

# Ecualizador de audio de 4 bandas

## Informe final

UNLP - Facultad de Ingeniería

Taller de Proyecto I (E0306)

Segundo Semestre de 2018

Grupo 11

Sanchez Agustín, 939/2

Stranieri Jorge, 917/5

Marzano Nicolás, 138/7

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Objetivos del Proyecto</b>	<b>4</b>
<b>3. Análisis de Requerimientos</b>	<b>5</b>
3.1 Requerimientos funcionales	5
3.2 Requerimientos no funcionales	6
3.3 Interacción con el usuario	6
<b>4. Diseño del Hardware</b>	<b>7</b>
4.1 Diagrama general	7
4.2 Esquemático de conexiones en KiCad	13
4.3 Diseño de PCB en KiCad	14
4.4 Listado BOM	16
<b>5. Diseño del software</b>	<b>17</b>
5.1 Arquitectura principal	17
5.2 Capa de interfaz de usuario	17
5.3 Capa de procesamiento de señal	19
5.4 Filtros	22
5.5 Código del proyecto	26
<b>6. Ensayos y mediciones</b>	<b>30</b>
<b>7. Conclusiones</b>	<b>49</b>
<b>8. Bibliografía</b>	<b>52</b>
<b>9. Anexos</b>	<b>53</b>

# 1. Introducción

## Objetivo

Se desea realizar una interfaz de usuario basada en display LCD y teclado para configuración de niveles de atenuación para cada banda. Los niveles de atenuación serán en 10 pasos dentro del rango de 0 dB a -20 dB.

Se desea tener una señal de audio de entrada la cual será procesada obteniendo una señal de audio de salida. Se busca implementar mediante 4 filtros pasabanda digitales con ganancia ajustable.

## Motivación

Si bien existen ejemplos de filtros FIR funcionando con los módulos de ADC y DAC de la EDU-CIAA, el proyecto apunta a demostrar sus capacidades para el procesamiento digital de señales y la ecualización de audio. De esta forma lograr realizar el ecualizador sería una contribución al Proyecto CIAA, ya que si bien en su web se presentan ejemplos desde sistemas de riego hogareño hasta drones y otros vehículos robot, no existe ningún caso de uso con aplicaciones de audio.

## Características principales

El proyecto se compone de una interfaz simple con el usuario, compuesta de una pantalla lcd y un teclado, un conector de audio para la señal de entrada, y un parlante. La funcionalidad principal es permitir al usuario usar el teclado para elegir los niveles de atenuación de cuatro bandas de frecuencias de la señal de entrada, y que el resultado se vea reflejado en la señal de salida, escuchandola por el parlante.

## Marco general del estado del arte de la tecnología

Desde fines de los 90, los ecualizadores paramétricos (aquellos que permiten seleccionar una parte del rango de frecuencia de una señal para alterar su potencia) empezaron a crecer en disponibilidad en la forma de equipamiento de procesamiento digital de señales (DSP). Hoy en día podemos abrir un reproductor de audio liviano y encontrar un ecualizador de 18 bandas que parece funcionar con un retardo imperceptible. Existe también equipamiento "outboard" (unidades externas de efectos) disponible comercialmente que usa DSP para ecualizar y agregar otros efectos a unidades de audio.

Si bien existe una diferencia grande entre un programa corriendo sobre un sistema operativo en una computadora comercial para consumo general y un software dedicado en una computadora de bajo consumo como la CIAA, y entre la CIAA y un hardware outboard completamente dedicado al procesamiento de audio, esperamos

demonstrar que es suficientemente capaz de al menos un procesamiento de efectos de audio básico.

## Uso de las ideas o conceptos del proyecto

Si bien no se espera que se logre un ecualizador viable para aplicaciones comerciales, una exitosa implementación del ecualizador dejará precedentes para el proyecto CIAA sobre la capacidad de la computadora de procesar audio, y su uso junto con electrónica de amplificación y acondicionamiento de señales analógicas.

## 2. Objetivos del Proyecto

El objetivo primario de este proyecto es el diseño e implementación de un ecualizador de audio de 4 bandas que pueda ser configurado por el usuario. No se tiene como objetivo realizar un producto con aplicaciones reales, sino que se busca realizar un prototipo a modo de prueba de concepto. Por lo tanto, se analizará los límites de procesamiento de la EDU-CIAA para lograr obtener un procesamiento digital en tiempo real de la señal de entrada mediante la utilización de 4 filtros FIR.

El proyecto se divide en 4 módulos importantes:

- 1) Interfaz de usuario.
- 2) Acondicionamiento de señal de entrada.
- 3) Procesamiento digital de la señal.
- 4) Acondicionamiento de señal de salida.

### Interfaz con el usuario

Se tendrá un LCD de 2x16 para presentar información sobre el estado actual del ecualizador al usuario. Se usará un teclado para modificar los valores de atenuación de las diferentes bandas de frecuencias. El ecualizador deberá permitir al usuario elegir el nivel de atenuación para cada una de las cuatro bandas de frecuencias definidas.

### Acondicionamiento de la señal de entrada

Se busca disponer una conexión directa con el puerto de audio de un celular. La señal de entrada será pre-acondicionada para acomodarla a los rangos necesarios para la entrada al ADC de la EDU-CIAA, para su posterior procesamiento digital.

### Procesamiento de la señal

Por software se manejará la separación de la señal en cuatro bandas mediante filtros FIR, la atenuación de las diferentes bandas según los parámetros elegidos por el usuario, y los módulos ADC y DAC.

Si el filtrado y procesado en la EDU-CIAA resulta ser suficientemente rápido, podría aumentarse la cantidad de bandas del ecualizador para probar el límite de la EDU-CIAA para el procesamiento de audio en tiempo real , así como también del orden de los filtros digitales a utilizar.

### Acondicionamiento de la señal de salida

El acondicionamiento de salida buscará amplificar la señal para alimentar un parlante pequeño y poder escuchar el resultado de la ecualización de la señal.

### 3. Análisis de Requerimientos

#### 3.1 Requerimientos funcionales

##### Señal de entrada

La señal debe poder ser tomada del puerto de salida de audio de un teléfono y leída por el ADC de la EDU-CIAA. Una señal de audio estándar tiene como valores máximos  $\pm 1V$ , por lo que se utilizarán estos valores como los límites para los cálculos del circuito de acondicionamiento. En el caso particular de la salida de audio de un teléfono celular, se tiene que los valores normales son menores a los mencionados previamente. La señal de audio será dentro del rango de frecuencia entre 20Hz a 22kHz, que se corresponde al rango del oído humano.

El ADC de la EDU-CIAA tiene un rango de entrada de 0 - 3.3V (o el valor de VDDA, que no debe exceder 3.6V) por lo que es necesario pre-acondicionar la señal, sumándole una componente de continua para que sea completamente positiva, y amplificando con una ganancia ajustada para tener la mejor calidad de señal posible dentro de los límites del ADC.

Para poder muestrear una señal de hasta 22 kHz, de acuerdo con el teorema de Nyquist necesitamos una frecuencia de muestreo de al menos el doble. El ADC de la EDU-CIAA puede llegar a una frecuencia de muestreo de 400 kHz así que los 44kHz mínimos necesarios no presentarán ningún problema.

##### Señal de salida

El DAC de la computadora también llega a una frecuencia de conversión de datos de 400kHz, así que no esperamos tener problemas de sincronización de la señal.

El DAC llega a un valor máximo igual a VDDA (3.3V) y necesita una carga de al menos  $1k\Omega$ . Se quiere alimentar un parlante chico de una impedancia de  $8\Omega$  a  $32\Omega$  con una señal lo más alta posible, por lo que se amplificará la señal de salida y acondicionará el circuito para proteger el DAC.

##### Procesamiento por software de la señal

Usando como base ejemplos proporcionados por el proyecto CIAA, la señal de entrada se descompondrá en cuatro bandas de frecuencias usando filtros FIR pasa-banda. El filtrado y atenuación de cada banda como así también la reconstrucción de la señal resultante deberá ser lo suficientemente rápido para tener una salida dentro de un tiempo razonable en el parlante, y buffers de la señal de tamaño manejable.

Teniendo un rango de frecuencia de señal de entrada de hasta 22 kHz, se la separará en cuatro bandas: 0 Hz a 2000 Hz, 2000 Hz a 5 kHz, 5 kHz a 10 kHz, y 10 kHz

a 22kHz. Cada banda será atenuada por separado según las preferencias del usuario, y finalmente las cuatro componentes serán sumadas para dirigir la señal procesada a la salida mediante el DAC.

## 3.2 Requerimientos no funcionales

Los requerimientos no funcionales son los siguientes:

- Utilizar la placa de desarrollo EDU-CIAA
- Realizar la programación del sistema con Firmware v2
- Realizar el diseño de la PCB en carácter de poncho y su posterior construcción.
- Realizar el procesamiento digital de audio con algoritmos que maximicen el tiempo de respuesta.
- Realizar la arquitectura del programa de forma que permita la ampliación de bandas del ecualizador.
- Realizar la totalidad del proyecto en un tiempo de desarrollo acotado al cronograma de la asignatura (febrero de 2019), obteniendo un prototipo final para su posterior evaluación.

## 3.3 Interacción con el usuario

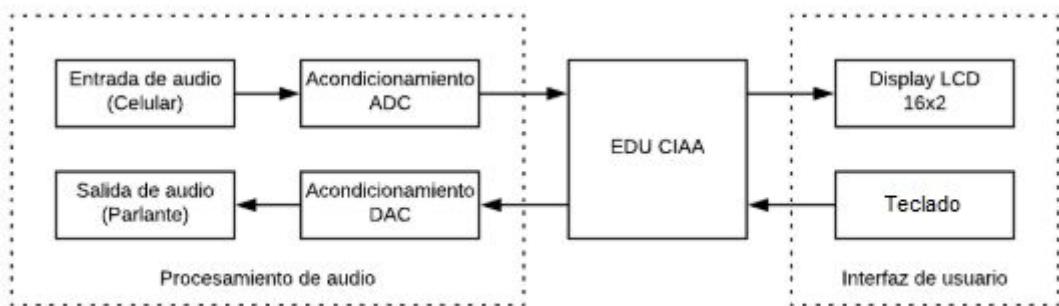
La interfaz con el usuario se dará mediante componentes que ya han sido probados y de uso fiable con la EDU-CIAA por lo que no presentarán desafíos mayores o problemas de requerimientos.

El teclado se usará para recibir el input del usuario. Debe poder elegirse entre las cuatro bandas de frecuencia, y elegir una atenuación para ella entre 0 y -20dB en pasos de 2dB. El teclado consta de cinco botones configurados con resistencias de pull-up de la EDU-CIAA.

La pantalla LCD deberá mostrar el estado actual del ecualizador, es decir la atenuación actual de cada banda. Mientras se está modificando la atenuación de una banda en particular, la pantalla deberá reflejar este cambio.

## 4. Diseño del Hardware

### 4.1 Diagrama general



**Figura 1.** Diagrama en bloques del sistema propuesto.

El proyecto se encuentra dividido en 3 secciones principales las cuales se ilustran en la **Fig 1**. Estas secciones son las siguientes: interfaz de usuario, circuito de señales de audio y la programación de la EDU-CIAA.

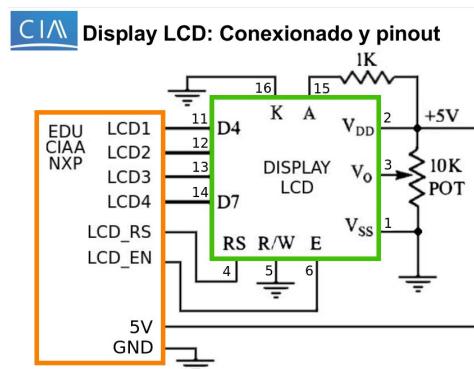
La interfaz de usuario consiste simplemente en un display LCD y un teclado que sirve como herramienta para realizar cambios en el sistema planteado.

Por otro lado contamos con la interfaz de audio, la cual se encarga de acondicionar tanto la señal de audio de entrada como de salida.

Por último, contamos con el cómputo realizado por la EDU-CIAA: administracion de perifericos, manejo de ADC y DAC y procesamiento de la señal a través de filtros digitales.

#### 4.1.1 Display LCD 16x2

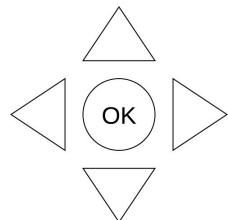
Se utilizará el mismo circuito y conexiones planteadas en el trabajo práctico #1 realizado previamente en la cátedra. Dicho circuito se encuentra plasmado en la **Fig. 2**.



**Figura 2.** Esquemático de conexiones del display LCD.

#### 4.1.2 Teclado

Para el teclado se había planteado utilizar un teclado externo matricial, pero se optó por usar cinco botones para formar un teclado embebido en la PCB del poncho. El diagrama del teclado se puede observar en la **Fig. 3**, y se caracteriza por ser un teclado *navegación de menú* típico.



**Figura 3.** Representación del teclado planteado.

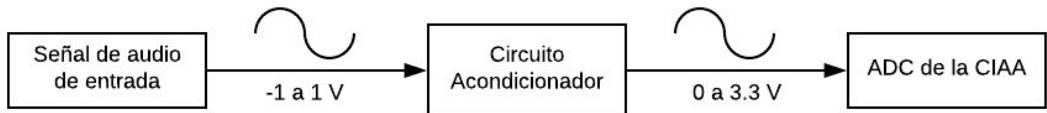
No existe la necesidad de utilizar resistencias de pull-up o pull-down externas, debido a que se pueden configurar los GPIOs con sus propias resistencias internas. Por lo tanto, se contarán con 5 pulsadores conectados a los GPIO de la forma en que se muestran en la **Tabla 1**.

**Tabla 1.** Conexiones de pulsadores con la EDU-CIAA

Tecla	Puerto
Arriba	GPIO8
Izquierda	GPIO1
OK	GPIO5
Derecha	GPIO7
Abajo	GPIO3

#### 4.1.3 Acondicionamiento de señal de audio de entrada

Se necesita acondicionar la señal de audio de entrada para poder ser leída de forma correcta y segura por el ADC de la EDU-CIAA. El circuito acondicionador, en términos generales, se comporta como se muestra en el diagrama de flujo de la **Fig. 4**.



**Figura 4.** Flujo de la señal de audio de entrada.

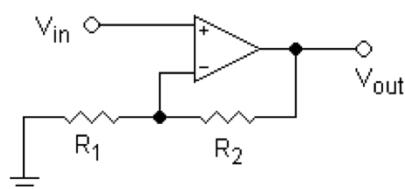
Necesitamos un amplificador operacional que cumpla los siguientes requisitos:

- **Configurado como no-inversor:**
- **Ser Single-Supply:** lo que significa que no requiere de tensiones simétricas para su correcto funcionamiento, lo que permitirá alimentarlo con 0 Volts y 3.3 o 5 Volts.
- **Que tenga un rango de salida de al menos 0 - 3.3 V:** para ser conectado adecuadamente con el ADC.

Inicialmente se había optado por utilizar el integrado *MC33202p* el cual es un amplificador operacional single-supply alimentado con 3.3 Volts y rail-to-rail, lo que significa que permite que la salida alcance voltajes muy cercanos a los de la alimentación y por lo tanto se puede utilizar más eficientemente el rango de tensión bajo el cual operan. Pero luego se decidió descartarlo debido a que en los primeros ensayos no se lograba alcanzar el funcionamiento esperado.

Por lo tanto, se optó por utilizar el amplificador operacional *LM358*, el cual cumple con los requisitos del proyecto y se encuentra disponible en el mercado local. A diferencia del anterior, es alimentado con 5 Volts y no cumple con las propiedades de ser rail-to-rail, lo cual no presenta problema alguno ya que la tensión de salida (con la tensión de alimentación mencionada) alcanza hasta 3.6. Se deben realizar los cálculos necesarios para no llegar a este nivel y lograr proteger la entrada al ADC.

En la **Fig. 5** se muestra la configuración de un amplificador operacional que permite amplificar la señal de entrada sin invertirla.

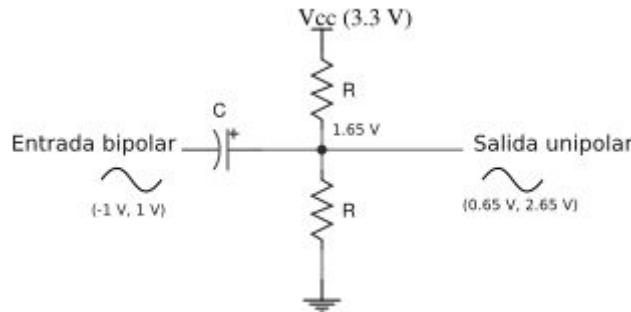


**Figura 5.** Configuración no inversora de un amplificador operacional ideal.

La ganancia dependerá de los valores de resistencia elegidos, la cual se ve definida por la siguiente fórmula:

$$V_{out} = V_{in} \cdot (1 + \frac{R_2}{R_1})$$

Como la señal de audio entrante presenta la característica de ser bipolar, el primer paso a resolver es realizarle un offset para que la señal pase a ser unipolar positiva. Eso puede lograrse simplemente con 2 resistencias de igual valor y un capacitor polarizado como se muestra en la **Fig. 6**.

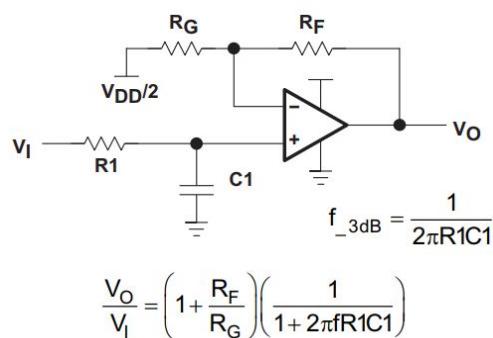


**Figura 6.** Circuito de conversión de señal de audio bipolar a unipolar.

Esto permite desplazar la señal de entrada unos 1.65 Volts, logrando así una señal unipolar de rango 0.65-2.65 Volts. Se podría utilizar esta señal como entrada del ADC, pero se estaría desperdiciando un 40% del rango de resolución, y tampoco se contaría con protección.

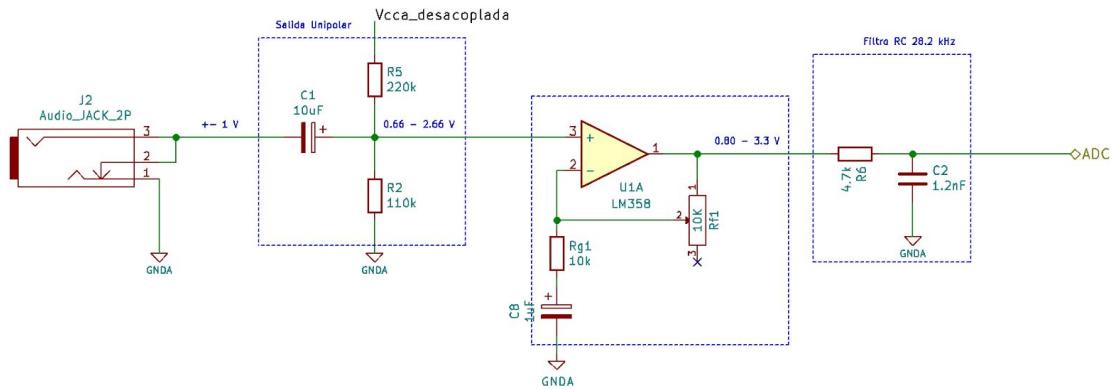
Notar que la impedancia del capacitor se verá definida por  $Z_c = \frac{1}{j2\pi f c}$  lo que para un valor de capacitancia 10 uF y frecuencia 15kHz sería menor a 1 Ω, por lo que es despreciable.

