FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UNA APLICACIÓN DE SÍNTESIS SONORA PARA SISTEMA OPERATIVO ANDROID

AUTOR

CRISTIAN DANILO RIVADENEIRA TERÁN

AÑO

2018

FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UNA APLICACIÓN DE SÍNTESIS SONORA PARA SISTEMA OPERATIVO ANDROID

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de ingeniero en sonido y acústica.

Profesor Guía

Msc. Adrián Paúl Cabezas Yánez

Autor

Cristian Danilo Rivadeneira Terán

Año

2018

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, desarrollo de una aplicación de síntesis sonora para sistema operativo Android, a través de reuniones periódicas con el estudiante Cristian Danilo Rivadeneira Terán, en el semestre 2018-1, 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".

_____

Adrián Paúl Cabezas Yánez

Master of creative industries (Music and sound)

CI: 171918954-8

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, desarrollo de una aplicación de síntesis sonora para sistema operativo Android, de Cristian Danilo Rivadeneira Terán, en el semestre 2018-1, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

José Antonio Álvarez-Torres Yépez

Magister 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.”

_____

Cristian Danilo Rivadeneira Terán

CI: 100404327-7

AGRADECIMIENTO

Quiero agradecer a quienes fueron mi principal apoyo para lograr este objetivo, mis padres, sin su ayuda no habría podido alcanzar el nivel de conocimiento que tengo ahora.

DEDICATORIA

Dedicado a mi familia, para demostrarles que las metas por más difíciles e inalcanzables que parezcan con esfuerzo y dedicación se pueden lograr.

RESUMEN

El producto final de esta investigación es una aplicación interactiva para dispositivos con sistema operativo *Android,* la cual plantea un sintetizador basado en síntesis aditiva/sustractiva desarrollada en el lenguaje de programación *Pure Data*, de esta manera se ofrecen las bases fundamentales para proveer herramientas de fácil acceso a profesionales del campo. El objetivo planteado se pudo lograr a partir de profundizar los conocimientos de las diferentes herramientas de programación como lo son: *Android Studio*, *Pure Data* y la librería *LIbpd*. En este proyecto se aplicó teoría revisada en la materia de Seminario de arte sonoro utilizando el lenguaje de programación *Java*, kit de desarrollo diseñado exclusivamente para la creación de aplicaciones para dispositivos móviles denominado *JDK7*, con las bases desarrollada se procedió a realizar la comunicación entre ellas usando la librería *pd-for-android*; Finalmente se probó la aplicación tanto en emuladores como en dispositivos reales para comprobar su funcionamiento.

ABSTRACT

The final product of this research is an interactive application for devices with Android operating system, this aplication is used for digital audio synthesis, with this the fundamental ideas some tolos will be offered to provide easy access tools to audio professionals. The objective could be achieved from deepening the knowledge of the different programming tools such as: Android Studio, Pure Data and the LIbpd library. Once the design of the synthesizer with its different sections was done, the design of a graphic interface for the Android operating system was carried out, taking into account the compatibility of devices provided by the same company. For this, the Java programming language was used, a development kit designed exclusively for the creation of applications for mobile devices called JDK 7. With the two tools developed, communication between them was made using pd-for-android. Finally, the application will run in emulators and real devices to check their operation.

ÍNDICE

1. INTRODUCCIÓN

El diseño de sonido en la actualidad se puede desarrollar con un sin número de herramientas, en este documento se muestran los diferentes pasos para desarrollar una de ellas, entre los principales se encuentran, el diseño de un sintetizador junto con su flujo de señal, el diseño de una interfaz gráfica y finalmente la comunicación entre diferentes tipos de lenguajes de programación para así obtener una aplicación compatible en dispositivos con sistema operativo *Android,* como ejemplo existe un *software* programado en Matlab el cual realiza síntesis de sonido mediante el método de la transformada de Fourier (Cortes, *Knott* y Osorio, 2012).

El desarrollo de aplicaciones móviles en los últimos años va en aumento ya que estos aportan a los usuarios el acceso inmediato e interactivo a distintas herramientas para uso cotidiano y profesional (Blanco et al, 2009). Es por esto que se propone desarrollar una aplicación para dispositivos móviles con sistema operativo *Android* que permita controlar un parche de síntesis sonora diseñado en Pure Data con el fin de proveer información útil para facilitar la comprensión de cómo poder embeber un programa dentro de otro y así poder hacerlos trabajar juntos, para en un futuro poder desarrollar aplicativos mucho más complejos y que se puedan emplear mucho más en el uso profesional del diseño de sonido.

2. ANTECEDENTES

Para poder desarrollar aplicaciones que tengan un gran alcance es necesario conocer la tendencia de uso de aplicaciones por parte de los usuarios, de esta manera obtendremos información importante que nos ayudará a organizar un plan detallado para el desarrollo del producto. Por ejemplo, saber qué sistema operativo es más utilizado y junto a que dispositivos trabaja, para esto se utilizó información detallada de un estudio realizado en 2013 (Gasca, Ariza y Delgado, p. 22). Los sistemas operativos con mayor uso en Latinoamérica son iOS y *Android* (Delía et al, 2013),

Puesto que el sistema operativo *Android* es usado por una amplia gama de marcas de dispositivos (Vanegas, 2012. p. 130), es mucho más fiable que la aplicación llegue a muchos más potenciales usuarios.

Existen algunas guías para el desarrollo de aplicaciones sobre sistema operativo *Android* como por ejemplo la elaborada por Carlos Alberto Vanegas (2012, p. 129). Con estas guías se puede tener mayor conocimiento de todas las herramientas necesarias para la elaboración de una aplicación.

Por otro lado, la inclusión de parches de *Pure Data* en distintos tipos de software para que sean los encargados de la generación y administración de sonido está en pleno desarrollo (Brinkmann et al, 2011), es por esto que es un campo de explotación muy amplio en el cual se pueden llegar a crear herramientas de uso profesional y que cumplan con los requisitos más primordiales por los usuarios.

Algunos ejemplos de aplicaciones para *Android* que utilizan *Pure Data* son: *Wave Trip* (Luckyframe, 2013), *PPP-LooperSynth* (Aubert, Recoules, 2017).

Actualmente se pueden encontrar emuladores de sintetizadores analógicos como por ejemplo el iMS-20 de *Korg,* para el cual existe una aplicación compatible con *iOS*.

3. OBJETIVOS

3.1 Objetivo general

Desarrollar una aplicación de síntesis sonora compatible con sistema operativo *Android*, a través del uso de los lenguajes de programación *Pure Data* y *Android Studio.*

3.2 Objetivos específicos

- Diseñar un diagrama de flujo de señal para un sintetizador aditivo/sustractivo a través del lenguaje de programación *Pure Data*.
- Diseñar una interfaz gráfica interactiva para el uso del sintetizador como aplicación, mediante el uso del lenguaje de programación *Android Studio*.

- Evaluar la funcionalidad de la aplicación a través de la metodología de desarrollo impulsado por pruebas.

4. ALCANCE

Desarrollo de una aplicación de síntesis sonora para el sistema operativo *Android*, la cual tendrá como fin poder realizar síntesis aditiva/sustractiva mediante una interfaz gráfica diseñada en *Android Studio*.

En la Figura 1, se muestra un diagrama de bloques con las distintas etapas del sintetizador a implementar:

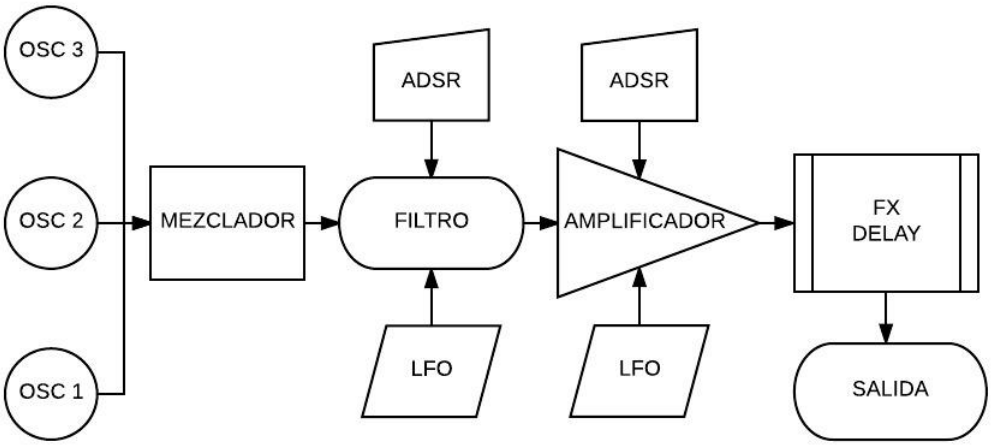


Figura 1. Diagrama sintetizador aditivo/sustractivo

5. MARCO TEÓRICO

5.1. Conceptos técnicos

5.1.1. *Software*

El *software* en informática es un conjunto de instrucciones lógicas que permiten a un computador realizar distintas tareas, es un componente intangible que controla el *hardware*; el s*oftware* puede ser creado a través de distintos lenguajes de programación.

5.1.2. Lenguaje de programación

El lenguaje de programación es un medio de comunicación entre el desarrollador y el computador, el cual permite crear instrucciones con un orden lógico para que el computador ejecute las tareas requeridas.

En detalle el lenguaje de programación transforma las instrucciones creadas por el desarrollador en lenguaje de máquina, el cual utiliza un conjunto de datos binarios manejados por el procesador de un computador.

5.1.3. Sistema operativo

El sistema operativo funciona como una interfaz que permite la comunicación entre el desarrollador, los recursos de *hardware* y *software* de un dispositivo.

Las principales funciones del sistema operativo son: proporcionar una interfaz gráfica para uso del dispositivo, gestión de periféricos de entrada/salida, gestión de memoria *RAM*, administración del procesador y gestión de la información.

5.1.4. Aplicación

Una aplicación o más conocida como (app) es un *software* diseñado exclusivamente para dispositivos móviles, estas pueden ser descargadas e instaladas en un dispositivo desde una tienda de distribución digital, existen algunos tipos de aplicaciones las cuales son: nativas, *web* e híbridas.

Las aplicaciones nativas son desarrolladas para un solo sistema operativo en concreto utilizando su propio kit de desarrollo *(SDK),* o pueden utilizar lenguaje *HTML* y requieren de internet para ser usadas, sin embargo, hay aplicaciones híbridas que pueden combinar los dos tipos de aplicaciones mencionadas anteriormente siendo así como ejemplo, una aplicación que sirva tanto para *Android* como para *iOS*.

5.2. Pure Data

Pure Data es un lenguaje de programación gráfico de código abierto, desarrollado por *Miller Puckette*, el cual está orientado específicamente para la creación de *software* multimedia.

Para la creación de los algoritmos *Pure Data* utiliza pequeñas cajas llamadas objetos las cuales realizan distintas operaciones según su tipo, para la transmisión de datos entre los distintos objetos se utilizan conectores llamados cables de parche (Puredata, s.f).

Existen dos versiones de *Pure Data:*

- *Pd Vanilla*
- *Pd-L2ork*

5.2.1. Pd Vanilla

Pd Vanilla es una versión básica de *Pure Data* que contiene un administrado de librerías llamado *Deken* el cual permite a *Pure Data* acceder a librerías externas desarrolladas por terceros permitiendo que los proyectos creados puedan ser usados por otros lenguajes de programación (puredata, s.f.).

5.2.2. Pd-L2ork

Pd-L2ork es una versión basada en la ya obsoleta *Pd-extended*, la cual incluye una librería nativa muy extensa y que no permite el acceso a librerías externas (puredata, s.f).

5.2.3. Parche

Se conoce como parche o *patch* a un software desarrollado en *Pure Data* y que lleva como extensión de archivo “.pd”.

5.2.4. Tipos de objetos

Pure Data posee una serie de objetos principales que son:

- Objeto
- Mensaje
- Número
- Símbolo
- Comentario

5.2.4.1. Objeto

El objeto depende de la función que se le desee asignar, existen varias funciones las cuales se escriben dentro del objeto y de estar dentro de la librería y escritas con la sintaxis correcta *Pure Data* las identificará.

5.2.4.2. Mensaje

Este objeto permite almacenar uno o varios mensajes, si este objeto recibe un mensaje proveniente de otro objeto este pasa inmediatamente dicho mensaje por su salida correspondiente, al pulsar sobre este objeto también se realiza el envío del mensaje que contenga.

5.2.4.3. Número

Número almacena datos de tipo numérico, estos datos se pueden ingresar por medio de teclado, ratón u otro objeto. Cuando recibe un dato inmediatamente lo transmite por su salida correspondiente.

5.2.4.4. Símbolo

Este almacena datos de tipo símbolo siempre y cuando reciba una señal de *bang* de otro objeto.

5.2.4.5. Comentario

El objeto comentario permite agregar caracteres de texto y sirven para insertar descripciones o etiquetas para distintas secciones de un parche.

5.2.5. Objetos utilizados

5.2.5.1. *Bang*

Bang permite enviar un pulso es decir un uno lógico.

5.2.5.2. *Toggle*

Este objeto permite enviar un dato binario, según su estado si está marcado envía un uno lógico y si no está marcado envía un cero lógico.

5.2.5.3. *Vslider/hslider*

Estos son deslizadores que permiten enviar datos numéricos comprendidos entre un máximo y un mínimo establecido por el desarrollador, mediante un control deslizante, el incremento numérico puede ser logarítmico o lineal.

5.2.5.4. *Canvas*

Canvas es un objeto que puede contener otros objetos de tal manera que puede mostrarlos según se configure, se utiliza para poder crear una interfaz gráfica dentro del parche de *Pure Data.*

5.2.5.5. *Subpatch*

El *subpatch* permite agrupar otros objetos con el fin de ahorrar espacio dentro de la ventana de desarrollo y mantener un proyecto ordenado. Puede otorgar una serie de entradas y salidas de así desearlo.

5.2.5.6. *mtof*

Tiene como función tomar como entrada un número *MIDI* y lo transforma a valor de frecuencia.

5.2.5.7. *dac~*

Este objeto permite enviar la señal recibida por sus entradas derecha e izquierda hacia al conversor analógico digital del dispositivo que esté ejecutando el parche.

5.2.5.8. *noise~*

Es un objeto que crea señal de ruido blanco.

5.2.5.9. *osc~*

Osc~ crea una señal sinusoidal recibe como parámetro principal la frecuencia de oscilación.

5.2.5.10. *phasor~*

Oscilador que genera una señal de tipo diente de sierra recibe como parámetro principal la frecuencia de oscilación.

5.2.5.11. *line~*

Este objeto permite crear un fundido de entrad o de salida por lo general para poder evitar ruidos de intermodulación, este objeto recibe como dato un par de números los cuales le indican que nivel alcanzar y en cuanto tiempo, el tiempo debe ser configurado en milisegundos.

5.2.5.12. *vline~*

Tiene como función crear una secuencia de fundidos, como entrada recibe un conjunto de pares de números los cuales configurarán la secuencia, este objeto es esencial para la creación de una envolvente.

5.2.5.13. *inlet~*

Crear una entrada de tipo señal en un *subpatch.*

5.2.5.14. *outlet~*

Crea una salida de tipo señal en un *subpatch.*

5.2.5.15. snapshot~

Es un objeto que toma muestras, es decir que toma una entrada de señal y nos permite convertirla en un dato de control.

5.2.5.16. +~

Este objeto permite sumar, en su entrada izquierda recibe la señal y en su entrada derecha recibe el dato de control que se sumará.

5.2.5.17. *~

Tiene como función multiplicar, en su entrada izquierda recibe la señal y en su entrada derecha recibe el dato de control que se multiplicará.

5.2.5.18. /~

Objeto de operación matemática que permite dividir, en su entrada izquierda recibe la señal y en su entrada derecha recibe el dato de control para el que se dividirá.

5.2.5.19. *delwrite~*

Esta aparta espacio de memoria para almacenar datos de una señal de audio con el fin de crear una línea de retardo, este objeto toma como entrada dos argumentos, el primero es el nombre de la línea de retardo, y el segundo representa el retardo que se requiere en milisegundos.

5.2.5.20. *s~*

Permite recibir señal, el parámetro principal es la etiqueta del control que escuchará para recibir la señal.

5.2.5.21. *r~*

Sirve para recibir señal, el parámetro principal es la etiqueta del control al que se enviará la señal.

5.2.5.22. *metro~*

Este control permite enviar una serie de *bangs* en un intervalo constante de tiempo, recibe como entrada un dato de tipo control binario el cual le dice cuando iniciar o cuando acabar él envió de la serie, y el otro dato de entrada permite configurar el tiempo en milisegundos en la que deben ocurrir cada envió de *bang.*

5.2.5.23. *lop~*

Tiene como función la de implementar un filtro pasa bajos, recibe en su entrada izquierda la señal, y en su entrada derecha recibe la frecuencia de corte del filtro.

5.2.5.24. *vd~*

Este objeto permite leer una línea de retarlo y devolverla, el tiempo en el que devuelve la señal está definida por la señal de entrada.

5.2.5.25. *sig~*

Es un control que transforma datos de tipo control en señal.

5.2.5.26. *pack*

Pack sirve para empaquetar un conjunto de datos, el número de datos que puede almacenar es configurada por el desarrollador.

5.2.5.27. *loadbang*

Este control permite enviar un *bang* cuando se abre o se carga un parche de *Pure Data.*

5.2.5.28. *moses*

Moses es un objeto que toma datos numéricos en su entrada y puede distribuirlos de la siguiente manera, los distribuye por su salida izquierda se son menores que el argumento de condición y los distribuye por su salida derecha si son mayores que el argumento de condición.

5.2.5.29. *set*

Este objeto utiliza un condicionante para poder realizar sus tareas, primero se debe configurar uno o más valores los cuales definirán el argumento de condición y dependiendo de cuál se cumpla hace pasar los datos por uno u otra salida.

5.3. *Libpd*

Libpd es una librería de código abierto que permite embeber un parche de *Pure Data* en muchos lenguajes de programación y por consiguiente en varios sistemas operativos, lo cual quiere decir que permite usar el *software* creado en *Pure Data* desde cualquier dispositivo móvil e incluso computadores, potenciando así la creación de *software* multimedia.

5.3.1. pd-for-android

La librería *pd-for-android* está diseñada exclusivamente para poder embeber y controlar un parche de *Pure Data* desde sistema operativo *Android*.

5.4. Android studio

Android studio es un lenguaje de programación denominado: entorno de desarrollo integrado (IDE), ya que está basado en *Java,* es considerado el lenguaje de programación oficial para desarrollo de aplicaciones para sistema operativo *Android* (Developer.android, s.f.).

Una de las ventajas de este lenguaje de programación es que ofrece un emulador de dispositivos que utilizan sistema operativo *Android,* para así poder compilar las aplicaciones desarrolladas y así poder depurarlas de una manera sencilla.

5.4.1. *Java*

Java es un lenguaje de programación que permite el desarrollo multiplataforma de *software* y contenido *web* seguro, es acompañado de varios kits de desarrollo que complementan la creación de aplicaciones y servicios informáticos (Oracle, s.f.).

5.4.2. *Java development kit (JDK)*

Java development kit (JDK) es un conjunto de herramientas necesarias para poder desarrollar aplicaciones, las cuales permiten probar, depurar y monitorizar aplicaciones basadas en lenguaje *Java.*

5.4.3. *Software development kit (SDK)*

Software development kit (JDK) también conocida como *API,* es un conjunto de herramientas que facilitan el desarrollo de aplicaciones ya que contienen funciones realizadas por terceros, de esta manera podemos usar recursos ya elaborados para que el desarrollo sea eficiente. El *(JDK)* tiene una versión específica para un tipo de *hardware* o *software* sobre el cual se esté desarrollando una aplicación (4rsoluciones, s.f.).

5.4.4. *XML*

XML (Extensible Markup Language) es un meta-lenguaje que se basa en un lenguaje llamado *Standard Generalized Markup Lenguage,* el cual permite organizar y etiquetar un conjunto de documentos.

La utilidad de XML es muy amplia pero su uso básico el de estructurar y representar datos, de igual manera permite tener una intercomunicación entre varios lenguajes de programación y aplicaciones *web*.

5.4.5. Actividad

Una actividad en *Android Studio* es un área donde un usuario puede interactuar con la aplicación, cada actividad tiene asignada una ventana en la cual se procede a crear una interfaz gráfica, dicha ventana puede ocupar toda la pantalla de un dispositivo o una pequeña sección, en caso de tener varias actividades están podrían sobreponerse.

Al crear una actividad se crean dos archivos que le corresponden, uno con extensión "*.java"* y el otro con extensión "*.xml",* estos son los responsables de poder realizar las distintas tareas designadas por el desarrollador dentro de una aplicación.

5.4.6. Diseño

Un diseño o *layout* dentro del lenguaje *Android Studio* es la estructura sobre la cual se creará la interfaz gráfica de una aplicación, sobre este diseño se insertan las vistas o grupo de vistas que ofrece *Android Studio.* Este diseño es creado usando XML.

Un diseño puede contener más diseños para así poder definir secciones dentro de una aplicación.

5.4.7. Vista

Una vista o *View* son objetos dentro del lenguaje de programación *Android Studio,* los cuales ayudan a representar una interfaz gráfica de una aplicación, estas vistas son creadas utilizando código *XML* y contenidas dentro de un *Layout.*

Se utiliza *Java* para poder controlar todas las acciones posibles de las vistas según su tipo utilizando los métodos.

5.4.7.1. Métodos

Los métodos son el conjunto de acciones que puede realizar una vista, estos se encuentran definidos en lenguaje *Java.*

Dentro de los métodos se puede crear el código responsable de que una vista ejecute una tarea definida por el desarrollador.

5.4.7.2. Vistas utilizadas

5.4.7.2.1. *Button*

Button es una vista la cual se puede tocar o presionar para que se ejecute una tarea.

5.4.7.2.2. *TextView*

TextView es una vista que permite desplegar caracteres de texto dentro de una interfaz gráfica de una aplicación.

5.4.7.2.3. *SeekBar*

SeekBar es una vista similar a una barra de progreso que posee un control el cual se puede deslizar al pulsarlo dentro del rango definido por el tamaño de la vista.

5.4.7.2.4. *Fragment*

Un *fragment* es un módulo dentro de una actividad que posee su propio evento de entrada, su principal uso es el de poder reutilizar secciones es decir llamarlas o quitarlas; esto hace que una aplicación sea mucho más dinámica. Se puede decir que un fragmento es como una sub actividad.

5.4.7.2.5. *ScrollView*

Un *scrollView* es un contenedor el cual permite desplazar un conjunto de vistas, con el fin de poder mostrar vistas que estén fuera del rango de visión que cubre una actividad.

5.4.8. Interfaz

Una interfaz dentro de *Android studio* es un archivo con extensión *.java* el cual permite crear funciones para poder pasar datos entre actividades y así poder tener un enlace entre todos los componentes dentro de una aplicación.

5.5. Síntesis de sonido

5.5.1. Síntesis analógica

Síntesis analógica es un proceso que permite crear sonidos a partir de señales continuas las cuales son generadas por osciladores que representan funciones matemáticas periódicas. Este proceso lo realiza un sintetizador que está conformado por dispositivos electrónicos (Gómez, 2009, pp. 2-3).

5.5.2. Síntesis digital

Síntesis digital es un proceso que permite crear sonidos a partir de señales discretas, estas pueden ser generadas por algoritmos dentro de un software dando así paso a la creación de varias técnicas de síntesis incluida la emulación de síntesis analógica, es por esto que la síntesis digital tiene un

cierto grado de ventaja frente a la síntesis analógica haciéndola más versátil (Gómez, 2009, pp. 2-3).

5.5.3. Síntesis aditiva

La síntesis aditiva es una técnica de síntesis que tiene como base fundamental el análisis de Fourier, esta técnica consiste en sumar varias ondas sinusoidales simples con diferentes amplitudes, con esto se pueden generar sonidos complejos (Romero, 2011, pp. 69).

