FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

EVENTOS SONOROS ALEATORIOS: DISEÑO DE ALGORITMO PARA COMPOSICIÓN MUSICAL

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero de Sonido y Acústica

Profesor Guía

Msc. Paúl Adrián Cabezas Yánez

Autor

Santiago Andrés Nieto Flores

Año

2019

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido el trabajo, “Eventos sonoros aleatorios: diseño de algoritmo para composición musical”, a través de reuniones periódicas con el estudiante Santiago Andrés Nieto Flores en el semestre 201920, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

_____

Paúl Adrián Cabezas Yánez

Master en Industrias Creativas en Música y Sonido

CI: 171918954-8

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, “Eventos sonoros aleatorios: diseño de algoritmo para composición musical”, del estudiante, Santiago Andrés Nieto Flores, en el semestre 201920, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

José Antonio Álvarez Torres Yépez

Magíster en Musicología

CI: 170823226-7

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

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

_____

Santiago Andrés Nieto Flores

CI:172456685-4

AGRADECIMIENTOS

A mi familia por todo el apoyo brindado durante mi carrera universitaria. También a todos mis docentes que me aportaron con todos sus conocimiento para poder formarme como un profesional.

DEDICATORIA

Dedico este trabajo a mi madre, por siempre estar a mi lado y ser la fuente de inspiración para poder luchar día a día por ser una mejor persona.

RESUMEN

El presente trabajo de titulación está enfocado en la implementación de una aplicación que produzca diferentes composiciones musicales con una estructura definida mediante el uso de un modelo de composición algorítmica. Para cumplir con este objetivo se decidió trabajar con el lenguaje de programación orientado a objetos llamado “MAX”, el mismo que permitió diseñar dos plataformas que crean música por aleatoriedad y linealidad. En el primer caso se construyó una plataforma que a partir de la selección de una escala mayor o menor se generan patrones melódicos o armónicos distintos. Para la segunda plataforma se ocupan las herramientas de control y creación de audio para ejecutar una canción base que siempre generará la misma salida.

Los resultados obtenidos se analizaron aplicando una transcripción de las piezas creadas en los programas en una partitura donde se puede evidenciar que se generan obras con un patrón bastante diferente en cada ejecución pero que en todos los casos se respetan los parámetros musicales definidos por el usuario.

ABSTRACT

The present degree work is focused on the implementation of an application that produces different musical compositions with a defined structure through the use of a model of algorithmic composition. To achieve this goal, it was decided to work with the object-oriented programming language called "MAX", the same that allowed designing two platforms that create music by randomness and linearity. In the first case, a platform was constructed that, from the selection of a major or minor scale, different melodic or harmonic patterns are generated. For the second platform, the control and audio creation tools are used to execute a base song that will always generate the same output.

The results obtained were analyzed applying a transcription of the pieces created in the programs in a score where it can be evidenced that musical pieces are generated with a quite different pattern in each execution but in all cases the musical parameters defined by the user are respected.

ÍNDICE

1. CAPÍTULO I: INTRODUCCIÓN 1
   1.1. Antecedentes 1
   1.2. Objetivos 4
      1.2.1. Objetivo general 4
      1.2.2. Objetivos específicos 4
   1.3. Alcance 5
   1.4. Justificación 5
   1.5. Hipótesis 5
2. CAPÍTULO 2: MARCO TEÓRICO 6
   2.1. Sistema interactivo musical 6
   2.2. Composición algorítmica 6
   2.3. Teoría musical 7
      2.3.1. Intervalo 7
      2.3.2. Escala 7
      2.3.3. Modos 8
      2.3.5. Armonía 9
      2.3.6. Ritmo 9
   2.4. MAX/MSP 9
      2.4.1. Herramientas utilizadas en MAX/MSP 10
   2.5. Protocolo MIDI 17
3. CAPÍTULO III: METODOLOGÍA 17
   3.1. Programa basado en eventos aleatorios 18
      3.1.1. Nota musical base 18
      3.1.2. Selección de la escala 19
      3.1.3. Configuración del tempo 21
      3.1.4. Cambio de sección 24
      3.1.5. Generación de la melodía 25
      3.1.6. Generación de la armonía 26
      3.1.7. Drum Machine 28
      3.1.8. Conexión Ableton Live 10 31
      3.1.9. Retorno de audio procedente de Ableton 10 32
      3.1.10. Sección de efectos 33
      3.1.11. Interfaz gráfica 34
   3.2. Programa basado en la linealidad 35
      3.2.1. Sección melódica y armónica 36
      3.2.2. Sección rítmica 37
      3.2.3. Interfaz gráfica 37
4. CAPÍTULO IV: ANÁLISIS DE RESULTADOS 38
   4.1. Composición Aleatoria 38

4.1.1. Análisis Melódico....39
4.1.2. Análisis Armónico....41
4.2. Composición Lineal....42
4.2.1. Análisis Melódico y Armónico....43
5. CONCLUSIONES Y RECOMENDACIONES....44
5.1. Conclusiones....44
5.2. Recomendaciones....45
REFERENCIAS....46
ANEXOS....49

1. CAPÍTULO I: INTRODUCCIÓN

1.1. Antecedentes

La composición musical es un arte que presenta varios elementos y herramientas que permiten a un compositor plasmar sus ideas en una obra, la cual será transmitida hacia un público e interpretada por músicos.

Tradicionalmente, el medio en el que se plasma toda esta información es una partitura que se convierte en una clase de “archivo” que almacena todos los datos necesarios para una buena ejecución de una determinada canción.

Sin embargo, los avances informáticos y el desarrollo de nuevas tecnologías han generado nuevas formas de composición musical como es el caso del *“Live Coding*” o codificación en vivo, que es el proceso por el que se efectúa la creación de una pieza musical en tiempo real mediante la escritura de código en forma de algoritmo en un determinado software de programación. En general, los artistas que emplean esta técnica construyen sus composiciones desde cero y a partir de un código inicial se procede a modificar, copiar, pegar o pausar el mismo para generar diferentes experiencias en el oyente y que de esta manera se produzca una especie de "codeo morfología", ya que el código y la música evolucionan juntos en un proceso entretejido observado por la audiencia (Magnusson, 2011).

Este nuevo enfoque y aplicación de los algoritmos en la música permite la implementación de un conjunto de reglas o instrucciones que pueden conducir a soluciones adecuadas para el desarrollo de técnicas aplicables para la composición de un tema musical como para la síntesis de sonido, muestreo o reconocimiento de obras musicales. Sin embargo, cabe resaltar que aunque el uso de códigos simplifique el trabajo, muchas veces la falta de factores humanos en el proceso de la composición algorítmica conduce a la aparición de malos productos musicales por lo que muchos defensores de la composición

algorítmica deciden, durante la ejecución del algoritmo, incluir factores humanos en la determinación de la calidad de la composiciones. Este tipo de composición se llama composición interactiva, por lo que, la opinión humana es un factor utilizado para medir la calidad de la composición y muchas veces se puede demostrar que este enfoque a menudo da mejores resultados en comparación con la composición automática (netamente el uso de códigos) , debido al hecho de que incluso un gran número de las reglas y restricciones en los algoritmos no pueden ser lo suficientemente buenas para evaluar la calidad de una melodía (Matić, 2010).

Por otra parte, otro tipo de composición musical utilizando algoritmos se basa en los sistemas de música auto generativa en donde existen reglas explícitas para que los códigos puedan generar una canción o un objeto de salida diferente con cada iteración. Cada programa ejecutado con esta lógica de programación presenta diversos grados de autonomía dependiendo de lo que el programador necesite y de esta forma se pueden crear sistemas algorítmicos que solo requieran la especificación de parámetros iniciales de salida y entrada para generar una canción. No obstante, se puede involucrar a un compositor interactuando con el sistema durante una presentación en vivo y obtener resultado similar al de un director musical controlando una orquesta (Eigenfeldt & Pasquier, 2011).

De igual forma, Los problemas de tiempo, latencia y frecuencia de muestreo han sido de gran importancia y preocupación para la generación de música por computadora y es por esta razón, que la mayor parte de los sistemas dedicados para la codificación en vivo contienen soluciones en cuanto a cómo se programan los eventos musicales a tiempo real. Y aunque estos sistemas muestran la mayor estabilidad posible para la reproducción musical hay varias limitantes para algunos compositores ya que no hay mucha profundidad en algunos términos de teoría musical como los micro tonos que en la ejecución musical común son normales y fáciles de incluir en un partitura (Magnusson, 2013).

Adicionalmente a esta problemática, otro aspecto a observar es la comparación que se hace entre un “Digital Audio Workstation” y un software de programación musical ya que ambos programas pueden desarrollar las mismas operaciones y ofrecer a un músico los mismos elementos como la capacidad de reproducir un “*Loop”,* para un artista de codificación en vivo resulta más beneficiosos usar programas como *ChucK* para aprovechar todo su potencial creativo y definir sus representaciones musicales, aunque para esto se debe conocer bien todas las instrucciones necesarias para poder ejecutar instrucciones en dicho programa (Collins, Blackwell, & Collins, 2011).

Es por esta razón, que las técnicas de codificación en vivo están aumentando drásticamente de manera que en el transcurso de la última década se ha producido un aumento del número de practicantes, el cual ha sido acompañado por un creciente número de idiomas y nuevos entornos de programación que ya no solo se enfocan en el uso de programas de escritorio sino que se han desarrollado herramientas web como GIBBER que es un software de “*Live Coding”* basado en JavaScript y que permite la ejecución de la síntesis de audio de alto nivel y la secuenciación en navegadores web como Google Chrome entre otros (Roberts & Kuchera-Morin, 2012).

Adicionalmente a lo mencionado anteriormente, también se están desarrollando nuevas interfaces de comunicación para la expresión musical como es el caso de ReacTacT que es una interfaz capaz de realizar una exploración en la aplicación de nuevos controladores novedosos para interactuar con los sistemas de música por computadora, ya que en la actualidad es común ver entornos de producción aumentados por una amplia variedad de controladores de música disponibles y que extienden el nivel de control que ofrece el controlador al músico para poder comunicar sus intenciones de forma más expresiva, mientras que el público experimenta una interpretación dinámica y atractiva en relación con una en donde el músico está estático detrás de una computadora portátil (Nuanáin & Sullivan, 2014).

Y no solo artistas comunes están usando estas herramientas de "*live coding*". La superestrella de la música dance Deadmau5 y el guitarrista de improvisación Derek Bailey representan, a través de su escritura y práctica, dos enfoques muy diferentes para actuar en vivo. Al considerar críticamente la práctica de la codificación en vivo en relación a cómo la computadora portátil y el software se pueden emplear como instrumentos musicales (Parkinson & Bell, 2015).

Inclusive hay experimentos como la experiencia de tener un programador en vivo entre músicos acústicos en una orquesta para que mediante extractos de audio de algunas sesiones grabadas se puedan plantear discusiones sobre qué tan profundamente los músicos deben entender el código para interactuar de manera efectiva en el contexto de la improvisación libre (Goulart & Antar, 2015). Para finalizar, cabe resaltar la creación de un movimiento llamado "*algorave"* que ha recibido exposición internacional en los últimos dos años, incluyendo una serie de conciertos en Europa y más allá, y la cobertura de la prensa en una serie de medios de comunicación (Collins & McLean, 2014).

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un algoritmo que produzca una composición musical mediante eventos aleatorios.

1.2.2. Objetivos específicos

Identificar el modelo para composición algorítmica que mejor se adapte para la creación de un tema musical.

Determinar el lenguaje de programación más apropiado que permita el desarrollo del algoritmo.

Analizar las diferentes composiciones producidas por el programa desarrollado y compararlo con una composición estática inicial.

1.3. Alcance

La investigación tiene como proyección la creación de un programa desarrollado en el software MAX/MSP, que genere una composición musical distinta en cada ejecución y que en su estructura se tenga una parte A, B, A.
El programa contará con una interfaz gráfica que permitirá la selección de algunos parámetros modificables como la escala, su nota musical base, el tempo, el volumen de los instrumentos y por último una opción de colocación de efectos.

1.4. Justificación

La creación de composiciones musicales a través de un algoritmo fomentará el uso de las nuevas tecnologías de información como un medio para que cualquier persona pueda expresar sus ideas y construir cualquier tipo de composición musical con tan solo escribir líneas de código específicas en el editor de texto de un determinado programa, de manera que se puedan generar melodías o armonías y modificar las mismas con diferentes sonidos e instrumentos.

Adicionalmente, se puede empezar a originar un movimiento en el país que reúna a las personas que usan esta forma de composición para de esta forma crear espacios donde se puedan efectuar conciertos en vivo utilizando computadores y códigos simulando los llamados "*algoraves*" que se desarrollan en varias ciudades de Europa y América del norte.

1.5. Hipótesis

El resultado principal de esta investigación está enfocado en la creación de un programa que genere composiciones musicales, donde cada nota musical, acorde y ritmo se seleccionen y se reproduzcan de manera aleatoria, obteniendo

diferentes resultados sonoros que puedan servir como una manifestación artística o como base de inspiración para músicos y compositores.

2. CAPÍTULO 2: MARCO TEÓRICO

En el presente capítulo, se presentan y explican las definiciones teóricas referentes a la comprensión de varios temas relacionados con la composición musical y algorítmica.

2.1. Sistema interactivo musical

Un sistema interactivo musical es cualquier dispositivo de hardware o software que permite a las personas efectuar cualquier tarea relacionada con la composición o ejecución de música en tiempo real.

