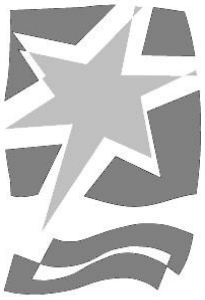
UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sonido e Imagen)

UNIVERSIDAD
POLITECNICA
DE VALENCIA



ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Implementación de Vúmetro Óptico, mediante dispositivo DSP SHARC de Analog Devices."

TRABAJO FINAL DE CARRERA

Autor/es:
Jorge Castillo Santos

Director/es:
D.José Marin-Roig Ramón

GANDIA, 2014

1. INTRODUCCIÓN Y OBJETIVOS.

1.1. OBJETIVOS

El objetivo principal de este Trabajo Final de Carrera (TFC) es diseñar un vúmetro óptico de leds construido mediante el circuito integrado LM3916 y el Procesador Digital de Señales (DSP), ADSP-21489 de Analog Devices.

1.2. INTRODUCCION.

¿QUÉ ES UN VÚMETRO?

Un vúmetro en ingles SVI (Stándar Volumen Indicator) es un dispositivo gracias al cual podemos obtener información de forma visual sobre el volumen de la señal que llega a nuestros oídos, hoy en día podemos encontrar este dispositivo en múltiples equipos de audio e incluso video.

Vúmetro analogico

Para entender el funcionamiento de este dispositivo tenemos que remontamos a su historia. La idea para desarrollar el vúmetro surgió en la convención *"Unknown Soldier on Armistice Day"(*celebrada en, New York en el año 1921) con el fin de proporcionar el nivel de audio suficiente sin llegar a percibir una distorsión en la señal ofrecida.

Durante las pruebas de sonido se dieron cuenta de que el mismo volumen de salida provocaba una distorsión a través de un receptor telefónico pero no en los altavoces. Por ello, se tuvo la idea de crear un dispositivo que ofreciera una información visual sobre los niveles de audio que podían entregarse sin llegar a ciertos límites los cuales provocasen esta distorsión, de esta forma se podría controlar dicho nivel a través de potenciómetros o faders.

Es por esto que hoy en día, a parte de ofrecernos una información visual del volumen, tenga su principal aplicación profesional en la óptima captación y emisión de señales de audio.

Su medición es lenta a propósito para reflejar de una manera clara al usuario los máximos y mínimos en la señal entregada.

Como norma general la señal no debe superar la franja marcada con 0VU, ya que esto podía ocasionar pérdidas en la calidad de sonido ocasionando problemas durante el procesado de la señal en equipos de audio digital, por otro lado, si la señal es demasiado baja la relación entre el ruido y la señal es más notable en la grabación, lo cual también dificulta su posterior procesado si lo hubiese.

1.3. DESCRIPCIÓN TECNICA.

El vúmetro está constituido por 20 diodos leds los cuales marcan el nivel desde los -20dB hasta los 3dB. Para la construcción del vúmetro led, ha sido necesaria la utilización de las herramientas Orcad Pspice y Orcad Layout con el fin de realizar una placa PCB (printed circuit board) en la cual integrar todos los elementos electrónicos necesarios para obtener un correcto funcionamiento.

La intención de este TFC es visualizar el nivel de salida de distintas señales que serán procesadas con el DSP y posteriormente enviadas a dicho vúmetro. Existen dos formas de visualizar el nivel de salida, el modo barra (todos los leds se encienden) y el modo punto (solo se enciende un led)

El procesado de las señales, consta de varios filtros digitales no recursivos o filtros FIR, que serán implementados en nuestro DSP mediante el entrono grafico Visual DSP++5.0.El diseño de los filtros, (el tipo de filtro, y el número de coeficientes) se realizará con la herramienta Filter Design & Analysis Tool, (FDA tool) incluida en Matlab.

Dado que a la hora de procesar señales de audio digitales, es necesario que se realicen de forma rápida y repetida distintas operaciones matemáticas que, normalmente están basadas en sumas y multiplicaciones (ej. Filtros digitales, FFTs, etc.), se ha decidido utilizar un DSP, ya que la arquitectura de este, está optimizada para la realización de dichas tareas de forma casi instantánea.