5.5.3.1. Análisis de Fourier

Este análisis realizado por el físico francés *Jean-Baptiste Joseph Fourier* surge al tratar de interpretar la conducción de calor sobre un anillo de hierro, al resolver el problema matemático se demostró que se puede descomponer una onda compleja cualquiera como una suma finita o infinita de ondas simples.

5.5.4. Síntesis sustractiva

La síntesis sustractiva es una técnica de síntesis que se basa en el uso de señales simples con contenido armónico como por ejemplo la onda de tipo diente de sierra, se la llama síntesis sustractiva ya que consiste en quitar mediante filtros, *LFOs* o envolventes el contenido armónico de dicha señal.

5.5.5. Envolvente (*ADSR*)

La envolvente de una onda sonora es la forma en la que varía su amplitud a través de un periodo de tiempo. Dicha envolvente está dividida en las siguientes secciones.

- *Atack*
- *Decay*
- *Sustain*
- *Realease*

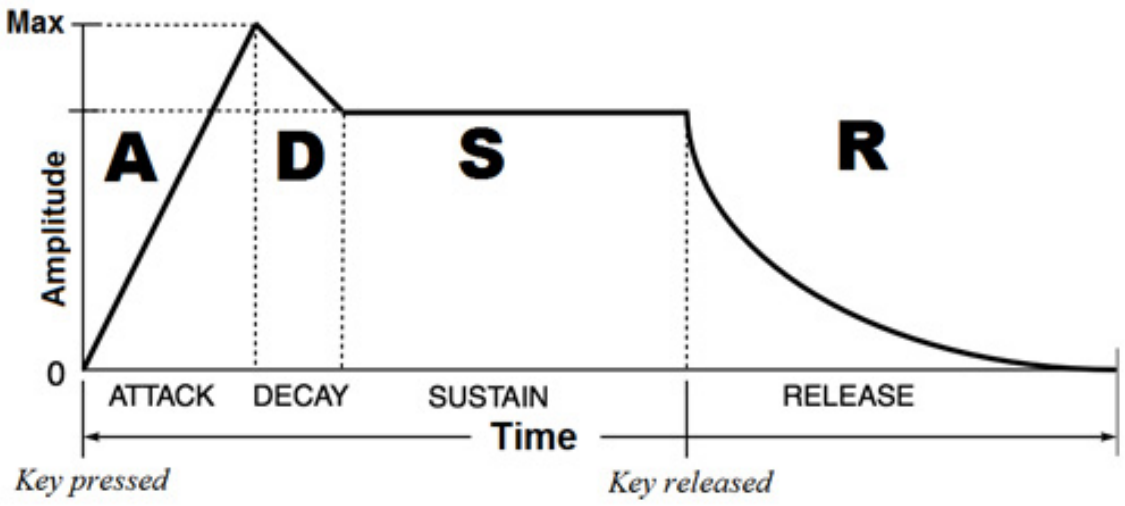


Figura 2. Envolvente de una señal. Recuperada de *songsofthecosmos.* (s.f.).

5.5.5.1. *Atack*

Atack es el tiempo que transcurre desde que la onda sonora inicia hasta que llega a su máxima amplitud.

5.5.5.2. *Decay*

Decay es el tiempo que transcurre desde que la onda sonora llega a su máxima amplitud hasta que llega al nivel de amplitud de sustain.

5.5.5.3. *Sustain*

Sustain es el tiempo en que la onda sonora se mantiene en amplitud constante luego del *decay.*

5.5.5.4. *Realease*

Realease es el tiempo que tarda la onda sonora en finalizar o desvanecerse.

5.5.6. Filtro digital

Un filtro digital es aquel que trabaja con señales discretas, y tiene como función modificar una señal a través de operaciones matemáticas, estas operaciones toman una secuencia de números y devuelve otra totalmente diferente. Se pueden generar filtros a través de un *loop* dentro de un algoritmo.

En audio existen tres tipos de filtros los cuales son: filtro pasa bajos, filtro pasa altos y filtro pasa banda, estos tienen controles principales para su funcionamiento y son: la frecuencia de corte y la pendiente del filtro.

La frecuencia de corte es aquella en la que se empiezan a atenuar las demás frecuencias según el tipo de filtro, por otro lado, la pendiente del filtro permite controlar la cantidad o la forma en la que se atenúan las frecuencias a partir de la frecuencia de corte, las pendientes de filtro más comunes son de: 6 dB/oct, 12 dB/oct y 24 dB/oct.

Por ejemplo, una pendiente de 6 dB/oct. quiere decir que se atenuaran 6 decibeles con cada duplicación de frecuencia.

5.5.7. Filtro pasa bajos y pasa altos

Estos filtros tienen como objetivo dejar pasar frecuencias por debajo (pasa bajos) o por encima (pasa altos) de una frecuencia de corte, las frecuencias son atenuadas modificando así una señal de entrada.

5.5.8. Filtro pasa banda

Este filtro funciona como filtro pasa altos y filtro pasa bajos a la vez permitiendo así tener un rango de frecuencias más discreto, es decir tiene dos frecuencias de corte, así como también dos pendientes de filtro.

5.5.9. *LFO*

LFO (Low Frecuency Oscilator) como su nombre lo indica es un oscilador de baja frecuencia que se utiliza como onda moduladora de cualquier parámetro variable dentro de un sintetizador como por ejemplo modular la amplitud de una señal.

La frecuencia a la que se puede configurar un *LFO* está por debajo de los 20 Hz, de esta manera los efectos que se puedan generar son fácilmente reconocibles.

5.5.10. *Delay* (efecto)

El *delay* como efecto de sonido consiste en tomar una señal y volverla a reproducir luego de un tiempo el cual puede ser preestablecido por el usuario, a esto se le puede añadir el nivel de la señal retardada, así como también la cantidad de retroalimentación de dicha señal, esto puede prolongar más el efecto.

5.6. Desarrollo impulsado por pruebas

El desarrollo impulsado por pruebas es una metodología de desarrollo de aplicaciones móviles que consiste en enfocarse en la versión final tal manera que durante el desarrollo se va realizando pruebas de pequeñas secciones para así depurar sobre la marcha, esto agiliza los resultados y se va manteniendo un cierto orden para avanzar de forma eficiente con el desarrollo de la aplicación (Amaya, 2013, pp. 114).

6. DESARROLLO

6.1. Diseño parche *Pure Data*

Para empezar el diseño del parche de *Pure Data* debemos instalar dicho software en su versión *Vanilla,* una vez hecho esto se procede a crear un nuevo parche. Para empezar, debemos configurar las debidas entradas y salidas tal como se muestra en la Figura 3.

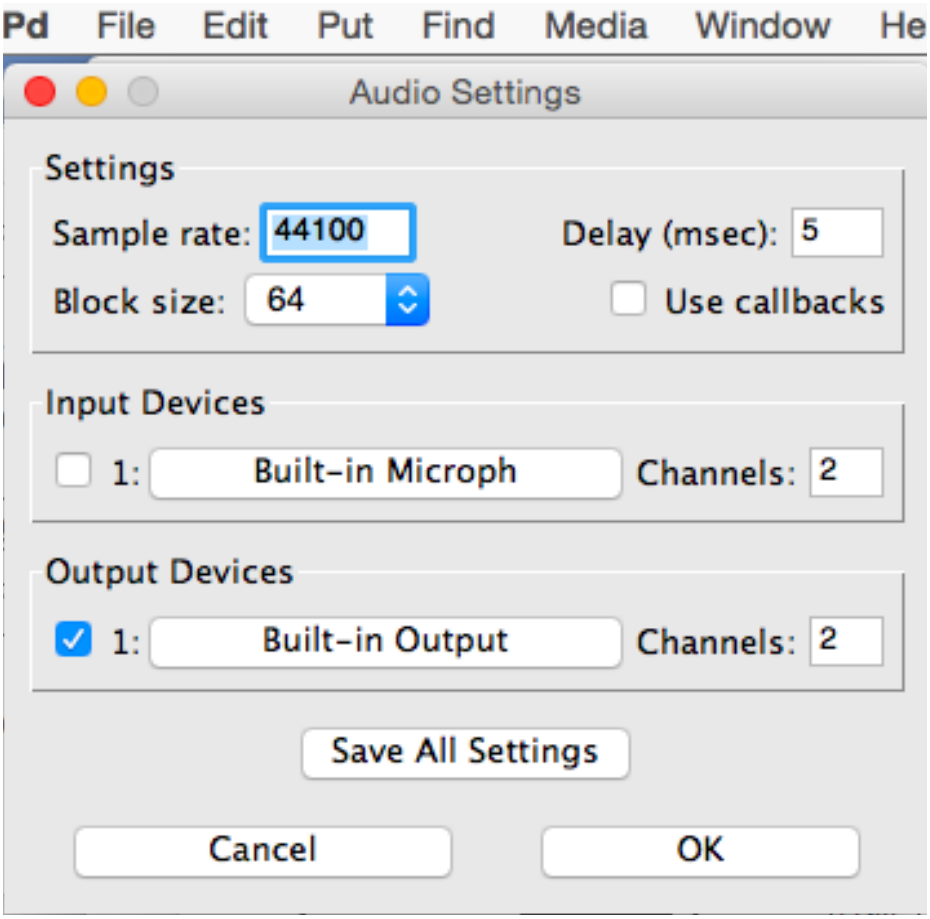


Figura 3. Configuración de audio en *Pure Data.*

Puesto que no será necesario usar una señal analógica de entrada desactivamos los dispositivos de entrada, de esta manera al cargar el parche en cualquier dispositivo nunca usará sus canales de entrada, con esto prevenimos cualquier tipo de problemas en los canales de audio de un dispositivo.

Una vez realizada la configuración de audio se precede a la creación del sintetizador.

6.1.1. Sintetizador

Siguiendo el diagrama de la figura 1 se empieza a programar cada una de las secciones del sintetizador.

La tabla 1 muestra todos los controles que tendrá el sintetizador, así como también los rangos de valores en los que trabajarán, estos controles son aquellos a los que el usuario podrá tener acceso por medio de la interfaz gráfica.

Tabla 1.

Parámetros de los controles del sintetizador.

Control	Rango	Tipo de valor	Unidades
On/Off	0 o 1	Amplitud	-
On/off rudio	0 o 1	Amplitud	-
On/off senoidal	0 o 1	Amplitud	-
On/off sierra	0 o 1	Amplitud	-
Nivel ruido	0 - 1	Amplitud	-
Nivel senoidal	0 - 1	Amplitud	-
Nivel sierra	0 - 1	Amplitud	-
Atack filtro	0 - 2000	Tiempo	milisegundos

Decay filtro	0 - 1	Amplitud	-
Sustain filtro	0 - 2000	Tiempo	Milisegundos
Realease filtro	0 - 2000	Tiempo	milisegundos
Atack	0 - 2000	Tiempo	milisegundos
Decay	0 - 1	Amplitud	-
Sustain	0 - 2000	Tiempo	milisegundos
Realease	0 - 2000	Tiempo	milisegundos
Bypass LFO filtro	0 o 1	Amplitud	
Profundidad LFO filtro	0 o 1	Amplitud	-
Rango LFO filtro	0 - 20	Frecuencia	Hz
Bypass LFO	0 o 1	Amplitud	
Profundidad LFO	0 o 1	Amplitud	-
Rango LFO	0 - 20	Frecuencia	Hz
Bypass filtro	0 o 1	Amplitud	-
Cutoff filtro	0 - 15000	Frecuencia	Hz
Bypass delay	0 o 1	Amplitud	-
Dry/wet delay	0 - 1	Amplitud	-
Delay	0 - 2000	Tiempo	milisegundos
Feedback	0 - 1	Amplitud	-

6.1.2.1. Osciladores

La sección de osciladores es la primera parte del sintetizador y donde comienza el flujo de señal, para programar esta sección comenzamos por crear un *subpatch* que contendrá todos los elementos correspondientes a dicha sección, en la Figura 4 se puede ver el *subpatch*.

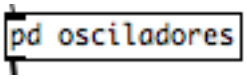


Figura 4. Subpatch osciladores.

La figura 4 muestra la programación de la sección osciladores.

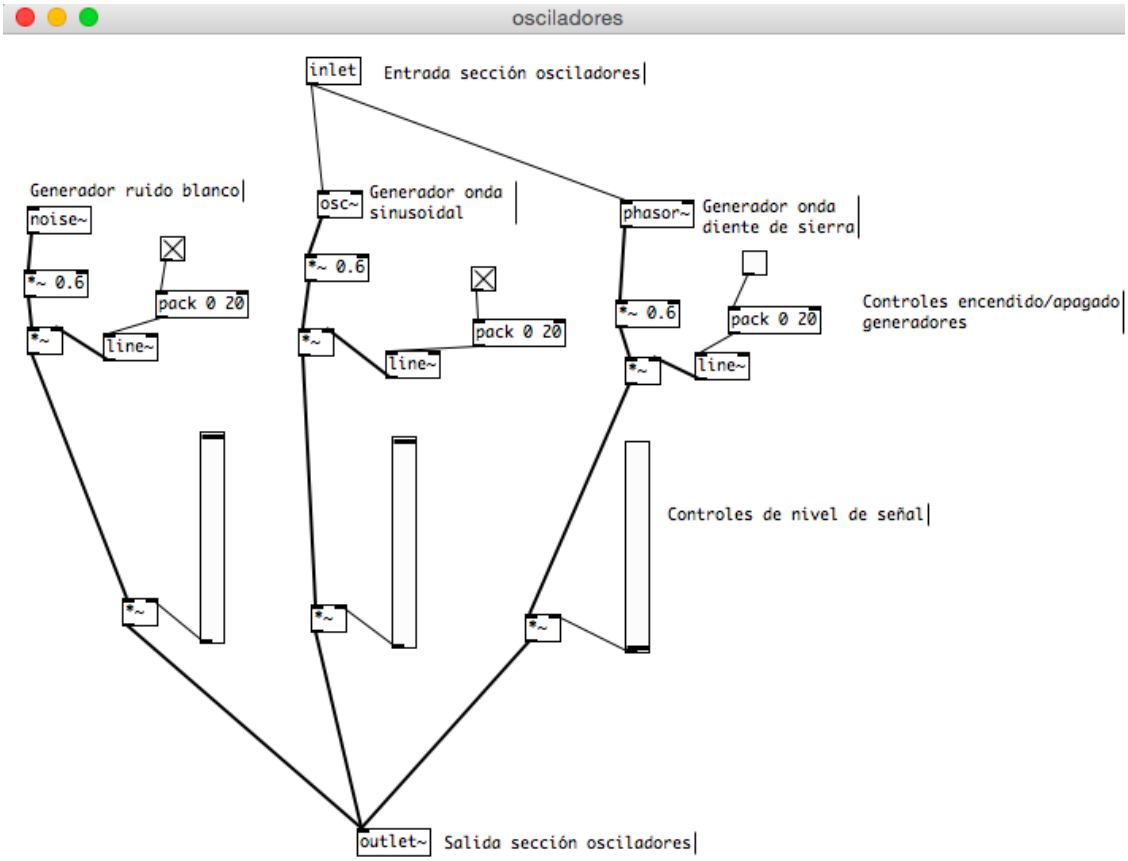


Figura 5. Programación osciladores.

El nivel de amplitud de los generadores de señal va de 0 a 1 siendo 0 el mínimo valor de amplitud y 1 el máximo.

La señal de entrada hacia los osciladores será una nota MIDI correspondiente a una nota musical, la cual será enviada por un teclado MIDI diseñado en

Android Studio, esta nota MIDI deberá ser transformada a frecuencia para esto se programaron los objetos mostrados en la Figura 6.

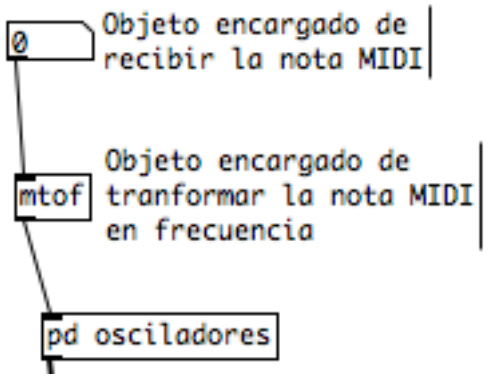


Figura 6. Programación de datos de entrada hacia los osciladores.

De esta manera se completa la programación de la sección de osciladores y se procede a programar la siguiente sección.

6.1.2.2. Filtro

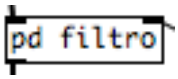


Figura 7. Subpatch filtro

El filtro programado en esta sección es un filtro pasa bajos con una pendiente de 6dB/oct. con una frecuencia de corte variable de 50 Hz a 15000 Hz, se debe tener en cuenta según el diagrama de la Figura 1 cuáles son secciones subsiguientes para así poder crear las entradas y salidas necesarias para el flujo de señal.

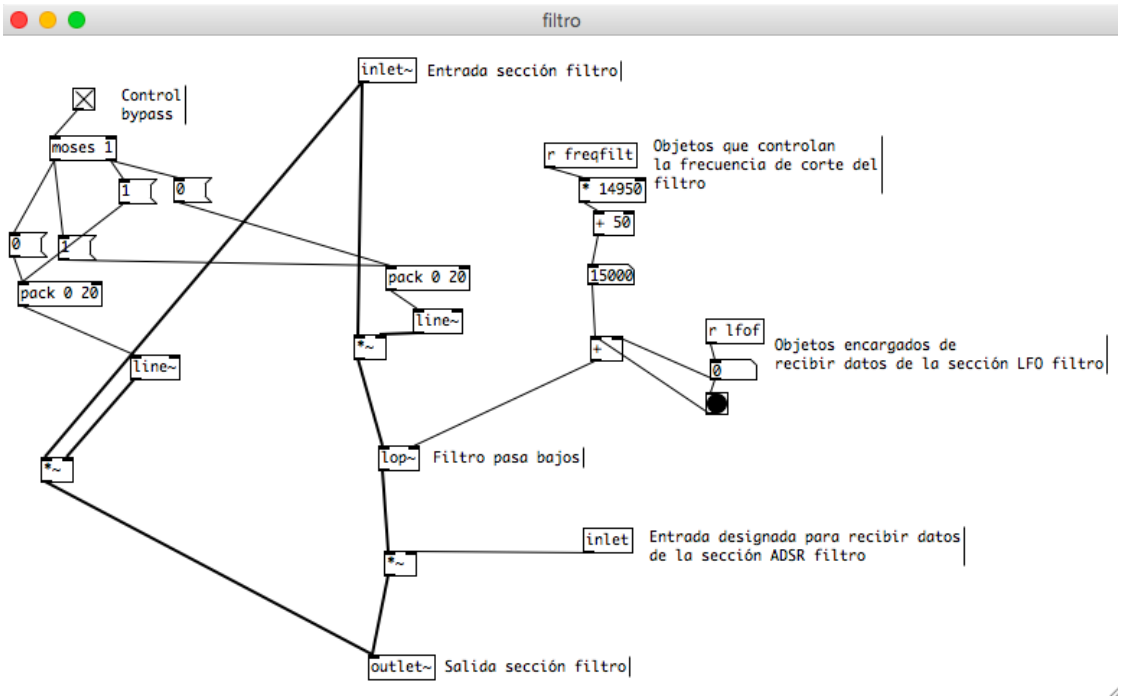


Figura 8. Programación filtro.

6.1.2.3. *ADSR* filtro

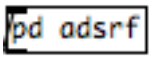


Figura 9. Subpatch ADSR filtro

La programación de la sección *ADSR* filtro permite controlar la envolvente de nivel de amplitud del filtro. La Figura 10 muestra su programación.

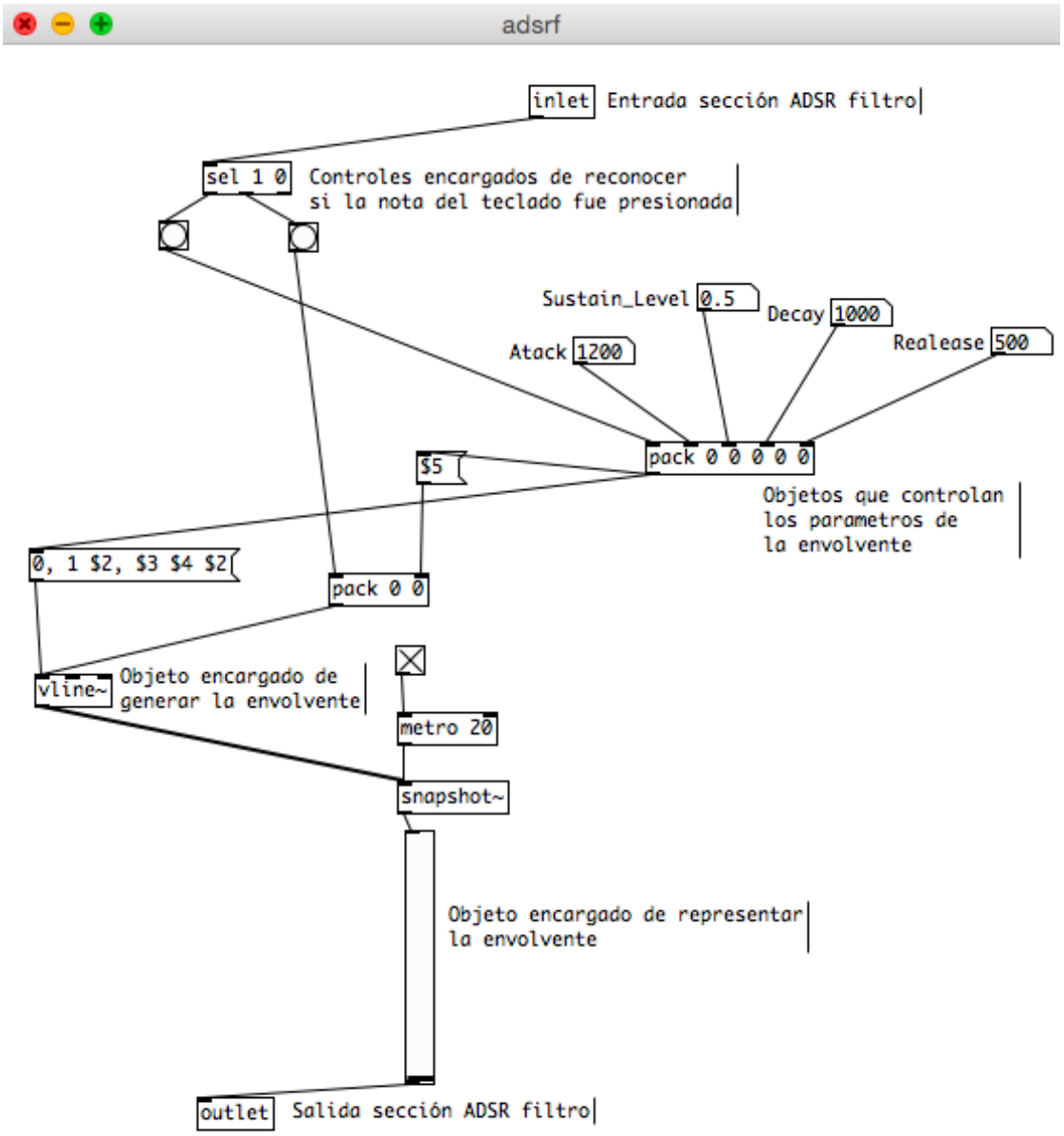


Figura 10. Programación *ADSR* filtro.

Los valores de los parámetros para control se detallan en la Tabla1.

El objeto set nos permite controlar la acción de tecla pulsada, si el valor es 1 cuando la tecla es pulsada envía señal por su salida izquierda y si el valor es 0 cuando la tecla deja de ser presionada envía señal por su salida derecha.

