FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

“DESARROLLO DE PROTOTIPO DE UNA APLICACIÓN PARA DISEÑO DE CAJAS ACÚSTICAS EN SISTEMA OPERATIVO ANDROID”

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

Profesor guía

Ing. Héctor Merino Navarro

Autor

Ulvio Alexander Calva Romero

2016

DECLARACIÓN DEL PROFESOR GUÍA

Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, 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.

_____

Hector Merino Navarro

MSc Postproducción digital especialidad de audio

CI:1756785562

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.

_____

Ulvio Alexander Calva Romero

CI: 070368859-8

AGRADECIMIENTOS

De manera especial quiero agradecer al ingeniero de sistemas David Cañar, que me apoyó en la investigación del proyecto. A mis padres, por tener la paciencia y la sabiduría con sus palabras en los momentos oportunos.

El amor de mi esposa, mi estímulo a perseverar.

El amor a mis hijos, mi inspiración.

El amor de Dios, mi bendición.

DEDICATORIA

Para dos personas que siempre estuvieron alentándome en cada momento de debilidad. Mi madre con su amor y paciencia que supo llegar a mi con sus palabras. Mi padre, mi ejemplo a seguir por su perseverancia y fortaleza.

RESUMEN

Para cumplir el objetivo planteado fue necesario estudiar los lenguajes de programación *Pure Data*, *Javascript* y *HTML*, logrando así conocer mejor sus prestaciones y aplicaciones en el problema planteado. Una vez conocidos los lenguajes de programación, se empezó por encontrar las relaciones matemáticas de las fórmulas para el cálculo de los parámetros *Thiele-Small* para el diseño de una caja acústica, con las funciones y los comandos del lenguaje de programación respectivo.

Primero se hizo un análisis de los requisitos y parámetros para el procedimiento de la aplicación móvil, para lo cual se utilizó la teoría planteada en clases de la asignatura cajas acústicas. Ingresando los datos al *Smartphone*, el programa es capaz de calcular los parámetros *Thiele-Small* y de encontrar el criterio de proporciones para una caja acústica.

Como herramienta anexa en la aplicación móvil, el usuario podrá generar frecuencias en el rango desde los 20 Hz hasta los 20000 Hz., además, puede generar ruido blanco.

ABSTRACT

To meet the stated objective was necessary to study the Pure Data programming languages, JavaScript and HTML, achieving better understand their features and applications on the problem posed. Once known programming languages, began to find the mathematical relationships of the formulas for calculating the Thiele-Small parameters for a speaker design, with functions and commands respective programming language.

First, an analysis of the requirements and process parameters for the mobile application, for which the theory put forth in classes subject loudspeakers used.

Entering data to the Smartphone, the program is able to calculate the Thiele-Small parameters and criteria to find the proportions for a speaker.

As a tool attached to the mobile application, the user can generate frequencies in the range from 20 Hz to 20,000 Hz., Also can generate white noise.

ÍNDICE

INTRODUCCIÓN

Los fenómenos físicos han sido siempre tema de investigación científica a lo largo de la historia de la humanidad, haciendo cada vez más extenso el mundo de la física en la actualidad. Uno de los fenómenos más interesantes se podría considerar al sonido por su complejidad de análisis y de estudio. Desde un tono puro a una onda sonora simple con una suma de armónicos, siendo diferente su comportamiento según el medio de propagación.

En el mundo informático actual se conocen diferentes tipos de software, diseñados para funciones específicas. Cada programa informático está capacitado para realizar de uno a varios procesos en forma contínua según cual sea el objetivo de su programador.

Los programas para computadoras se pueden encontrar desde en un teléfono móvil hasta en un automóvil.

La tecnología existente en los teléfonos móviles, conocidos como *Smartphones,* ha avanzado en gran medida en los últimos años, por esta razón, se ha propuesto implementar una aplicación móvil, la cual, realizará cálculos matemáticos para encontrar los parámetros *Thiele-Small* de una caja acústica.

OBJETIVO

Diseñar una aplicación para dispositivos móviles, bajo el sistema operativo *Android*, la cual realizará cálculos matemáticos para encontrar los parámetros *Thiele-Small* de una caja acústica.

CAPÍTULO I

1. MARCO TEÓRICO

1.1. Software

El software es toda la parte lógica de una computadora, toda la parte intangible. Su clasificación viene dada por sistemas operativos, lenguajes de programación y software de aplicación.

Por medio del software se puede controlar varios periféricos en la computadora conocidos como hardware.

1.1.1. Sistemas Operativos

Los sistemas operativos se encargan de controlar las acciones en curso de un ordenador y de dispositivos móviles, además tienen dos tareas importantes que cumplir, las cuales son:

Control de entrada y salida.- El sistema operativo controla los dispositivos a conectarse a un ordenador, por ejemplo los periféricos, el procesador, los programas y la memoria RAM.

Manejo de archivos.- Los sistemas operativos permiten realizar tareas de copiado, cortado y pegado de archivos, creación de carpetas y eliminar archivos.

1.1.2. Lenguajes de programación

Un lenguaje de programación es un conjunto de instrucciones dadas por un usuario para la interpretación de un ordenador o dispositivo que trabaje con un sistema operativo. Estas instrucciones se encargan de realizar una función específica, ejecución de un programa.

Existe el lenguaje de máquina, el cual, no es posible interpretar por seres humanos por ser este en código binario.

Por esta razón se desarrollaron los lenguajes de programación, para ser interpretados y utilizados por el usuario, sin necesidad de entender datos binarios. Estos lenguajes se encargan de transformase en lenguaje de máquina para que el procesador pueda ejecutarlo.

Generalmente existen dos grupos principales de lenguajes de programación que dependen exclusivamente al procesamiento de sus comandos:

- Lenguajes imperativos.
- Lenguajes funcionales.

1.1.2.1. Lenguaje de programación imperativo

Estos tipos de lenguajes se basan principalmente en trabajar en bloques condicionales, donde, se crean instrucciones dependientes de una condición dada, la cual, si es verdadera o falsa, continúa a otro bloque de instrucciones. Los primeros lenguajes de programación trabajaban con este concepto.

1.1.2.2. Lenguaje de programación funcional

Lenguaje funcional, trabaja con instrucciones dadas por el usuario, donde, su funcionamiento se basa en funciones pre establecidas, llamadas en su momento de operación respectiva, dando como resultado otra instrucción, donde, esta puede ser utilizada como entrada, para realizar otro procedimiento.

Este tipo de lenguaje de programación es también conocido como lenguaje procedimental, y es el más utilizado hoy en día por los programadores.

1.1.3. Software de aplicación

El software de aplicación es diseñado para ejecutar tareas específicas por el usuario directamente sin tener que entrar al lenguaje de programación.

Estos programas son utilizados más en los negocios financieros como bancos, empresas de comercio, software de diseño asistido, base de datos, telecomunicaciones, software educativo y desde algunos años atrás en dispositivos móviles, como smartphones y tablets.

1.2. Caja acústica

Es un recinto acústico hecho de un material capaz de soportar grandes vibraciones producidas por el altavoz. Las cajas acústicas son construidas con la finalidad de dar mejor calidad al sonido del altavoz.

1.2.1. Parámetros Thiele-Small

Entre los años 1961 y 1973, un ingeniero australiano y un científico americano estudiaron las curvas de impedancia versus frecuencia, y lograron desarrollar parámetros electro-mecánico-acústicos para conocer el comportamiento de un altavoz, sus nombres eran *Neville Thiele* y *Richard H. Small.*

Esta combinación de conocimientos se realizó por medio de un documento que dio la vuelta al mundo en el año de 1961, a cargo, del ingeniero *Neville Thiele*, donde establecía fórmulas que implicaban como resolver principalmente problemas de resonancia de un altavoz dentro de una caja acústica. Fue cuando el científico americano *Richard H. Small* se interesó en el tema, y se contactó con *Neville Thiele* para desarrollar la investigación, logrando así crear los parámetros Thiele-Small, que serían una base para la construcción de cajas acústicas hasta la actualidad.(Millán, 2011).

Los parámetros más conocidos y utilizados son:

- F_{S} .- Frecuencia de resonancia del altavoz al aire libre.
- Q_{TC}.- Coeficiente de sobretensión del sistema en la frecuencia de resonancia de la caja.
- Q_{TS}.- Coeficiente de sobretensión total del altavoz.
- F_{-3}.- Frecuencia de corte en -3dB.
- F_{C}.- Frecuencia de resonancia del recinto.
- V_{AB}.- Volumen de aire equivalente a la elasticidad acústica del aire de la caja.
- V_{B}.- Volumen de la caja.
- V_{AS}.- Volumen de aire equivalente a la elasticidad de la suspensión del altavoz.
- F_{B}.- Frecuencia de resonancia de la caja.

- S.- Coeficiente de sobretensión de la caja.
- $M_{AP.-}$ Masa acústica del respiradero.
- L.- Longitud del respiradero.
- $S_{V.-}$ Superficie del respiradero.
- $L_{V.-}$ Medida de corrección de longitud del respiradero.
- $L_{I.-}$ Longitud definitiva del respiradero.

1.2.2. Criterio de proporciones para recintos cerrados

Existen dos criterios utilizados en esta tesis, el primero es el *Acoustic Ratio*, que es la razón entre la intensidad directa de la fuente sonora y la intensidad del sonido reverberante de las paredes de la caja, en un punto específico. El segundo es el *Golden Ratio*, es una razón que nace desde la antigüedad y se puede encontrar en la naturaleza, tiene relación con el número áureo. (Basantes, 2009, p.34)

Tabla 1 *Criterios de proporciones*

	Alto	Ancho	Profundidad
Acoustic Ratio	1.2599	1	0.7937
Golden Ratio	1.618	1	0.618

1.3. Pure Data un Lenguaje de Programación

En los años 90, se creó el lenguaje de programación gráfico llamado, *Pure Data.* Su autor fue *Miller Puckette,* aunque en la actualidad su desarrollo se basa en la cooperación de programadores que se encargan de añadirle extensiones al lenguaje, ya que, este es de código abierto, obteniendo así una plataforma completa de flujo de datos. Este lenguaje fue diseñado para la creación de música por computadora y trabajos multimedia.

Pure Data está basado en el procesamiento de flujo de datos por una interfaz gráfica, la cual, simula un ambiente de control de audio. Se utilizan objetos como bloques, donde, se conectan las entradas y salidas de datos, dependiendo de la programación que se desea hacer.El conjunto de estos bloques de información, se los conoce como parches o *Patchs*, donde, estos realizan una determinada función programada por el desarrollador. La comunidad de usuarios de este lenguaje, comparten los parches realizados y son reutilizados por varias personas en el mundo, sin ningún costo por ser código libre. También, se podría añadir el desarrollo que se ha logrado, gracias a la comunidad de desarrolladores, establecer una comunicación entre dispositivos electrónicos por vía LAN, y a la vez, a diferentes partes del mundo, haciendo música en tiempo real entre usuarios.

13.1. **Tipos de Objetos**

1.3.1.1. **Objeto**.- La funcionalidad del objeto dependerá de la información que este dentro del mismo, existen tipos de objetos predefinidos por el lenguaje, el cual los reconocerá.

1.3.1.2. **Números**.- Como su nombre los dice, es un tipo de objeto numérico, donde, su trabajo dependerá del algoritmo que se desea realizar.

1.3.1.3. **Mensajes.-** Es utilizado para compartir la información de datos a los objetos.

1.3.1.4. **Símbolo**.- En esta casilla se almacena un símbolo hasta que recibe un llamado por medio de un bang u otro símbolo.

1.3.1.5. **Comentarios.-** Su función es de mantener al usuario informado de los procedimientos dentro del *patch*.

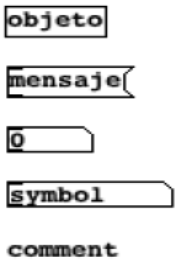

Figura 1. Objetos principales de Pure Data

1.3.1. Objetos Utilizados

1.3.1.1. **[pow].-** Objeto predefinido que realiza la potencia entre dos números.

1.3.1.2. **[r $0].-** Objeto que se encarga de recibir un valor numérico y asignarlo a una variable.

1.3.1.3. **[s $0].-** Objeto que se encarga de enviar un valor numérico asignado a una variable.

1.3.1.4. **[*].-** Objeto predefinido que realiza el producto entre dos números.

1.3.1.5. **[/].-** Objeto predefinido que realiza la división entre dos números.

1.3.1.6. **[-].-** Objeto predefinido que realiza la resta entre dos números.

1.3.1.7. **[noise~].-** Tipo de objeto predefinido que reproduce ruido blanco.

1.3.1.8. **[dac~].-** Tipo de objeto que realiza la conversión de una señal digital a analógica.

CAPÍTULO I I

2. DESARROLLO

2.1. Herramientas de programación en Pure Data

Antes de empezar a programar en el lenguaje mencionado, se debe configurar la tarjeta de sonido del ordenador después de su instalación, mediante una prueba realizada por el mismo programa, como se puede observar en la figura 2.

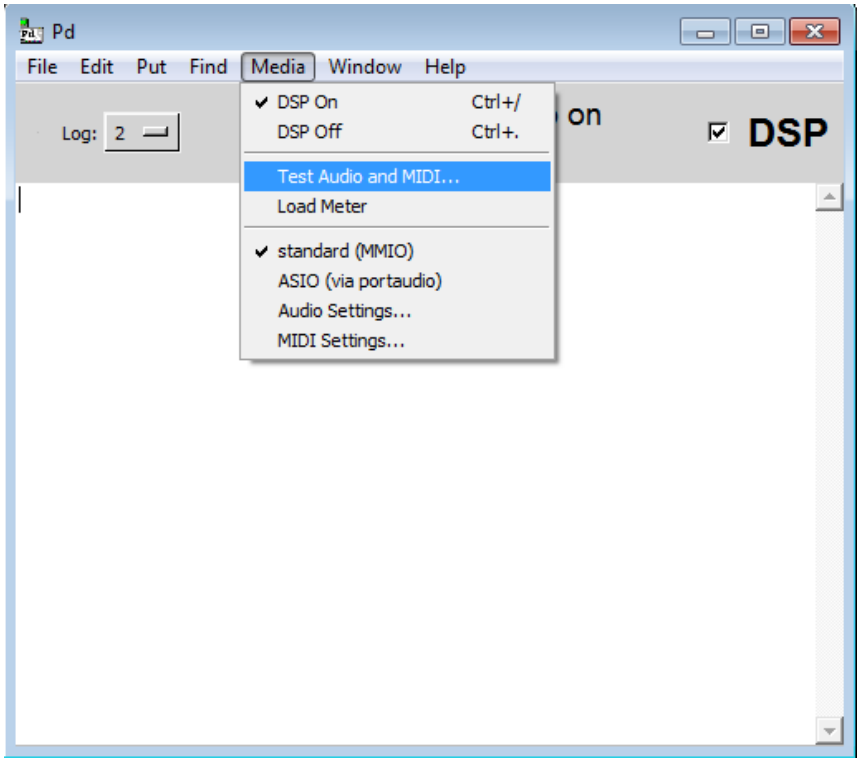

Figura 2. Test de audio en Pure Data

Al revisar las herramientas de programación que posee el lenguaje *Pure Data,* se observa que es de fácil apreciación y muy práctico, en el sentido de entender su flujo de datos.

Como primer ejemplo, se ingresan datos numéricos y se realiza una operación matemática básica.

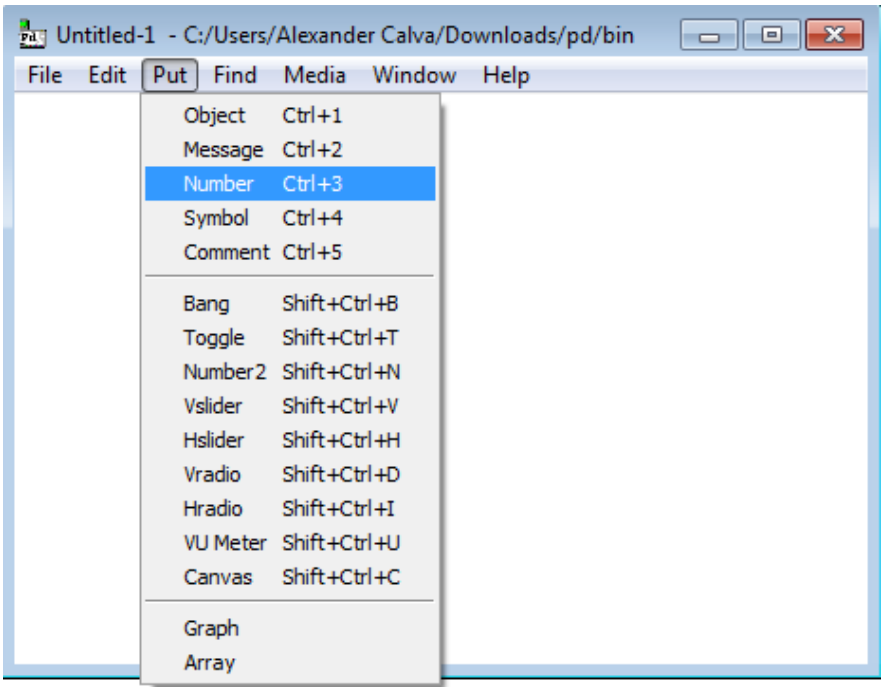

Figura 3. Objeto Number

Como se muestra en la figura 3, se crea un nuevo programa donde se va a realizar el primer ejemplo y se escoge el objeto *Number* tres veces, porque se necesita dos casilleros para ingresar los datos y otro para visualizar el resultado; para poder hacer la operación matemática, que en este caso va a ser una suma, se escoge el objeto *Object,* donde se escribirá el signo '+'

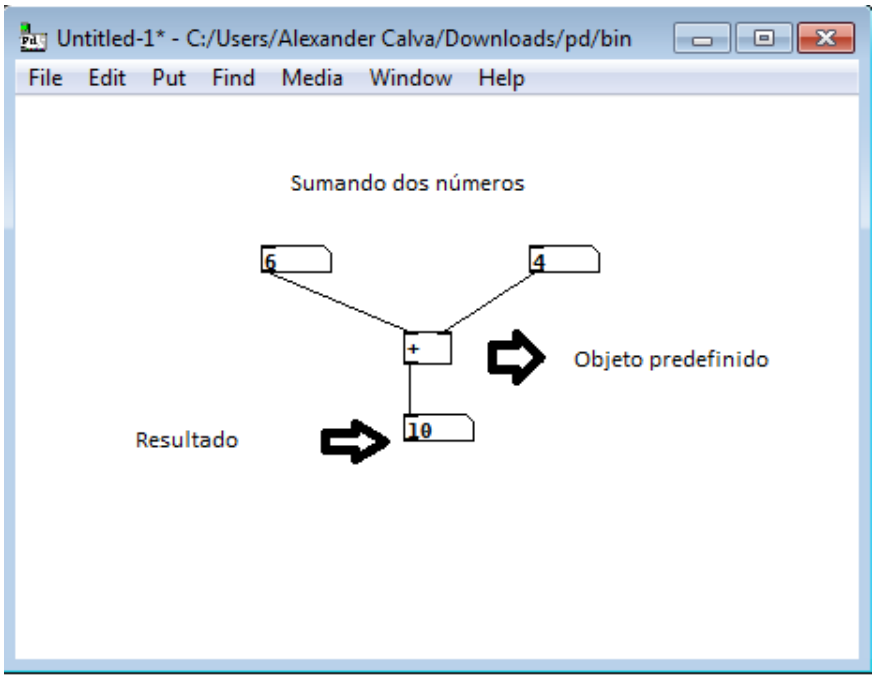

Figura 4. Suma de dos números

Como se logra ver en la figura 4, cada casillero u objeto posee una entrada y una salida de datos, a estos se los conoce como, *Inlets* = entradas y *Outlets* = salidas.

Pure Data maneja dos modos de programación, el primero es el modo de edición, donde, se realizará cualquier modificación al programa y el segundo es el modo de ejecución, que como su nombre mismo lo dice, se encarga de ejecutar el programa planteado. Para cambiar de un modo a otro, se utilizan las teclas Ctrl + E.

La unión que existe entre los casilleros de datos, se realiza con el puntero del *Mouse* del ordenador, emulando a una conexión física de flujo de información.

2.2. Alcances.- Realizando la investigación de las limitaciones del lenguaje *Pure Data*, se obtuvo el resultado, de que su interfaz gráfica no permitía hacer una aplicación móvil, por lo cual, se necesita una conexión a otro lenguaje de programación para dicha función.

Más adelante se explicará cómo se realizó este tipo de enlace de lenguajes.

2.3. Fórmulas de los parámetros *Thiele – Small*

Después de conocer las herramientas básicas de *Pure Data,* se ingresan las primeras fórmulas orientadas al proyecto planteado.

La primera ecuación es de la frecuencia de resonancia del recinto acústico. (García, s.f., p 3)

$$F_C = \frac{Q_{TC} * F_S}{Q_{TS}} \qquad \text{(Ecuación 1)}$$

La programación en el lenguaje Pure Data, sería:

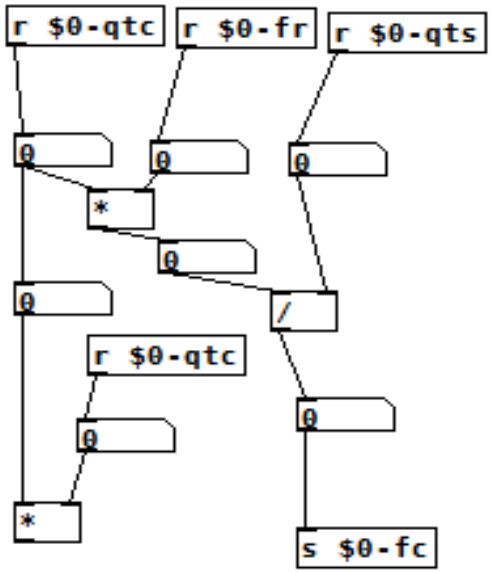

Figura 5. Cálculo de la frecuencia de resonancia

Donde se reciben los valores de [qtc], [fr] y [qts] enviados desde otro lenguaje de programación, en este caso se envían desde el Apache Cordova de la siguiente manera:

'Pd.send(Patch.patchId + '-qtc,[qtc]); ',

‘Pd.send(Patch.patchId + '-fr,[fr]); '.

‘Pd.send(Patch.patchId + '-qts,[qts]);’

En la expresión [r $0-qtc], se explicándoles que la letra ‘r’ significa *Receive* y ‘$0’ separa un espacio en memoria para almacenar a la variable llamada [qtc], de igual manera se utiliza el mismo procedimiento para las variables [fr] y [qts].

Luego de recibir los datos numéricos en cada casillero del objeto tipo *Object*, se transfieren a otro objeto, pero en este caso de tipo *Number* respectivamente.

Al analizar la fórmula se puede notar que existen dos operaciones matemáticas, una multiplicación y una división, por lo cual, se procede a insertar un objeto tipo *Object* para realizar el producto entre la variable [qtc] y [fr] que es la misma que [fs], añadiendo un [*] que significa multiplicación en este lenguaje de programación. El resultado se puede visualizar insertando otro objeto de tipo *Number.*

Se recuerda que las entradas de datos en cada casillero u objeto, se realizan por la parte superior del mismo y las salidas de datos serían por la parte inferior.

Ya teniendo el resultado de la multiplicación se realiza el cálculo de la división, utilizando un procedimiento similar visto anteriormente, en donde, se inserta un objeto de tipo *Object* añadiendo [/] que significa división en *Pure Data*, en consecuencia, se escoge el resultado de la multiplicación y se va a dividir para el valor almacenado en la variable [qts].

Para poder visualizar el resultado se añade nuevamente un objeto de tipo *Number,* conectando la salida del cálculo de la división a la entrada de este objeto.

Si se desea enviar este resultado a otro *Patch* dentro del mismo lenguaje o a otro lenguaje de programación, se utiliza la sintaxis [s $0-fc], donde, la letra ‘s’ significa *Send* y ‘$0’ separa un espacio de memoria del ordenador, con el nombre de [fc] para poder ser almacenada y enviada al destino correspondiente.

