CAPITULO IV: DESARROLLO DEL PROGRAMA ENCARGADO DE EMULAR LA CONSOLA ANALÓGICA

4.1 COMPONENTES BUSCADOS EN EL PROGRAMA

La idea del proyecto es emular una consola de audio, así que es indispensable que el programa tenga los mismos componentes de una consola de audio analógica y además contar con la posibilidad de agregar funciones adicionales en el futuro, como seria el caso de agregar eco o entradas de sonido directamente desde la computadora, estas características se pueden hacer programando la consola.

Como requisito principal necesitamos un programa muy estable, ya que va a correr en espectáculos, donde es inadmisible una falla en el sistema de sonido, por este motivo se escogió programar para la plataforma Mac OS X, ya que este sistema nos brinda la estabilidad necesaria para correr las aplicaciones y nos las herramientas de programación vienen integradas de manera gratuita con su sistema operativo.

Para emular una consola de audio necesita tener las mismas partes y controles que esta, de esta manera es necesario que cuente de manera independiente por cada canal de entrada con un control de volumen, controles de ecualización para compensación de las diferentes respuestas en frecuencia de los componentes, un control de ganancia para dar una cierta amplificación a la señal (que en nuestro caso va a quedar incluida en el control de volumen), un control de balance para elegir por que canal tiene salida la señal, un indicador de saturación del canal y por ultimo un botón que permita silenciar el canal de entrada.

Para el control de la salida es necesario un controlador de volumen por cada canal y un indicador visual del nivel de la señal.

Para poder configurar las entradas por cada canal, necesitamos usar una lista, en donde se muestren los micrófonos conectados y nos de la posibilidad de escoger uno u otro para cada canal. De manera similar es necesario colocar controles para que el usuario pueda escoger el dispositivo de salida del sonido, de tal manera que el sonido salga por los dispositivos correctos, pudiendo colocar mas de un dispositivo de salida.

El programa final va a estar programado en el lenguaje llamado Objective-C, pensado especialmente para Mac, este lenguaje de programación es

orientado a objetos y gracias a esto tiene muchos parecidos a Java[23], factor que nos ayuda a la hora de hacer el código con elementos conocidos y para poder exportar el código. Para hacer la programación se utilizara el programa XCode, proporcionado por Apple gratuitamente con la compra del sistema operativo.

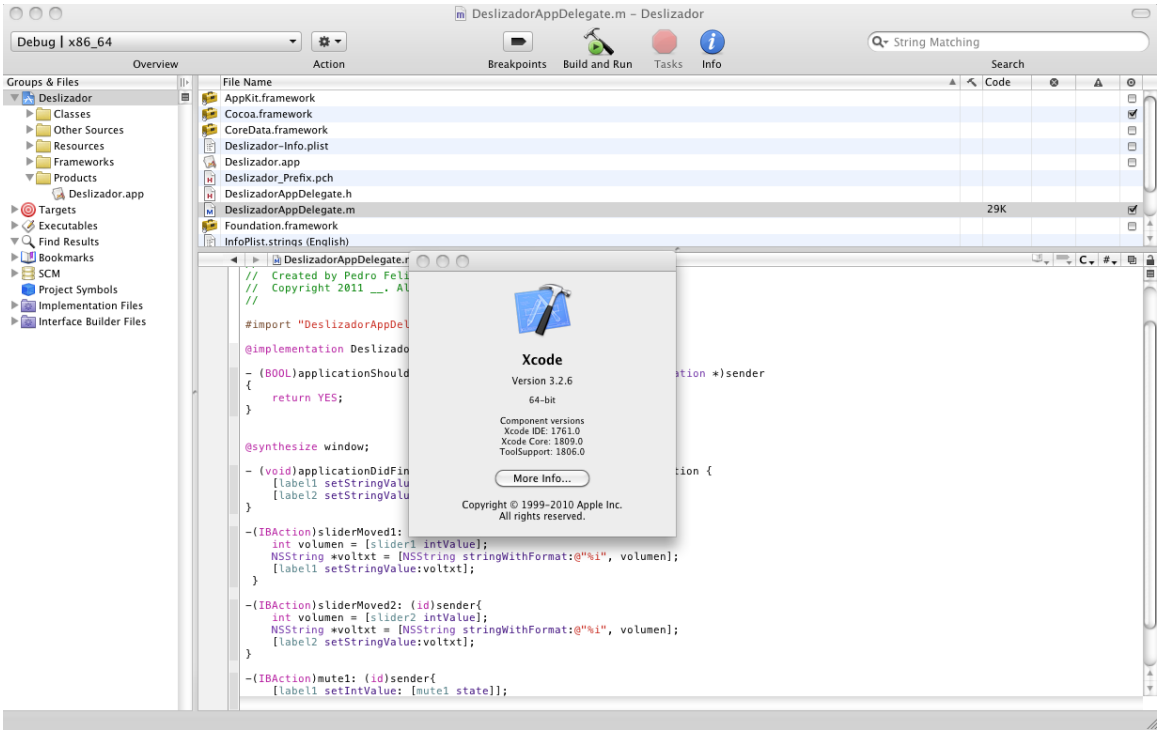

Figura 34. Ventana grafica para la programación en Xcode.

Este programa nos permite modificar los archivos de las clases y headers (archivos con terminación .m y .h) que creamos para poder hacer funcional el programa, también nos brinda la posibilidad de lanzar el programa necesario para la creación y edición de la ventana de nuestro programa, el cual se llama Interface Builder.