El objeto encargado de enviar 0 o 1 será un botón programado en *Android Studio.*

6.1.2.4. *LFO* filtro

El *LFO* del filtro funciona para modular la frecuencia de corte del filtro pasa bajos.

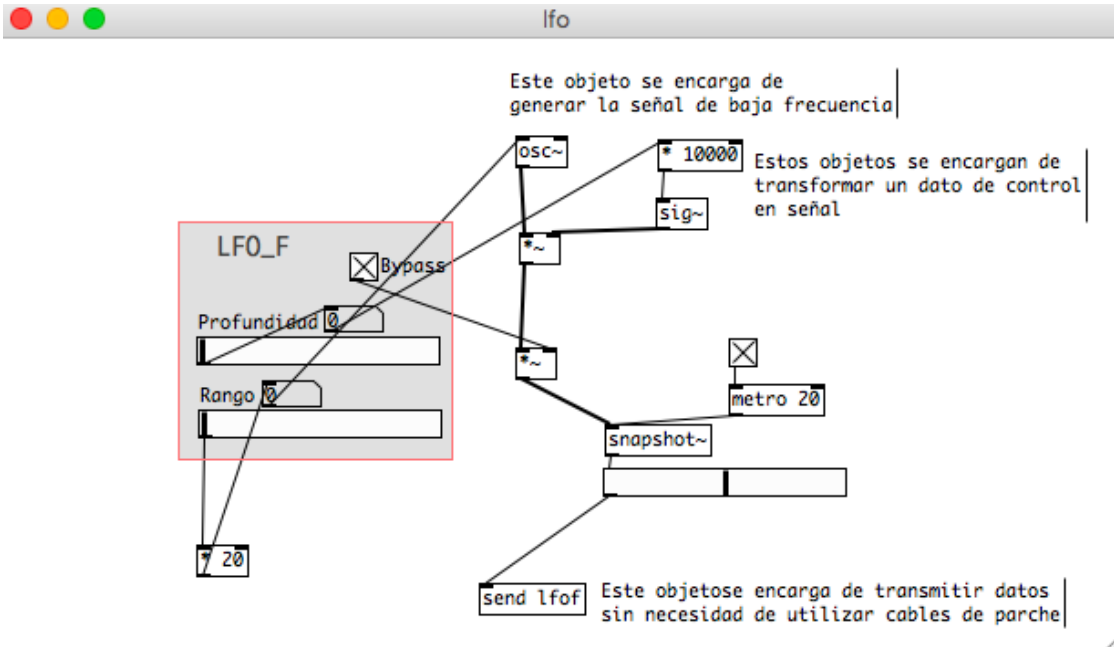


Figura 11. Programación *LFO* filtro.

En la figura 11, se muestran todos los componentes necesarios para poder crear un *LFO*, los valores de los parámetros para controlar el LFO son los siguientes, la profundidad que trabaja en un rango de 0 a 1, que controla la frecuencia de corte de la señal y el rango que controla la frecuencia de oscilación de señal la cual va de 0Hz a 20Hz.

6.1.2.5. *ADSR*

Este controla la envolvente la señal total del sintetizador.

Su estructura es igual al *ADSR* filtro.

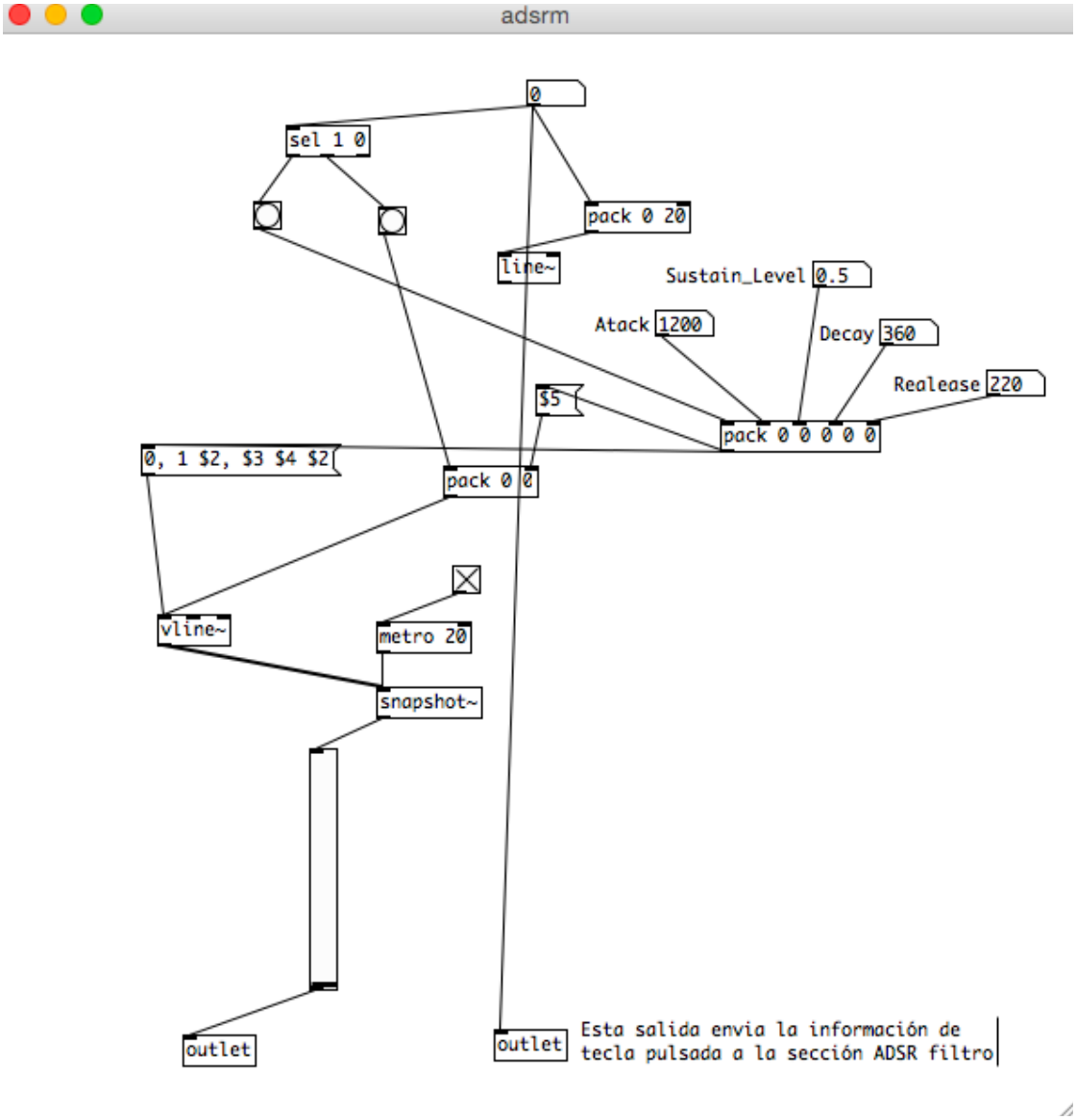


Figura 12. Programación *ADSR.*

6.1.2.6. *LFO*

Este *LFO* modula el nivel de amplitud total de la señal del sintetizador. La estructura de este *LFO* es la misma que el *LFO* filtro.

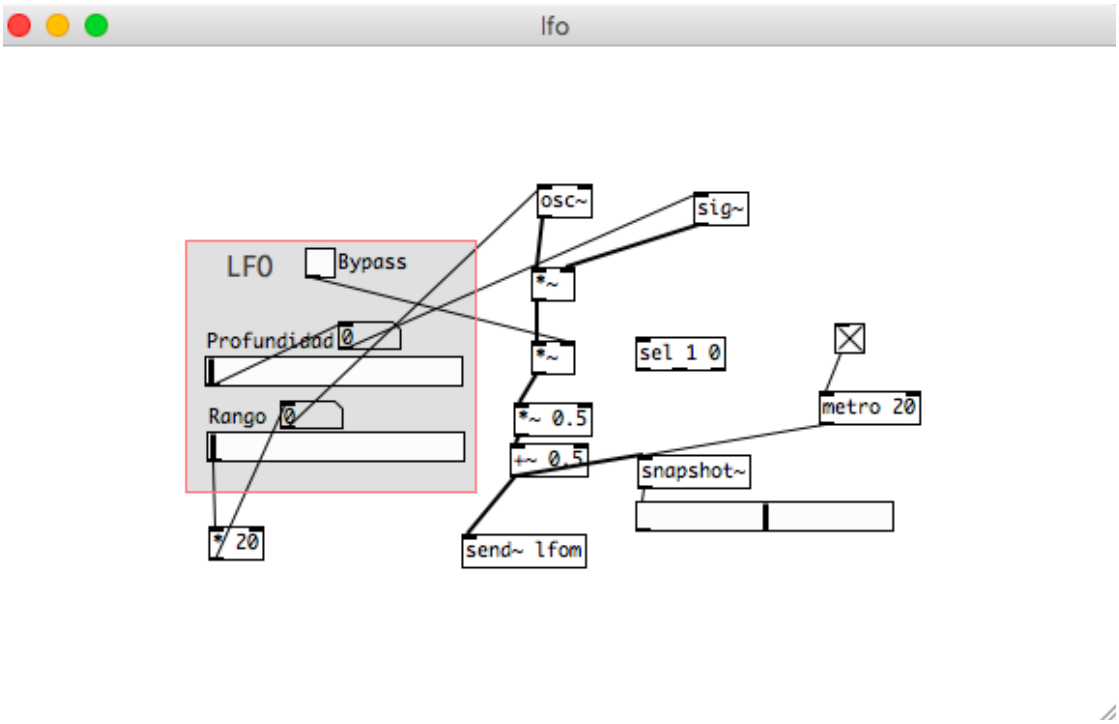


Figura 13. Programación *LFO.*

6.1.2.7. *Delay*

Se ha decidido usar el efecto de audio *delay* ya que es uno de los más usados, y *Pure Data* nos ofrece objetos con los que se puede programar un *delay* sencillo. La Figura 14 muestra su programación.

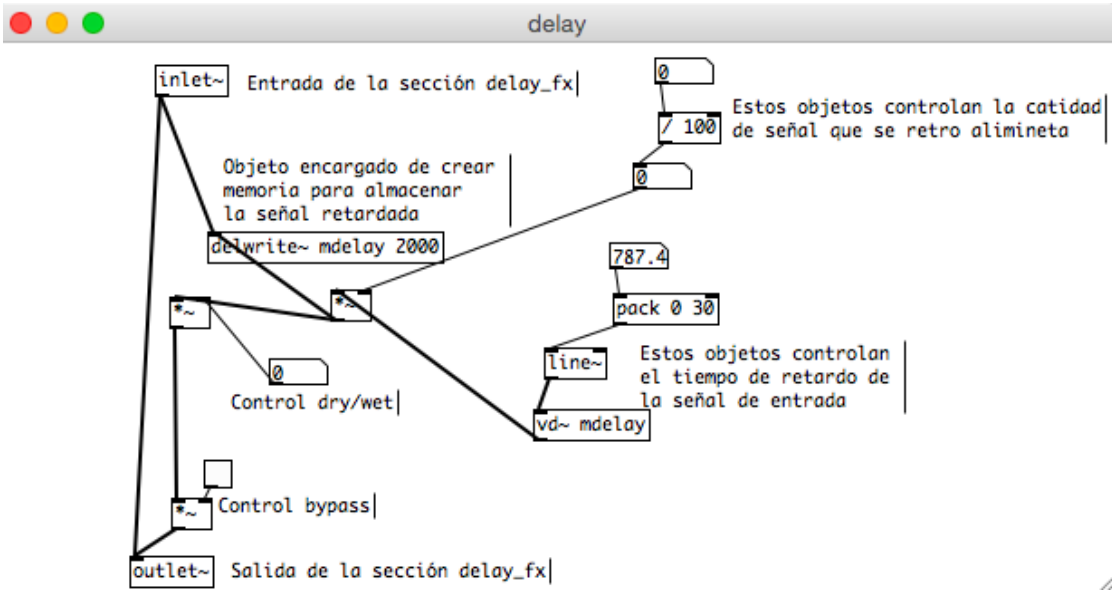


Figura 14. Programación *delay.*

6.1.2.8. Interfaz gráfica del parche

Para proceder a diseñar la interfaz gráfica del parche en *Pure Data* se ordenaron todas las secciones y se interconectan para obtener el flujo de señal detallado en la Figura 1, para esto se creó un s*ubpatch* llamado sintetizador que agrupará todas las secciones programadas.

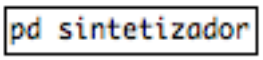


Figura 15. Subpatch sintetizador.

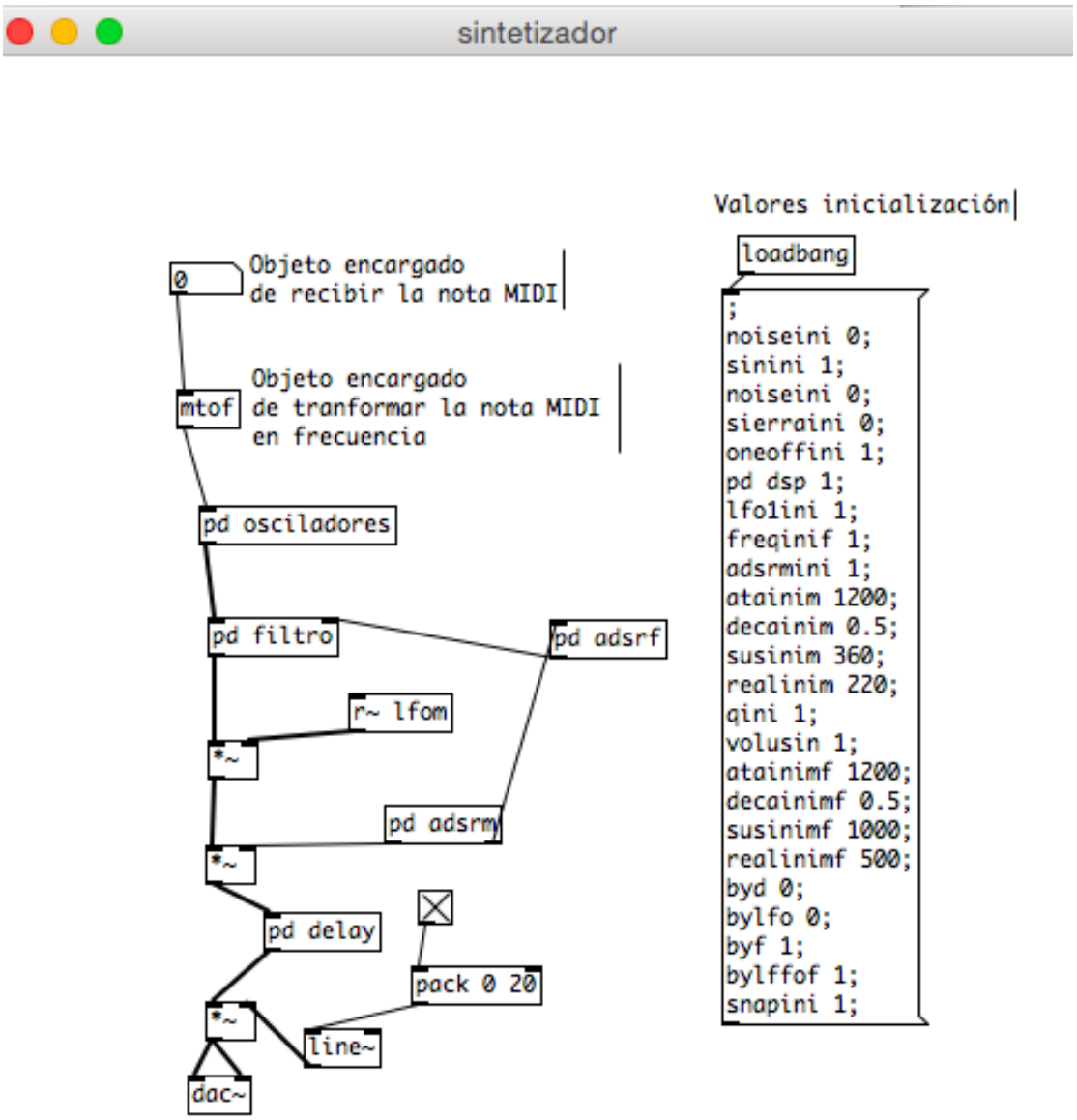


Figura 16. Programación sintetizador.

En la Figura 16 se muestra toda la programación del sintetizador, se pueden observar las diferentes secciones interconectadas según el flujo de señal establecido.

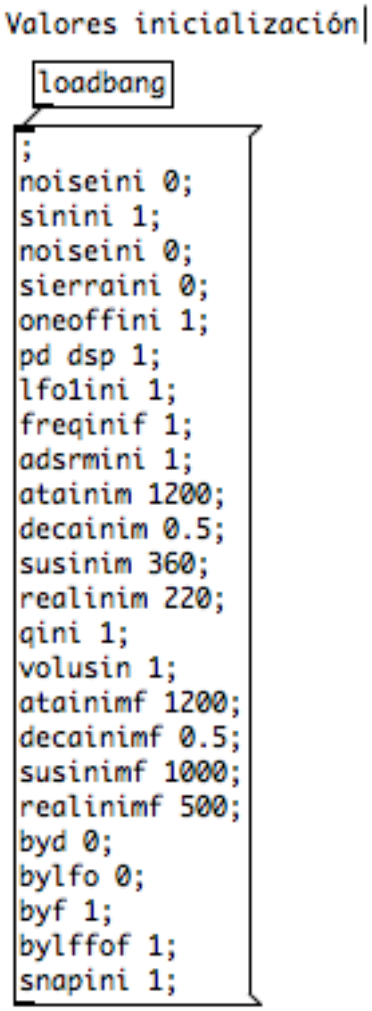


Figura 17. Valores de inicialización.

La Figura 17 muestra la programación de los valores por defecto que tendrá el sintetizador al momento de que el parche cargue.

Una vez organizadas todas las secciones se procede a diseñar la interfaz gráfica para esto se utilizó objetos *canvas* los cuales nos permiten mostrar los distintos controles a los que el usuario podrá tener acceso.

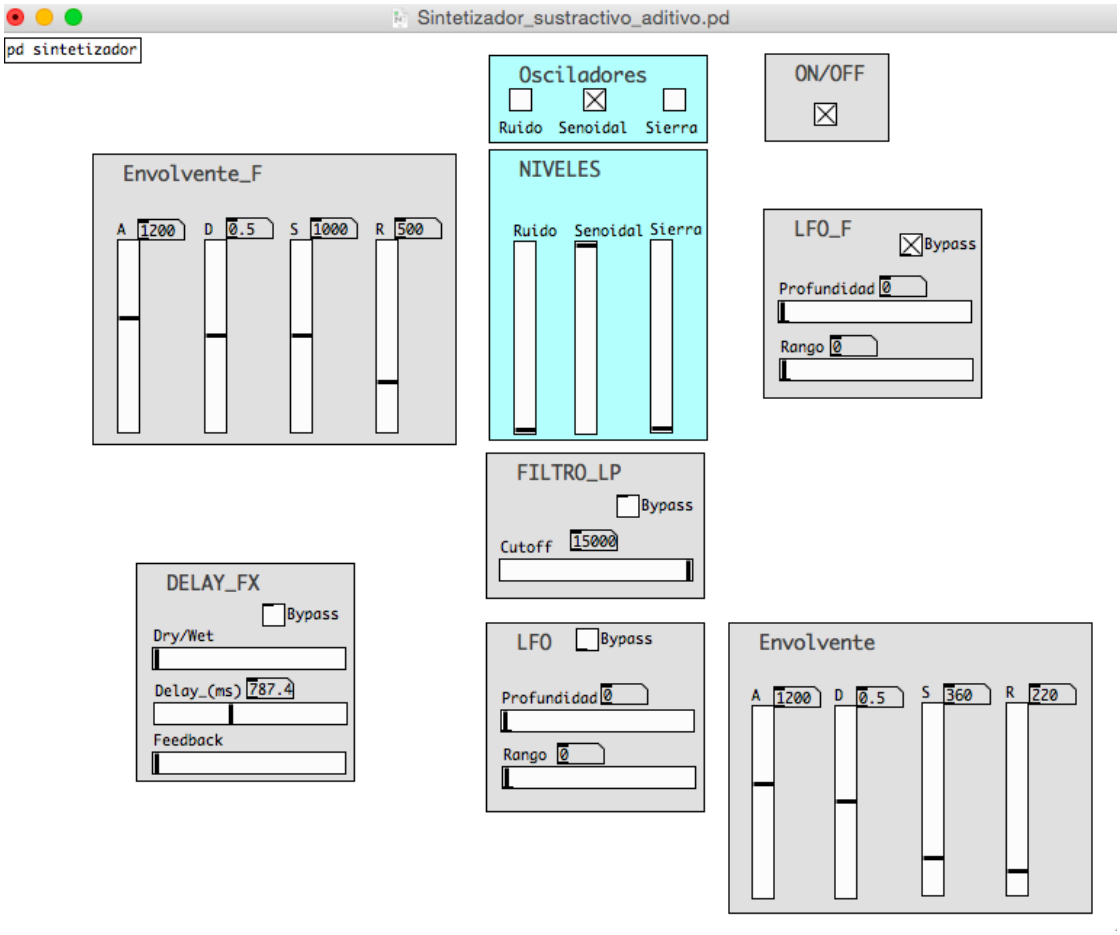


Figura 18. Interfaz gráfica del parche sintetizador.

En la Figura 18, se muestran todos los controles que tiene el sintetizador, cada uno de estos controles tiene asignada una etiqueta la cual se encargará de recibir los datos provenientes de la programación en *Android Studio.*

La Tabla 2 muestra los controles con sus respectivas etiquetas.

Tabla 2.

Controles con sus respectivas etiquetas de control.

Control	**Etiqueta**
On/Off	oneoffini
On/off rudio	noiseini
On/off senoidal	sinoini
On/off sierra	sierraini

Nivel ruido	volunoise
Nivel senoidal	volusin
Nivel sierra	volusier
Atack filtro	atainimf
Decay filtro	decainimf
Sustain filtro	susinimf
Realease filtro	realinimf
Atack	atainim
Decay	decainim
Sustain	susinim
Realease	realinim
Profundidad LFO filtro	deeplfof
Rango LFO filtro	ratelfof
Bypass LFO	bylfo
Profundidad LFO	deeplfo
Rango LFO	ratelfo
Bypass filtro	byf
Cutoff filtro	freqinim
Bypass delay	byd
Dry/wet delay	dwc
Delay	delayc
Feedback	feedc
Nota MIDI	pitch

6.2. Diseño de la interfaz gráfica con *Android Studio*

Android Studio permite crear la aplicación que podrá ser ejecutada en cualquier dispositivo móvil que disponga de sistema operativo *Android*, en este proyecto permitirá generar una interfaz gráfica para que el usuario pueda acceder a todos los controles del sintetizador.

6.2.1. Configuraciones previas

Para comenzar a diseñar la interfaz gráfica con *Android Studio* primero se debe instalar en un computador todas las herramientas necesarias para el desarrollo, tales como versión de *Android Studio*, *SDK* y *JDK*.

La Tabla 3 muestra las herramientas utilizadas.

Tabla 3.

Herramientas usadas para el diseño de la interfaz gráfica.

Herramienta	**Versión**
Android Studio	2.3.1
JDK	7
SDK	API 24: Android 7.0 (Nougat)

Una vez listas las herramientas se crea un proyecto nuevo en *Android studio* con la configuración deseada.