El datasheet del amplificador operacional TLV2374, que tiene similares características que el amplificador seleccionado, recomienda el circuito de la **Fig. 7** para la aplicación que se busca en este proyecto.



**Figura 7.** Circuito de uso típico del tipo de amplificadores operacionales utilizado.

Con todo lo presentado previamente, concluimos en el circuito descripto por la **Fig. 8**. De izquierda a derecha se puede ver: el jack de audio para la señal de entrada, la conversión de la señal bipolar a unipolar, el amplificador operacional configurado como no inversor, un filtro RC pasa bajo con frecuencia de corte de 28 kHz y por último la entrada al ADC.



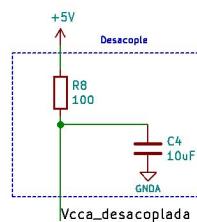
**Figura 8.** Circuito de entrada de audio diseñado par la práctica.

Para calcular la ganancia del amplificador se planteó que cuando a la entrada se tuviese 2.65v se deseaba 3.3v a la salida, por lo tanto:

$$Ganancia = \frac{V_{out}}{V_{in}} = \frac{3.3}{2.65} = 1.245283 = 1 + \frac{R_f}{R_g}$$

Por lo tanto, la señal que entraría al ADC terminaría siendo del rango 0.80 - 3.3 Volts, perdiendo solamente un 24%. Para el cálculo de las resistencias, fijamos  $R_g = 10\text{k}\Omega$  lo que nos daria  $R_f = 2452.8\Omega$ . Se optó por utilizar un potenciómetro de  $10\text{k}\Omega$  para poder ajustar mejor en la practica.

Por último, tanto para la referencia para la conversión de señal bipolar a unipolar como para la alimentación de los amplificadores operacionales, se desacopló la alimentación como se puede observar en la **Fig. 9**. El fin de esto es para mantener la alimentación limpia de los ruidos que podrían generarse durante la ejecución del sistema, y asegurar una referencia para el acondicionamiento de señales estable.



**Figura 9.** Desacople de alimentación.

#### 4.1.4 Acondicionamiento de señal de audio de salida

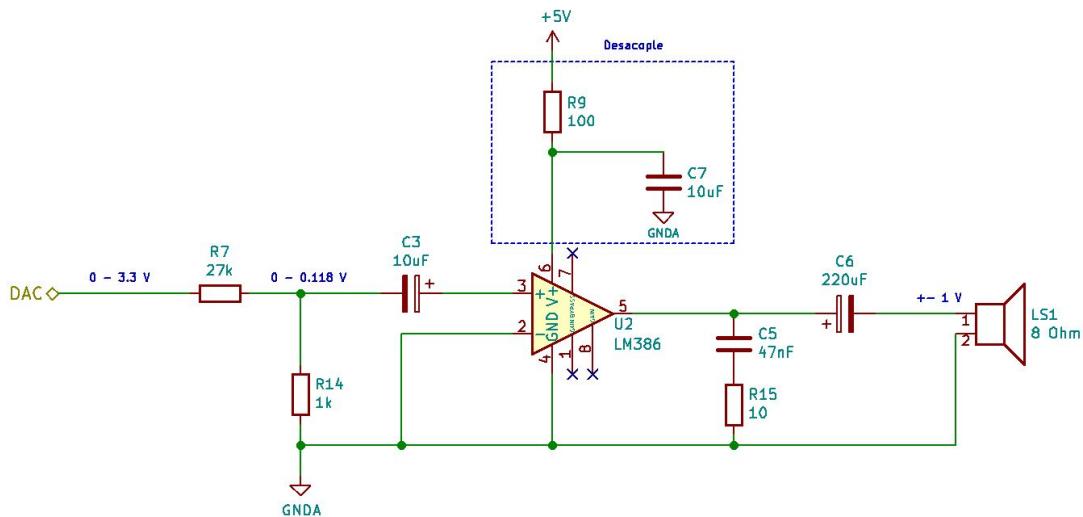
Para poder alimentar un parlante de  $4\text{-}32\Omega$  es necesario amplificar y acondicionar la señal de salida del módulo DAC.

La salida del DAC de la EDU-CIAA toma valores de 0 a VDDA (3.3V), y necesita una carga de al menos  $1\text{k}\Omega$ . Se conecta DAC mediante un divisor de potencia a tierra, con su salida intermedia entrando a un amplificador a través de un capacitor funcionando como filtro pasa-altos. Con resistencias en el orden de  $10\text{k}\Omega$  se llega a corrientes de un máximo de  $330\text{uA}$ , lo que está perfectamente dentro de los límites del DAC.

El amplificador a usar es el LM386, un amplificador de audio de bajo voltaje, que tiene un valor máximo para sus entradas de  $\pm 0.4\text{V}$  y una ganancia interna fija de  $G=20$ .

El amplificador de audio se alimentará con 5V para tener una salida de 0-5V. Ésta pasará por un capacitor filtro pasa-altos para eliminar la componente de continua y alimentará el parlante. Con un capacitor de  $47\text{nF}$  y resistencia de  $10\Omega$  se forma un filtro pasabajos con frecuencia de corte alta para ayudar a disminuir el ruido producido al amplificar una señal de potencia tan baja como es  $0.4\text{V}$ .

La **Fig. 10** ilustra el circuito resultante.



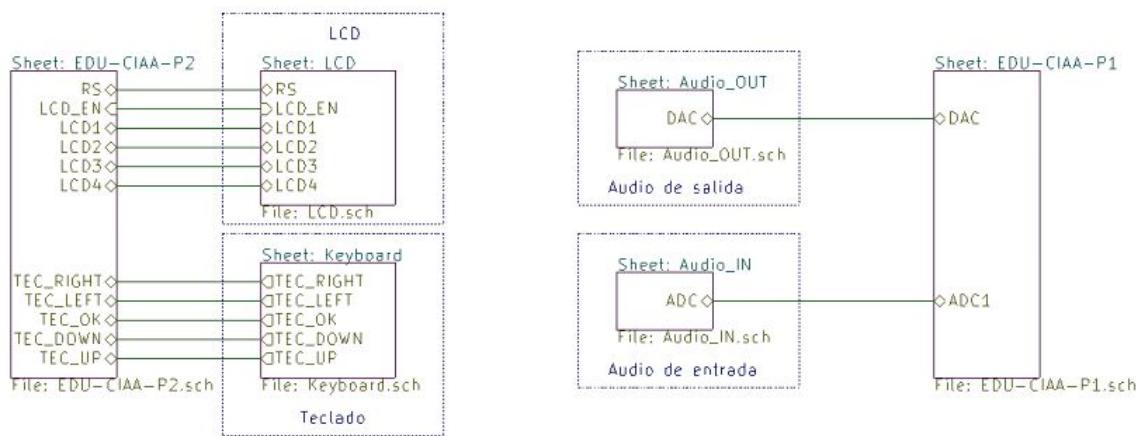
**Figura 10.** Circuito de acondicionamiento de señal de entrada.

## 4.2 Esquemático de conexiones en KiCad

Para llevar a cabo el proceso del diseño del poncho, se utilizó el conjunto de programas de desarrollo de KiCad que permite diseñar esquemáticos de circuitos electrónicos y su posterior conversión a PCBs.

Se optó por realizar una metodología de diseño por diferentes hojas jerárquicas (funcionalidad que provee el programa mencionado), como se lo puede ver en el esquemático principal en la **Fig. 11**. En esta figura se encuentran, las conexiones principales del poncho, en particular las conexiones de cada módulo que lo compone con los puertos de la EDU-CIAA.

Para visualizar los esquemáticos completos de cada parte funcional del poncho diseñado, referenciarse a las hojas adjuntas en el [Anexo B](#).



**Figura 11.** Conexiones principales entre el poncho y la EDU-CIAA.

### Conexiones del poncho con EDU-CIAA

Para explicar las conexiones entre el poncho y la EDU-CIAA, se obviaron los terminales correspondientes a la alimentación.

#### Conecotor P1

Ambos circuitos de audio, tanto el de entrada como el de salida, se conectan exclusivamente con el puerto 1. Las conexiones son las siguientes:

- Audio de entrada: la señal analógica de entrada ya procesada por este circuito se la transmite al pin denominado ADC1 de la EDU-CIAA, para su posterior procesamiento por software.
- Audio de salida: como la EDU-CIAA posee un solo conversor digital-análogo, la conexión con este circuito debe ser estrictamente por el pin denominado DAC.

## Conektor P2

Los componentes restantes del poncho, el circuito del display LCD y el del teclado, se conectan con este puerto.

- Display LCD: la asociación de pines es la que se explicó previamente en la **Fig. 2**.
- Teclado: se utilizaran los pines de GPIO 1,3,5,7 y 8 para conectar a los botones de IZQUIERDA, ABAJO, ACEPTAR, DERECHA y ARRIBA correspondientemente.

## 4.3 Diseño de PCB en KiCad

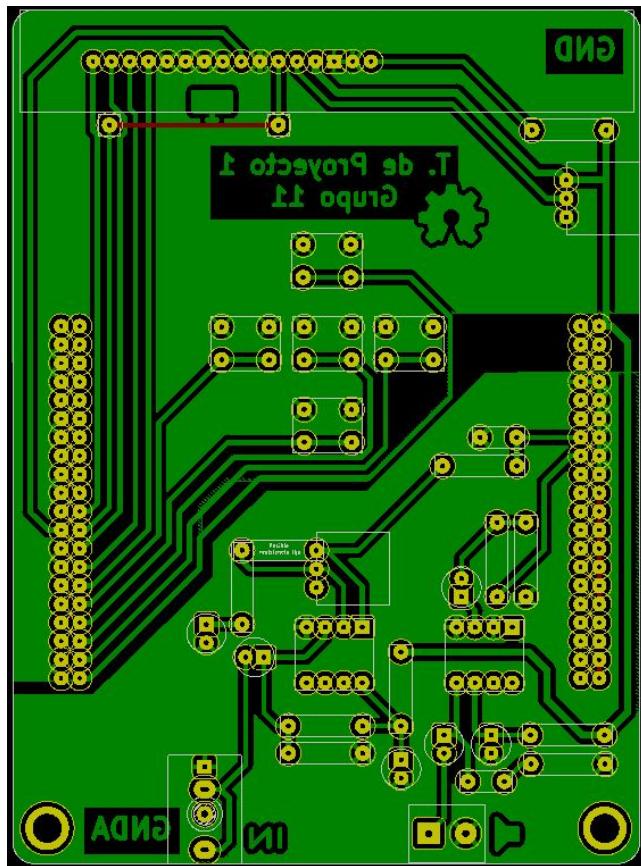
Para el diseño del poncho se utilizó una plantilla de un poncho grande facilitado por el mismo proyecto de la EDU-CIAA. Dicha plantilla se caracteriza por ser una única PCB que conecta los 2 puertos y que tienen dimensiones similares a la pcb de la EDU-CIAA. La desventaja principal, es que cubre los botones existentes, pero al tener nuestro propio teclado en el poncho, no resulta de importancia.

En la **Fig. 12** se puede observar la PCB final, en la cual se puede distinguir las partes principales del circuito. Se hizo énfasis en realizar un diseño ordenado y simplista a la vista, teniendo como objetivo secundario reducir la cantidad de jumpers necesarios. Como se observa, se logró utilizar uno solo (representado por la traza de color rojo). Se centró en utilizar la técnica explicada por la cátedra que consiste en aprovechar los espacios internos de los componentes como camino posible.

En primer lugar, en la parte superior de la imagen se encuentra la sección correspondiente al display LCD. Es de interés mencionar que fue necesario diseñar un footprint para el display LCD a utilizar, debido que la configuración de pines que posee es diferentes a los footprints ya existentes. También se lo hizo para que sirva como guia del tamaño y simetría que ocuparía dentro del poncho.

En el centro, se ubica el teclado que permite la interacción del sistema, con la disposición explicada previamente.

En la parte inferior, se encuentran los componentes asociados al procesamiento de audio, tanto de entrada como de salida. En particular, del lado izquierdo se encuentra el jack de audio por el que se recibira la señal a procesar y del lado derecho, se dispone de la bornera a la cual se conectara el parlante para la salida de audio.



**Figura 12.** Vista de la PCB diseñada en KiCad.

Se puede observar como además se premedito las conexiones con los puertos de la EDU-CIAA y la disposición de los correspondientes componentes, de forma tal que se pueda disponer de 2 áreas de relleno comunes para las referencias de GND y GNDA. Debido a que el puerto 1 es el que posee los pines analogicos de alimentacion, se lo reservo para los circuitos de procesamiento de audio que necesitan de alimentación estable, obteniendo así el campo de GNDA (zona inferior en la figura). Para el caso del puerto 2, se lo eligió para que sea la interfaz de los componentes digitales, tales como el LCD y el teclado, obteniendo así el campo común de GND (zona superior en la figura).

En el [Anexo C](#) se encuentra una imagen que muestra una simulación 3D de lo que sería la PCB a fabricar.

#### 4.3.1 Parámetros de diseño

- Perforaciones de componentes THT de 0.7 mm.
- Pads circulares de 2 mm de diámetro mínimo.
- Margen de tracks y pads mínimo 0.7mm.
- Tracks de 0.7mm de ancho mínimo.
- Utilizar alivio termico para los terminales de GND y GNDA

## 4.4 Listado BOM

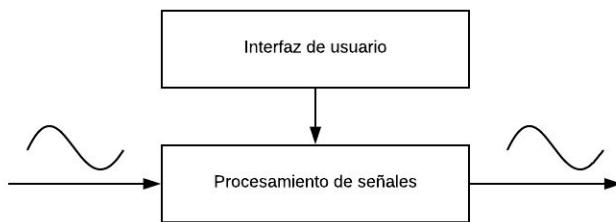
Haciendo uso de las herramientas que KiCad provee, se utilizó en particular la que corresponde a la generación de archivos BOM, que contienen los datos de los componentes del circuito diseñado previamente. Los datos de este archivo se ven plasmados en la **Tabla D1** del [Anexo D](#). Es importante mencionar que en esta tabla se eliminó la columna correspondientes al proveedor del componente, debido a que no le dimos uso para este proyecto, al ser de carácter de prototipado.

## 5. Diseño del software

Para la programación del software, se utilizó principalmente el Firmware v2 junto con algunas funcionalidades del framework SAPI.

### 5.1 Arquitectura principal

A grandes rasgos se puede definir al sistema como dos capas independientes bien marcadas. Por un lado la interfaz de usuario que es la que controla el teclado y el display, y a su vez es la que permite realizar cambios en configuraciones del sistema. Por el otro lado se tiene la capa encargada al procesamiento de la señal la cual no se comunica con la interfaz de usuario, es decir es independiente a ella para su funcionamiento.



**Figura 13.** Capas del sistema.

La capa de procesamiento tiene un único fin: ecualizar la señal de audio de forma constante con las configuraciones de la atenuación de las bandas configuradas al momento. Por lo que una vez iniciado el programa, el procesamiento de audio se producirá siempre y sin interrupción. Los cambios en los valores de atenuación se ven reflejados instantáneamente.

Es de importancia que el sistema tenga un control de tiempo real para lograr hacer el sensado de la señal, a la misma frecuencia con la que fueron diseñados los circuitos digitales a utilizar, para que el cálculo correspondiente no de errores por problema de sincronización.

### 5.2 Capa de interfaz de usuario

Esta capa lleva a cabo las siguientes tareas:

1. **Manejar el teclado:** realiza la lógica de interfaz con el teclado del poncho. Es el que permite realizar cambios en el sistema. Por lo que es el que dispara los cambios de parámetros del sistema.
2. **Manejar el display:** actualizar la pantalla del display cuando sea necesario.
3. **Proveer un sistema de menú:** se encarga de exponer un sistema configurable a través de un menú de usuario, en el que se permitirá realizar cambios tales como

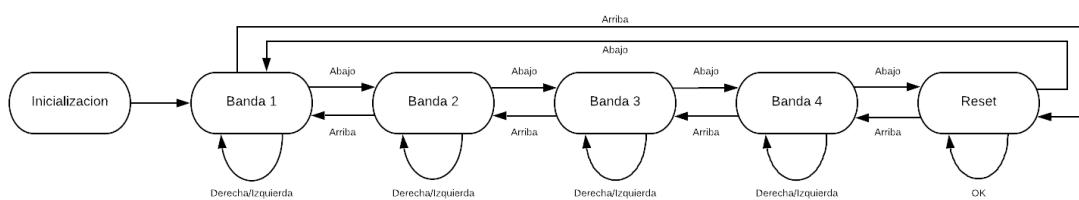
elegir la banda a atenuar, bajar o subir los decibeles, resetear a valores predeterminados, entre otros.

Por lo tanto, la interfaz con el usuario se compone de cinco botones y una pantalla LCD. Los botones están ordenados en forma de cruz direccional, con un botón de “aceptar”/“OK” en el medio. Dependiendo del input del usuario mediante los botones, el display LCD mostrará el menú de opciones con las diferentes bandas y sus respectivas atenuaciones.

Dicho menú se compone de 5 opciones:

- Modificar atenuación de banda 1.
- Modificar atenuación de banda 2.
- Modificar atenuación de banda 3.
- Modificar atenuación de banda 4.
- Resetear todas las bandas.

Mediante la acción de las teclas de Arriba/Abajo se puede desplazar por las opciones mencionadas. Una vez elegida una banda, se le permite modificar la atenuación con las teclas de Izquierda/Derecha. Luego de la inicialización del sistema se pasa directamente a la selección de la primera banda de frecuencia. En la **Fig. 14** se muestra la transición de opciones del menú mencionadas.



**Figura 14.** Estados del menú de usuario.

Debido a que la pantalla LCD solo posee dos filas, se muestra en la primera la opción actual sobre la que se está desplazando junto a la atenuación actual de la banda, si lo corresponde, y en la segunda fila se muestra la opción que le continua. De forma adicional y para dejarle claro al usuario la opción seleccionada, se le agrega un indicativo de la forma “>>” antes de la selección. En la siguiente figura se observa lo detallado.

>	>	B	a	n	d	a		1		-	1	2		d	B
B	a	n	d	a		2									

**Figura 15.** Visualización del menú en el display.

Cualquier accionar de las teclas del teclado provoca un refrescado de la pantalla LCD para poder ver los cambios visualmente como así también la modificación del nivel de atenuación de la banda seleccionada genera un cambio instantáneo en la salida actual de la señal.

Seleccionar la opción de “Reset” y presionando la tecla OK, genera que se restaure el nivel por defecto de atenuación de las bandas, el cual es 0dB en todas.

