontrados para otros secuenciadores en investigaciones previas[25]. Complementariamente, La tabla 8 muestra los resultados del retraso de reloj MIDI para el dispositivo interactuando con diferentes combinaciones de software y hardware, la comparación de estos resultados con los obtenidos para nuestro instrumento control no revelan errores significativos en el comportamiento del SUB!

Entre las diferentes piezas de software para las cuales se realizaron pruebas de interacción se encuentra el SUB! synt, un sintetizador diseñado para generar sonido a partir de los mensajes recibidos por el secuenciador y complementar las posibilidades creativas e interpretativas del sistema. La prueba comparativa de compatibilidad para la interacción de este software con el secuenciador no arrojó errores; más información fue obtenida a partir de las pruebas de satisfacción realizadas, las cuales arrojaron resultados positivos en cuanto a las capacidades creativas e interpretativas del sintetizador (fig. 54 y 55) . La complejidad del entorno de Supercollider y una necesidad de mejorar la interfaz gráfica del sintetizador, también fueron aspectos evidenciados en los resultados de la encuesta (fig. 52 y 53).

Los resultados previamente enunciados, en conjunto con lo revelado por la encuesta de satisfacción en relación con la portabilidad del secuenciador (Fig. 40 y 41) y la prueba

[25] Particularmente la documentada en el artículo "Checking tempo stability of MIDI sequencers" presentado a la AES por Mario Perron en 1994.

de comparación de peso y tamaño entre el SUB! y la MPC (tabla 13), sugieren un buen desempeño del secuenciador en el ámbito de la producción musical relacionado con la interpretación en vivo u otras situaciones que requieran dispositivos fáciles de transportar y de usar. En cuanto al ámbito creativo, el secuenciador prueba ser eficiente no solo por la facilidad, rapidez y estabilidad brindada para programar (fig. 44 y 45) sino también por permitir a los usuarios programar secuencias de manera deseada o de acuerdo a sus necesidades creativas tal como lo demuestran los resultados ilustrados en la fig. 43.

En el proceso evaluativo, surge como principal desventaja del dispositivo su reducida capacidad de almacenamiento y edición de secuencias. A pesar de que no se obtuvieron errores en las pruebas de estabilidad y precisión en el almacenamiento y envío de mensajes MIDI con software y hardware externo, los resultados mostrados en la tabla 14 y la carencia de opciones de edición por parte del SUB! , sugieren un pobre desempeño por parte de este en ámbitos de la producción musical relacionados con procesos de edición. Esto puede ser atribuido al hecho de que, a diferencia de la MPC, el SUB! no está diseñado para funcionar como una estación de trabajo completa, sino como un instrumento musical para ser ejecutado en tiempo real.

Siempre ha existido una amplia brecha entre los instrumentos electrónicos y los músicos tradicionalistas; quizá por su complejidad o costo, los secuenciadores digitales rara vez son usados o siquiera conocidos por instrumentistas clásicos o jazzistas. El SUB! es un instrumento cuyo diseño minimalista y económico que prueba ser fácil y practico de usar, dando un paso hacia adelante en cerrar la brecha y contribuir, desde la ingeniería, con el proceso vital de generar nuevas sonoridades y formas de arte.

7. RECOMENDACIONES

La facilidad de interacción con la matriz de tonos puede verse mejorada mediante la utilización de una pantalla más grande, esto incrementaría los costos de fabricación del instrumento pero permitiría una mayor libertad al momento de programar las secuencias.

Una investigación conjunta con ingenieros mecánicos y diseñadores permitirá optimizar el material utilizado para la interfaz física o carcasa del instrumento (en especial la pedalera) en búsqueda de una mayor resistencia al impacto sin incrementar el factor peso.

Por último, se recomienda realizar pruebas del instrumento sobre usuarios sin conocimiento musical, en especial niños. Esto podría revelar nuevos horizontes en cuanto a las aplicaciones del SUB!, como por ejemplo , la posibilidad de ser utilizado como herramienta didáctica para niños.

Bibliografía

- M. Kusek, David; A Microcomputer Based Polyphonic Sequencer for Music Synthesizers; 1979.
- Hillen. Peter; A Microprocessor Based Sequencer For Voltage Controlled Electronic Music Synttetizers; 1977
- Mayer, Wetzei; A New Generation Of Musical Sequencers;
- Muro, Don; The Art Of Sequencing: A Step By Step Approach; Alfred Publishing; 1993.
- Miles Huber. David; The midi manual; 2007
- Rothsein Joseph, MIDI a comprensive introduction, 1997.
- Edward V. Ramirez, Melvyn Weiss; Microprocessor fundamentals: hardware and software, 1980.
- Wiley John, DAFX(Digital audio effects),2002.
- Powell John; How music works; 2010.
- Valdez, Pallas; Microcontroladores: Fundamentos y aplicaciones con PIC, 2007
- Wood, Smith; THE 'USI' I OR UNIVERSAL SYNTHESIZER INTERFACE; 1981.
- Mandado, Enrique; Sistemas Digitaltes; 2007.
- Panero, Julius; Dimensiones humanas para los espacios interiores; Gustavo Gili, 2007.
- Russ, Mark; Síntesis y muestreo; 2009
- Bouza, Carlos. Estadística Teoría Básica y Ejercicios. Editorial Felix Varela. La Habana. 2004
- Collins, Mike. Working with MIDI. Focal Press, 2011.
- Scott Wilson; David Cottle; Nick Collins *The SuperCollider Book*. The MIT Press, 2011.

ANEXOS.