Estos sistemas están formados en su mayoría por la combinación de elementos como botones, teclas, *switches* o sensores que facilitan la manipulación y variación de componentes musicales como la armonía, melodía y ritmo según las preferencias del usuario (Manzo, 2012).

2.2. Composición algorítmica

Este término está ligado al proceso por el cual un compositor emplea una serie de instrucciones y operaciones dentro de un programa de computador con el objetivo de lograr crear una pieza musical (Simoni & Dannenberg, 2013).

Existen varios modelos computacionales para diseñar e implementar un algoritmo útil para la composición musical. Dos de los métodos utilizados en esta investigación son:

- **Método Estocástico:** En este método, los resultados obtenidos a la salida van a ir adquiriendo resultados diferentes cada vez que se ejecute el programa y esto se debe a que la mayor parte de los procesos que se llevan a cabo se encuentran desarrollándose en base a eventos aleatorios y distribuciones probabilísticas (Collado, 2014).

- **Método Determinista:** En este modelo no se maneja ningún principio relacionado con la teoría del azar sino que se procesan datos y variables para que se produzca siempre una misma salida a partir de datos de entrada fijos. En este caso se trabajan con patrones y secuencias establecidas y que se mantienen de manera constante para que todos los eventos que sucedan en el futuro estén ligados con acontecimientos previos. (Collado, 2014).

2.3. Teoría musical

Comprende la investigación de los diferentes componentes que permiten a través del desarrollo de diversas técnicas, la agrupación armónica de notas para llegar a estructurar una obra musical (Allen, 2018).

2.3.1. Intervalo

Es un parámetro que se encarga de determinar cuánta distancia existe entre dos notas musicales. La principal función de esta herramienta es la de colaborar en la formación de escalas y acordes.

Se ocupan a los tonos y semitonos como las unidades de medida para poder determinar dicha separación (Roca & Molina, 2006).

2.3.2. Escala

El resultado provocado por la unión ordenada de varias notas musicales separadas por una cantidad determinada de pasos se denomina escala. Existen diferentes maneras de juntar estos tonos (T) y semitonos (S), pero para esta investigación se utilizaron únicamente dos tipos de combinaciones que son la mayor y la menor. En primer lugar la escala mayor se forma por el el uso del siguiente patrón: T-T-S-T-T-T-S. Por otro lado, su relativa menor se rige al siguiente esquema: T-S-T-T-S-T-T. (Allen, 2018).

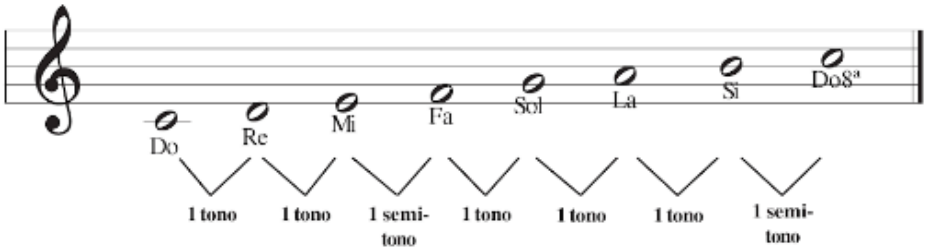


Figura 1. Representación de la escala de Do Mayor.

Tomado de (IGER, 2009).

2.3.3. Modos

Para organizar de manera sistemática los intervalos de varios sonidos que forman parte de una octava se utilizan a los modos (Peñalver Vilar, 2009). Esta herramienta fue desarrollada en la época de la antigua Grecia donde se presentaron los primeros sistemas modales que se convirtieron en la base para que en la actualidad se puedan utilizar los modos diatónicos o naturales que se pueden observar en la figura 2.

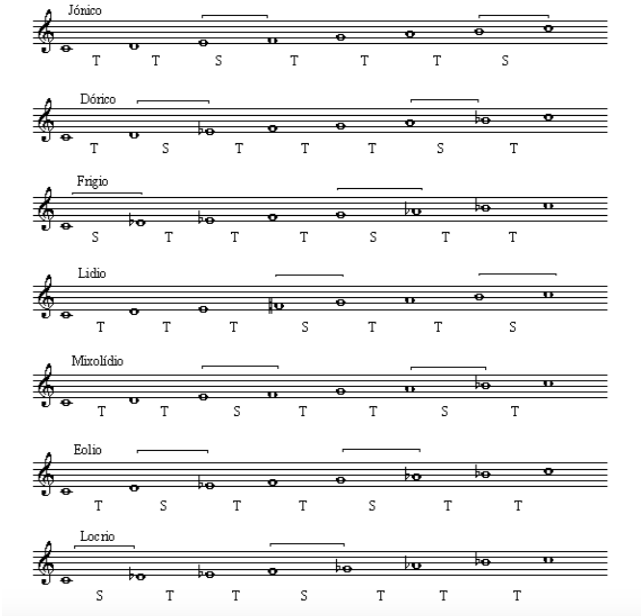


Figura 2. Representación de los modos naturales.

Tomado de (Peñalver Vilar, 2009).

2.3.4. Melodía

La agrupación de notas musicales de diferentes alturas pero con una duración específica similar da origen a una unidad independiente y equilibrada que se conoce como melodía (Castro Lobo, 2003).

2.3.5. Armonía

Involucra al estudio de todos los enlaces y conexiones que se pueden establecer entre dos o más sonidos que se estén ejecutando de manera simultánea. Además, cabe destacar que este término se convierte en la base desde la cual se puede crear la parte melódica. Por último, se pueden aplicar las funciones armónicas para añadir a la composición una tensión, relajación o equilibrio, dependiendo de lo que el compositor quiera transmitir (Gabis et al, 2006).

2.3.6. Ritmo

Se considera al ritmo como el elemento base de la composición musical, que está directamente ligado al estudio de la distribución en función del tiempo de las notas y silencios. La principal sensación que produce este elemento en los oyentes es la sensación de movimiento (Castro Lobo, 2003).

2.4. MAX/MSP

Para realizar la parte práctica de esta investigación se utilizó el programa MAX/MSP que funciona como un entorno gráfico interactivo donde se puede procesar datos de audio y multimedia en tiempo real.

Este software usa un lenguaje de programación llamado MAX donde de una forma sencilla se conectan objetos gráficos *(objects)* a través del uso de cables virtuales (*patch cords).* Estos objetos pueden realizar cálculos, producir o procesar sonidos, renderizar imágenes o ser configurados para ser utilizados como una interfaz gráfica. Además, gracias a la capacidad de procesamiento

referentes a la síntesis de sonido y señales, se pueden crear sintetizadores, sampleadores o efectos de procesamiento como reverberaciones, ecos o moduladores (Cipriani & Giri, 2010).

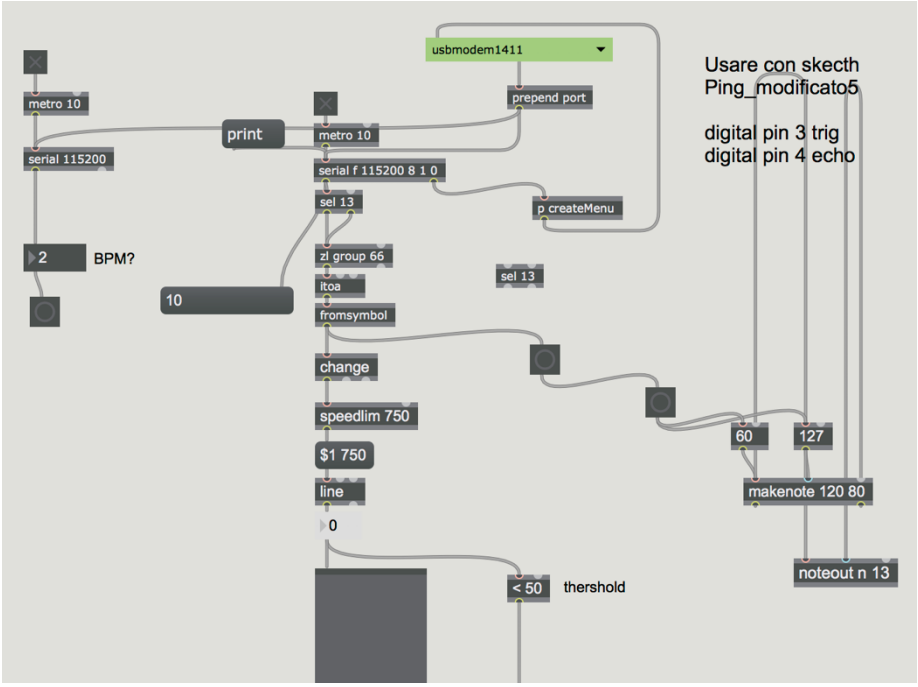


Figura 3. Entorno visual del programa MAX/MSP.

Tomado de (Cycling74, s.f).

2.4.1. Herramientas utilizadas en MAX/MSP

Al tratarse de un entorno netamente visual se crean varios objetos dentro de una pantalla en blanco nombrada como *patch* donde a sus vez se pueden ir encapsulando la suma de algunos procesos dentro de *subpatches* para mantener un orden óptico estético.
Gracias a la amplia librería que presenta MAX/MSP se pudieron utilizar los siguientes objetos para desarrollar los algoritmos.

BUTTON: Envía una explosión cuando se hace clic en él. Adicionalmente, parpadea cuando está recibiendo un mensaje.

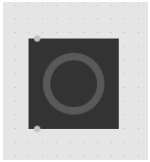


Figura 4. Objeto *button.*

TOGGLE: Funciona como un conmutador. Envía un 0 a la salida cuando se encuentra apagado y un 1 cuando se enciende.

Figura 5. Objeto *toggle.*

NUMBER: Facilita el ingreso y salida de números enteros.

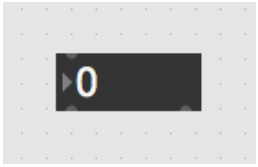


Figura 6. Objeto *number.*

MESSAGE: Muestra en pantalla cualquier mensaje que facilite el manejo de argumentos específicos.

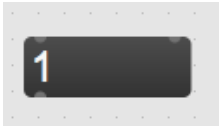


Figura 7. Objeto *message.*

COMMENT: Permite colocar un texto con fines explicativos.

Figura 8. Objeto *comment.*

RANDOM: Genera números aleatorios dentro de un rango delimitado por un argumento.

Figura 9. Objeto random.

ROUTE: Intenta hacer coincidir el primer argumento de un mensaje con los argumentos propios del objeto de ruta. La salida de la derecha pasa cualquier mensaje que no coincida con ninguna otra opción.

Figura 10. Objeto *route.*

UNPACK: Separa y envía cada elemento proveniente de un mensaje a una salida independiente.

Figura 11. Objeto *unpack.*

SEL: Produce de forma selectiva un pulso en respuesta a cualquier entrada que coincida con sus argumentos.

Figura 12. Objeto *sel.*

COUNTER: Realiza un conteo delimitado por un grupo de argumentos.

Figura 13. Objeto *counter.*

INT: Almacena cualquier número entero.

Figura 14. Objeto *int.*

GATE: Decide si un mensaje determinado va a ser transportado a través de una salida específica.

Figura 15. Objeto *gate.*

MAKENOTE: Da salida a un mensaje de activación de notas MIDI emparejado con un valor de velocidad y un mensaje de desactivación de notas.

Figura 16. Objeto *makenote.*

NOTEOUT: Transmite mensajes de notas musicales a un dispositivo MIDI.

Figura 17. Objeto *noteout.*

METRO: Actúa como un metrónomo que genera impulsos en un intervalo regular especificado.

Figura 18. Objeto *metro.*

TEMPO: Produce una salida que puede ser controlada por argumentos que modifiquen los pulsos por minuto y las divisiones en las que se van a ejecutar determinadas notas musicales.

Figura 19. Objeto *tempo.*

LOADMESS: Muestra y envía un mensaje automáticamente al momento en que se abre el programa.

Figura 20. Objeto *loadmess.*

LOADBANG: Crea y envía un pulso automáticamente cuando se abre el archivo.

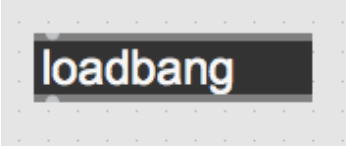


Figura 21. Objeto *loadbang.*

KSLIDER: Envía y muestra la información de notas musicales mediante un teclado en pantalla.

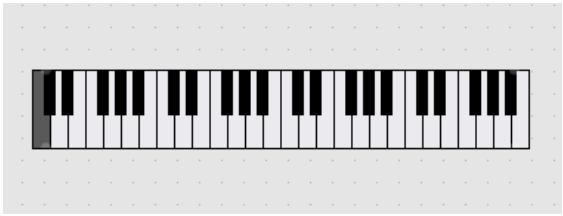


Figura 22. Objeto *kslider.*

+: Ejecuta la operación de suma entre dos objetos determinados y envía el resultado a su salida.

Figura 23. Objeto +.

UMENU: Crea un menú emergente modificable para poder seleccionar entre varias opciones de texto o numéricas.

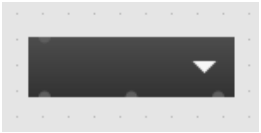


Figura 24. Objeto *umenu.*

AUTOPATTR: Permite que varios objetos seleccionados se incluyan automáticamente en un sistema que almacena patrones.

Figura 25. Objeto *autopattr.*

PATTRSTORAGE: Modifica y almacena datos dentro de un sistema de matrices.