La capa de procesamiento de señal, al tener que realizar intensos cálculos en tiempo real, necesita casi la totalidad de tiempo del procesador. Mientras que la velocidad de respuesta al presionado de botones de la interfaz de usuario no necesita de una frecuencia de lectura del estado de los botones tan alta: una espera de hasta 50-100ms es aceptable para los tiempos de respuesta humanos. Como consecuencia se tiene una tarea de conversión y output de datos de audio muy frecuente y una de interfaz con el usuario que ocurre con mucha menos frecuencia.

## 5.3 Capa de procesamiento de señal

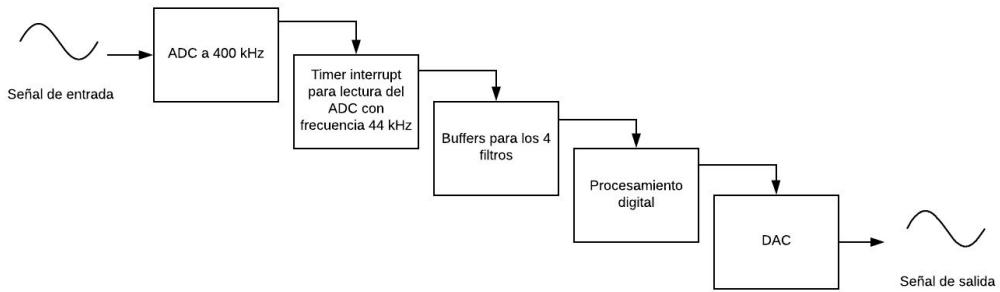
Esta capa lleva a cabo las siguientes tareas:

1. **Procesamiento digital de entrada:** procesa la señal que proviene del ADC. Dependiendo de las configuraciones actuales del sistema, realizará las atenuaciones acordes.
2. **Procesamiento digital de salida:** toma la señal procesada de la tarea anterior y realiza la lógica necesaria para controlar al DAC y exponer la señal de audio.

Inicialmente se planteó desarrollar esta capa basándose en una máquina de estados finita, pero debido a que se optó por mantener la lógica de procesamiento corriendo constantemente en el *background* y a la simpleza de las interacciones con el usuario, se decidió no realizarlo de esa manera. Los cambios en el sistema simplemente se ven reflejados al realizar acciones por el teclado, lo que produce los cambios instantáneos en los parámetros de las bandas.

Como se quiere muestrear con el ADC la señal de entrada con frecuencias de hasta 22 KHz, por el teorema de muestreo de Nyquist-Shannon se sabe que se necesita una frecuencia de muestreo de al menos 44 kHz. Como se mencionó previamente esta frecuencia no genera problema alguno ya que la frecuencia máxima de muestreo del ADC es de 400 kHz. El verdadero problema que se tuvo que considerar, que se verá justificado luego en la sección ensayos, es los límites de tiempo que se tiene para el procesamiento de la señal. Con una frecuencia de 44 kHz resulta que el tiempo en el que se debería realizar el procesamiento de la señal mediante los 4 filtros es de 22,72 uS. Si el procesamiento tardase más que el tiempo estimado, se tendría una pérdida de calidad en la señal de salida.

El flujo de operación de esta capa se puede ver en la siguiente **Figura 16**:



**Figura 16.** Flujo de la capa de procesamiento digital.

Para realizar la lectura de la señal de entrada, inicialmente se configura el ADC a la frecuencia máxima de conversión, 400 kHz, y se le indica que inicie la conversión del dato. Se configura un Timer que dispara una interrupción a una frecuencia estipulada igual a la frecuencia de muestreo necesaria, 44 kHz. Dicha interrupción cumple la siguiente función:

- 1) Leer el resultado de la conversión.
- 2) Colocarlo el valor dentro de los 4 buffers correspondientes a cada filtro.
- 3) Indicar al ADC que inicie la nueva conversión, de manera que a la siguiente iteración de la rutina de interrupción el valor ya se encuentra disponible. Esto se puede asegurar, debido a que la velocidad de conversión del ADC (400 kHz) es un orden de magnitud mayor que la de la frecuencia de dicha interrupción (44 kHz).
- 4) Indicar con un flag que el nuevo valor está listo para ser procesado por el ecualizador. De esta forma se libera inmediatamente la interrupción pasando a ejecución normal del procesador.

En conclusión, el ADC está configurado para funcionar a su velocidad máxima, pero no en modo continuo sino que se le indica el inicio de conversión mediante la interrupción de Timer funcionando a 44 kHz. El resultado se lee en la siguiente iteración, asegurando la finalización de su resultado y de forma no bloqueante.

Por el otro lado se tiene la rutina correspondiente al procesamiento de la señal, que se ejecuta al leer la activación del flag previamente mencionado. Dicha rutina se encarga de:

- 1) Realizar el cálculo de los 4 filtros.
- 2) Aplicar las atenuaciones correspondientes a cada banda.
- 3) Reconstruir la señal.
- 4) Por último, transmitir el valor resultante al módulo DAC de la EDU-CIAA para su escritura como señal de salida.

## Atenuaciones

Se debe permitir al usuario atenuar las bandas desde 0 dB hasta -20 dB con pasos de 2 dB. Se realizó el cálculo previo de las constantes por las cuales se debe multiplicar la amplitud de la señal original para obtener dichas atenuaciones, con la siguiente fórmula:

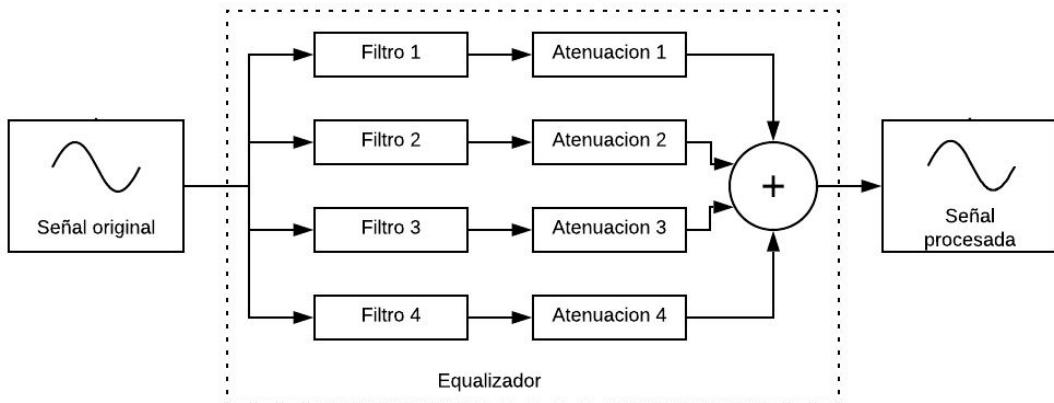
$$x \text{ dB} = 20 \cdot \log_{10} \frac{\text{Señal salida [V]}}{\text{Señal entrada [V]}}$$

Y la tabla resultante es la siguiente:

**Tabla 2.** Tabla de constantes de atenuación.

Atenuación [dB]	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20
Constante	1	0.794	0.631	0.501	0.398	0.316	0.251	0.199	0.158	0.126	0.1

## Composición de la señal resultante



**Figura 17.** Flujo del procesamiento de la señal en el ecualizador.

La forma en que se procesa la señal se encuentra graficada en la **Fig 17**. El proceso consiste en diseccionar la señal en las diferentes bandas mediante los filtros digitales FIR, lo cual permite aplicar las atenuaciones por separado sin afectar las otras bandas y por último realizar la suma de las 4 componentes obteniendo así la señal procesada final.

En el caso que se tenga filtros de alta resolución y que no se le aplique atenuación en ninguna de las bandas la señal resultante será la misma que la señal original.

Para llevar la aplicación en tiempo real de los filtros FIR, se sigue la definición de su ecuación en diferencias, la cual expresa que para obtener la salida solo se

necesitan entradas actuales y anteriores. Si definimos  $y(n)$  como el resultado de la señal filtrada,  $N$  el número de coeficientes del filtro,  $h(k)$  los coeficientes de la respuesta impulso del filtro y  $x(n)$  la señal de entrada, tenemos:

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot x(n - k)$$

Por lo tanto el algoritmo que sigue cada filtro es de la siguiente forma:

```

acumulador = 0
k = 0
recorrer el buffer{
    aux = buffer [i] * coeficientes [k]
    acumulador = acumulador + aux
    k++
}
valor_filtrado = acumulador

```

Este simple procedimiento es el que conlleva la mayor parte del tiempo de CPU ya que son alrededor de 100 multiplicaciones y sumas de números de 64 bits, por cada uno de los filtros. Para la variable **acumulador** se utiliza una representación de 64 bits, denominada **long long** en código C, debido a que se debe realizar multiplicaciones binarias con los coeficientes representados en 32 bits.

## 5.4 Filtros

Para el procesamiento de la señal filtrada se utilizó como base un ejemplo ya existente dentro del Firmware v2 denominado “*adc\_fir\_dac*” y aportado por el Ing. Ridolfi del proyecto CIAA. Hay que mencionar que fue de gran importancia contar con este antecedente debido a que todo el algoritmo del procesamiento de la señal se encuentra escrito en código assembler permitiendo así el mínimo tiempo de retardo para su procesamiento.

En nuestro caso particular se tuvo que diseñar 4 filtros FIR digitales, los cuales cumplen con su función a través de una retroalimentación por lo que se debe contar con buffers de la historia de la señal. Para ello se utilizó la herramienta provista por el siguiente link <http://t-filter.engineerjs.com/>, la cual permite ingresar los parametros deseados, crear los filtros con la aproximación más cercana y visualizar la función de ganancia con respecto a la frecuencia de la señal.

En particular se desean los filtros correspondientes a las siguientes bandas: 0 a 2 kHz, 2 kHz a 5 kHz, 5 kHz a 10 kHz y 10 kHz a 22 kHz. Se podría diseñar los filtros de forma que se comporten de la manera más cercana a un filtro ideal para cumplir con

estas bandas, pero debido a las características del proyecto que tiene como requisito poder ser ejecutado con un procesador de 204 MHz y en tiempo real, se tuvo que diseñar los filtros con poca calidad para tener un menor número de coeficientes del filtro resultante.

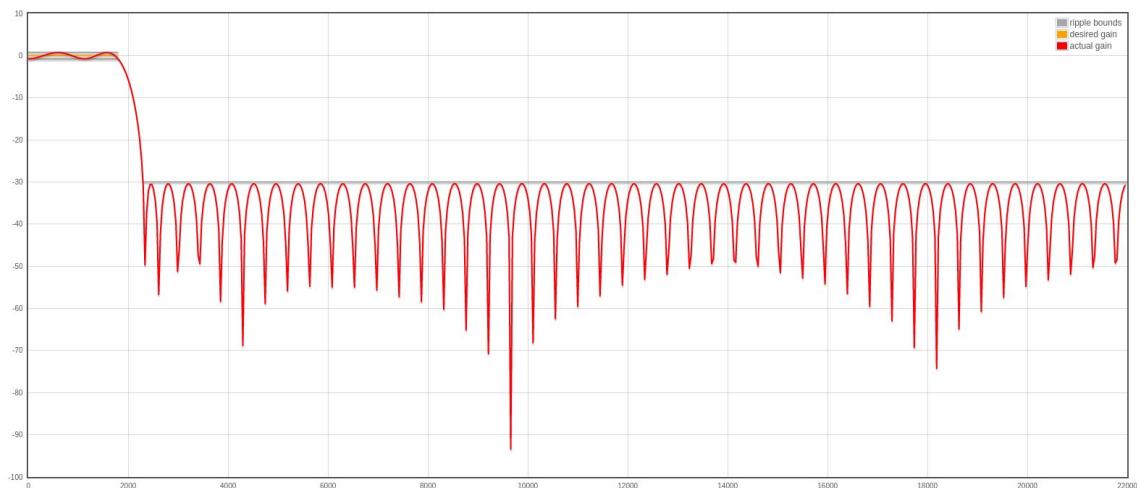
En particular todos los filtros obtenidos cumplen con los siguientes parámetros:

- Ripple Effect de no más de 2 dB en la banda de frecuencias que deja pasar.
- Dejar pasar el rango de frecuencias con ganancia 1.
- Rechazar bandas con una atenuación de -30dB, generando así menor cantidad de coeficientes pero al mismo tiempo dejando un residuo que no es mínimo (un 4% de la amplitud original).
- Que en la zona de frecuencias en las que intersectan 2 filtros, se logre una suma total lo más cercano a lo que corresponde a la señal original. Esto significa por ejemplo que en el punto exacto de intersección y para cada filtro se tenga una atenuación del 50%.
- No superar los 100 coeficientes, ya que traería retardos en el procesamiento.

#### 5.4.1 Filtro 1

Este filtro en particular se comporta como filtro pasa-bajo y se lo puede visualizar en la **Fig. 18**. La gráfica representa en rojo a la ganancia real del filtro, en gris los valores límites estipulados y en naranja la ganancia deseada para las frecuencias que no son filtradas. Los parámetros con los cuales fue diseñado fueron:

- Dejar pasar la banda de 0 - 1800 Hz.
- Rechazar bandas de 2300 - 22000 Hz.



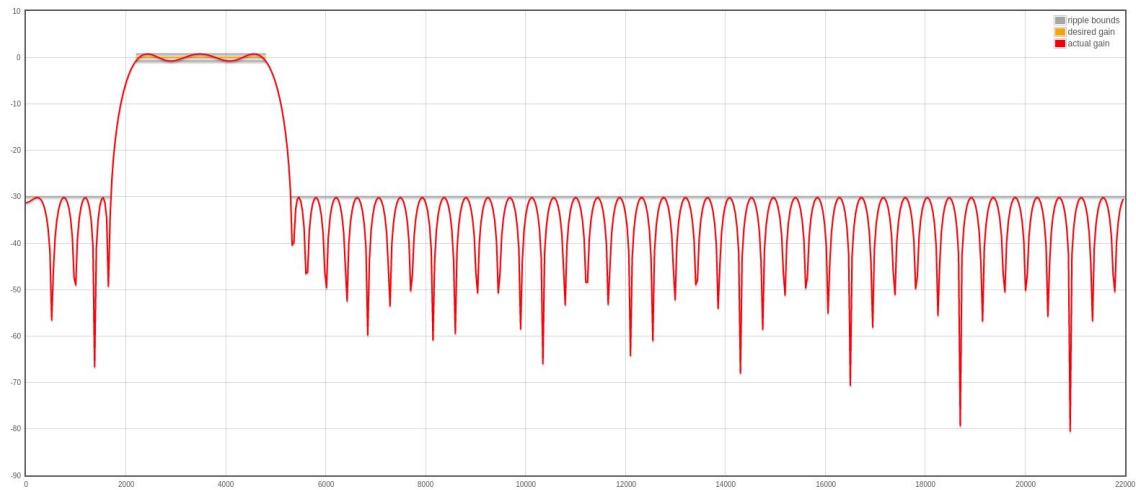
**Figura 18.** Función de ganancia del filtro 1 diseñado.

Se obtuvo 99 coeficientes.

### 5.4.2 Filtro 2

Este filtro en particular se comporta como filtro pasa-banda y se lo puede visualizar en la **Fig. 19**. Los parámetros con los cuales fue diseñado fueron:

- Rechazar bandas de 0 - 1700 Hz.
- Dejar pasar la banda de 2200 - 4800 Hz.
- Rechazar bandas de 5300 - 22000 Hz.



**Figura 20.** Función de ganancia del filtro 2 diseñado.

Se obtuvo 101 coeficientes.

### 5.4.3 Filtro 3

Este filtro en particular se comporta como filtro pasa-bajo y se lo puede visualizar en la **Fig. 21**. Los parámetros con los cuales fue diseñado fueron:

- Rechazar bandas de 0 - 4700 Hz.
- Dejar pasar la banda de 5200 - 9800 Hz.
- Rechazar bandas de 10300 - 22000 Hz.



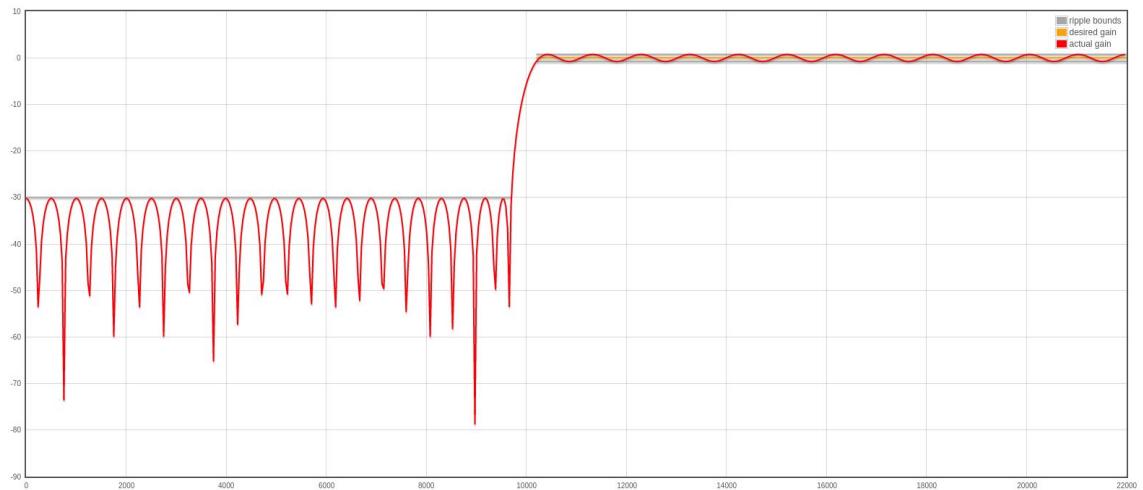
**Figura 21.** Función de ganancia del filtro 3 diseñado.

Se obtuvieron 99 coeficientes.

#### 5.4.4 Filtro 4

Este filtro en particular se comporta como filtro pasa-alto y se lo puede visualizar en la **Fig. 22**. Los parámetros con los cuales fue diseñado fueron:

- Rechazar bandas de 0 - 9700 Hz.
- Dejar pasar la banda de 10200 - 22000 Hz.



**Figura 22.** Función de ganancia del filtro 4 diseñado.

Se obtuvieron 93 coeficientes.

## 5.5 Código del proyecto

El código total del proyecto se puede obtener en el [repositorio del proyecto](#). En particular la estructura de código es la siguiente:



**Figura 23.** Archivos del proyecto

- **db\_table:** contiene las constantes de atenuación calculadas previamente.
- **equalizer:** módulo correspondiente a la capa de procesamiento de la señal de audio. Es el que maneja el flujo de datos entre el ADC, los filtros y el DAC.
- **filterX:** archivos correspondientes a cada filtro que contiene las constantes obtenidas tras su diseño.
- **ui:** módulo correspondiente a la capa de interfaz de usuario, para manejar el teclado, refrescar la pantalla LCD y realizar cambios sobre los parámetros del **equalizer**.
- **fir\_q31:** código que define una estructura de tipo filtro y sus comportamientos: agregar y quitar valores al buffer de historia de la señal y el cálculo de la salida del filtro.
- **asm\_fir\_q31:** implementación de las funciones anteriores pero en código assembler correspondiente al microprocesador de la EDU-CIAA. Permiten velocidades de cálculo superiores.