Anexo A

Código Secuenciador 8x8.

```
#include <LiquidCrystal.h>   // Llamado a la librería para la pantalla alfanumérica
#include "T6963.h"           // Llamado a la librería para la pantalla grafica
#include "gfxdata.h"
#include "Times_New_Roman__14.h"

LiquidCrystal lcd(22, 23, 24, 25, 26, 27); // Asignación de puertos para la pantalla alfanumérica

T6963 LCD(240,128,6,32);// numero de pixeles y fondo para pantalla grafica

//Declaración de variables//
byte midi_start = 0xfa; // Asignación de variable para mensaje de start MIDI
byte midi_stop = 0xfc; // Asignación de variable para mensaje de stop MIDI
byte midi_clock = 0xf8;// Asignación de variable para el reloj proveniente del controlador maestro
byte midi_continue = 0xfb;// Asignación de variable para mensaje de continue MIDI
int play_flag = 0;//bandera controladora de recepción midi
byte datomidi; //Variable que recibe el valor proveniente del puerto RX
int a=-1; // variable que suma los mensajes recibidos por el reloj controlador maestro
int paso=0;// variable que marca los pasos que recorre la matriz
int x = 0; // variable que toma los valores en x del touch screen
int y = 0; // variable que toma los valores en y del touch screen
int v[8][8];//matriz principal donde se almacena los valores de asignación de notas
int G1[8][8];// matriz que almacena los valores de la matriz v para la memoria 1
int G2[8][8];// matriz que almacena los valores de la matriz v para la memoria 2
int G3[8][8];// matriz que almacena los valores de la matriz v para la memoria 3
int g1; //variable para q no vuelva a reescribí matriz G1
int g2;//variable para q no vuelva a reescribir matriz G2
int g3;//variable para q no vuelva a reescribir matriz G3
int fila; // variable que va recorriendo las filas por paso de la matriz
int val1;//variable para tomar valores de puerto análogo.
int val2;//variable para tomar valores de puerto análogo
int val3;//variable para tomar valores de puerto análogo
int val4;//variable para tomar valores de puerto análogo
int bot1=1;//bandera que controla en que memoria de almacenamiento esta
int bot2=0;//bandera que controla en que memoria de almacenamiento esta
int bot3=0;//bandera que controla en que memoria de almacenamiento esta
int almacenar1=0;// bandera que controla si hay algo almacenado
```

```
int almacenar2=0;// bandera que controla si hay algo almacenado
int almacenar3=0;// bandera que controla si hay algo almacenado

int escala;// variable que almacena valores para las escalas
int octava;// variable que almacena valores para las octavas
int time;// variable que almacena valores para cambios de tiempo
int tonalidad;// variable que almacena valores para las tonalidades


int j;// variable que recorre los ciclos para lleno de vectores
int i;// variable que recorre los ciclos para lleno de vectores

void setup(){
 Serial.begin(31250); //inicio de puerto serial, velocidad para envió y recepción de mensajes MIDI
    lcd.begin(16, 2); //inicio de pantalla alfanumérica
    lcd.setCursor(1, 0);// llevar cursor a posición
    lcd.print("8 STEP MATRIX");//imprimir mensaje

    LCD.Initialize();//inicio de pantalla grafica
    LCD.Rectangle(0,0,240,128,1); //Valores para impresión de la cuadrilla en la pantalla grafica
    LCD.createLine(0,15,240,15,1);
    LCD.createLine(0,31,240,31,1);
    LCD.createLine(0,47,240,47,1);
   LCD.createLine(0,63,240,63,1);
   LCD.createLine(0,79,240,79,1);
   LCD.createLine(0,95,240,95,1);
   LCD.createLine(0,111,240,111,1);
   LCD.createLine(26,0,26,128,1);
   LCD.createLine(27,0,27,128,1);
   LCD.createLine(56,0,56,128,1);
   LCD.createLine(57,0,57,128,1);
   LCD.createLine(86,0,86,128,1);
   LCD.createLine(87,0,87,128,1);
   LCD.createLine(116,0,116,128,1);
   LCD.createLine(117,0,117,128,1);
   LCD.createLine(146,0,146,128,1);
   LCD.createLine(147,0,147,128,1);
   LCD.createLine(176,0,176,128,1);
   LCD.createLine(177,0,177,128,1);
   LCD.createLine(207,0,207,128,1);
   LCD.createLine(208,0,208,128,1);

  pinMode(8,OUTPUT);//definición de puertos de salida para leds
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
```

```
}

void botones(){ //subprograma para almacenamiento
  if (bot1==1){   //programación para botón 1
       if (almacenar1==1){ // bandera para leer o escribir en matriz de memoria 1
    if(g1==0){
     for (i=0; i<=7;i++){// se almacenan los valores de matriz v en G1
       for (j=0; j<=7;i++){
      G1[i][j]= v[i][j];
     }
     }
     g1=1;// se cambia bandera para que no vuelva a reescribir

    }
      for (i=0; i<=7;i++){// lleva lo se escribe lo almacenado de G1 en v para su reproducción
        for (j=0; j<=7;i++){
        v[i][j]= G1[i][j];
       }

      }
     }

   }

  if (bot2==1){  // programación para el botón 2

   if (almacenar2==1){ //bandera leer o escribir
     if (g2==0){
     for (i=0; i<=7;i++){ // almacenamiento
        for (j=0; j<=7;i++){
         G2[i][j] = v[i][j] ;
           }
         }
      g2=1;
    }

   for (i=0; i<=7;i++){// lectura
      for (j=0; i<=7;i++){
      v[i][j]= G2[i][j];
      }
    }
    }

}
if (bot3==1){// programación botón 3
     if (almacenar3==1){// bandera leer o escribir
    if(g3==0){
     for (i=0; i<=7;i++){ //almacenamiento
      for (j=0; j<=7;i++){
       G3[i][j] = v[i][j] ;
        }
```

```
      }
        g3=1;
        }
   for (i=0; i<=7;i++){// lectura
    for (j=0; j<=7;i++){
       v[i][j]= G3[i][j];
    }
   }
    }

}
}

 void loop(){
   lcd.clear(); //aclarar pantalla
   if (digitalRead(6)==HIGH){ //lectura de botón para memoria 1
       bot1=1; // prende bandera de memoria 1 y apaga las demás
      bot2=0;
       bot3=0;
       digitalWrite(8,LOW);// prende led para memoria 1 y apaga las demás
       digitalWrite(9,LOW);
       digitalWrite(10,HIGH);
        lcd.setCursor(0, 0);
         lcd.print("Memo1");//imprime que memoria esta en pantalla

   }
    if (digitalRead(6)==HIGH){// lectura de botón para memoria 2
       bot1=0;// prende bandera de memoria 2 y apaga las demás
      bot2=1;
       bot3=0;
        digitalWrite(8,LOW);// prende led para memoria 2 y apaga las demás
       digitalWrite(9,HIGH);
       digitalWrite(10,LOW);
        lcd.setCursor(0, 0);
         lcd.print("Memo2");//imprime que memoria esta en pantalla
   }

 if (digitalRead(4)==HIGH){// lectura de botón para memoria 2
       bot1=0;// prende bandera de memoria 2 y apaga las demás
      bot2=0;
       bot3=1;
        digitalWrite(8,HIGH);// prende led para memoria 2 y apaga las demás
       digitalWrite(9,LOW);
       digitalWrite(10,LOW);
        lcd.setCursor(0, 0);
         lcd.print("Memo3");//imprime que memoria esta en pantalla
   }


  if (digitalRead(2)==HIGH){ //lectura botón de almacenar
     if (bot1==1){//activación de banderas dependiendo en que memoria este
```

```