Una manera fácil de comprobar si la programación esta correcta, es enviar directamente desde el mismo proyecto o *Patch*, los datos a calcular. Esto se puede realizar de la siguiente forma:

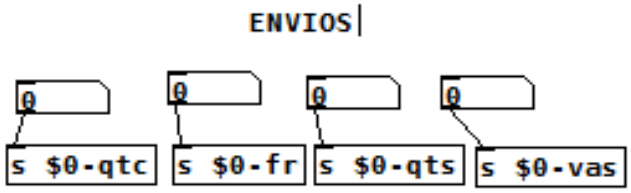

Figura 6. Envío de valores dentro de Pure Data

Donde se utilizan tres objetos de tipo *Number* para escribir los valores numéricos y tres objetos de tipo *Object* para enviar la información, teniendo en cuenta que los nombres de las variables sean los mismos.

La siguiente ecuación es F_{-3} que es la frecuencia de corte a – 3 dB.

$$F_3 = F_C * \sqrt{\frac{A+\sqrt{4+A^2}}{2}} \qquad \text{(Ecuación 2)}$$

$$A = \frac{1}{Q_{TC}{}^2} - 2 \qquad \text{(Ecuación 3)}$$

La codificación correspondiente para el ingreso de los datos, se realizó en *Javascript,* que es un lenguaje de programación orientado a un entorno *Web*.

En el siguiente bloque de programación, se crean variables para asignarles valores numéricos y se realizan cálculos matemáticos correspondientes a la ecuación 2. La sintaxis en Javascript es similar a la del lenguaje de programación C.

```
var A = (1 / (qtc) * (qtc)) - 2; // Se asigna el cálculo a la variable [A]
var B = (A * A) + 4;             // Se asigna el cálculo a la variable [B]
var C = Math.sqrt(B);//Se calcula la raíz cuadrada de [B] y se asigna a
var D = (A + C) / 2;  // Se suma las variables [A] y [C], se divide para 2
var E = Math.sqrt(D); // Se calcula la raíz cuadrada de [D]
var F3 = fc * E;  //Rresultado de la   fórmula F-3
```

Los valores de las variables [qtc] y [fc] son enviados desde el lenguaje *Pure Data*, como se explicó anteriormente.

Para calcular el volumen de la caja acústica se hace mediante las siguientes ecuaciones:

$$V_{AB} = \frac{V_{AS}}{\alpha} \qquad \text{(Ecuación 4)}$$

$$\alpha = \left(\frac{F_C}{F_S} \right)^2 - 1 \qquad \text{(Ecuación 5)}$$

La programación en *Pure Data* quedaría así:

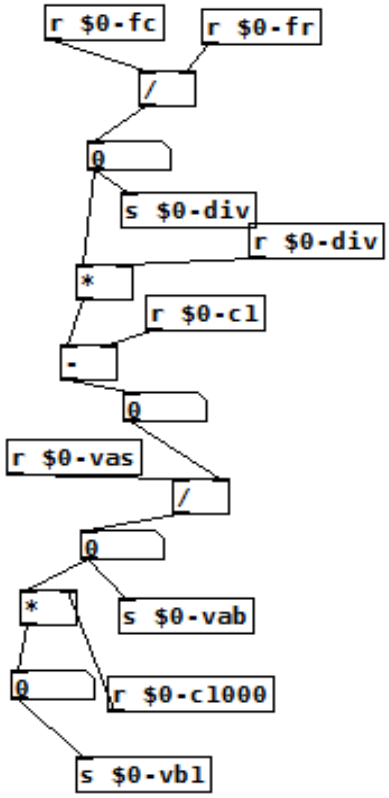

Figura 7. Cálculo del volumen de una caja acústica

En la figura 7 se observa como se reciben los datos de las variables [fc] y [fr] con los objetos [r $0-fc] y [r $0-fr] que son enviados desde el lenguaje de programación *Apache Cordova*. Luego se dividen los dos valores de las variables y el resultado se asigna a la variable [div], luego se envía ese valor para hacer el cálculo de la potencia al cuadrado de la misma división entre [fc] y [fr], siguiendo con el orden de la programación, se resta el resultado con la constante [c1] que fue almacenada en el *Patch* con el valor de 1, como se puede observar en la figura 8.

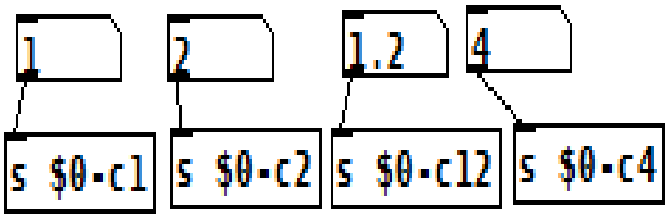

Figura 8. Ejemplo de asignación de valores a constantes en Pure Data

El resultado de esta operación servirá como divisor para la próxima división, donde, el dividendo será la variable [vas] que su valor es enviado desde el lenguaje *Apache Cordova*, obteniendo así el resultado final del volumen de la caja acústica, que es asignado a una variable llamada [vab] y enviada de regreso al *Apache Cordova.* Para calcular el volumen de la caja no amortiguada, se multiplica el resultado anterior por la constante [c1000], que tiene como asignación el valor de 1000 y el resultado es asignado a la variable [vb1] e igualmente es enviado al lenguaje de programación *Apache Cordova*.

Siguiendo con el procedimiento del ingreso de las fórmulas, a continuación, se va a explicar su programación para obtener los parámetros *Thiele-Small* de una caja acústica con *Bass Reflex.*

Para calcular el volumen:

$$V_B = S * V_{AS} * Q_{TS}^{\ 2} \qquad \text{(Ecuación 6)}$$

La programación quedaría así:

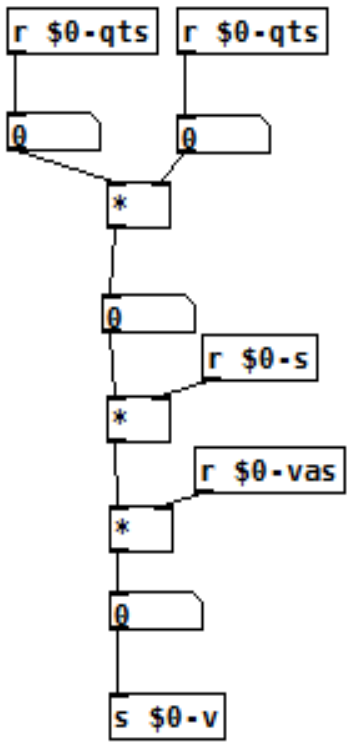

Figura 9. Cálculo del volumen de una caja acústica en *Pure Data*

En los casilleros de tipo *Object* se reciben los datos de las variables [qts], [vas] y [s], estos valores son enviados desde el *Apache Cordova.*

La variable [qts] es ingresada dos veces para realizar el cálculo de la potencia al cuadrado, luego, el resultado se multiplica por la variable [s] que es el coeficiente de sobretensión de la caja acústica y por la variable [vas], el resultado final se asigna a una variable llamada [v] donde es enviada al *Apache Cordova*. A continuación se va a calcular F_{-3}:

$$F_{-3} = \sqrt{\frac{V_{AS} * F_R^{\,2}}{V_B}} \qquad \text{(Ecuación 7)}$$

La programación quedaría así:

```
var v1 = document.getElementById("entrada5").value;        // Se asigna a la variable [v1] el valor de VB se calculó anteriormente.
var F = fr * fr;        // Se crea la variable [F] para asignar el cálculo de FR2.
var G = F * vas;        // Se crea la variable [G] para asignar el producto de [F] por VAS.
var H = G / v1;         // Se crea la variable [H] para asignar la división entre [G] y VB.
var F3i = Math.sqrt(H);     // Se crea la variable [ F3i] para asignar el resultado de la raíz cuadrada de [H], que sería el valor de F-3
caja.entrada6.value = F3i.toFixed(2);     //Se visualiza el resultado en el dispositivo móvil con dos decimales.
var fbass = F3i;        //Se almacena el valor de la respuesta en una variable llamada [fbass].
```

El lenguaje de programación utilizado para calcular F_{-3} es *Javascript.*

Cálculo de la frecuencia de resonancia de la caja.

$$F_B = \frac{F_{-3}}{\alpha^{0.13}} \qquad \text{(Ecuación 8)}$$

$$\alpha = \frac{V_{AS}}{V_B} \qquad \text{(Ecuación 9)}$$

La programación sería:

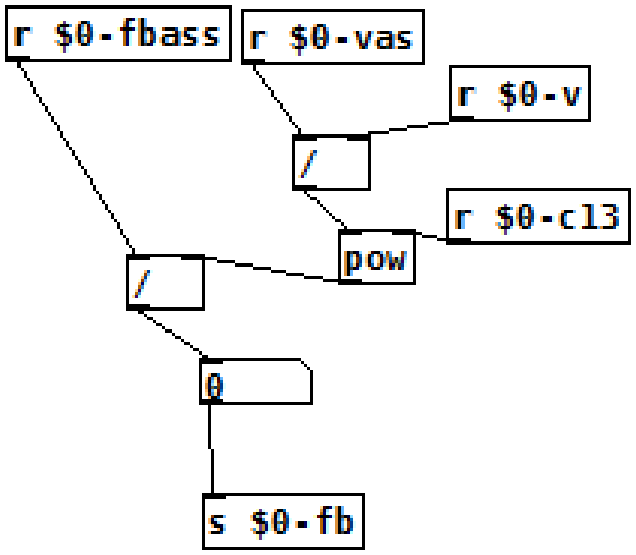

Figura 10. Cálculo de F_B en Pure Data

Se recibe el valor de F_{-3} que fue almacenado en la variable [fbass] y el valor de V_{AS} asignado a la variable [vas], se realiza la división y el resultado es elevado a la potencia con el valor de 0,13.

Para calcular la potencia en el *Pure Data* se utiliza el objeto [pow], luego se divide F_{-3} [fbass] para el resultado anterior y se obtiene la respuesta final del cálculo, que sería $F_{B.}$

Al final se asigna a una variable llamada [fb] el resultado final y se envía al *Apache Cordova.*

Cálculo de la superficie del respiradero.

$$S_V = \left(\frac{d}{2}\right)^2 * \pi \qquad \text{(Ecuación 10)}$$

La programación para encontrar S_V sería:

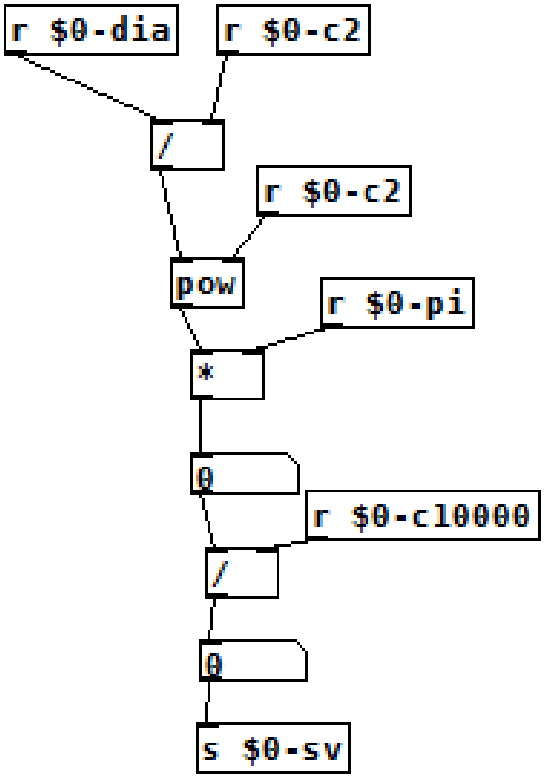

Figura 11. Cálculo de S_V en Pure Data

Se recibe el valor del diámetro que es ingresado por el usuario y asignado a la variable [dia], se realiza la división para la constante [c2] que tiene el valor de 2. Su resultado es elevado a la potencia al cuadrado, utilizando el objeto [pow], consecuentemente el resultado se multiplica por el valor de π que es recibido como constante [pi]. Para transformar el resultado de centímetros cuadrados a metros cuadrados, se divide para 10000 el cual su valor es asignado a la constante [c10000] de la misma manera como se asignaron las otras constantes. El resultado final es almacenado en la variable [sv] y es enviado al *Apache Cordova*.

Cálculo de la elasticidad de la caja.

$$C_{AB} = \frac{V_B}{140449}$$ (Ecuación 11)

La programación sería:

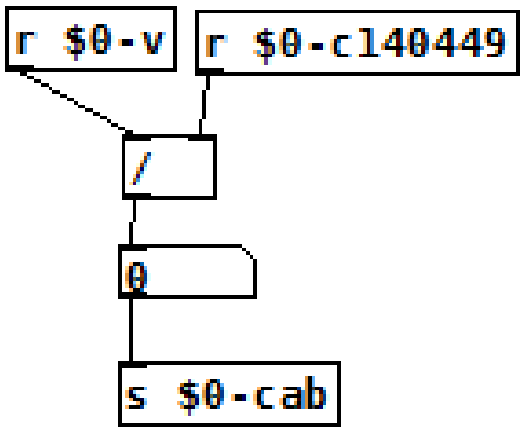

Figura 12. Cálculo de C_{AB} en Pure Data

Se recibe el valor de V_B con la variable [v] y el valor de la constante [c140449] que su número asignado es 140449, después se procede a realizar la división entre estos dos valores y su resultado es almacenado en una variable llamada [cab], que a la vez es enviada al *Apache Cordova.*

Cálculo de la masa acústica del respiradero.

$$M_{AP} = \frac{1}{4\pi^2 * F_B{}^2 * C_{AB}}$$ (Ecuación 12)

La programación sería:

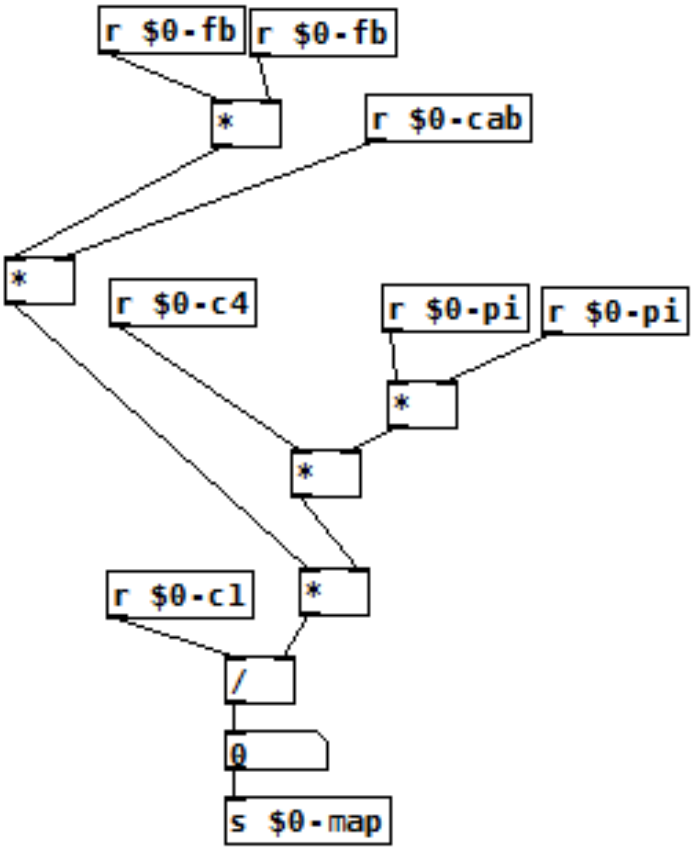

Figura 13. Cálculo de M_{AP} en Pure Data

Se recibe el valor de F_B por medio de la variable [fb] y se realiza la potencia al cuadrado, el resultado es multiplicado por el valor de C_{AB} que es la variable [cab], luego, es enviada desde el *Apache Cordova*.

La constante [c4] almacenada con el valor de 4 y la constante de **TT** asignada con [pi], son recibidas en el *Pure Data,* luego se calcula la potencia al cuadrado de [pi], donde, el resultado es multiplicado por la constante [c4], a continuación se realiza el producto entre el resultado final y la primera respuesta donde intervenían [fb] y [cab], consecuentemente, se divide la constante [c1] que tiene como valor asignado 1 para la última respuesta encontrada, su valor total es almacenado en la variable [map], para ser enviado al *Apache Cordova*.

Cálculo de la longitud del respiradero:

$$L = \frac{M_{AP} * S_V}{1.18}$$

(Ecuación 13)

La programación sería:

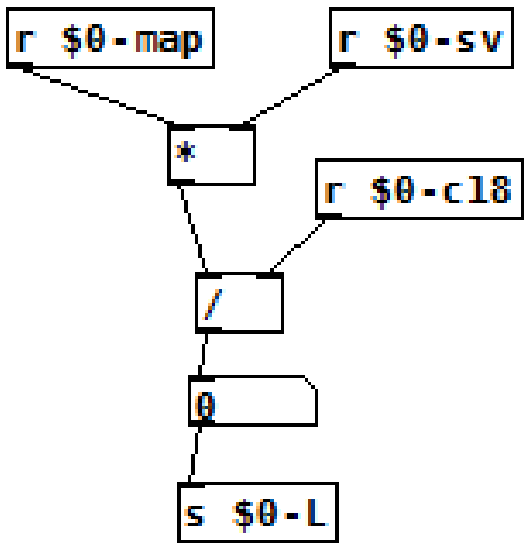

Figura 14. Cálculo de L en *Pure Data*

Se reciben los valores de M_{AP} que está asignado a la variable [map], S_V que está asignado a la variable [sv] y a la constante [c18] que tiene almacenada el valor de 1,18 que procede de la ecuación 13, luego, se multiplican las variables [map] por [sv] y su resultado se divide para la constante [c18].

Para ser enviada al *Apache Cordova* la respuesta final, se almacena en una variable llamada [L].

Cálculo de la corrección de la longitud.

$$L_V = 0.82 * \sqrt{S_V}$$ (Ecuación 14)

En el siguiente bloque de programación en Javascript, se crea la variable [sv] y se asigna el valor de [sv] calculada anteriormente en la figura 11. Los siguientes pasos se explica en cada línea de código.

La programación sería:

```
var sv = document.getElementById("entrada8").value;  // Se almacena el valor de SV en sv].
var LV = 0.82 * Math.sqrt(sv):  // La raíz cuadrada de [sv] es multiplicada por '0.82' y es almacenada en la variable [LV].
Pd.send(patch.patchId + '-LV', [LV]); // El resultado es enviado a Pure Data para los siguientes cálculos.
caja.entrada11.value = LV.toFixed(2); // Se visualiza la respuesta final en el dispositivo móvil y se limita a dos decimales.
```

Cálculo de la longitud definitiva:

$$L_l = L - L_V$$ (Ecuación 15)

La programación sería:

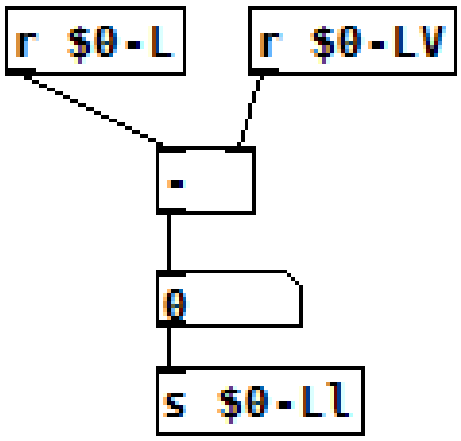

Figura 15. Cálculo de la longitud definitiva en Pure Data

Se recibe el valor de L en la variable [L] y se resta el valor de L_V en la variable [LV], su resultado es asignado a la variable [Ll] y enviado al *Apache Cordova.*

2.4. Herramienta *GEM*

GEM es una herramienta de trabajo de *Pure Data* orientada al entorno gráfico, sus iniciales significan "*Graphics Environment for Multimedia*".Con esta herramienta se puede manipular, procesar y visualizar videos e imágenes. Existen parches en internet donde se pueden encontrar ejemplos realizados por desarrolladores y también es posible acceder a través del *Pd Visor* de ayuda de *Pure Data*, en el navegador. Es muy importante saber que esta herramienta sólo funciona en *Pure Data Extended,* que no es más que una versión de *Pure Data* con librerías externas desarrolladas por programadores informáticos. Estas librerías son externas al lenguaje de programación, por motivo de ser de código libre.

2.5. Compatibilidad de Pure Data con otros lenguajes de programación

Pure Data por ser un lenguaje de programación gráfico orientado para la creación de música por computadora, tiene relación directa con *Max/MSP*, que es el lenguaje que precede a *Pure Data* por ser creados por la misma persona, *Miller Puckette*. *Max/MSP* trabaja igual que *Pure Data* utilizando bloques de construcción de código, siendo un flujo de datos, esto se hace extensible por medio de una *API* (*Application Programming Interface)* que por ser pública puede ser aprovechada por desarrolladores de otros lenguajes de programación en *Phyton, Javascript, Ruby* y más lenguajes afines, logrando así una comunicación abierta con otras plataformas informáticas.

2.5.1. LibPd.- Como se explicó anteriormente sobre la compatibilidad de *Pure Data* con otras plataformas informáticas, *LibPd* es una de las opciones para relacionarse, con otros lenguajes de programación, dado que, es una librería de datos informáticos escritos en lenguaje C y haciendo esto posible trabajar con dispositivos móviles y páginas web.
LibPd no es más que un *Plug in* como se conoce en un software dedicado a la ingeniería de sonido, donde, este se encarga de aprovechar las prestaciones de *Pure Data* enlazándolas con un dispositivo móvil.

2.5.2. Mobmuplat, realizado con Pure Data.- *MobMuPlat* es una aplicación realizada para dispositivos móviles, donde el usuario puede crear *Software* de audio personalizado que se desarrolla en *Pure Data* y el enlace con las plataformas móviles se realiza con *LibPd.* Esta aplicación posee parches programables con ejemplos realizados por defecto. Se encuentra disponible gratis en las tiendas digitales de *IOS* y *Android.*
Este proyecto no se enfocó a la programación utilizando *LibPd*, por motivo de su limitación para trabajar en el sistema operativo *Windows.*

2.5.3. PdDroidParty.- Es una aplicación para dispositivos móviles muy parecida a *MobMuPlat*, donde es una ventana o una interfaz gráfica de los proyectos realizados en Pure Data.
Se descarga la aplicación al móvil y se enlaza con la memoria externa para poder ejecutar un parche de *Pure Data.*

2.5.4. Android Studio, lenguaje exclusivo para *Android*

Android Studio fue desarrollado en el año de 2013 como un entorno de programación exclusivo para el sistema operativo *Android*. En diciembre del año 2014 fue el lanzamiento oficial de la versión estable, reemplazando a Eclipse como el IDE (*Integrated Development Environment*) oficial para dicha plataforma de dispositivos móviles.

Una de sus principales características es la renderización en tiempo real, además del uso de plantillas para la creación de aplicaciones móviles comunes en *Android.*

2.5.5. OSC para Android con Pure Data

OSC (*Open Sound Control*) es un protocolo para la creación de redes informáticas, donde se puede interactuar con sonidos producidos por sintetizadores, ordenadores o cualquier otro dispositivo multimedia. Se puede controlar varios instrumentos musicales electrónicos en todo el mundo y sincronizarlos de una manera eficaz.

En *Pure Data* se puede crear programas utilizando *OSC* como protocolo de comunicación entre un ordenador y un dispositivo móvil, para así poder controlar su funcionamiento por medio de una interfaz gráfica y táctil.

2.6. Aplicaciones realizadas con *Pure Data*

Existen pocas aplicaciones creadas en este lenguaje, ya que *Pure Data* es utilizado más para proyectos de elaboración de osciladores de ondas de sonido y sintetizadores.

A continuación se presentarán algunas aplicaciones disponibles en la tienda digital para *Android*, que se conoce como *Play Store*.