### 5.5.1 Main

El **main** simplemente se encarga de la inicialización del módulo correspondiente a la interfaz de usuario, el correspondiente al procesamiento del ecualizador y a la placa propiamente dicha. Luego se encarga de mantener la ejecución constante de las rutinas de cada módulo. Por lo tanto sigue la siguiente forma:

```

boardConfig()
equalizer_init()
ui_init()
while(infinito){
    equalizer_loop()
    if(fin delay 50 ms no bloqueante)
        ui_loop()
}

```

Las funciones mencionadas se explicaran a continuación.

### 5.5.2 Equalizer

Este módulo posee los 4 filtros diseñados con sus buffers correspondientes y las atenuaciones actuales de cada banda. Como se explicó previamente tiene lógica para el manejo del ADC y DAC y se encarga del procesamiento de audio en tiempo real.

Posee tres funciones principales: `equaizer_init()`, `equalizer_loop()` que es llamada constantemente por el `main` y `sampling_interrupt()` ejecutada constantemente a 44 kHz.

```

funcion equalizer_init()
{
    inicializar la estructura de los 4 filtros
    inicializar DAC
    inicializar ADC con velocidad 400 kHz
    indicar inicio de conversión ADC
    inicializar un Timer a 44kHz
}

```

```

funcion sampling_interrupt()
{
    //manejador correspondiente al timer de 44kHz
    value = leer valor del ADC
    indicar inicio de nueva conversión ADC
    colocar "value" en los buffers de cada filtro
    adc_sample_flag = 1
}

```

```

funcion equalizer_loop()
{
    if(adc_sample_flag == 1){
        adc_sample_flag = 0
        realizar el procesamiento de filtrado
        aplicar atenuacion en cada banda
        construir señal de salida
        escribir en el DAC
    }
}

```

### 5.5.3 UI - User Interface

Es el módulo que implementa la capa de interfaz de usuario, por lo que se encarga del control del display LCD y del input del usuario mediante el teclado.

Como se detalló previamente en “5.2 Capa de interfaz de usuario”, implementa un menu de seleccion de banda para poder modificar su atenuación correspondiente. Para ello se podría haber implementado una estructura de máquina de estado propiamente dicha, pero para acelerar el desarrollo del proyecto simplemente se mantiene registro de la selección actual mediante una variable y acorde a esta realizar las modificaciones pertinentes según la interacción con el teclado.

Posee 2 funciones principales: `ui_init()` y `ui_loop()` la cual se llama por el loop del `main` cada 50 milisegundos, tiempo suficiente para la correcta interacción con el usuario y suficientemente grande para no generar retrasos en el procesamiento de audio.

```

funcion ui_loop()
{
    comprobar interacción del teclado
    if(se apretaron teclas DOWN o UP){
        cambiar selección actual de opción del menú
    }else if(se apretaron RIGHT o LEFT){
        modificar la atenuación de forma acorde
    }else if(se apretó OK con selección RESET){
        borrar modificaciones de atenuaciones
    }
}

```

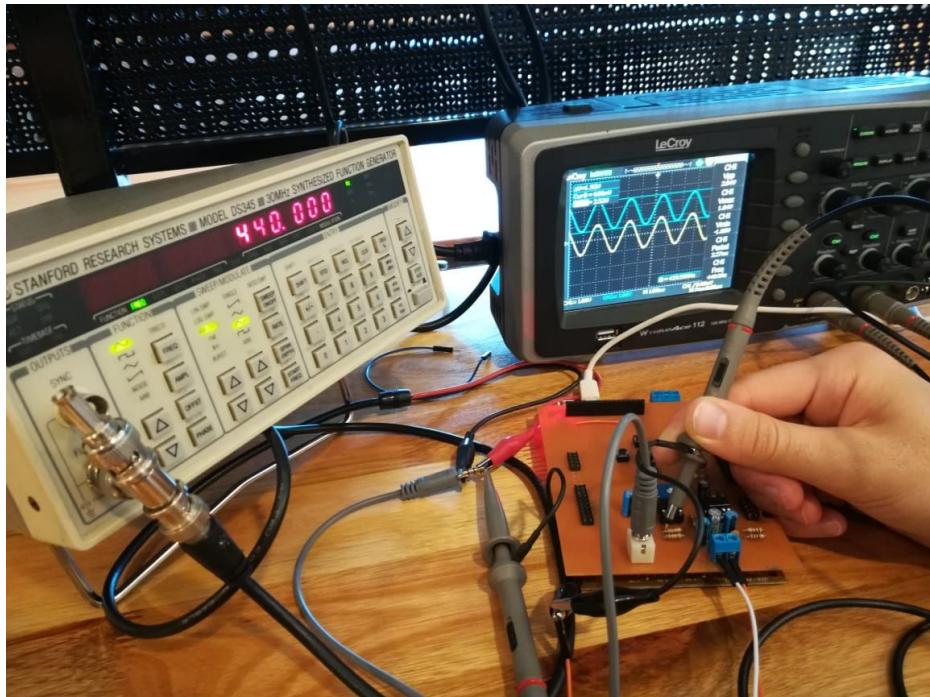
```
funcion ui_init()
{
    inicializar LCD
    inicializar los GPIOs de las teclas
}
```

## 6. Ensayos y mediciones

En primer instancia se realizaron los prototipos de los circuitos aislados sobre una protoboard para asegurar su funcionamiento correcto y determinar si era necesario efectuar algún cambio. El único cambio importante que realizamos fue el del cambio del amplificador MC33202p por el LM358.

Tras haber realizado los re-diseños pertinentes según estos primeros ensayos y habiendo confirmado su correcto funcionamiento como un sistema en conjunto se pasó a la etapa de construcción de la PCB. Los resultados se pueden observar en el [Anexo F - Circuito impreso final](#).

Por último se realizaron los ensayos y mediciones sobre la placa final, las cuales se detallaran a continuación. Para ello se hizo uso de un generador de señales produciendo una sinusoida de  $+1\text{ V}$  de amplitud, valores máximo para una señal de audio, y modificando los rangos de frecuencia para observar el comportamiento. Con un osciloscopio de dos canales se fueron registrando las mediciones. En la **Fig. 24** se ve un ejemplo de ensayo.

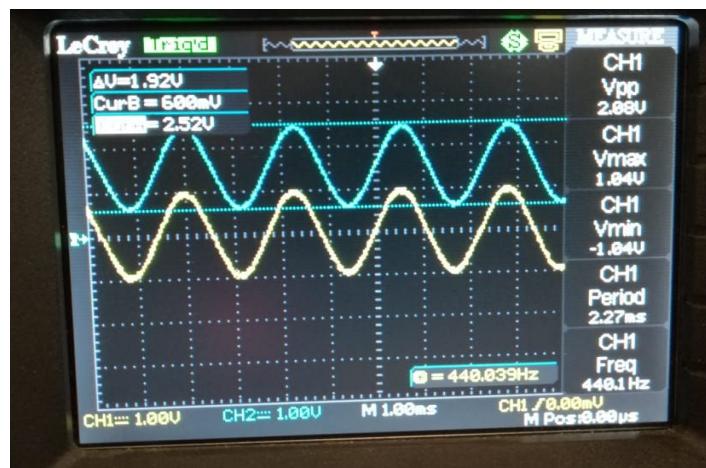


**Figura 24.** Metodología de ensayo sobre el poncho y EDU -CIAA.

## 6.1 Acondicionamiento de señal de entrada

Se comenzó haciendo pruebas sobre una protoboard, usando el amplificador operacional MC33202p, pero aún con ayuda de los profesores encargados del ATEI no se lograba conseguir una buena respuesta y se tenía demasiada distorsión a frecuencias relativamente bajas. Por consenso con otro grupo que se encontraba en una situación similar y el profesor de la cátedra, se terminó cambiando por el amplificador operacional LM358, que resultó tener una respuesta apropiada para nuestro rango de frecuencias y amplitudes. Así se decidió en un diseño final del circuito de acondicionamiento de señal de entrada.

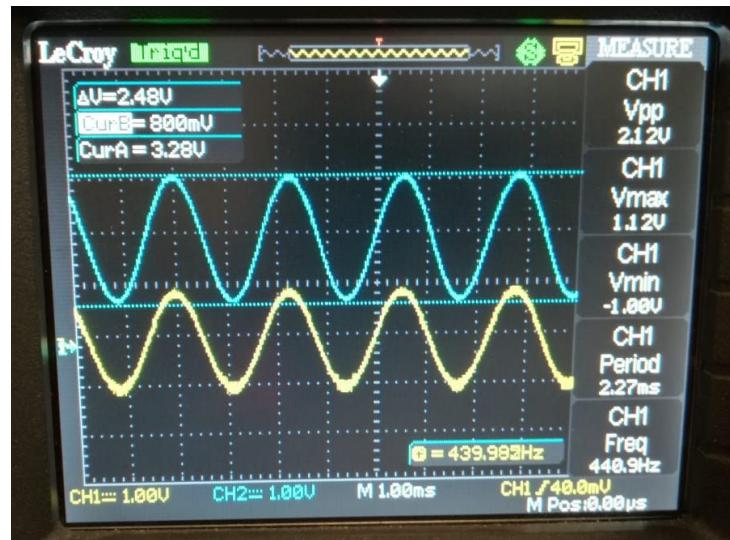
A continuación se detallan los ensayos hechos sobre el circuito de acondicionamiento de señal de entrada, una vez se tuvo el proyecto completo ensamblado y funcionando:



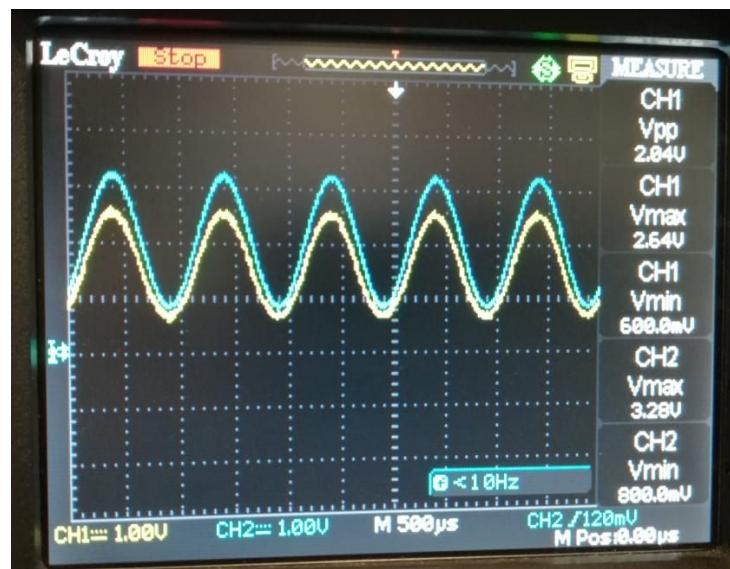
**Figura 25.** Entrada de audio y conversión unipolar.

Se muestra la comparación entre la señal generada como entrada de audio (amarillo) y la señal luego de la conversión unipolar (azul). Se produce un offset al agregar una componente de continua de  $\sim 1.66V$ , la amplitud se mantiene y la señal pasa de  $\pm 1V$  a  $0.6V-2.6V$ . Este coincide con los cálculos realizados previamente.

Luego en la figura **Fig. 26** se muestra en azul la señal obtenida a la salida del LM358. Se ajustó y midió el potenciómetro, cambiando la ganancia, hasta encontrar que la salida alcance 3.3V, y obtuvimos un valor de  $2.55k\Omega$ , lo que se acerca al valor teórico calculado previamente de  $2.45 k\Omega$ .



**Figura 26.** Salida del amplificador.



**Figura 27.** Entrada vs salida del amplificador.

En la **Fig. 27** se ve el mismo punto de salida del amplificador, comparado con su entrada directa. La señal pasa de aproximadamente 0.6V - 2.64V a 0.8V - 3.3V, acercándose a los límites del ADC de la CIAA.

Entre el amplificador y la entrada al ADC se encuentra un circuito RC funcionando como filtro pasa bajo, para filtrar las frecuencias altas que no son necesarias para este proyecto. En la **Fig. 28** se muestra la respuesta obtenida durante los ensayos probando con diferentes frecuencias. Como es lo esperado, a frecuencias más altas existe una atenuación de la señal.



**Figura 28.** Filtro RC a la entrada del ADC con frecuencias de entrada 10kHz (izquierda) y 30kHz (derecha).

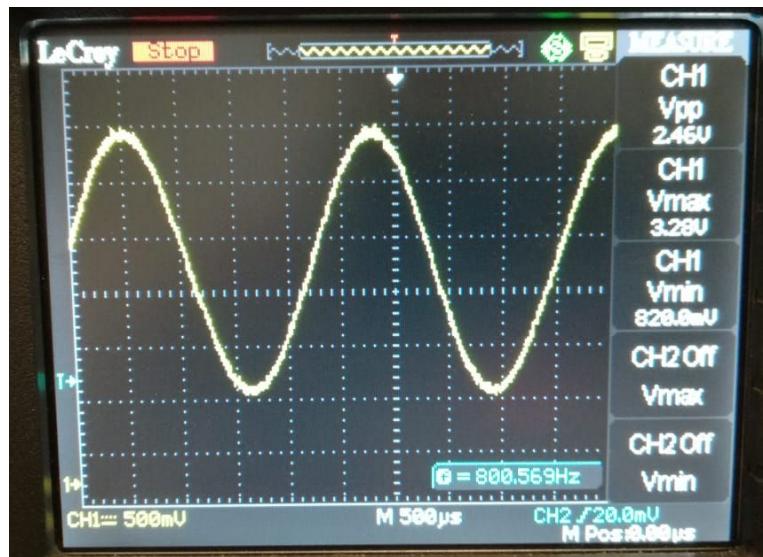
Se puede apreciar una distorsión de la señal a 30kHz que no supimos atribuir a más que una mala respuesta en frecuencia del circuito de entrada, pero como no es una frecuencia en el rango que debíamos manejar no importó al proyecto y sólo queda como anécdota.

En ensayos descritos en las siguientes páginas se estableció cuál es el límite real de frecuencias que puede manejar el ecualizador, que es mucho menor a los 30kHz donde encontramos la distorsión.

## 6.2 Acondicionamiento de señal de salida

El circuito de salida de audio también comenzó probándose aislado en una protoboard. Al principio tuvo problemas de distorsión y saturación; la primera resuelta al desacoplar la alimentación del amplificador con un filtro, y la segunda cambiando los valores del divisor de tensión en la salida del DAC por tener una ganancia inesperadamente grande. Una vez hechos estos cambios, el circuito funcionó sin problemas y se decidió que el diseño sería final.

A continuación, los ensayos realizados una vez completado el PCB, ensamblado el proyecto y corriendo el sistema de ecualizador de audio.



**Figura 29.** Salida del DAC con frecuencia de entrada de 800 Hz.

Se hicieron los ensayos siguientes del circuito de salida con una señal de entrada que ronde los valores máximos para probar con la mayor amplitud que tendremos en el DAC, de aproximadamente 0.8V a 3.3V. Por lo tanto se ve como la amplitud de salida del DAC es la misma que la señal a la entrada del ADC. El ecualizador en condiciones normales no provoca ningún tipo de atenuación.



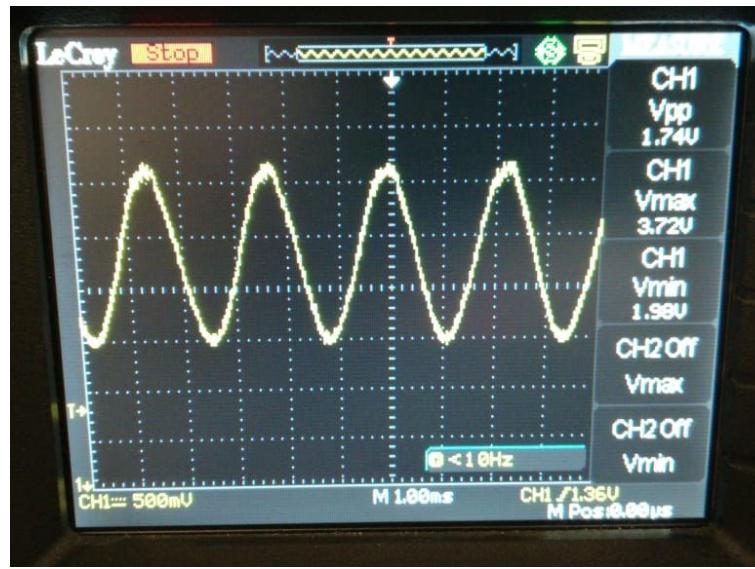
**Figura 30.** Salida del divisor de tensión.

El primer paso en el circuito de salida es pasar por un divisor de tensión que limite la tensión de salida y presente una carga al DAC de forma que no exceda su límite de corriente de salida. Las resistencias de  $27\text{k}\Omega$  y  $1\text{k}\Omega$  dan como resultado una señal pequeña con bastante ruido relativo y que el instrumento muestra estar acotada por 2mV y 150mV, a diferencia de los 29mV a 118mV esperados.



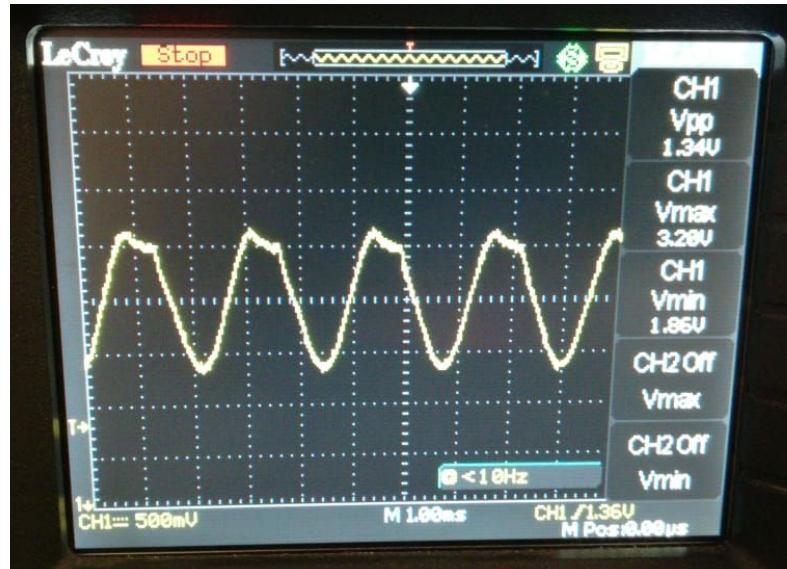
**Figura 31.** Entrada del amplificador de audio.

Entre el divisor de tensión y la entrada del amplificador de audio hay un capacitor que funciona como un pequeño filtro pasa altos para eliminar la componente de continua de la señal de salida, ya que el amplificador acepta como máximo absoluto una entrada de  $\pm 0.4\text{V}$ . Por culpa del ruido de la señal, el instrumento muestra unas cotas de -38.4mV a 69.6mV, a diferencia de los  $\pm 45\text{mV}$  esperados.