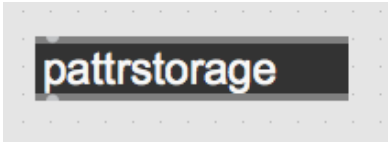


Figura 26. Objeto *pattstorage.*

PRESET: Almacena y recupera los parámetros asignados a cualquier objeto mediante el uso del clic de un mouse.

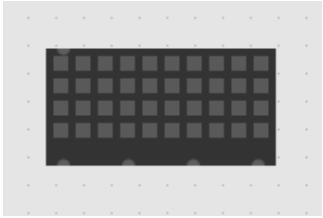


Figura 27. Objeto *preset.*

GAIN~: Es un control deslizante que modifica el volumen de varias señales de audio.

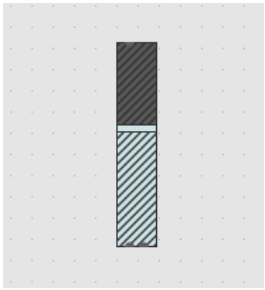


Figura 28. Objeto *gain~.*

ADC~: Convierte señal de audio analógico a digital procedente de un dispositivo externo y las envía independientemente a cada una de sus salidas.

Figura 29. Objeto *adc~*.

EZDAC~: Funciona como una herramienta de conversión que transforma la señal de audio digital procesada en MAX en audio analógico para poder ser escuchado.

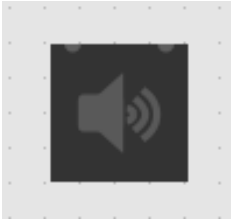


Figura 30. Objeto *ezdac~.*

2.5. Protocolo MIDI

La comunicación que se puede establecer entre varios dispositivos electrónicos relacionados con la creación musical se efectúa mediante el uso del lenguaje MIDI. Para poder establecer estas conexiones se debe tener como mínimo un dispositivo receptor que interpreta los mensajes de control enviados por un equipo transmisor. La mayoría de estos flujos de información manipulan datos como por ejemplo el de las propiedades de una nota musical (altura y velocidad), los mismos que se representan a través de un rango numérico que abarca todos los valores entre el cero y el ciento veintisiete.

Todo el proceso de envío y recepción de mensajes se lleva a cabo mediante la selección de diferentes canales (generalmente nombrados con letras o números) para de esta forma poder trabajar de manera independiente (Bauer, 2014).

3. CAPÍTULO III: METODOLOGÍA

Dentro de este apartado se detalla todos los aspectos que se vieron involucrados en el proceso de desarrollar piezas musicales mediante el uso de herramientas que empleen principios de programación orientada a objetos. Para esto, se diseñaron e implementaron dos programas de composición algorítmica. El primer código fue diseñado en base a la teoría estocástica por lo que se aplicaron únicamente eventos aleatorios para obtener la mayor parte de las variables. El

segundo algoritmo se encuentra fundamentado en la leyes deterministas por lo que siempre va a generar el mismo resultado a partir de datos de entrada constantes. Los resultados que arrojan cada uno de estos sistemas tienen un comportamiento diferente que se va a comparar en la fase de análisis de resultados.

3.1. Programa basado en eventos aleatorios

La construcción de este dispositivo tiene como finalidad obtener una creación compuesta por una estructura musical A-B-A. El usuario tiene la libertad de escoger una nota inicial que genera una escala de la cual se seleccionan y ejecutan al azar melodías y acordes. Después de un tiempo determinado se produce aleatoriamente una nueva nota base con la que se forma la parte B y luego se culmina la obra volviendo a tomar el valor tonal que se fijó en un principio.

Asimismo, se añadió una máquina de ritmos conformada por archivos de audio que simulan algunos instrumentos percutivos electrónicos y que se reproducen siguiendo patrones aleatorios.

El control de todos los procesos mencionados se efectúa mediante la intervención de varios objetos que regulan el comportamiento de parámetros como el volumen, tempo, entre otros.

3.1.1. Nota musical base

El centro de la composición musical es una nota base que puede ser alterada en cualquier momento. La inclusión de este parámetro en el programa se da gracias al uso de la función *number* que permite relacionar las notas de un piano con su respectivo valor numérico. De esta forma, modificando las propiedades de visualización del objeto en la ventana de inspector se logra obtener un cambio de formato decimal (números enteros) a MIDI, logrando que se manipulen únicamente datos representados en una codificación en el sistema de notación

musical anglosajona (C D E F G A B) acompañados de un carácter numérico que indica la octava en la que están situados.

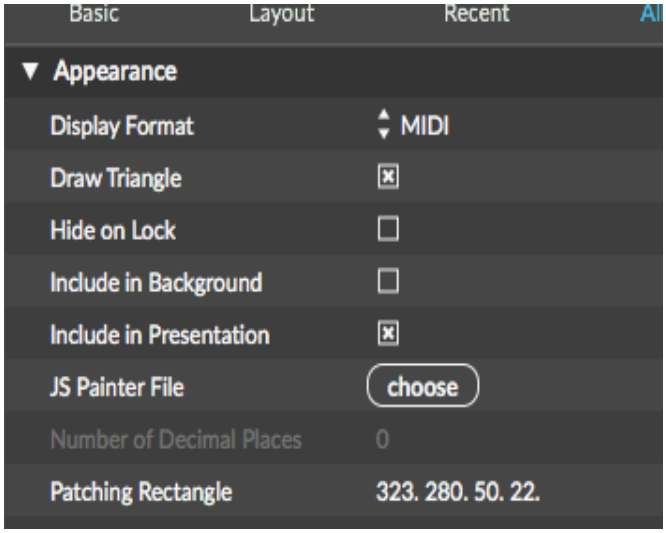


Figura 31. Ventana de modificación del objeto *number* a MIDI.

Un detalle a remarcar, es el hecho de que se usa un mensaje de *loadmess* con argumento de sesenta para que en el caso de que el usuario no seleccione ninguna nota base se cargue automáticamente al abrir el programa un Do de la octava 3 (C3).

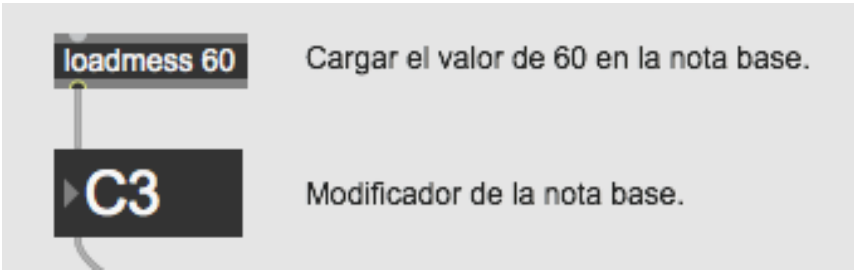


Figura 32. Conexión del *loadmess* 60 con el objeto de la nota base.

3.1.2. Selección de la escala

El uso de una escala en términos musicales limita el rango de notas que pueden utilizarse dentro de una octava. Es por esta razón, que se vio la necesidad de crear un mecanismo que ejecute esta acción en el programa. Para lo cual, a

partir de la nota base seleccionada en la primera etapa, se van a ir añadiendo medios tonos o tonos para ir completando la escala deseada.

Dentro del entorno gráfico de programación de MAX se diseñó un *subpatch* donde en primer lugar se colocó un *umenu* con las opciones “Mayor” o “Menor” refiriéndose a las escalas disponibles. Para que esta función sea activada fue necesario crear un objeto de número y asignar a cada opción un valor siendo mayor = 0 y menor = 1.

Luego se conectaron la salida del objeto numérico con la entrada de un objeto *route* con argumentos 0 y 1 para que si los valores de entrada y argumento son iguales (0=0 o 1=1) se produzca la selección del mensaje que el usuario a seleccionado.

Como se mencionó anteriormente la selección de las notas se da por un proceso matemático por ende en el caso de ser mayor se genera un mensaje con la información “ 2 2 1 2 2 2 1” y si es menor “ 2 1 2 2 1 2 2”, donde el número 1 representa el aumento de medio tono y el 2 representa un tono completo.
Cabe destacar que se conecta a la entrada del mensaje asignado para la escala mayor un objeto *loadbang* que envía un pulso de activación para que por defecto si el usuario no escoge alguna opción el programa funcione sin ningún problema.

Posteriormente, el mensaje seleccionado por el usuario pasa por el objeto *unpack* que se encarga de descomponer los valores en siete partes que van a ser almacenadas en una variable *number* independiente.

Este valor numérico de 2 o 1 va a ser sumado mediante el uso de la operación + 0, con el número MIDI equivalente a la nota base seleccionada por el usuario. El resultado obtenido va a generar la segunda nota de la escala a la cual se le va adiciona el siguiente valor del mensaje para de esta forma y repitiendo este proceso siete veces más se obtendrá todas las notas de la escala.

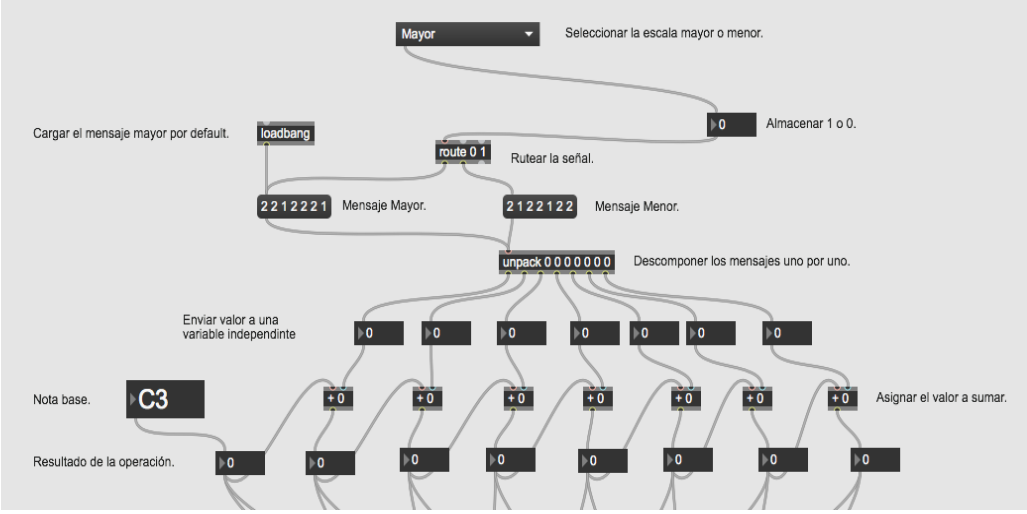


Figura 33. Patch de la generación de las notas de la escala.

3.1.3. Configuración del tempo

La velocidad a la que se ejecuta la composición está controlada gracias a la acción de un objeto llamado *tempo*, cuyos argumentos de control permiten definir a través de un botón *toggle* cuando empieza a correr el metrónomo. Adicionalmente, colocando un espacio y a continuación un determinado número se logra definir los *beats* por minuto a los que se van a generar las notas y con la última opción de entrada se puede configurar las divisiones de una nota es decir se pueden crear desgloses en relación a la siguiente tabla:

Tabla 1.

Valores numéricos asignados a cada subdivisión.

Whole Note (Redonda)	1
Half Note (Blanca)	2
Quarter Note (Negra)	4
Eighth Note (Corchea)	8
Sixteenth Note (Semicorchea)	16

Todos estos factores relacionados con el tiempo generan diferentes posibilidades para cada una de las secciones de la composición debido a que se quiere provocar una sensación de ejecución más real que se asemeja al comportamiento de una creación hecha por seres humanos. La forma en que se consiguió cumplir con esta situación fue construyendo tres *subpatches* que producen diferentes formas de ejecución de la melodía y sección rítmica mediante la aplicación de los objetos *random, route* y mensajes que ocupan únicamente los valores numéricos mencionados en la tabla 1.

Estos *subpatches* se accionan por la intervención de un interruptor *toggle* que envía un pulso a la entrada de un botón normal, el cual tiene la función de transmitir esta pulsación a la entrada de un objeto *random* que producirá únicamente valores entre el cero y el dos. La cantidad obtenida mediante el proceso aleatorio se almacena en una variable numérica que posteriormente va a ser comparada dentro de un objeto *route*. En el caso de que exista una coincidencia entre el argumento y el número aleatorio se procede a aplastar el mensaje que corresponde a esa opción. Existen tres tipos de mensajes que cuentan con los valores de cuatro, ocho y dieciséis respectivamente. La información escogida se asigna a la entrada cuatro del objeto tempo la cual se encargará de marcar las subdivisiones de ejecución.

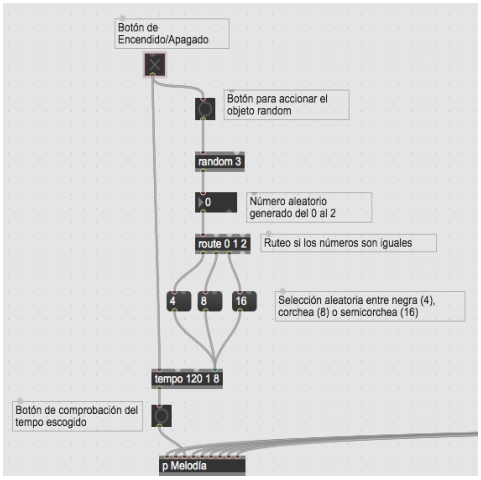


Figura 34. Patch del control del tempo de la melodía.

Es importante resaltar, que en el caso de la sección armónica se diseña un *subpatch* en el que se coloca una cantidad fija de dos en el último argumento del objeto tempo con el objetivo de alargar más la ejecución de las notas y experimentar la sensación de un acorde más no de una melodía.