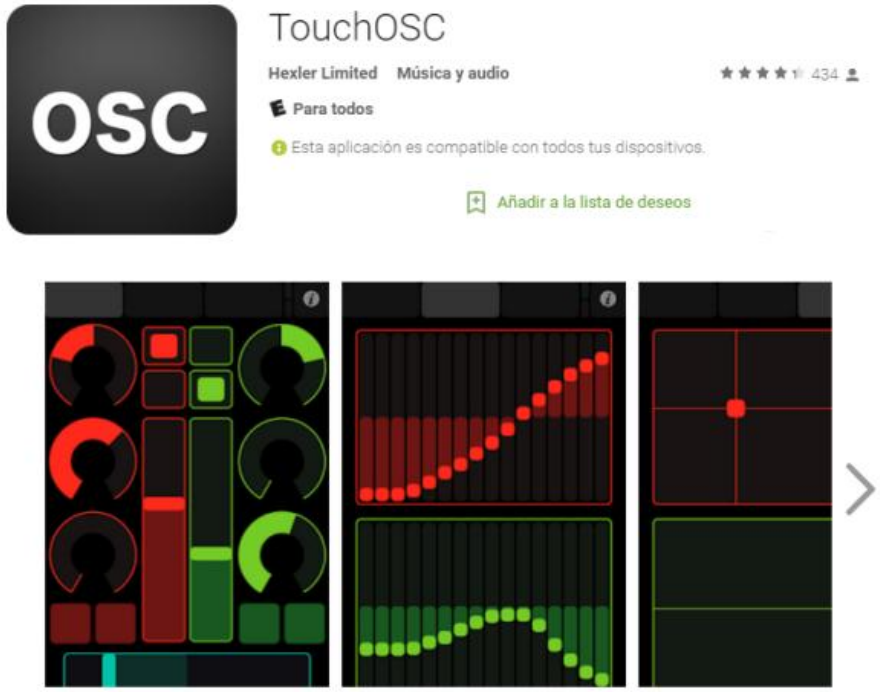

Figura 16. Google(2016).TouchOSC

Esta aplicación fue hecha con *OSC* antes mencionado. Sirve como controlador táctil personalizable para ejecutar un programa realizado en *Pure Data* y en las siguientes plataformas:

Apple Logic Pro/Express, Ableton Live, Max/MSP/Jitter, Max for Live, OSCulator, VDMX, Resolume Avenue/Arena, Modul8, NI Traktor, NI Reaktor, Quartz Composer, Derivative TouchDesigner, Isadora, etc. (Zapata,2014).

Figura 17. Google(2016).mPD

Esta aplicación es una simulación del entorno de trabajo de Pure Data, siendo muy útil para poder entender mejor sus prestaciones como lenguaje de programación.

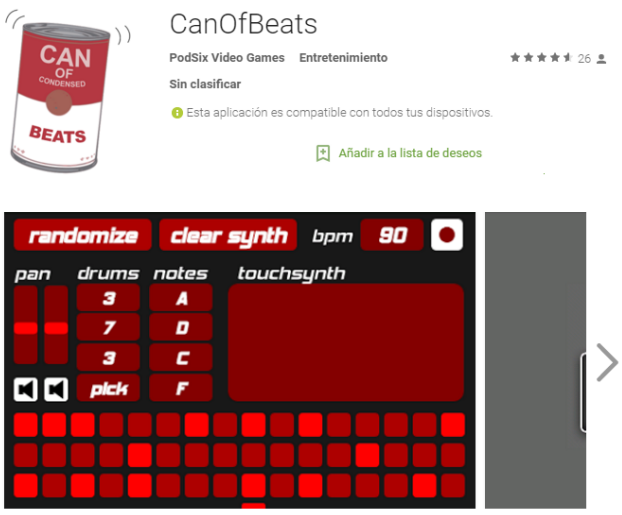

Figura 18. Google(2016).CanOfBeats

La siguiente aplicación fue creada con la implementación de *LibPd* y su principal característica es hacer *Beats* utilizando *Pads* de ritmos, donde también se puede modificar el tiempo y la afinación.

2.7. OSX en PC

Conociendo las limitaciones de *LibPd* para operar en el entorno de *Microsoft Windows*, se investigó la posibilidad de instalar el sistema operativo de *Apple OSX* en una *PC*, llegando a la conclusión que sí es posible. No se optó por hacer este proceso por limitaciones de *Hardware* del equipo utilizado para el proyecto.

2.8. WebPd

Anteriormente se explicó algunas maneras de cómo realizar una aplicación para un dispositivo móvil. *WebPd* trabaja similarmente como *LibPd,* pero con la ventaja de que su ejecución es totalmente compatible con *Javascript,* utilizando *Web API* de audio como herramienta de conexión para reproducir un sonido. Su entorno puede trabajar en *Apple OSX* y en *Microsoft Windows*.

Después de haber estudiado la programación de las fórmulas de los parámetros *Thiele-Small* en *Pure Data*, se ha elegido enlazarlo con una plataforma muy versátil y con muchas prestaciones importantes, conocido como *Microsoft Visual Studio*.

2.9. Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado que en inglés se representa con las iniciales *IDE* (*Integrated Development Environment),* donde su funcionalidad está establecida para el sistema operativo *Windows.* Puede trabajar en múltiples lenguajes de programación como en *Visual Basic, C#, C++, F#, Java, Python, PHP, Ruby, Javascript* y en varios entornos de desarrollo *Web* como *Django, ASP.NET*, etc.

Los desarrolladores de aplicaciones y de páginas *Web* lo eligen por su estabilidad y versatilidad, además por existir la comunicación entre estaciones de trabajo y dispositivos móviles dentro de un mismo entorno. *Windows Azure* es un editor *Online* de programación, el cual brinda los beneficios necesarios para poder seguir trabajando desde cualquier lugar que tenga conexión a internet.

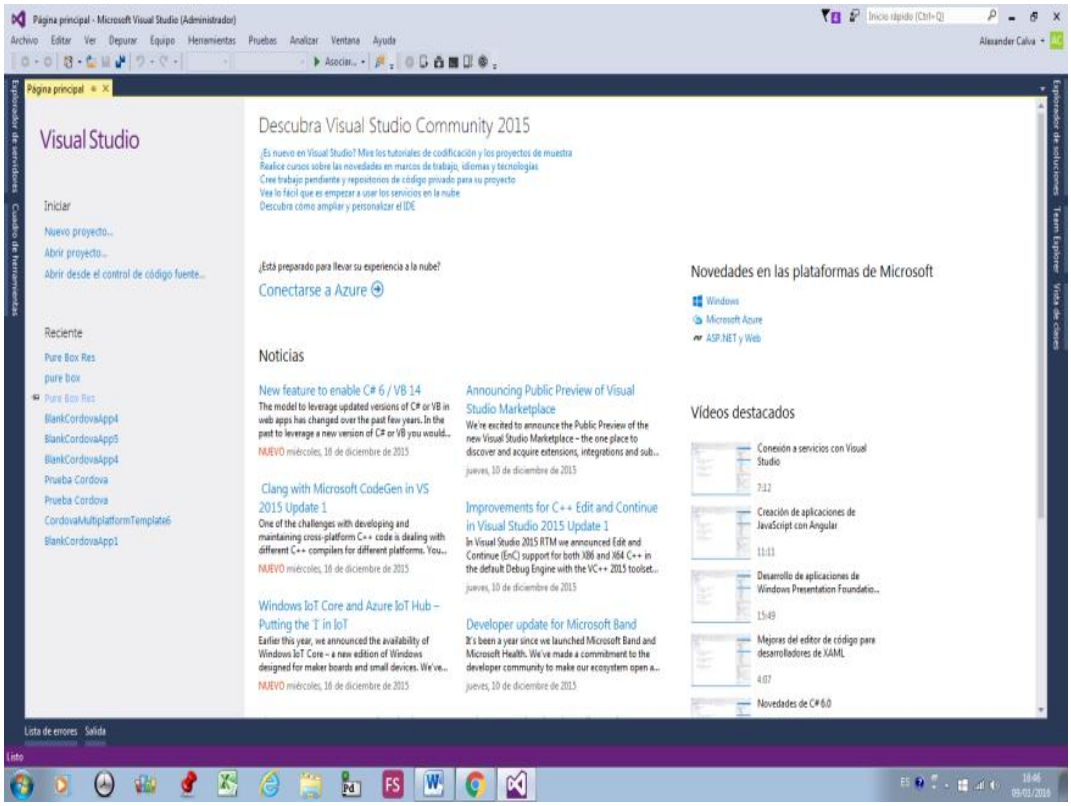

Figura 19. Microsoft Visual Studio

Para crear un nuevo proyecto se visualiza la siguiente ventana.

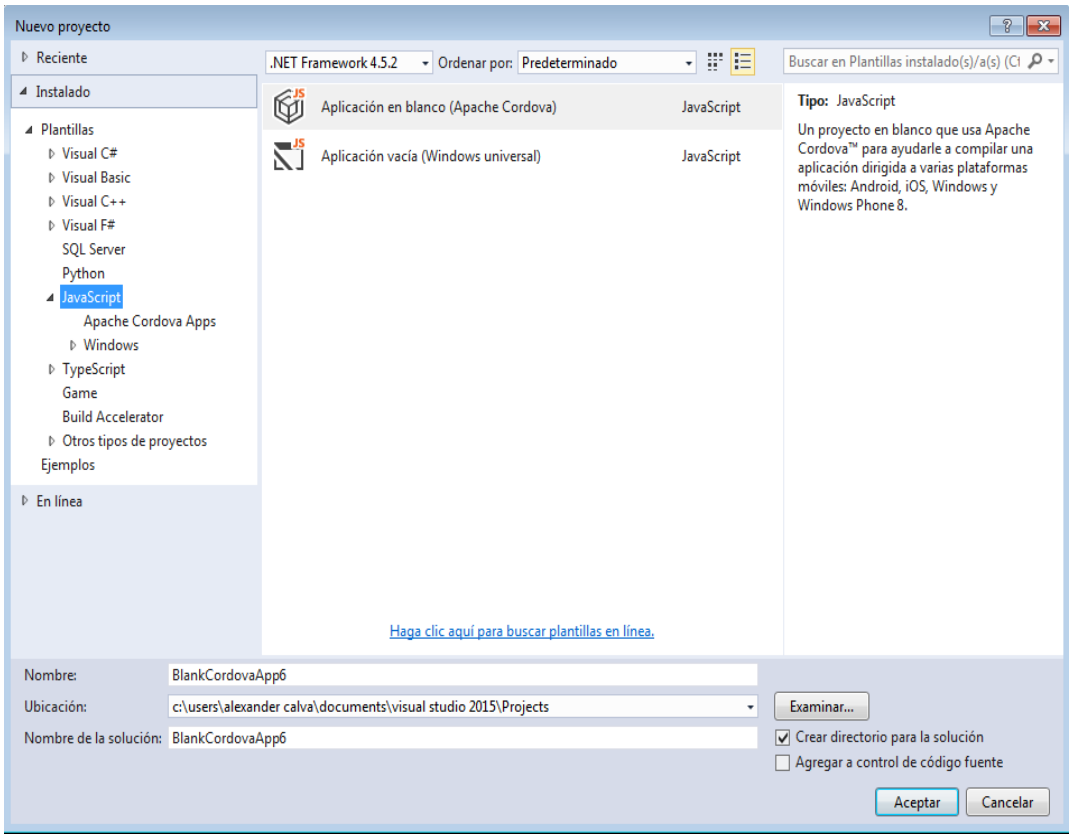

Figura 20. Aplicación en blanco para Apache Cordova

Existe una versión gratuita de este *IDE* que se llama Visual Studio Community, el cual tiene las funciones principales para un desarrollador en programación.

2.10. PhoneGap

Hace algunos años atrás para programar en *Android* se necesitaba utilizar los lenguajes de programación *Java* y *XML*, el cual era un método nativo y complicado. En el año 2009 la empresa canadiense *Nitobi* crea *PhoneGap*, un *Framework* multiplataforma donde se puede trabajar en los tres lenguajes básicos y necesarios para desarrollar aplicaciones para dispositivos móviles dentro de un mismo entorno, los cuales son *Javascript, HTML5 y CSS3* logrando así, poder ser ejecutadas en cualquier sistema operativo móvil.

PhoneGap puede acceder a los elementos principales de un dispositivo móvil como la cámara, el acelerómetro, almacenamiento, etc. Debido a la conexión que posee por *API*.

2.11. Apache Cordova

Es un Framework de código libre, enfocado a la programación multiplataforna, es decir, para dispositivos móviles con sistemas operativos IOS, Windows Phone o Android. En este Framework se trabaja con tres entornos de programación, lo cuales son Javascript, HTML y CSS simultáneamente, dentro de un mismo proyecto.

En febrero de 2012 se renombra a *PhoneGap* como *Apache Cordova*, para evitar cualquier problema legal futuro con la empresa *Adobe*, el nombre de *Cordova* es adquirido por ser el nombre de la calle donde se encontraba la empresa *Nitobi* en *Vancouver* (Canadá).

No existen diferencias importantes entre *PhoneGap* y *Apache Cordova*, su entorno de trabajo es el mismo, lo que varía es el sitio de descarga desde internet, siendo los dos *Open Source*.

A la hora de compilar un programa *PhoneGap* tiene una ventaja, la herramienta *Adobe PhoneGap Build* que puede ejecutar un programa sin tener los *SDK* (*Software Development Kit*) de cada plataforma, en otras palabras, si se quiere compilar una aplicación para *Iphone* no se necesitaría una *MAC* para hacerlo, además esta herramienta compila en otras plataformas del mercado como *Android, Windows Phone, Blackberry 5/6/7 y WebOS.*

Algo muy importante es saber, que para poder compilar una aplicación comercial el servicio no es gratuito, solo sirve para casos de *Open Source*.

2.11.1. Emulación de una *App* para *Android.-* En *Visual Studio* dentro del *Apache Cordova* posee una herramienta muy necesaria para desarrollar una aplicación, se trata de un emulador de un dispositivo móvil de la plataforma que se desee trabajar, en este caso se va a simular a un teléfono inteligente con sistema operativo *Android.* Esta herramienta sirve para conocer los resultados en tiempo real de una aplicación y desarrollar correctamente la interfaz de usuario.

El desarrollador puede elegir el dispositivo móvil a simular, en este caso se tiene el *Ripple-Nexus S, Ripple-Nexus 7 (Tablet), Ripple-Nexus (Galaxy).* En la misma lista de opciones de la ventana del emulador, se visualiza la opción de dispositivo, el cual es para conectar directamente un *Smartphone* a la computadora por medio de un cable *USB* y ejecutar la aplicación en el mismo dispositivo Antes de hacer ese proceso se debe instalar las herramientas y controladores respectivos del *Smartphone*, Vpara lograr así una completa compatibilidad con la computadora y el *Software* utilizado.
La herramienta *SDK,* se explicará más adelante sobre la importancia que tiene para poder ejecutar una aplicación en un dispositivo móvil con sistema operativo *Android*.

2.11.2. Enlazando Pure Data con Apache Cordova

Se ha explicado sobre las dos plataformas de programación, donde *Pure Data* es un lenguaje de programación gráfico basado en estructura de datos por bloques y del *Framework Apache Cordova* que es un entorno de programación multiplataforma.

Ahora se realizará el enlace entre estos dos programas y se detallará que se necesita para hacer este proceso. Primeramente se crea un nuevo proyecto dentro de *Visual Studio* donde se escogerá la herramienta *Apache Cordova* como se observa en la figura 21.

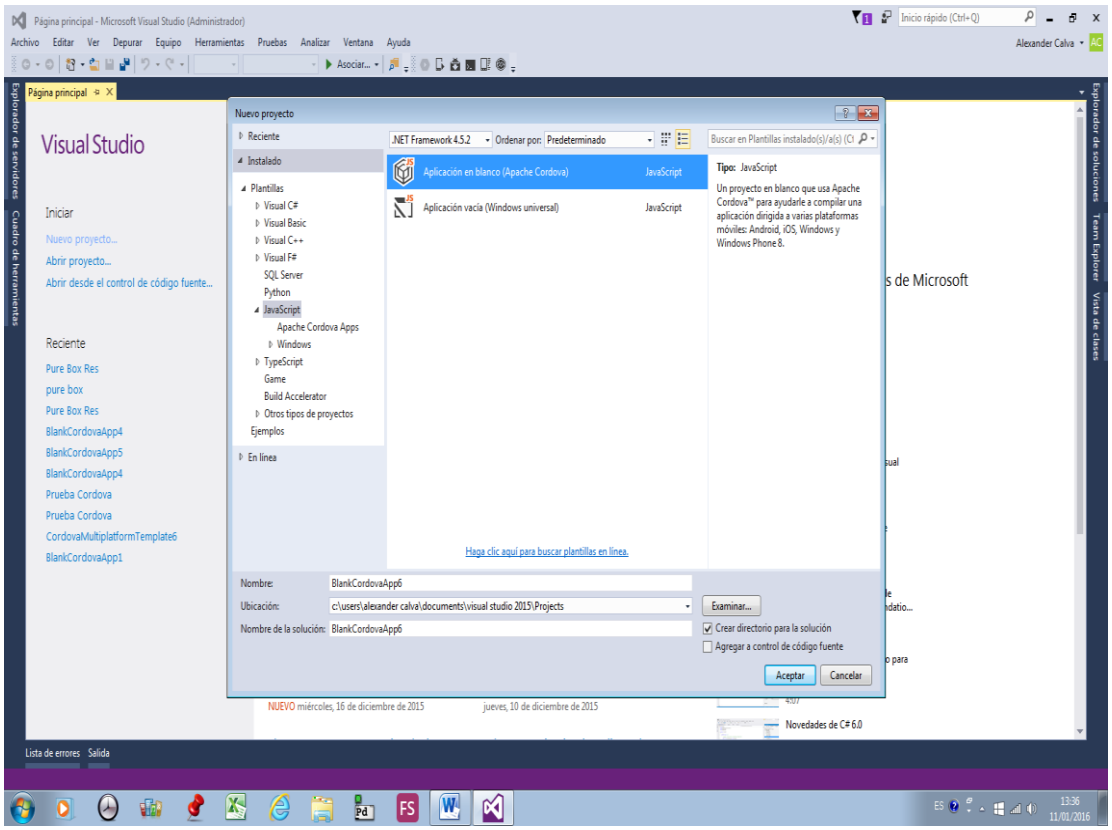

Figura 21. Proyecto nuevo en Apache Cordova

Si al momento de crear un nuevo proyecto no aparece la opción de aplicación en blanco (*Apache Cordova*), se necesita instalar los componentes externos de *Visual Studio* y asegurarse de que dentro de ellos este *Apache Cordova*.

Ya una vez creado el nuevo proyecto se visualizará una nueva pantalla donde se despliega una introducción del Apache Cordova y sus principales características.

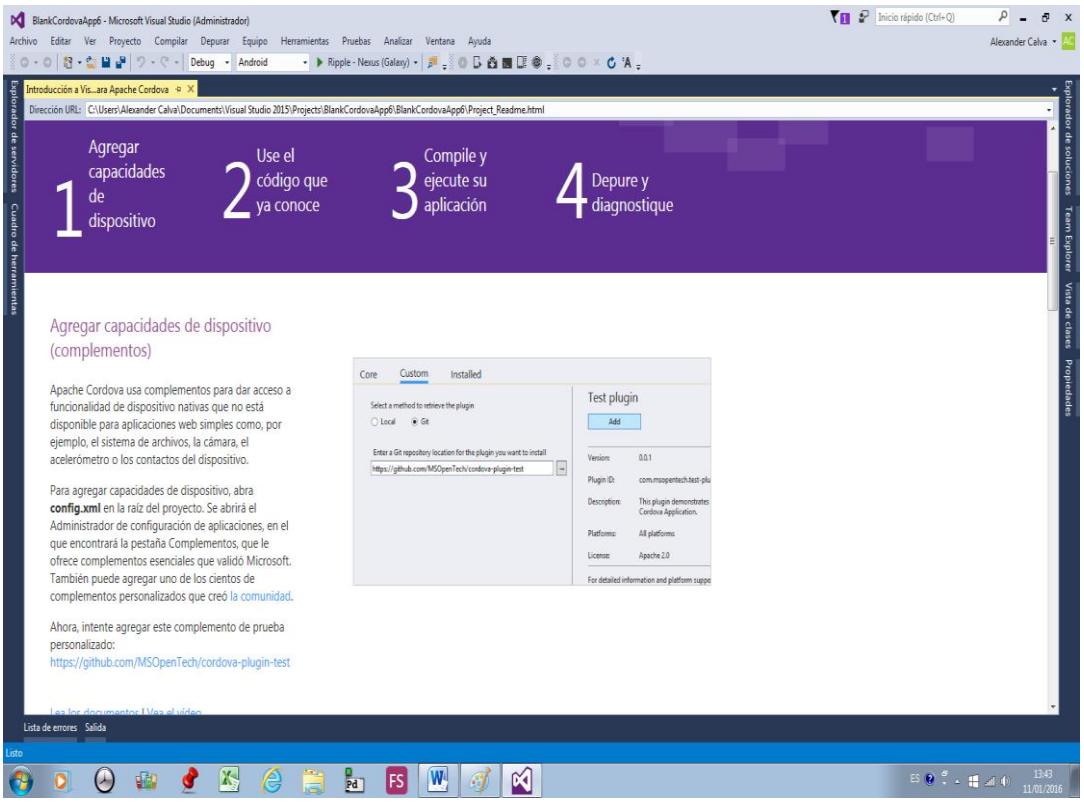

Figura 22. Introducción al Apache Cordova

Entendida la introducción se acepta las sugerencias y se ingresa al nuevo proyecto que por defecto se llamará *BlankCordovaApp.sln*, donde se visualizan las siguientes carpetas.

Dentro de la carpeta llamada *Scripts* se va a incluir el archivo *WebPd* del cual se explicó anteriormente, donde será llamado desde el lenguaje HTML5 más adelante.

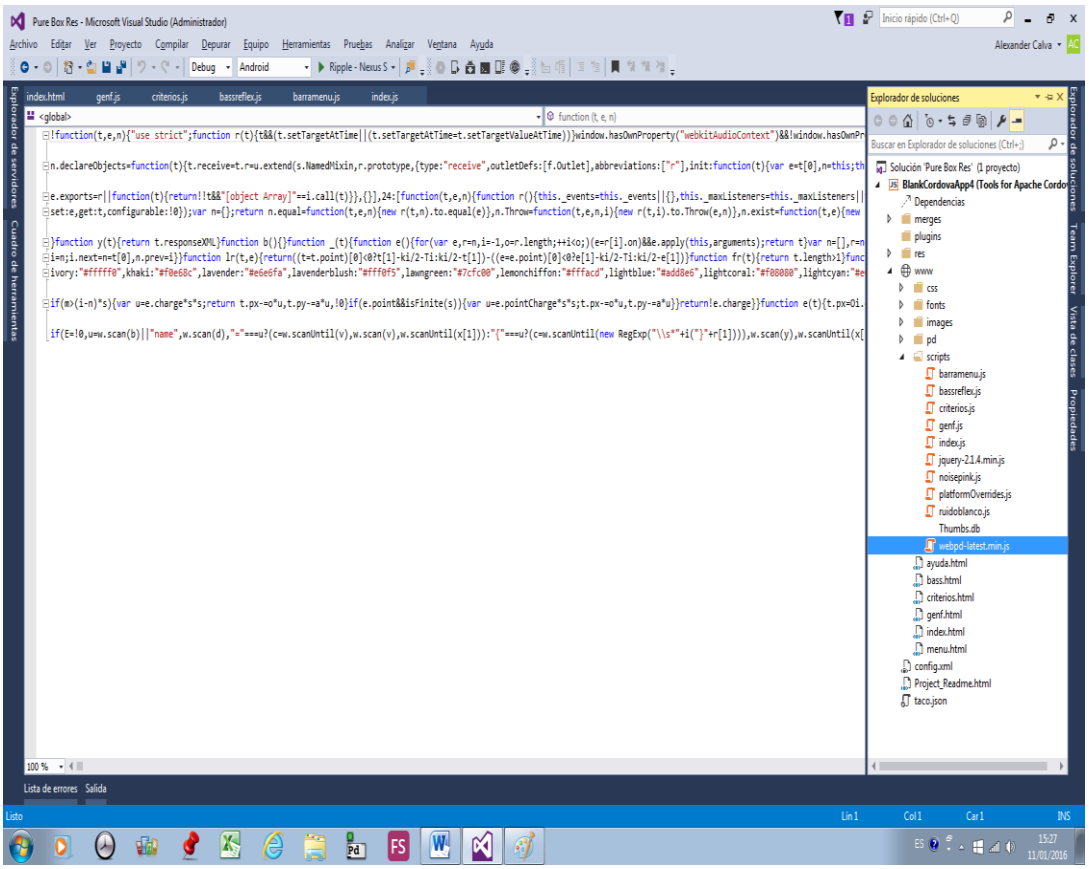

Figura 23. Incluyendo WebPd en el nuevo proyecto

Una vez insertado el archivo *WebPd* dentro de la capeta *Scripts*, se crea un nuevo archivo y se escoge de la lista que se despliega a continuación.