 almacenar1=1;
}
if(bot2==1){

 almacenar2=1;
}
if(bot3==1){
 almacenar3=1;
}
  }

    if (digitalRead(3)==HIGH){//lectura botón de reprogramar
       if (bot1==1){//desactivación de banderas dependiendo en que memoria este
  almacenar1=0;

  g1=0;} //desactivación de bandera para que cuando vuelva almacenar, haga la reescritura.

   if (bot2==1){
 g2=0;//desactivación de bandera para que cuando vuelva almacenar, haga la reescritura.
  almacenar2=0;}

   if (bot3==1){
   g3=0;//desactivación de bandera para que cuando vuelva almacenar, haga la reescritura.

  almacenar3=0;}
    }

    botones(); //llamado a subprograma

 val1=analogRead(4);//lectura de puerto análogo
       lcd.setCursor(11, 0);
       if (val1>=0&&val1<=255){// asignación de valores para octava dependiendo de la lectura de puerto análogo
       lcd.print("Oct1" );// impression de valor
        octava=36;}
        if (val1>255&&val1<=510){
        lcd.print("Oct2" );
        octava=48;}
        if (val1>510&&val1<=765){
        lcd.print("Oct3" );
        octava=60;}
        if (val1>765&&val1<=1023){
        lcd.print("Oct4" );
        octava=72;}
 val2=analogRead(5);//lectura de puerto análogo
        lcd.setCursor(6, 0);
        if (val2>=0&&val2<=341){// asignación de valores para escala dependiendo de la lectura de puerto análogo
        lcd.print("Esc1" );// impresión de valor
        escala=1;}
        if (val2>341&&val2<=682){
```

```
        lcd.print("Esc2" );
        escala=2;}
        if (val2>682&&val2<=1023){
        lcd.print("Esc3" );
        escala=3;}
   val3=analogRead(6);//lectura de puerto análogo
        lcd.setCursor(9, 1);
        if (val3>=0&&val3<=85){// asignación de valores para tonalidad dependiendo de la lectura
de puerto análogo
        lcd.print("Ton C_" );// impresión de valor
        tonalidad=0;}
        if (val3>85&&val3<=170){
        lcd.print("Ton C#" );
        tonalidad=1;}
         if (val3>170&&val3<=255){
        lcd.print("Ton D_" );
        tonalidad=2;}
       if (val3>255&&val3<=341){
        lcd.print("Ton Eb" );
        tonalidad=3;}
      if (val3>341&&val3<=426){
      lcd.print("Ton E_" );
        tonalidad=4;}
      if (val3>426&&val3<=511){
     lcd.print("Ton F_" );
        tonalidad=5;}
      if (val3>511&&val3<=596){
      lcd.print("Ton F#" );
        tonalidad=6;}
      if (val3>596&&val3<=682){
    lcd.print("Ton G_" );
        tonalidad=7;}
      if (val3>682&&val3<=767){
     lcd.print("Ton G#" );
        tonalidad=8;}
      if (val3>767&&val3<=852){
    lcd.print("Ton A_" );
        tonalidad=9;}
      if (val3>852&&val3<=937){
    lcd.print("Ton Bb" );
        tonalidad=10;}
      if (val3>937&&val3<=1023){
    lcd.print("Ton B_" );
        tonalidad=11;}

    val4=analogRead(7);//lectura de puerto análogo
      lcd.setCursor(0, 1);
     if (val4>=0&&val4<=341){// asignación de valores para tonalidad dependiendo de la lectura
de puerto analogo
         lcd.print("time1/08" );}// impresión de valor
      if (val4>341&&val4<=682){
```

```
        lcd.print("time1/16" );}
      if (val4>682&&val4<=1023){
        lcd.print("time1/04" );}


  if(Serial.available() > 0) {// condición para activar recepción de datos solo cuando recibe
  datomidi = Serial.read(); //almacenamiento de dato en variable
 if(datomidi == midi_start) {
  play_flag = 1;//activacion de bandera si recibe mensaje de start
  }
 else if(datomidi == midi_continue) {
 play_flag = 1;//activacion de bandera si recibe mensaje de continue
 }
 else if(datomidi == midi_stop) {
   paso=0;//volver a empezar el recorrido de pasos cuando para
   a=-1;   //volver a empezar el conteo de datos recibidos cuando para
 play_flag = 0;//desactivacion de bandera si recibe mensaje de stop
 }


 else if((datomidi == midi_clock) && (play_flag == 1)) { //condición para hacer acción cuando recibe los mensajes del reloj


   a=a+1; //suma de conteo mensajes del reloj
   if((bot1==1&&almacenar1==0)||(bot2==1&&almacenar2==0)||(bot3==1&&almacenar3==0)){
//condición para lectura de pantalla solo si no hay nada almacenado en la memoria que se encuentre


  pinMode( A3, INPUT);      // asignación de puertos análogos para la recepción de valor x del touch screen
  pinMode( A1, INPUT );
  pinMode( A0, OUTPUT );
  digitalWrite( A0, LOW);
  pinMode( A2, OUTPUT );
  digitalWrite( A2, HIGH );
  x=analogRead(1 );   //  Lectura del valor x


  pinMode( A2, INPUT );      // asignación de puertos análogos para la recepción de valor x del touch screen
  pinMode( A0, INPUT );
  pinMode( A1, OUTPUT );
  digitalWrite( A1, HIGH);
  pinMode( A3, OUTPUT );
  digitalWrite( A3, LOW );
  y=analogRead(0 ); //lectura del valor y

  if (x>=860&&x<=940){//condiciones para clasificar los valores de posicion del touch screen
```

```
  if (y>=800&&y<=870){
  if (v[7][0]==0) //condición para asignar y pintar cuadro en pantalla
        {
          LCD.drawrectbyte(0,113,13,4,0b111111);// pintar cuadro en pantalla
          if (escala==1) { //condición para asignar escala
          v[7][0]=octava+tonalidad+12;} // asignación de valor con tonalidad octava y escala.
          if (escala==2) {
          v[7][0]=octava+tonalidad+16;}
          if (escala==3) {
          v[7][0]=octava+tonalidad+13;}
        }
        else { //condicion para desmarcar cuadro y quitar asignación de valor
          v[7][0]=0; //
          LCD.drawrectbyte(0,113,13,4,0b00000000); //borrar cuadro
        }
  }
   if (y>=735&&y<=790){// repetición de mismas condiciones para las 8X8 posiciones distintas
en la matriz.
  if (v[6][0]==0){
          LCD.drawrectbyte(0,97,13,4,0b111111);
          if (escala==1) {
          v[6][0]=octava+tonalidad+11;}
          if (escala==2) {
          v[6][0]=octava+tonalidad+14;}
          if (escala==3) {
          v[6][0]=octava+tonalidad+12;}
        }
        else {
          v[6][0]=0;
          LCD.drawrectbyte(0,97,13,4,0b00000000);
        }
  }
   if (y>=670&&y<=715){
   if (v[5][0]==0){
          LCD.drawrectbyte(0,81,13,4,0b111111);
          if (escala==1) {
          v[5][0]=octava+tonalidad+9;}
          if (escala==2) {
          v[5][0]=octava+tonalidad+12;}
          if (escala==3) {
          v[5][0]=octava+tonalidad+10;}
        }
        else {
          v[5][0]=0;
          LCD.drawrectbyte(0,81,13,4,0b00000000);
        }
  }
   if (y>=610&&y<=660){
   if (v[4][0]==0){
          LCD.drawrectbyte(0,65,13,4,0b111111);
           if (escala==1) {
```

```
          v[4][0]=octava+tonalidad+7;}
          if (escala==2) {
          v[4][0]=octava+tonalidad+9;}
          if (escala==3) {
          v[4][0]=octava+tonalidad+9;}
        }
        else {
          v[4][0]=0;
          LCD.drawrectbyte(0,65,13,4,0b00000000);
        }
}
 if (y>=530&&y<=600){
if (v[3][0]==0){
          LCD.drawrectbyte(0,49,13,4,0b111111);