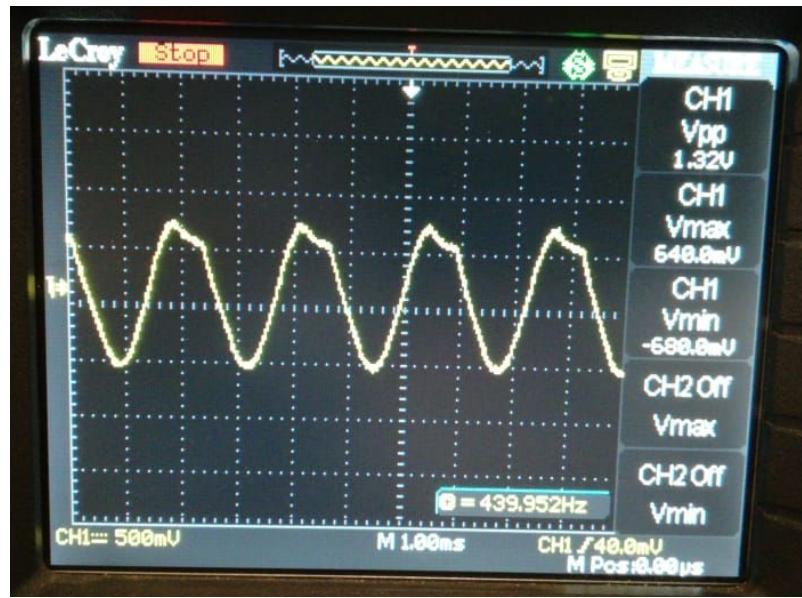
**Figura 32.** Salida del amplificador sin carga.

A la salida del amplificador y sin una carga, nos encontramos con una señal de mucho mayor amplitud de la que se prevé con la ganancia de  $G=20$ , lo que no pudimos atribuir a algún efecto del circuito en particular. En este punto hay un filtro pasabajos que busca eliminar el ruido introducido en la etapa de división de tensión antes de llevar la señal al parlante.



**Figura 33.** Salida del amplificador con carga.

Conectando una carga (parlante) al final del circuito se puede notar una distorsión en la señal que parecía ser proporcional al valor de la impedancia interna del parlante.



**Figura 34.** Señal de entrada al parlante.

Antes del parlante existe otro filtro pasa-altos para centrar la señal en 0V eliminando la componente de continua. En **Fig. 34** se muestra que al final del acondicionamiento de la señal, el parlante está recibiendo una señal de aproximadamente  $\pm 600\text{mV}$ .

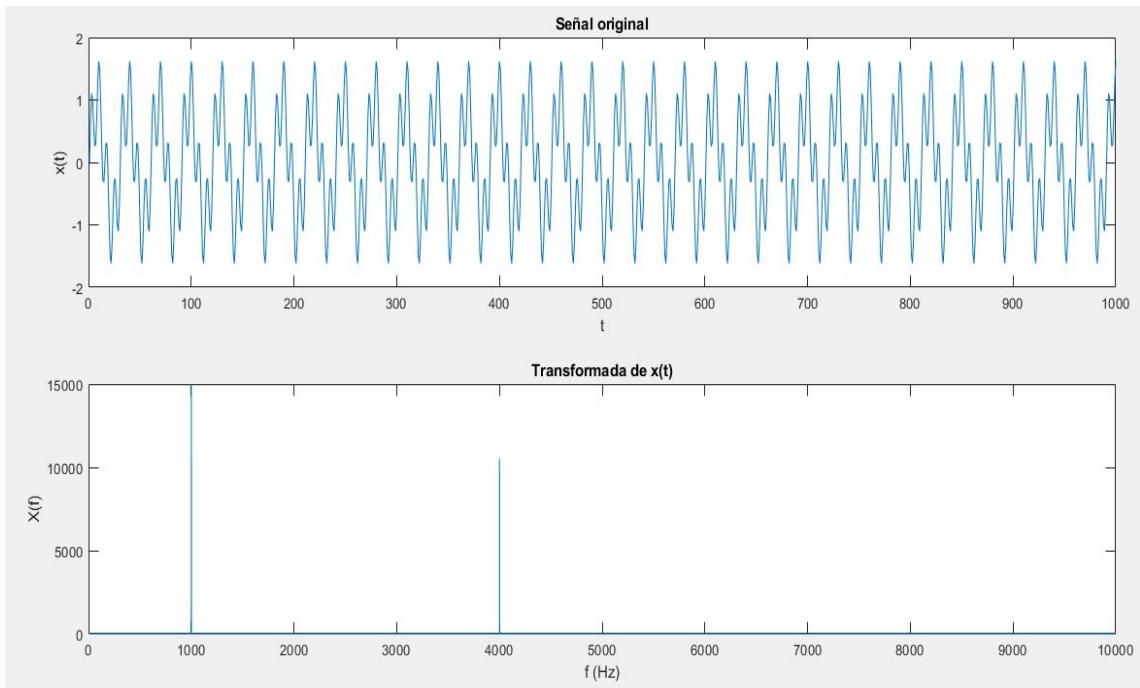
## 6.3 Simulación de filtros en MatLab

Para la utilización de los filtros digitales requeridos en el proyecto utilizamos el programa Matlab para realizar simulaciones del funcionamiento de los mismos con señales senoidales. Además, Matlab permite crear filtros digitales con el comando “filterDesigner” con el cual armamos los filtros para este trabajo. Otra herramienta que se utilizó como base en la prueba de filtros fue la que se encuentra en la página: <http://t-filter.appspot.com>. Uno de los ejemplos provistos por la cátedra, denominado “adc\_fir\_dac”, en la cual se utiliza un filtro digital, se creó con esta página, que ofrece un entorno más sencillo que Matlab para la creación de filtros digitales, y nos da la posibilidad de elegir el tipo de los datos de los coeficientes para armar el filtro. En nuestro proyecto utilizamos esta página para el armado de los filtros digitales, con sus coeficientes de tipo int y la precisión en punto fijo para esos coeficientes de 32 bits.

Para la etapa de filtrado se hicieron simulaciones sobre el software Matlab para corroborar el funcionamiento de los filtros digitales a señales generadas a una frecuencia determinada. Para saber si el filtro digital era adecuado para las señales de entrada aplicamos la Transformada de Fourier de forma que se aprecie la respuesta en frecuencia del filtro. Se pudo verificar que las frecuencias que dejan pasar y las que atenúan fueran las esperadas.

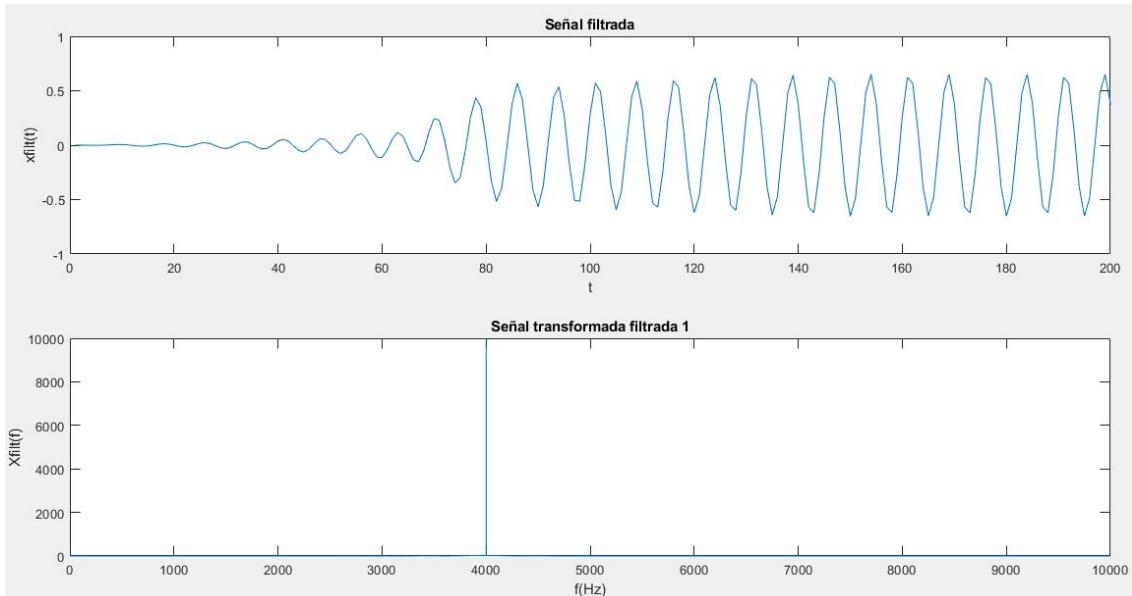
El script que se realizó se puede encontrar en el [Anexo E](#) para reproducir los resultados obtenidos. En el siguiente ejemplo tomamos dos filtros digitales, cada uno con un rango de frecuencias pasabanda que son de: 200 a 2000 Hz y de 2000 a 5000 Hz respectivamente. Es importante saber de antemano para que no haya inconvenientes utilizando Matlab, es que la fft (implementación de la Transformada Rápida de Fourier, “Fast Fourier Transform”) precisa que los valores de las frecuencias utilizadas en una función estén normalizadas. Esto es que las frecuencias a usar de ejemplo tienen que estar divididas por la frecuencia de muestreo utilizada. Si no utilizamos los valores de frecuencias normalizadas no se puede comprobar con la Transformada de Fourier si los filtros están funcionando adecuadamente ya que los resultados de la transformada no son adecuados. Se detallaran los resultados del script a continuación.

En la **Fig. 35** se muestra la señal original de entrada que está descrita por una suma de funciones senos, cada una con una frecuencia normalizada, que para este caso son de 1000 Hz y 4000 Hz respectivamente. Luego se puede ver su Transformada de Fourier mostrando la respuesta en frecuencia de  $x(t)$  en la que se ve claramente que las frecuencias que describen esta señal son justamente las ingresadas en la señal de entrada.

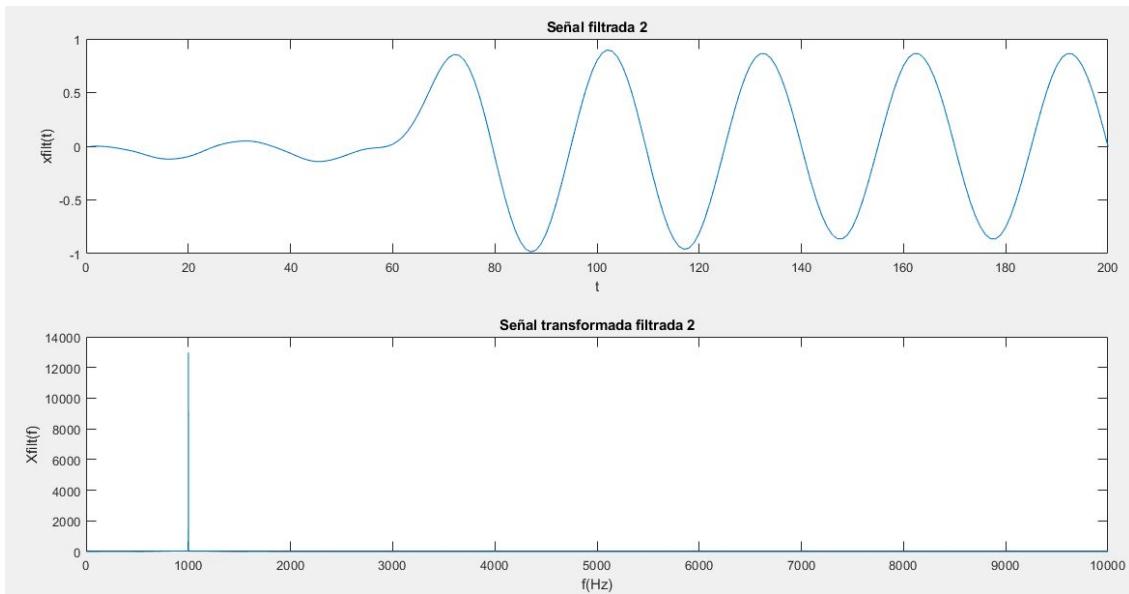


**Figura 35.** Señal original y su correspondiente respuesta en frecuencia.

A continuación se aplica a la señal original cada uno de los filtros pasa-banda mencionados obteniendo por separado las componentes que conformaban la señal original. En la **Fig. 36** se ve el resultado de aplicar el filtro de banda 2000 - 5000 hz, obteniendo la sinusoides de frecuencia 4000 Hz. Y en la **Fig. 37** se ve el resultado de aplicar el filtro de banda 200 - 2000 hz, con la sinusoides de frecuencia 1000 Hz.

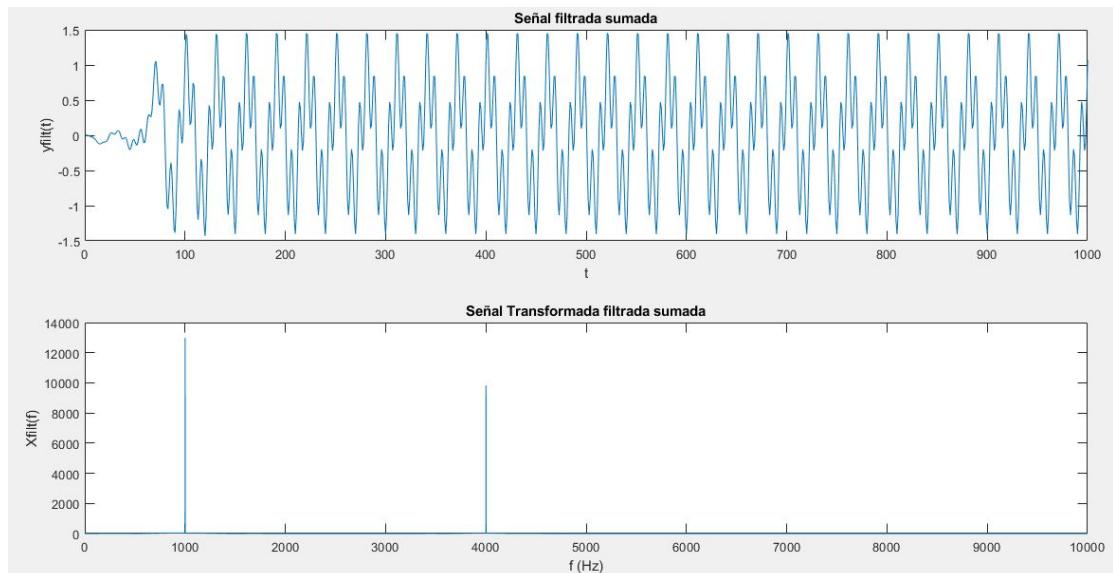


**Figura 36.** Señal filtrada por el primer filtro y la respuesta en frecuencia de la nueva señal.

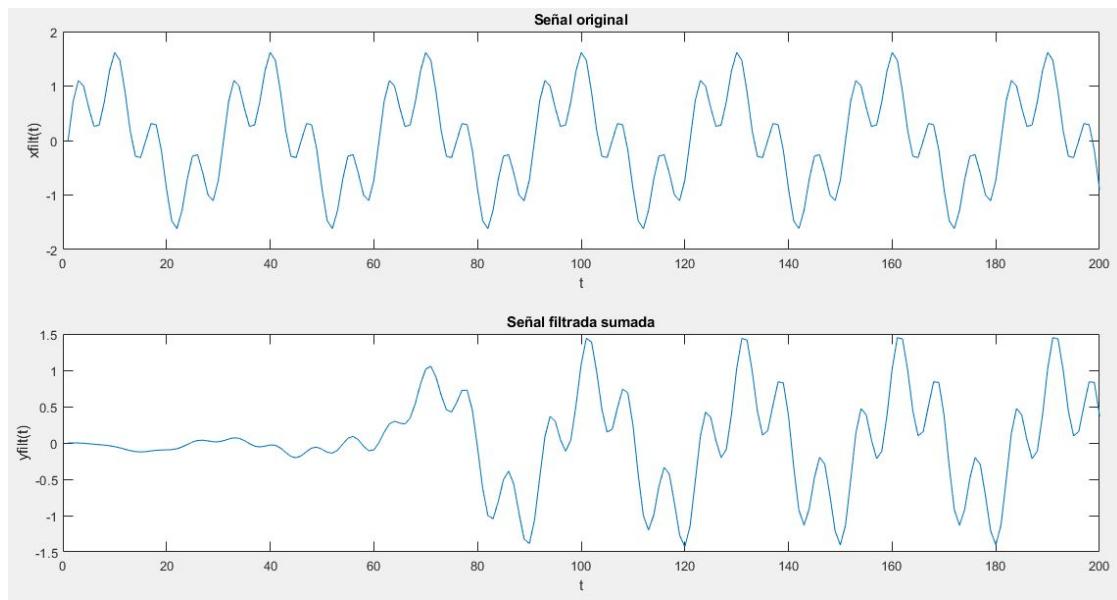


**Figura 37.** Señal filtrada por el segundo filtro y la respuesta en frecuencia de la nueva señal.

La **Fig. 38** muestra el resultado de sumar las señales filtradas por separado y su correspondiente respuesta en frecuencia. Y como se esperaba la respuesta en frecuencia es la misma que la de la señal de entrada original. Y en la **Fig. 39** se puede apreciar la comparación entre las dos.



**Figura 38.** Señal conformada por la suma de las dos señales filtradas de la original y su respuesta en frecuencia.



**Fig. 39.** Comparación entre la señal original de entrada y la señal filtrada de salida.

Como se puede ver, además del retardo de la señal filtrada sumada debido a la aplicación de los filtros digitales, la señal resultante es igual a la señal de entrada original.

En conclusión, este script simula los pasos que se realizan dentro modulo ecualizador ejecutándose dentro de la EDU-CIAA y los resultados ideales esperados.

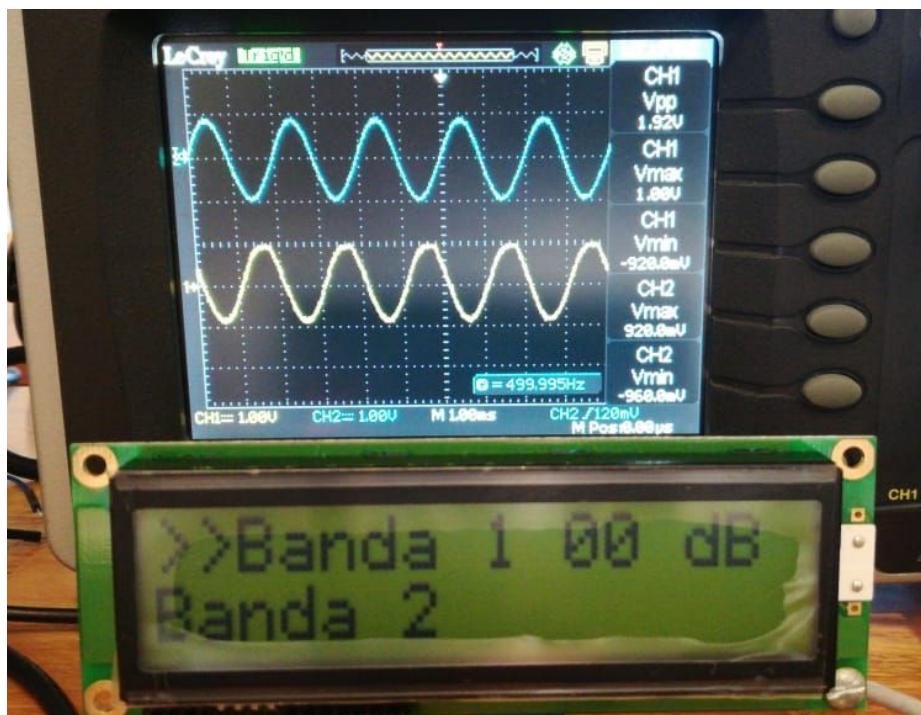
## 6.4 Ecualizador

Una vez que se probaron los circuitos de entrada y salida (y el funcionamiento, trivial llegado este punto, de los botones y pantalla LCD de interfaz con el usuario), se procedió a probar el funcionamiento del software y su capacidad para atenuar las diferentes bandas de frecuencias. A continuación se presentan la descripción e imágenes de dichos ensayos.