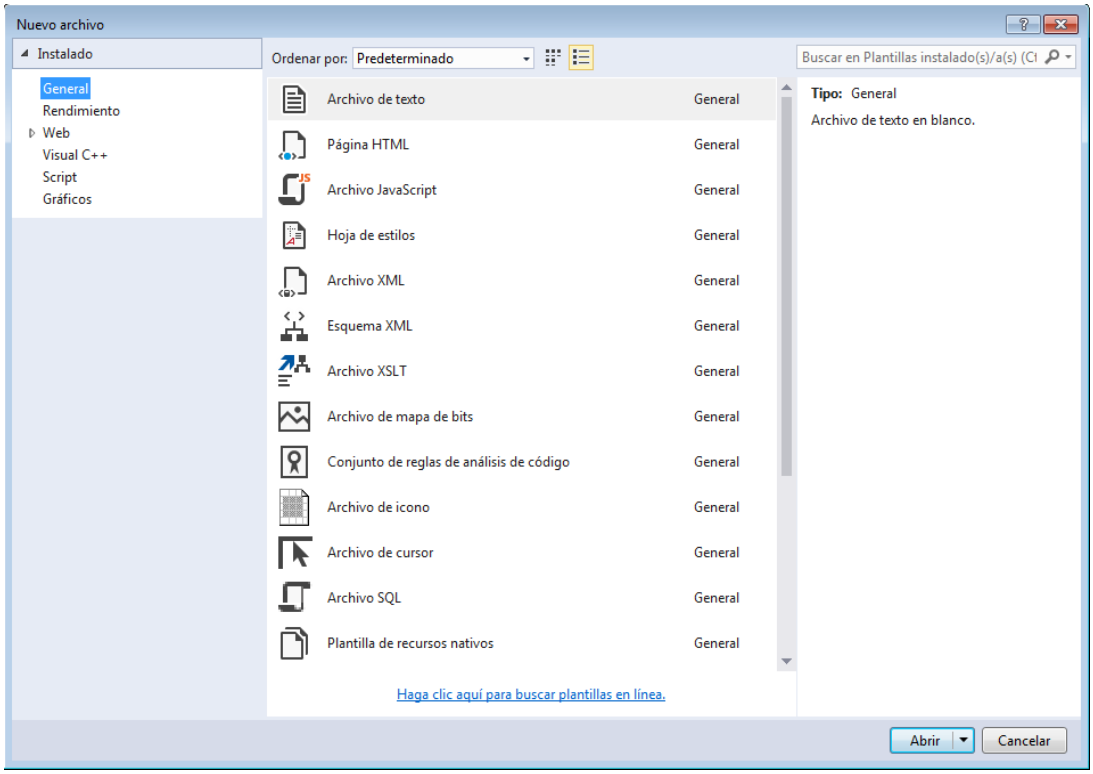

Figura 24. Creando un nuevo archivo de programación

Los tipos de archivo Java*script* contienen código *Javascript* como su nombre lo dice, los cuales sirven para programar todos los cálculos y funciones a desarrollar en la aplicación, el que dice "Página *HTML* "se encarga de la programación HTML, es decir, estilo páginas *Web*, en otras palabras visualizará al usuario la aplicación en la pantalla del dispositivo móvil. Y por último el tipo de archivo que se va a utilizar, es el que dice "Hoja de Estilos", este tipo de archivo se encarga de enriquecer los estilos para desarrollar páginas *HTML*, esto lo hace por medio de plantillas por defecto ya establecidas, también se lo conoce como *CSS3*.

2.11.3. Envío de datos desde *Apache Cordova* a *Pure Data*

Ya dentro del nuevo proyecto se realizará una prueba de enlace entre *Apache Cordova* y *Pure Data*, para lo cual primero se va a crear un programa sencillo en *Pure Data*, donde va a generar frecuencias establecidas por el usuario, este programa se llamará “sound.pd”, como se observa en la figura 25.

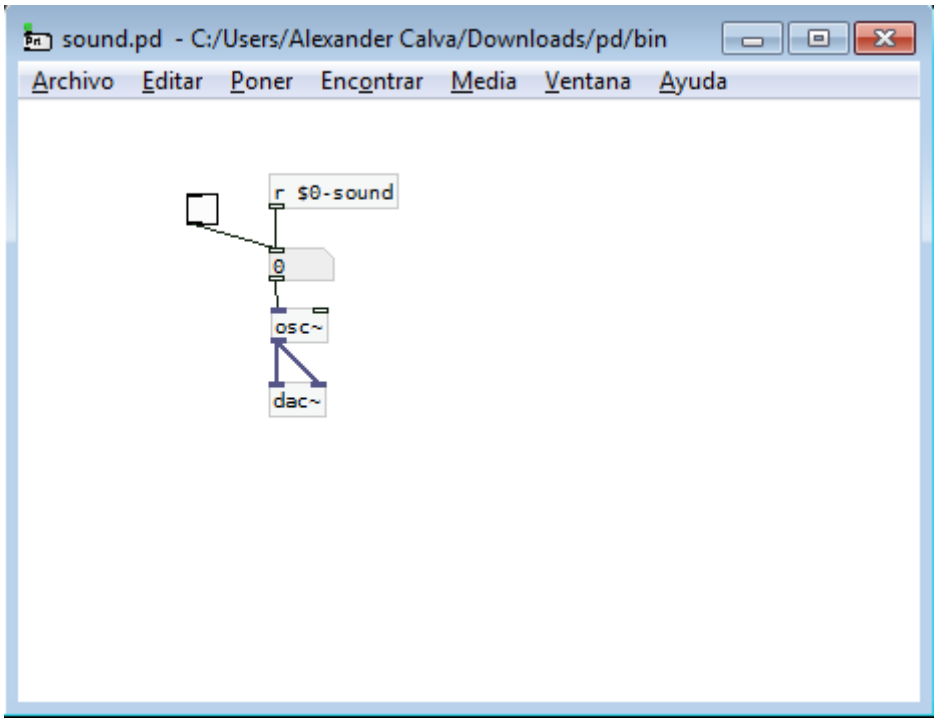

Figura 25. Programa generador de frecuencias en Pure Data

Donde [r $0-sound] significa la recepción del valor de la frecuencia ingresada por el usuario y almacenada en la variable llamada [sound]. Este valor es ingresado por medio de la interfaz creada en *Apache Cordova.* Este bloque de código está conectado con un objeto de tipo *Number*, para poder visualizar el valor de la frecuencia ingresado.

A continuación se conecta un objeto de tipo *Object* donde se escribe [osc~], este código representa a una oscilación de ondas de frecuencia realizada según el valor ingresado por el usuario, luego para finalizar se conecta un objeto de tipo *Object* donde se escribe [dac~] , el cual realiza una conversión de una señal digital a analógica, por esta razón se puede escuchar la frecuencia generada.

Todo el proceso está controlado por un objeto llamado *Toggle* que se puede apreciar como un cuadrado en forma de *Switch*, el cual activa o desactiva la generación de la frecuencia ingresada.

Una vez creado el programa “sound.pd”, se almacena dentro del proyecto de *Apache Cordova* en la carpeta llamada “www” y a su vez dentro de la carpeta llamada “pd”.

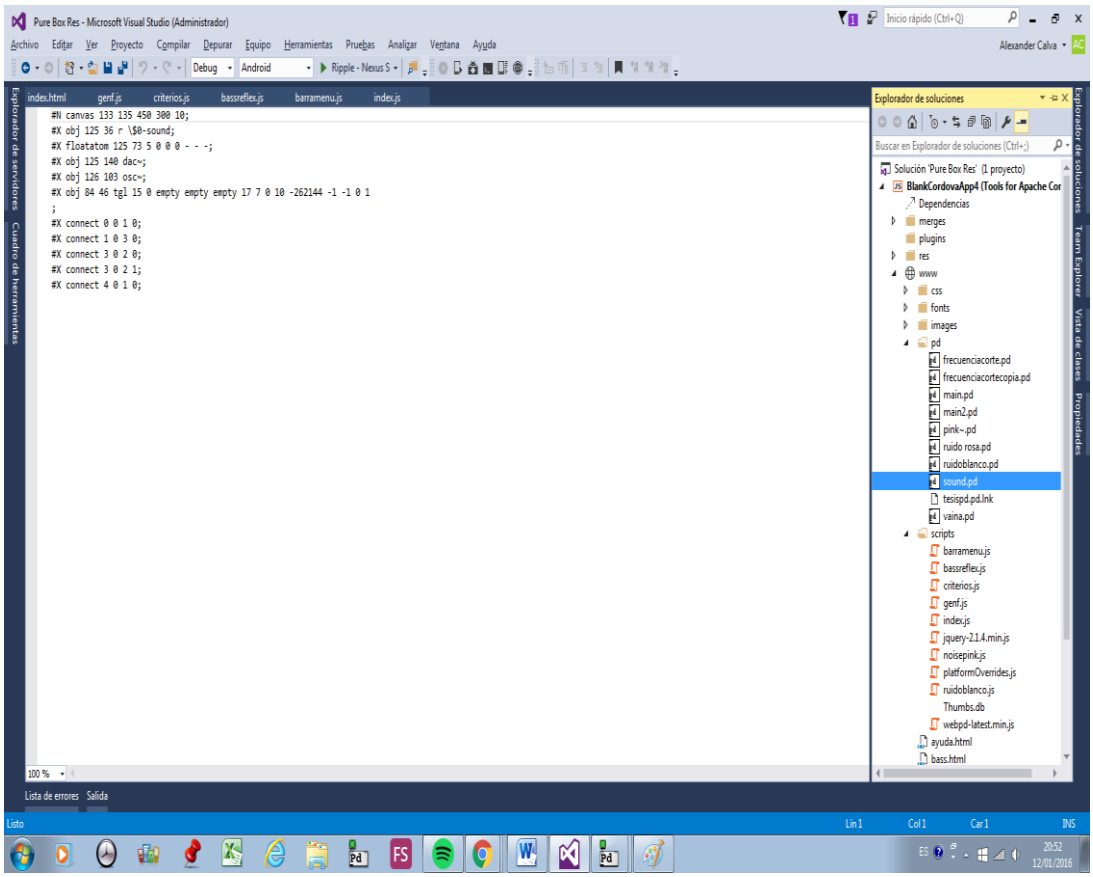

Figura 26. Incluyendo el programa de Pure Data en el proyecto

Ahora se procederá a realizar la programación en el *Apache Cordova,* para lo cual se creará un archivo de tipo *Script* y se llamará “genf.js”, como se muestra en la figura 27.

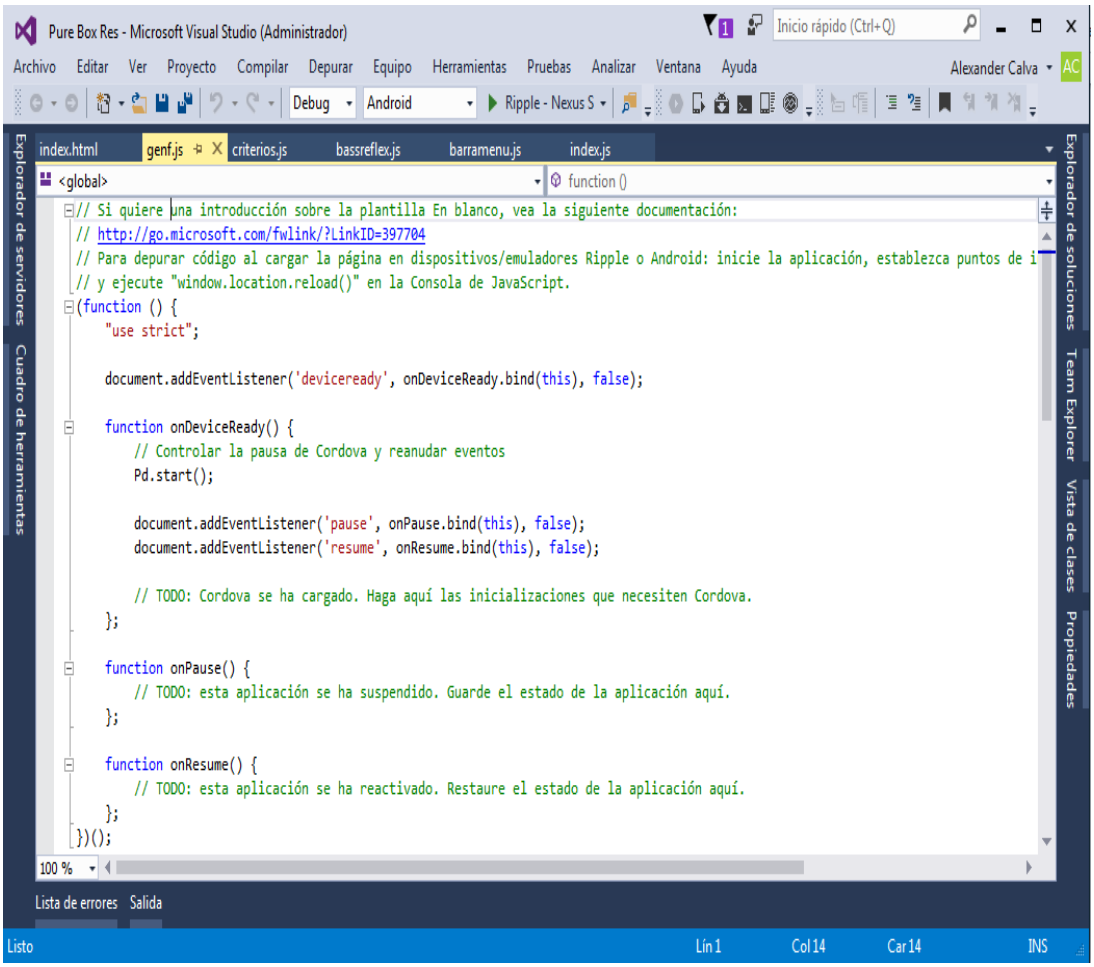

Figura 27. Programa generador de frecuencias en Javascript

Al crear el archivo mencionado se crea por defecto una plantilla con funciones necesarias para la correcta ejecución del programa. Luego se empieza por la programación, se recuerda que se está trabajando en el lenguaje *Javascript* por esta razón el archivo tiene la extensión “js”.

En el siguiente bloque de programación se crea una función llamada Sonido, la cual, se encarga de realizar los cálculos respectivos para reproducir una frecuencia elegida por el usuario. Para esto se necesita incluir una variable llamada contador, que almacena una sucesión de números a manera de un contador, además se va a llamar al *Patch* creado en *Pure Data* para generar una frecuencia y se va a asignar a una variable llamada [patch].

```
//Se crea la función llamada Sonido.
function Sonido() {
//Se define [cont] como variable contador.
        cont = cont + 1;
        // Se utiliza una sentencia If condicional.
        If (cont == 1) {        /
        // Se crea la variable [patch] que servirá para almacenar el parche de Pure Data anteriormente realizado.
        var patch
        // Se llama al archivo “sound.pd” que se encuentra dentro de la carpeta “pd”.
                $.get('./pd/sound.pd', function (mainStr) {
        // Se carga el parche de Pure Data y se lo asigna a la variable [patch]
                patch = Pd.loadPatch(mainStr);
        //Se asigna a la variable [frec] el valor de la frecuencia ingresado por el usuario.
                        var frec = parseFloat($("#FREC").val());
        // La sentencia while sirve como control para validar la entrada de los datos, que sean desde 20  a 20000 Hz.
                        while (frec < 20 || frec > 20000) {
        //Despliegue de ventana con mensaje visual para el usuario, hasta que ingrese un número dentro del rango auditivo establecido..
                                alert("Frecuencia fuera de rango auditivo");
        break;
        }
```

```
//validación de la entrada de datos, si cumple con la condición dada, continúa con el cálculo.
                    if (frec >= 20 && frec <= 20000) {
//Envío para Pure Data el valor ingresado por el usuario y almacenado en la variable [frec].
                        Pd.send(patch.patchId + '-sound', [frec]);
// Activa el objeto llamado Toggle en Pure Data y ejecuta el programa.
                        Pd.start();
} else {
//Mensaje visual donde informa al usuario que no ha ingresado ningún dato.
                        alert("Ingrese frecuencia a generar");
refresh(); // Refresca la página.
}
})
}
cont = cont + 1;       //Suma uno al contador.

};    // Finalización de la función sonido.
```

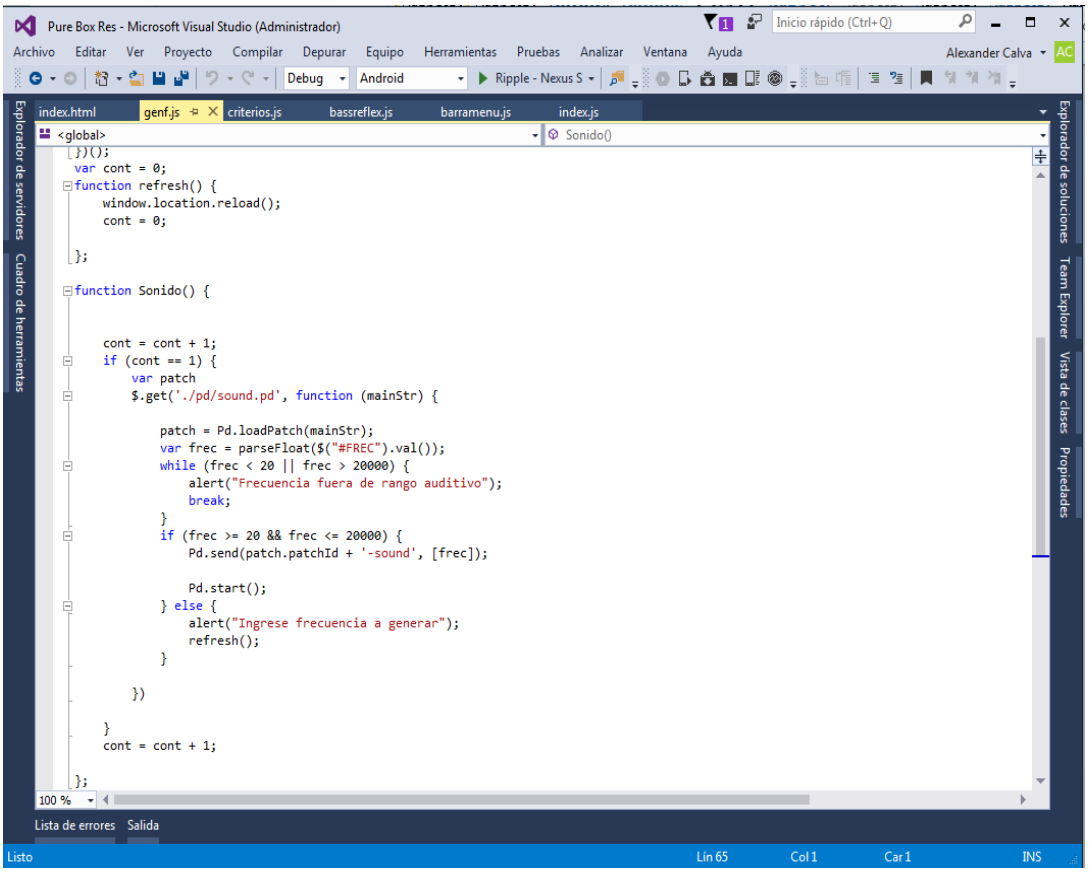

Figura 28. Programación en Javascript para generar una frecuencia

2.11.4. Recepción de datos del *Apache Cordova*

Establecido el envío de información como se detalló anteriormente al *Pure Data*, ahora se explicará cómo se reciben los datos enviados al *Apache Cordova*. Para este caso se tomará como ejemplo el cálculo de la frecuencia de resonancia, que fue hecho en *Pure Data.*

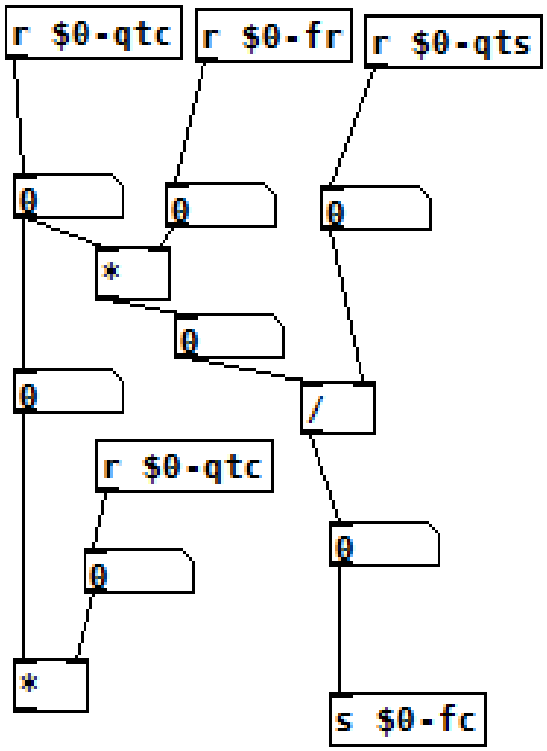

Figura 29. Envío de resultado desde Pure Data a Apache Cordova

Como se observa en la figura 29, se envía información desde el *Apache Cordova* el valor de las variables [qtc], [fr] y [qts], luego se hace el cálculo para encontrar [fc] que es la frecuencia de resonancia de una caja acústica y se envía el resultado por medio de la expresión [s $0-fc], donde este valor es recibido por el *Apache Cordova* de la siguiente manera.

```
Pd.receive(patch.patchId + '-fc', function (res)
{
        $("#entrada1").val(res);
});
```

// Se recibe el valor de [fc] y se lo asigna a una casilla de texto llamada "entrada1" donde se visualizará el resultado. Esta casilla está en código fuente *HTML*.

HTML como se explicó anteriormente, está dedicado para la creación de páginas y entornos *Web*. Este lenguaje se utiliza para poder crear la interfaz de la aplicación, la cual se explicará más adelante.

2.12. Ingreso del código fuente para calcular los parámetros *Thiele –Small*

El proyecto establecido como ya se ha detallado, consiste en calcular los parámetros *Thiele-Small* utilizando la programación informática. Para lo cual se ingresará el código fuente de los lenguajes respectivos.

2.13. Programación para la aplicación móvil

Para poder crear una aplicación o una *App* se necesita manejar tres tipos de lenguajes de programación, los cuales son *HTML5*, *CSS3 y JavaScript.* Para realizar este proyecto se incluyó además otro lenguaje, el cual es *Pure Data.*

2.13.1. HTML5, última versión para programación en páginas web

HTML5 es la versión más reciente para la creación de páginas Web y especialmente está enfocado para la elaboración de aplicaciones móviles.
El código fuente que se utilizó para la creación de la aplicación para el proyecto, se realizó en partes, el primero a explicar es del menú de la *App*.

Se tomará en cuenta para la explicación el código realizado, más que el que viene por defecto en la programación.

En el siguiente bloque de programación se crea la interface de usuario, es decir, la pantalla que se visualizará en el *Smartphone* para el menú.