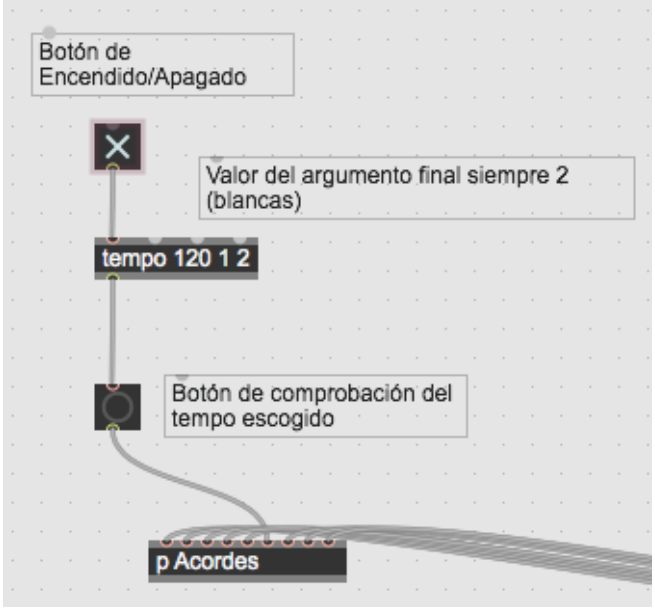


Figura 35. Patch del control del tempo de los acordes.

Cada uno de los *subpatches* creados recibe información acerca de los *beats* por minutos proporcionada por un control numérico que permite al usuario acelerar o disminuir la velocidad de reproducción de la pieza musical.
Adicionalmente, de la misma forma que en el caso de la nota base se coloca un valor de cien en un *loadmess* para empezar las creaciones con este tiempo.

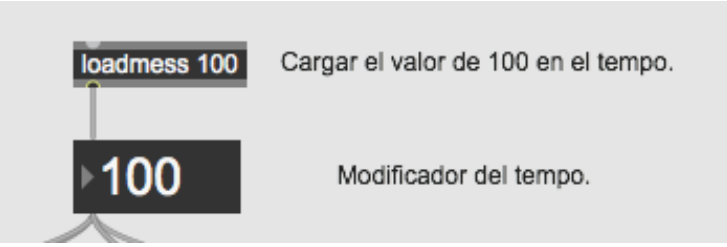


Figura 36. Conexión del *loadmess* 100 con el objeto que controla el tempo.

3.1.4. Cambio de sección

Una composición musical consta de varias partes o secciones que se nombran generalmente con las letras A o B. El algoritmo diseñado tiene como finalidad empezar y culminar con una parte A cuya melodía y armonía se encuentran definidas por la nota base seleccionada por el usuario o generada automáticamente. El cambio de sección para poder entrar a una parte B que cuente con una nueva nota base se la efectúa mediante la elaboración de un *subpatch* llamado “p cambio de sección”.

En la construcción de este *patch* se utilizó un objeto *counter* cuyo principal funcionamiento es el de servir como un contador que inicia en el valor de uno y cuando llega a sesenta se produce un pulso que acciona un botón que desencadena en la aparición de un número aleatorio entre cero y veintitrés. El rango numérico mencionado se define con el objetivo de obtener resultados sonoros que no produzcan molestias al oyente por lo que se ocupan únicamente las notas que forman parte de las octavas entre Do 3 (C3) y Si 4 (B4). Por tanto, se aplica la operación +60 y se suma el número aleatorio obtenido anteriormente con la constante de sesenta. El resultado de este proceso se envía a la entrada de la variable numérica MIDI que maneja la nota base de la escala por lo que la sección “B” cuenta con una estructura melódica y armónica diferente.

La duración de esta parte “B” se controla a través de otro contador que se activa al momento que se cambie la tonalidad de la composición. Al llegar a sesenta se envía un pulso a un *subpatch* que activa uno de los patrones de un objeto *preset* que tiene almacenado el valor de la nota inicial. El retorno de esta variable se ejecuta de manera inmediata y después de cumplir un ciclo fijado con otro contador se apaga el programa.

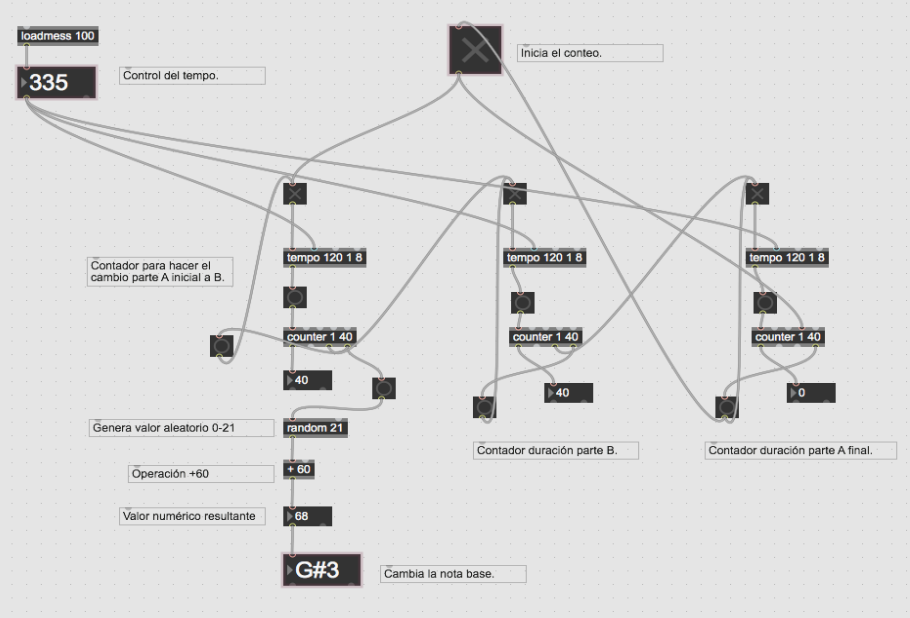


Figura 37. Patch del cambio de secciones A-B-A' del programa.

3.1.5. Generación de la melodía

La creación melódica se efectúa diseñando un *subpatch* llamado "p melodía". El primer objeto que se utiliza es un *random* con argumento de ocho ya que se va a trabajar con esta cantidad para representar las notas musicales obtenidas en el *patch* de la escala. El número generado aleatoriamente se almacena en una variable y posteriormente se aplica una suma con una constante de valor uno. Con este proceso se asegura que la ejecución de notas siempre sea constante.

El resultado de esta operación es almacenado y enviado a un *route* que se encarga de activar el botón que corresponda dependiendo del número que se ha seleccionado. Como hay ocho notas disponibles se crean la misma cantidad de botones y posteriormente se emplea el objeto *int* que almacena un entero y que en su entrada izquierda recibe un pulso para soltar la información que se le asigne en su parte derecha. Las salidas de estas variables se conectan a un objeto llamado *makenote* que es el encargado de transformar esta información numérica en MIDI. Los argumentos de un *makenote* permiten modificar el volumen de ejecución de las notas (*velocity*) y la duración de las mismas. Para poder apreciar de mejor manera el sonido generado por una nota se colocó un

valor de cinco mil para el volumen y quinientos para la duración. Adicionalmente, se usó el objeto *kslider* que es la simulación de un teclado donde se representan las notas que se van reproduciendo. Por último se empleó un objeto *noteout* para lograr escuchar todo lo que se estaba generando ya que al ser datos MIDI se necesita enviar esta información a algún dispositivo de audio.

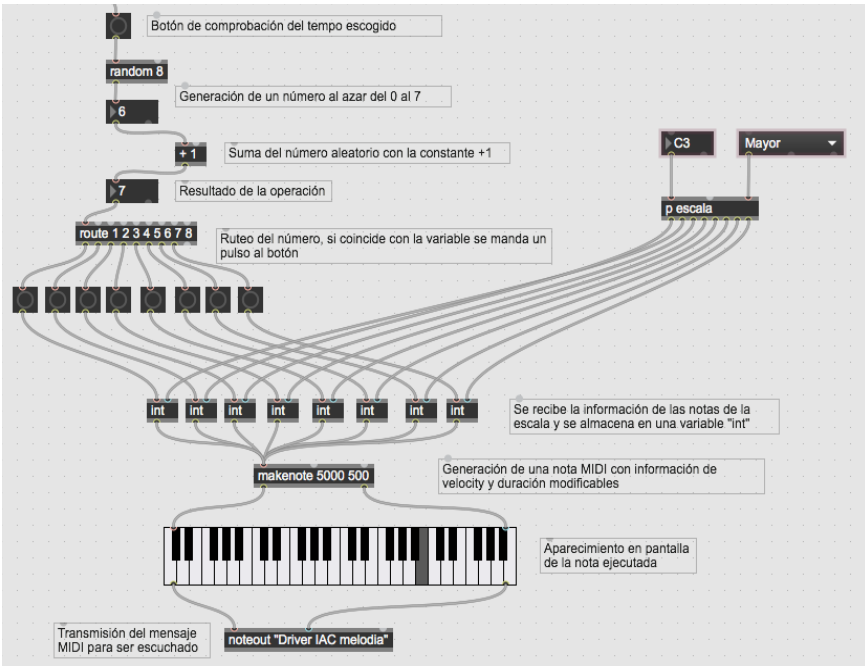


Figura 38. Patch de la generación de la melodía.

3.1.6. Generación de la armonía

En lo referente a la construcción de los acordes hay que mencionar que se usó una estructura similar al *subpatch* de la melodía con las diferencias de que para poder ejecutar varios notas de manera simultánea se debió modificar una propiedad del objeto *kslider*. Por lo que en la pestaña de *Display Mode* de la ventana inspector del objeto en cuestión se cambió el modo de reproducción de monofónico a polifónico.

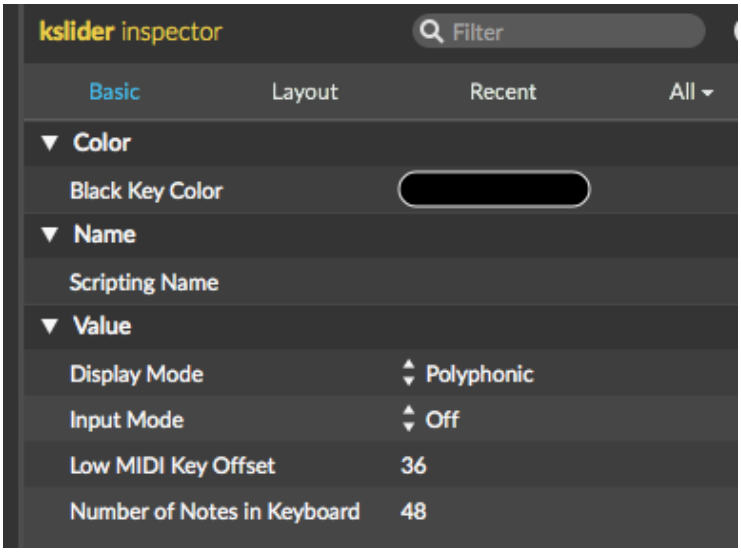


Figura 39. Ventana de modificación del objeto *kslider* a modo polifónico.

Una vez efectuada esta modificación se procedió a repetir el proceso que se ocupó en la sección melódica, tomando en consideración que en este caso los botones creados para ejecutar las notas ahora almacenan tres datos al mismo tiempo ya que van a dar origen a la creación de acordes de triada, caracterizados por presentar todos sus elementos conformantes separados por un intervalo constante.

Mientras se desarrollaban los acordes surgió la necesidad de completar los últimos cuatro con notas que estén una octava más arriba que las empleadas convencionalmente. Para esto, se seleccionaron los valores asignados en la segunda, tercera, cuarta y quinta notas de la escala y se sumó un valor constante de doce para provocar un aumento de octava.

Finalmente los acordes conformados se almacenan en nuevas variables *int* y se asignan a un *makenote* que presentará un volumen de ejecución de doscientos y una duración de mil.

El cierre de este *subpatch* se da insertando un nuevo *noteout* que transportará únicamente la información referente a la armonía.

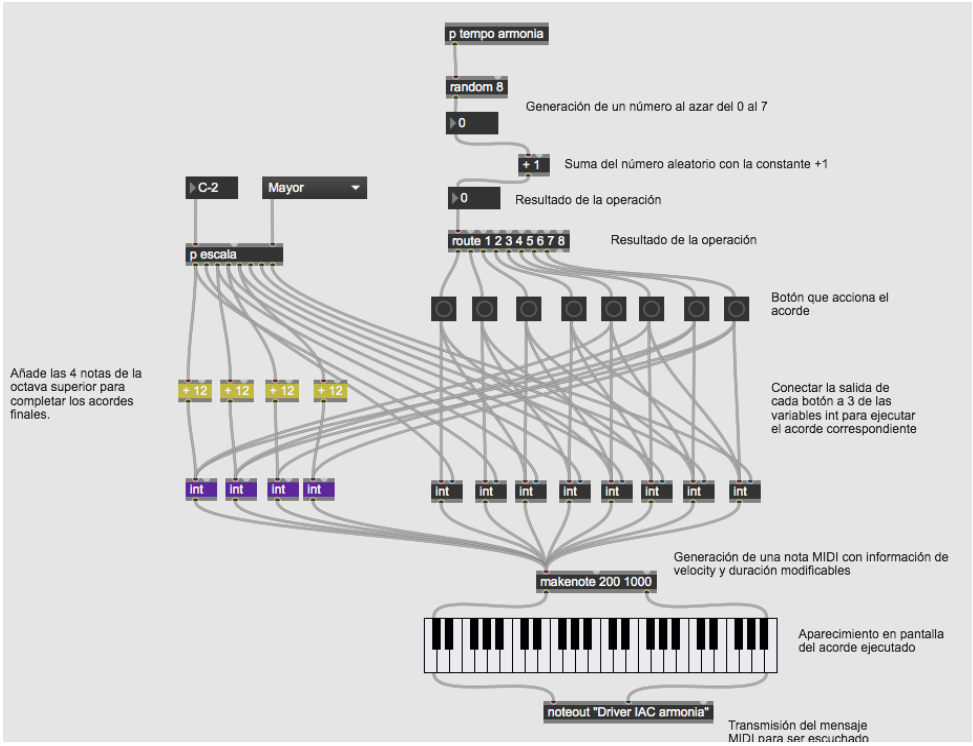


Figura 40. Patch de la generación de la armonía.