[23] *(Altenberg, Clarke and Mougin 2008)*

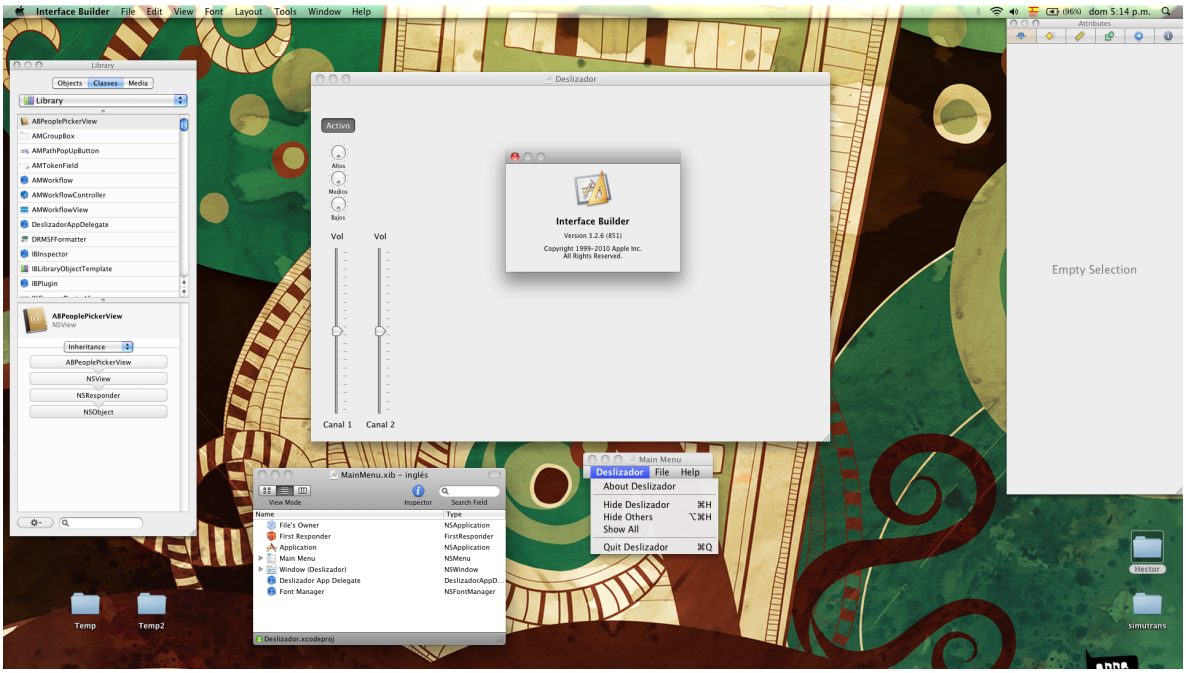

Figura 35. Programa para la creación de la interfaz grafica del programa.

En este programa nos podemos encargar de la elaboración de la ventana de nuestro programa, así como los menús y las opciones visuales, además de tener la oportunidad de crear clases y eventos dentro de estas desde la misma ventana.

Con las opciones brindadas por estos dos programas, ya estamos listos para poder comenzar a trabajar en el diseño y la implementación de nuestro programa.

Una vez que conocemos los componentes necesarios en la consola, comenzamos con el diseño de la ventana que va a tener el programa. Para asegurar la semejanza con una consola de audio analógica basamos el diseño en una consola real, además la semejanza con una consola real reduce el tiempo de aprendizaje para el usuario final.

De esta manera identificamos elementos claves, que son necesarios tener en nuestra consola y procedemos a establecer un diagrama esquemático de lo que necesitamos tener por cada canal de la consola.

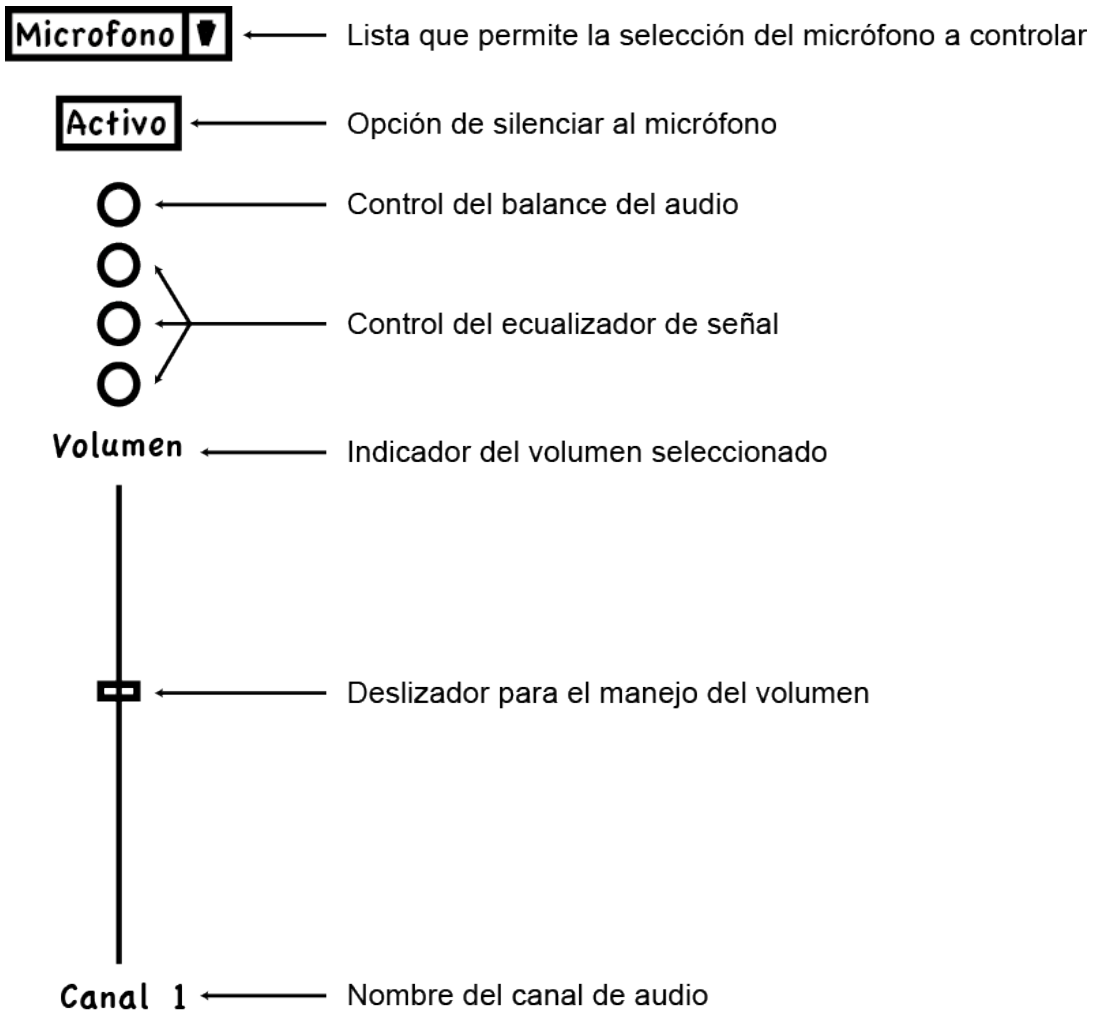

Figura 36. Esquema de los componentes buscados en la interfaz.

A continuación nos lanzamos a la programación de la interfaz grafica para poder determinar los elementos con los que íbamos a hacer nuestro programa y poder obtener una vista aproximada de cómo se verá el sistema.

Para poder realizar la ventana del programa utilizamos el programa Interface Builder, el cual nos proporciona ciertos elementos gráficos ya programados con anticipación, de tal manera que nos simplifica bastante las cosas a la hora de programar. Con base el diseño buscado, realizamos el programa y obtenemos una ventana que se asemeja bastante a nuestro boceto.