2.13.1.1. Programación del menú

```
<!DOCTYPE html>
<html lang=”en” xmlns=”http://www.w3.org/1999/xhtml”>
//Encabezado, donde se incluyen archivos de tipo CSS.
<head>
<meta charset=”utf-8” />
<title> MENU INICIO</title>
<link href=”css/index.css” rel=”stylesheet” />
<link rel=” stylesheet” href=”fonts/style.css” >
<meta name=”viewport” content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>
</head>
//Cuerpo de la programación, donde se escoge el fondo y estilo del entorno Web.
<body style=”background-image:url(‘images/fondored2.png’);”> // Fondo de pantalla.
<!-- <body style=”background-color: #DAA520”>→ // Color de fondo.
<header>
// Llamando al tipo de clases declaradas en CSS para establecer íconos y estilos.
<div class=” 47ont_bar”>
//Escogiendo la clase de tipo menú y su ícono.
<a href=”#” class=” bt-menu”><span class=”icon-list”></span>MENÚ</a>
</div>
<nav>
<ul>
// Al dar un click el usuario se direcciona automáticamente a la página menú.html.
<li><a href=”# “ onclick=”window.location.href=’menu.html’”><span class=”icon-flag”>
</span> Inicio</a></li>     //Etiqueta inicio.
//Llamando a la página index.html.
```

```
<li><a href=”# “ onclick=”window.location.href=’index.html’”>
<span class=”icon-document-landscape”>
</span> Caja Cerrada</a></li> // Etiqueta de caja cerrada.
//Direccionando con un click a la página bass.html, que es la que calcula una caja Bass Reflex.
<li><a href=”# “ onclick=”window.location.href=’bass.html’”><span class=”icon-notification”>
</span> Caja con Bass Reflex</a></li>      //Etiqueta Bass Reflex
//Direccionando con un click a la página criterios.html, para calcular los criterios de proporciones.

<li><a href=”#”onclick=”window.location.href=’criterios.html’”>
<span class=”icon-progress-empty”>// Tipo de ícono
</span>Criterios de Proporciones </a></li> // Etiqueta Criterios de proporciones.
//Direccionando con un click a la página genf.html, para generar frecuencias escogidas.

<li><a href=”#” onclick=” window.location.href=’genf.html’”><span class=”icon-sound-mix”></span>Generación de frecuencias</a></li> //Etiqueta Generación de frecuencias.
//Direccionando con un click a la página ayuda.html, ayuda sobre los parámetros Thiele-Small

<li><a href=”#” onclick=” window.location.href=’ayuda.html’”>
<span class=”icon-help-with-circle”></span>Glosario</a></li>          //Etiqueta Glosario
</ul>
</nav>
</header>
// Se incluyen scripts con librerías y páginas de códigos fuente.
```

```
<script src=”scripts/jquery-2.1.4.min.js”></script>
<script src=”cordova.js”></script>
<script src=”scripts/platformOverrides.js”></script>
<script src=”scripts/barramenu.js”></script>
<script src=”scripts/index.js”></script>
<script src=”scripts/webpd-latest.min.js”></script>
</body> // Finalización del cuerpo de la página
</html> // Fin de la programación.
```

La imagen del menú en el dispositivo móvil quedaría así.

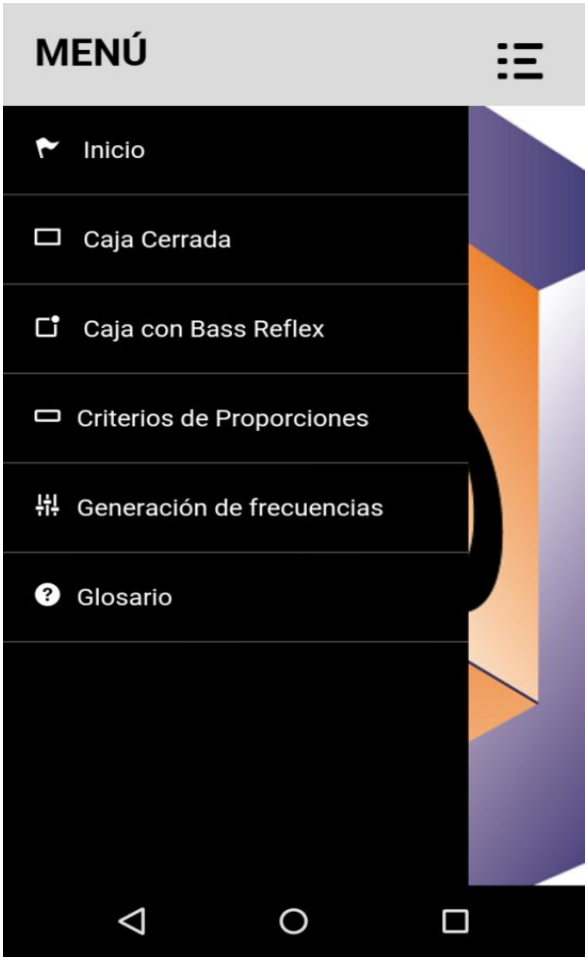

Figura 30. Menú desplegable de la aplicación móvil

Siguiente paso es la programación para calcular los parámetros *Thiele-Small* de una caja acústica cerrada.

2.13.1.2. Programación para una caja acústica cerrada

```
<!DOCTYPE html>
<html>
<head>
<meta charset=”utf-8” />
<title>PruebaCordova</title>
<!—PruebaCordova references →
<link href=”css/index.css” rel=”stylesheet” /> //Vinculando con CSS.
<link href=”css/50ont50.css” rel=”stylesheet” />//Llamando a los estilos para los botones en CSS
//Estilos de fuentes creados en CSS.
<link rel=” stylesheet” href=”fonts/style.css”>
<meta name=”viewport” content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>
</head>
//Cuerpo de la página.
<body style=”background-image:url(‘images/logopbcenter.png’);”> //Fondo de pantalla
<header>
//Incluyendo las clases tipo menú creadas en CSS.
<div class=” menu_bar”>
<a href=”#” class=” bt-menu”><span class=”icon-list”>
</span>MENÚ</a> //Etiqueta MENÚ.
</div>
<nav>
<ul>
// Configuración para crear vínculos para otras páginas al dar un click.
<li><a href=”# “ onclick=”window.location.href=’menu.html’”><span class=”icon-flag”></span> Inicio</a></li>
```

```
<li><a href="# " onclick="window.location.href='index.html'"><span class="icon-document-landscape"></span> Caja Cerrada</a></li>
<li><a href="# " onclick="window.location.href='bass.html'"><span class="icon-notification"></span> Caja con Bass Reflex</a></li>
<li><a href="#" onclick="window.location.href='criterios.html'"> <span class="icon-progress-empty"></span>Criterios de Proporciones </a></li>
<li><a href="#" onclick=" window.location.href='genf.html'"><span class="icon-sound-mix"></span>Generación de frecuencias</a></li>
<li><a href="#" onclick=" window.location.href='ayuda.html'"><span class="icon-help-with-circle"></span>Glosario</a></li>
</ul>
</nav>
</header>
<form id="caja">
<center>
<font color="#aa793b">
<br />
<p><h1><center>PURE BOX</center></h1></p> //Etiqueta PURE BOX.
<br /><br />
<div>
<table border="1" width="200"> //Creación de una tabla.
<tr>
<td>
<label for="number-pattern">Q<sub>TC</sub></label> //Etiqueta QTC
//Ingreso del valor de QTC realizado por el usuario.
<input type="number" id="QTC" value=" ">
</td>
<td>
//Etiqueta FR
     <label for="number-pattern">F<sub>R</sub></label>
//Ingreso del valor de FR realizado por el usuario.
<input type="number" id="FR">
```

```
</td>
</tr>
<tr>
<td>
<label for=”number-pattern”>Q<sub>TS</sub></label> //Etiqueta Q_TS
//Ingreso del valor de Q_TS realizado por el usuario.
<input type=”number” id=”QTS” pattern=”[0-9]*”>
</td>
<td>
<label for=”number-pattern”>V<sub>AS</sub></label> //Etiqueta V_AS
//Ingreso del valor de V_AS realizado por el usuario.
<input type=”number” id=”VAS” pattern=”[0-9]*”>
</td>
</tr>
</table>  //Finalización de la tabla.
</div>
</52ont>
</center>
<div>
<br />
<center>
//Llamando a la función Cálculo creada en Javascript al hacer un click en el botón.
<a href=”#” onclick =”Calculo();” class=”button blue radius”>
<span class=”icon-calculator”></span>Calcular // Etiqueta Calcular.
</a>
<52ont color=”#fff”> // Escogiendo el color de la fuente.
<table border=”1” width=”300”> // Creando tabla para los resultados
<caption align=top><h2> RESULTADOS </h2></caption>// Etiqueta
<br /></h2>
<br />
<br />
```

```
<tr>
//Etiquetas de los resultados.
<th>Fc (Hz)</th>
<th>F<sub>-3</sub> (Hz)</th>
<th>V<sub>AB</sub> (L)</th>
<th>V<sub>B</sub> (L)</th>
</tr>
<tr>
// Deshabilitando la escritura en las Casillas de los resultados y configurando su tamaño.
<td> <input type=”text” id=”entrada1” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” name=”entrada2” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada3” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada4” disabled=”disabled” size=” 5” maxlength=”50” /></td>
</tr>
</table> // Fin de la tabla.
</53ont>
</center>
</div>
<center>
// Borrado de todos los datos al dar un click en el botón ‘Borrar’.
<a href=”#” onclick=”refresh()” class=”button yellow radius”>
<span class=”icon-cycle”></span>Borrar
</a>
// Incluyendo archivos y librerías externas.
</center>
</form>
<!—Cordova reference, this is added to your app when it’s built. →
```

```
<script src=”scripts/jquery-2.1.4.min.js”></script>
<script src=”cordova.js”></script>
<script src=”scripts/platformOverrides.js”></script>
<script src=”scripts/barramenu.js”></script>
<script src=”scripts/index.js”></script>
<script src=”scripts/webpd-latest.min.js”></script>
</body>
</html> //Finalización de la programación.
```

La interfaz quedaría así:

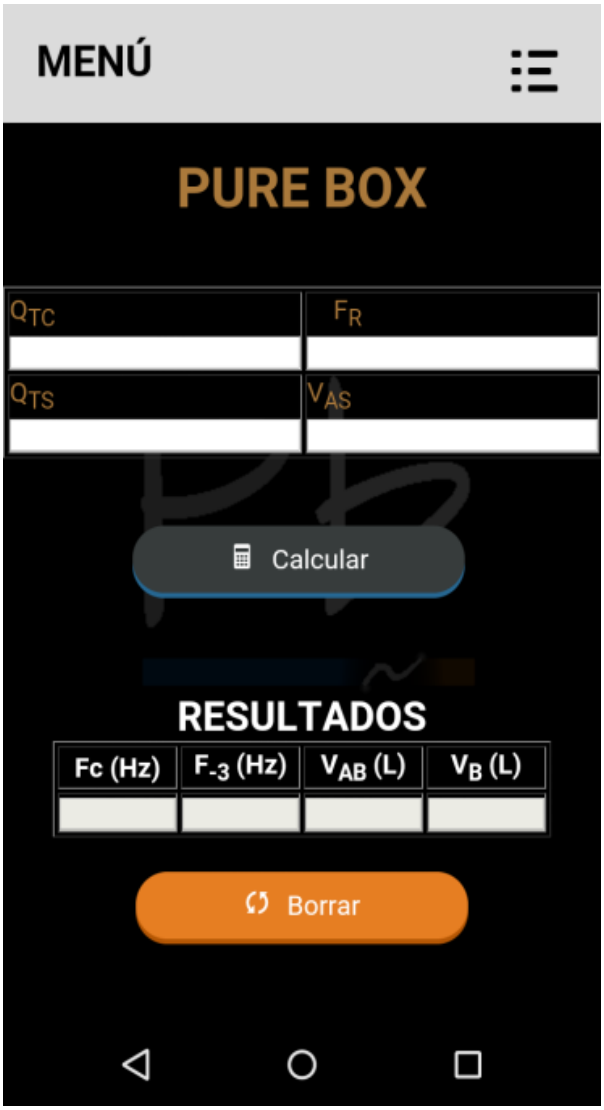

Figura 31. Interfaz de usuario para el cálculo de una caja acústica

2.13.1.3. Programación para una caja acústica con *Bass Reflex.*

```
//Esta parte es la misma de las otras opciones de cálculo, porque programa el menú de la aplicación.
<!DOCTYPE html>
<html>
<head>
<meta charset=”utf-8” />
<title>BASS REFLEX</title>
<link rel=” stylesheet” href=”fonts/style.css”>
<meta name=”viewport” content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>
<!—PruebaCordova references →
<link href=”css/index.css” rel=”stylesheet” />
<link href=”css/boton.css” rel=”stylesheet” />
</head>
<body style=”background-image:url(‘images/logopbcenter.png’);”>
//Menú.
<header>
<div class=” menu_bar”>
<a href=”#” class=” bt-menu”><span class=”icon-list”></span>MENÚ</a>
</div>
<nav>
<ul>
<li><a href=”# “ onclick=”window.location.href=’menu.html’”><span class=”icon-flag”></span> Inicio</a></li>
<li><a href=”# “ onclick=”window.location.href=’index.html’”><span class=”icon-document-landscape”></span> Caja Cerrada</a></li>
<li><a href=”# “ onclick=”window.location.href=’bass.html’”><span class=”icon-notification”></span> Caja con Bass Reflex</a></li>
<li><a href=”#” onclick=”window.location.href=’criterios.html’”> <span class=”icon-progress-empty”></span>Criterios de Proporciones </a></li>
```

```
<li><a href=”#” onclick=” window.location.href=’genf.html’”><span class=”icon-sound-mix”></span>Generación de frecuencias</a></li>
<li><a href=”#” onclick=” window.location.href=’ayuda.html’”><span class=”icon-help-with-circle”></span>Glosario</a></li>
</ul>
</nav>
</header>
<form id=”caja”>
<center>
<font color=”#aa793b”>
<br />
<p><h1><center>PURE BOX</center></h1></p>
<br />
<div>
<table border=”1” width=”200”>
<tr>
<td>
//Etiqueta F_R.
<label for=”number-pattern”>F<sub>R</sub></label>
//Ingreso del valor de F_R realizado por el usuario.
<input type=”number” id=”FR”>
</td>
<td>
//Etiqueta Q_TS
<label for=”number-pattern”>Q<sub>TS</sub></label>
//Ingreso del valor de Q_TS realizado por el usuario.
<input type=”number” id=”QTS” pattern=”[0-9]*”>
</td>
</tr>
<tr>
<td>
//Etiqueta V_AS
```

```
<label for=”number-pattern”>V<sub>AS</sub></label>
//Ingreso del valor de V_AS realizado por el usuario.
<input type=”number” id=”VAS” pattern=”[0-9]*”>
</td>
//Etiqueta S
<td>
<label for=”number-pattern”>S</label>
//Ingreso del valor de S realizado por el usuario.
<input type=”number” id=”S” pattern=”[0-9]*”>
</td>
</tr>
<tr>
<td>
//Etiqueta Diámetro
<label for=”number-pattern”>Diámetro (cm)</label>
//Ingreso del valor del diámetro realizado por el usuario.
<input type=”number” id=”DIA” pattern=”[0-9]*”>
</td>
</tr>
</table>
</div>
</font>
</center>
<div>
<center>
//Llamando a la función cálculo realizada en Javascript.
<a href=”#” onclick=”Calculo();” class=”button blue radius”>
<span class=”icon-calculator”></span>Calcular
</a>
<font color=”#fff”>
<br />
<table border=”1” width=”300”>
```

```
<caption align=top><p><h3> RESULTADOS</h3></p> </caption>
<tr>
<th>V<sub>B</sub> (m<sup>3</sup>)</th>
<th>F<sub>-3</sub> (Hz)</th>
<th>F<sub>B</sub> (Hz)</th>
<th>S<sub>V</sub> (m<sup>2</sup>)</th>
</tr>
<tr>
<td> <input type=”text” id=”entrada5” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” name=”entrada6” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada7” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada8” disabled=”disabled” size=” 5” maxlength=”50” /></td>
</tr>
<tr>
<th>M<sub>AP</sub></th>
<th>L (m)</th>
<th>L<sub>V</sub> (m)</th>
<th>L<sub>l</sub> (m)</th>
</tr>
<tr>
<td> <input type=”text” id=”entrada9” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” name=”entrada10” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada11” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada12” disabled=”disabled” size=” 5” maxlength=”50” /></td>
```

```
</tr>
</table>
</font>
</center>
</div>
<center>
<!--<label id=”salida” />→
<!----<input type=”button” value=”Comparar” onclick=”window.open(‘nueva.html’,’nuevaVentana’,’width=300, height=400’)” />→
<a href=”#” onclick=”refresh()” class=”button yellow radius”>
<span class=”icon-cycle”></span>Borrar
</a>
</center>
</form>
<!—Cordova reference, this is added to your app when it’s built. →
<script src=”scripts/jquery-2.1.4.min.js”></script>
<script src=”cordova.js”></script>
<script src=”scripts/platformOverrides.js”></script>
<script src=”scripts/bassreflex.js”></script>
<script src=”scripts/webpd-latest.min.js”></script>
<script src=”scripts/barramenu.js”></script>
</body>
</html>
```

La interfaz quedaría así:

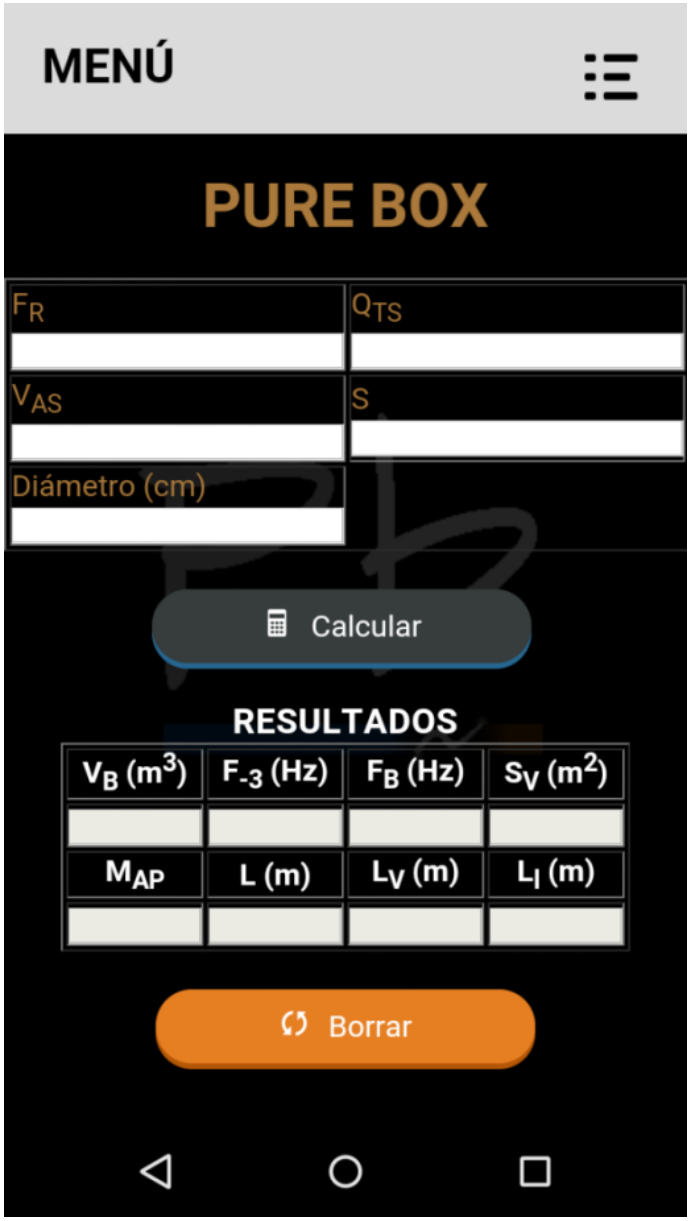

Figura 32. Interfaz de usuario para la opción de Bass Reflex

2.13.1.4. Programación para encontrar los criterios de proporciones

```
//Esta parte es la misma de las otras opciones de cálculo, porque programa el menú de la aplicación.
<!DOCTYPE html>
<html>
<head>
<meta charset=”utf-8” />
<title>Proporciones acústicas</title>
<!—PruebaCordova references →
<link href=”css/index.css” rel=”stylesheet” />
<link href=”css/boton.css” rel=”stylesheet” />
<link rel=” stylesheet” href=”fonts/style.css”>
<meta name=”viewport” content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>
</head>
<body style=”background-image:url(‘images/logopbcenter.png’);”>
//Menú
<header>
<div class=” menu_bar”>
<a href=”#” class=” bt-menu”><span class=”icon-list”></span>MENÚ</a>
</div>
<nav>
<ul>
<li><a href=”# “ onclick=”window.location.href=’menu.html’”><span class=”icon-flag”></span> Inicio</a></li>
<li><a href=”# “ onclick=”window.location.href=’index.html’”><span class=”icon-document-landscape”></span> Caja Cerrada</a></li>
<li><a href=”# “ onclick=”window.location.href=’bass.html’”><span class=”icon-notification”></span> Caja con Bass Reflex</a></li>
<li><a href=”#” onclick=”window.location.href=’criterios.html’”> <span class=”icon-progress-empty”></span>Criterios de Proporciones </a></li>
```

```
<li><a href="#" onclick=" window.location.href='genf.html'"><span class="icon-sound-mix"></span>Generación de frecuencias</a></li>
<li><a href="#" onclick=" window.location.href='ayuda.html'"><span class="icon-help-with-circle"></span>Glosario</a></li>
</ul>
</nav>
</header>
<form id="caja">
<center>
<font color="#aa793b">
<br />
<p><h1><center>PURE BOX</center></h1></p>
<div>
</font>
</center>
<div>
<center>
<font color="#fff">
<table border="1" width="200">
<tr>
<td>
<label for="number-pattern">Q<sub>TC</sub></label> //Etiqueta QTC
//Ingreso del valor de QTC realizado por el usuario.
<input type="number" id="QTC" value=" ">
</td>
<td>
<label for="number-pattern">Q<sub>TS</sub></label> //Etiqueta QTS.
//Ingreso del valor de QTS realizado por el usuario.
<input type="number" id="QTS">
</td>
</tr>
<tr>
```

```
<td>
<label for=”number-pattern”>V<sub>AS</sub></label>  //Etiqueta V_AS.
//Ingreso del valor de V_AS realizado por el usuario.
<input type=”number” id=”VAS” pattern=”[0-9]*”>
</td>
</tr>
//Encontrando los cálculos de las dimensiones con el criterio ACOUSTIC RATIO.
</table>
<table border=”1” width=”300”>
<caption align=top><h2> ACOUSTIC RATIO </h2></caption>
<br />
<br />
<tr>
<th>Altura (m)</th>          //Etiqueta Altura.
<th>Ancho (m)</th>                  //Etiqueta Ancho.
<th>Profundidad (m)</th> //Etiqueta Profundidad.
</tr>
//Deshabilitando la escritura por el usuario en las casillas de texto de los resultados.
<tr>
<td> <input type=”text” id=”entrada13” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” name=”entrada14” disabled=”disabled” size=” 5” maxlength=”50” /></td>
<td> <input type=”text” id=”entrada15” disabled=”disabled” size=” 5” maxlength=”50” /></td>
</tr>
</table>
//Llamando la función Cálculo hacienda un click en el botón Calcular.
<a href=”#” onclick=”Calculo();” class=”button blue radius”>
<span class=”icon-calculator”></span>Calcular
```

```
</a>
//Encontrando los cálculos de las dimensiones con el criterio GOLDEN RATIO.
<table border=”1” width=”300”>
<caption align=top><h2> GOLDEN RATIO </h2></caption>
<br /></h2>
<tr>
<th>Altura (m)</th>                    //Etiqueta Altura.
<th>Ancho (m)</th>                     //Etiqueta Ancho.
<th>Profundidad (m)</th>               //Etiqueta Profundidad.
//Deshabilitando la escritura por el usuario en las casillas de texto de los resultados.
</tr>
<tr>
<td> <input type=”text” id=”entrada16” disabled=”disabled” size=” 5”
maxlength=”50” /></td>
<td> <input type=”text” name=”entrada17” disabled=”disabled” size=” 5”
maxlength=”50” /></td>
<td> <input type=”text” id=”entrada18” disabled=”disabled” size=” 5”
maxlength=”50” /></td>
</tr>
</table>
</font>
</center>
</div>
<center>
//Borrando toda la información visualizada anteriormente.
<a href=”#” onclick=”refresh()” class=”button yellow radius”>
<span class=”icon-cycle”></span>Borrar
</a>
//Incluyendo librerías y archivos a la página HTML.
</center>
</form>
```

```
<script src=”scripts/jquery-2.1.4.min.js”></script>
<script src=”cordova.js”></script>
<script src=”scripts/platformOverrides.js”></script>
<script src=”scripts/barramenu.js”></script>
<script src=”scripts/index.js”></script>
<script src=”scripts/criterios.js”></script>
<script src=”scripts/webpd-latest.min.js”></script>
</body>
</html>          //Finalización de la programación.
```