          if (escala==1) {
          v[3][0]=octava+tonalidad+5;}
          if (escala==2) {
          v[3][0]=octava+tonalidad+7;}
          if (escala==3) {
          v[3][0]=octava+tonalidad+6;}
        }
        else {
          v[3][0]=0;
          LCD.drawrectbyte(0,49,13,4,0b00000000);
        }
}
 if (y>=450&&y<=520){
   if (v[2][0]==0){
          LCD.drawrectbyte(0,33,13,4,0b111111);
          if (escala==1) {
          v[2][0]=octava+tonalidad+4;}
          if (escala==2) {
          v[2][0]=octava+tonalidad+4;}
          if (escala==3) {
          v[2][0]=octava+tonalidad+5;}
        }
        else {
          v[2][0]=0;
          LCD.drawrectbyte(0,33,13,4,0b00000000);
        }
}
 if (y>=390&&y<=440){
if (v[1][0]==0){
          LCD.drawrectbyte(0,17,13,4,0b111111);
          if (escala==1) {
          v[1][0]=octava+tonalidad+2;}
          if (escala==2) {
          v[1][0]=octava+tonalidad+2;}
          if (escala==3) {
          v[1][0]=octava+tonalidad+1;}
        }
```

```
            else {
              v[1][0]=0;
              LCD.drawrectbyte(0,17,13,4,0b00000000);
            }
    }
     if (y>=310&&y<=380){
     if (v[0][0]==0){
                LCD.drawrectbyte(0,2,13,4,0b111111);
                if (escala==1) {
                v[0][0]=octava+tonalidad;}
                if (escala==2) {
                v[0][0]=octava+tonalidad;}
                if (escala==3) {
                v[0][0]=octava+tonalidad;}
            }
            else {
              v[0][0]=0;
              LCD.drawrectbyte(0,2,12,4,0b00000000);
            }
    }
  }
if (x>=780&&x<=850){
    if (y>=800&&y<=870){
   if (v[7][1]==0){
                LCD.drawrectbyte(30,113,13,4,0b111111);
                if (escala==1) {
                v[7][1]=octava+tonalidad+12;}
                if (escala==2) {
                v[7][1]=octava+tonalidad+16;}
                if (escala==3) {
                v[7][1]=octava+tonalidad+13;}
            }
            else {
              v[7][1]=0;
              LCD.drawrectbyte(30,113,13,4,0b00000000);
            }
    }
     if (y>=735&&y<=790){
     if (v[6][1]==0){
                LCD.drawrectbyte(30,97,13,4,0b111111);
                if (escala==1) {
                v[6][1]=octava+tonalidad+11;}
                if (escala==2) {
                v[6][1]=octava+tonalidad+14;}
                if (escala==3) {
                v[6][1]=octava+tonalidad+12;}
            }
            else {
              v[6][1]=0;
              LCD.drawrectbyte(30,97,13,4,0b00000000);
            }
```

```
 }
  if (y>=670&&y<=715){
if (v[5][1]==0){
            LCD.drawrectbyte(30,81,13,4,0b111111);
            if (escala==1) {
            v[5][1]=octava+tonalidad+9;}
            if (escala==2) {
            v[5][1]=octava+tonalidad+12;}
            if (escala==3) {
            v[5][1]=octava+tonalidad+10;}
          }
          else {
            v[5][1]=0;
            LCD.drawrectbyte(30,81,13,4,0b00000000);
          }
 }
  if (y>=610&&y<=660){
 if (v[4][1]==0){
            LCD.drawrectbyte(30,65,13,4,0b111111);
            if (escala==1) {
            v[4][1]=octava+tonalidad+7;}
            if (escala==2) {
            v[4][1]=octava+tonalidad+9;}
            if (escala==3) {
            v[4][1]=octava+tonalidad+9;}
          }
          else {
            v[4][1]=0;
            LCD.drawrectbyte(30,65,13,4,0b00000000);
          }
 }
  if (y>=530&&y<=600){
if (v[3][1]==0)
        {
            LCD.drawrectbyte(30,49,13,4,0b111111);
            if (escala==1) {
            v[3][1]=octava+tonalidad+5;}
            if (escala==2) {
            v[3][1]=octava+tonalidad+7;}
            if (escala==3) {
            v[3][1]=octava+tonalidad+6;}
          }
          else {
            v[3][1]=0;
            LCD.drawrectbyte(30,49,13,4,0b00000000);
          }
 }
  if (y>=450&&y<=520){
  if (v[2][1]==0){
            LCD.drawrectbyte(30,33,13,4,0b111111);
            if (escala==1) {
```

```
            v[2][1]=octava+tonalidad+4;}
            if (escala==2) {
            v[2][1]=octava+tonalidad+4;}
            if (escala==3) {
            v[2][1]=octava+tonalidad+5;}
          }
          else {
            v[2][1]=0;
            LCD.drawrectbyte(30,33,13,4,0b00000000);
          }
   }
    if (y>=390&&y<=440){
   if (v[1][1]==0){
            LCD.drawrectbyte(30,17,13,4,0b111111);
            if (escala==1) {
            v[1][1]=octava+tonalidad+2;}
            if (escala==2) {
            v[1][1]=octava+tonalidad+2;}
            if (escala==3) {
            v[1][1]=octava+tonalidad+1;}
          }
          else {
            v[1][1]=0;
            LCD.drawrectbyte(30,17,13,4,0b00000000);
          }
   }
    if (y>=310&&y<=380){
    if (v[0][1]==0){
             LCD.drawrectbyte(30,2,12,4,0b111111);
            if (escala==1) {
            v[0][1]=octava+tonalidad;}
            if (escala==2) {
            v[0][1]=octava+tonalidad;}
            if (escala==3) {
            v[0][1]=octava+tonalidad;}
          }
          else {
            v[0][1]=0;
            LCD.drawrectbyte(30,2,12,4,0b00000000);
          }
   }
  }

if (x>=690&&x<=770){
    if (y>=800&&y<=870){
   if (v[7][2]==0)
         {
            LCD.drawrectbyte(62,113,13,4,0b111111);
            if (escala==1) {
            v[7][2]=octava+tonalidad+12;}
            if (escala==2) {
```

```
            v[7][2]=octava+tonalidad+16;}
            if (escala==3) {
            v[7][2]=octava+tonalidad+13;}
          }
          else {
            v[7][2]=0;
            LCD.drawrectbyte(62,113,13,4,0b00000000);
          }
 }
  if (y>=735&&y<=790){
if (v[6][2]==0)
        {
            LCD.drawrectbyte(62,97,13,4,0b111111);
            if (escala==1) {
            v[6][2]=octava+tonalidad+11;}
            if (escala==2) {
            v[6][2]=octava+tonalidad+14;}
            if (escala==3) {
            v[6][2]=octava+tonalidad+12;}
          }
          else {
            v[6][2]=0;
            LCD.drawrectbyte(62,97,13,4,0b00000000);
          }
 }
  if (y>=670&&y<=715){
  if (v[5][2]==0)
        {
            LCD.drawrectbyte(62,81,13,4,0b111111);
            if (escala==1) {
            v[5][2]=octava+tonalidad+9;}
            if (escala==2) {
            v[5][2]=octava+tonalidad+12;}
            if (escala==3) {
            v[5][2]=octava+tonalidad+10;}
          }
          else {
            v[5][2]=0;
            LCD.drawrectbyte(62,81,13,4,0b00000000);
          }
 }
  if (y>=610&&y<=660){
 if (v[4][2]==0)
        {
            LCD.drawrectbyte(62,65,13,4,0b111111);
            if (escala==1) {
            v[4][2]=octava+tonalidad+7;}
            if (escala==2) {
            v[4][2]=octava+tonalidad+9;}
            if (escala==3) {
            v[4][2]=octava+tonalidad+9;}