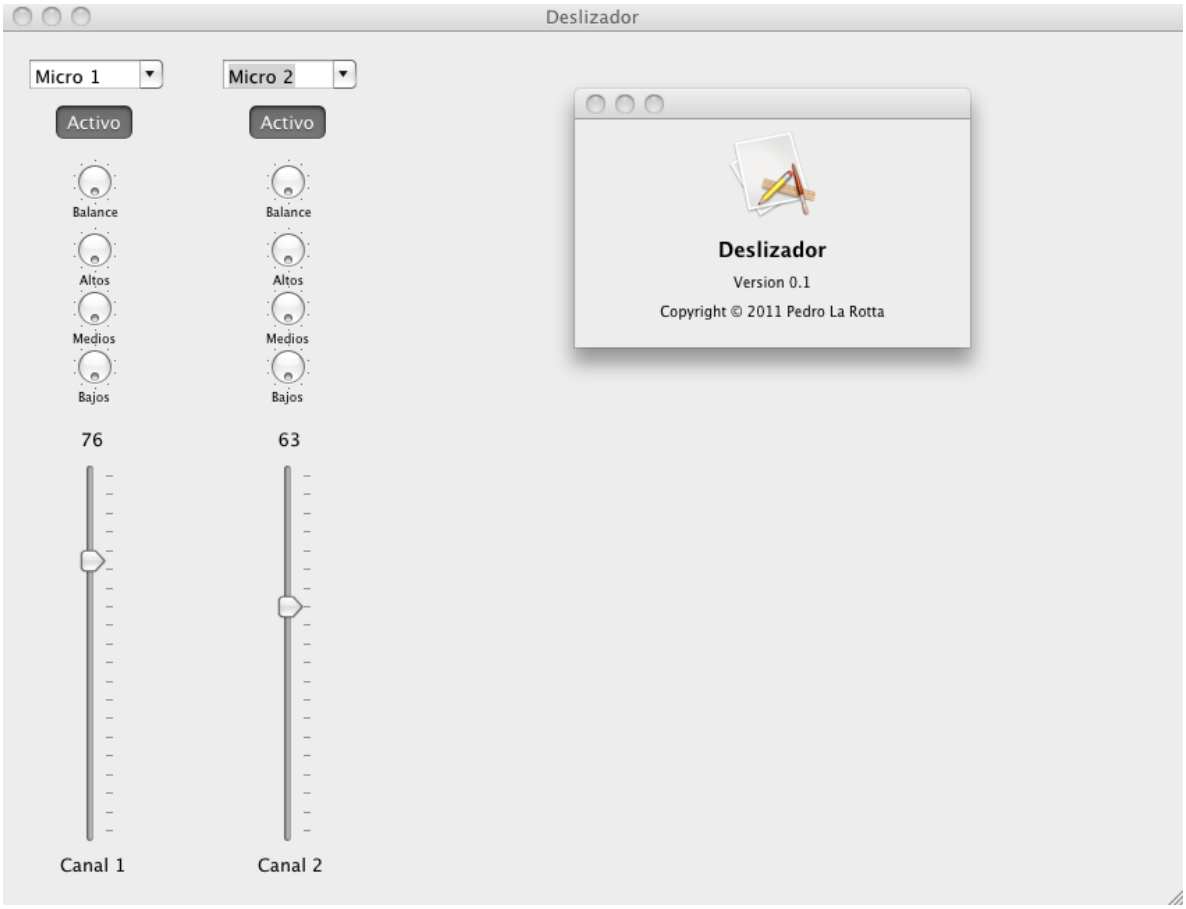

Figura 37. Primer boceto del programa que emula la consola.

Para poder personalizar el programa, decidí colocarle como nombre "AudioBoom", esto después de realizar una búsqueda en Google y ver que el nombre no ha sido usado para productos de software. Después de seleccionar el nombre era importante darle un logo en especial, para que

el cliente pueda distinguirlo de otros programas y de esta manera ganar reconocimiento visual de la marca.

El logo fue hecho en Adobe Illustrator CS5 e incorpora además del nombre, una bocina con ondas saliendo de esta para relacionar el programa con un software de manejo de audio.

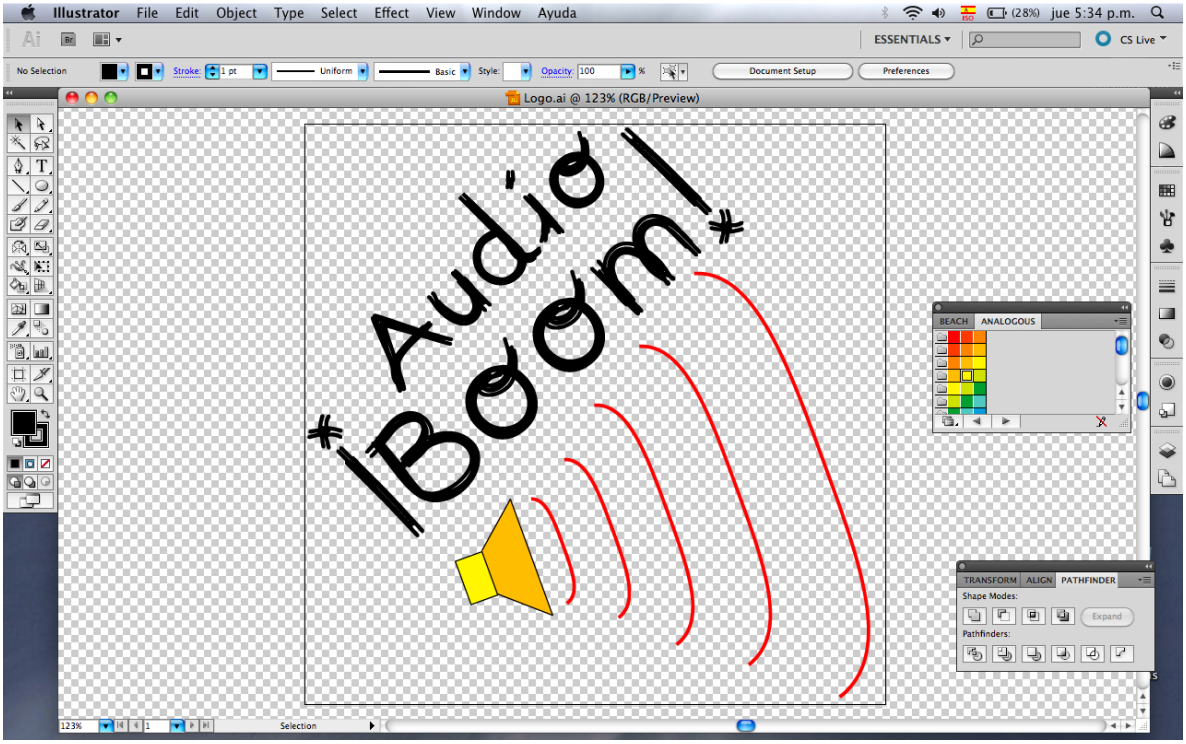

Figura 38. Realización del logo en Illustrator.