La interfaz de la aplicación para esta parte quedaría así:

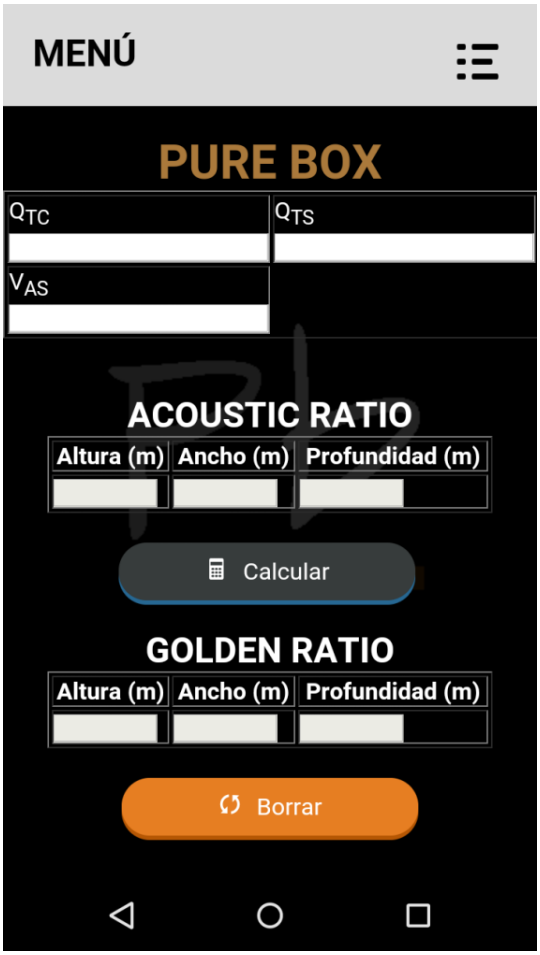

Figura 33. Interfaz de usuario, criterio de proporciones

2.13.1.5. Programación para generar frecuencias

```
//Esta parte es la misma de las otras opciones de cálculo, vuelve al menú de la aplicación.
<!DOCTYPE html>
<html lang=”en” xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<meta charset=”utf-8” />
<title> MENU INICIO</title>
<link href=”css/index.css” rel=”stylesheet” />
<link rel=” stylesheet” href=”fonts/style.css”>
<link href=”css/boton.css” rel=”stylesheet” />
<meta name=”viewport” content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>
</head>
<body style=”background-image:url(‘images/logopbcenter.png’);”>
<!-- <body style=”background-color: #DAA520”>→
//Menú.
<header>
<div class=” menu_bar”>
<a href=”#” class=” bt-menu”><span class=”icon-list”></span>MENÚ</a>
</div>
<nav>
<ul>
<li><a href=”# “ onclick=”window.location.href=’menu.html’”><span class=”icon-flag”></span> Inicio</a></li>
<li><a href=”# “ onclick=”window.location.href=’index.html’”><span class=”icon-document-landscape”></span> Caja Cerrada</a></li>
<li><a href=”# “ onclick=”window.location.href=’bass.html’”><span class=”icon-notification”></span> Caja con Bass Reflex</a></li>
<li><a href=”#” onclick=”window.location.href=’criterios.html’”> <span class=”icon-progress-empty”></span>Criterios de Proporciones </a></li>
```

```
<li><a href=”#” onclick=” window.location.href=’genf.html’”><span class=”icon-sound-mix”></span>Generación de frecuencias</a></li>
<li><a href=”#” onclick=” window.location.href=’ayuda.html’”><span class=”icon-help-with-circle”></span>Glosario</a></li>
</ul>
</nav>
</header>
<center>
<font color=”#aa793b”>
<br />
<p><h1>PURE BOX</h1></p>
</font>
</center>
<br/><br />
<div>
<center>
<font color=”#fff”>
<table border=”1” width=”200”>
<tr>
<td>
//Etiqueta FRECUENCIA.
<center><label for=”number-pattern”>FRECUENCIA</label></center>
//Ingreso del valor de la frecuencia realizado por el usuario.
<input type=”number” id=”FREC” value=” “>
</td>
</tr>
</table>
</font>
<br />
<br />
//Llamando a la función ‘Sonido’ creada en Javascript, al hacer un click sobre el botón ‘Generar Frecuencia’.
```

```
<a href="#" onclick="Sonido();" class="button blue radius">
<span class="icon-controller-play"></span>Generar Frecuencia
</a>
//Llamando a la función 'Blanco'  creada en Javascript, al hacer un click sobre el botón 'Ruido Blanco'.
<a href="#" onclick="Blanco();" class="button yellow radius">
<span class="icon-rss"></span>Ruido Blanco
</a>
//Deteniendo la generación de frecuencia al dar click en el botón 'Parar'.
<a href="#" onclick="refresh()" class="button radius">
<span class="icon-controller-stop"></span>Parar
</a>
</center>
</div>
//Incluyendo librerías y archivos en la página HTML.
<script src="scripts/jquery-2.1.4.min.js"></script>
<script src="cordova.js"></script>
<script src="scripts/platformOverrides.js"></script>
<script src="scripts/barramenu.js"></script>
<script src="scripts/index.js"></script>
<script src="scripts/criterios.js"></script>
<script src="scripts/genf.js"></script>
<script src="scripts/ruidoblanco.js"></script>
<script src="scripts/webpd-latest.min.js"></script>
</body>
</html>        //Finalización de la programación.
```

La interfaz quedaría así:

Figura 34. Interfaz de usuario, generador de frecuencias

2.13.1.6. Programación para visualizar el glosario

```
//Esta parte es la misma de las otras opciones de cálculo, vuelve al menú de la aplicación.
<!DOCTYPE html>
<html>
<head>
<meta charset=”utf-8” />
<title>AYUDA</title>
<link rel=” stylesheet” href=”fonts/style.css”>
<meta name=”viewport” content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>
<!—PruebaCordova references →
<link href=”css/index.css” rel=”stylesheet” />
<link href=”css/boton.css” rel=”stylesheet” />
</head>
<body style=”background-image:url(‘images/logopbcenter.png’);”>
//Menú.
<header>
<div class=” menu_bar”>
<a href=”#” class=” bt-menu”><span class=”icon-list”></span>MENÚ</a>
</div>
<nav>
<ul>
<li><a href=”# “ onclick=”window.location.href=’menu.html’”><span class=”icon-flag”></span> Inicio</a></li>
<li><a href=”# “ onclick=”window.location.href=’index.html’”><span class=”icon-document-landscape”></span> Caja Cerrada</a></li>
<li><a href=”# “ onclick=”window.location.href=’bass.html’”><span class=”icon-notification”></span> Caja con Bass Reflex</a></li>
<li><a href=”#” onclick=”window.location.href=’criterios.html’”> <span class=”icon-progress-empty”></span>Criterios de Proporciones Acústicas </a></li>
```

Generación de frecuencias
Glosario

</nav>
</header>
<section>

//Escribiendo el título de la página.
<p>
<h2>
Parámetros Thiele-Small de una 71ajá acústica.
</br>
</h2>
<h5>
//Explicación de cada parámetro *Thiele-Small* como ayuda para el usuario.
F_S(F_R) = Frecuencia de resonancia del altavoz al aire libre.
</br>
Q_{TC} = Coeficiente de sobretensión del sistema en la frecuencia de resonancia de la caja.</br></br>
Q_{TS} = Coeficiente de sobretensión total del altavoz.</br></br>
F₋₃ = Frecuencia de corte en -3dB.</br></br>
Fc = Frecuencia de resonancia.</br></br>
V_{AB} = Volumen de aire equivalente a la elasticidad acústica del aire de la caja.</br></br>
V_B = Volumen de la caja.</br></br>
V_{AS} = Volumen de aire equivalente a la elasticidad de la suspensión del
altavoz.</br></br>
F_B = Frecuencia de resonancia de la caja.</br></br>
S = Coeficiente de sobretensión de la caja.</br></br>
M_{AP} = Masa acústica del respiradero.</br></br>

L = Longitud del respiradero.</br></br>
S_V = Superficie del respiradero.</br></br>
L_V = Medida de corrección de longitud del respiradero.</br></br>
L_l = Longitud definitiva del respiradero.</br>
</h5>
</p>

</section>
<script src=”scripts/jquery-2.1.4.min.js”></script>
<script src=”cordova.js”></script>
<script src=”scripts/platformOverrides.js”></script>
<script src=”scripts/bassreflex.js”></script>
<script src=”scripts/webpd-latest.min.js”></script>
<script src=”scripts/barramenu.js”></script>
</body>
</html>

La interfaz quedaría de la siguiente manera:

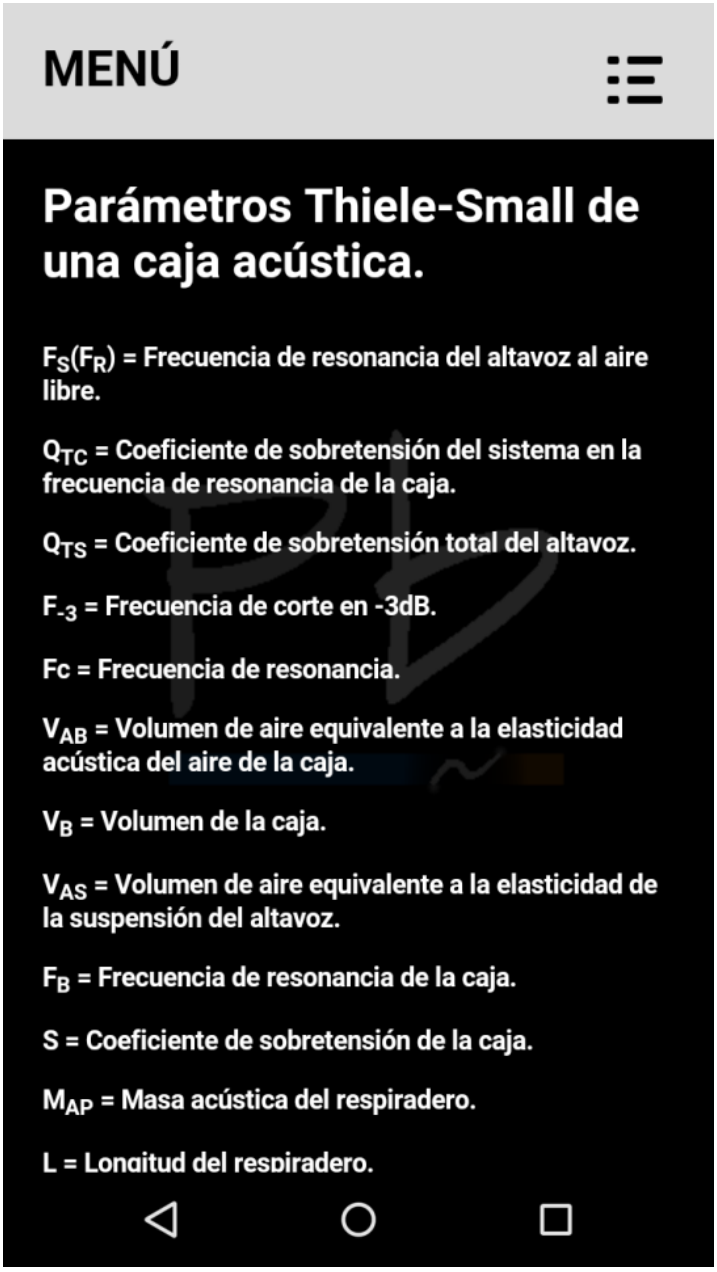

Figura 35. Interfaz de usuario, glosario

2.13.2. CSS3, aplicando estilos visuales para páginas web

Como se explicaba anteriormente en la programación *HTML* se requería incluir estilos de letras y botones, para lo cual se necesitaba una programación externa llamada *CSS*.
La versión *CSS3* es la última versión del lenguaje y actualizada con orientación más hacia los dispositivos móviles.

Se han utilizado dos archivos de este código los cuales son, index.css y botón.css. El primero se encarga de la interfaz principal de la aplicación y el segundo de los detalles gráficos de los botones.

A continuación se va a incluir el código utilizado en el archivo index.css.

```
body {
background-color: #000; // Color de fondo de pantalla.
}
//Imagen establecida como fondo de pantalla.
Body{
background:url(‘images/PUREBOX.png’) no-repeat fixed center center;
}
//Estableciendo márgenes.
* {
margin:0;
padding:0;
}
body {
background:#000;
}
//Estableciendo ancho del encabezado.
Header {
width:100%;
}
```

```
//Márgenes del menú.
Header nav {
width:90%;
max-width:800px;
margin:20px auto;
background:#000;
}
.menu_bar {
display:none;
}
header nav ul {
overflow:hidden;
list-style:none;
}
//Las siguientes clases son para definir la configuración de  los ítems del menú.
header nav ul li {
float:left;
}
header nav ul li a {
color:#fff;
padding:20px;
display:block;
text-decoration:none;
}
header nav ul li span {
margin-right:10px;
}
header nav ul li a:hover {
background:#808080;
}
section {
padding:20px;
```

```
}
@media screen and (max-width:800px ) {
header nav {
width:80%;
height:100%;
left:-100%;
margin:0;
position: fixed;
}
//Letras del menú
header nav ul li {
display:block;
float:none;
border-bottom:1px solid rgba(255,255,255, .3);
}
//Menú opción de color a escoger
.menu_bar {
display:block;
width:100%;
background:#ccc;
}
//Barra de 76édi
.menu_bar .bt-menu {
display:block;
padding:20px;
background:rgba(255,255,255, .3);
color:#000;
text-decoration:none;
font-weight: bold;
font-size:25px;
-webkit-box-sizing:border-box;
-moz-box-sizing:border-box;
```

```
box-sizing:border-box;
}
.menu_bar span {
float:right;
font-size:40px;
}
}
```

La siguiente sección de programación es del botón.css, que se encarga de la configuración de los botones de las páginas hechas en *HTML*.

```
*{
margin:0;
padding:0;
}
//Estilos Generales
.button {
background:#7f8c8d;
color:#fff;
display:inline-block;
font-size:1em;
margin:20px;
padding:10px 0;
text-align:center;
width:200px;
text-decoration:none;
box-shadow:0px 3px 0px #373c3c;
}
.button span {
margin-right:10px;
}
//Configuración de los botones con colores personalizados.
```

```
.button.blue {
background:#373c3c;
box-shadow:0px 3px 0px #266792;
}
.button.yellow {
background:#e67e22;
box-shadow:0px 3px 0px #b55704;
}
//Tamaños de los botones.
.button.medium {
width:350px;
}
.button.large {
width:450px;
}
.button.radius {
border-radius:50px;
}
//Efectos, para los botones.
.button:hover {
box-shadow:0px 0px 0px;
padding-top:7px;
}
```

2.13.3. JavaScript, lenguaje orientado a objetos

El lenguaje de programación Javascript es el encargado de realizar los cálculos matemáticos junto con Pure Data dentro en este proyecto. A continuación se explicará la programación utilizada para la aplicación.

2.13.3.1. Programación del menú

```
// Programación de la animación del menú.

$(document).ready(main);

var 79ontador = 1;

function main() {

$(‘.menu_bar’).click(function () {

// $(‘nav’).toggle();

if (79ontador == 1) {

$(‘nav’).animate({

left: ‘0’

});

contador = 0;

} else {

contador = 1;

$(‘nav’).animate({

left: ‘-100%’
```

2.13.3.2. Programación para una caja acústica cerrada

```
// Funciones incluídas por defecto.
(function () {
“use strict”;
document.addEventListener(‘deviceready’, onDeviceReady.bind(this), false);
function onDeviceReady() {
// Controlar la pausa de Cordova y reanudar eventos
document.addEventListener(‘pause’, onPause.bind(this), false);
document.addEventListener(‘resume’, onResume.bind(this), false);
// TODO: Cordova se ha cargado. Haga aquí las inicializaciones que necesiten Cordova.
};
function onPause() {
// TODO: esta aplicación se ha suspendido. Guarde el estado de la aplicación aquí.
};
function onResume() {
// TODO: esta aplicación se ha reactivado. Restaure el estado de la aplicación aquí.
};
})();
function refresh() {
window.location.reload();
};
// Inicio de la función cálculo.
Function Calculo() {
// Se crea una variable con el nombre [patch] y se carga el Patch de Pure Data.
Var patch
$.get(‘./pd/frecuenciacortecopia.pd’, function (mainStr)
// Se obtiene el archivo frecuenciacortecopia.pd.
{
patch = Pd.loadPatch(mainStr);
```

```
// Se crea la variable [qtc] y se asigna el valor ingresado.
Var qtc = parseFloat($(“#QTC”).val());
// Validación de los datos ingresados.
While ((qtc > 2 || qtc <= 0)) {
alert(“Ingrese valores de QTC mayores a 0 y menores a 2”) // Mensaje desplegado.
Break;
}
if (qtc <= 2 && qtc > 0) {
// Se crea la variable [qts] y se asigna el valor ingresado
var qts = parseFloat($(“#QTS”).val());
// Validación de los datos ingresados.
While ((qts >= 1 || qts <= 0)) {
alert(“Ingrese valores de QTS mayores a 0 y menores a 1”); // Mensaje desplegado.
Break;
}
// Condicional de la variable [qts].
If (qts < 1 && qts > 0) {
// Se crea la variable [vas] y se asigna el valor ingresado
var vas = parseFloat($(“#VAS”).val());
while (vas <= 0) {
alert(“Ingrese valores de VAS mayores a 0”);   // Mensaje desplegado.
Break;
}
//alert($(“#QTC”).val());  //Mensaje de alerta de control.
If (vas > 0) {
// Se crea la variable [fr] y se asigna el valor ingresado si cumple el condicional If.
Var fr = parseFloat($(“#FR”).val());
while (fr <= 0) {
alert(“Ingrese valores de FR mayores a 0”);     // Mensaje desplegado.
```

```
Break;
}
if (fr > 0) {
// Se crea la variable [s] y se asigna el valor ingresado si cumple el condicional If.
Var s = parseFloat($(“#S”).val());
// Se crea la variable [dia] que representa al valor del diámetro, si cumple el condicional If.
Var dia = parseFloat($(“#DIA”).val());
//Se crean las constantes con sus valores respectivos, para ser utilizados en los cálculos de las ecuaciones antes mencionadas, para una caja acústica cerrada.
var c13 = 0.13;
var c1 = 1;
var c2 = 2;
var c12 = 1.2;
var c1000 = 1000;
var c140449 = 140449;
var pi = 3.14;
var c10000 = 10000;
var c18 = 1.18;
var c4 = 4;
}
}
}
}
//Se envían los datos ingresados a Pure Data con sus respectivos nombres de las variables.
Pd.send(patch.patchId + ‘-qtc’, [qtc]);
Pd.send(patch.patchId + ‘-fr’, [fr]);
Pd.send(patch.patchId + ‘-qts’, [qts]);
Pd.send(patch.patchId + ‘-vas’, [vas]);
Pd.send(patch.patchId + ‘-s’, [s]);
```

```
Pd.send(patch.patchId + '-dia', [dia]);
Pd.send(patch.patchId + '-c1', [c1]);
Pd.send(patch.patchId + '-c2', [c2]);
Pd.send(patch.patchId + '-c12', [c12]);
Pd.send(patch.patchId + '-c4', [c4]);
Pd.send(patch.patchId + '-c13', [c13]);
Pd.send(patch.patchId + '-c18', [c18]);
Pd.send(patch.patchId + '-c1000', [c1000]);
Pd.send(patch.patchId + '-pi', [pi]);
Pd.send(patch.patchId + '-c10000', [c10000]);
Pd.send(patch.patchId + '-c140449', [c140449]);
//Se reciben los resultados desde Pure Data
Pd.receive(patch.patchId + '-fc', function (res) {
$("#entrada1").val(res);
});
Pd.receive(patch.patchId + '-vb1', function (res) {
$("#entrada3").val(res);
});
Pd.receive(patch.patchId + '-vb', function (res) {
$("#entrada4").val(res);
});
//Se vuelven a enviar a Pure Data los valores de QTC y VAS, la razón se la explicará al final del proyecto, en las conclusiones.
Pd.send(patch.patchId + '-qtc', [qtc]);
Pd.send(patch.patchId + '-vas', [vas]);
//El valor que se visualiza como resultado en la casilla de texto FC se lo asigna a la variable [fc].
Var fc = document.getElementById("entrada1").value;
var A = (1 / (qtc) * (qtc)) – 2;
var B = (A * A) + 4;
//[Math.sqrt()] es una función definida por defecto en Javascript, se la utiliza para calcular la raíz cuadrada de un número.
```

```
Var C = Math.sqrt(B);
var D = (A + C) / 2;
var E = Math.sqrt(D);
var F3 = fc * E;
84ajá.entrada2.value = F3.toFixed(2);
//Se recibe el valor de la variable [v] enviada desde Pure Data y se asigna su valor a la entrada 5 declarada en HTML.
Pd.receive(patch.patchId + '-v', function (res) {
$("#entrada5").val(res);
});
//Se envía el valor de la variable [qts] a Pure Data nuevamente.
Pd.send(patch.patchId + '-qts', [qts]);
//El valor desplegado en la casilla de resultado de [v] se lo asigna a la variable [v1].
Var v1 = document.getElementById("entrada5").value;
var F = fr * fr;
var G = F * vas;
var H = G / v1;
var F3i = Math.sqrt(H);
caja.entrada6.value = F3i.toFixed(2);
var fbass = F3i;
//Se reciben los valores de las variables [fb] y [sv] enviadas desde Pure Data.
Pd.receive(patch.patchId + '-fb', function (res) {
$("#entrada7").val(res);
});
Pd.receive(patch.patchId + '-sv', function (res) {
$("#entrada8").val(res);
});
//Se envían los datos ingresados a Pure Data con sus respectivos nombres de las variables.
Pd.send(patch.patchId + '-vas', [vas]);
Pd.send(patch.patchId + '-dia', [dia]);
```

```
Pd.send(patch.patchId + '-c2', [c2]);
Pd.send(patch.patchId + '-pi', [pi]);
Pd.send(patch.patchId + '-c1000', [c10000]);
Pd.send(patch.patchId + '-c13', [c13]);
var cab = v1 / c140449;
Pd.send(patch.patchId + '-cab', [cab]);
//Se recibe el valor de la variable [map].
//Pd.receive(patch.patchId + '-map', function (res) {
//    $("#entrada9").val(res);
//});
//Se recibe el valor de la variable [L].
Pd.receive(patch.patchId + '-L', function (res) {
$("#entrada10").val(res);
});
var sv = document.getElementById("entrada8").value;
var LV = 0.82 * Math.sqrt(sv);
Pd.send(patch.patchId + '-LV', [LV]);
caja.entrada11.value = LV.toFixed(2);
//Se recibe el valor de la variable [L1].
Pd.receive(patch.patchId + '-Ll', function (res) {
$("#entrada12").val(res);
});
// var v = document.getElementById("entrada5").value;
//Se envían los datos ingresados a Pure Data con sus respectivos nombres de las variables.
Pd.send(patch.patchId + '-qtc', [qtc]);
Pd.send(patch.patchId + '-vas', [vas]);
Pd.send(patch.patchId + '-qts', [qts]);
Pd.send(patch.patchId + '-fbass', [fbass]);
Pd.send(patch.patchId + '-fb1', [fb1]);
Pd.send(patch.patchId + '-c1', [c1]);
Pd.send(patch.patchId + '-qtc', [qtc]);
```

```
Pd.send(patch.patchId + ‘-fr’, [fr]);
Pd.send(patch.patchId + ‘-qts’, [qts]);
Pd.send(patch.patchId + ‘-vas’, [vas]);
Pd.send(patch.patchId + ‘-s’, [s]);
Pd.send(patch.patchId + ‘-dia’, [dia]);
Pd.send(patch.patchId + ‘-c2’, [c2]);
Pd.send(patch.patchId + ‘-c12’, [c12]);
Pd.send(patch.patchId + ‘-c4’, [c4]);
Pd.send(patch.patchId + ‘-c13’, [c13]);
Pd.send(patch.patchId + ‘-c18’, [c18]);
Pd.send(patch.patchId + ‘-c1000’, [c1000]);
Pd.send(patch.patchId + ‘-c10000’, [c10000]);
Pd.send(patch.patchId + ‘-c140449’, [c140449]);
Pd.send(patch.patchId + ‘-c1’, [c1]);
Pd.send(patch.patchId + ‘-pi’, [pi]);
Pd.send(patch.patchId + ‘-c4’, [c4]);
Pd.send(patch.patchId + ‘-cab’, [cab]);
Pd.send(patch.patchId + ‘-c1’, [c1]);
var fb = document.getElementById(“entrada7”).value;
var fb1 = fb;
var map= 1/((fb1*fb1)*cab*4*(3.14*3.14));
caja.entrada9.value = map.toFixed(2);
var L = (map * sv) / c18;
caja.entrada10.value = L.toFixed(2);
var Ll = L – LV;
caja.entrada12.value = Ll.toFixed(2);
})
}
```