```

```
          }
          else {
            v[4][2]=0;
            LCD.drawrectbyte(62,65,13,4,0b00000000);
          }
 }
  if (y>=530&&y<=600){
if (v[3][2]==0)
        {
            LCD.drawrectbyte(62,49,13,4,0b111111);
             if (escala==1) {
             v[3][2]=octava+tonalidad+5;}
             if (escala==2) {
             v[3][2]=octava+tonalidad+7;}
             if (escala==3) {
             v[3][2]=octava+tonalidad+6;}
          }
          else {
            v[3][2]=0;
            LCD.drawrectbyte(62,49,13,4,0b00000000);
          }
 }
  if (y>=450&&y<=520){
  if (v[2][2]==0)
        {
            LCD.drawrectbyte(62,33,13,4,0b111111);
             if (escala==1) {
             v[2][2]=octava+tonalidad+4;}
             if (escala==2) {
             v[2][2]=octava+tonalidad+4;}
             if (escala==3) {
             v[2][2]=octava+tonalidad+5;}
          }
          else {
            v[2][2]=0;
            LCD.drawrectbyte(62,33,13,4,0b00000000);
          }
      }


  if (y>=390&&y<=440){
 if (v[1][2]==0)
        {
            LCD.drawrectbyte(62,17,13,4,0b111111);
             if (escala==1) {
             v[1][2]=octava+tonalidad+2;}
             if (escala==2) {
             v[1][2]=octava+tonalidad+2;}
             if (escala==3) {
             v[1][2]=octava+tonalidad+1;}
          }
```

```
            else {
              v[1][2]=0;
              LCD.drawrectbyte(62,17,13,4,0b00000000);
            }
          }

     if (y>=310&&y<=380){
    if (v[0][2]==0)
            {
               LCD.drawrectbyte(62,2,12,4,0b111111);
              if (escala==1) {
              v[0][2]=octava+tonalidad;}
              if (escala==2) {
              v[0][2]=octava+tonalidad;}
              if (escala==3) {
              v[0][2]=octava+tonalidad;}
            }
            else {
              v[0][2]=0;
              LCD.drawrectbyte(62,2,12,4,0b00000000);
            }
        }
  }

if (x>=600&&x<=680){
    if (y>=800&&y<=870){
    if (v[7][3]==0)
           {
              LCD.drawrectbyte(92,113,13,4,0b111111);
              if (escala==1) {
              v[7][3]=octava+tonalidad+12;}
              if (escala==2) {
              v[7][3]=octava+tonalidad+16;}
              if (escala==3) {
              v[7][3]=octava+tonalidad+13;}
            }
            else {
              v[7][3]=0;
              LCD.drawrectbyte(92,113,13,4,0b00000000);
            }
    }
     if (y>=735&&y<=790){
    if (v[6][3]==0)
           {
              LCD.drawrectbyte(92,97,13,4,0b111111);
              if (escala==1) {
              v[6][3]=octava+tonalidad+11;}
              if (escala==2) {
              v[6][3]=octava+tonalidad+14;}
              if (escala==3) {
              v[6][3]=octava+tonalidad+12;}
```

```
        }
        else {
          v[6][3]=0;
          LCD.drawrectbyte(92,97,13,4,0b00000000);
        }
}
 if (y>=670&&y<=715){
if (v[5][3]==0)
       {
          LCD.drawrectbyte(92,81,13,4,0b111111);
          if (escala==1) {
          v[5][3]=octava+tonalidad+9;}
          if (escala==2) {
          v[5][3]=octava+tonalidad+12;}
          if (escala==3) {
          v[5][3]=octava+tonalidad+10;}
        }
        else {
          v[5][3]=0;
          LCD.drawrectbyte(92,81,13,4,0b00000000);
        }
}
 if (y>=610&&y<=660){
 if (v[4][3]==0)
       {
          LCD.drawrectbyte(92,65,13,4,0b111111);
          if (escala==1) {
          v[4][3]=octava+tonalidad+7;}
          if (escala==2) {
          v[4][3]=octava+tonalidad+9;}
          if (escala==3) {
          v[4][3]=octava+tonalidad+9;}
        }
        else {
          v[4][3]=0;
          LCD.drawrectbyte(92,65,13,4,0b00000000);
        }
     }

 if (y>=530&&y<=600){
if (v[3][3]==0)
       {
          LCD.drawrectbyte(92,49,13,4,0b111111);
          if (escala==1) {
          v[3][3]=octava+tonalidad+5;}
          if (escala==2) {
          v[3][3]=octava+tonalidad+7;}
          if (escala==3) {
          v[3][3]=octava+tonalidad+6;}
        }
        else {
```

```
          v[3][3]=0;
          LCD.drawrectbyte(92,49,13,4,0b00000000);
        }
}
 if (y>=450&&y<=520){
 if (v[2][3]==0)
       {
          LCD.drawrectbyte(92,33,13,4,0b111111);
           if (escala==1) {
           v[2][3]=octava+tonalidad+4;}
           if (escala==2) {
           v[2][3]=octava+tonalidad+4;}
           if (escala==3) {
           v[2][3]=octava+tonalidad+5;}
        }
        else {
          v[2][3]=0;
          LCD.drawrectbyte(92,33,13,4,0b00000000);
        }
}
 if (y>=390&&y<=440){
 if (v[1][3]==0)
       {
          LCD.drawrectbyte(92,17,13,4,0b111111);
           if (escala==1) {
           v[1][3]=octava+tonalidad+2;}
           if (escala==2) {
           v[1][3]=octava+tonalidad+2;}
           if (escala==3) {
           v[1][3]=octava+tonalidad+1;}
        }
        else {
          v[1][3]=0;
          LCD.drawrectbyte(92,17,13,4,0b00000000);
        }
}
 if (y>=310&&y<=380){
if (v[0][3]==0)
        {
           LCD.drawrectbyte(92,2,12,4,0b111111);
           if (escala==1) {
           v[0][3]=octava+tonalidad;}
           if (escala==2) {
           v[0][3]=octava+tonalidad;}
           if (escala==3) {
           v[0][3]=octava+tonalidad;}
        }
        else {
          v[0][3]=0;
          LCD.drawrectbyte(92,2,12,4,0b00000000);
        }
```

```
  }
 }

if (x>=510&&x<=600){
   if (y>=795&&y<=870){
   if (v[7][4]==0)
          {
             LCD.drawrectbyte(122,113,13,4,0b111111);
             if (escala==1) {
             v[7][4]=octava+tonalidad+12;}
             if (escala==2) {
             v[7][4]=octava+tonalidad+16;}
             if (escala==3) {
             v[7][4]=octava+tonalidad+13;}
          }
          else {
            v[7][4]=0;
            LCD.drawrectbyte(122,113,13,4,0b00000000);
          }
   }
    if (y>=725&&y<=770){
   if (v[6][4]==0)
          {
             LCD.drawrectbyte(122,97,13,4,0b111111);
             if (escala==1) {
             v[6][4]=octava+tonalidad+11;}
             if (escala==2) {
             v[6][4]=octava+tonalidad+14;}
             if (escala==3) {
             v[6][4]=octava+tonalidad+12;}
          }
          else {
            v[6][4]=0;
            LCD.drawrectbyte(122,97,13,4,0b00000000);
          }
   }
    if (y>=660&&y<=715){
    if (v[5][4]==0)
          {
             LCD.drawrectbyte(122,81,13,4,0b111111);
             if (escala==1) {
             v[5][4]=octava+tonalidad+9;}
             if (escala==2) {
             v[5][4]=octava+tonalidad+12;}
             if (escala==3) {
             v[5][4]=octava+tonalidad+10;}
          }
          else {
            v[5][4]=0;
            LCD.drawrectbyte(122,81,13,4,0b00000000);
          }
```

```
}
 if (y>=580&&y<=640){
if (v[4][4]==0)
       {