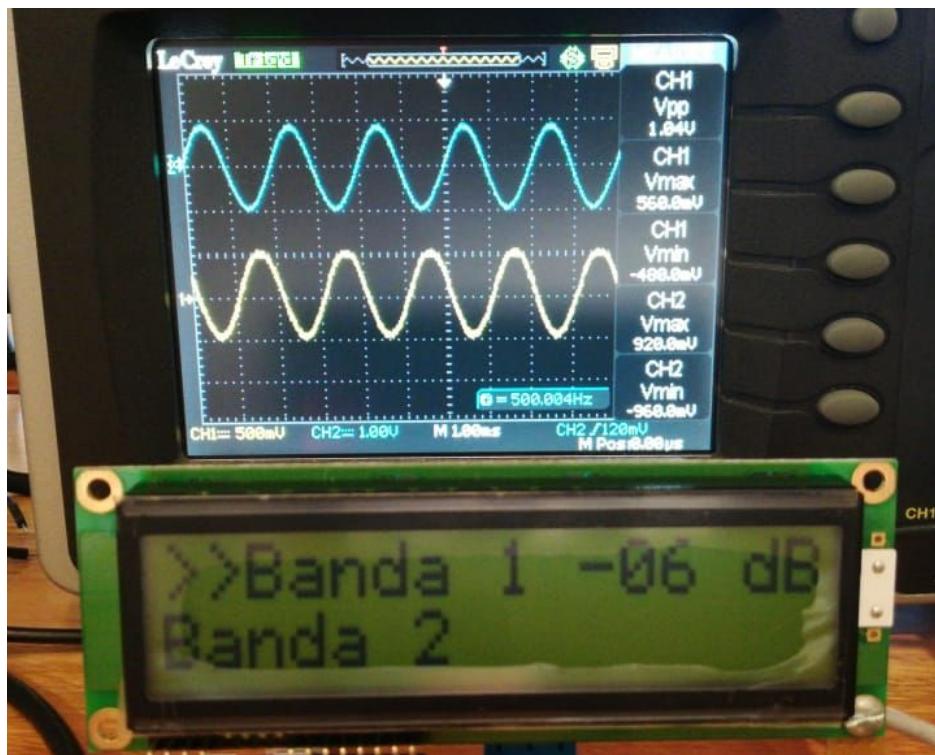
Como se ve en la **Fig. 40** se comenzó probando la entrada/salida del sistema completo poniendo una señal de 500Hz entre  $\pm 1V$  (azul, CH2) y viendo que la salida (amarilla, CH1) no tiene ningún cambio salvo por un retraso/desfase que puede ser provocado por un efecto conjunto del filtrado de la señal y su procesamiento por software.

En la **Fig 41.** se ve como al atenuar la banda 1 que contiene a 500 Hz por 6 dB, la señal de entrada se reduce a aproximadamente la mitad en su amplitud, resultado esperado según el estudio teórico previo.

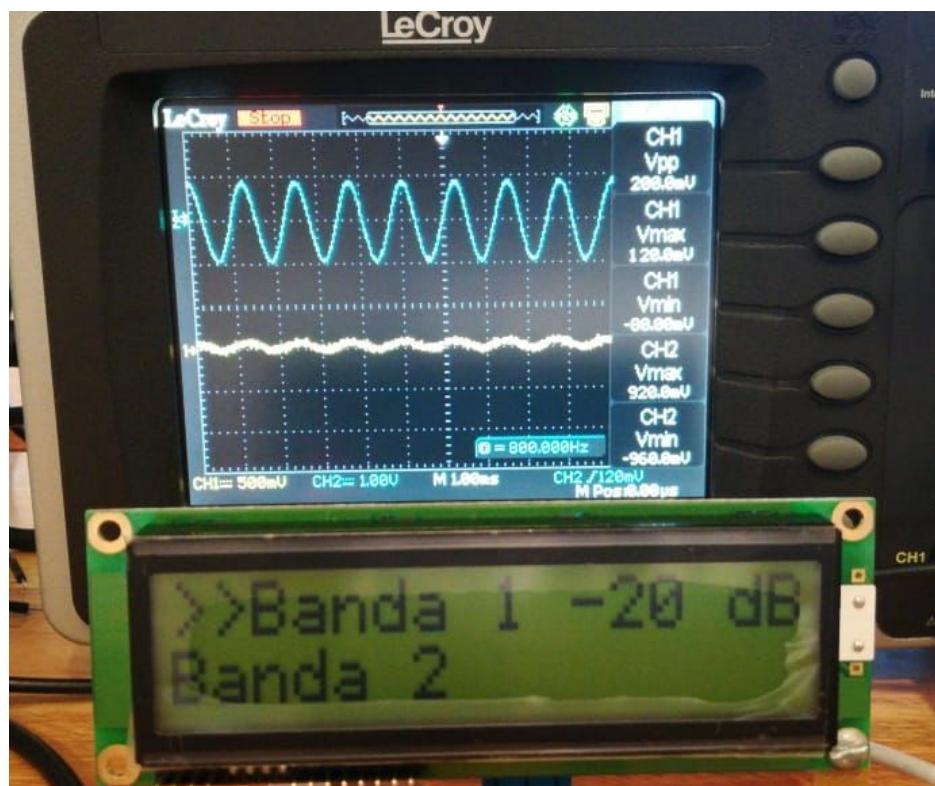
En la **Fig. 42** se puede ver una señal de 800Hz exitosamente atenuada por 20 dB y llegando a una amplitud de 200 mV lo que corresponde lo esperado, a aproximadamente un 10% de la amplitud de la señal de entrada.



**Figura 40.** Señal de 500 Hz sin atenuar.

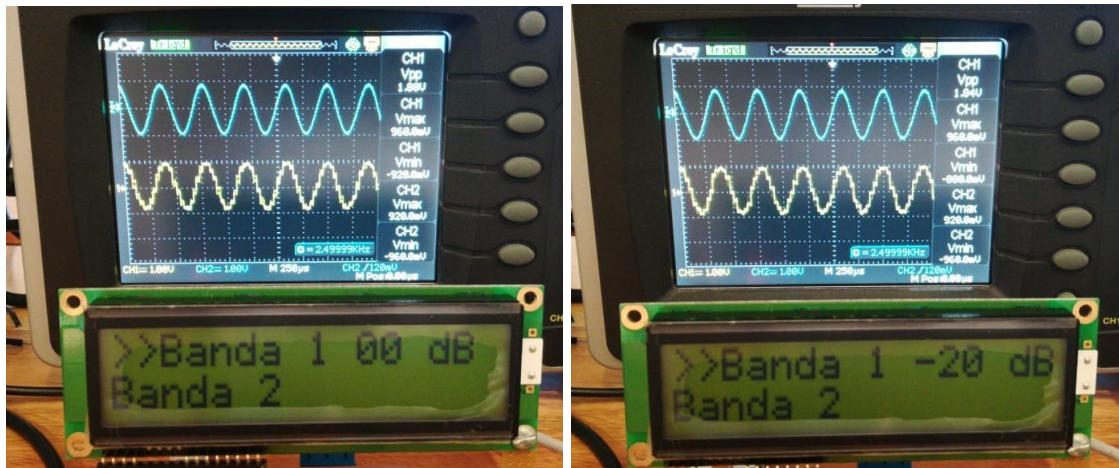


**Figura 41.** Señal de 500Hz atenuada por 6dB.

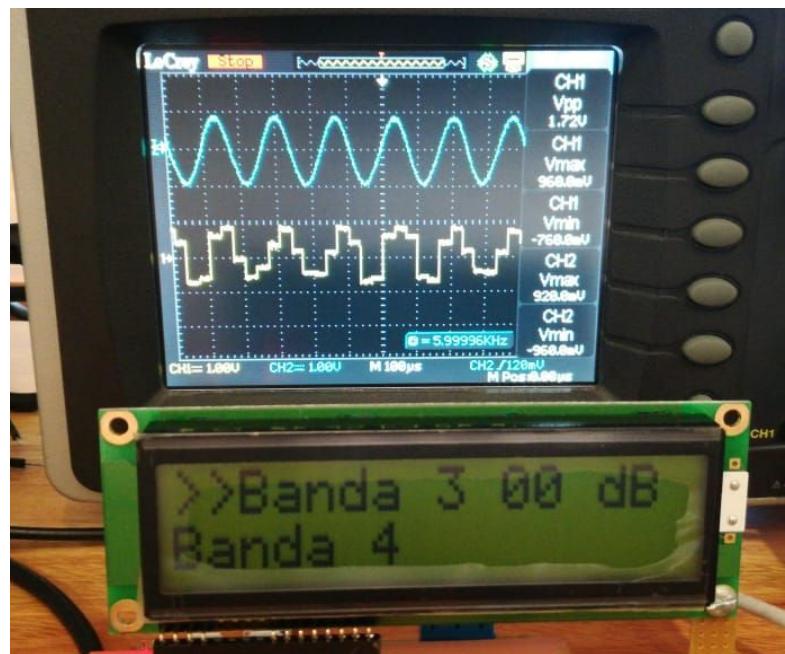


**Figura 42.** Señal de 800Hz atenuada por 20dB.

Luego se pasó a probar con una señal de 2500Hz. Se puede notar que la resolución de la señal de salida por el DAC se empieza a hacer visible. Esta señal recae en la segunda banda, por lo que primero se probó modificar los valores de atenuación de todas las bandas menos la propia para confirmar que no se generen cambios, como lo es esperado. Como se lo ve en la **Fig. 43** se probó con atenuación de 0 y 20 dB sobre la banda 1.



**Figura 43.** Señal de 2500Hz, atenuando todas las bandas menos las propias.

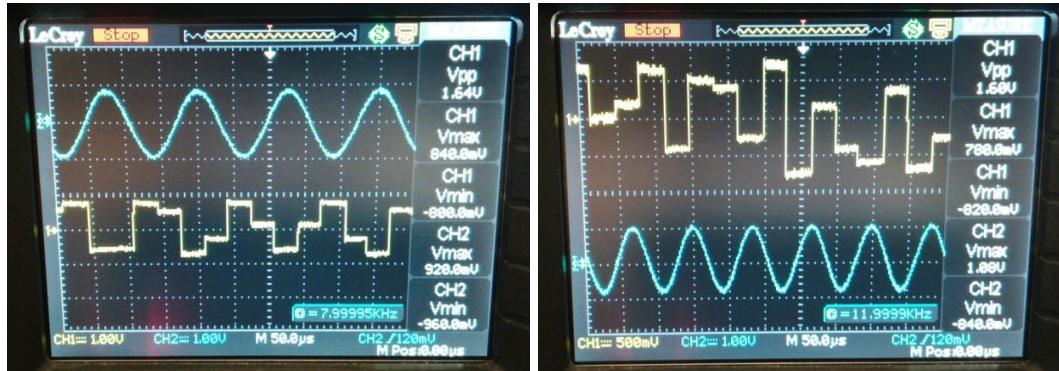


**Figura 44.** Entrada/Salida de señal de 6kHz .

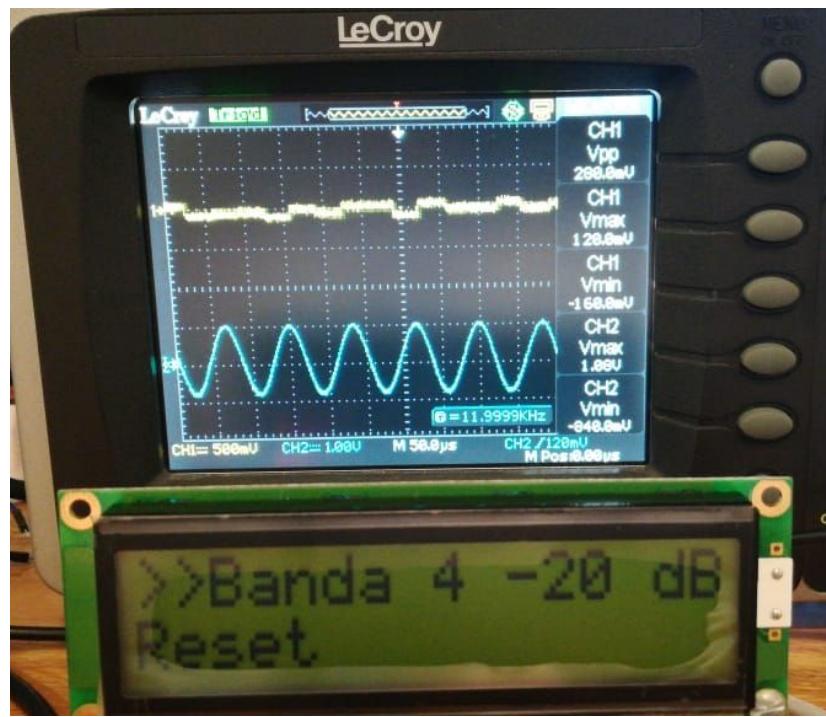
Se siguió probando con señales de frecuencias más altas. Ya a los 6kHz es bastante baja la resolución lograda, como se ve en la **Fig. 44**. Habiendo probado con menos cantidad de bandas de ecualización y logrado una mejor respuesta, concluimos que el problema está en que el procesamiento de la señal no es lo suficientemente rápido

para lograr una buena resolución mientras se manejan cuatro filtros FIR con alrededor de 100 constantes cada uno.

Se siguió aumentando la frecuencia de la señal de entrada para poder ver mejor este efecto, ver **Fig. 45**. Ya a los 12 kHz la señal de salida parece ser sólo ruido. Se hicieron pruebas con música de un celular y a oído pareciera que la manipulación de bandas de frecuencia por encima de los 10 kHz no afectan mucho a lo perceptible con parlantes comunes, pero no es algo adecuado para un procesamiento más refinado de audio.



**Figura 45.** Entrada/Salida de señales de 8kHz y 12kHz.



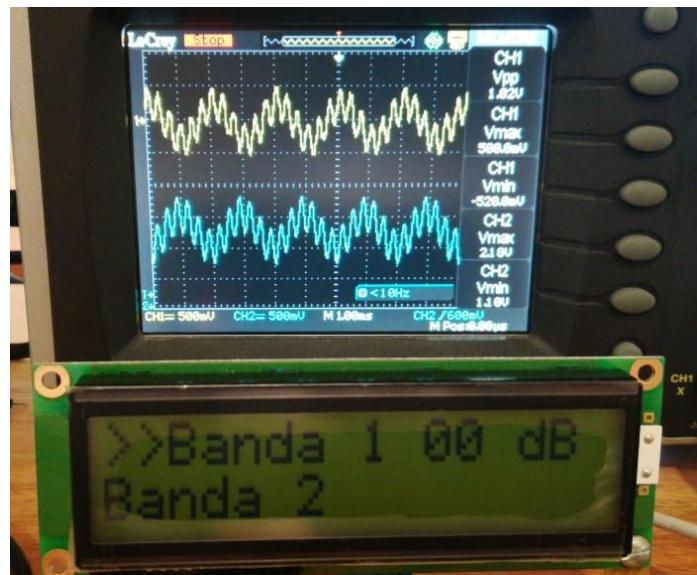
**Figura 46.** Señal de 12kHz atenuada 20 dB.

Con la **Fig. 46**, concluimos que la atenuación de cada banda funciona de manera correcta.

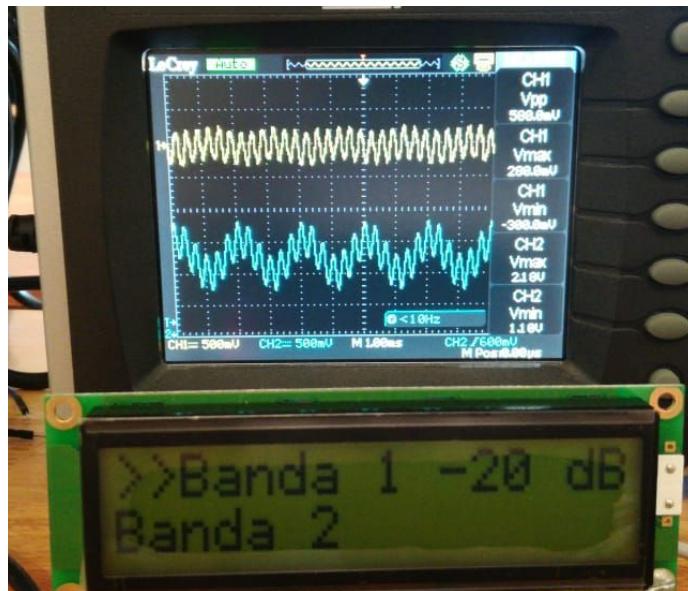
En un ensayo sin resultados gráficos se probó con una entrada de audio reproduciendo música directamente desde el celular. Anecdóticamente se percibió que la mayoría del sonido se producía en las primeras tres bandas de frecuencias definidas, ya que de todos modos sería muy difícil de percibir sonidos de 15kHz a 20kHz en medio de los otros más habituales en la música contemporánea.

#### 6.4.1 Procesando señales compuestas

De la misma forma que se hizo en el ensayo teórico sobre MatLab, se procedió a probar con una señal formada por la suma de dos sinusoides, de 500Hz y 3kHz, generadas como sonido directamente desde el celular con una aplicación. Sin ningún tipo de atenuación, se puede ver perfectamente en la entrada (azul) y en la salida (amarillo) la señal de 3kHz “montada” sobre la de 500Hz.

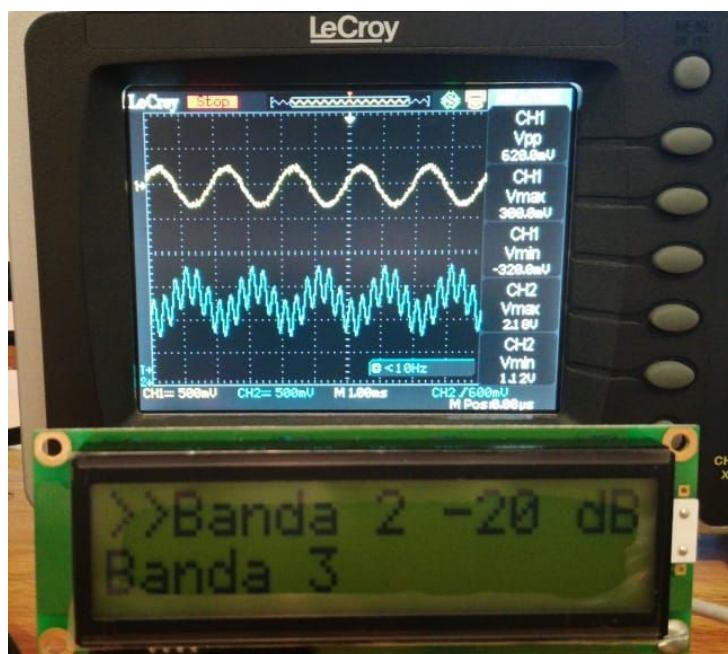


**Figura 47.** Entrada/Salida de señal formada por sinusoides de 500 Hz y 3 kHz.



**Figura 48.** Entrada/Salida de señal 500 Hz + 3 kHz atenuando solo banda 1.

Como se ve en la **Fig. 48**, al atenuar la primera banda de frecuencias, la componente de 500 Hz pierde amplitud y la salida se asemeja a la componente de 3 kHz aislada. Mientras que al atenuar solo la segunda banda, la componente de 3 kHz es la que pierde amplitud, y la señal de salida se asemeja a la componente de 500Hz. Esto se puede apreciar en la **Fig 49**. Concluimos que los diferentes filtros funcionan como lo esperado en tandem.