3.1.7. Drum Machine

La sección rítmica se encuentra formada por la agrupación de cinco elementos de percusión (Bombo, Caja, *Rimshot*, Campana y Aplauso). La idea principal para lograr la ejecución de cada uno de los elementos que conforman el *drum machine* es la creación de un secuenciador por pasos similar al que presenta el programa Fl Studio, en donde como se puede observar en la figura 41, existen botones modificables que al momento de aplastarlos y encenderlos van a producir el sonido del instrumento al que se encuentren asignado provocando la creación de diferentes ritmos.

La máquina de ritmos está agrupando a todos los instrumentos de percusión, por ende reciben la información proveniente del dispositivo de tempo y la de un contador que se encarga de controlar la parte del patrón que se va a ir reproduciendo.

Figura 41. Secuenciador por pasos del programa *FI Studio*.

Tomado de (Licenselounge, s.f).

3.1.7.1. Subpatch del instrumento de percusión

Los cinco instrumentos se agruparon en *subpatches* independientes, los mismos reciben la información enviada desde el contador y mediante el uso del objeto *sel* se efectuó la generación de pulsos que van pasando por los botones creados. Cada botón tiene una asignación numérica comenzando desde el cero y terminando en el siete. Los pulsos generados son enviados por la salida de los botones a la entrada derecha de un *gate* que almacena esta información y mediante un análisis que realiza este objeto a los datos que llegan por su entrada izquierda se va a liberar el pulso almacenado con destino a otro botón que está conectado a un mensaje con un valor numérico de uno que hace que se reproduzca el audio del instrumento. Posteriormente, se conectó el audio a un control de ganancia individual y a uno general.

En relación con los patrones de ejecución de cada instrumento se llevó a cabo un proceso de almacenamiento y lectura de estos parámetros mediante el uso de los objetos *pattstorage* y *autopattr*. *Pattstorage* permite guardar datos de una variable determinada en una matriz como la que se muestra en la figura 42. Para evitar confusiones y facilitar el trabajo se cambiaron los nombres de los botones *toggle* creados y se colocó abreviaciones como “k1” en el caso del bombo.

Autopattr asigna automáticamente las variables nombradas a la matriz y guarda los estados que manejan.

	Name	Priority	Interp	Data
☒	k1	0	linear	1
☒	k2	0	linear	0
☒	k3	0	linear	0
☒	k4	0	linear	1
☒	k5	0	linear	0
☒	k6	0	linear	1
☒	k7	0	linear	0
☒	k8	0	linear	1

Figura 42. Matriz de almacenamiento del objeto *pattstorage.*

Una vez asignadas las variables a las matrices se utiliza un *metro* que libera un pulso cada dos segundos para generar una cantidad aleatorio entre cero y tres. Cada uno de estos números tiene asignado un patrón de ejecución del instrumento que se modifica aplastando los botones *toggle* y dejando espacios en blanco para generar silencios. Los patrones de cada instrumento se guardan aplastando la tecla *Shift* y presionando un clic en cualquiera de los bancos de memoria que ofrece el objeto *preset*. *Preset* envía estos datos directamente a la matriz del *pattstorage* y van a ir cambiando según el número aleatorio que salga en cada ejecución.

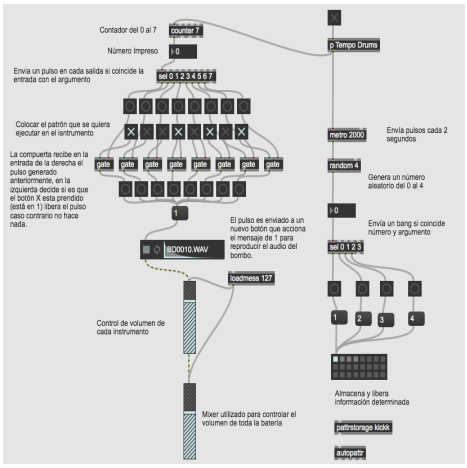


Figura 43. Patch de la generación del patrón de ejecución del bombo.

El esquema utilizado es exactamente el mismo para los cinco instrumentos con la variación del audio utilizado y el nombre de los ocho botones *toggle*.

3.1.8. Conexión Ableton Live 10

Tanto la selección de instrumentos como el ajuste de volumen de la sección melódica y armónica se llevó a cabo efectuando un envío de datos MIDI desde MAX hacia el programa Ableton 10. Para esto se utilizó un dispositivo MIDI *llamado Inter-Application Communication* o abreviado en siglas IAC.

Los objetos de noteout construidos en los puntos anteriores se utilizan junto con el mensaje “Driver IAC armonía” o “Driver IAC melodía” según se requiera. En Ableton se crea una nueva sesión con seis instrumentos diferentes a los que se asigna en la pestaña de *MIDI From* el dispositivo de entrada IAC correspondiente y el canal MIDI del cual van a recibir datos.

En cada ejecución del programa se van a seleccionar de manera al azar los instrumentos que van a ser ejecutados. Es por esto que en MAX se construye una estructura con combinaciones de objetos *random* y *route* para que cada vez que se accione el botón de arranque del programa se escoja un mensaje MIDI diferente.

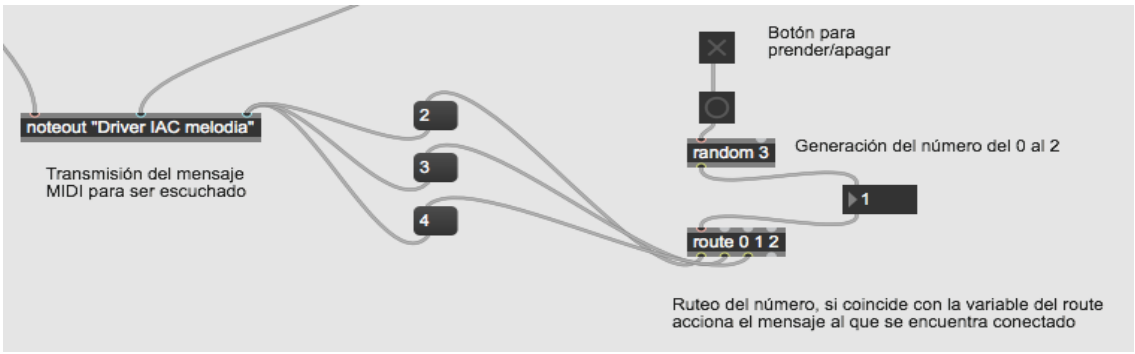


Figura 44. Patch de la selección de instrumento.

3.1.9. Retorno de audio procedente de Ableton 10

Con el fin de poder controlar el volumen de los instrumentos de melodía y armonía dentro del programa MAX, se creó un *subptach* llamado “p retorno”, donde mediante el objeto *adc~* y el uso de la interfaz de audio virtual *soundflower* se logró asignar a cada instrumento una salida independiente desde los canales de Ableton 10 hacia la entrada del conversor análogo-digital.

En ambos programas, dentro de las preferencias de entrada y salida de audio se seleccionó al dispositivo *soundflower* como la herramienta capaz de enviar y recibir la información necesaria.

Figura 45. Selección de la interfaz de audio Soundflower para ambos programas.

Finalmente, en Max se construyó el *subpatch* mencionado anteriormente, el cual se muestra en la figura 46.
Las salidas uno y dos del *adc~* se conectan a un controlador de ganancia independiente para luego ser transportadas a un control general.

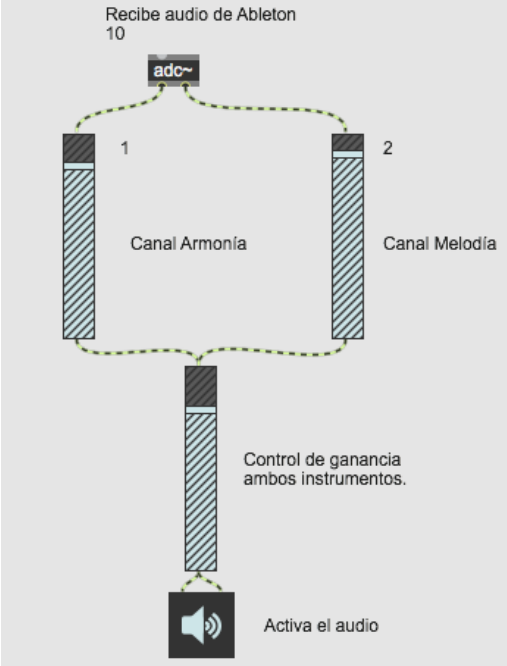


Figura 46. Patch del regreso del audio de Ableton a Max.

3.1.10. Sección de efectos

En la mayoría de las estaciones de trabajo de audio digital existe la opción de colocar una serie de dispositivos cuya función principal es la de modificar varias propiedades sonoras como la dinámica, contenido espectral o espacialidad. Dentro del programa creado, se incorporó la posibilidad de colocar tres efectos a la suma total del audio de la melodía, armonía y *drum machine*. Este proceso se lo llevó a cabo usando el objeto de audio *gate~*, el mismo que junto a la acción de tres interruptores *toggle* van a decidir si el audio va a ser procesado o no por algún efecto.

Uno de los aspectos importantes a mencionar en este apartado, es la utilización de la plataforma BEAP incluida en las últimas versiones del programa MAX. Dentro de esta librería se escogieron un filtro pasa altos, un *delay* simple y un compresor estéreo para manipular la señal resultante. El beneficio que se tuvo al ocupar estas herramientas fue el de ahorrar tiempo ya que se contaba con estructuras ya diseñadas que solo requieren de una señal de entrada para funcionar.

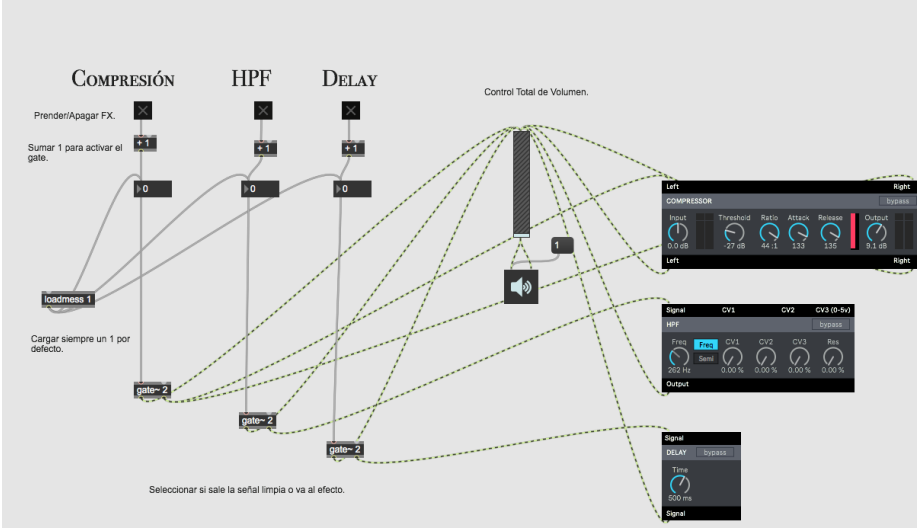


Figura 47. Patch del procesador de efectos disponible.

3.1.11. Interfaz gráfica

Después de culminar todo el proceso ligado con la programación se procedió al diseño de una interfaz gráfica amigable para el usuario mediante la herramienta de *Presentation Mode*. En esta ventana se colocaron tres paneles divididos de manera que en el primero se agrupen tanto el interruptor de encendido y apagado del código como también todos los parámetros básicos relacionados con la selección del tempo, escala y nota base de la composición.

En la segunda sección se colocan todos los controladores referentes al volumen de todos los instrumentos y grupos para manejar la cantidad de señal que se va a escuchar. Por último en la parte inferior se sitúan a los tres efectos junto con su botón de acción.

Finalmente, en la parte superior se asignó el nombre de “RANDOM COMPOSER MACHINE” a todo el *patch* final.