         LCD.drawrectbyte(122,65,13,4,0b111111);
          if (escala==1) {
          v[4][4]=octava+tonalidad+7;}
          if (escala==2) {
          v[4][4]=octava+tonalidad+9;}
          if (escala==3) {
          v[4][4]=octava+tonalidad+9;}
        }
        else {
          v[4][4]=0;
          LCD.drawrectbyte(122,65,13,4,0b00000000);
        }
}
 if (y>=510&&y<=570){
if (v[3][4]==0)
       {
         LCD.drawrectbyte(122,49,13,4,0b111111);
          if (escala==1) {
          v[3][4]=octava+tonalidad+5;}
          if (escala==2) {
          v[3][4]=octava+tonalidad+7;}
          if (escala==3) {
          v[3][4]=octava+tonalidad+6;}
        }
        else {
          v[3][4]=0;
          LCD.drawrectbyte(122,49,13,4,0b00000000);
        }
}
 if (y>=430&&y<=500){
if (v[2][4]==0)
       {
         LCD.drawrectbyte(122,33,13,4,0b111111);
          if (escala==1) {
          v[2][4]=octava+tonalidad+4;}
          if (escala==2) {
          v[2][4]=octava+tonalidad+4;}
          if (escala==3) {
          v[2][4]=octava+tonalidad+5;}
        }
        else {
          v[2][4]=0;
          LCD.drawrectbyte(122,33,13,4,0b00000000);
        }
}
 if (y>=360&&y<=420){
if (v[1][4]==0)
```

```
          {
            LCD.drawrectbyte(122,17,13,4,0b111111);
             if (escala==1) {
             v[1][4]=octava+tonalidad+2;}
             if (escala==2) {
             v[1][4]=octava+tonalidad+2;}
             if (escala==3) {
             v[1][4]=octava+tonalidad+1;}
          }
          else {
            v[1][4]=0;
            LCD.drawrectbyte(122,17,13,4,0b00000000);
          }
   }
    if (y>=270&&y<=350){
  if (v[0][4]==0)
          {
             LCD.drawrectbyte(122,2,12,4,0b111111);
             if (escala==1) {
             v[0][4]=octava+tonalidad;}
             if (escala==2) {
             v[0][4]=octava+tonalidad;}
             if (escala==3) {
             v[0][4]=octava+tonalidad;}
          }
          else {
            v[0][4]=0;
            LCD.drawrectbyte(122,2,12,4,0b00000000);
          }
   }
 }
if (x>=425&&x<=505){
   if (y>=795&&y<=870){
  if (v[7][5]==0)
          {
             LCD.drawrectbyte(152,113,13,4,0b111111);
             if (escala==1) {
             v[7][5]=octava+tonalidad+12;}
             if (escala==2) {
             v[7][5]=octava+tonalidad+16;}
             if (escala==3) {
             v[7][5]=octava+tonalidad+13;}
          }
          else {
            v[7][5]=0;
            LCD.drawrectbyte(152,113,13,4,0b00000000);
          }
   }
    if (y>=725&&y<=770){
   if (v[6][5]==0)
          {
```

```
          LCD.drawrectbyte(152,97,13,4,0b111111);
          if (escala==1) {
          v[6][5]=octava+tonalidad+11;}
          if (escala==2) {
          v[6][5]=octava+tonalidad+14;}
          if (escala==3) {
          v[6][5]=octava+tonalidad+12;}
        }
        else {
          v[6][5]=0;
          LCD.drawrectbyte(152,97,13,4,0b00000000);
        }
}
 if (y>=660&&y<=715){
if (v[5][5]==0)
       {
          LCD.drawrectbyte(152,81,13,4,0b111111);
          if (escala==1) {
          v[5][5]=octava+tonalidad+9;}
          if (escala==2) {
          v[5][5]=octava+tonalidad+12;}
          if (escala==3) {
          v[5][5]=octava+tonalidad+10;}
        }
        else {
          v[5][5]=0;
          LCD.drawrectbyte(152,81,13,4,0b00000000);
        }
}
 if (y>=580&&y<=640){
 if (v[4][5]==0)
       {
          LCD.drawrectbyte(152,65,13,4,0b111111);
          if (escala==1) {
          v[4][5]=octava+tonalidad+7;}
          if (escala==2) {
          v[4][5]=octava+tonalidad+9;}
          if (escala==3) {
          v[4][5]=octava+tonalidad+9;}
        }
        else {
          v[4][5]=0;
          LCD.drawrectbyte(152,65,13,4,0b00000000);
        }
}
 if (y>=510&&y<=570){
if (v[3][5]==0)
       {
          LCD.drawrectbyte(152,49,13,4,0b111111);
          if (escala==1) {
          v[3][5]=octava+tonalidad+5;}
```

```
          if (escala==2) {
          v[3][5]=octava+tonalidad+7;}
          if (escala==3) {
          v[3][5]=octava+tonalidad+6;}
        }
        else {
          v[3][5]=0;
          LCD.drawrectbyte(152,49,13,4,0b00000000);
        }
}
 if (y>=430&&y<=500){
if (v[2][5]==0)
       {
          LCD.drawrectbyte(152,33,13,4,0b111111);
          if (escala==1) {
          v[2][5]=octava+tonalidad+4;}
          if (escala==2) {
          v[2][5]=octava+tonalidad+4;}
          if (escala==3) {
          v[2][5]=octava+tonalidad+5;}
        }
        else {
          v[2][5]=0;
          LCD.drawrectbyte(152,33,13,4,0b00000000);
        }
}
 if (y>=360&&y<=420){
if (v[1][5]==0)
       {
          LCD.drawrectbyte(152,17,13,4,0b111111);
          if (escala==1) {
          v[1][5]=octava+tonalidad+2;}
          if (escala==2) {
          v[1][5]=octava+tonalidad+2;}
          if (escala==3) {
          v[1][5]=octava+tonalidad+1;}
        }
        else {
          v[1][5]=0;
          LCD.drawrectbyte(152,17,13,4,0b00000000);
        }
}
 if (y>=270&&y<=350){
 if (v[0][5]==0)
        {
           LCD.drawrectbyte(152,2,12,4,0b111111);
          if (escala==1) {
          v[0][5]=octava+tonalidad;}
          if (escala==2) {
          v[0][5]=octava+tonalidad;}
          if (escala==3) {
```

```
            v[0][5]=octava+tonalidad;}
          }
          else {
            v[0][5]=0;
            LCD.drawrectbyte(152,2,12,4,0b00000000);
          }
  }
 }
if (x>=330&&x<=420){
  if (y>=795&&y<=870){
   if (v[7][6]==0)
        {
           LCD.drawrectbyte(182,113,13,4,0b111111);
           if (escala==1) {
           v[7][6]=octava+tonalidad+12;}
           if (escala==2) {
           v[7][6]=octava+tonalidad+16;}
           if (escala==3) {
           v[7][6]=octava+tonalidad+13;}
          }
          else {
            v[7][6]=0;
            LCD.drawrectbyte(182,113,13,4,0b00000000);
          }
  }
   if (y>=725&&y<=770){
   if (v[6][6]==0)
        {
           LCD.drawrectbyte(182,97,13,4,0b111111);
           if (escala==1) {
           v[6][6]=octava+tonalidad+11;}
           if (escala==2) {
           v[6][6]=octava+tonalidad+14;}
           if (escala==3) {
           v[6][6]=octava+tonalidad+12;}
          }
          else {
            v[6][6]=0;
            LCD.drawrectbyte(182,97,13,4,0b00000000);
          }
  }
   if (y>=660&&y<=715){
 if (v[5][6]==0)
        {
           LCD.drawrectbyte(182,81,13,4,0b111111);
           if (escala==1) {
           v[5][6]=octava+tonalidad+9;}
           if (escala==2) {
           v[5][6]=octava+tonalidad+12;}
           if (escala==3) {
           v[5][6]=octava+tonalidad+10;}
```

```
          }
          else {
            v[5][6]=0;
            LCD.drawrectbyte(182,81,13,4,0b00000000);
          }
 }
  if (y>=580&&y<=640){
if (v[4][6]==0)