Para conseguir el objetivo principal será necesario:

- Comprender el funcionamiento de nuestro DSP para la posterior implementación de los filtros.
- Seleccionar los distintos tipos de filtros FIR que vamos a utilizar.
- El análisis del funcionamiento de la herramienta Filter Design & Analysis Tool (FDA tool) de Matlab para la obtención de los coeficientes de los filtros.
- El análisis del funcionamiento del circuito integrado LM3916 para la construcción del vúmetro Led.
- El análisis del funcionamiento de la herramienta Orcad Layout para el diseño de una placa PCB.

2. DESCRIPCION DEL PROYECTO.

Como se muestra en el siguiente esquema, el procesado de la señal se dividirá en 3 etapas.

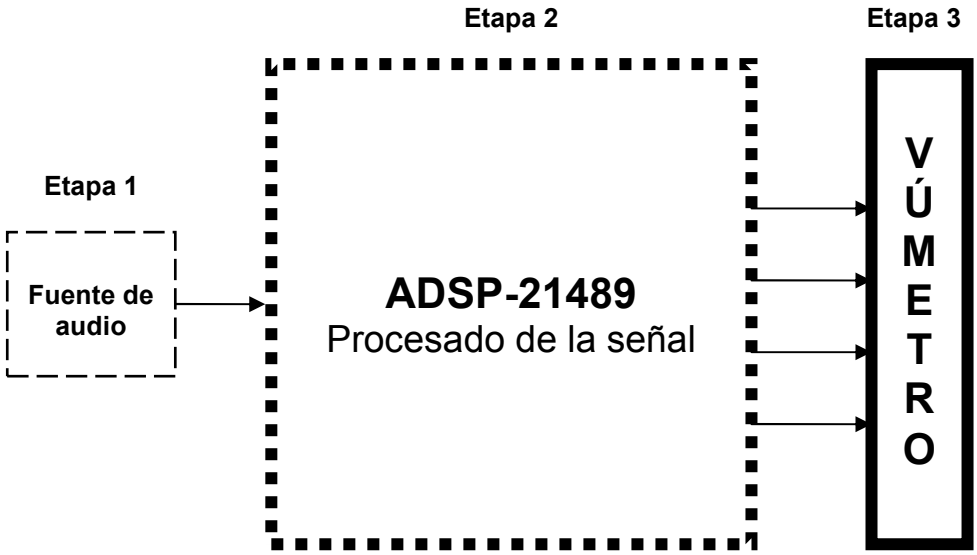


Figura 1 Esquema de funcionamiento.

La primera etapa será encaminar nuestra señal analógica de audio hacia el DSP donde internamente será convertida a una señal digital en un ADC (Analog Digital Converter) para después ser procesada.

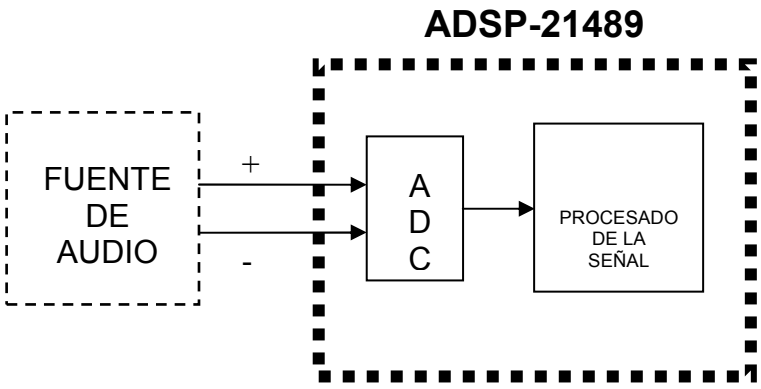


Figura 2 Primera etapa.

En la segunda etapa nuestra señal de audio, ahora digital, será procesada y encaminada a 4 salidas independientes OUT L1, OUT R1, OUT L2 y OUT R2.

Estas salidas están asociadas a los botones SW8, SW9, SW10 y SW11 respectivamente (los cuales vienen integrados en nuestro DSP y de los que hablaremos mas adelante).