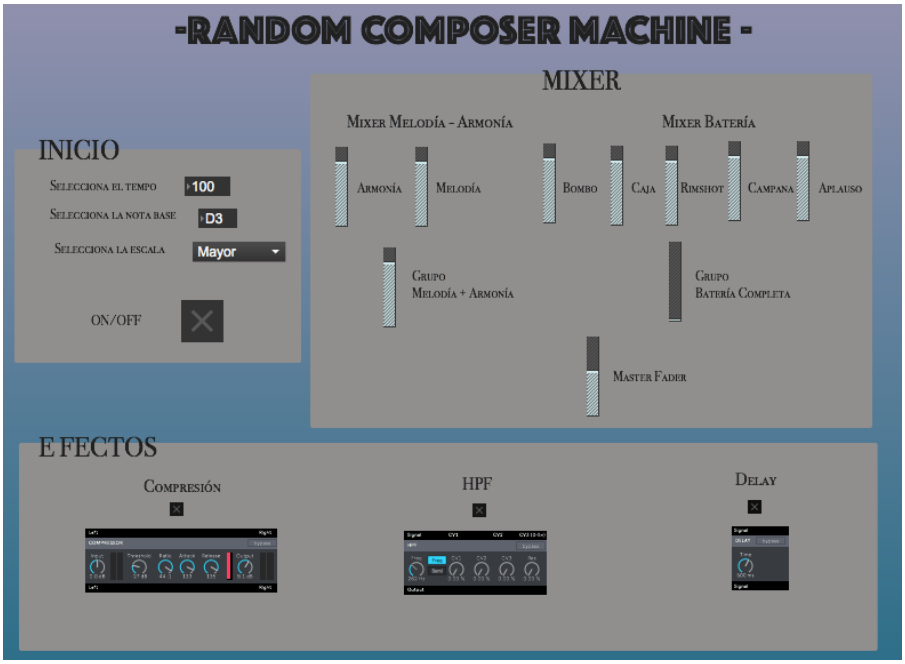


Figura 48. Interfaz gráfica del programa "*Random Composer Machine".*

3.2. Programa basado en la linealidad

Un programa lineal arroja el mismo resultado en todas sus ejecuciones. Por lo tanto, para conseguir este objetivo se transcribió mediante el uso de objetos de programación visual todos los datos relacionados con la melodía y armonía de la obra "Someday My Prince Will Come" escrita por Larry Morey. Además, se conformó una sección de batería con un bombo, una caja y un hi hat, los cuales se reproducen siguiendo un patrón del libro "260 Drum Machine Patterns".

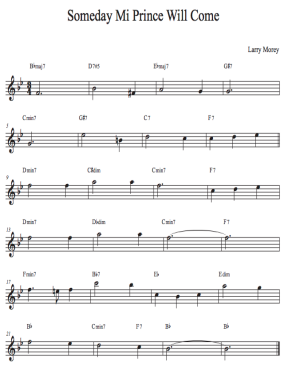


Figura 49. Partitura base para construir la composición lineal.

3.2.1. Sección melódica y armónica

La creación de esta sección viene dada por el uso de objetos como el tempo que va a determinar a qué velocidad se mueve un contador delimitado con valores de uno hasta el cuarenta y cuatro. La razón de la elección de estos números es porque según la partitura se tiene esta cantidad de notas musicales.
La salida numérica del contador es procesada por un selector que envía pulsos a una serie de botones que van a ir reproduciendo la información de la nota musical en MIDI contenida en un mensaje.

Para la armonía se pusieron las notas que forman parte del acorde separadas por una coma.

Todo los datos resultantes de los procedimientos mencionados se trasladan a un *makenote* y a un *kslider* para finalmente ser enviados por el mecanismo del "IAC" a dos canales de Ableton 10 cargados con instrumentos acústicos (Piano para la melodía y un ensamble de cuerdas para la armonía).

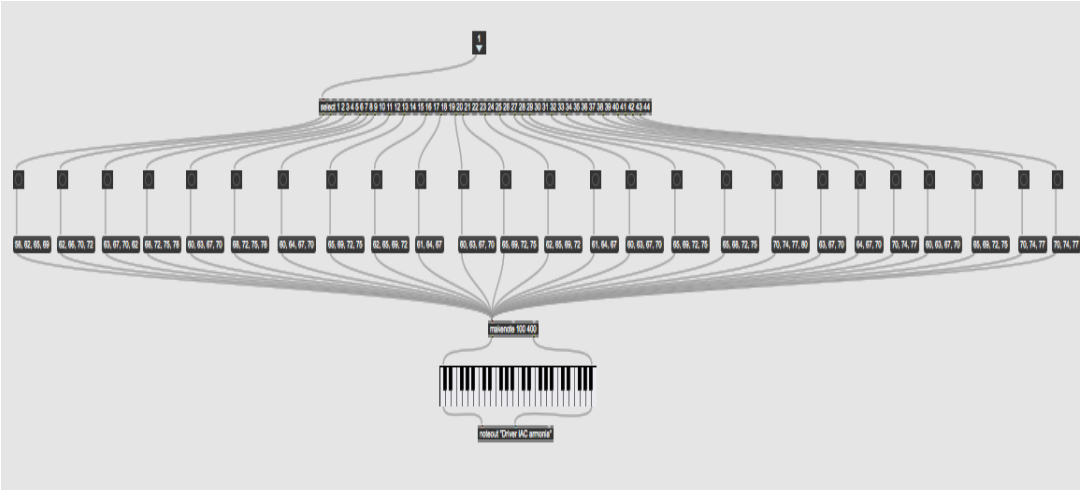


Figura 50. Patch de la transcripción armónica.

3.2.2. Sección rítmica

Se estableció un set de batería simple cuya ejecución está ligada a los objetos *tempo*, *counter* y *select* mencionados en el apartado anterior. La configuración de cada uno de estos sistemas es bastante similar a la empleada en el *drum machine* de la composición aleatoria con la diferencia de que en este caso se almacena en el objeto *preset* un patrón constante que se encuentra definido por la información detallada en la figura 51, donde se marca con color negro los lugares que van a ser reproducidos por cada instrumento.

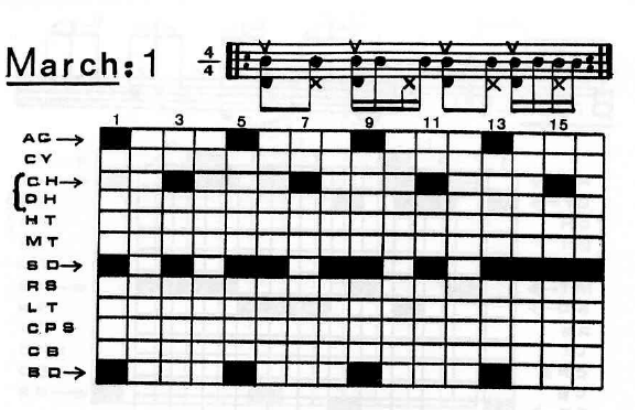


Figura 51. Patrón del género musical Marcha.
Tomado de (Bardet, 1987).

3.2.3. Interfaz gráfica

Los elementos más representativos y con mayor funcionalidad para el programa se incluyeron en la ventana de presentación de MAX.

Al tratarse de un código sencillo se coloca únicamente un interruptor de acción del programa, seguido por el controlador de tiempo y por último se añade un pequeño *mixer* para regular el volumen de cada elemento instrumental.

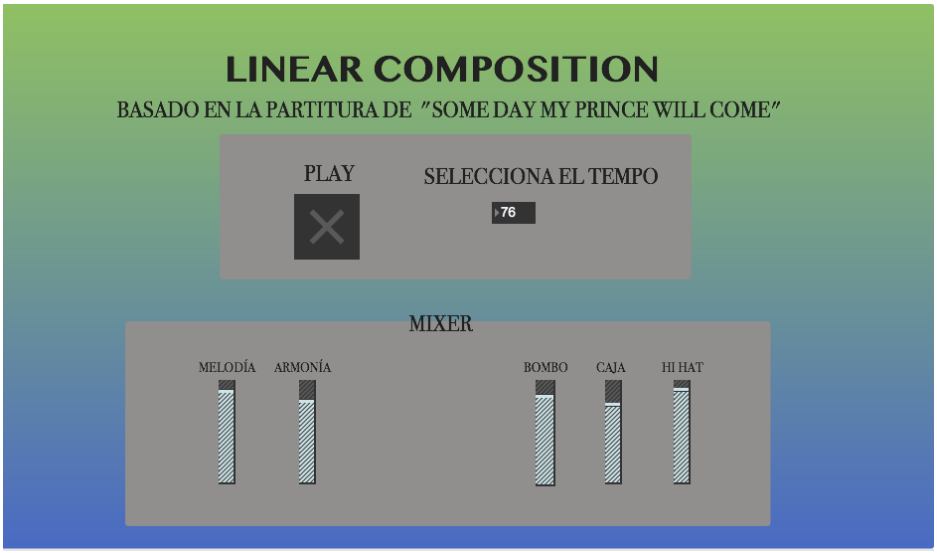


Figura 52. Interfaz gráfica del programa "*Linear Composition*".

4. CAPÍTULO IV: ANÁLISIS DE RESULTADOS

Se expone a continuación el análisis descriptivo de los datos obtenidos durante la creación de dos programas de composición algorítmica.

Se debe tomar en cuenta el óptimo funcionamiento de los elementos que integran estas plataformas. Esto implica, el envío de los datos generados por las salidas MIDI provenientes de MAX hacia las entradas del programa Logic Pro X. En este último dispositivo se utilizaron las herramientas de grabación y edición de notas para identificar el comportamiento musical que se desarrolla.

4.1. Composición Aleatoria

Cabe destacar que los resultados obtenidos en cada ejecución del programa es diferente por lo que se plantean tres situaciones en las que se elige una escala y una nota base para observar los cambios de estructura que presentan tanto la melodía como armonía originadas.

4.1.1. Análisis Melódico

En cuanto a la parte melódica, se toma en cuenta un aspecto fundamental que consiste en comprobar que las notas generadas se encuentren dentro de la escala escogida y generada aleatoriamente. Además gracias a la naturaleza de las composiciones se puede efectuar un análisis modal de cada sección producida por el programa.

Para lo cual, se presentan a continuación tres casos que serán expuestos a detalle.

4.1.1.1. Escala mayor con nota base C3

En este caso, se eligió como tonalidad base Do, que se encarga de conformar la escala mayor relacionada con la parte A y A'. En lo referente a la sección B el algoritmo selecciona la nota Fa sostenido con el fin de originar una sensación auditiva distinta.

En lo referente a la modalidad se puede determinar que en la parte A existe un modo La eólico mientras que A' presenta un Fa sostenido dórico.

Cabe recalcar, que las figuras musicales producidas para esta ocasión varían entre semicorcheas y corcheas únicamente a excepción de una redonda que marca el final de la composición.

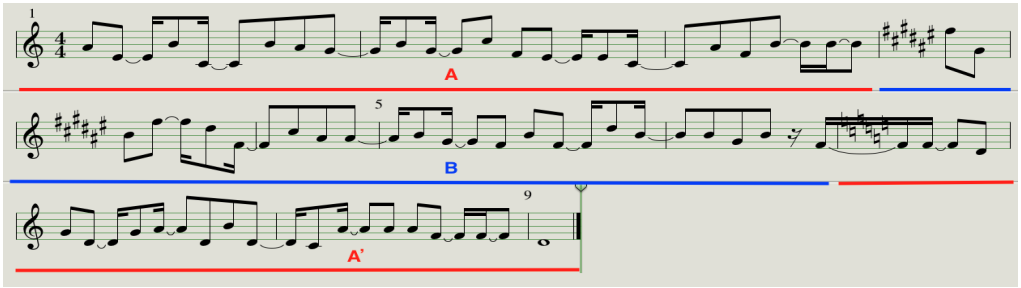


Figura 53. Partitura del resultado melódico obtenido con una escala mayor y nota base C3.

4.1.1.2. Escala menor con nota base F3

La partitura presentada en la figura 54, muestra la melodía creada en base a un patrón de tonalidades menores Fa, Sol, Fa. Al igual que en el primer caso se incorporan corcheas y semicorcheas con la excepción de la añadidura de tresillos debido a las subdivisiones aportadas por la lógica de programación empleada.

Al igual que en el caso anterior se establece que en la parte A existe un modo de Do mixolidio y en A' hay un Re bemol lidio.

A pesar de que para este caso se usa el mismo algoritmo, la cantidad de notas aumenta de forma notable. Adicionalmente, se evidencia la presencia de algunos silencios de semicorchea en algunas secciones que complementan la pieza musical.

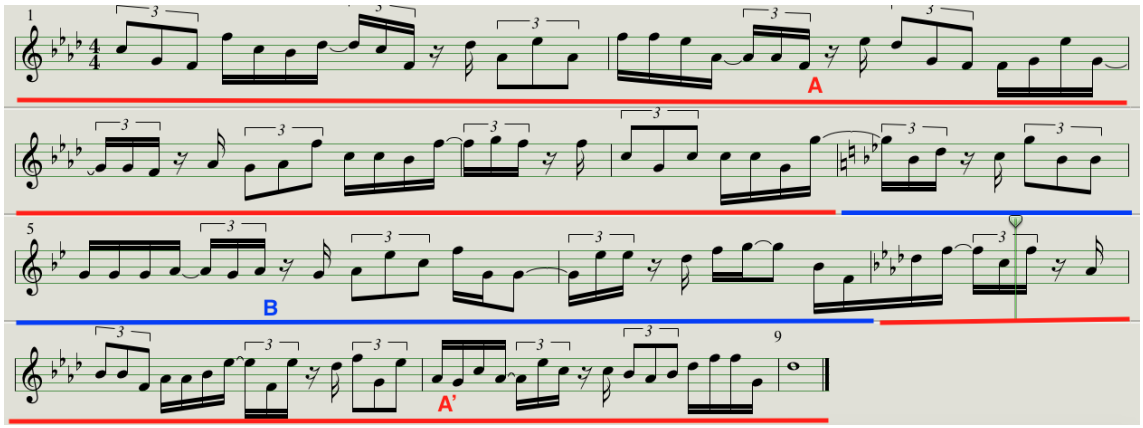


Figura 54. Partitura del resultado melódico obtenido con una escala menor y nota base F3.

4.1.1.3. Escala mayor con nota base A4

En este último ejemplo, se emplea una melodía con un número reducido de notas producidas dentro de un campo tonal mayor definido por el patrón La, Do, La pertenecientes a la octava cuatro.