Una vez creado la interfaz grafica y los elementos de personalización, procedemos a crear nuestra clase y a declarar las acciones pertinentes para la clase, de tal manera que nos permita editar los parámetros del sonido que nosotros queremos, para esto es necesario manipular los archivos .h y .m creados al inicio del programa.

El próximo paso es crear la ventana con los componentes finales que va a tener nuestro programa, para esto contemplamos la opción de incluir cuatro canales de audio de entrada y un canal estéreo de salida.

Por cada canal incluimos los mismos componentes que en el primer diseño y obtenemos el resultado mostrado en la figura 38.

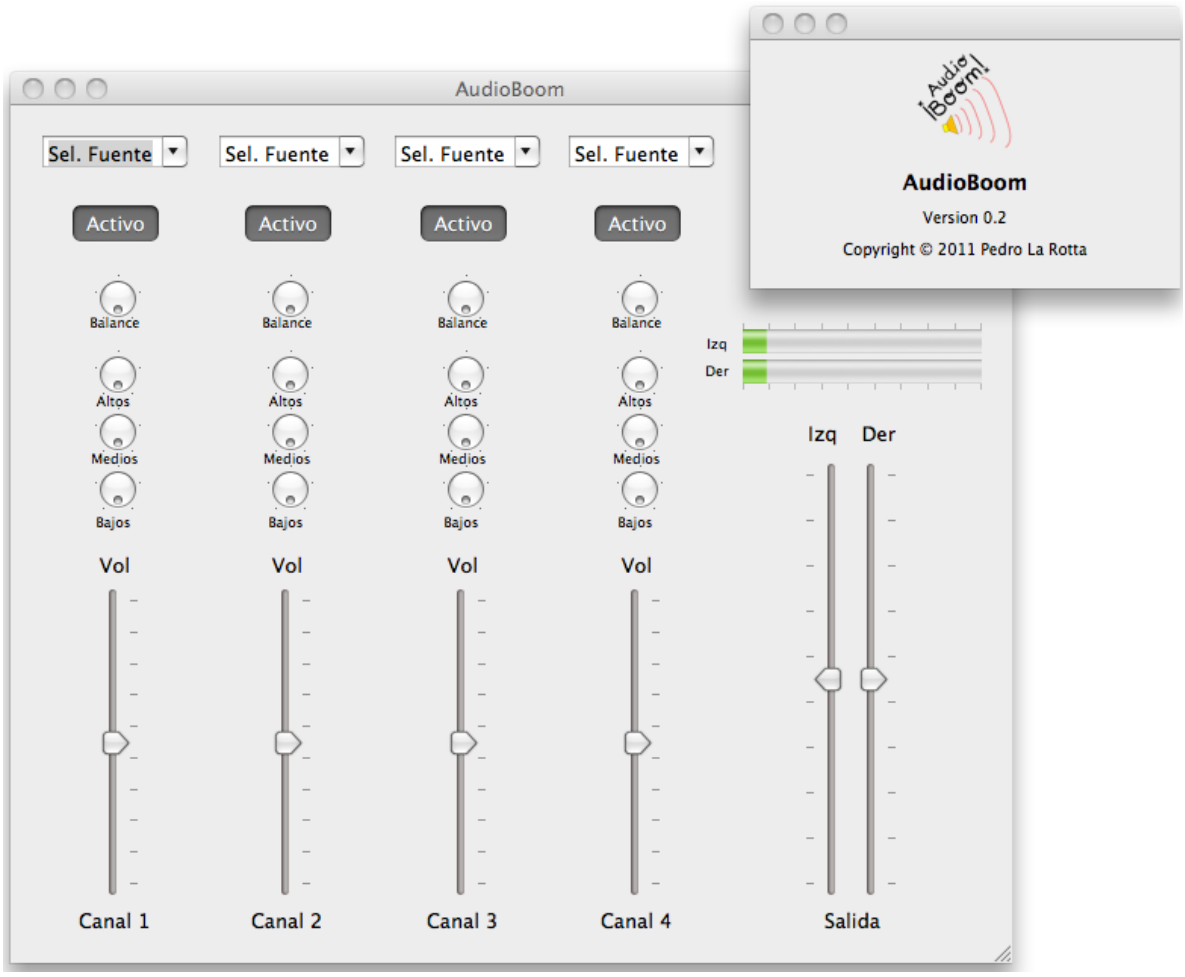

Figura 39. Diseño final de nuestra consola para cuatro canales.

Una vez desarrollada la interfaz grafica, procedemos a programar cada uno de los componentes del diseño.

4.3 PROGRAMACIÓN DEL CONTROL DE VOLUMEN Y CAMBIOS DESDE LA INTERFAZ GRAFICA:

El código de programación se divide en dos grandes partes, la primera es la encargada de los componentes de la interfaz grafica y los datos que se obtienen de esta, es en esta parte donde recuperamos los valores manejados por el usuario y devolvemos al usuario la información con los movimientos de los controles de la interfaz.

La segunda parte es la encargada de la mezcla de canales, la ecualización y el envío de la información al canal de salida.

Para la primera parte utilizamos un método por cada acción a realizar y alteramos una variable concerniente a toda la clase conforme la acción realizada por cada botón.

Los métodos utilizados para esta parte, se pueden ver de la siguiente manera:

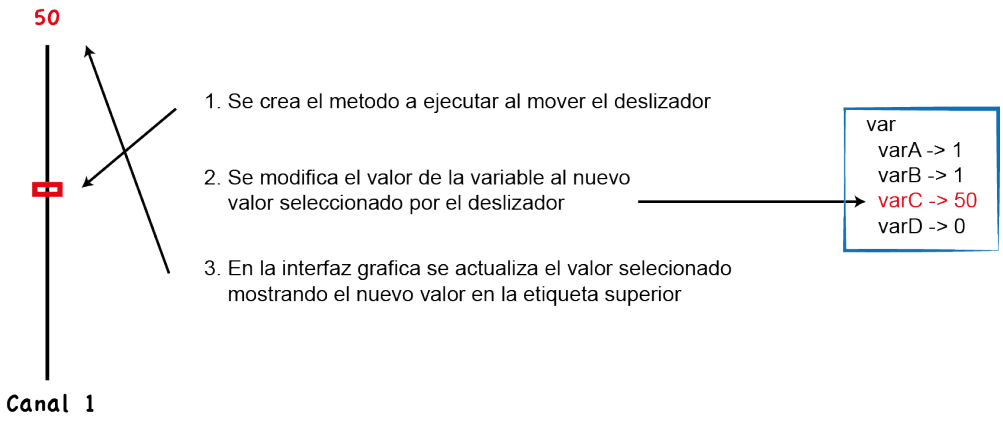

Figura 40. Descripción del método a utilizar.