**Figura 49.** Entrada/Salida de señal 500 Hz + 3 kHz atenuando solo banda 2.

## 6.5 Problemas con el procesamiento

A pesar de internamente estar intentando escribir al DAC con una frecuencia de 44kHz, se hizo evidente que el filtrado completo no llegaba a realizarse suficientemente rápido como para lograr una resolución razonable a frecuencias altas. El procesamiento de la señal de audio, en particular los cálculos de los valores de salida de los filtros, generan un retardo tal que la frecuencia de escritura al DAC se disminuye de 44 kHz a ~30 kHz. Lo que genera que para frecuencias mayores a 10 kHz se produzca prácticamente ruido en la salida.

Se consideró que se podría mejorar esta respuesta del sistema si los filtros fueran de menor orden para acortar el tiempo de procesado, pero luego de probar con bandas más permisivas o mayores valores de ripple no se consiguió obtener un menor orden sin comprometer demasiado la calidad de los filtros. Sí se pudo notar un cambio importante en la reducción de cantidad de bandas, pero esto afecta al objetivo de tener múltiples bandas para atenuar, para cumplir las características de ecualizador.

## 7. Conclusiones

### 7.1 Cumplimiento de objetivos planteados

Si bien se logró un prototipo final que cumple con los objetivos básicos establecidos al principio del desarrollo del proyecto de poder filtrar y atenuar individualmente 4 bandas, encontramos limitaciones en la placa de desarrollo utilizada.

La EDU-CIAA resultó tener una capacidad limitada para el procesamiento de señales en tiempo real. Si bien es capaz de aplicar uno o dos filtros produciendo una buena respuesta en el rango de frecuencias que percibe el oído humano, ya con cuatro filtros simultáneos de orden rondando a 100 coeficientes por filtro no se alcanza a cubrir el rango entero sin presentar demasiados retardos y distorsiones en las frecuencias más altas.

Probar con audio real permite percibir las diferentes atenuaciones, pero los resultados obtenidos parecen indicar que una ecualización de audio útil o de aplicación práctica no es algo alcanzable para la EDU-CIAA, al menos con el tiempo de investigación y desarrollo que se le asignó al proyecto durante la materia.

No se descarta que con optimizaciones a nivel de implementación de los filtros, mejor uso del hardware de la CIAA (ya que por ejemplo se posee un core secundario), o haciendo un filtrado más complejo a nivel electrónico que tome algunas de las funciones que nosotros implementamos a nivel de software, se puedan lograr muchos mejores resultados.

El proyecto en sí dio lugar a la primer gran aplicación práctica conjunta de los conocimientos adquiridos en las materias de la carrera dirigidas a la electrónica y electrotecnia. Como consecuencia nos llevamos un mejor entendimiento de cuestiones como las implicaciones de usar amplificadores operacionales no ideales, el trabajar con un procesador sin saber si es suficientemente potente para la aplicación requerida, y el acondicionamiento de señales analógicas.

### 7.2 Cumplimiento de requerimientos

Los requerimientos establecidos previamente al comienzo de la materia están casi en su totalidad cumplidos. Con lo que no se pudo cumplir es con una implementación adecuada de los 4 filtros pasa-banda, debido a las capacidades de la EDU-CIAA, como se comentó anteriormente. Como el alcance del proyecto era probar los alcances de la EDU-CIAA para procesamiento en tiempo real y filtrado digital de varias bandas, los resultados obtenidos no son tan desalentadores, ya que la EDU-CIAA bien podría implementar mejores filtros digitales con la utilización de menos bandas.

En cuanto a los circuitos de acondicionamiento de las señales, tanto el entrada como el de salida, logramos implementarlos de forma adecuada y satisfactoria para los parámetros necesarios a este proyecto y para la placa utilizada. De todas formas se encuentra abierto a mejoras, en particular en lo relacionado a que amplificador operacional utilizar para incrementar rendimiento y calidad del procesamiento de la señal.

Se utilizó de forma satisfactoria el Firmware v2 para el desarrollo del sistema informático, junto a las librerías sAPI. Si bien en ciertos aspectos esta última aceleró el proceso de desarrollo del software, en algunos aspectos sentimos que no permite la configuración flexible y completa del micro, por lo que algunas funcionalidades que provee no las utilizamos como tal, sino que realizamos implementaciones propias.

## 7.3 Actividad individual

Las actividades que fueron realizadas para realizar el proyecto fueron divididas en tres partes para cada integrante del grupo. Las tareas llevadas a cabo se dividieron en: software para la EDU-CIAA, cálculo de los límites eléctricos adecuado, acondicionamiento y realización de circuitos y el desarrollo y prueba de los filtros digitales.

Además de las tareas realizadas individualmente por cada integrante en paralelo, también se trató de trabajar en el proyecto en conjunto el mayor tiempo posible para asegurar el buen funcionamiento de las partes en conjunto. Luego en el armado de la PCB, mediciones y testeos se trabajó en grupo en todo momento.

- **Stranieri, Jorge:** Investigación sobre filtros digitales FIR, su desarrollo e implementación en MATLAB. Testeo de los filtros digitales en MATLAB y aplicación.
- **Sanchez, Agustín:** Diseño del esquemático y PCB sobre el entorno de desarrollo KiCad. Programación del sistema informático del proyecto sobre la EDU-CIAA.cad. Desarrollo de software en firmware v2 para el procesamiento de la señal de entrada con la EDU-CIAA e implementación de la interfaz de usuario..
- **Marzano, Nicolás:** cálculo de los parámetros electrónicos adecuados para la implementación del proyecto. Estudio de diferentes amplificadores operacionales posibles y circuitos de acondicionamiento de señal.

Las horas invertidas en el proyecto se estimaron a partir de las horas de cursada de la materia, las horas extracurriculares empleadas en la investigación, y además las horas empleadas en el laboratorio. Por alumno se invirtieron: 64 horas de cursada, 32 horas de investigación y desarrollo, y aproximadamente 60 horas en el mes de febrero para la implementación final y armado del prototipo, para un total de 156 horas por persona. Por lo tanto se invirtieron en el proyecto aproximadamente 468 horas, y la

cantidad de horas invertidas con respecto a la ingeniería del proyecto equivaldrían a 276 horas en conjunto.

## 7.4 Presupuesto

Estimando un costo de 350 pesos argentinos por hora hombre, a 276 horas de desarrollo, ingeniería e implementación, se concluye en un presupuesto de \$96.600 .

Contando las horas de capacitación y desarrollo realizado durante las horas de cursada, se suman otros \$67.200 para un total de \$163.800 .

El costo total de los instrumentos comprados específicamente para el prototipado y testeo del proyecto, sumado a todos los componentes necesarios para su implementación, es un número que no se consideró en su momento y no se posee en exactitud, pero que se acotó en \$600~\$800.

En total, esto da un presupuesto final de \$164.600 .

## 8. Bibliografía

- Proyecto CIAA: <http://www.proyecto-ciaa.com.ar/>
- Repositorio CIAA en github: <https://github.com/ciaa>
- Teorema de Nyquist:  
[https://es.wikipedia.org/wiki/Teorema\\_de\\_muestreo\\_de\\_Nyquist-Shannon](https://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon)
- Diseño y análisis de filtros en procesamiento de audio:  
[https://www.exabyteinformatica.com/uoc/Audio/Procesamiento\\_de\\_audio/Procesamiento\\_de\\_audio\\_\(Modulo\\_2\).pdf](https://www.exabyteinformatica.com/uoc/Audio/Procesamiento_de_audio/Procesamiento_de_audio_(Modulo_2).pdf)
- Alan V. Oppenheim, Alan S. Willsky (Segunda Edición) Señales y sistemas.
- Manual de KiCad: [http://kicad-pcb.org/help/documentation/#\\_getting\\_started](http://kicad-pcb.org/help/documentation/#_getting_started)
- Ponchos CIAA:  
<http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:ciaa:ponchos>
- Firmware v2 CIAA: [https://github.com/ciaa/firmware\\_v2](https://github.com/ciaa/firmware_v2)

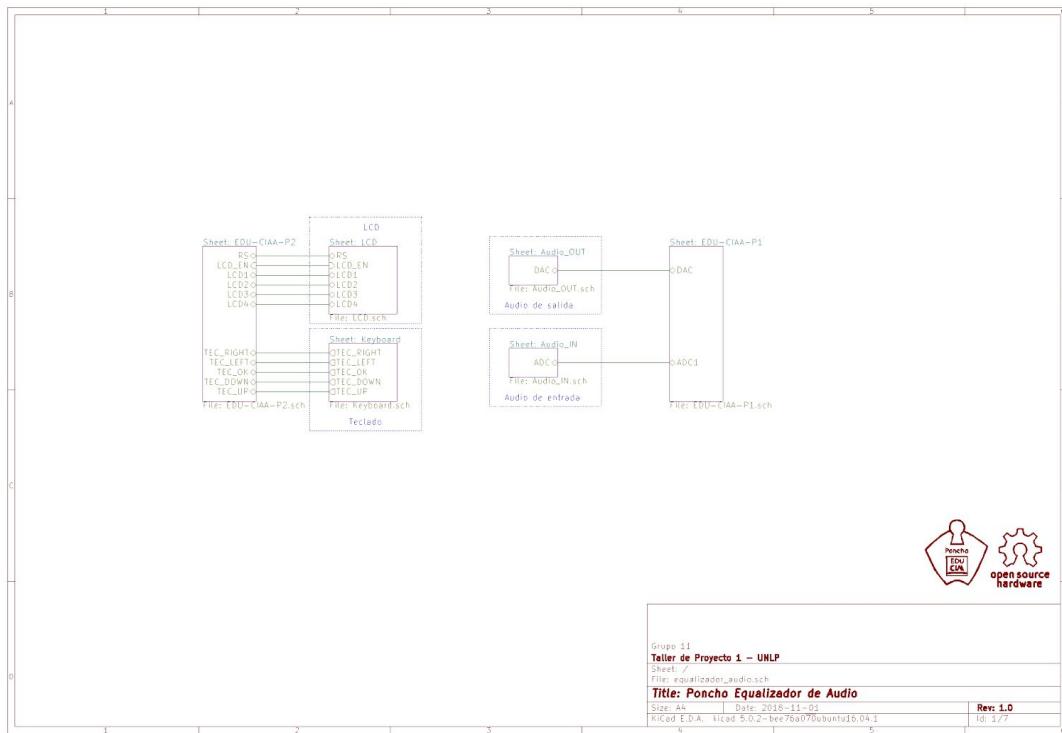
## 9. Anexos

### 9.1 Anexo A - Hojas de datos:

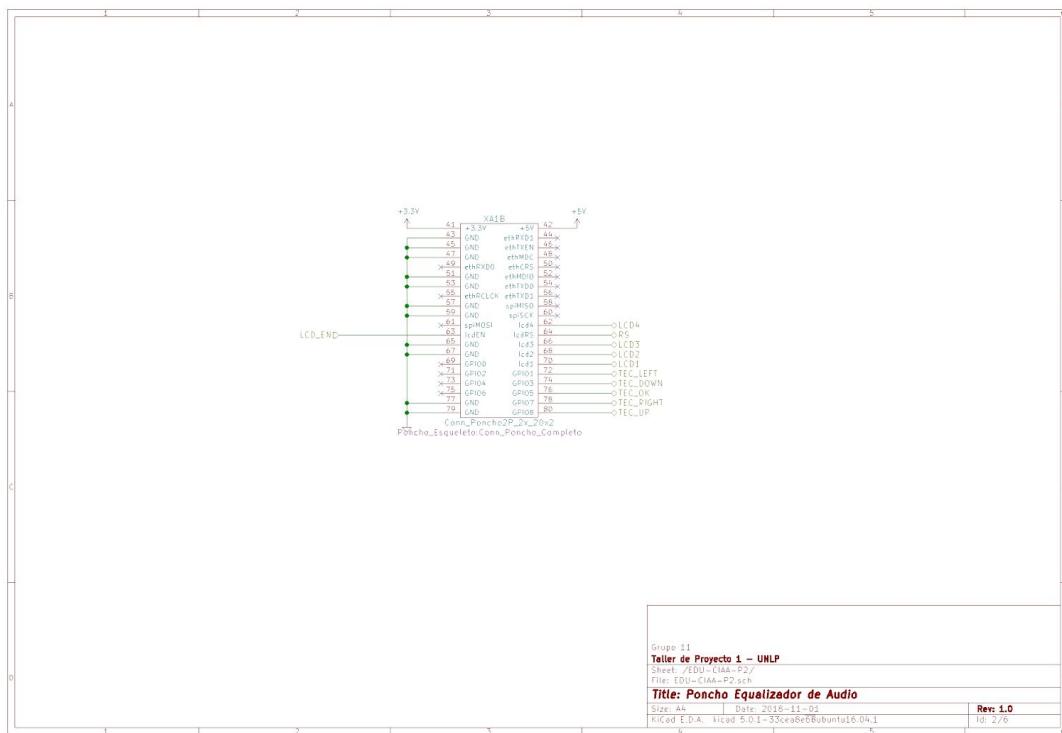
- Datasheet LM386: <http://www.ti.com/lit/ds/symlink/lm386.pdf>
- Datasheet LPC435X\_3X\_2X\_1X:  
[https://www.nxp.com/docs/en/data-sheet/LPC435X\\_3X\\_2X\\_1X.pdf](https://www.nxp.com/docs/en/data-sheet/LPC435X_3X_2X_1X.pdf)
- Winstar Display LCD:  
<https://www.compel.ru/item-pdf/d5b376fad4c6ffff45f2d4de8b58ba3/pn/winstar~wh1602a-ygh-ctk.pdf>
- Datasheet LM358:  
<http://pdf1.alldatasheet.com/datasheet-pdf/view/3067/MOTOROLA/LM358.html>