2.13.3.3. Programación para calcular una caja acústica con *Bass Reflex*

```
(function () {
“use strict”;
document.addEventListener(‘deviceready’, onDeviceReady.bind(this), false);
function onDeviceReady() {
// Controlar la pausa de Cordova y reanudar eventos
document.addEventListener(‘pause’, onPause.bind(this), false);
document.addEventListener(‘resume’, onResume.bind(this), false);
// TODO: Cordova se ha cargado. Haga aquí las inicializaciones que necesiten Cordova.
};
function onPause() {
// TODO: esta aplicación se ha suspendido. Guarde el estado de la aplicación aquí.
};
function onResume() {
// TODO: esta aplicación se ha reactivado. Restaure el estado de la aplicación aquí.
};
})();
function refresh() {
window.location.reload();
};
//El código es muy similar al anterior explicado, sólo varían los nombres de las variables y los cálculos matemáticos.
Function Calculo() {
var patch
$.get(‘./pd/frecuenciacortecopia.pd’, function (mainStr) {
patch = Pd.loadPatch(mainStr);
var fr = parseFloat($(“#FR”).val());
while (fr <= 0 ) {
alert(“Ingrese valores de FR mayores a 0”)
```

```
break;
}
if (fr > 0 ){
var qts = parseFloat($(“#QTS”).val());
while ((qts >= 1 || qts <= 0)) {
alert(“Ingrese valores de QTS mayores a 0 y menores a 1”);
break;
}
if (qts < 1 && qts > 0) {
var vas = parseFloat($(“#VAS”).val());
while (vas <= 0) {
alert(“Ingrese valores de VAS 88ngres a 0”);
break;
}
//alert($(“#QTC”).val());
if (vas > 0) {
var s = parseFloat($(“#S”).val());
while (s <= 0) {
alert(“Ingrese valores de S mayores a 0”);
break;
}
if (s > 0) {
var dia = parseFloat($(“#DIA”).val());
while( dia <= 0){
alert(“Ingrese valores del 88ngressa mayores a 0”);
break;
}
if (dia > 0) {
var c13 = 0.13;
var c1 = 1;
var c2 = 2;
var c12 = 1.2;
```

```
var c1000 = 1000;
var c140449 = 140449;
var pi = 3.14;
var c10000 = 10000;
var c18 = 1.18;
var c4 = 4;
}
}
}
}
}
//Envía los datos 89ngressados a Pure Data.
Pd.send(patch.patchId + '-fr', [fr]);
Pd.send(patch.patchId + '-qts', [qts]);
Pd.send(patch.patchId + '-vas', [vas]);
Pd.send(patch.patchId + '-s', [s]);
Pd.send(patch.patchId + '-dia', [dia]);
Pd.send(patch.patchId + '-c1', [c1]);
Pd.send(patch.patchId + '-c2', [c2]);
Pd.send(patch.patchId + '-c12', [c12]);
Pd.send(patch.patchId + '-c4', [c4]);
Pd.send(patch.patchId + '-c13', [c13]);
Pd.send(patch.patchId + '-c18', [c18]);
Pd.send(patch.patchId + '-c1000', [c1000]);
Pd.send(patch.patchId + '-pi', [pi]);
Pd.send(patch.patchId + '-c10000', [c10000]);
Pd.send(patch.patchId + '-c140449', [c140449]);

// Recibe los resultados desde Pure Data
Pd.send(patch.patchId + '-vas', [vas]);
Pd.receive(patch.patchId + '-v', function (res) {
$("#entrada5").val(res);
```

```
});
Pd.send(patch.patchId + '-qts', [qts]);
var v1 = document.getElementById("entrada5").value;
var F = fr * fr;
var G = F * vas;
var H = G / v1;
var F3i = Math.sqrt(H);
caja.entrada6.value = F3i.toFixed(2);
var fbass = F3i;
Pd.receive(patch.patchId + '-fb', function (res) {
$("#entrada7").val(res);
});
Pd.receive(patch.patchId + '-sv', function (res) {
$("#entrada8").val(res);
});
//Envía los datos ingresados a Pure Data.
Pd.send(patch.patchId + '-vas', [vas]);
Pd.send(patch.patchId + '-dia', [dia]);
Pd.send(patch.patchId + '-c2', [c2]);
Pd.send(patch.patchId + '-pi', [pi]);
Pd.send(patch.patchId + '-c1000', [c10000]);
Pd.send(patch.patchId + '-c13', [c13]);
var cab = v1 / c140449;
Pd.send(patch.patchId + '-cab', [cab]);
Pd.receive(patch.patchId + '-L', function (res) {
$("#entrada10").val(res);
});
var sv = document.getElementById("entrada8").value;
var LV = 0.82 * Math.sqrt(sv);
Pd.send(patch.patchId + '-LV', [LV]);
caja.entrada11.value = LV.toFixed(2);
Pd.receive(patch.patchId + '-Ll', function (res) {
```

```
$("#entrada12").val(res);
});
// var v = document.getElementById("entrada5").value;
//Envía los datos ingresados a Pure Data.
Pd.send(patch.patchId + '-vas', [vas]);
Pd.send(patch.patchId + '-qts', [qts]);
Pd.send(patch.patchId + '-fbass', [fbass]);
Pd.send(patch.patchId + '-fb1', [fb1]);
Pd.send(patch.patchId + '-c1', [c1]);
Pd.send(patch.patchId + '-fr', [fr]);
Pd.send(patch.patchId + '-qts', [qts]);
Pd.send(patch.patchId + '-vas', [vas]);
Pd.send(patch.patchId + '-s', [s]);
Pd.send(patch.patchId + '-dia', [dia]);
Pd.send(patch.patchId + '-c2', [c2]);
Pd.send(patch.patchId + '-c12', [c12]);
Pd.send(patch.patchId + '-c4', [c4]);
Pd.send(patch.patchId + '-c13', [c13]);
Pd.send(patch.patchId + '-c18', [c18]);
Pd.send(patch.patchId + '-c1000', [c1000]);
Pd.send(patch.patchId + '-c10000', [c10000]);
Pd.send(patch.patchId + '-c140449', [c140449]);
Pd.send(patch.patchId + '-c1', [c1]);
Pd.send(patch.patchId + '-pi', [pi]);
Pd.send(patch.patchId + '-c4', [c4]);
Pd.send(patch.patchId + '-cab', [cab]);
Pd.send(patch.patchId + '-c1', [c1]);
var fb = document.getElementById("entrada7").value;
var fb1 = fb;
var map = 1 / ((fb1 * fb1) * cab * 4 * (3.14 * 3.14));
caja.entrada9.value = map.toFixed(2);
var L = (map * sv) / c18;
```

```
caja.entrada10.value = L.toFixed(2);
var Ll = L – LV;
caja.entrada12.value = Ll.toFixed(2);
})
}
```

2.13.3.4. Programación para encontrar los criterios de proporciones

```
//Para este proceso no se utilizó Pure Data.
(function () {
“use strict”;
document.addEventListener(‘deviceready’, onDeviceReady.bind(this), false);
function onDeviceReady() {
// Controlar la pausa de Cordova y reanudar eventos
document.addEventListener(‘pause’, onPause.bind(this), false);
document.addEventListener(‘resume’, onResume.bind(this), false);
// TODO: Cordova se ha cargado. Haga aquí las inicializaciones que necesiten Cordova.
};
function onPause() {
// TODO: esta aplicación se ha suspendido. Guarde el estado de la aplicación aquí.
};
function onResume() {
// TODO: esta aplicación se ha reactivado. Restaure el estado de la aplicación aquí.
};
})();
function refresh() {
window.location.reload();
};
//Función declarada para realizar la 92aíz cuadrada de cualquier radical.
```

```
Function raizN(x, n) {
return Math.exp(Math.log(x) / n);
}
//Inicio de la función 'Calculo'.
Function Calculo() {
var qtc = parseFloat($("#QTC").val());   //Almacenando el valor ingresado de QTC.
While ((qtc > 2 || qtc <= 0)) {
alert("Ingrese valores de QTC mayores a 0 y menores a 2") // Mensaje desplegado.
Break;
}
if (qtc <= 2 && qtc > 0) {    //Condicional If
var qts = parseFloat($("#QTS").val());//Almacenando el valor ingresado de QTS.
While ((qts >= 1 || qts <= 0)) {
alert("Ingrese valores de QTS mayores a 0 y menores a 1"); // Mensaje desplegado.
Break;
}
if (qts < 1 && qts > 0) {
var vas = parseFloat($("#VAS").val()); //Almacenando el valor ingresado de VAS.
While (vas <= 0) {
alert("Ingrese valores de VAS mayores a 0"); // Mensaje desplegado.
Break;
}
if (vas > 0) {
var div = ((qtc / qts) * (qtc / qts)) – 1;
var vb = vas / div;
var 93aíz = raizN(vb, 3); // Calculando la raíz cúbica de la variable [vb].
Var altura = (93aíz) * 1.618;
var profundidad = (93aíz) * 0.618;
```

```
// La expresión to.Fixed() es para dejar la respuesta al número de decimales deseados.
Caja.entrada13.value = altura.toFixed(2);
caja.entrada14.value = 94aíz.toFixed(2);
caja.entrada15.value = profundidad.toFixed(2);
var altura1 = (94aíz) * 1.2599;
var prof = (94aíz) * 0.7937;
caja.entrada16.value = altura1.toFixed(2);
caja.entrada17.value = 94aíz.toFixed(2);
caja.entrada18.value = prof.toFixed(2);
}
}
}
}
```

2.13.3.5. Programación para generar una frecuencia

```
(function () {
“use strict”;
document.addEventListener(‘deviceready’, onDeviceReady.bind(this), false);
function onDeviceReady() {
// Controlar la pausa de Cordova y reanudar eventos
document.addEventListener(‘pause’, onPause.bind(this), false);
document.addEventListener(‘resume’, onResume.bind(this), false);
// TODO: Cordova se ha cargado. Haga aquí las inicializaciones que necesiten Cordova.
};
function onPause() {
// TODO: esta aplicación se ha suspendido. Guarde el estado de la aplicación aquí.
};
```

```
function onResume() {
// TODO: esta aplicación se ha reactivado. Restaure el estado de la aplicación aquí.
};
})();
var cont = 0;
function refresh() {
window.location.reload();
cont = 0;
};
//Inicio de la función Sonido.
Function Sonido() {
cont = cont + 1; // Creando un Contador para ser utilizado como control de entrada de datos.
If (cont == 1) {
var patch
$.get('./pd/sound.pd', function (mainStr) { // Cargando el 95rchive de Pure Data 'sound.pd'.
patch = Pd.loadPatch(mainStr); //Asignando el Patch de Pure Data a la variable [patch].
//Almacenando el valor de la frecuencia ingresada por el usuario a la variable [frec].
Var frec = parseFloat($("#FREC").val());
while (frec < 20 || frec > 20000) {
alert("Frecuencia fuera de rango auditivo"); //Mensaje desplegado.
Break;
}
if (frec >= 20 && frec <= 20000) {
// Si cumple con la condición el valor de la frecuencia ingresado por el usuario, se envía para Pure Data para su cálculo.
Pd.send(patch.patchId + '-sound', [frec]);
Pd.start();
```

```
} else {
alert(“Ingrese frecuencia a generar”); // Caso contrario se despliega un mensaje.
Refresh();
}
})
cont = cont + 1;
};
```

2.13.3.6. Código fuente de la función para generar ruido blanco.

```
(function () {
“use strict”;
document.addEventListener(‘deviceready’, onDeviceReady.bind(this), false);
function onDeviceReady() {
document.addEventListener(‘pause’, onPause.bind(this), false);
document.addEventListener(‘resume’, onResume.bind(this), false);
};
function onPause() {
// TODO: esta aplicación se ha suspendido. Guarde el estado de la aplicación aquí.
};
function onResume() {
};
})();
function refresh() {
window.location.reload();
};
function Blanco() {
var patch
$.get(‘./pd/ruidoblanco.pd’, function (mainStr){
patch = Pd.loadPatch(mainStr); // Se asigna el Patch a la variable [patch].
```

```
Pd.start(); // Se da la orden de empezar Pure Data con el Toggle.
})
```

2.14. Resultados de la aplicación en el emulador de *Apache Cordova*

Una vez detallada la programación en todos los tipos de lenguajes utilizados, se prueba la aplicación con un emulador para dispositivos con sistema operativo *Android*.

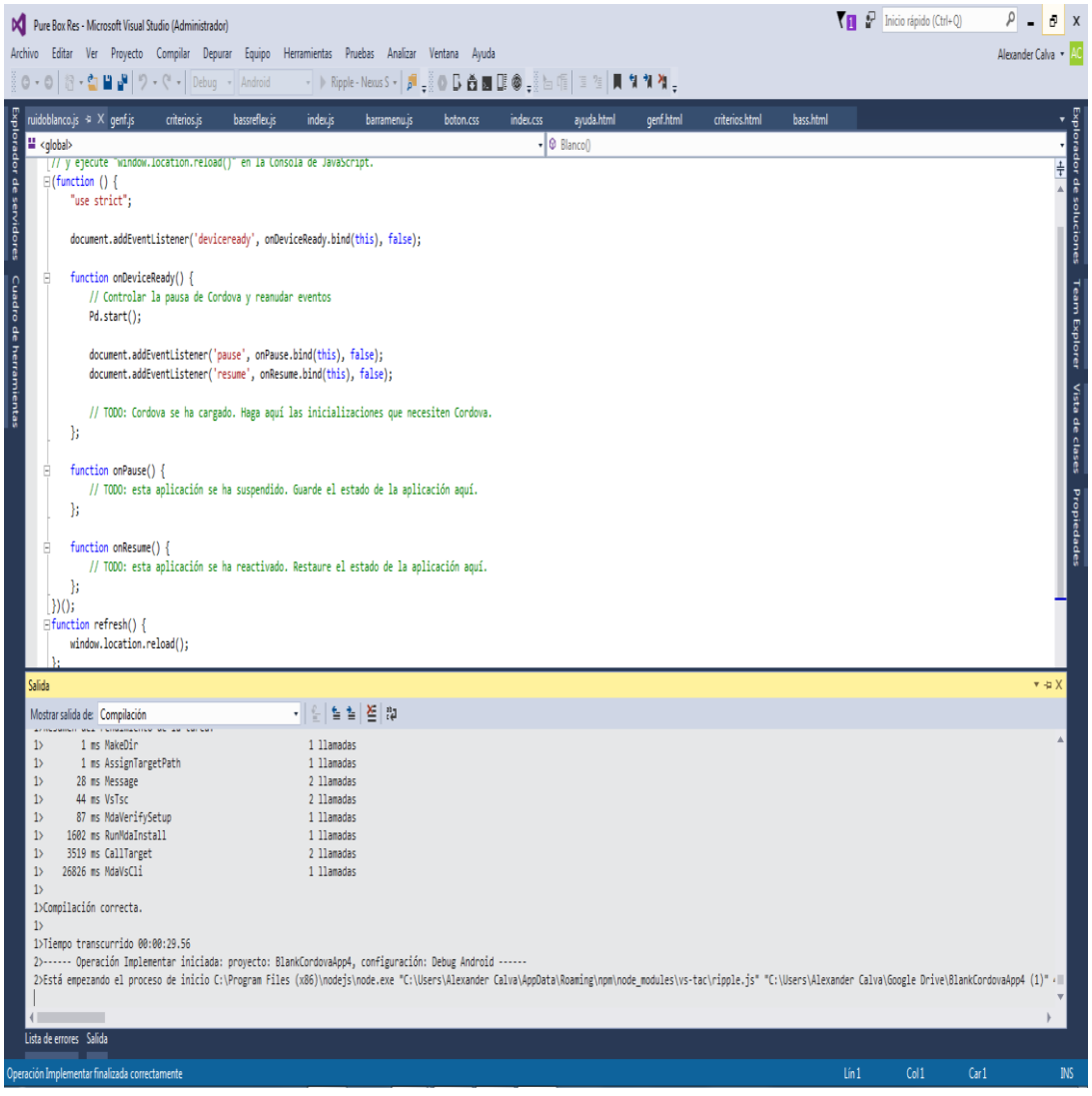

Figura 36. Compilación del programa para aplicar el emulador

Para poder realizar esta simulación se debe compilar el código y comprobar que no existan errores. A continuación se visualizará la imagen del uso de un emulador para *Android* en *Apache Cordova.* Compilación realizada con éxito y ahora se despliega esta ventana.

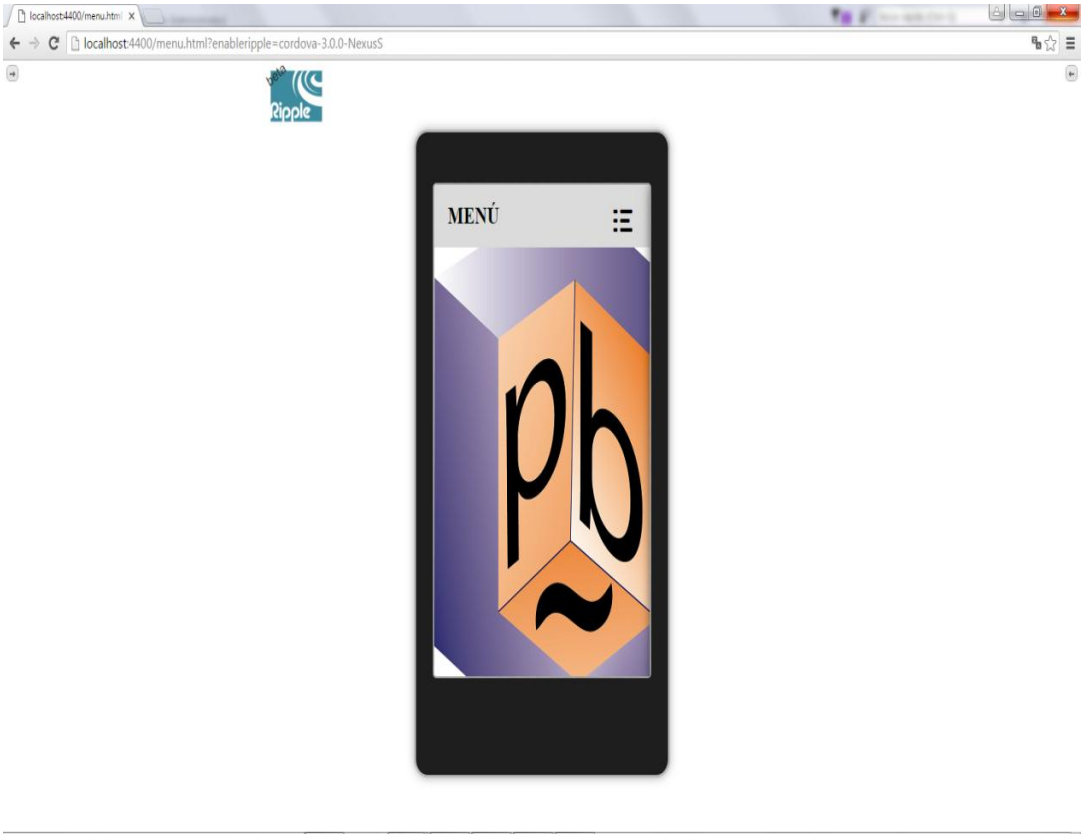

Figura 37. Emulador de Android

2.15. Objetos de *Pure Data* compatibles con *WebPd*

Los objetos compatibles son los siguientes:

- %
- **~
- +
- +~
- -
- -~
- /
- abs
- array
- atan
- bng
- bp~
- change
- clip~
- cos
- dac~
- del
- delay
- delread~
- delwrite~
- exp
- f
- float
- floatatom
- hip~
- hradio
- hsl
- i
- inlet

- inlet~
- int
- line~
- loadbang
- log
- lop~
- metro
- mod
- moses
- msg
- mtof
- nbx
- noise~
- osc~
- outlet
- outlet~
- pack
- pd
- phasor~
- pow
- print
- r
- random
- receive
- s
- sqrt
- samplerate~
- sel
- select
- send
- sig~

- sin
- soundfiler
- spigot
- square~
- symbolatom
- t
- table
- tabread4~
- tabread~
- tan
- text
- tgl
- timer
- triangle~
- trigger
- until
- vcf~
- vd~
- vradio
- vsl
- vu

2.16. Análisis de resultados

Para poder demostrar el correcto funcionamiento de la aplicación realizada, se va a tomar como ejemplo un ejercicio resuelto en internet, donde para calcular los parámetros Thiele Small de una caja cerrada los datos son los siguientes:

QTC = 1 FR = 19 QTS = 0.32

VAS = 0.54

Figura 38. Comprobación de resultados para una caja acústica cerrada

Los resultados se visualizan en la aplicación y concuerdan con los datos reales del ejercicio, los cuales son:

FC = 59.37 Hz F-3 = 46.68 Hz VAB = 61.60 L

VB = 51.34 L

Los datos para calcular los parámetros de una caja con Bass Reflex con un diámetro de 10 cm, serían:

S = 5.7 FR = 33 Hz. QTS = 0.37 VAS = 0.057 m^3

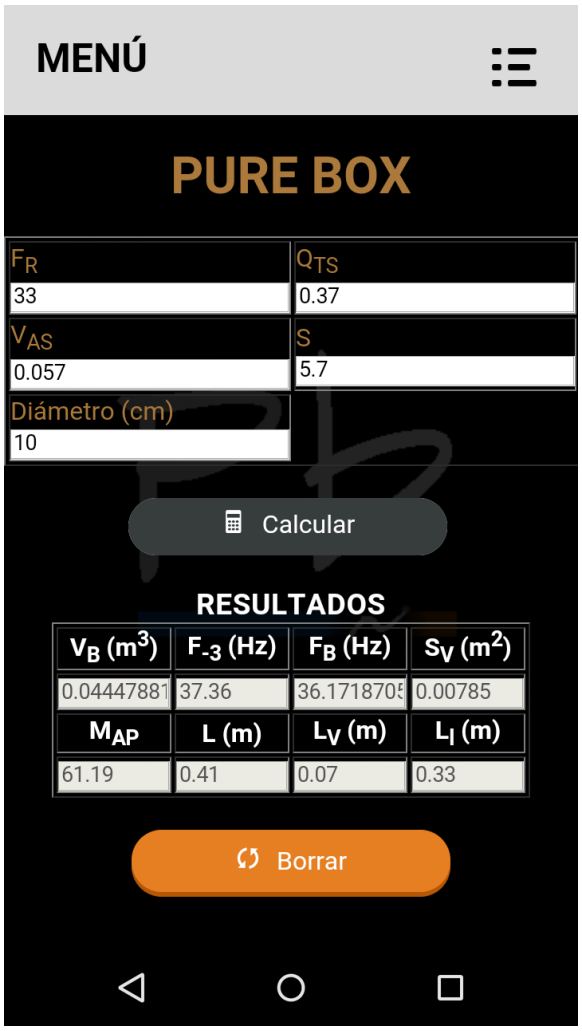

Figura 39. Comprobación, cajá acústica con Bass Reflex

Las respuestas en la aplicación concuerdan con las del ejercicio planteado.