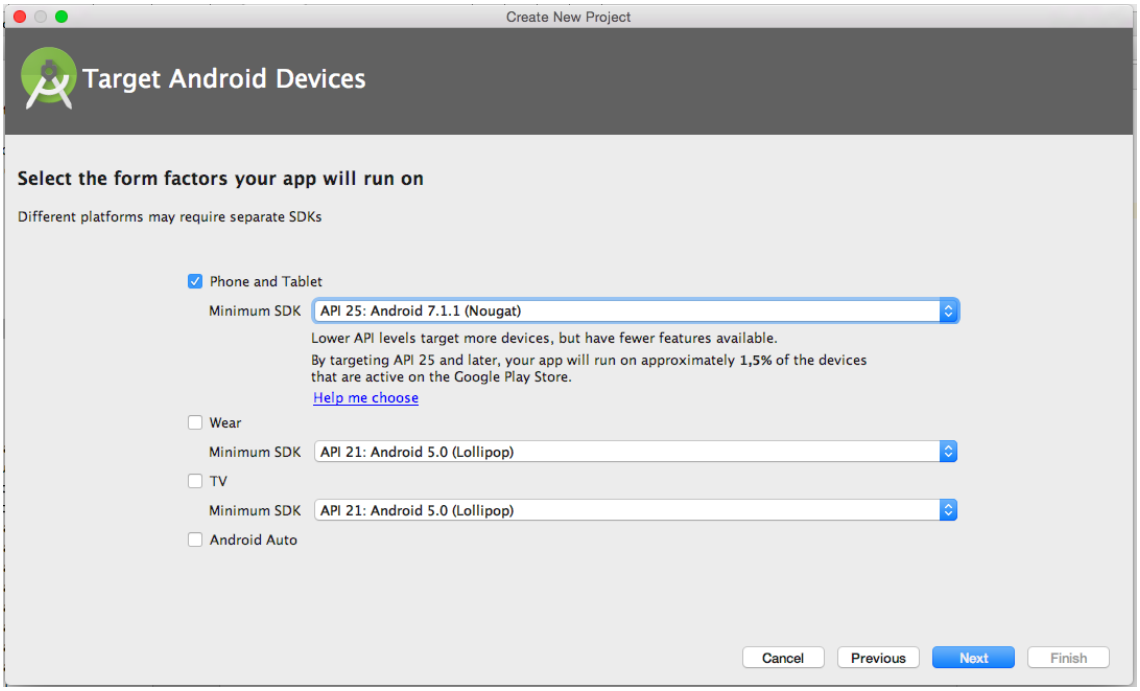


Figura 19. Configuración SDK proyecto nuevo.

Con el proyecto creado se procede a crear la interfaz gráfica para sistema operativo *Android,* la cual tiene como función poder controlar los parámetros del sintetizador programado en *Pure Data,* desde un dispositivo móvil.

6.2.2. Diseño de secciones de la interfaz gráfica

La aplicación consta de tres secciones principales, la primera que contendrá el menú, la segunda donde se cargará los controles del sintetizador y la tercera que contendrá el teclado *MIDI*.

La distribución de las secciones mencionadas anteriormente se la hará con la aplicación configurada en tipo *Lansdscape* es decir con el dispositivo en orientación horizontal. La Figura 20 muestra el esquema de distribución.

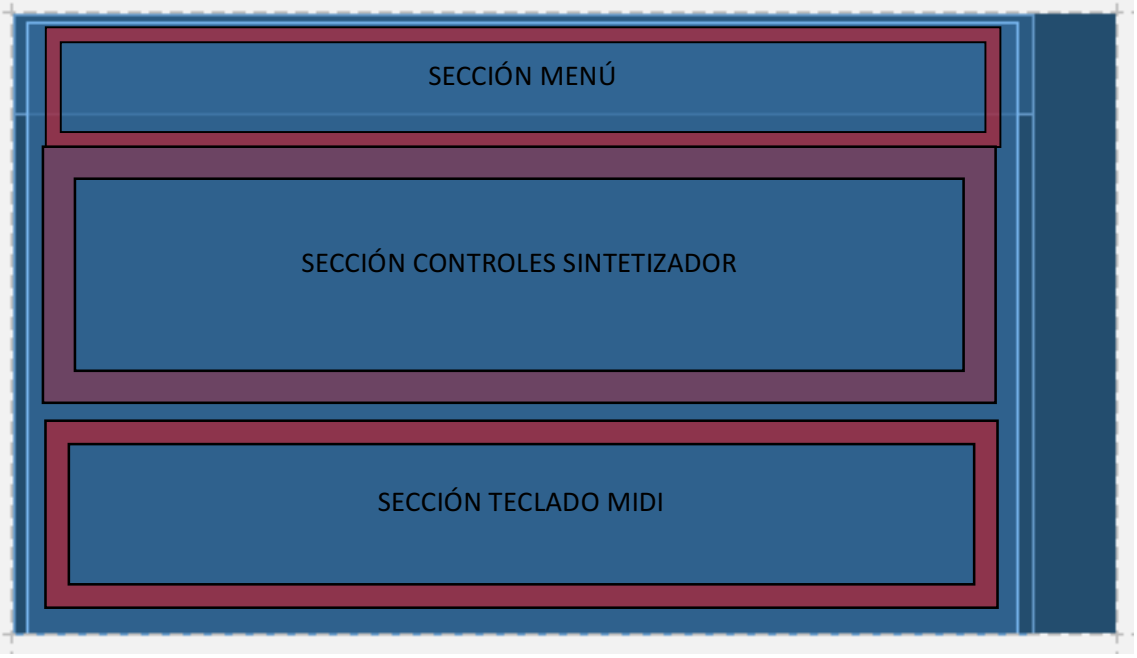


Figura 20. Distribución de las secciones de la aplicación en la pantalla del dispositivo.

6.2.3.1. Sección menú

Para programar esta sección se utiliza primero la vista de diseño de *Android Studio* para la distribución de los diferentes controles.

En la actividad principal se crearán todos fragmentos que contendrán cada sección de la aplicación.

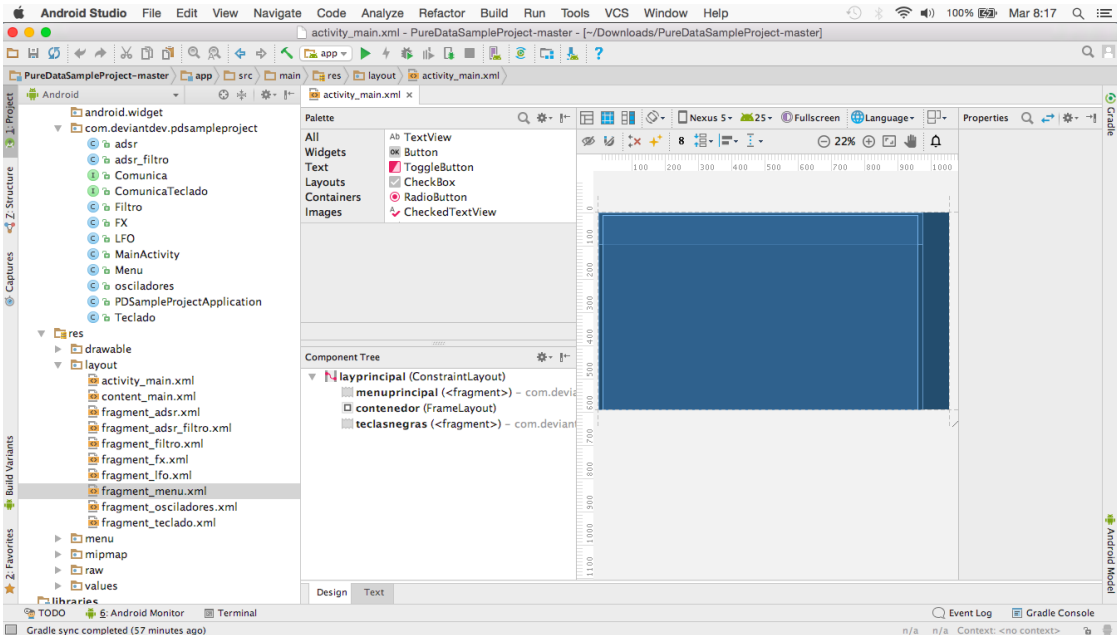


Figura 21. Creación de fragmentos para cada sección de la aplicación.

En el fragmento de menú se crean botones los cuales llamarán a cada sección del sintetizador. La Figura 22 muestra la distribución de los botones.

Figura 22. Fragmento menú.

El bloque de código para este fragmento se encuentra en el Anexo A.

6.2.3.2. Sección controles de sintetizador

En esta sección aparecerán los distintos fragmentos que contengan los controles de cada sección del sintetizador según el botón presionado en el menú. El código de esta sección se detalla en el Anexo B.

6.2.3.2.1. Fragmento osciladores

Las vistas de este fragmento permiten controlar los niveles de señal de cada oscilador, así como también activar e inactivar cada uno de ellos.

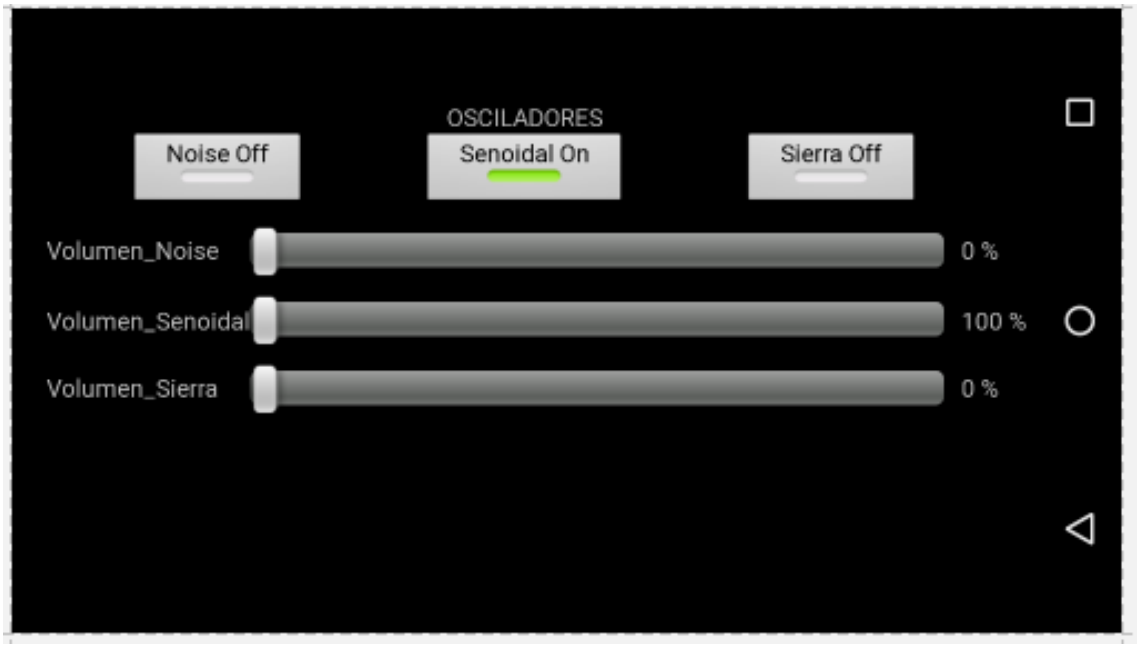


Figura 23. Fragmento osciladores.

El código *XML* de este fragmento está detallado en el Anexo C.

6.2.3.2.2. Fragmento filtro

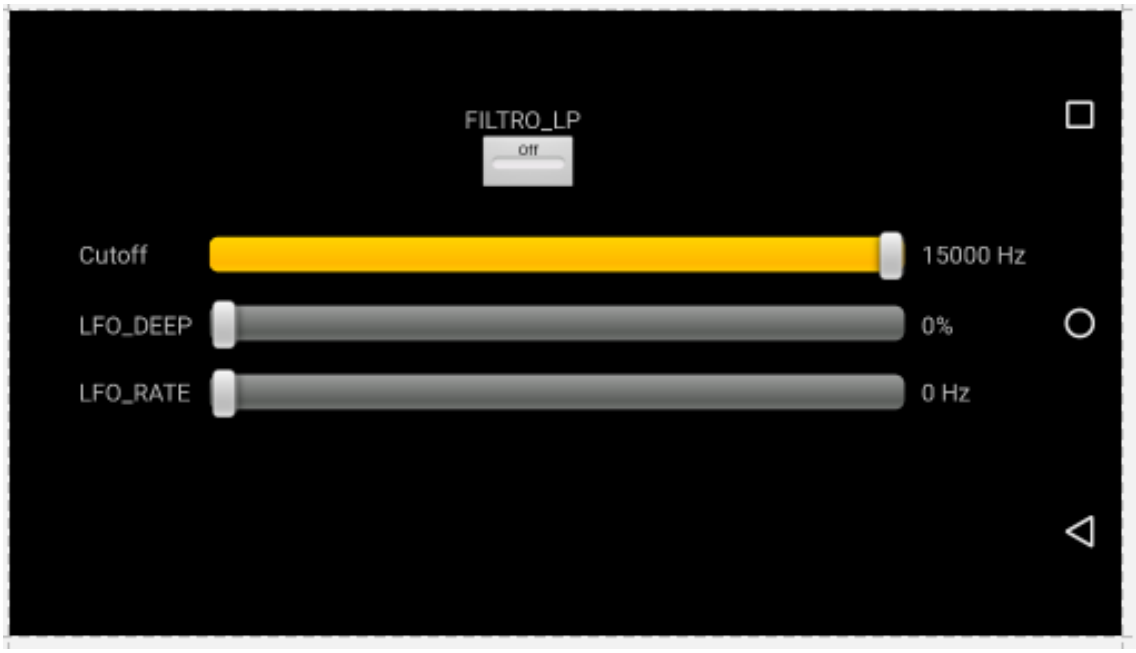


Figura 24. Fragmento filtro.

Las vistas controlan el encendido del filtro, la frecuencia de corte del filtro, la profundidad y rango del *LFO.*

El código XML está detallado en el Anexo D.

6.2.3.2.3. Fragmento *ADSR* Filtro

Este fragmento tiene los controles para utilizar la envolvente del filtro.

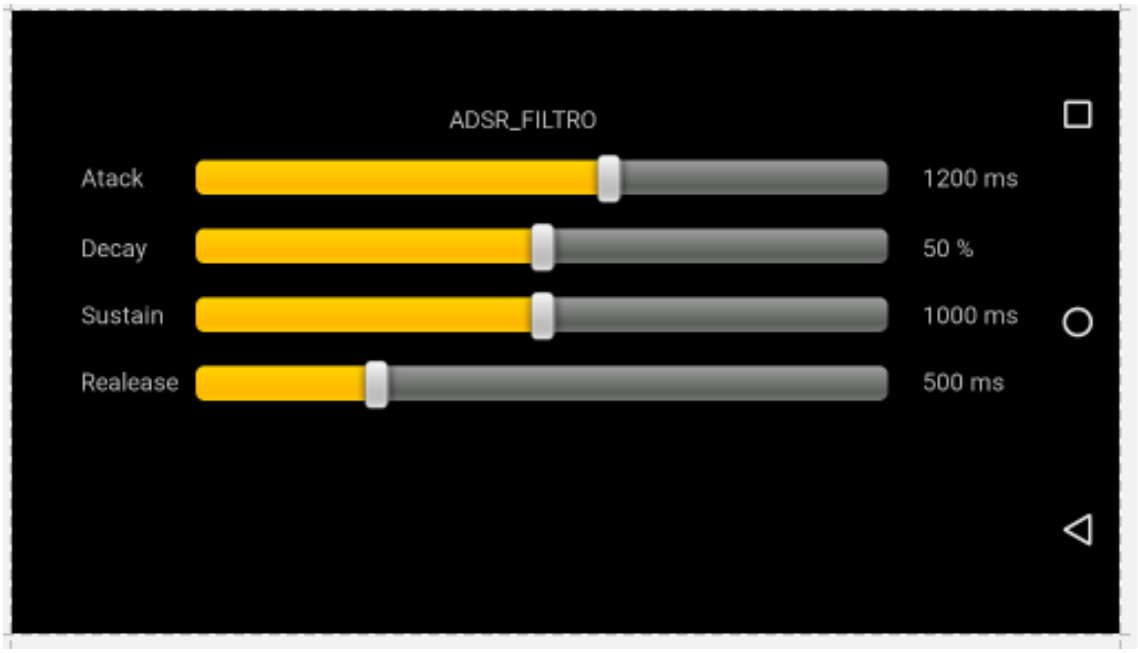


Figura 25. Fragmento *ADSR* filtro.

Código *XML* para el fragmento *ADSR* filtro se detalla en el Anexo E.

6.2.3.2.4. Fragmento *LFO*

Figura 26. Fragmento *LFO.*

En la figura 26 se muestran las vistas para controlar la profundidad y el rango de nivel para utilizar el *LFO* del sintetizador.

El código *XML* es el detallado en el Anexo F.

6.2.3.2.5. Fragmento *ADSR*

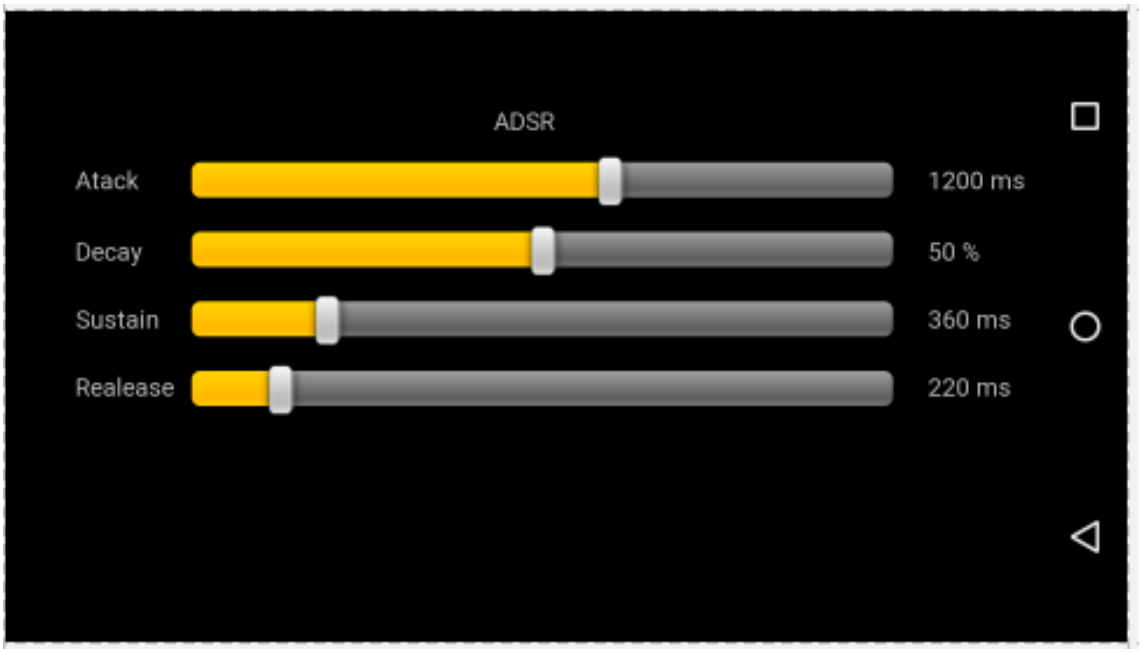


Figura 27. Fragmento *ADSR*

Este fragmento contiene deslizadores que permitirán el control de la envolvente principal del sintetizador.

El código *XML* se detalla en el Anexo G.

6.2.3.2.5. Fragmento *FX*

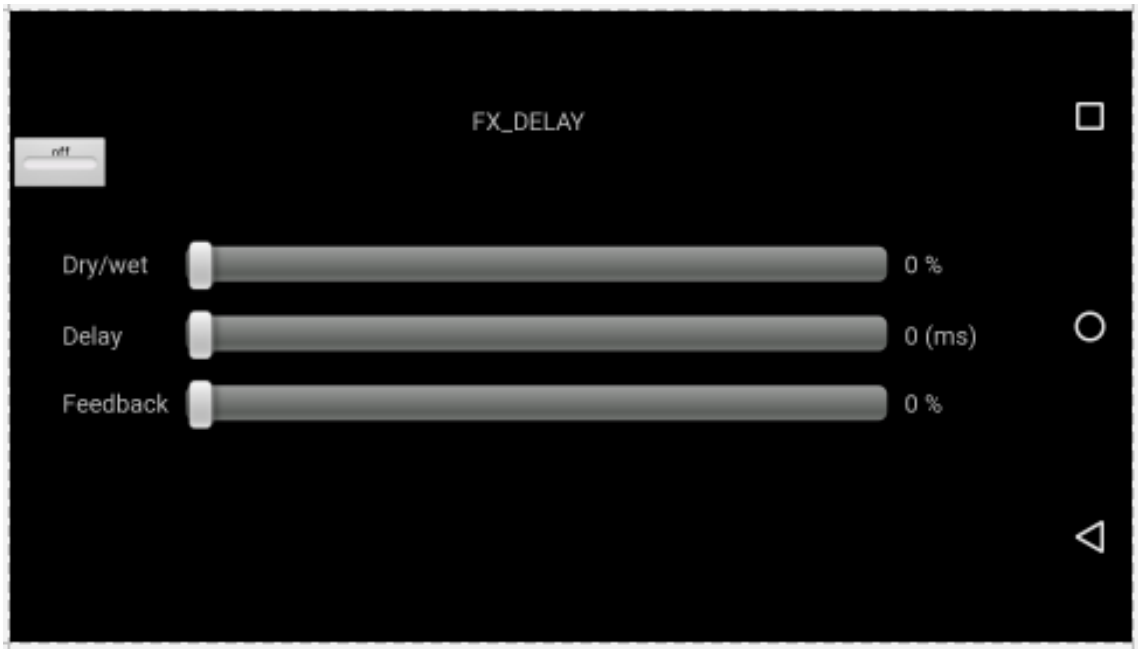


Figura 28. Fragmento *FX.*

El fragmento *FX* contiene las vistas para utilizar el efecto de *delay,* entre las cuales están el nivel de efecto, el tiempo de retardo y la retroalimentación, el código *XML* se muestra en el Anexo H.

6.2.3.3. Sección teclado *MIDI*

En este fragmento se dibuja un teclado *MIDI* de tres octavas que será el encargado de enviar el número *MIDI* hacia la entrada de los osciladores.

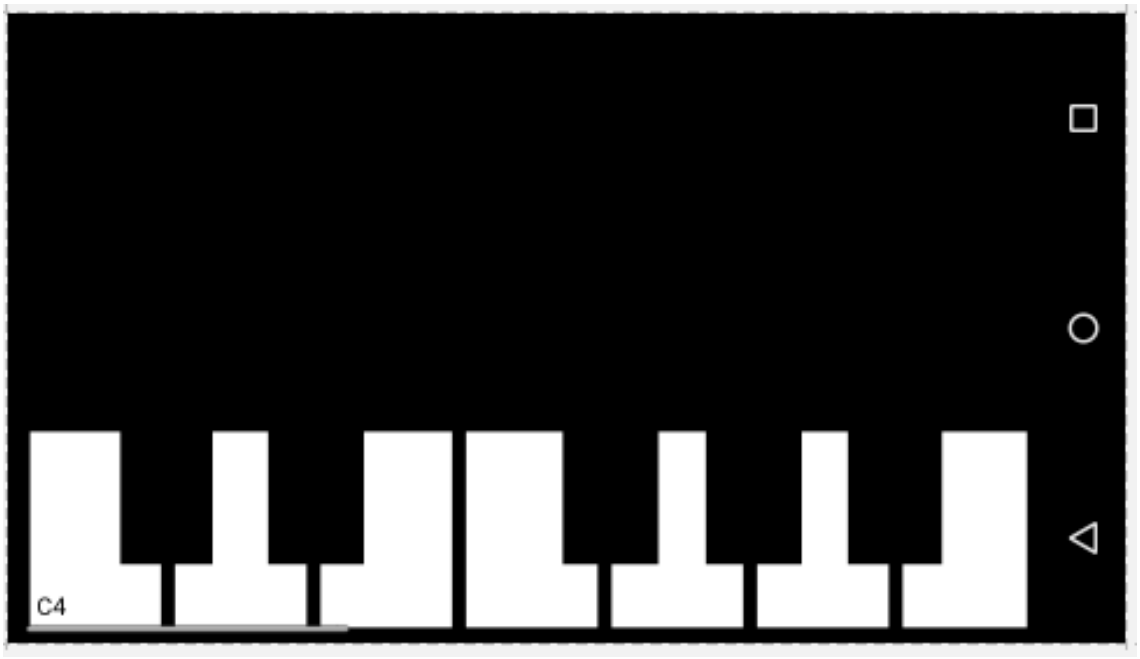


Figura 29. Fragmento teclado.