El bloque de programación para los métodos de esta parte es el siguiente:

```
1)    -(IBAction)sliderMoved1: (id)sender{
            vol1 = [slider1 intValue];
2)          NSString *voltxt = [NSString stringWithFormat:@"%i", vol1];
3)          [label1 setStringValue:voltxt];
      }
```

Donde las líneas se pueden interpretar como:

1. Define el método de la clase, indica al programa que se ejecute este código al realizar el movimiento del deslizador de volumen correspondiente.
2. Cambia el valor de una variable del método, en este caso el valor del volumen se cambia, este valor se utilizara en la segunda parte del programa y servirá para modificar la amplitud de la señal de este canal.
3. Regresa la información al usuario de forma grafica, en la interfaz grafica se modifica una etiqueta para mostrar el valor actual del volumen de este canal.

Esta clase de código se implementa para los deslizadores del volumen, los controles del ecualizador y el control del balance del canal. Solo para los controles del volumen, se agrega el tercer paso, ya que para los otros no es necesario regresar la información del valor al usuario.

4.4 PROGRAMACIÓN DE LA MEZCLA DE CANALES Y ECUALIZADOR:

En la segunda parte del código, se ejecuta la parte importante, que es el tratamiento de la señal, para lograr esto se siguen los siguientes pasos:

1. Se realiza la unión de los datos capturados por un micrófono a determinado canal.
2. Se multiplican las señales de audio que llegan por el volumen indicado para cada canal. En caso de estar desactivado el canal (mute) las señales de entrada se multiplican por cero, dando lugar a la anulación del sonido.
3. La señal se procesa por medio de los filtros del ecualizador para poder mejor su calidad.
4. Dependiendo del ajuste del balance, la señal se envía por un determinado canal o por los dos.
5. La suma de todos los canales es multiplicada por el valor de volumen maestro del sistema y se procede a enviar la señal por la red hasta las bocinas.

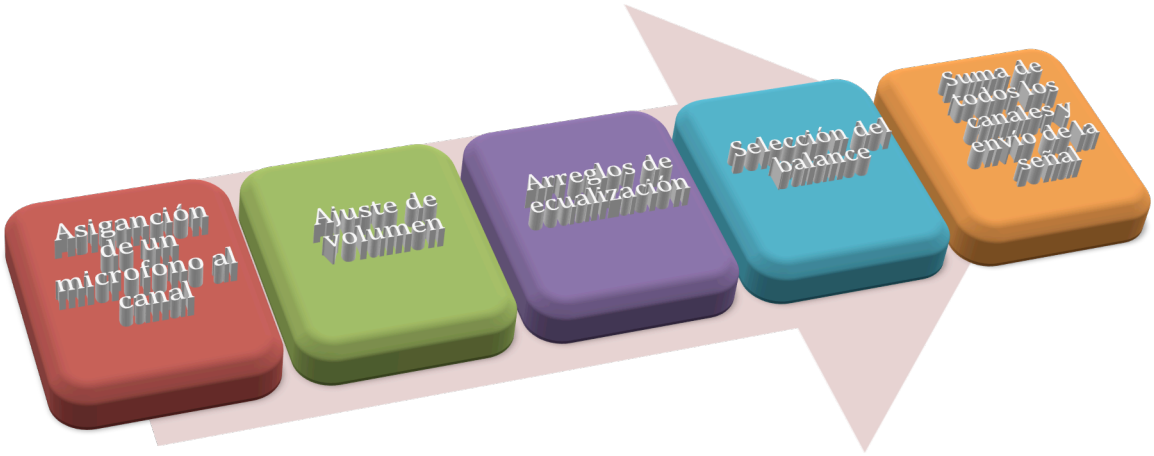

Figura 41. Diagrama del tratamiento de la señal.

Todo este procesamiento se va a ejecutar cada vez que llegue una muestra al canal en cuestión, de tal manera que si se tiene un canal libre, este no consume ningún recurso de procesamiento de la computadora.

Para poder implementar el ecualizador de tres canales en la consola, fue necesaria la introducción de tres filtros por cada canal, donde cada filtro se encarga de una banda de frecuencias.

Para el desarrollo de estos filtros se escogió la modalidad IIR (Infinite Impulse Response) y los filtros se desarrollaron por medio de Matlab y su herramienta FDATool.

Esta herramienta encuentra los coeficientes para los filtros deseados tan solo con poner las frecuencias de corte requeridas, las frecuencia de muestreo del sistema, el orden del filtro y por ultimo que clase de filtro queremos utilizar, así obtuvimos los coeficientes para los filtros requeridos.

Todos los filtros fueron ajustados como filtros Butterworth de cuarto orden y dos etapas, dando la posibilidad de tener un código corto y una calidad aceptable para las frecuencias de corte, donde la potencia de paso es la mitad de la potencia en la banda de transmisión, es decir, manejan los -3dB en las frecuencias de corte.

4.4.1 FILTRO PASA BAJAS

Este filtro se encarga de la región baja del audio, esta diseñado para tener una frecuencia de corte de 880 Hz:

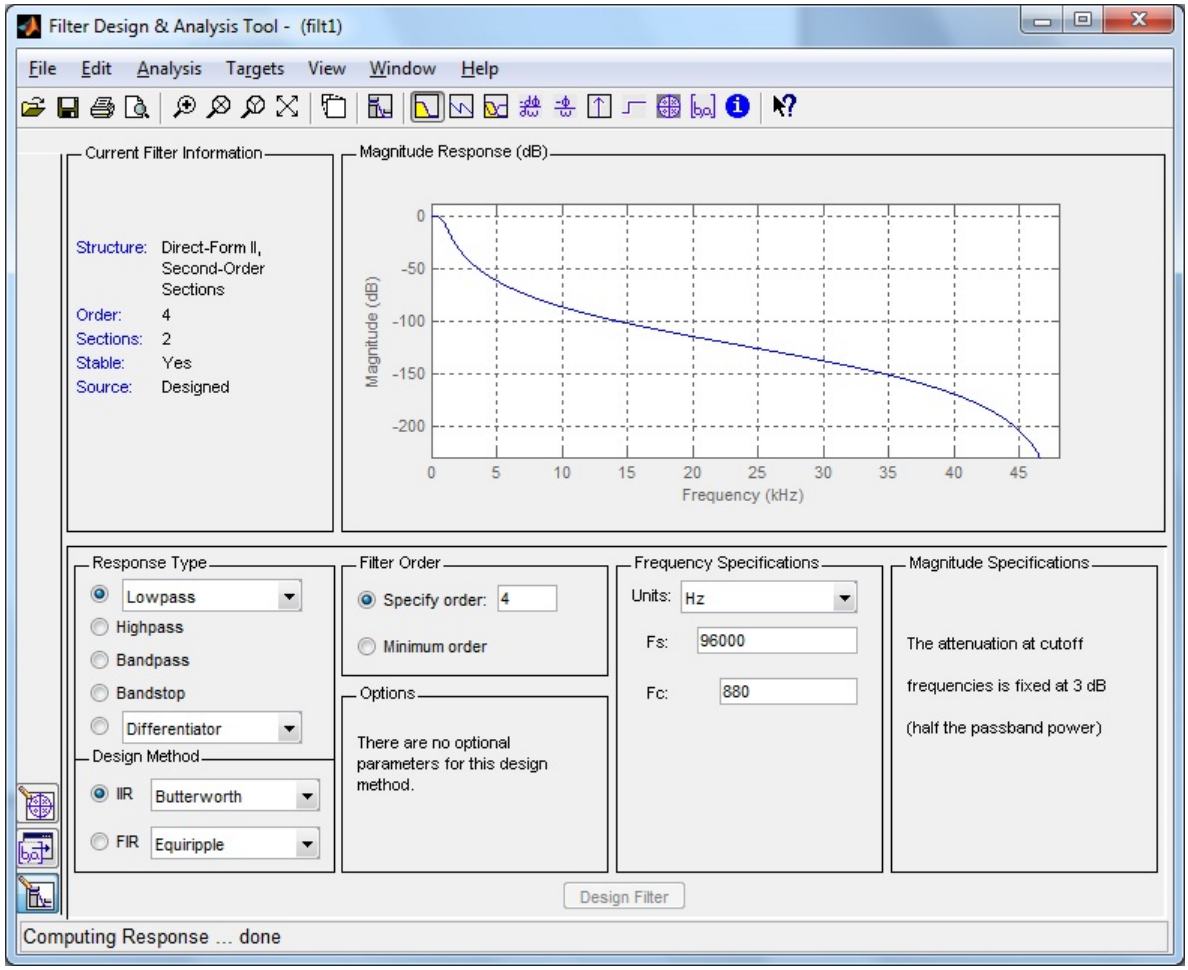

Figura 42. Diseño y respuesta en frecuencia del filtro paso bajas.

Los coeficientes obtenidos para este filtro están expresados en la siguiente matriz:

SOS matrix:
1 2 1 1 -1.9536471324152267 0.95689201824802606
1 2 1 1 -1.8958576999639873 0.89900660115988107

Donde la primera fila corresponde a los elementos del primer filtro y la segunda a los elementos del filtro posterior. De esta primera fila, las tres

primeras columnas corresponden a los elementos del numerador y las ultimas tres a los elementos del denominador. Podemos expresar esta matriz como dos ecuaciones de transferencia en z (3) y (4), una para cada sección del filtro:

$$H(z) = \frac{1+2z^{-1}+1z^{-2}}{1-1.9536z^{-1}+0.9569z^{-2}} \quad (3)$$

$$H(z) = \frac{1+2z^{-1}+1z^{-2}}{1-1.8958z^{-1}+0.8990z^{-2}} \quad (4)$$

4.4.2 FILTRO PASA BANDA

Para poder filtrar las frecuencias medas de la voz, fue necesario utilizar un filtro pasa banda que se centre en las frecuencias medias, así que diseñamos un filtro con frecuencia de corte inferior de 880Hz y una frecuencia superior de 5000Hz:

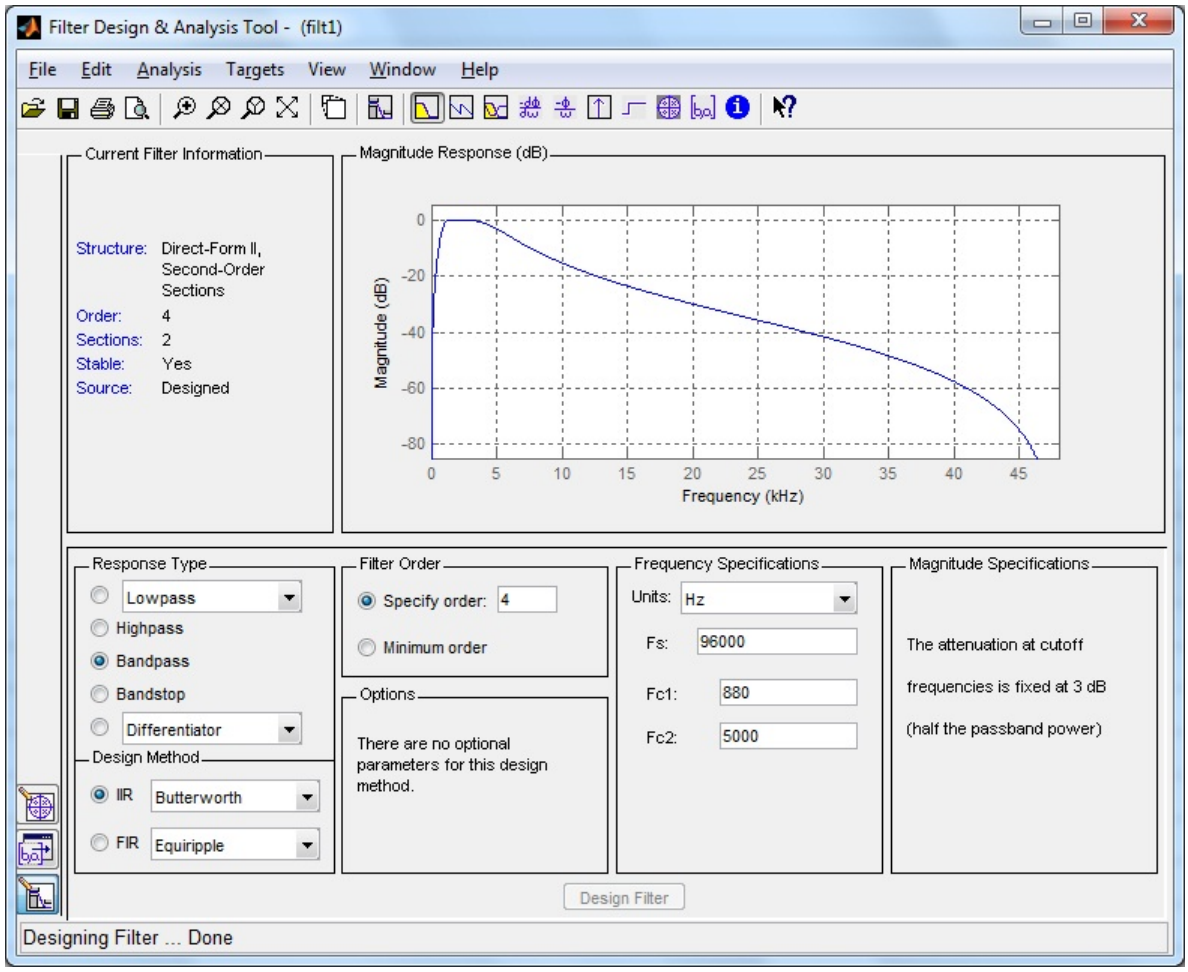

Figura 43. Diseño y respuesta en frecuencia del filtro pasa banda para frecuencias medias.

En este caso los coeficientes están dados por la matriz:

SOS matrix:
1 0 -1 1 -1.9286199706611717 0.93267604732883835
1 0 -1 1 -1.6592484503199194 0.73227524560661217

Y las ecuaciones de transferencia en el dominio de z son (5) y (6):

$$H(z) = \frac{1+0z^{-1}-1z^{-2}}{1-1.9286z^{-1}+0.9326z^{-2}} \quad (5)$$

$$H(z) = \frac{1+0z^{-1}-1z^{-2}}{1-1.6592z^{-1}+0.7322z^{-2}} \quad (6)$$

4.4.3 FILTRO PASO ALTAS

Para poder realizar el filtro de la banda de frecuencias superior, realizamos un filtro paso altas, en este caso la frecuencia de corte es 5000Hz y obtenemos el filtro con las siguientes características:

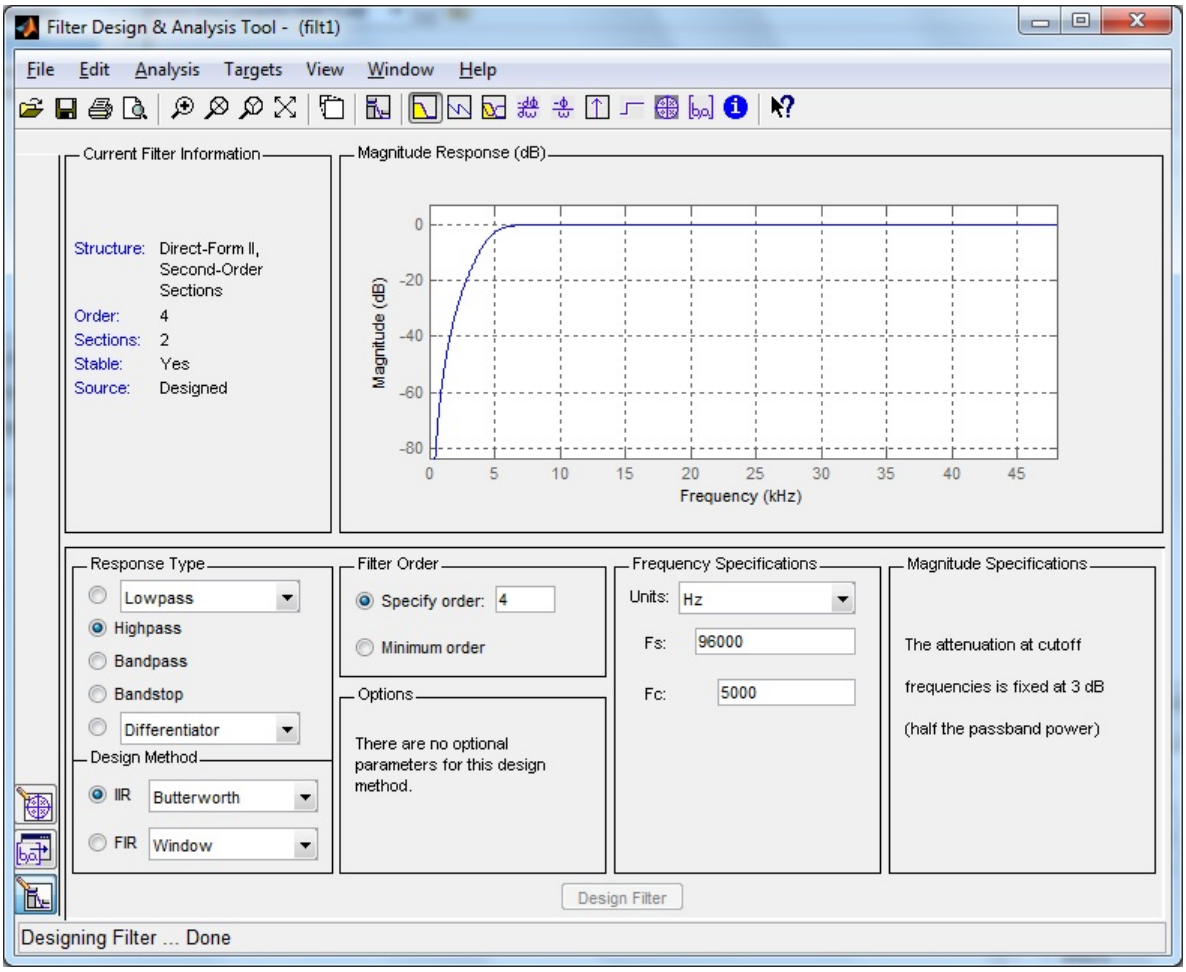

Figura 44. Diseño y respuesta en frecuencia del filtro pasa altas.

La matriz que muestra nuestros coeficientes es:

SOS matrix:
1 -2 1 1 -1.6864150849844293 0.78092874274009016
1 -2 1 1 -1.460217505426548 0.5420541177680368

Para este caso las ecuaciones de transferencia en el dominio z quedan de la siguiente manera (7) y (8):

$$H(z) = \frac{1-2z^{-1}+1z^{-2}}{1-1.6864z^{-1}+0.7809z^{-2}} \quad (7)$$

$$H(z) = \frac{1-2z^{-1}+1z^{-2}}{1-1.4602z^{-1}+0.5420z^{-2}} \quad (8)$$

4.4.4 PROGRAMACIÓN DEL ECUALIZADOR

Una vez obtenidas las funciones de transferencia de todas los filtros, procedemos a realizar la ecuación en diferencias en el dominio del tiempo discreto para poder programarla. Para hacer esto vemos el diseño del filtro, ya que desde aquí es mas fácil obtener la ecuación en diferencias.