VB = 0.044 L. F-3 = 37.36 Hz. FB = 36.17 Hz. SV = 0.008 m^2 MAP = 61.19

L = 0.41 m. LV = 0.07 m. LI = 0.33 m.

Y para calcular la validación de la aplicación, se calculará los criterios de proporciones para un recinto acústico. A continuación los datos a ingresar serían:

QTC = 0.7 QTS = 0.6

VAS = 0.54

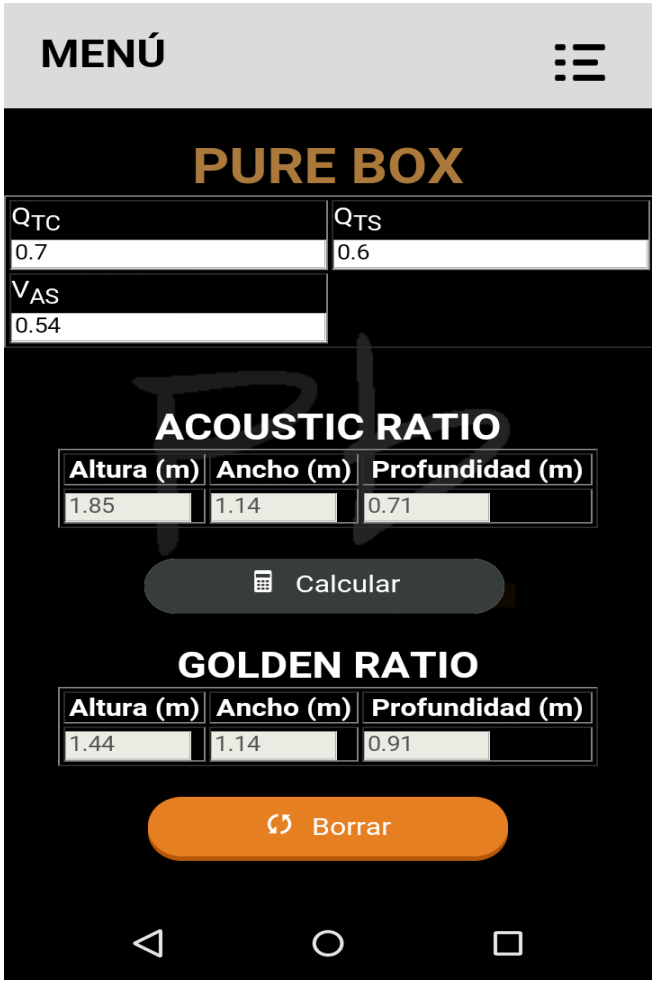

Figura 40. Comprobación, criterio de proporciones

Los resultados serían:

Acoustic Ratio

Ancho = 1.14 m. Altura = 1.85 m. Profundidad = 0.71 m.

Golden Ratio

Ancho = 1.14 m. Altura = 1.44 m. Profundidad = 0.91 m.

El siguiente ítem, el generador de frecuencia no puede validarse por escrito, por ser de uso auditivo.

2.17. Incompatibilidad con versiones del sistema operativo *Android*

Para realizar este proyecto basado en el sistema operativo Android, se necesitó un dispositivo móvil que su funcionamiento se base en este mismo Software. Al ejecutar la aplicación en el Smartphone se notó que no todas las versiones de este sistema operativo eran compatibles con el Apache Cordova. El proyecto se realizó en la versión de Visual Studio 2015 por lo cual, se requería trabajar con la versión de Android 5.0. Se aclara que sólo para la realización de la aplicación y su comprobación se necesitaba esta versión de sistema operativo. Una vez finalizada la App y subida a la tienda electrónica de Play Store, podría ser compatible con versiones anteriores a la 5.0.

2.18. Configuración del dispositivo móvil para ejecutar la aplicación

La configuración del Smartphone para ejecutar la aplicación es muy sencilla, depende de la marca del dispositivo para seguir el mismo orden de pasos.

En este caso el dispositivo móvil que se utilizó es el Motorola Moto G (Segunda generación) con sistema operativo *Android* 5.0.2 y para realizar la conexión entre el ordenador y el *Smartphone*, se necesitó instalar el *Driver USB* de Motorola que se encuentra en internet y además el SDK Studio que se explicará más adelante. (Piqué, 2013)

Lo primero que se debe hacer es ingresar a la configuración general del dispositivo, escoger la opción que dice “Acerca del teléfono” luego buscar la opción que dice “Versión de Android” y darle cinco *clicks* encima hasta que se visualice un mensaje diciendo que se ha habilitado la opción para desarrollador.

Una vez realizado estos pasos, se regresa hacia atrás a configuración general y se va a desplegar ya la opción que dice “Programador”, donde se hace un click y se activa este modo. Ya una vez dentro del modo programador se busca la opción que dice “Depuración por *USB*” y se la activa, luego para finalizar se conecta el dispositivo al ordenador por medio del cable *USB*.

2.19. SDK Studio, herramienta para ejecución de *Android en PC*

Las iniciales *SDK* quieren decir *Software Development Kit* que traducido se entiende como un kit de desarrollo de programas, con el cual se puede desarrollar aplicaciones móviles y poder ejecutarlas desde el ordenador por medio de un emulador. Este kit de desarrollo está orientado exclusivamente para el sistema operativo *Android* y trae consigo todos los componentes y herramientas necesarias que se instalarán en el dispositivo móvil para su correcto funcionamiento.

Después de haber descargado el *SDK* se ejecuta el *SDK Manager* donde se descargará la versión de *Android* requerida. (Subirats, 2014)

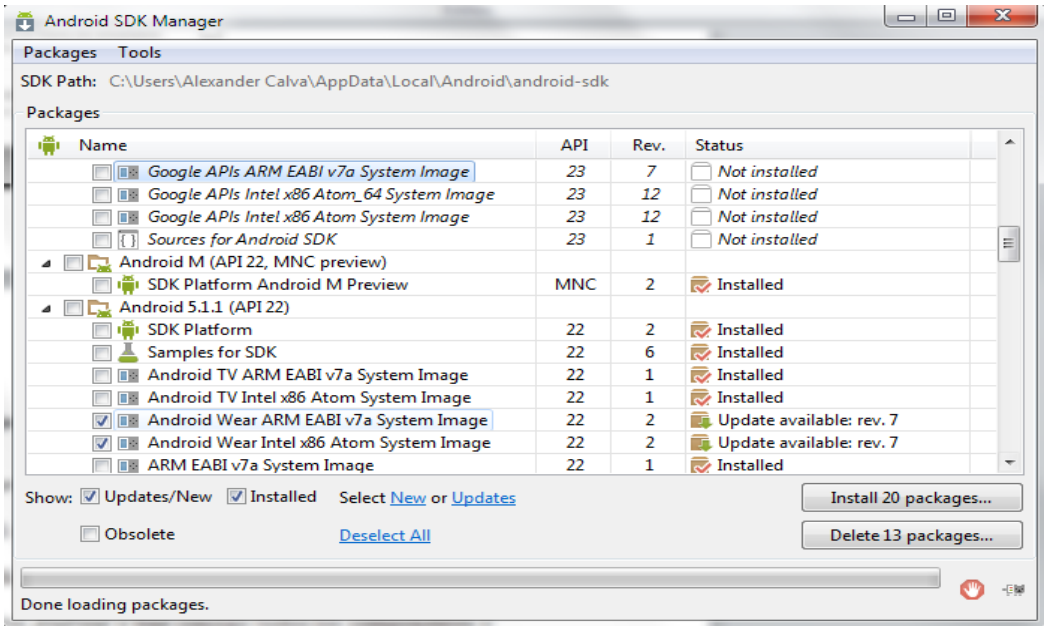

Figura 41. SDK Manager

2.20. Culminación de la aplicación

Google es la empresa que tiene los derechos de *Android*, solicita que todas las aplicaciones realizadas para su plataforma sean firmadas por sus desarrolladores. Esto impide la piratería y violaciones de derechos de autor de las mismas. Este proceso se hace desde el mismo ordenador que fue creada la aplicación móvil, ya que cada computadora posee un número de serie único en el mundo. Para realizar la firma de la aplicación se necesita hacerla en el sistema operativo de disco, en otras palabras si se está trabajando en *Microsoft Windows* se hará en D.O.S. ahora conocido como MS-DOS (*MicroSoft Disk Operating System*) por trabajar en las nuevas versiones de *Windows.*

Se recuerda que para realizar la firma de la *App* se necesita instalar las herramientas SDK y JDK (*Java Development Kit.*). Para realizar este proceso se debe ingresar al MS-DOS y escribir los siguientes códigos.

Keytool –genkey –v –keystore my release-key-keystore-alias alias_name-keyalg RSA- keysize 2048-validity 10000

Con este código se crea una clave de acceso única para modificaciones futuras de la aplicación, realizada por el desarrollador, además la hace válida a la *App* por 10000 días.

Jarsigner-verbose-sigalg SHA1withRSA-digestalg SHA1-keystore my-release-key-keystore my reléase-key.keystore my-application.apk alias_name

Con la siguiente instrucción se escoge el archivo *Unsigned APK* dentro de la carpeta del proyecto, que no es más que el nombre de la aplicación sin firmar y se le asigna la clave creada anteriormente.

Jarsigner-verify-verbose-certs my_application.apk

Al ingresar esta instrucción se verifica que la aplicación este firmada correctamente.

En los dispositivos móviles con *Android* se pueden instalar también aplicaciones que no consten en la tienda de *Play Store*, al desactivar en la configuración del Smartphone ‘Instalar solo aplicaciones firmadas por *Android*’. Al estar firmadas las aplicaciones aseguran que no serán dañinas para el dispositivo, eso respalda Google.

2.21. Opinión general de usuarios sobre la aplicación móvil

Al solicitar criterios de diferentes personas sobre el desenvolvimiento de la aplicación, se llegó a la conclusión, de que está muy bien realizada y de fácil comprensión. Se podría mejorar en el aspecto visual, por ejemplo, al realizar la generación de frecuencias se despliegue una ilustración del resultado.

2.22. Análisis económico

Todo proyecto debe contar con un presupuesto para lograr su objetivo. Para esta investigación se necesitó la inversión de un dispositivo móvil con sistema operativo Android actualizado, la cual fue de $500.

El tiempo invertido para la realización práctica del proyecto fue de seis meses, si se hace un cálculo promedio de las horas diarias utilizadas que fueron cuatro y se toma en cuenta una jornada laboral normal en el medio, se podría concluir que se trabajó media jornada diaria durante seis meses. Un programador gana promedio al mes $1000, por lo cual, utilizando los cálculos anteriores, se llega a la conclusión, que el costo del proyecto fue de $3500.

2.23. Proyecciones de la aplicación

Al finalizar una investigación siempre se analizan proyecciones futuras para su mejoramiento, en este caso, una de las principales sería, hacerla pública con un soporte de correo electrónico para recibir comentarios y sugerencias de los usuarios. Otra proyección sería, contar con una base de datos de cajas acústicas sugeridas por otros usuarios.

3. CONCLUSIONES Y RECOMENDACIONES

3.1. Conclusiones

Al empezar el proyecto se investigó los posibles alcances del mismo, teniendo como conclusión que se puede mejorar la aplicación para poder hacerla pública y ofrecer un servicio educativo para los estudiantes de la carrera de ingeniería de sonido.

Pure Data como lenguaje de programación es muy versátil por su compatibilidad con otros lenguajes y se eligió *Visual Studio* para este proyecto por el conocimiento previo de su sintaxis.

Se opta por no trabajar con *LibPd* por su plataforma de desarrollo que es en *Apple OSX* y la planificación estaba dada para operar en *Microsoft Windows* 7.

Tanto *MobMuPlat* como *PdDroidParty* poseen interfaces predefinidas, donde el usuario puede asignar comandos o funciones programadas previamente, por lo cual, no fue de utilidad para este trabajo.

Android Studio es un lenguaje exclusivo para programar aplicaciones para el sistema operativo *Android* como dice su nombre, mas, no tiene las características conocidas para ser compatible con *WebPd* según se investigó, trabaja con *LibPd*.

Existen en el mercado digital pocas aplicaciones móviles desarrolladas con *Pure Data,* las cuales fueron explicadas anteriormente. La ventaja de este proyecto es abrir un nuevo enlace entre un lenguaje de programación orientado a la interacción multimedia con el mundo de las aplicaciones móviles.

Microsoft Visual Studio fue la mejor elección para desarrollar este proyecto, porque es un entorno de trabajo que sirve para varias plataformas móviles, es decir, se pueden crear aplicaciones para *Smartphones* con sistemas operativos como *IOS de Apple*, *Windows Phone* de *Microsoft* y *Android* de *Google* en el cual fue realizado este trabajo de investigación.

Se tuvo que reenviar los datos ingresados por el usuario a *Pure Data*, por una razón desconocida de prioridad de ingreso de las variables a calcular.

Para poder ejecutar la aplicación en el dispositivo móvil, se necesitó que esté instalada la versión de *Android* 5.0, contrario a esto salía un error de compilación.

No todos los objetos utilizados en *Pure Data* eran compatibles con el *Apache Cordova*, por tal razón se programó de una forma más extensa al utilizar operaciones matemáticas. Por ejemplo, no se podía hacer el cálculo de varias sumas, restas y multiplicaciones en una sola expresión con el objeto [expr], por no ser coincidente.

Existen dos versiones de *Pure Data*, la primera es la *Vanilla* que es la básica y más antigua y la segunda en la *Extended* que es similar a la primera versión, con la diferencia de incluir librerías creadas por desarrolladores informáticos. Para realizar una aplicación utilizando Pure Data como herramienta de desarrollo, se debe utilizar la versión *Vanilla* por no tener problemas de licencias de librerías.

3.2. RECOMENDACIONES

Apache Cordova al ser un *Framework* libre, tiene conflictos de actualizaciones donde se pierde estabilidad al desarrollar un proyecto a largo plazo, lo cual, requiere descargar nuevos componentes para poder ejecutar un proyecto al subir de versión. Se recomienda sacar respaldos de la carpeta del proyecto para evitar futuros incovenientes.

Para poder enlazar *Pure Data* con *Apache Cordova* se necesitó hacer ejercicios sencillos de comprobación, donde se envió dos números a *Pure Data* para que realice una operación matemática y devuelva el resultado visualizado en la interfaz de usuario, se aconseja realizar este preceso hasta lograr la comunicación total entre las dos plataformas.

En lo posible adquirir un *Smartphone* de última generación para la programación de la aplicación, la razón, por la facilidad de instalación de los componentes compatibles con el ordenador, que establecen la conexión total entre ambos.

Se recomienda trabajar con el sistema operativo Android actualizado hasta la fecha de la creación de la aplicación móvil, por razones de compatibilidad con las herramientas SDK.

Si se desea crear aplicaciones móviles para distintas plataformas de sistemas operativos móviles a la vez, se recomienda no utilizar el *Apache Cordova*, sino, el *Phone Gap*. La razón se debe por el uso de sus componentes complementarios de compilación, es decir, *Phone Gap* posee todos los instaladores para su compilación y ejecución de la aplicación, sin importar en que sistema operativo este trabajando en el ordenador.

REFERENCIAS

Alvarez,R.(2001).Manual de HTML.DesarrolloWeb.com.[Versión electrónica]. Recuperado el 12 de septiembre de 2015 de http://www.desarrolloweb.com/articulos/556.php

Arturo,C.(2014).Cómo hacer menú de navegación adaptable a dispositivos móviles(Responsive Design). Recuperado el 4 de octubre de 2015 de http://www.falconmasters.com/web-design/menu-de-navegacion-responsive/

Basantes,F.(2009).Guía práctica y metodología para el diseño y construcción de un sistema de dos vías en una caja acústica con reflector de bajos. Recuperado el 20 de junio de 2015 de http://dspace.udla.edu.ec/bitstream/33000/4038/1/UDLA-EC-TISA-2009-06(S).pdf

Betopvd.(2012).Instalar JDK en Windows 7. Recuperado el 20 de octubre de 2015 de https://www.youtube.com/watch?v=wFLgZym41mg

Bolaños,D.(2015).Guía para rootear el Motorola Moto G,desbloquear bootloader,instalar recovery, kernel y roms. Recuperado el 18 de octubre de 2015 de http://www.movilzona.es/2015/05/20/motorola-moto-g-root-bootloader-recovery-kernel-roms/

Capuzano A.(s.f.).Cálculo de una caja cerrada. Recuperado el 11 de junio de 2015 de http://www.pcpaudio.com/pcpfiles/doc_altavoces/cajas/cerrada/cerrada.html

Carrodeguas,N.(2015).Como cambiar y modificar el estilo CSS de las páginas web con Javascript. Recuperado el 27 de septiembre de 2015 de https://norfipc.com/inf/javascript-como-cambiar-modificar-estilo-css-paginas-web.html

Developers.(s.f.).Signing Your Applications.Herramientas. Recuperado el 21 de octubre de 2015 de http://developer.android.com/intl/es/tools/publishing/app-signing.html#studio

FalconMasters.(2014).Cómo hacer botones Flat usando íconos con CSS. Recuperado el 29 de septiembre de 2015 de https://www.youtube.com/watch?v=3LiOUvuQ15Q

FalconMasters.(2014).Cómo utilizar íconos personalizados en nuestro sitio web mediante fuentes y CSS. Recuperado el 29 de septiembre de 2015 de https://www.youtube.com/watch?v=U7GsS5lhGm8

García,C.(s.f.).Diseño de cajas acústicas.Electroacústica.Universidad del País Vasco.[Versión electrónica]. Recuperado el 7 de abril de 2015 de http://aholab.ehu.es/users/imanol/akustika/IkasleLanak/Dise%F1o%20de%20cajas%20acusticas.pdf

GitHub.(2015).WebPd/OBJECTLIST.md. Recuperado el 9 de agosto de 2015 de https://github.com/sebpiq/WebPd/blob/master/OBJECTLIST.md

GitHub.(2016).Sebpiq/WebPd. Recuperado el 6 de agosto de 2015 de https://github.com/sebpiq/WebPd

Gonzalez,E.(2006-2016).Crear tablas HTML coldspan y rowspan.Unir celdas:horizontal y vertical tr,td,th.Caption o título(CU00719B). Recuperado el 17 de septiembre de 2015 de http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=516:crear-tablas-html-colspan-y-rowspan-unir-celdas-horizontal-y-vertical-tr-td-th-caption-o-titulo-cu00719b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192

Google.(2016).CanOfBeats.Google Play. Recuperado el 28 de julio de 2015 de https://play.google.com/store/apps/details?id=cx.mccormick.canofbeats

Google.(2016).mPD. Recuperado el 25 de julio de 2015 de https://play.google.com/store/apps/details?id=org.mpd

Google.(2016).TouchOSC. Recuperado el 18 de julio de 2015 de https://play.google.com/store/apps/details?id=net.hexler.touchosc_a

Hartikainen,J.(2015).Javascript errores y cómo solucionarlos. Recuperado el 24 de septiembre de 2015 de http://davidwalsh.name/fix-javascript-errors

IcoMoon.(s.f.).Recuperado el 1 de octubre de 2015 de file:///C:/Users/Alexander%20Calva/Documents/Visual%20Studio%202015/Projects/BlankCordovaApp4/BlankCordovaApp4/www/icomoon/demo.html

Iglesia Intermedia.(s.f.).MobMuPlat Mobile Music Platform. Recuperado el 14 de julio de 2015 de http://www.mobmuplat.com/

Iribar,A.(s.f.).HTML básico –I Cómo crear una página web.[Versión electrónica]. Recuperado el 18 de septiembre de 2015 de http://paginaspersonales.deusto.es/airibar/Ed_digital/HTML/HTML_1.html#top

Jordá,S.(2004). Manual de Introducción a PD.[Versión electrónica] Recuperado el 7 de junio de 2015 de http://www.tecn.upf.es/~sjorda/PD/IntroduccionPD3.pdf

Kreidler,J.(2009). Programando Música Electrónica en Pd. Recuperado el 5 de junio de 2015 de http://lucarda.com.ar/pd-tutorial/index.html

LIBROSWEB.(2016).Fondos. [Versión electrónica]. Recuperado el 18 de septiembre de 2015 de https://librosweb.es/libro/css/capitulo_4/fondos.html

Lopez,M.(2014).Video de presentación de Apache Cordova. Recuperado el 30 de agosto de 2015 de https://www.youtube.com/watch?v=EAaiD06G_1Q

Mccormick,C.(2013).PdDroidParty. Recuperado el 8 de julio de 2015 de http://droidparty.net/

Microsoft.(2016).Mostrar texto en una página web(Javascript). Recuperado el 20 de septiembre de 2015 de https://msdn.microsoft.com/es-es/library/yfc4b32c(v=vs.94).aspx

Microsoft.(2016).Problemas conocidos de herramientas para Apache Cordova. Recuperado el 7 de septiembre de 2015 de https://www.visualstudio.com/explore/cordova-known-issues-vs

Millán,A.(2011).Parámetros Thiele-Small. Recuperado el 30 de mayo de 2015 de http://www.diffusionmagazine.com/index.php/biblioteca/categorias/cientifica/196-parametros-thiele-small

Pellis,G.,Vargas,G.,Zambroni,E.(2011).Cajas acústicas, características y aplicaciones. Recuperado el 20 de mayo de 2015 de http://www.profesores.frc.utn.edu.ar/electronica/fundamentosdeacusticayelectroacustica/pub/file/FAyE0511E2-Pellis-Vargas-Zambroni.pdf

Piqué,D.(2013).Curso Android depuración USB con teléfono. Recuperado el 15 de octubre de 2015 de https://www.youtube.com/watch?v=jJMfR9nox18

Pouquet,P.(2016).Lenguajes de programación. Recuperado el 5 de mayo de 2015 de http://es.ccm.net/contents/304-lenguajes-de-programacion

Romero,M.(s.f.).Objetos de control. Recuperado el 3 de mayo de 2015 de http://cargocollective.com/max-pd-tutorial/objetos-de-control

Solis,C.(2012).Tutorial:Configurar equipos Android para probar aplicaciones. Recuperado el 11 de octubre de 2015 de http://revolucion.mobi/2012/10/30/tutorial-configurar-equipos-android-para-probar-aplicaciones/

Sonic Network, S.L.(2016). Pdlib, PureData llega a Android, IOS y más. Recuperado el 3 de julio de 2015 de

http://www.hispasonic.com/noticias/pdlib-puredata-llega-android-ios-mas/6376

Subirats,J.(2014).Qué es y para qué sirve el SDK. Recuperado el 7 de octubre de 2015 de http://www.fandroides.com/que-es-y-para-que-sirve-el-sdk/

Virtualnata.(2015). Etiqueta <button>. Recuperado el 22 de septiembre de 2015 de http://www.virtualnauta.com/html-etiqueta-button

Wikiversidad.(2013). Glosario de Pure Data. Recuperado el 27 de abril de 2015 de https://es.m.wikiversity.org/wiki/Glosario_de_Pure_Data

Zapata, E. (2014).Processing on Android to OSC to Pure Data. Recuperado el 29 de junio de 2015 de https://www.youtube.com/watch?v=hRDgdkQHtmM