La programación para el teclado se muestra en el Anexo I.

6.3. Enlace y comunicación del parche de *Pure Data* con *Android Studio*

Una vez diseñada la interfaz gráfica de la aplicación se procede a escribir el código para que las diferentes vistas interactúen con el parche de sintetizador de *Pure Data;* esta programación se la realiza en lenguaje *Java.*

El primer paso para esto es crear una interfaz para que los fragmentos puedan comunicarse con la actividad principal, la cual se encarga de enviar los datos de control al parche de *Pure Data.*

El nombre de la interfaz será “Comunica” y su código de programación se muestra en el Anexo J.

Adicionalmente a esta interfaz se creó una independiente para el teclado *MIDI* llamada “ComunicaTeclado”, el código se muestra a continuación:

```
public interface ComunicaTeclado {

//Función que permiten enviar el número MIDI y el estado de cada tecla hacia la actividad principal.

    public void controltecla (int quetecla, float estado);
```

}

Una vez configuradas las interfaces se procede a escribir el código para que las vistas dentro de cada fragmento generen datos cuando el usuario interactúe con ellas.

6.3.1. Programación *Java* para la comunicación entre *Android Studio* y *Pure Data*

Este código pone en funcionamiento las diferentes secciones de la aplicación, permite comunicar todas las vistas y generar datos a partir de como el usuario las modifique, de igual manera permite inicializar los datos de cada control al momento de ejecutar la aplicación, cada sección usa una interfaz en común llamada “Comunica” mencionada anteriormente, la cual concentra todos los datos en la actividad principal encargada de la comunicación final con el algoritmo del sintetizador.

Se muestra el código *Java* para cada sección en los anexos K, L, M, O y P.

6.3.2. Programación *Java* para la actividad principal

En la actividad principal contendrá el código general encargado de enviar y recibir datos necesarios para que el parche de *Pure Data* funcione.

La comunicación surge utilizando las etiquetas asignadas a cada control las cuales están mostradas en la Tabla 2, es decir cada vista de control en la interfaz gráfica tiene asignado un control dentro del parche de *Pure Data,* para enviar los datos se designa la etiqueta seguida del valor a enviar.

Para que la comunicación ocurra de manera correcta debemos incluir la librería *pd-for-android* y el parche del sintetizador dentro de nuestro proyecto, la cual se explica en el siguiente apartado.

6.3.6.1. Añadir la librería *pd-for-android* al proyecto de *Android Studio*

Para esto se debe acceder al archivo build.glade y agregar el repositorio JCenter de la siguiente manera:

buildscript {

```
    repositories {
        jcenter() //Repositorio JCenter
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.1'
    }
}
allprojects {
    repositories {
        jcenter()//Repositorio JCenter
}
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

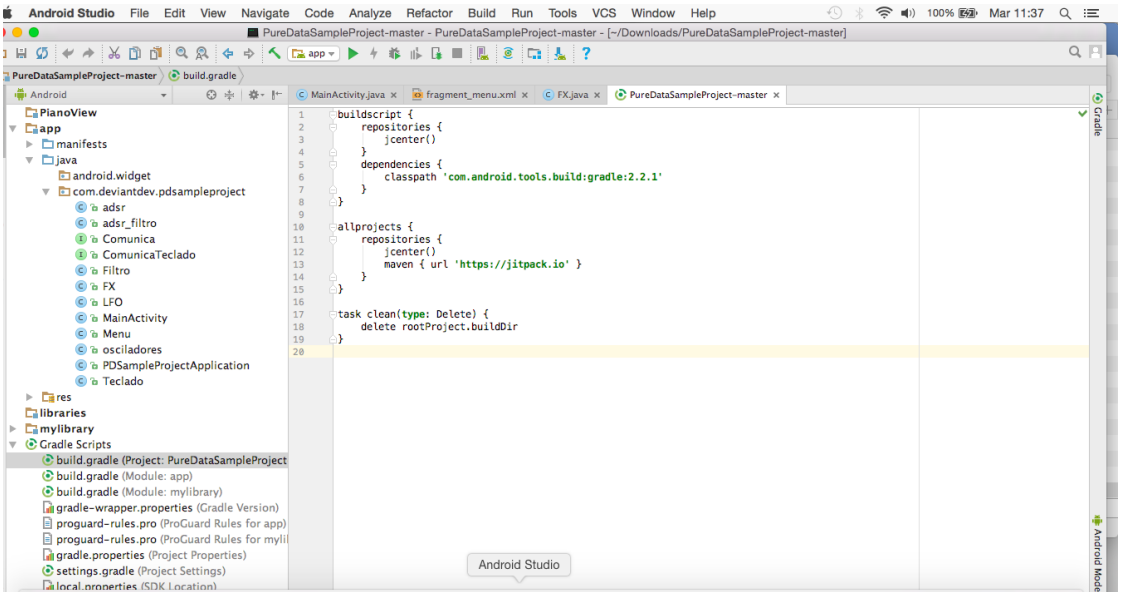


Figura 30. Agregar repositorio JCenter.

Luego se debe agregar una dependencia en el módulo de la aplicación.