La modalidad de todas las secciones de esta composición presentan un modo jónico en las notas de La y Do mayor respectivamente.

La presencia de figuras musicales como blancas, negras y silencios de corchea y semicorchea causan un efecto rítmico más espaciado en comparación con los casos anteriores.

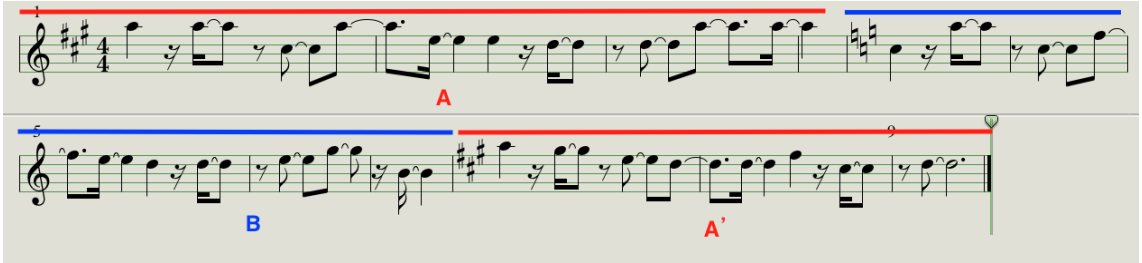


Figura 55. Partitura del resultado melódico obtenido con una escala mayor y nota base A4.

4.1.2. Análisis Armónico

El método para interpretar el comportamiento de los acordes implementados a través del programa se desarrolla mediante la utilización del mismo procedimiento anterior, destacando que las notas y figuras musicales mantienen un ritmo bastante similar con variaciones de las figuras musicales en pocas ocasiones.

4.1.2.1. Escala mayor con nota base D3

Los fragmentos armónicos obtenidos en este caso se producen en base a las notas Re, Do, Re. En esta situación se manifiesta una alteración tonal bastante perceptible al oído que se encuentra provocada por un cambio de octava. Además, la aparición de silencios de blanca, negra, corchea y semicorchea después de cada acorde se genera por los límites establecidos en el algoritmo.

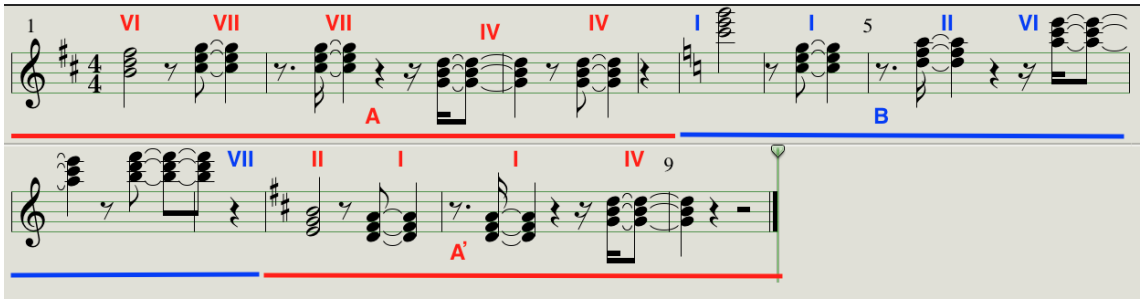


Figura 56. Partitura del resultado armónico obtenido con una escala mayor y nota base D3.

4.1.2.2. Escala menor con nota base B3

Las figuras musicales que aparecen en esta partitura son bastante similares a las evidenciadas en el apartado anterior a excepción de una combinación de tres redondas que terminan la composición musical. Los acordes que se encuentran conformando las secciones A y B emplean una escala menor a partir de las notas Si y Do respectivamente.

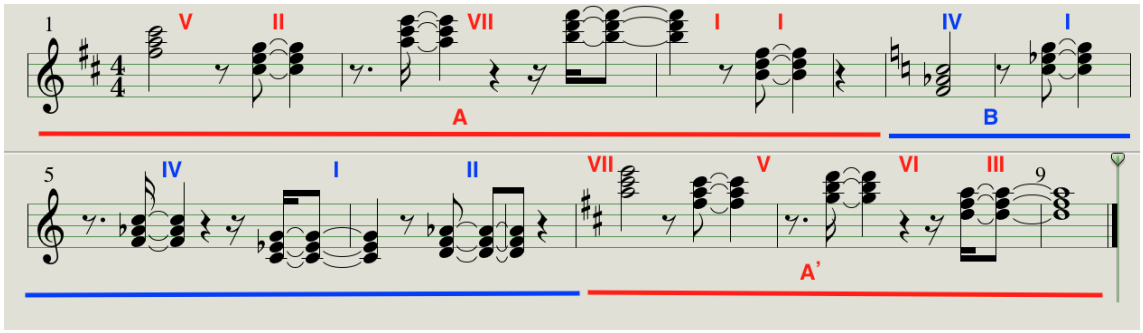


Figura 57. Partitura del resultado armónico obtenido con una escala menor y nota base B3.

4.2. Composición Lineal

Al tratarse de una transcripción directa de una composición musical ya establecida se espera que los datos generados adquieran una notación melódica y armónica bastante cercana.

Es por esto que se efectuaron las transcripciones de estas dos propiedades con el fin de observar los resultados obtenidos.

4.2.1. Análisis Melódico y Armónico

La melodía y armonía que se producen a partir de la composición algorítmica presenta las mismas notas y acordes que se evidencian en la partitura base pero no existe una equivalencia exacta en relación con las figuras y compases generados. Esto desemboca en problemas con la percepción rítmica debido a que existe la aparición de corcheas con punto, semicorcheas, tresillos y varios silencios que no se asemejan a las blancas con punto, negras y corcheas presentes en la pieza base.

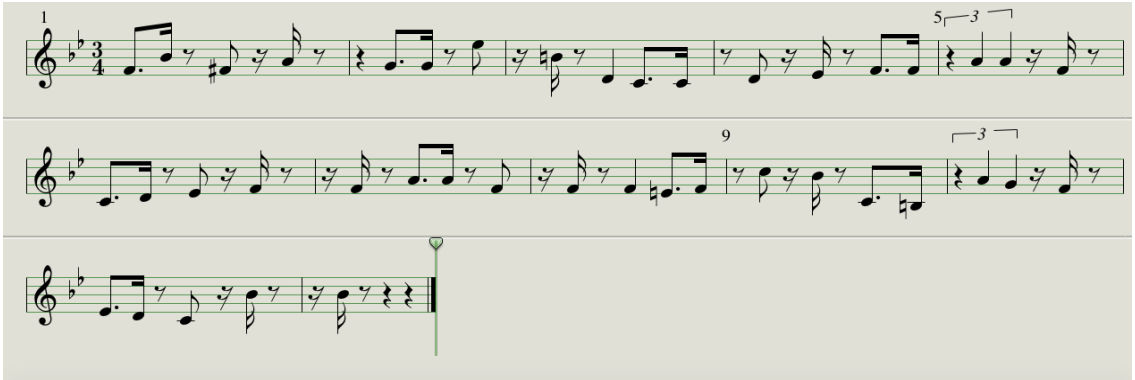


Figura 58. Partitura del resultado melódico obtenido de la composición lineal.

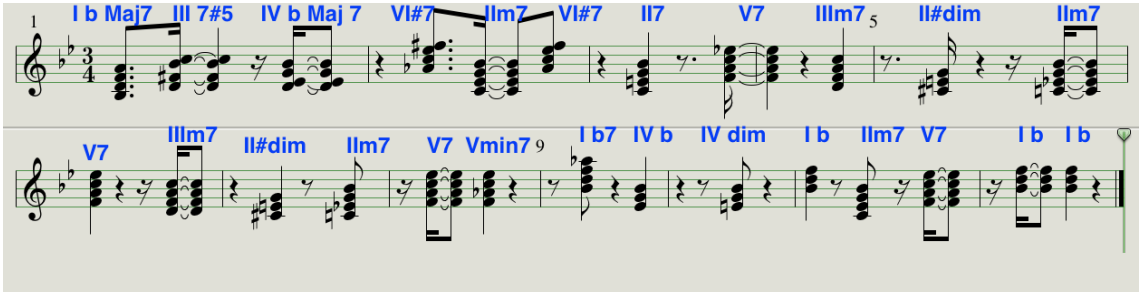


Figura 59. Partitura del resultado armónico obtenido de la composición lineal.

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Después de culminar el trabajo de investigación se puede inferir que los lenguajes de programación por sus propiedades pueden convertirse en herramientas capaces de producir expresiones artísticas similares a las creadas por la intervención de los seres humanos. Cabe mencionar que al tratarse de procesos elaborados a través del uso de un computador se pueden generar algunos inconvenientes que afectan a las cualidades musicales directamente ya que pese a la similitud que existe en las composiciones creadas tanto por inteligencia artificial como por la intervención humana, siempre influirá el sentimiento con el que un individuo realice la interpretación de una pieza musical.

Es así que, los modelos de composición algorítmica empleados entregan una pieza musical que cuenta con una estructura definida pero en el caso de los eventos aleatorios es importante destacar que resulta difícil catalogar a los resultados obtenidos en un determinado género musical como rock, pop o jazz debido a las características que posee el método estocástico aplicado. Sin embargo, se podría plantear que este tipo de códigos generarían un nuevo género musical ya que posee una biblioteca completa con la mayor parte de los acordes y melodías disponibles.

En cuanto a los resultados obtenidos podemos evidenciar que en lo referente a las obras musicales creadas en base a procesos aleatorios resulta difícil valorar la intensidad con la que la obra causará una sensación auditiva agradable al oyente ya que existen constantes variaciones entre los patrones armónicos, melódicos y rítmicos, mientras que las creaciones humanas tienden a ser acogidas con mayor facilidad.

Cabe destacar la importancia que han tenido las interconexiones que se efectúan entre los lenguajes de programación y las diferentes herramientas tecnológicas como interfaces de audio digital, protocolos MIDI y programas especializados en

el tratamiento de audio, a la hora de llevar a cabo la creación e interpretación de piezas musicales creadas por un computador.

Finalmente, se puede mencionar que para este caso se utilizaron los modelos de composición por algoritmos que poseen una compatibilidad directa con el lenguaje de programación aplicado a objetos para generar un proceso eficiente y ágil.

5.2. Recomendaciones

Para la ejecución de los dos programas de composición algorítmica es necesario contar con la versión 8.0.4 de la plataforma MAX. En el caso de no tener la licencia de este *software* se puede descargar una prueba gratis durante treinta días. Adicionalmente, el usuario debe tener instalada la interfaz digital *soundflower* y Ableton Live 10 para poder escuchar los instrumentos seleccionados para las partes de la melodía y armonía.

La mayoría de los procesos ejecutados se encuentran almacenados en *subpatches* por lo que si se desea explorar los contenidos de estos elementos se debe dar doble *click* sobre el objeto en cuestión.

De igual manera, debido al uso del objeto *pattstorage* para poder almacenar los patrones creados en la batería se crean varios archivos de extensión *.json* que son guardados en la misma carpeta donde se guardan cada uno de los programas principales.

Por último, respecto a la ejecución del código basado en aleatoriedad es importante señalar que no hay una limitación en la selección de la nota base pero se recomienda por cuestiones de sonoridad trabajar dentro de las octavas tres y cuatro.

REFERENCIAS

Bauer, W. I. (2014). *Music learning today: digital pedagogy for creating, performing, and responding to music*. *Choice Reviews Online*, *51*(12), 51-6651-51–6651. https://doi.org/10.5860/choice.51-6651

Blackwell, A. F., & Collins, N. (2011). *The Programming Language as a Musical Instrument 1 Introduction*. *Psychology of Programming Languages Interest Group*. https://doi.org/10.1002/cbic.201000754

Castro Lobo, M. R. (2003). Música para todos : Una Introducción al estudio de la música. Editorial de la Universidad de Costa Rica. Recuperado el 25 de abril del 2019 de https://books.google.com.ec/books/about/M%C3%BAsica_Para_Todos_Una_Introducci%C3%B3n_Al.html?hl=es&id=hA5YLhK3XEwC&redir_esc=y

Cipriani, A., & Giri, M. (2010). *Electronic Music and Sound Design; Vol. 1*. Recuperado el 12 de Abril del 2019 de https://www.virtual-sound.com/demo/emasd_demo.pdf

Collado, L., (2014). Estudio e implementación de métodos de composición algorítmica con propósitos exploratívos. Recuperado el 31 de Mayo del 2019 de https://grfia.dlsi.ua.es/cm/pfc/lsirvent/memoria.pdf

Collins, N., & McLean, A. (2014). *Algorave: A Survey of the History, Aesthetics and Technology of Live Performance of Algorithmic Electronic Dance Music*. *Proceedings of the International Conference on New Interfaces for Musical Expression*. Recuperado el 22 de Marzo del 2019 de https://pdfs.semanticscholar.org/b06c/81b4686635c384422abff1196c2ca65eb3b8.pdf

Eigenfeldt, A., & Pasquier, P. (2011). *Negotiated Content: Generative Soundscape Composition by Autonomous Musical Agents in Coming Together: Freesound Audio Metaphor (AuMe) View project Naos: Biometric Architecture View project Negotiated Content: Generative Soundscape*

Composition by Autonomo. Recuperado el 20 de marzo del 2019 de https://www.researchgate.net/publication/228411309

Gabis, C., Senno, J., & Ozṅ, R. (2006). Armonía funcional. Recuperado el 6 de abril del 2019 de https://books.google.com.ec/books/about/Armonía_funcional.html?id=gcaPx0xQMm4C&redir_esc=y