        {
            LCD.drawrectbyte(182,65,13,4,0b111111);
             if (escala==1) {
             v[4][6]=octava+tonalidad+7;}
             if (escala==2) {
             v[4][6]=octava+tonalidad+9;}
             if (escala==3) {
             v[4][6]=octava+tonalidad+9;}
          }
          else {
            v[4][6]=0;
            LCD.drawrectbyte(182,65,13,4,0b00000000);
          }
 }
  if (y>=510&&y<=570){
 if (v[3][6]==0)
        {
            LCD.drawrectbyte(182,49,13,4,0b111111);
             if (escala==1) {
             v[3][6]=octava+tonalidad+5;}
             if (escala==2) {
             v[3][6]=octava+tonalidad+7;}
             if (escala==3) {
             v[3][6]=octava+tonalidad+6;}
          }
          else {
            v[3][6]=0;
            LCD.drawrectbyte(182,49,13,4,0b00000000);
          }
 }
  if (y>=430&&y<=500){
 if (v[2][6]==0)
        {
            LCD.drawrectbyte(182,33,13,4,0b111111);
             if (escala==1) {
             v[2][6]=octava+tonalidad+4;}
             if (escala==2) {
             v[2][6]=octava+tonalidad+4;}
             if (escala==3) {
             v[2][6]=octava+tonalidad+5;}
          }
          else {
            v[2][6]=0;
```

```
            LCD.drawrectbyte(182,33,13,4,0b00000000);
          }
        }

  if (y>=360&&y<=420){
 if (v[1][6]==0)
        {
          LCD.drawrectbyte(182,17,13,4,0b111111);
           if (escala==1) {
           v[1][6]=octava+tonalidad+2;}
           if (escala==2) {
           v[1][6]=octava+tonalidad+2;}
           if (escala==3) {
           v[1][6]=octava+tonalidad+1;}
         }
         else {
           v[1][6]=0;
           LCD.drawrectbyte(182,17,13,4,0b00000000);
         }
  }
   if (y>=270&&y<=350){
  if (v[0][6]==0)
         {
            LCD.drawrectbyte(182,2,12,4,0b111111);
           if (escala==1) {
           v[0][6]=octava+tonalidad;}
           if (escala==2) {
           v[0][6]=octava+tonalidad;}
           if (escala==3) {
           v[0][6]=octava+tonalidad;}
         }
         else {
           v[0][6]=0;
           LCD.drawrectbyte(182,2,12,4,0b00000000);
         }
  }
}
 if (x>=200&&x<=320){
  if (y>=795&&y<=870){
  if (v[7][7]==0)
         {
           LCD.drawrectbyte(212,113,13,4,0b111111);
           if (escala==1) {
           v[7][7]=octava+tonalidad+12;}
           if (escala==2) {
           v[7][7]=octava+tonalidad+16;}
           if (escala==3) {
           v[7][7]=octava+tonalidad+13;}
         }
         else {
           v[7][7]=0;
```

```
          LCD.drawrectbyte(212,113,13,4,0b00000000);
        }
}
 if (y>=725&&y<=770){
if (v[6][7]==0)
       {
          LCD.drawrectbyte(212,97,13,4,0b111111);
          if (escala==1) {
          v[6][7]=octava+tonalidad+11;}
          if (escala==2) {
          v[6][7]=octava+tonalidad+14;}
          if (escala==3) {
          v[6][7]=octava+tonalidad+12;}
        }
        else {
          v[6][7]=0;
          LCD.drawrectbyte(212,97,13,4,0b00000000);
        }
}
 if (y>=660&&y<=715){
if (v[5][7]==0)
       {
          LCD.drawrectbyte(212,81,13,4,0b111111);
          if (escala==1) {
          v[5][7]=octava+tonalidad+9;}
          if (escala==2) {
          v[5][7]=octava+tonalidad+12;}
          if (escala==3) {
          v[5][7]=octava+tonalidad+10;}
        }
        else {
          v[5][7]=0;
          LCD.drawrectbyte(212,81,13,4,0b00000000);
        }
}
 if (y>=580&&y<=640){
if (v[4][7]==0)
       {
          LCD.drawrectbyte(212,65,13,4,0b111111);
          if (escala==1) {
          v[4][7]=octava+tonalidad+7;}
          if (escala==2) {
          v[4][7]=octava+tonalidad+9;}
          if (escala==3) {
          v[4][7]=octava+tonalidad+9;}
        }
        else {
          v[4][7]=0;
          LCD.drawrectbyte(212,65,13,4,0b00000000);
        }
}
```

```
  if (y>=510&&y<=570){
if (v[3][7]==0)
        {
          LCD.drawrectbyte(212,49,13,4,0b111111);
           if (escala==1) {
           v[3][7]=octava+tonalidad+5;}
           if (escala==2) {
           v[3][7]=octava+tonalidad+7;}
           if (escala==3) {
           v[3][7]=octava+tonalidad+6;}
         }
         else {
           v[3][7]=0;
           LCD.drawrectbyte(212,49,13,4,0b00000000);
         }
 }
  if (y>=430&&y<=500){
  if (v[2][7]==0)
        {
          LCD.drawrectbyte(212,33,13,4,0b111111);
           if (escala==1) {
           v[2][7]=octava+tonalidad+4;}
           if (escala==2) {
           v[2][7]=octava+tonalidad+4;}
           if (escala==3) {
           v[2][7]=octava+tonalidad+5;}
         }
         else {
           v[2][7]=0;
           LCD.drawrectbyte(212,33,13,4,0b00000000);
         }
 }
  if (y>=360&&y<=420){
if (v[1][7]==0)
        {
          LCD.drawrectbyte(212,17,13,4,0b111111);
           if (escala==1) {
           v[1][7]=octava+tonalidad+2;}
           if (escala==2) {
           v[1][7]=octava+tonalidad+2;}
           if (escala==3) {
           v[1][7]=octava+tonalidad+1;}
         }
         else {
           v[1][7]=0;
           LCD.drawrectbyte(212,17,13,4,0b00000000);
         }
 }
  if (y>=270&&y<=350){
 if (v[0][7]==0)
        {
```

```
            LCD.drawrectbyte(212,2,12,4,0b111111);
            if (escala==1) {
            v[0][7]=octava+tonalidad;}
            if (escala==2) {
            v[0][7]=octava+tonalidad;}
            if (escala==3) {
            v[0][7]=octava+tonalidad;}
          }
          else {
            v[0][7]=0;
            LCD.drawrectbyte(212,2,12,4,0b00000000);
          }
  }
 }


   }



   if (a%time==0){//condición para llevar a al envió de mensajes midi dependiendo del valor de tiempo y contador
      if(paso==0){//condición para leer variable de tiempo cuando empiece el recorrido de la matriz
        val4=analogRead(7);
     if (val4>=0&&val4<=341){
        time=24;}
      if (val4>341&&val4<=682){
        time=12;}
      if (val4>682&&val4<=1023){
        time=48;}
     }
     if(digitalRead(7)==HIGH){//lectura de switch para enviar o no mensajes MIDI
      for(fila=0;fila<=7;fila++){// recorrido de filas por paso para el envió de mensajes
       if (v[fila][paso]!=0){
      noteOn(0x90,v[fila][paso],0x45); //llamado al subprograma para el envió de mensajes midi con valores de nota asignados
         }
       }
     }
     paso=paso+1;// aumento de variable para seguir a el siguiente paso
      if(paso==8){//condición para volver a empezar conteo de pasos cuando este llegue al octavo paso

        paso=0;
      }

     }


    }
```

```
  }
 }

 void noteOn(int cmd, int pitch, int velocity) { //subprograma para envío de mensajes MIDI
Serial.write(cmd); //envío de byte de activación de nota
  Serial.write(pitch);//envío de byte de valor de nota
  Serial.write(velocity);//envío de mensaje de velocity de nota
}
```