```
android {
    compileSdkVersion 24
    buildToolsVersion '25.0.0'
    defaultConfig {
        applicationId "com.deviantdev.pdsampleproject"
        minSdkVersion 19
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.android.support:design:24.2.1'
```

compile 'org.puredata.android:pd-core:1.0.1'//Dependencia para acceder a la librería *pd-for-android*

compile 'com.android.support.constraint:constraint-layout:1.0.2'

compile 'com.android.support:support-v4:24.2.1'

compile 'com.github.marcinmoskala:ArcSeekBar:0.31'

testCompile 'junit:junit:4.12'

}

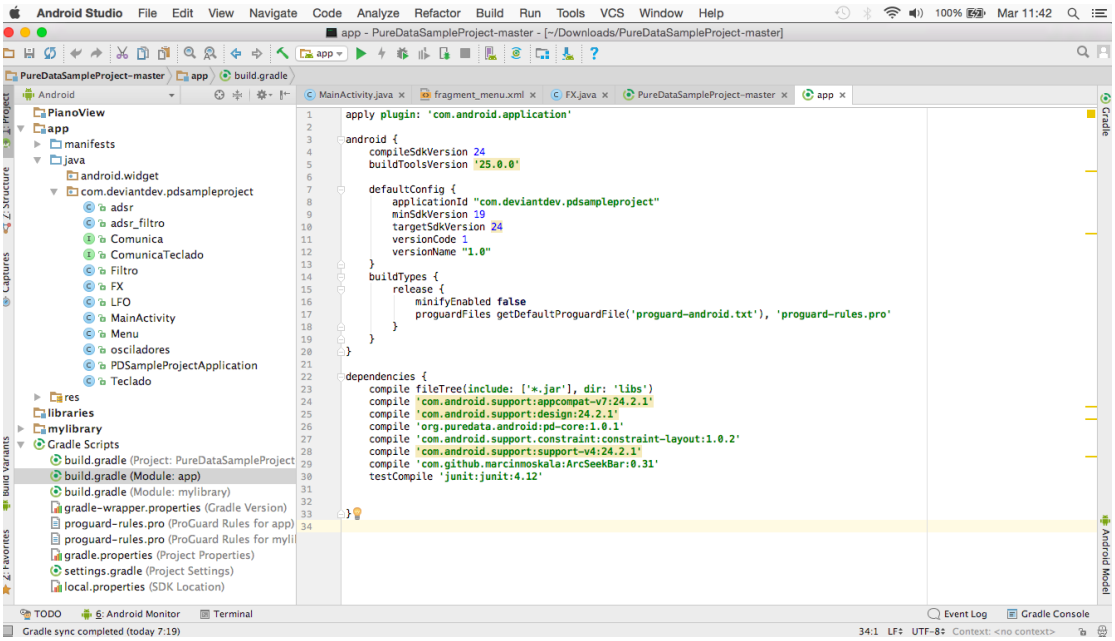


Figura 31. Agregar dependencia *pd-for-android.*

Luego de haber agregado la librería *pd-for-android,* se debe agregar el parche de *Pure Data* para esto debemos comprimir en ".zip" el archivo del sintetizador que tiene como extensión ".pd"; una vez hecho esto se debe crear una carpeta con nombre "raw" dentro del la carpeta "res" que se encuentra en el proyecto de *Android Studio, y* finalmente pegamos el archivo comprimido del parche dentro de la carpeta "raw"; en la Figura 31 se muestra el directorio en el que se debe agregar el parche comprimido.

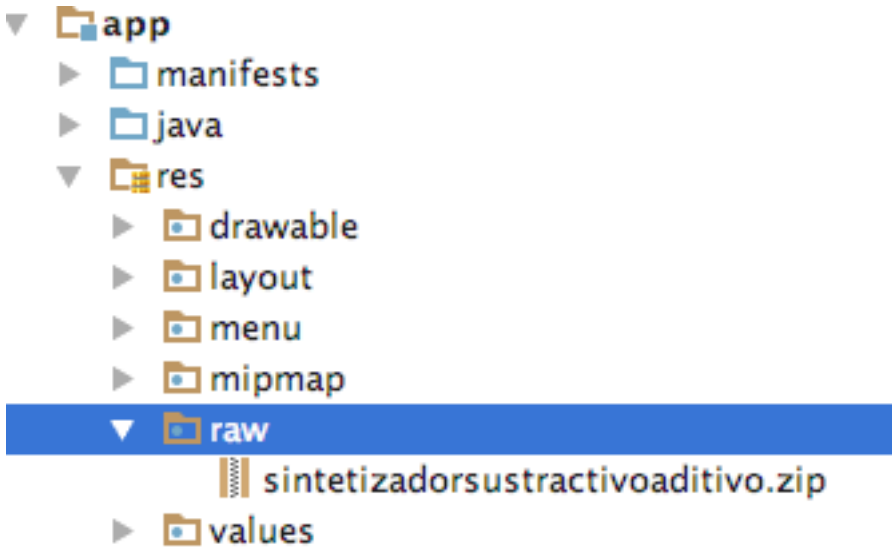


Figura 32. Agregar el parche de *Pure Data* dentro del proyecto de *Android Studio*

Ahora se procede a escribir el código para enviar los datos desde la actividad principal hacia el parche de *Pure Data.*

El código se muestra en el Anexo Q.

6.3.3. Compilación de la aplicación

Con todo el código listo se procede a compilar la aplicación para esto usamos la herramienta que nos proporciona *Android Studio.*

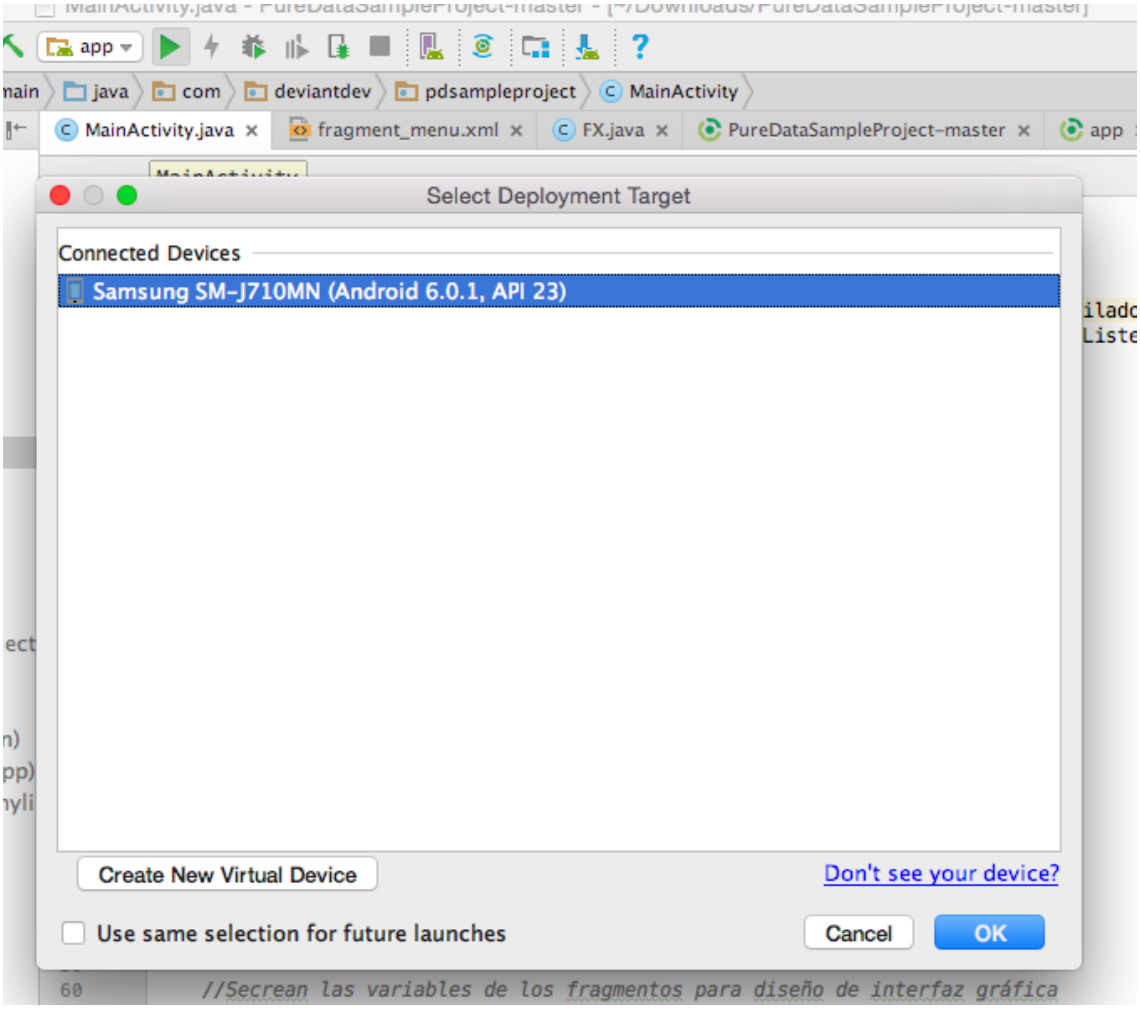


Figura 33. Herramienta compilación y depuración de *Android Studio.*

7. RESULTADOS

7.1. Aplicación de síntesis sonora para sistema operativo *Android*

La aplicación se compila en un dispositivo real y se prueban cada una de las secciones, a continuación, se muestran capturas de pantalla de la aplicación ejecutándose en un dispositivo móvil.

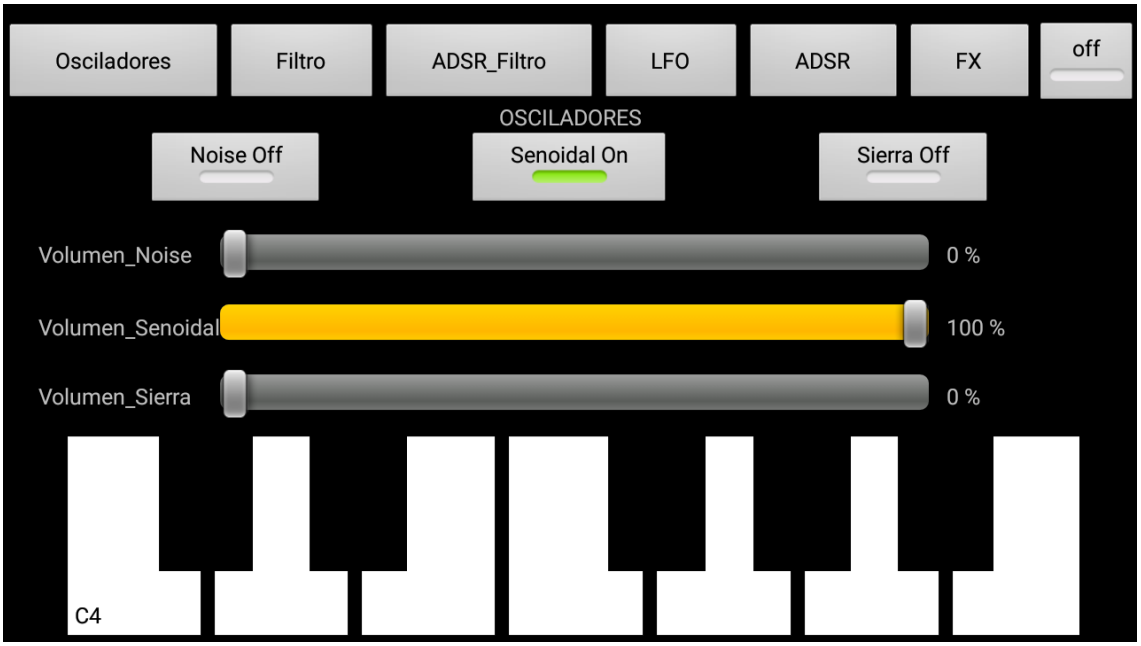


Figura 34. Aplicación síntesis sonora sección: osciladores.

En esta sección se puede controlar los niveles de señal de cada uno de los osciladores, así como también poderlos activar o desactivar.

El funcionamiento de los osciladores empieza al pulsar los botones del teclado *MIDI* ubicado en la parte inferior, el cual cuenta con tres octavas a las cuales se pueden acceder deslizando de manera horizontal dicha sección.

En la parte superior se puede ver el menú general que permite acceder a todas las funciones del sintetizador.

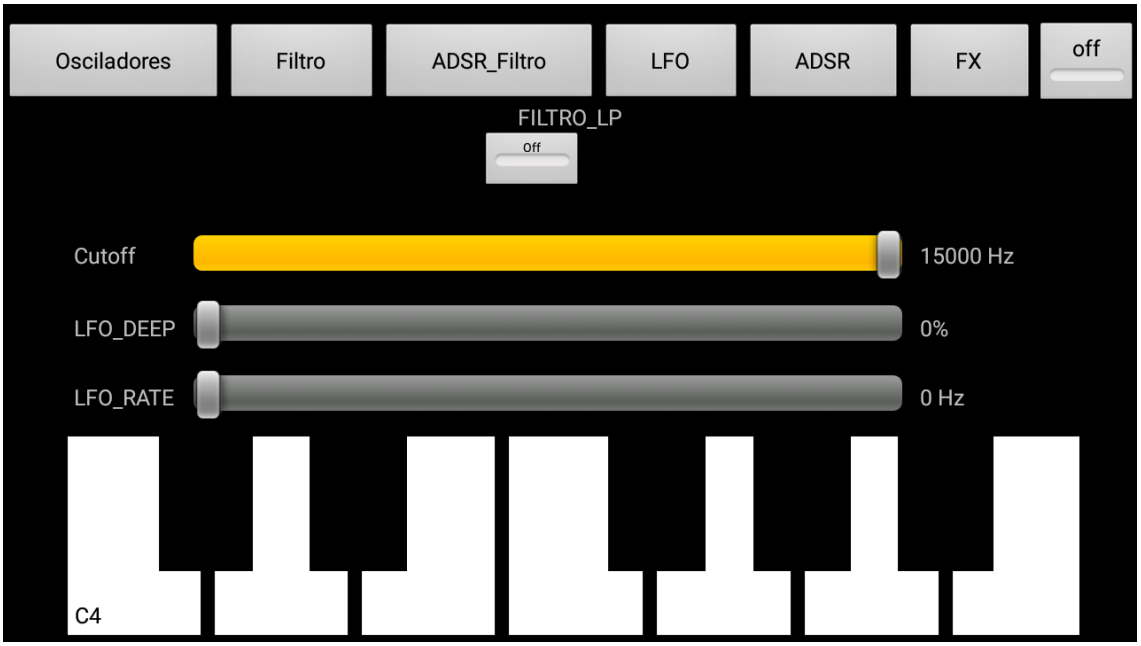


Figura 35. Aplicación síntesis sonora sección: filtro.

Los controles de esta sección se utilizan para controlar la frecuencia de corte del filtro y su modulación con el *LFO.*

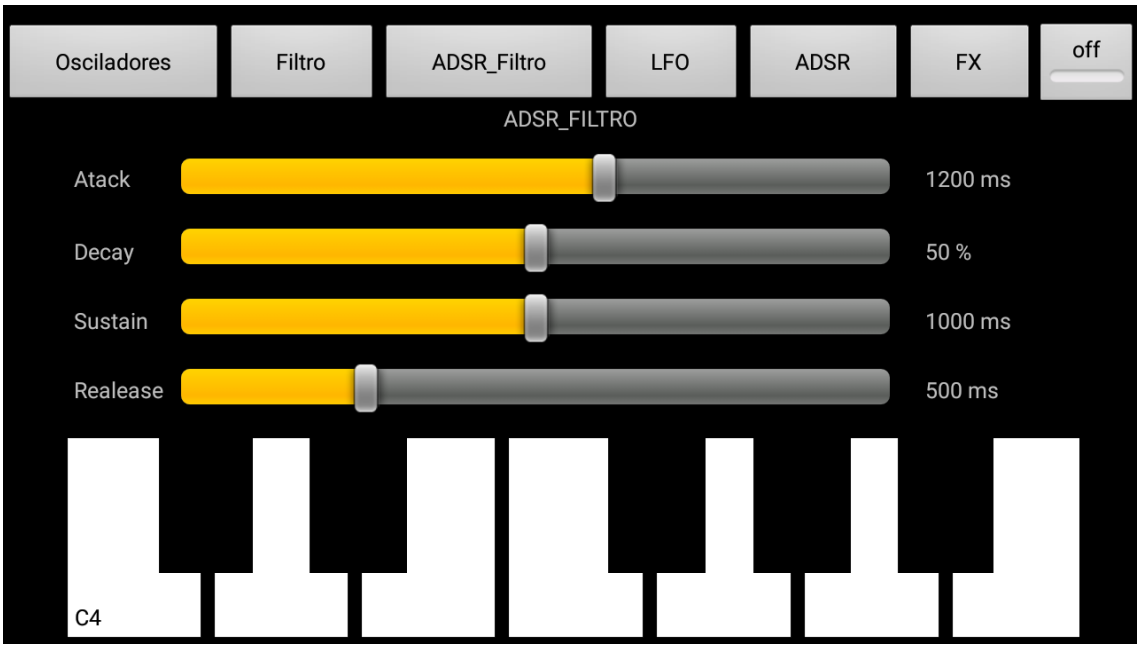


Figura 36. Aplicación síntesis sonora sección: *ADSR* filtro.

Los controles de esta sección modifican la frecuencia de corte del filtro siguiendo la envolvente programada.

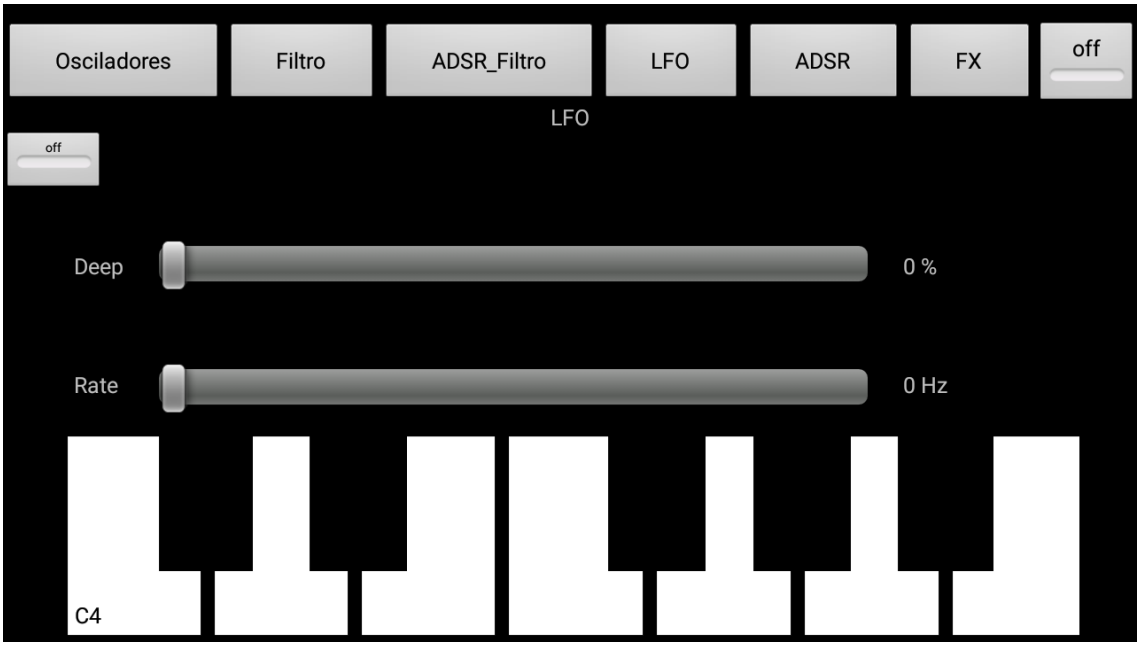


Figura 37. Aplicación síntesis sonora sección: *LFO.*

Esta sección tiene los controles que permiten modular la amplitud de la señal total del sintetizador para eso e tiene un control de profundidad y el rango de oscilación que va de 0 Hz a 20 Hz, este *LFO* se puede activar o desactivar.

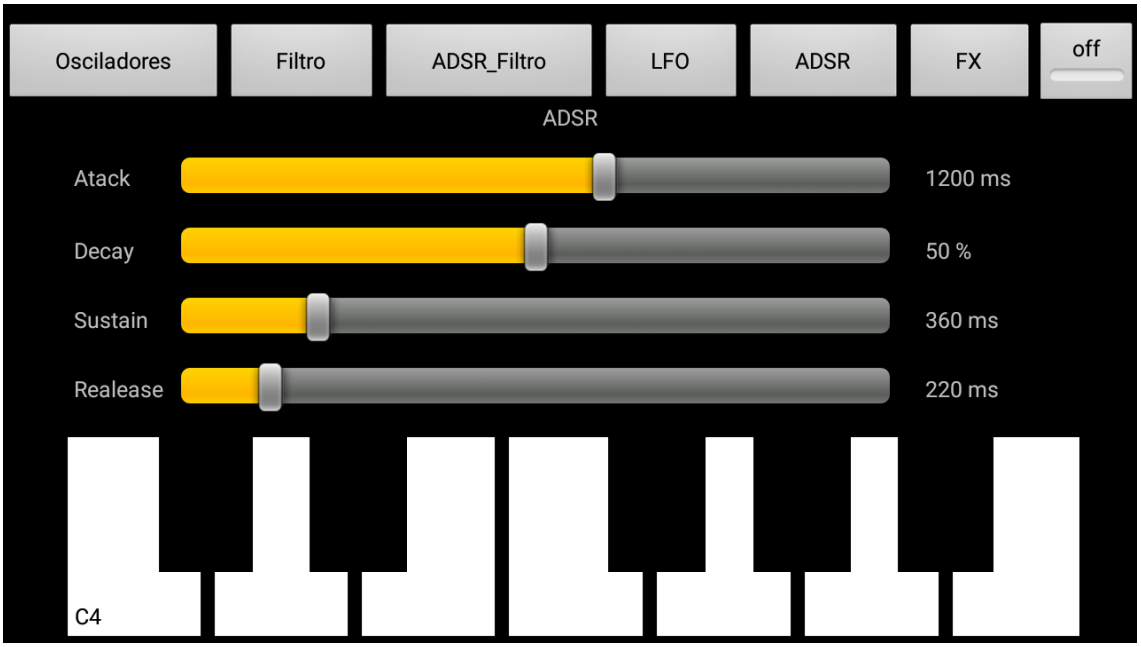


Figura 38. Aplicación síntesis sonora sección: *ADSR.*

Los controles que se pueden observar controlan la envolvente de la señal proveniente del sintetizador.

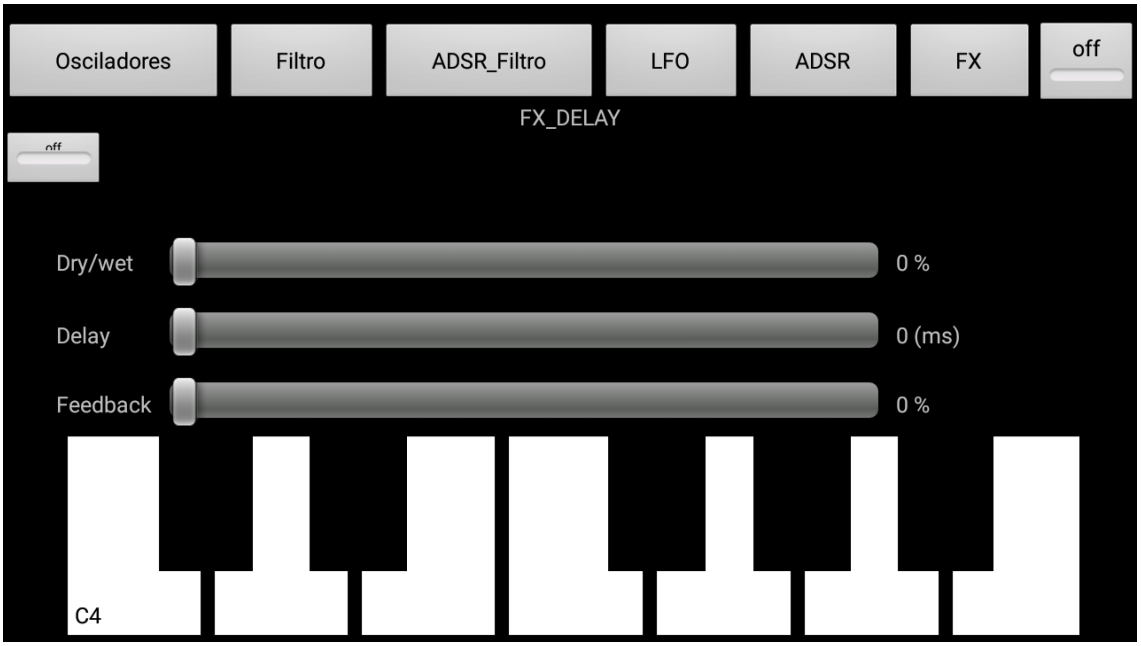


Figura 39. Aplicación síntesis sonora sección: *FX.*

En esta captura de pantalla se observan los controles de un efecto de *delay* los cuales controlan el nivel de señal del efecto, su retardo en milisegundos y la retroalimentación representada en porcentaje.

7.2. Comparación de resultados en dispositivos

Para probar la aplicación y comprobar la compatibilidad en dispositivos se utilizó una herramienta en línea gratuita llamada *Monkop* la cual se encarga de comprobar posibles errores dentro de la compilación y también prueba la aplicación en diferentes dispositivos.

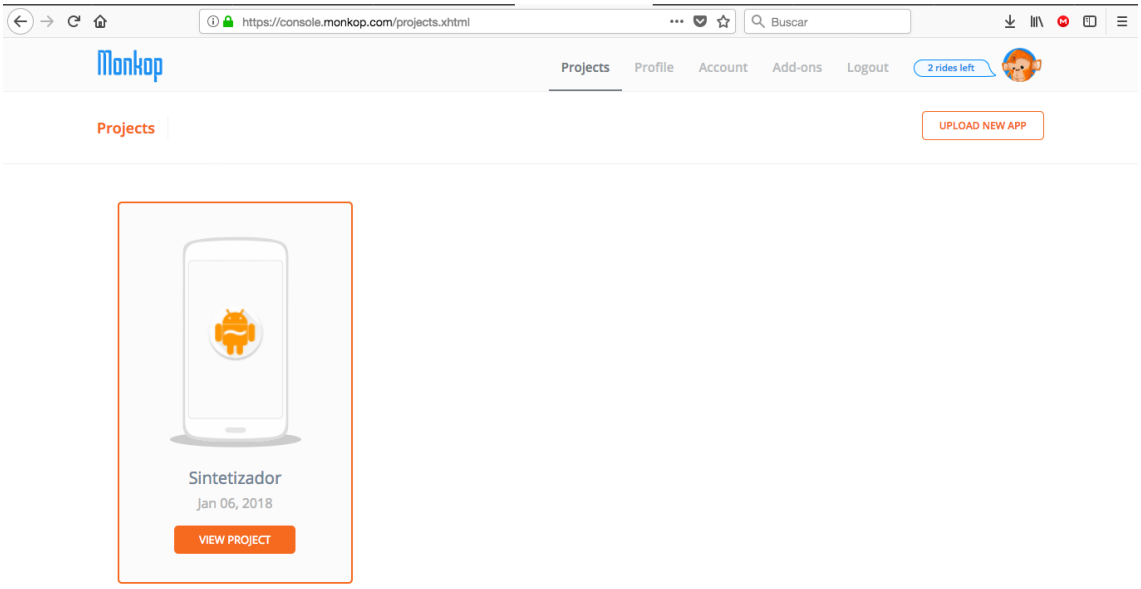
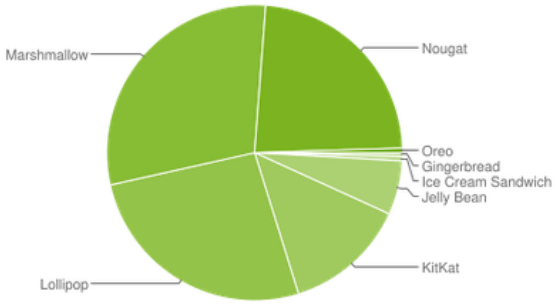


Figura 40. Carga de la aplicación hacia la herramienta *Monkop.*

Para la elección de los dispositivos la herramienta utiliza una base de datos oficial recopilada por *Google* y desplegada en la página de desarrolladores del sistema operativo *Android,* esto con el fin de priorizar perfiles de dispositivos (Developer.android, s.f.).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%



Datos recopilados durante un período de 7 días hasta 11/12/2017.
No se muestran versiones con una distribución inferior al 0,1%.

Figura 41. Datos de distribución de las diferentes versiones del sistema operativo *Android.* Recuperada de *Developer.android.* (s.f.).

Los datos recopilados por la herramienta de *Monkop* se muestran en la Tabla 4.

Tabla 4.

Datos recopilados por la herramienta Monkop.

Dispositivo	SO	Tamaño /Densidad	Tiempo de ejecución	Tiempo de arranque	Uso de batería	Ejecutable	Mínimo uso de CPU	Máximo uso de CPU	Uso de memoria
Samsung Galaxy Note 3 (SM-N900V)	4.4 (API: 19)	normal/xhdpi	10m05s.573	01s.030	Media	Si	4%	10%	18%
Google Nexus 9	6.0 (API: 23)	xlarge/xhdpi	10m08s.229	05s.728	Media	Si	10%	23%	5%
LG G2 (LS980)	5.0.x (API: 21)	normal/xxhdpi	10m09s.787	00s.729	Media	Si	12%	24%	6%
Google Nexus 10	5.1 (API: 22)	xlarge/xhdpi	10m05s.063	01s.016	Media	Si	12%	25%	4%
Moto G (XT1031)	5.1 (API: 22)	normal/xhdpi	10m03s.861	01s.705	Media	Si	11%	18%	9%
Moto X 2014 (XT1096)	5.1 (API: 22)	normal/xxhdpi	10m03s.928	00s.802	Media	Si	6%	12%	9%
Droid Turbo 2 (XT1585)	6.0 (API: 23)	normal/xxxhdpi	10m06s.267	01s.551	Media	Si	5%	11%	11%
Moto G4 Plus (XT1644)	7.0 (API: 24)	normal/xxhdpi	10m05s.161	00s.851	Media	Si	4%	9%	5%

Los datos de la Tabla 4 nos dan una idea del funcionamiento de la aplicación en diferentes dispositivos con estos datos podemos realizar cálculos de promedio y tener un resumen general.

Los cálculos mencionados se muestran en la Tabla 5.

Tabla 5.

Cálculos resultantes a partir de los datos reflejados por Monkop.

Tiempo total de prueba	Promedio de arranque	Promedio uso mínimo CPU	Promedio uso máximo CPU	Promedio uso memoria
01h20m47s. 869	1676,5 ms	8%	17%	8%

Con los datos de la Tabla 5 podemos observar que el uso de CPU y de memoria de la aplicación es bajo por lo que no se requiere de muchos recursos de un dispositivo, de esta manera se la puede catalogar como eficiente. Por otro lado, hay una observación en cuanto al tiempo de arranque de la aplicación en el dispositivo Google Nexus 9 mostrado en la Tabla 4, el cual es de 5,728 segundos, no es conveniente ya que el tiempo de arranque recomendado no debe exceder los 5 segundos (Dimensional Research, 2015); esto puede deberse a una falla de renderización del dispositivo.

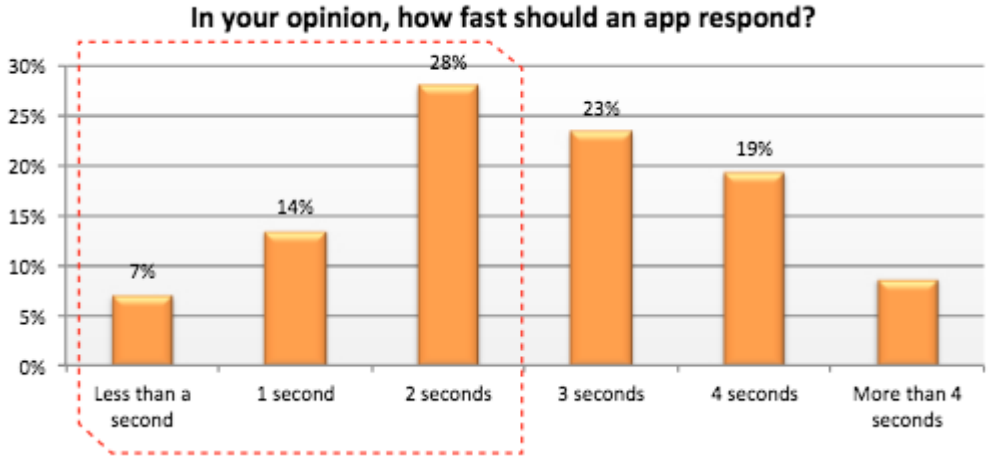


Figura 42. Resultados de la pregunta sobre tiempo de respuesta de una aplicación en la encuesta *Failing to meet mobile app user expectations,* (Dimesional Research).

La herramienta *Monkop* dice que el uso de la batería es medio, esto se debe al uso constante del altavoz en los dispositivos para que esta aplicación funcione,

las optimizaciones de las vistas pueden mejorar el uso de la batería. (Report.monkop, s.f.).

La prueba de la aplicación en dispositivos normales y grandes con diferentes densidades de pantalla no presentó ningún tipo de error.

Monkop tampoco reportó ningún error de ejecución por lo que se puede decir que la aplicación es estable.

8. CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones

Diseñar un sintetizador con *Pure Data,* permite ahorrar tiempo y llevar una organización visual y mejor entendible por ser un lenguaje orientado a objetos, también posibilita la creación de programas audiovisuales complejos con tamaños de archivos muy pequeños, esto ayuda a poder incluirlo de manera más fácil dentro de un programa más grande.

Android Studio proporciona las herramientas necesarias para poder crear una interfaz gráfica interactiva de manera sencilla y rápida, pero su mayor potencial es el de contar con librerías desarrolladas por terceros para la comunicación, esto facilitó la unión con *Pure Data.* La capacidad de poder emular cualquier dispositivo móvil con *Android Studio,* posibilitó probar la aplicación en diferentes ambientes y así medir su compatibilidad.

El desarrollo de la aplicación de síntesis sonora para sistema operativo *Android,* deja como base importante, una guía completa de cómo realizar la comunicación entre el lenguaje oficial para desarrollo de aplicaciones *Android Studio* y *Pure Data*; de esta manera se podría crear algoritmos de *Pure Data* más complejos y con mayores prestaciones.

La interfaz gráfica es muy interactiva por el simple hecho de estar diseñada exclusivamente para un dispositivo móvil, pero esto se podría mejorar desarrollando vistas más personalizadas.

La metodología usada para desarrollar esta aplicación hizo que el tiempo sea invertido eficientemente, la depuración fue muy fácil y permitía avanzar con las diferentes secciones en un periodo de tiempo muy favorable.

La herramienta *Monkop* ayudó con la prueba de la aplicación en varios dispositivos en un tiempo muy óptimo, lo cual permitió definir que la aplicación es estable y que puede ser usada en varios dispositivos sin presentar errores de gran magnitud, y de manera eficiente.

8.2. Recomendaciones

Para mejorar la aplicación se podría contratar los servicios de un diseñador gráfico el cual podría crear archivos de imagen personalizados para las vistas de la aplicación, de esta manera se optimizaría la distribución de los elementos dentro de la pantalla del dispositivo, mejoraría la renderización de las vistas lo cual se traduce en menor uso de la batería y su aspecto estaría listo para la distribución en las tiendas de aplicaciones.

Optimizar el algoritmo tratando de hacer más eficiente el proyecto podría hacer que la aplicación pese menos y ocupe menos recursos de los dispositivos móviles.

Con esta base de comunicación entre los dos lenguajes de programación se podría diseñar un sintetizador más complejo, como por ejemplo uno que posibilite el uso de tablas de onda, o a su vez que permita cargar respuestas de impulso para tener efectos de sonido más complejos.

REFERENCIAS

4rsoluciones. (s.f.). ¿Qué es un kit de desarrollo de software (SDK)? Recuperado el 17 de Diciembre de 2017 de http://www.4rsoluciones.com/blog/que-es-un-kit-de-desarrollo-de-software-sdk-2/

Amaya, Y. (2013). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Revista de Tecnología, 12(2), 111-124.

Aubert y Recoules. (2017). PPP-LooperSynth. Recuperado el 13 de Diciembre de 2017 de http://b2renger.github.io/PdDroidPublisher/ppp-samples.

Balderrabano, S., Gallo, A., y Mesa, P. (2013). Aspectos compositivos en la música para videojuegos: la música procedural. In IX Jornadas Nacionales de Investigación en Arte en Argentina. La Plata.

Barberá, J. C. (2015). Aplicación Android para reproducción de audio en plataforma arduino. (Tesis doctoral). Universidad politécnica de Valencia.

Blanco, P., Camarero, J., Fumero, A., Werterski, A., y Rodríguez, P. (2009). Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone. Dr. en Ing. Sist. Telemáticos. Universidad politécnica de Madrid.

Brinkmann, P., Kirn, P., Lawler, R., McCormick, C., Roth, M., y Steiner, H. C. (2011). Embedding pure data with libpd. In Proceedings of the Pure Data Convention (Vol. 291).

Cortes, J. A. C., Knott, A. M., y Osorio, J. A. C. (2012). Aproximación a la síntesis de la música a través del análisis de Fourier. Scientia et Technica, 2(52), 129-135.

Delía, L., Galdamez, N., Thomas, P. J., y Pesado, P. M. (2013). Un análisis experimental de tipo de aplicaciones para dispositivos móviles. In XVIII Congreso Argentino de Ciencias de la Computación. Universidad Nacional de la Plata.

Developer.android. (s.f.). Conoce Android Studio. Recuperado el 18 de Diciembre de 2017 de https://developer.android.com/studio/intro/index.html?hl=es-419.

Developer.android. (s.f.). Estadísticas de instalación de Google Play. Recuperado el 5 de Enero de 2018 de https://developer.android.com/about/dashboards/index.html

Developer.android. (s.f.). Datos de distribución de las diferentes versiones del sistema operativo Android. Recuperado de https://developer.android.com/about/dashboards/index.html

Dimensional Research. (2015). Failing to meet mobile app user expectation. A mobile app user survey. Recuperado de https://techbeacon.com/sites/default/files/gated_asset/mobile-app-user-survey-failing-meet-user-expectations.pdf

Dimensional Research. (2015). Resultados de la pregunta sobre tiempo de respuesta de una aplicación en la encuesta Failing to meet mobile app user expectations. Recuperado de https://techbeacon.com/sites/default/files/gated_asset/mobile-app-user-survey-failing-meet-user-expectations.pdf

Farnell, A. (2010). Designing Sound. Londres: Mit University Press Group Ltd.

Gasca, M. C. M., Ariza, L. L. C., y Delgado, B. M. (2014). Metodología para el desarrollo de aplicaciones móviles. Revista Tecnura, 18(40), 20-35.

Gómez. E. (2009). Introducción a la síntesis de sonidos. Departamento de sonología. Escuela superior de música de Cataluña. Recuperado de http://www.dtic.upf.edu/~egomez/teaching/sintesi/SPS1/Tema1-IntroduccionSintesi.pdf

Hoyos, J. G., Marín, D. G., y Uribe, D. S. (2010). Plataforma embebida para la síntesis y procesamiento de audio. XV simposio de tratamiento de señales, imágenes y visión artificial. Escuela Colombiana de Ingeniería Julio Garavito, Bogotá.

Luckyframe. (s.f.). Wave trip. Recuperado el 16 de Diciembre de 2017 de http://luckyframe.co.uk/press/sheet.php?p=wave_trip.

Ma, L., Gu, L., y Wang, J. (2014). Research and development of mobile application for android platform. International Journal of Multimedia and Ubiquitous Engineering, 9(4), 187-198.

Martínez, R., y Murillo, N. (2016). Síntesis de sonidos utilizando la DSP Blackfin BF53 (Tesis Doctoral). Universidad de El Salvador.

Menchén, M. R., Barbancho, A. M., Herrero Platero, M. I., Tardón, L. J., y Barbancho, I. (2016). Herramienta para mejorar la afinación vocal en Android. ATIC Research Group, ETSI Telecomunicacion. Recuperado de: https://riuma.uma.es/xmlui/bitstream/handle/10630/11994/Barbancho_AfinaVocal.pdf?sequence=1

Oracle. (s.f.). Java SE. Recuperado el 18 de Diciembre del 2017 de https://www.oracle.com/es/java/technologies/index.html.

Puckette, M. (1996). Pure Data: another integrated computer music environment. Proceedings of the second intercollege computer music concerts, 37-41.

Puredata. (s.f). Pure Data. Recuperado el 16 de Diciembre de 2017 de https://puredata.info

Reyes, A. G. (2009) Funcionamiento de un sintetizador de sonido basado en el análisis de fourier sobre señales periódicas. Universidad industrial de Santander. Recuperado el 12 de Diciembre de 2017 de: http://s3.amazonaws.com/academia.edu.documents/32947887/funcionamiento_de_un_sintetizador_de_sonido_basado_en_el_analisis_de_fourier_sobre_senales_periodicas.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1497864938&Signature=WXoKPQd1vR3ZxKOsDAydwpKwZRA%3D&response-content-disposition=inline%3B%20filename%3DUNIVERSIDAD_INDUSTRIAL_DE_SANTANDER.pdf

Report.monkop, (s.f.) Reports & results. Power usage. Recuperado el 5 de Enero del 2018 de https://report.monkop.com/3v5-90eef221-b839-4f04-9b23-70513dcfc4b5/report/report/index.html#battery

Romero. M. (2011). Técnicas de síntesis y procesamiento de sonido y su aplicación en tiempo real. Revista de investigación multimedia. 3(3). (69-83).

Vanegas, C. A. (2013). Desarrollo de aplicaciones sobre Android. Revista Vínculos, 9(2), 129-145.

ANEXOS

Anexo A. Programación fragmento menú

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_weight="1"
        android:orientation="horizontal">
// Botones del menu
        <Button
            android:id="@+id/osciladores_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Osciladores" />
        <Button
            android:id="@+id/filtro_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Filtro" />
        <Button
            android:id="@+id/adsr_f_btn"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="ADSR_Filtro" />
        <Button
            android:id="@+id/lfo_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="LFO" />
        <Button
            android:id="@+id/adsr_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="ADSR" />
        <Button
            android:id="@+id/fx_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="FX" />
// Botón de encendido apagado
        <ToggleButton
```

```
        android:id="@+id/sint_on"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:checked="true"
        android:text="ToggleButton"
        android:textOff="off"
        android:textOn="on" />
</LinearLayout>
```