Goulart, A. J. H., & Antar, M. D. (2015). *Live Coding the computer as part of a free improvisation orchestra of acoustic instruments*. https://doi.org/10.5281/ZENODO.19317

Magnusson, T. (2011). *Algorithms as Scores: Coding Live Music. Leonardo Music Journal*, *21*(21), 19–23. https://doi.org/10.1162/LMJ_a_00056

Magnusson, T. (2013). *The Threnoscope: A Musical Work for Live Coding Performance. Live 2013 (International Conference on Software Engineering)*. Recuperado el 13 de abril del 2019 de https://pdfs.semanticscholar.org/ded5/44c65a9e89d8bcd98c0a168936d58831de59.pdf?_ga=2.230787423.1703893689.1565197868-1718386408.1559967246

Manzo, V. J. (2012). *Max/MSP/Jitter for music: a practical guide to developing interactive music systems for education and more*. Recuperado el 7 de mayo del 2019 de https://books.google.at/books?id=f5NHDAAAQBAJ&source=gbs_book_other_versions%0Ahttp://www.gbv.de/dms/bowker/toc/9780199777686.pdf

Matić, D. (2010). *A genetic algorithm for composing music*. *Yugoslav Journal of Operations Research*, *20*, 157–177. https://doi.org/10.2298/YJOR1001157M

Nuanáin, C. Ó., & Sullivan, L. O. (2014). *Real-time algorithmic composition with a tabletop musical interface*. *Proceedings of the 9th Audio Mostly on A*

Conference on Interaction With Sound - AM '14, 1–7. https://doi.org/10.1145/2636879.2636890

Parkinson, A., & Bell, R. (2015). *Deadmau5, Derek Bailey, and the Laptop Instrument–Improvisation, Composition, and Liveness in Live Coding. In Proceedings of the First International Conference on Live Coding.* Recuperado el 25 de marzo del 2019 de https://pdfs.semanticscholar.org/8868/b65000702581d1b011d6226b36c896e2bbcb.pdf

Peñalver Vilar, J. M. (2009). Lenguaje musical II. Recuperado el 28 de junio del 2019 de http://repositori.uji.es/xmlui/handle/10234/23862

Roberts, C., & Kuchera-Morin, J. (2012). *Gibber: Live Coding Audio in the Browser. Proceedings of the International Computer Music Conference*. https://doi.org/10.1111/j.1467-8519.2012.01968.x

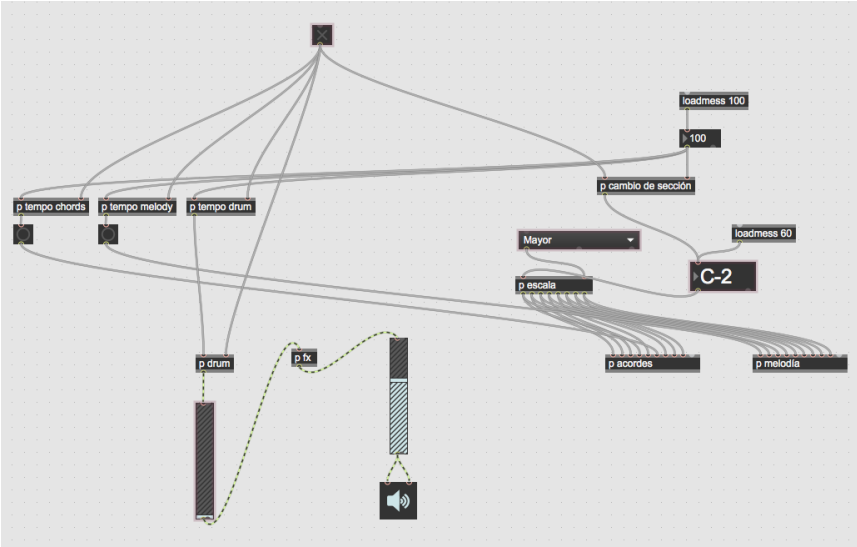
Roca, D., & Molina , E. (2006). Vademecum musical. Recuperado el 20 de mayo del 2019 de http://cerezo.pntic.mec.es/jarran8/webpersonal/Docs/Vademecum_musical.pdf

Simoni, M., & Dannenberg, R. B. (2013). *Algorithmic composition: A guide to composing music with Nyquist. In Algorithmic Composition: A Guide to Composing Music with Nyquist.* Recuperado el 31 de Mayo del 2019 de http://www.scopus.com/inward/record.url?eid=2-s2.0 84917515146&partnerID=40&md5=912f234eb299a51f0d23e91cdb62ada5

ANEXOS

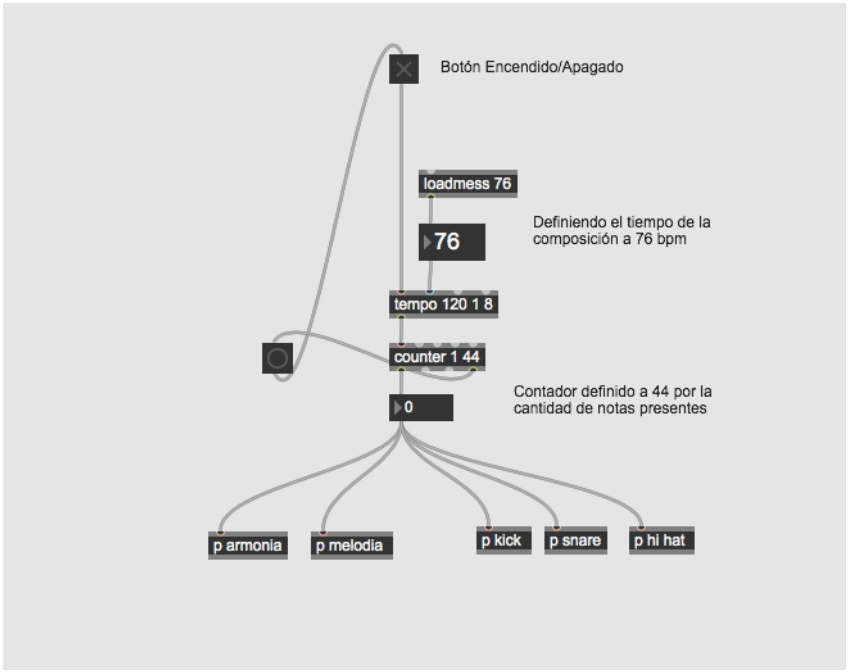
ANEXO I

Vista de *patch* del programa de composición aleatorio.



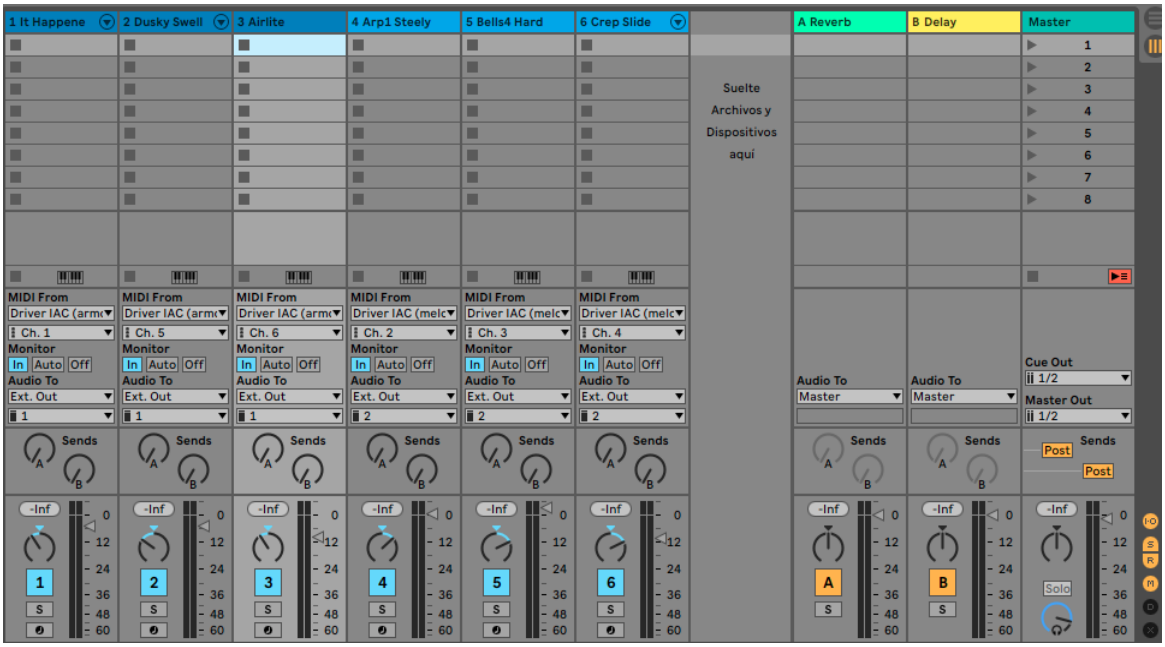
ANEXO II

Vista de *patch* del programa de composición lineal.



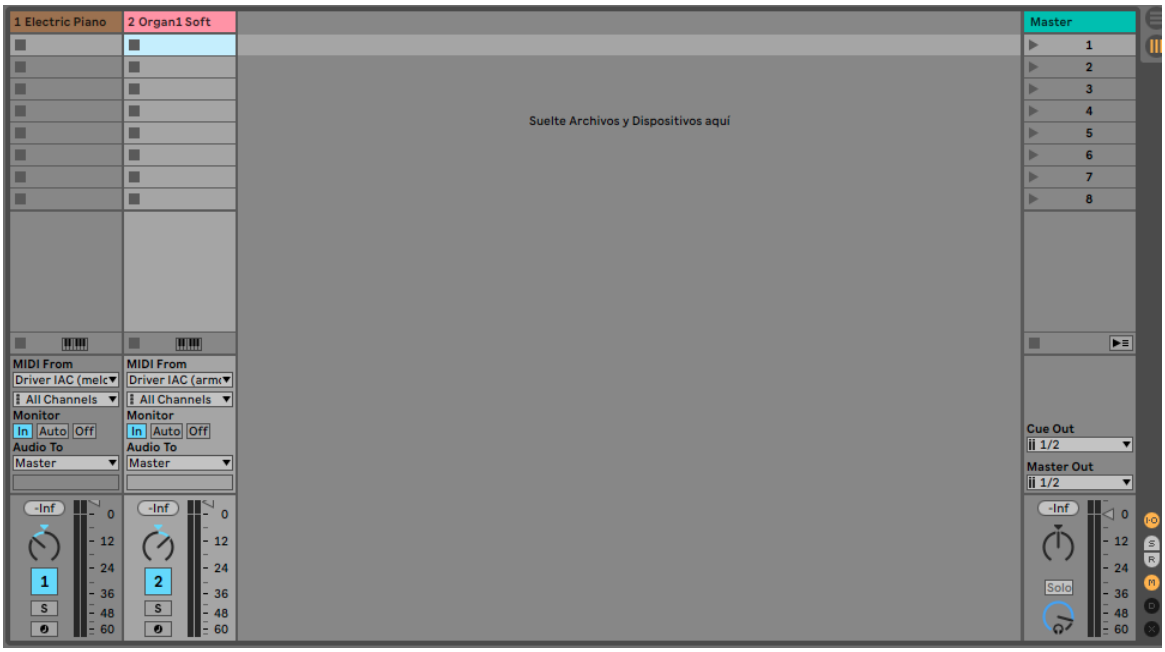
ANEXO III

Vista del programa Ableton 10 para la composición aleatoria.



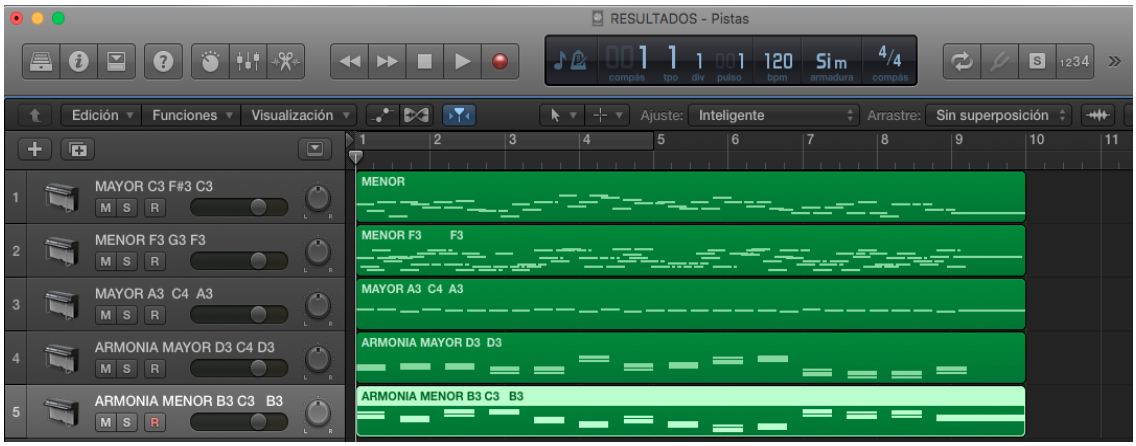
ANEXO IV

Vista del programa Ableton 10 para la composición lineal.



ANEXO V

Grabaciones MIDI en el programa Logic Pro X de los resultados obtenidos de las composiciones algorítmicas aleatorias.



ANEXO VI

Grabaciones MIDI en el programa Logic Pro X de los resultados obtenidos de la composición algorítmica lineal.