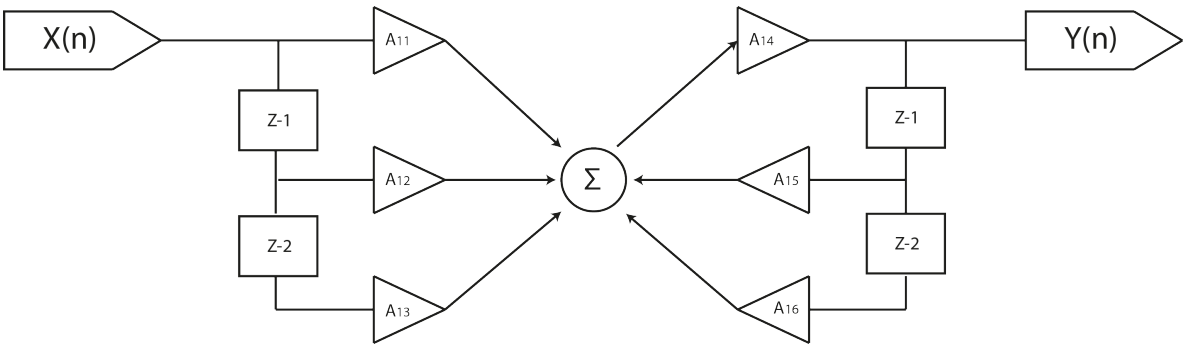

Figura 45. Diseño de la suma de diferencias para los diferentes filtros del proyecto.

En la imagen se puede observar los elementos de la matriz obtenida en Matlab y se puede llegar a un modelo general de ecuación de diferencias para ser programado:

$$Y(n)= A_{14}\,(A_{11}X[n] + A_{12}X[n-1] + A_{13}X[n-2] + A_{15}Y[n-1] + A_{16}X[n-2]) \quad (9)$$

Como resultado de la ecuación (9), tenemos la cadena a programar dentro de nuestro programa. Como el filtro esta diseñado para tener dos secciones, la salida de esta primera parte va concatenada a la segunda parte del filtro, la cual tiene la misma estructura y solo se cambian los coeficientes, utilizándose los valores de la segunda columna de la matriz.

Una vez que el audio de los diferentes canales ha sido capturado, filtrado y mezclado por el programa, procedemos a enviar la señal correspondiente de salida a las bocinas.

Este paso se realiza de forma análoga a la entrada de la información, es decir, la señal de audio de salida es enviada a través de la red de área local hasta la bocina, donde se contará con un equipo especial que haga la función opuesta al micrófono y como paso intermedio un amplificador.

Como salida del programa obtenemos los datos sin comprimir, así que cada muestra va a ser de 24 bits y una velocidad de muestreo de 96 000 Hz, de manera que todos los cálculos realizados para conocer el ancho de banda de la señal para el micrófono, son iguales para la salida de audio, teniendo la oportunidad de manejar hasta 41 canales de salida, solo hace falta conectar bocinas e indicar al sistema hacia que dirección (o direcciones) IP debe de enviar la información de salida.

Una vez que la señal de audio llega a la bocina, se inicia el proceso para pasar la señal a un formato analógico que pueda ser trabajado por el amplificador y con ello obtener una salida de audio a un volumen razonable.

Para lograr esto es necesario que el sistema contenga un DAC y su controlador de red, el cual se puede configurar para desempaquetar los datos, haciendo innecesario el uso de un controlador. Al dejar de utilizar el controlador en la salida, se hace imposible el uso de alguna clase de codificación de la señal para ahorrar ancho de banda, ya que los datos no se pueden procesar para obtener la señal de salida original.

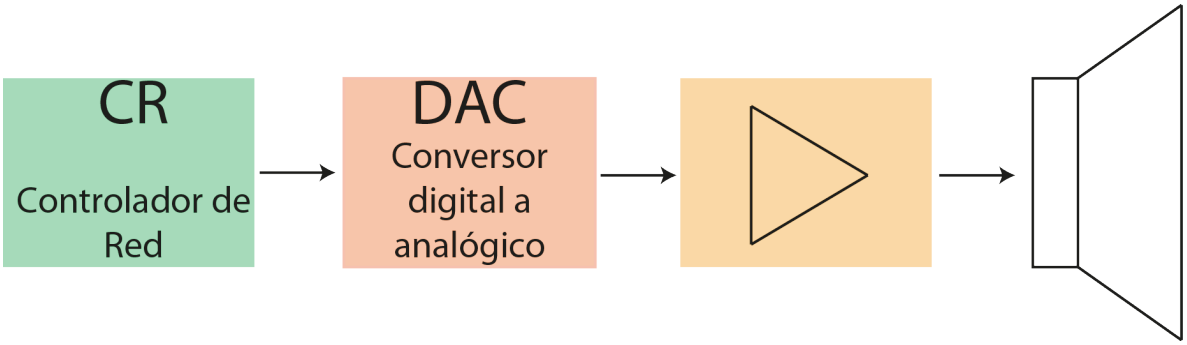

Figura 46. Componentes buscados a la salida del sistema.

Para este caso no es necesario contar con un conversor digital a analógico de 24 bits, siendo remplazado por uno de menor valor, siempre y cuando se haga la reducción en la consola y entendiendo que al hacer esto, se perdería cierta calidad en la señal. Este conversor es el encargado de transformar las señales digitales a su representación analógica, esta señal de salida es demasiado pequeña para ser reproducida por la bocina, lo que hace necesario el uso de un amplificador de potencia para la alimentación de la bocina.

El amplificador puede ser de cualquier clase, pero se buscara uno que tenga un consumo energético con pocas perdidas y tenga una buena respuesta para todas las frecuencias.

Todo este sistema de salida necesita tener alimentación eléctrica para poder funcionar, lamentablemente no de puede lograr una alimentación por medio del mismo cable Ethernet, ya que la cantidad de potencia necesitada supera la cantidad de potencia que se puede transferir por medio de este cable bajo la configuración PoE (Power over Ethernet). De llegar a contar con este sistema, solo es posible alimentar al controlador de red y el conversor digital a analógico.

Dada la necesidad energética del amplificador es conveniente contar con una fuente de alimentación cercana, es decir, una fuente de voltaje alterno o en su caso una batería. Aprovechando la necesidad de esta fuente de alimentación, todo el sistema receptor sacara su energía de ese punto, evitando complicar el sistema o sobre cargar el switch con componentes de PoE para las salidas de audio. Para poder lograr esto es necesario la alimentación del sistema por medio de un eliminador de baterías o adaptador de voltaje.