Anexo B. Programación controles sintetizador

El siguiente bloque de código XML muestra la programación para esta tarea.

```
// Fragmento que contiene el menú
<fragment
        android:id="@+id/menuprincipal"
        android:name="com.deviantdev.pdsampleproject.Menu"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout="@layout/fragment_menu"
        tools:layout_constraintLeft_creator="1"
        tools:layout_constraintRight_creator="1">
    </fragment>
// Fragmento que contiene las secciones del sintetizador
    <FrameLayout
        android:id="@+id/contenedor"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@+id/teclasnegras"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/menuprincipal"
        tools:layout_constraintRight_creator="1">
// Fragmento que contiene el teclado MIDI
    </FrameLayout>
    <fragment
        android:id="@+id/teclasnegras"
        android:name="com.deviantdev.pdsampleproject.Teclado"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:layout="@layout/fragment_teclado"
        tools:layout_constraintRight_creator="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent">
    </fragment>
```

Anexo C. Programación fragmento osciladores

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="55dp"
    android:orientation="horizontal">
  </LinearLayout>
  <TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="OSCILADORES"
    android:textAlignment="center" />
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <Space
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="0.07" />
    <ToggleButton
      android:id="@+id/noiseboton"
      android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_weight="0.03"
    android:textOff="Noise Off"
    android:textOn="Noise On" />
<Space
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0.07" />
<ToggleButton
    android:id="@+id/senoboton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0.03"
    android:checked="true"
    android:text="ToggleButton"
    android:textOff="Senoidal Off"
    android:textOn="Senoidal On" />
<Space
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0.07" />
<ToggleButton
    android:id="@+id/sierraboton"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.03"
        android:text="ToggleButton"
        android:textOff="Sierra Off"
        android:textOn="Sierra On" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
</LinearLayout>

<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="133dp"
    android:orientation="horizontal">
    <Space
        android:layout_width="20dp"
        android:layout_height="wrap_content" />
    <LinearLayout
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical">
<Space
    android:layout_width="match_parent"
    android:layout_height="7dp" />
<TextView
    android:id="@+id/textView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Volumen_Noise" />
<Space
    android:layout_width="match_parent"
    android:layout_height="24dp" />
<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Volumen_Senoidal" />
<Space
    android:layout_width="match_parent"
    android:layout_height="22dp" />
<TextView
```

```
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Volumen_Sierra" />
</LinearLayout>
<LinearLayout
    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <SeekBar
        android:id="@+id/vnoise_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:progress="0" />
    <Space
        android:layout_width="0dp"
        android:layout_height="10dp" />
    <SeekBar
        android:id="@+id/vsin_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Space
        android:layout_width="0dp"
```

```
        android:layout_height="10dp" />

    <SeekBar
        android:id="@+id/vsier_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
<Space
    android:layout_width="10dp"
    android:layout_height="wrap_content"
     />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/noisel_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
            android:text="0 %" />
        <Space
            android:layout_width="match_parent"
            android:layout_height="24dp" />
        <TextView
            android:id="@+id/senol_tx"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="100 %" />
        <Space
            android:layout_width="match_parent"
            android:layout_height="22dp" />
        <TextView
            android:id="@+id/Sierral_tx"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="0 %" />
    </LinearLayout>
</LinearLayout>
```

Anexo D. Programación fragmento filtro

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"></LinearLayout>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="FILTRO_LP"
        android:textAlignment="center" />
    <ToggleButton
        android:id="@+id/encendidofiltro_btn"
        android:layout_width="wrap_content"
        android:layout_height="34dp"
        android:layout_marginLeft="270dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        android:textSize="8sp" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
    android:orientation="horizontal">
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
</LinearLayout>
<Space
    android:layout_width="0dp"
    android:layout_height="20dp" />
<LinearLayout
    android:layout_width="match_parent"
```

```
android:layout_height="133dp"
android:orientation="horizontal">
<Space
    android:layout_width="40dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Cutoff" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="24dp" />
    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="LFO_DEEP" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
    <TextView
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="LFO_RATE" />
</LinearLayout>
<Space
    android:layout_width="10dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Space
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content" />
    <SeekBar
        android:id="@+id/cutoff_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="100" />
    <Space
        android:layout_width="0dp"
        android:layout_height="10dp" />
    <SeekBar
        android:id="@+id/flfo_d_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Space
        android:layout_width="0dp"
        android:layout_height="10dp" />
    <SeekBar
        android:id="@+id/flfo_r_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
<Space
```

```
    android:layout_width="10dp"
    android:layout_height="wrap_content"
    />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/cutoff_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="15000 Hz" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="24dp" />
    <TextView
        android:id="@+id/deepf_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="0%" />
```

```
        <Space
            android:layout_width="match_parent"
            android:layout_height="22dp" />
        <TextView
            android:id="@+id/ratef_tx"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="0 Hz" />
    </LinearLayout>
</LinearLayout>
```

Anexo E.

Programación fragmento *ADSR* filtro

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"></LinearLayout>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ADSR_FILTRO"
        android:textAlignment="center" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Space
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.07" />
        <Space
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
</LinearLayout>
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:orientation="horizontal">
    <Space
        android:layout_width="40dp"
        android:layout_height="wrap_content" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
android:orientation="vertical">
<Space
    android:layout_width="match_parent"
    android:layout_height="7dp" />
<TextView
    android:id="@+id/textView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Atack" />
<Space
    android:layout_width="match_parent"
    android:layout_height="24dp" />
<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Decay" />
<Space
    android:layout_width="match_parent"
    android:layout_height="22dp" />
<TextView
    android:id="@+id/textView5"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Sustain" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Realease" />
</LinearLayout>
<Space
    android:layout_width="10dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <SeekBar
        android:id="@+id/af_ata_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="2000"
```

```
    android:progress="1200" />
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<SeekBar
    android:id="@+id/af_dec_sb"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="2000"
    android:progress="1000" />
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<SeekBar
    android:id="@+id/af_sus_sb"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="2000"
    android:progress="1000" />
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<SeekBar
```

```
        android:id="@+id/af_real_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="2000"
        android:progress="500" />
</LinearLayout>
<Space
    android:layout_width="20dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/ataf_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="1200 ms" />
    <Space
        android:layout_width="match_parent"
```

```
        android:layout_height="24dp" />
    <TextView
        android:id="@+id/decaf_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="50 %" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
    <TextView
        android:id="@+id/susf_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="1000 ms" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
    <TextView
        android:id="@+id/realf_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="500 ms" />
</LinearLayout>
```

```
</LinearLayout>
```

Anexo F.

Programación fragmento *LFO*

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="55dp"
    android:orientation="horizontal"></LinearLayout>
  <TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="LFO"
    android:textAlignment="center" />
  <ToggleButton
    android:id="@+id/encendidolfo_btn"
    android:layout_width="wrap_content"
    android:layout_height="35dp"
    android:text="ToggleButton"
    android:textOff="off"
    android:textOn="on"
    android:textSize="8sp" />
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:orientation="horizontal">
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
</LinearLayout>
<Space
    android:layout_width="0dp"
    android:layout_height="25dp" />
<LinearLayout
    android:layout_width="match_parent"
```

```
android:layout_height="133dp"
android:orientation="horizontal">
<Space
    android:layout_width="40dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Deep" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="50dp" />
    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Rate" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
</LinearLayout>
<Space
    android:layout_width="20dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <SeekBar
        android:id="@+id/llfo_d_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:progress="0" />
    <Space
        android:layout_width="0dp"
        android:layout_height="40dp" />
    <SeekBar
        android:id="@+id/llfo_r_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Space
        android:layout_width="0dp"
        android:layout_height="10dp" />
</LinearLayout>
<Space
    android:layout_width="20dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
```

```
            android:id="@+id/deeplfo_tx"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="0 %" />
        <Space
            android:layout_width="match_parent"
            android:layout_height="50dp" />
        <TextView
            android:id="@+id/ratelfo_tx"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="0 Hz" />
        <Space
            android:layout_width="match_parent"
            android:layout_height="22dp" />
    </LinearLayout>
</LinearLayout>
```

Anexo G.

Programación fragmento *ADSR*

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"></LinearLayout>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ADSR"
        android:textAlignment="center" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Space
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.07" />
        <Space
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
</LinearLayout>
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:orientation="horizontal">
    <Space
        android:layout_width="40dp"
        android:layout_height="wrap_content" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
android:orientation="vertical">
<Space
    android:layout_width="match_parent"
    android:layout_height="7dp" />
<TextView
    android:id="@+id/textView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Atack" />
<Space
    android:layout_width="match_parent"
    android:layout_height="24dp" />
<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Decay" />
<Space
    android:layout_width="match_parent"
    android:layout_height="22dp" />
<TextView
    android:id="@+id/textView5"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Sustain" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Realease" />
</LinearLayout>
<Space
    android:layout_width="10dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <SeekBar
        android:id="@+id/ad_ata_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="2000"
```

```
    android:progress="1200" />
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<SeekBar
    android:id="@+id/ad_dec_sb"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="2000"
    android:progress="1000" />
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<SeekBar
    android:id="@+id/ad_sus_sb"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="2000"
    android:progress="360" />
<Space
    android:layout_width="0dp"
    android:layout_height="10dp" />
<SeekBar
```

```
        android:id="@+id/ad_real_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="2000"
        android:progress="220" />
</LinearLayout>
<Space
    android:layout_width="20dp"
    android:layout_height="wrap_content"
    />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/ata_tx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="1200 ms" />
    <Space
```

```
    android:layout_width="match_parent"
    android:layout_height="24dp" />
<TextView
    android:id="@+id/deca_tx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="50 %" />
<Space
    android:layout_width="match_parent"
    android:layout_height="22dp" />
<TextView
    android:id="@+id/sus_tx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="360 ms" />
<Space
    android:layout_width="match_parent"
    android:layout_height="22dp" />
<TextView
    android:id="@+id/real_tx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="220 ms" />
```

```
    </LinearLayout>
</LinearLayout>
```

Anexo H.

Programación fragmento *FX*

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"></LinearLayout>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="FX_DELAY"
        android:textAlignment="center" />
    <ToggleButton
        android:id="@+id/encendidodelay_btn"
        android:layout_width="wrap_content"
        android:layout_height="33dp"
        android:text="ToggleButton"
        android:textOff="off"
        android:textOn="on"
        android:textSize="8sp" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
    android:orientation="horizontal">
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.07" />
</LinearLayout>
<Space
    android:layout_width="0dp"
    android:layout_height="25dp" />
<LinearLayout
    android:layout_width="match_parent"
```

```
android:layout_height="133dp"
android:orientation="horizontal">
<Space
    android:layout_width="30dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Space
        android:layout_width="match_parent"
        android:layout_height="7dp" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dry/wet" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="24dp" />
    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Delay" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="22dp" />
    <TextView
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Feedback" />
</LinearLayout>
<Space
    android:layout_width="10dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <SeekBar
        android:id="@+id/dryw_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:progress="0" />
```

```
    <Space
        android:layout_width="0dp"
        android:layout_height="10dp" />
    <SeekBar
        android:id="@+id/dela_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Space
        android:layout_width="0dp"
        android:layout_height="10dp" />
    <SeekBar
        android:id="@+id/fed_sb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
<Space
    android:layout_width="10dp"
    android:layout_height="wrap_content" />
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Space
```

```
    android:layout_width="match_parent"
    android:layout_height="7dp" />
<TextView
    android:id="@+id/dw_tx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="0 %" />
<Space
    android:layout_width="match_parent"
    android:layout_height="24dp" />
<TextView
    android:id="@+id/delay_tx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="0 (ms)" />
<Space
    android:layout_width="match_parent"
    android:layout_height="22dp" />
<TextView
    android:id="@+id/feed_tx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="0 %" />
```

```
    </LinearLayout>
</LinearLayout>
```

Anexo I.

Programación fragmento teclado

```
// Deslizador horizontal que permitirá movernos entre las diferentes octavas del teclado
<HorizontalScrollView
        android:layout_width="575dp"
        android:layout_height="125dp"
        android:fillViewport="true"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="230dp"
        android:layout_marginLeft="0dp"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginRight="0dp"
        app:layout_constraintRight_toRightOf="parent">
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="horizontal">
            <android.support.constraint.ConstraintLayout
                android:layout_width="583dp"
                android:layout_height="match_parent"
                android:orientation="horizontal"
                tools:layout_editor_absoluteX="8dp"
                tools:layout_editor_absoluteY="8dp">
```

```
<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="0dp"
    android:layout_height="120dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0"
    tools:layout_constraintLeft_creator="1"
    tools:layout_constraintRight_creator="1">
// Botones que conforman una octava del teclado
    <Button
        android:id="@+id/c_btn"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/presonalizate" />
    <Button
        android:id="@+id/d_btn"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
```

```
    android:layout_weight="1"
    android:background="@drawable/presonalizate"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toRightOf="@+id/f_btn"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />
<Button
    android:id="@+id/e_btn"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="@drawable/presonalizate"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toRightOf="@+id/d_btn"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<Button
    android:id="@+id/f_btn"
```

```
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="@drawable/presonalizate"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toRightOf="@+id/g_btn"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />
<Button
    android:id="@+id/g_btn"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="@drawable/presonalizate"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toRightOf="@+id/a_btn"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />
<Button
```

```
    android:id="@+id/a_btn"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="@drawable/presonalizate"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toRightOf="@+id/b_btn"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />
<Button
    android:id="@+id/b_btn"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="@drawable/presonalizate"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toRightOf="@+id/c_btn"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />
```

```
</LinearLayout>
<TextView
    android:id="@+id/textView9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="101dp"
    android:text="C4"
    android:textColor="#000000"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<Button
    android:id="@+id/cs_btn"
    android:layout_width="53dp"
    android:layout_height="80dp"
    android:background="@drawable/teclanegra"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.105"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.111"
    tools:layout_constraintBottom_creator="1"
```

```
    tools:layout_constraintTop_creator="1" />
<Button
    android:id="@+id/fs_btn"
    android:layout_width="55dp"
    android:layout_height="80dp"
    android:background="@drawable/teclanegra"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.585"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.111" />
<Button
    android:id="@+id/gs_btn"
    android:layout_width="55dp"
    android:layout_height="80dp"
    android:layout_marginEnd="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginStart="8dp"
    android:background="@drawable/teclanegra"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.748"
```

```
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.111" />
<Button
    android:id="@+id/ds_btn"
    android:layout_width="55dp"
    android:layout_height="80dp"
    android:background="@drawable/teclanegra"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.266"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.111" />
<Button
    android:id="@+id/bb_btn"
    android:layout_width="54dp"
    android:layout_height="80dp"
    android:layout_marginEnd="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginStart="8dp"
```

```
        android:background="@drawable/teclanegra"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.905"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.111" />
</android.support.constraint.ConstraintLayout>
```

Anexo J.

Programación java interfaz “Comunica”

```
public interface Comunica {

//Funciones que permiten enviar datos del fragmento osciladores hacia la actividad principal.

    public void controlboton (int quecontrol, float valor);

    public void controlseek (int quecontrol2, float valor2);

//Función que permiten enviar datos del fragmento ADSR hacia la actividad principal.

    public void controladsr(int quecontrols, float valorp);

//Función que permiten enviar datos del fragmento ADSR filtro hacia la actividad principal.

    public void controladsrf(int quecontrolaf, float valorf);

//Función que permiten enviar datos del fragmento filtro hacia la actividad principal.

    public void controlfiltro(int quecontrolf, float valorf);

//Función que permiten enviar datos del fragmento LFO hacia la actividad principal.

    public void controllfo(int quecontrollfo, float valorlfo);

//Función que permiten enviar datos del fragmento FX hacia la actividad principal.

    public void controlfx(int quecontrolfx, float valorfx);

//Funciones que permiten enviar de los controles de bypass hacia la actividad principal.

    public void bfiltro(float valor);
```

```
    public void bplfo(float valor);

    public void bpfx(float valor);

    public void sinton(float valor);

}
```

Anexo K.

Programación *Java* fragmento osciladores

```
public class osciladores extends Fragment {

    private OnFragmentInteractionListener mListener;

    private float progreso;

    private float quevalor;

    private final int [] MISCONTROLES = {R.id.noiseboton, R.id.senoboton,
R.id.sierraboton};// Se crea un vector que contedrá todas las vistas de tipo
ToggleButton del fragmento osciladores

    private final int [] MISCONTROLESEEK = {R.id.vnoise_sb, R.id.vsin_sb,
R.id.vsier_sb};// Se crea un vector que contedrá todas las vistas de tipo
SeekBar del fragmento osciladores

    private final int [] MISCONTROLESTX= {R.id.noisel_tx, R.id.senol_tx,
R.id.Sierral_tx};// Se crea un vector que contedrá todas las vistas de tipo
TextView del fragmento osciladores

    SeekBar ini;

    public osciladores() {

        // Required empty public constructor

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        View osciladores = inflater.inflate(R.layout.fragment_osciladores,
container, false);
```

```
ini= (SeekBar) osciladores.findViewById(R.id.vsin_sb);
ini.setProgress(100);
ToggleButton botoncontrol;
SeekBar seekcontrol;
TextView controltexto;
for (int i =0; i<MISCONTROLES.length;i++){ // Este bucle pemite recorrer todas las vistas para saber si el usuario a usado un control.
botoncontrol = (ToggleButton) osciladores.findViewById(MISCONTROLES[i]);
final int quecontrol =i;
final ToggleButton finalBotoncontrol = botoncontrol;
botoncontrol.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
if (finalBotoncontrol.isChecked()) {
Activity estaactividad = getActivity();
quevalor = 1f;
((Comunica) estaactividad).controlboton(quecontrol, quevalor);// Se envía un 1 flotante si los botones de encnedio de los osciladores es activado
}else{
Activity estaactividad = getActivity();
quevalor = 0;
```

```
                    ((Comunica) estaactividad).controlboton(quecontrol, quevalor); //
Se envía un 0 flotante si los botones de encnedio de los osciladores han sido
desactivados.
                }
            }
        });
            }
    for (int j =0; j<MISCONTROLESEEK.length;j++){
        seekcontrol = (SeekBar)
osciladores.findViewById(MISCONTROLESEEK[j]);
        controltexto = (TextView)
osciladores.findViewById(MISCONTROLESTX[j]);
        final int quecontrolseek =j;
        seekcontrol.setMax(100);
        final TextView finalControltexto = controltexto;
        final int finalI = j;
        final SeekBar finalSeekcontrol = seekcontrol;
        seekcontrol.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progreso1,
boolean b) {
                Activity estaactividad = getActivity();
                progreso = progreso1;
```

```
                ((Comunica) estaactividad).controlseek(quecontrolseek, progreso);
// Se envía el progreso del SeekBar responsable de cambiar el nivel de señal
de cada oscilador
                if (finalI ==0) {
                    finalControltexto.setText(progreso1 + " %");
// Se envía el progreso del SeekBar a la etiqueta de texto que muestra el
progreso del nivel.
                }
                if (finalI ==1) {
                    finalControltexto.setText(progreso1 + " %");
                }
                if (finalI ==2) {
                    finalControltexto.setText(progreso1 + " %");
                }
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
    }
    return osciladores;
```

```
    }
    }
}
```

Anexo L.

Programación *Java* fragmento filtro

```
public class Filtro extends Fragment {

    private final int [] MISCONTROLESFILTRO = {R.id.cutoff_sb, R.id.flfo_d_sb,
R.id.flfo_r_sb};// Se crea un vector que contedrá todas las vistas de tipo
SeekBar del fragmento filtro

    private final int [] MISCONTROLESTX = {R.id.cutoff_tx, R.id.deepf_tx,
R.id.ratef_tx};// Se crea un vector que contedrá todas las vistas de tipo
TextView del fragmento filtro

    private float dato;

    public Filtro() {

        // Required empty public constructor

    }

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        View filtro = inflater.inflate(R.layout.fragment_filtro, container, false);

        SeekBar seekcontrolfiltro;

        TextView controltexto;

        for (int i =0; i<MISCONTROLESFILTRO.length;i++){
```

```
        seekcontrolfiltro = (SeekBar)
filtro.findViewById(MISCONTROLESFILTRO[i]);
        controltexto = (TextView) filtro.findViewById(MISCONTROLESTX[i]);
        final int quecontrolfiltro =i;
        seekcontrolfiltro.setMax(100);
        final TextView finalControltexto = controltexto;
        final int finalI = i;
        seekcontrolfiltro.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progresofil,
boolean b) {
                Activity estaactividad = getActivity();
                ((Comunica) estaactividad).controlfiltro(quecontrolfiltro, progresofil);
// Se envía el progreso del SeekBar responsable de cambiar la frecuencia de
corte del filtro
                if (finalI ==0) {
                    finalControltexto.setText(progresofil*149.5+50 + " Hz");
// Se envía el progreso del SeekBar hacia el TextView
                }
                if (finalI ==1) {
                    finalControltexto.setText(progresofil + " %");
                }
                if (finalI ==2) {
                    finalControltexto.setText(progresofil/5 + " Hz");
```

```
            }
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
    });
}
final ToggleButton encendedor;
encendedor = (ToggleButton) filtro.findViewById(R.id.encendidofiltro_btn);
encendedor.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton,
boolean b) {
        if (encendedor.isChecked()) {
            Activity estaactividad = getActivity();
            dato = 0;
            ((Comunica) estaactividad).bfiltro(dato); // Se envía un 0 si el botón
de encendido del filtro esta inactivo.
        } else{
            Activity estaactividad = getActivity();
```

```
                dato = 1f;
                ((Comunica) estaactividad).bfiltro(dato);
// Se envía un 1 si el botón de encendido del filtro esta activo.
            }
        }
    });
    return filtro;
}
public interface OnFragmentInteractionListener {
}
}
```

Anexo M.

Programación *Java* fragmento *ADSR* filtro

```
public class adsr_filtro extends Fragment {

    private final int [] MISCONTROLESADSRF = {R.id.af_ata_sb,
R.id.af_dec_sb, R.id.af_sus_sb, R.id.af_real_sb}; // Se crea un vector que
contedrá todas las vistas de tipo SeekBar del fragmento ADSR filtro

    private final int [] MISCONTROLESTX = {R.id.ataf_tx, R.id.decaf_tx,
R.id.susf_tx, R.id.realf_tx}; // Se crea un vector que contedrá todas las vistas de
tipo TEXTVIEW del fragmento ADSR filtro

    public adsr_filtro() {

        // Required empty public constructor

    }

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        View adsrfiltro = inflater.inflate(R.layout.fragment_adsr_filtro, container,
false);

        SeekBar seekcontroladsrf;

        TextView controltexto;

        for(int i=0; i<MISCONTROLESADSRF.length;i++){
```

```
        seekcontroladsrf = (SeekBar)
adsrfiltro.findViewById(MISCONTROLESADSRF[i]);

        controltexto = (TextView)
adsrfiltro.findViewById(MISCONTROLESTX[i]);

        final int quecontroladsrf =i;

        seekcontroladsrf.setMax(2000);

        final TextView finalControltexto = controltexto;

        final int finall = i;

        seekcontroladsrf.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

            @Override

            public void onProgressChanged(SeekBar seekBar, int progresoadsrf,
boolean b) {

                Activity estaactividad = getActivity();

                ((Comunica) estaactividad).controladsrf(quecontroladsrf,
progresoadsrf); // Se envía el progreso del control SeekBar responsable de
controlar los tiempos de las secciones de la envolvente del filtro

                if (finall ==0) {

                    finalControltexto.setText(progresoadsrf + " ms");
// Se envía el progreso del control SeekBar hacia el TextView para poder
visualizar los valores de progreso del control.

                }

                if (finall ==1) {

                    finalControltexto.setText(progresoadsrf/20 + " %");

                }
```

```
                if (finalI ==2) {
                    finalControltexto.setText(progresoadsrf + " ms");
                }
                if (finalI ==3) {
                    finalControltexto.setText(progresoadsrf + " ms");
                }
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
    }
    return  adsrfiltro;
}
public interface OnFragmentInteractionListener {
}
}
```

Anexo N.