## 9.2 Anexo B - Esquemáticos

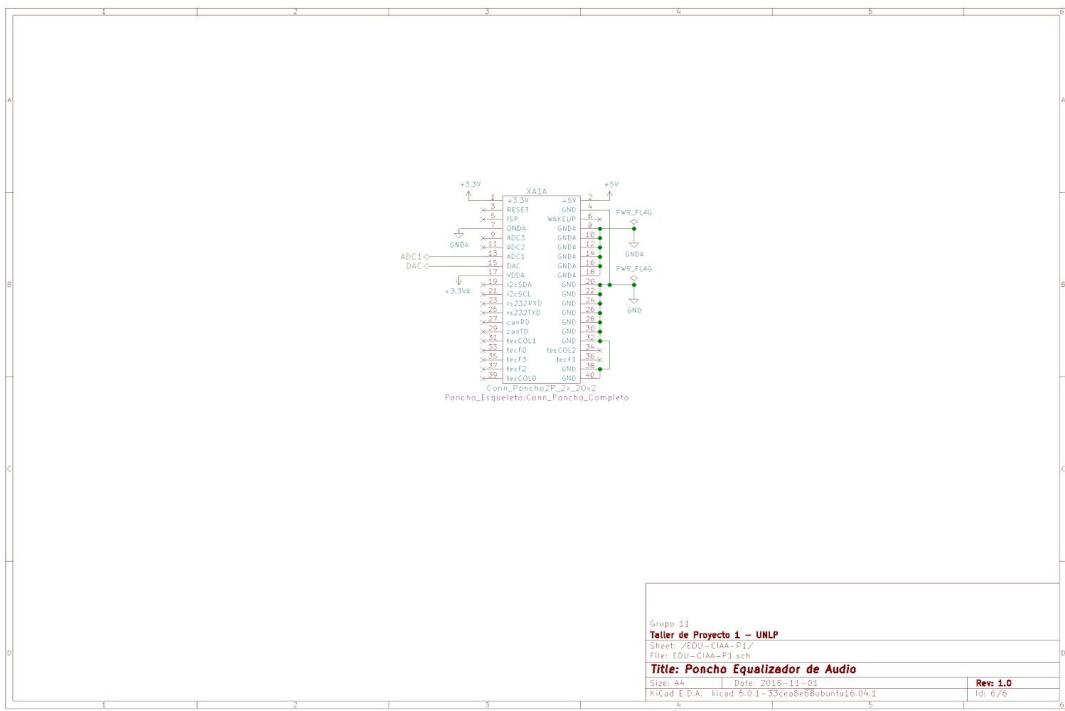
### Diagrama del sistema general



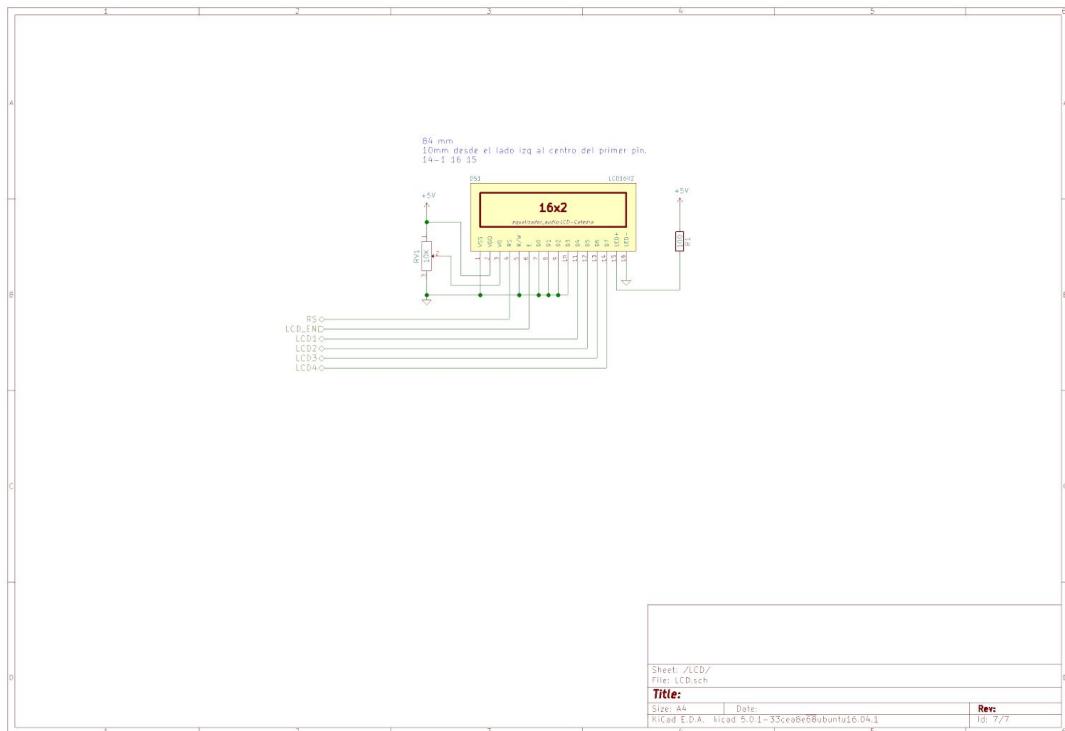
### Puerto 2 de la EDU-CIAA



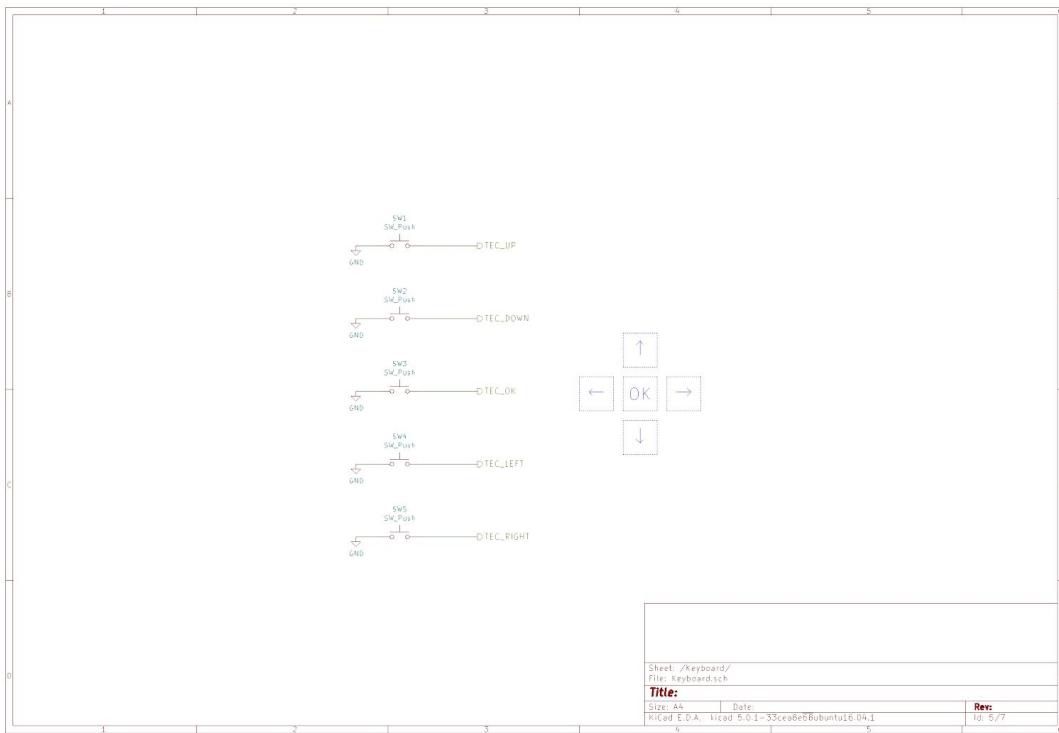
## Puerto 1 de la EDU-CIAA



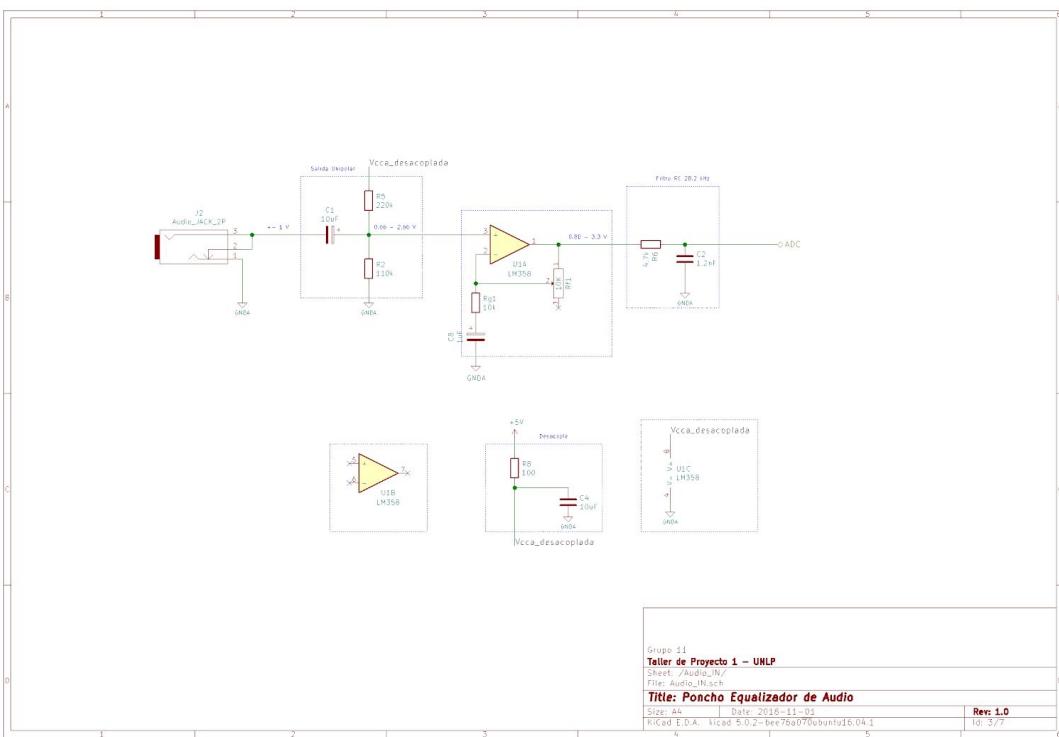
## Display LCD



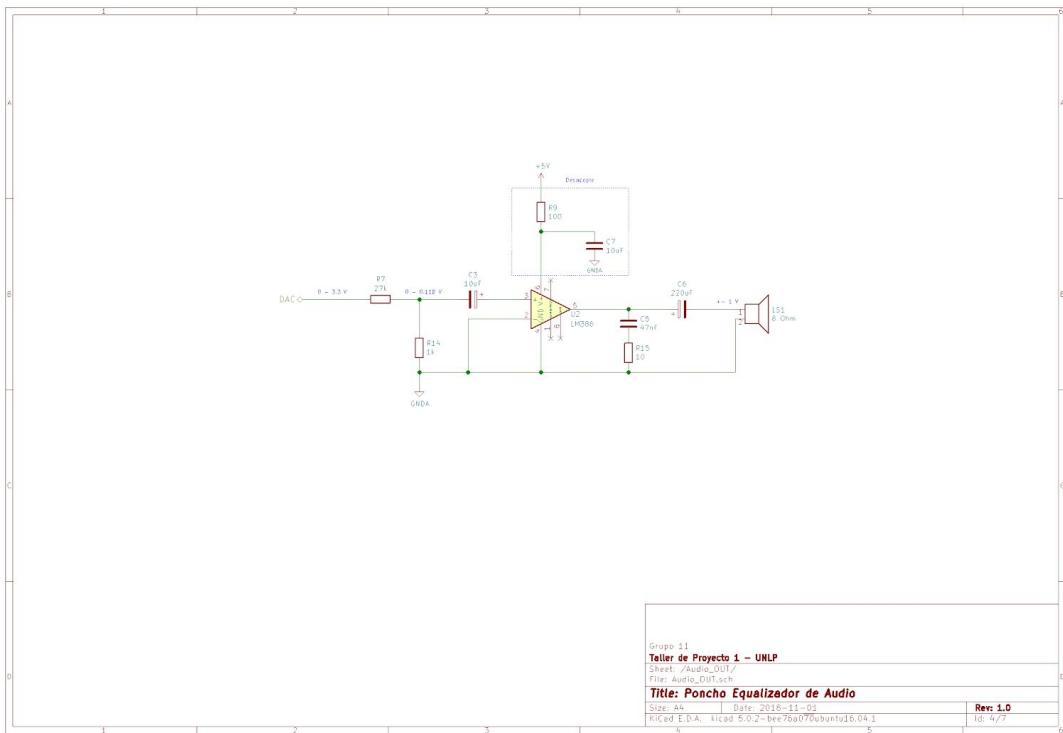
## Teclado



## Audio de entrada

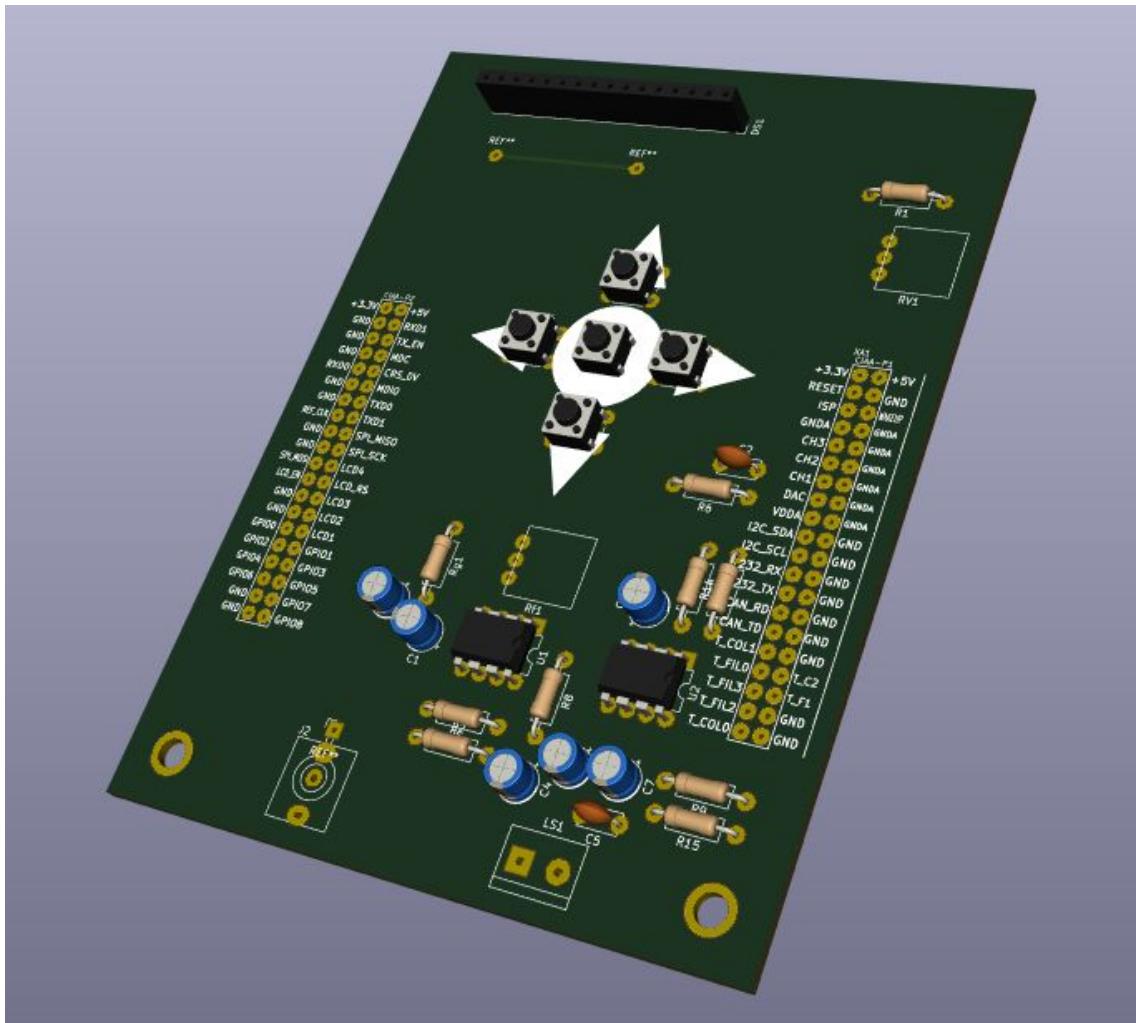


## Audio de salida



### 9.3 Anexo C - Render 3D

#### Render 3D



## 9.4 Anexo D - Listado BOM

**Tabla D1.** Listado de componentes BOM.

Id	Designator	Package	Quantity	Designation
1	XA1	poncho_recortado	1	Conn_Poncho2P _2x_20x2
2	SW1,SW2,S W3,SW4,S W5	SW_PUSH_6mm_H5mm	5	SW_Push
3	U1	DIP-8_W7.62mm	1	LM358
4	U2	DIP-8_W7.62mm	1	LM386
5	J2	Jack_3.5mm_QingPu_WQP-PJ398SM_Vertical_CircularHoles	1	Audio_JACK_2P
6	Rf1,RV1	potenciometro_acostado	2	10K
7	C7,C1,C3,C 4	CP_Radial_D5.0mm_P2.50mm	4	10uF
8	DS1	LCD-Catedra	1	LCD16X2
9	C2	C_Disc_D5.0mm_W2.5mm_P5.00mm	1	0.01uF
10	C5	C_Disc_D5.0mm_W2.5mm_P5.00mm	1	47nF
11	LS1	TerminalBlock_bornier-2_P5.08mm	1	8 Ohm
12	R1,R8,R9	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	3	100
13	R2	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	100k
14	R5	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	220k
15	R6	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	4.7k
16	R7	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	27k
17	R14	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	1k
18	R15	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	10
19	Rg1	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	1	10k
20	C6	CP_Radial_D5.0mm_P2.50mm	1	250Uf
21	C8	CP_Radial_D5.0mm_P2.50mm	1	1F

## 9.5 Anexo E - Código MatLab

```
1. Fs = 30000; % Frecuencia de Muestreo.
2. T = 1/Fs; % Período de muestreo.
3. L = round(Fs); % Ancho de la señal.
4. t = (0:L-1); % Vector de tiempo.
5.
6. fn1=4000/Fs; %Frecuencia normalizada 1.
7. fn2=1000/Fs; %Frecuencia normalizada 2.
8.
9. %Señal original
10. x = 0.7*sin(2*pi*fn1*t) + sin(2*pi*fn2*t);
11. subplot(3,3,1); %Divide en secciones para graficar en una sola figura.
12. plot(x);
13. title('Señal original');
14. xlabel('t');
15. ylabel('x(t)');
16. xlim([0 200]);
17.
18. %Transformada de Fourier de x
19. X = abs(fft(x));
20. subplot(3,3,2);
21. plot(X);
22. title('Transformada de x(t)');
23. xlabel('f (Hz)');
24. ylabel('X(f)');
25. xlim([0 10000]);
26.
27. % Señal filtrada 1
28. xfilt=filter(FiltroEjemplo2,x);
29. subplot(3,3,3);
30. plot(xfilt);
31. title('Señal filtrada');
32. xlabel('t');
33. ylabel('xfilt(t)');
34. xlim([0 200]);
35.
36. %Señal filtrada 2
37. x2filt=filter(pasabanda200,x);
38. subplot(3,3,4);
39. plot(x2filt);
40. title('Señal filtrada 2');
41. xlabel('t');
42. ylabel('x2filt(t)');
43. xlim([0 200]);
44.
45. % %Transformada filtrada 1
46. Xfilt = abs(fft(xfilt));
47. subplot(3,3,5);
48. plot(Xfilt);
49. title('Señal transformada filtrada 1');
50. xlabel('f (Hz)');
51. ylabel('Xfilt(f)');
52. xlim([0 10000]);
```

```

53.
54. %Transformada filtrada 2
55. X2filt =abs(fft(x2filt));
56. subplot(3,3,6);
57. plot(X2filt);
58. title('Señal transformada filtrada 2');
59. xlabel('f(Hz)');
60. ylabel('Xfilt(f)');
61. xlim([0 10000]);
62.
63. %Senial sumada
64. subplot(3,3,7);
65. yfilt = xfilt + x2filt;
66. plot(yfilt);
67. title('Señal filtrada sumada');
68. xlabel('t');
69. ylabel('yfilt(t)');
70. xlim([0 200]);
71.
72. %Transformada senial sumada
73. subplot(3,3,8);
74. Yfilt =abs(fft(yfilt));
75. plot(Yfilt);
76. title('Señal Transformada filtrada sumada');
77. xlabel('f (Hz)');
78. ylabel('Xfilt(f)');
79. xlim([0 10000]);
80.
81. %Filtros
82. function Hd = FiltroEjemplo2
83. %FILTROEJEMPLO2 Returns a discrete-time filter object.
84.
85. % MATLAB Code
86. % Generated by MATLAB(R) 9.3 and DSP System Toolbox 9.5.
87. % Generated on: 25-Feb-2019 00:52:22
88.
89. % Equiripple Bandpass filter designed using the FIRPM function.
90.
91. % All frequency values are in Hz.
92. Fs = 30000; % Sampling Frequency
93.
94. Fstop1 = 1500; % First Stopband Frequency
95. Fpass1 = 2000; % First Passband Frequency
96. Fpass2 = 4000; % Second Passband Frequency
97. Fstop2 = 4500; % Second Stopband Frequency
98. Dstop1 = 0.001; % First Stopband Attenuation
99. Dpass = 0.057501127785; % Passband Ripple
100. Dstop2 = 0.0001; % Second Stopband Attenuation
101. dens = 20; % Density Factor
102.
103. % Calculate the order from the parameters using FIRPMORD.
104. [N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1 ...
105. 0], [Dstop1 Dpass Dstop2]);
106.

```

```

107. % Calculate the coefficients using the FIRPM function.
108. b = firpm(N, Fo, Ao, W, {dens});
109. Hd = dfilt.dffir(b);
110. end
111. function Hd2 = pasabanda200
112. %PASABANDA200 Returns a discrete-time filter object.
113.
114. % MATLAB Code
115. % Generated by MATLAB(R) 9.3 and Signal Processing Toolbox 7.5.
116. % Generated on: 30-Oct-2018 17:15:35
117.
118. % Equiripple Bandpass filter designed using the FIRPM function.
119.
120. % All frequency values are in Hz.
121. Fs = 30000; % Sampling Frequency
122.
123. N = 128; % Order
124. Fstop1 = 0; % First Stopband Frequency
125. Fpass1 = 200; % First Passband Frequency
126. Fpass2 = 2000; % Second Passband Frequency
127. Fstop2 = 2300; % Second Stopband Frequency
128. Wstop1 = 40; % First Stopband Weight
129. Wpass = 1; % Passband Weight
130. Wstop2 = 40; % Second Stopband Weight
131. dens = 20; % Density Factor
132.
133. % Calculate the coefficients using the FIRPM function.
134. b = firpm(N, [0 Fstop1 Fpass1 Fpass2 Fstop2 Fs/2]/(Fs/2), [0 0 1 1 0 ...
135.           0], [Wstop1 Wpass Wstop2], {dens});
136. Hd2 = dfilt.dffir(b);
137.
138. %[Y,Fs] = audioread('C:\Program
Files\MATLAB\R2017b\toolbox\matlab\audiovideo\bigbang.wav');
139. %X = filter(Hd,Y);
140. %sound(X,Fs)
141.
142. % [EOF]
143. end

```

## 9.6 Anexo F - Circuito impreso final