Anexo B.

Listado de egresados de licenciatura en música en universidad del atlántico.

Año 2011	
DAVID ELIAS	PACHECO MIRANDA
DANIEL ARTURO	CERON FREYLE
MYRIAM DEL SOCORRO	NARVAEZ DE SALAS
LICETTE ESTHER	BARRIOS POLO
JONATHAN JOSE	CABRERA MENDOZA
DIANA CAROLINA	VERGARA ZAMBRANO
JENNIFFER	GALVAN PULIDO
YURANY JANETH	NIEBLES CARDONA
YAIR ENRIQUE	MENDOZA CASTRO
KEWIN LUIS	ARTEAGA COGOLLO
EUNICE ISABEL	ECHAVARRIA RIVAS
MELINA	LIZARAZO GARZON
AMANDA PAOLA	CORONELL MORALES
KELLYS JOHANA	GARCIA FONTALVO

YOHANA CAROLINA	ALVARADO ECHEVERRIA
ZULIANYS DEL CARMEN	RINCON PEÑA
ALVARO	ELLES PEREZ
RAFAEL GERVASIO	CUADRADO MORA
JEFFREY JOSEPH	HERRERA ACOSTA
JUAN MIGUEL	RIOS REDONDO
LUIS ANGEL	TRUJILLO GONZALEZ
MIGUEL ENRIQUE	NISPERUZA PALMA
WILLIAM JUNIOR	ESPEJO LOPEZ
RONALD DAVID	ARTUZ RUA
ANDREA MILENA	AREVALO ROQUE
DARWIN EDUARDO	NIÑO DURAN
FRANK JOSUE	RUIZ GOMEZ