Programación *Java* fragmento *LFO*

```
public class LFO extends Fragment {

    private final int [] MISCONTROLESLFO = {R.id.llfo_d_sb, R.id.llfo_r_sb};// Se crea un vector que contendrá todas las vistas de tipo SeekBar del fragmento FLO

    private final int [] MISCONTROLESTX = {R.id.deeplfo_tx, R.id.ratelfo_tx};

    private float estado; // Se crea un vector que contendrá todas las vistas de tipo TextView del fragmento FLO

    public LFO() {

        // Required empty public constructor

    }

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        View lfo = inflater.inflate(R.layout.fragment_lfo, container, false);

        SeekBar seekcontrollfo;

        TextView controltexto;

        for (int i=0;i<MISCONTROLESLFO.length;i++){
```

```
        seekcontrollfo = (SeekBar) lfo.findViewById(MISCONTROLESLFO[i]);
        controltexto = (TextView) lfo.findViewById(MISCONTROLESTX[i]);
        final int quecontrollfo =i;
        final int finalI = i;
        seekcontrollfo.setMax(100);
        final TextView finalControltexto = controltexto;
        seekcontrollfo.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progresolfo,
boolean b) {
                Activity estaactividad = getActivity();
                ((Comunica) estaactividad).controllfo(quecontrollfo, progresolfo);
//Se envía el progreso del control SeekBar responsable de controlar la
profundidad y el rango del LFO
                if (finalI==0) {
                    finalControltexto.setText(progresolfo + " %");
//Se envía el progreso del control SeekBar hacia el TextView mostando los
valores de cada contol
                }
                if (finalI==1) {
                    finalControltexto.setText(progresolfo/5 + " Hz");
                }
            }
```

```
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
    }
    final ToggleButton encendidolfo;
    encendidolfo = (ToggleButton) lfo.findViewById(R.id.encendidolfo_btn);
    encendidolfo.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton,
boolean b) {
            if (encendidolfo.isChecked()) {
                Activity estaactividad = getActivity();
                estado = 1f;
                ((Comunica) estaactividad).bplfo(estado); //Se envía un 1 si el
control de encendido esta activo
            } else{
                Activity estaactividad = getActivity();
                estado = 0;
                ((Comunica) estaactividad).bplfo(estado);
```

```
//Se envía un 0 si el control de encendido esta inactivo
                }
            }
        });
        return lfo;
    }
    public interface OnFragmentInteractionListener {
    }
}
```

Anexo O.

Programación *Java* fragmento *ADSR*

```
public class adsr extends Fragment {

    private OnFragmentInteractionListener mListener;

    private final int [] MISCONTROLESADSR = {R.id.ad_ata_sb, R.id.ad_dec_sb,
R.id.ad_sus_sb, R.id.ad_real_sb};// Se crea un vector que contedrá todas las
vistas de tipo SeekBar del fragmento ADSR

    private final int [] MISCONTROLESTX = {R.id.ata_tx, R.id.deca_tx,
R.id.sus_tx, R.id.real_tx}; // Se crea un vector que contedrá todas las vistas de
tipo TextView del fragmento ADSR

    public adsr() {

        // Required empty public constructor

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        View adsr = inflater.inflate(R.layout.fragment_adsr, container, false);

        SeekBar seekcontroladsr;

        TextView controltexto;

        for(int i=0; i<MISCONTROLESADSR.length;i++){

            seekcontroladsr = (SeekBar)
adsr.findViewById(MISCONTROLESADSR[i]);

            controltexto = (TextView) adsr.findViewById(MISCONTROLESTX[i]);

            final int quecontroladsr =i;
```

```
        seekcontroladsr.setMax(2000);
        final TextView finalControltexto = controltexto;
        final int finalI = i;
        seekcontroladsr.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progresoadsr,
boolean b) {
                Activity estaactividad = getActivity();
                ((Comunica) estaactividad).controladsr(quecontroladsr,
progresoadsr); //Se envía el progreso del control SeekBar responsable de
controlar los tiempos de las diferentes scciones de la envolvente principal del
sintetizador
                if (finalI ==0) {
//Se envía el progreso del control SeekBar hacia el TextView para que muestre
el valor del parámetro.
                    finalControltexto.setText(progresoadsr + " ms");
                }
                if (finalI ==1) {
                    finalControltexto.setText(progresoadsr/20 + " %");
                }
                if (finalI ==2) {
                    finalControltexto.setText(progresoadsr + " ms");
                }
                if (finalI ==3) {
```

```
                finalControltexto.setText(progresoadsr + " ms");
            }
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
    });
  }
  return adsr;
}
@Override
public void onAttach(Context context) {
  super.onAttach(context);
  if (context instanceof OnFragmentInteractionListener) {
    mListener = (OnFragmentInteractionListener) context;
  } else {
    throw new RuntimeException(context.toString()
        + " must implement OnFragmentInteractionListener");
  }
}
```

```
    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        void onFragmentInteraction(Uri uri);
    }
}
```

Anexo P.

Programación *Java* fragmento *FX*

```
public class FX extends Fragment {

   private final int [] MISCONTROLESFX = {R.id.dryw_sb, R.id.dela_sb,
R.id.fed_sb};// Se crea un vector que contedrá todas las vistas de tipo SeekBar
del fragmento FX

   private final int [] MISCONTROLESTX = {R.id.dw_tx, R.id.delay_tx,
R.id.feed_tx};// Se crea un vector que contedrá todas las vistas de tipo
TextView del fragmento FX

   private float estado;

   public FX() {

      // Required empty public constructor

   }

   @Override

   public void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

   }

   @Override

   public View onCreateView(LayoutInflater inflater, ViewGroup container,

                            Bundle savedInstanceState) {

      // Inflate the layout for this fragment

      View fx = inflater.inflate(R.layout.fragment_fx, container, false);

      SeekBar seekcontrolfx;

      TextView controltexto;

      for (int i=0;i<MISCONTROLESFX.length;i++){
```

```
        seekcontrolfx = (SeekBar) fx.findViewById(MISCONTROLESFX[i]);
        controltexto = (TextView) fx.findViewById(MISCONTROLESTX[i]);
        final int quecontrolfx =i;
        seekcontrolfx.setMax(100);
        final TextView finalControltexto = controltexto;
        final int finalI = i;
        seekcontrolfx.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progresofx,
boolean b) {
                Activity estaactividad = getActivity();
                ((Comunica) estaactividad).controlfx(quecontrolfx, progresofx); //Se
envía el progreso del control SeekBar responsable de controlar el dry/wet, el
delay y el feedback del efecto de delay
                if (finalI ==0) {
//Se envía el progreso del control SeekBar hacia el TextView para poder ver el
valor del parámetro
                    finalControltexto.setText(progresofx + " %");
                }
                if (finalI ==1) {
                    finalControltexto.setText(progresofx*20 + " ms");
                }
                if (finalI ==2) {
                    finalControltexto.setText(progresofx + " %");
```

```
            }
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
    });
}
final ToggleButton bpfx;
bpfx = (ToggleButton) fx.findViewById(R.id.encendidodelay_btn);
bpfx.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton,
boolean b) {
        if (bpfx.isChecked()) {
//Permite envía un 0 o un 1 para activar o desactivar el efecto de delay.
            Activity estaactividad = getActivity();
            estado = 1f;
            ((Comunica) estaactividad).bpfx(estado);
        } else{
            Activity estaactividad = getActivity();
```

```
                estado = 0;
                ((Comunica) estaactividad).bpfx(estado);
            }
        }
    });
    return fx;
}
public interface OnFragmentInteractionListener {
}
}
```

Anexo Q.

Programación *Java* actividad principal

```
public class MainActivity extends Activity implements View.OnClickListener,
osciladores.OnFragmentInteractionListener,
adsr.OnFragmentInteractionListener,
        Filtro.OnFragmentInteractionListener,
adsr_filtro.OnFragmentInteractionListener, FX.OnFragmentInteractionListener,
LFO.OnFragmentInteractionListener, Comunica, ComunicaTeclado {
        private static final String TAG = "Sintetizador";
    /**
     * Se crean las variables tipo vista para todos los controles disponibles.
     */
    //Botones menu
    Button botonosciladores;
    Button botonadsr;
    Button botonadsrfiltro;
    Button botonfiltro;
    Button botonfx;
    Button botonlfo;
        //Botones tipo Toggle osciladores y encedido/apagado
        ToggleButton btnToggleSound;
        ToggleButton btnnoise;
        //Secrean las variables de los fragmentos para diseño de interfaz gráfica
    osciladores fragosciladores;
```

```
    adsr fragadsr;
    adsr_filtro fragadsrfiltro;
    Filtro fragfiltro;
    FX fragfx;
    LFO fraglfo;
        /**
         * El PdService es proporcionado por la libreria pd-for-android.
         */
        private PdService pdService = null;


        /**
         * Se inicializa el servicio de pure data para la reproducción de audio y
para poder recibir comandos de control.
         */
        private final ServiceConnection pdConnection = new
ServiceConnection() {
                @Override
                public void onServiceConnected(ComponentName name, IBinder
service) {
                        pdService = ((PdService.PdBinder)service).getService();
                        initPd();
                        try {
                                int sampleRate =
AudioParameters.suggestSampleRate();
                                pdService.initAudio( sampleRate, 0, 2, 8 );
```

```
                pdService.startAudio();
            } catch (IOException e) {
                toast(e.toString());
            }
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            pdService.stopAudio();
        }
    };
    /**
     * Se inicializa la interfaz de audio de pure data y se carga el archivo del parche del sintetizador almacenado dentro del los recursos de la app.
     */
    @SuppressWarnings("ResultOfMethodCallIgnored")
    private void initPd() {
        File patchFile = null;
        try {
            PdBase.setReceiver(new PdUiDispatcher());
            PdBase.subscribe("android");
            File dir = getFilesDir();
```

```
                    IoUtils.extractZipResource(
getResources().openRawResource( R.raw.sintetizadorsustractivoaditivo ), dir,
true );
                    patchFile = new File( dir, "sintetizadorsustractivoaditivo.pd"
);
                    PdBase.openPatch( patchFile.getAbsolutePath() );
                } catch (IOException e) {

                    finish();
                } finally {
                    if (patchFile != null) {
                        patchFile.delete();
                    }
                }
        }
        //Inicializamos todos los parámetros al abrir la aplicación
        @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

                botonosciladores = (Button) findViewById(R.id.osciladores_btn);
                botonosciladores.setOnClickListener(this);
        botonadsr = (Button) findViewById(R.id.adsr_btn);
```

```
        botonadsr.setOnClickListener(this);
        botonadsrfiltro = (Button) findViewById(R.id.adsr_f_btn);
        botonadsrfiltro.setOnClickListener(this);
        botonfiltro = (Button) findViewById(R.id.filtro_btn);
        botonfiltro.setOnClickListener(this);
        botonfx = (Button) findViewById(R.id.fx_btn);
        botonfx.setOnClickListener(this);
        botonlfo = (Button) findViewById(R.id.lfo_btn);
        botonlfo.setOnClickListener(this);
                btnnoise = (ToggleButton) findViewById(R.id.senoboton);
        fragosciladores = new osciladores();
        fragadsr = new adsr();
        fragadsrfiltro = new adsr_filtro();
        fragfx = new FX();
        fraglfo = new LFO();
        fragfiltro = new Filtro();
                AudioParameters.init(this);
                bindService(new Intent(this, PdService.class), pdConnection,
BIND_AUTO_CREATE);
                initGui();
          }
    @Override
        protected void onDestroy() {
                super.onDestroy();
```

```
            try {
                unbindService(pdConnection);
            } catch (IllegalArgumentException e) {
                pdService = null;
            }
        }
        /**
         * Valores por defecto del parche de pure data en caso de que no se
hayan agregado directamente en el parche
         */
        public void initGui() {
            //Boton encendido apagado
            this.btnToggleSound = (ToggleButton) findViewById(
R.id.sint_on);
            this.btnToggleSound.setOnCheckedChangeListener( new
CompoundButton.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged( CompoundButton
buttonView, boolean isChecked ) {
                    if ( btnToggleSound.isChecked() ) {

                        PdBase.sendFloat( "oneoffini", 1f );
                    } else {
                        PdBase.sendFloat( "oneoffini", 0 );
                    }
```

```
                }
            } );
        }

        private void toast(final String text) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast toast =
Toast.makeText(getApplicationContext(), "", Toast.LENGTH_SHORT);
                    toast.setText(TAG + ": " + text);
                    toast.show();
                }
            });
        }

        //Acción botones de menu, tiene como función llamar a cada uno de los
fregmentos segun el boton que se haya presionado
        @Override
        public void onClick(View view) {
       FragmentManager fragmentManager = getFragmentManager();
       FragmentTransaction transaction = fragmentManager.beginTransaction();
switch (view.getId()){
   case R.id.osciladores_btn:
```

```
        transaction.replace(R.id.contenedor, fragosciladores);
        transaction.commit();
        break;
    case R.id.adsr_btn:
        transaction.replace(R.id.contenedor, fragadsr);
        transaction.commit();
        break;
    case R.id.adsr_f_btn:
        transaction.replace(R.id.contenedor, fragadsrfiltro);
        transaction.commit();
        break;
    case R.id.filtro_btn:
        transaction.replace(R.id.contenedor, fragfiltro);
        transaction.commit();
        break;
    case R.id.lfo_btn:
        transaction.replace(R.id.contenedor, fraglfo);
        transaction.commit();
        break;
    case R.id.fx_btn:
        transaction.replace(R.id.contenedor, fragfx);
        transaction.commit();
        break;
```

```
}
        }
    @Override
    public void onFragmentInteraction(Uri uri) {
    }
        //Envio de señal del teclado
        //Para cada tecla se envia al control pitch un numero MIDI desde el C4 hasta el C6 cumpliendo las tres octavas
        //También se envía el estado del botón lo cual inicia el flujo de señal
        @Override
        public void controltecla(int quetecla, float estado) {
                switch (quetecla){

                        case 0:
                                PdBase.sendFloat("pitch",60);//Ésta función envia el numero MIDI hacia el control dentro del parche llamado "pitch"
                                PdBase.sendFloat("stat",estado);//Ésta función envia el estado del boton (presionado/nopresionado) hacia en control dentro del parche llamado "stat"
                                break;
                        case 1:
                                PdBase.sendFloat("pitch",61);
                                PdBase.sendFloat("stat",estado);
                                break;
                        case 2:
```

```
            PdBase.sendFloat("pitch",62);
            PdBase.sendFloat("stat",estado);
            break;
        case 3:
            PdBase.sendFloat("pitch",63);
            PdBase.sendFloat("stat",estado);
            break;
        case 4:
            PdBase.sendFloat("pitch",64);
            PdBase.sendFloat("stat",estado);
            break;
        case 5:
            PdBase.sendFloat("pitch",65);
            PdBase.sendFloat("stat",estado);
            break;
        case 6:
            PdBase.sendFloat("pitch",66);
            PdBase.sendFloat("stat",estado);
            break;
        case 7:
            PdBase.sendFloat("pitch",67);
            PdBase.sendFloat("stat",estado);
            break;
```

```
case 8:
        PdBase.sendFloat("pitch",68);
        PdBase.sendFloat("stat",estado);
        break;
case 9:
        PdBase.sendFloat("pitch",69);
        PdBase.sendFloat("stat",estado);
        break;
case 10:
        PdBase.sendFloat("pitch",70);
        PdBase.sendFloat("stat",estado);

        break;
case 11:
        PdBase.sendFloat("pitch",71);
        PdBase.sendFloat("stat",estado);
        break;
case 12:
        PdBase.sendFloat("pitch",72);
        PdBase.sendFloat("stat",estado);
        break;
case 13:
        PdBase.sendFloat("pitch",73);
```

```
            PdBase.sendFloat("stat",estado);
            break;
        case 14:
            PdBase.sendFloat("pitch",74);
            PdBase.sendFloat("stat",estado);
            break;
        case 15:
            PdBase.sendFloat("pitch",75);
            PdBase.sendFloat("stat",estado);
            break;
        case 16:
            PdBase.sendFloat("pitch",76);
            PdBase.sendFloat("stat",estado);
            break;
        case 17:
            PdBase.sendFloat("pitch",77);
            PdBase.sendFloat("stat",estado);
            break;
        case 18:
            PdBase.sendFloat("pitch",78);
            PdBase.sendFloat("stat",estado);
            break;
        case 19:
```

```
            PdBase.sendFloat("pitch",79);
            PdBase.sendFloat("stat",estado);

            break;
        case 20:
            PdBase.sendFloat("pitch",80);
            PdBase.sendFloat("stat",estado);

            break;
        case 21:
            PdBase.sendFloat("pitch",81);
            PdBase.sendFloat("stat",estado);
            break;
        case 22:
            PdBase.sendFloat("pitch",82);
            PdBase.sendFloat("stat",estado);
            break;
        case 23:
            PdBase.sendFloat("pitch",83);
            PdBase.sendFloat("stat",estado);
            break;
        case 24:
            PdBase.sendFloat("pitch",84);
```

```
            PdBase.sendFloat("stat",estado);
            break;
        case 25:
            PdBase.sendFloat("pitch",85);
            PdBase.sendFloat("stat",estado);
            break;
        case 26:
            PdBase.sendFloat("pitch",86);
            PdBase.sendFloat("stat",estado);
            break;
        case 27:
            PdBase.sendFloat("pitch",87);
            PdBase.sendFloat("stat",estado);
            break;
        case 28:
            PdBase.sendFloat("pitch",88);
            PdBase.sendFloat("stat",estado);

            break;
        case 29:
            PdBase.sendFloat("pitch",89);
            PdBase.sendFloat("stat",estado);
            break;
```

```
case 30:
        PdBase.sendFloat("pitch",90);
        PdBase.sendFloat("stat",estado);
        break;
case 31:
        PdBase.sendFloat("pitch",91);
        PdBase.sendFloat("stat",estado);
        break;
case 32:
        PdBase.sendFloat("pitch",92);
        PdBase.sendFloat("stat",estado);
        break;
case 33:
        PdBase.sendFloat("pitch",93);
        PdBase.sendFloat("stat",estado);
        break;
case 34:
        PdBase.sendFloat("pitch",94);
        PdBase.sendFloat("stat",estado);
        break;
case 35:
        PdBase.sendFloat("pitch",95);
        PdBase.sendFloat("stat",estado);
```

```
                    break;
                }
            }
            //Envio de señal sección osciladores
            @Override
            public void controlboton(int quecontrol, float valor) {
                switch (quecontrol){
                    case 0:
                        PdBase.sendFloat("noiseini", valor);//Esta función
envia un uno o un cero al encido/apagado de los osciladores
                        if (valor==1) {
                            Toast.makeText(this, "Noise encencido",
Toast.LENGTH_SHORT).show();
                        }else{
                            Toast.makeText(this, "Noise apagado",
Toast.LENGTH_SHORT).show();
                        }
                        break;
                    case 1:
                        PdBase.sendFloat("sinini", valor);
                        if (valor==1) {
                            Toast.makeText(this, "Senoidal encencido",
Toast.LENGTH_SHORT).show();
                        }else{
```

```
                                Toast.makeText(this, "Senoidal apagado",
Toast.LENGTH_SHORT).show();
                            }
                            break;
                        case 2:
                            PdBase.sendFloat("sierraini", valor);
                            if (valor==1) {
                                Toast.makeText(this, "Sierra encencido",
Toast.LENGTH_SHORT).show();
                            }else{
                                Toast.makeText(this, "Sierra apagado",
Toast.LENGTH_SHORT).show();
                            }
                            break;
                    }
                }
                //Envio señal control volumen osciladores
                //Envia señal hacia los niveles de los osciladores según el control que el
usuario haya usado
                @Override
                public void controlseek(int quecontrol2, float valor2) {
                    switch (quecontrol2){
                        case 0:
                            PdBase.sendFloat("volunoise", valor2/100f);//Esta
función envia valores flotantes hacia el control de volumen del oscilador ruido
```

```
                    break;
                case 1:
                    PdBase.sendFloat("volusin", valor2/100f);
                    break;
                case 2:
                    PdBase.sendFloat("volusier", valor2/100f);
                    break;
            }
        }
        //Envio de señal adsr
        //Envia señal de los controles de la envolvente del filtro
        @Override
        public void controladsr(int quecontrols, float valorp) {
            switch (quecontrols){
                case 0:
                    PdBase.sendFloat("atainim", valorp); //Envia valores
flotantes hacia el control de ataque de la envolvente
                    break;
                case 1:
                    PdBase.sendFloat("decainim", valorp/2000f);
                    break;
                case 2:
                    PdBase.sendFloat("susinim", valorp);
```

```
                break;
            case 3:
                PdBase.sendFloat("realinim", valorp);
                break;
        }
    }
    //Envio de señal adsr filtro
    @Override
    public void controladsrf(int quecontrolaf, float valorf) {
        switch (quecontrolaf){
            case 0:
                PdBase.sendFloat("atainimf", valorf);
                break;
            case 1:
                PdBase.sendFloat("decainimf", valorf/2000f);
                break;
            case 2:
                PdBase.sendFloat("susinimf", valorf);
                break;
            case 3:
                PdBase.sendFloat("realinimf", valorf);
                break;
        }
```

```
}
//Envio de señal seccioón filtro
@Override
public void controlfiltro(int quecontrolf, float valorf) {
        switch (quecontrolf){
                case 0:
                        PdBase.sendFloat("freqinif", valorf/100f);
                        break;
                case 1:
                        PdBase.sendFloat("deeplfof", valorf/100f);
                        break;
                case 2:
                        PdBase.sendFloat("ratelfof", valorf/100f);
                        break;
        }
}
//Envio de señal sección lfo
@Override
public void controllfo(int quecontrollfo, float valorlfo) {
        switch (quecontrollfo){
                case 0:
                        PdBase.sendFloat("deeplfo", valorlfo/100f);
                        break;
```

```
                case 1:
                        PdBase.sendFloat("ratelfo", valorlfo/100f);
                        break;
        }
}
//Envio de señal sección fx
@Override
public void controlfx(int quecontrolfx, float valorfx) {
        switch (quecontrolfx){
                case 0:
                        PdBase.sendFloat("dwc", valorfx/100f);
                        break;
                case 1:
                        PdBase.sendFloat("delayc", valorfx*20f);
                        break;
                case 2:
                        PdBase.sendFloat("feedc", valorfx/0.8f);
                        break;
        }
}
//Esta linea de código controla el bypass de cada sección
@Override
public void bfiltro(float valor) {
```

```
            PdBase.sendFloat("byf", valor);
        }
        @Override
        public void bplfo(float valor) {
            PdBase.sendFloat("bylfo", valor);
        }
        @Override
        public void bpfx(float valor) {
            PdBase.sendFloat("byd", valor);
        }
        @Override
        public void sinton(float valor) {
            PdBase.sendFloat( "oneoffini", valor);
        }
}
```