En las salidas OUT L1, OUT R1 y OUT L2, se podrá variar el tipo de filtrado, dependiendo de las veces que se pulse cada botón, es decir, cada vez que se pulse un botón el filtrado en esa salida cambiará entre un filtrado Paso Bajo, filtrado Paso Banda y filtrado Paso Alto.

En cambio en la salida OUT R2, únicamente habrá dos tipos de señales, la primera será la señal de entrada, sin ningún tipo de procesado y cuando se pulse el botón SW11, la señal de salida será la señal ecualizada, para ello sumaremos las tres señales procesadas con los distintos filtros, y disminuiremos su amplitud para evitar la saturación de la señal.

La obtención de los coeficientes de los filtros se hará mediante la herramienta FDAtool de Matlab. Dichos coeficientes serán calculados para que haya el menor rizado posible a la hora de filtrar nuestra señal.

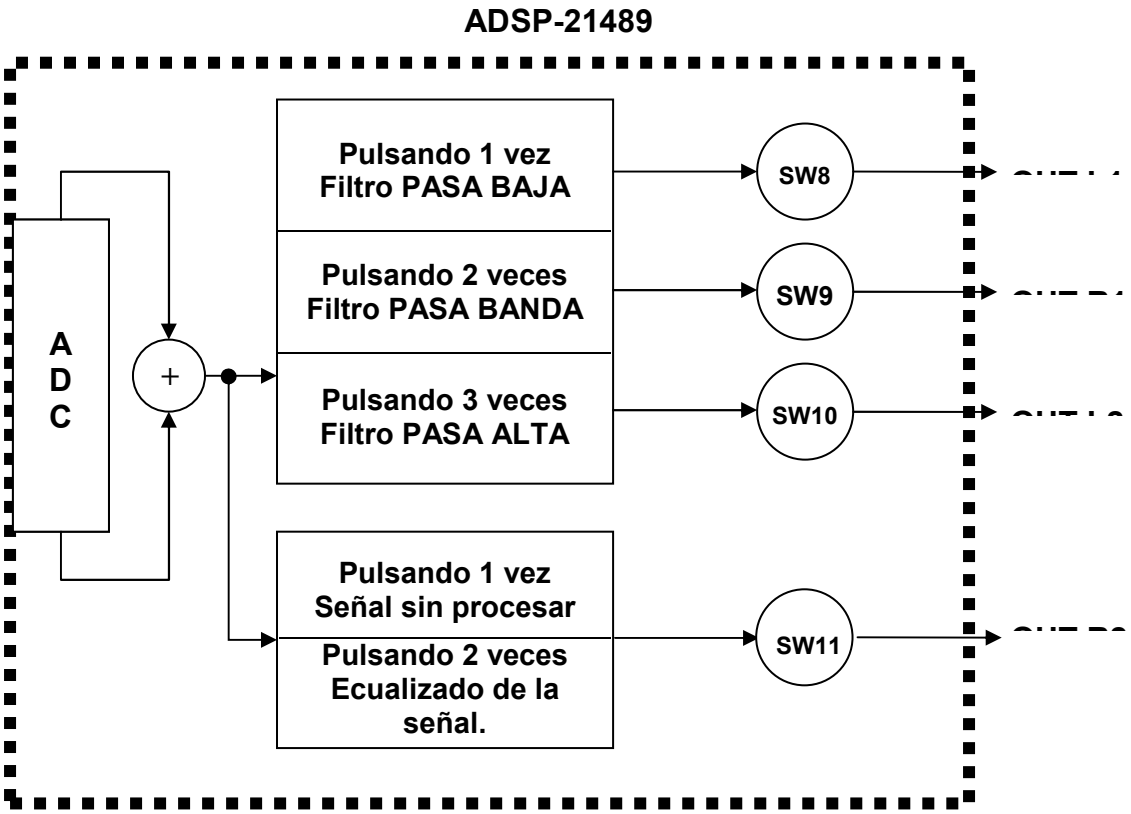


Figura 3 Segunda etapa.

La tercera etapa será la visualización del nivel de señal en el vúmetro óptico led.

Dicho vúmetro se podrá conectar a cualquiera de las salidas para poder ver su nivel de potencia en cada caso.

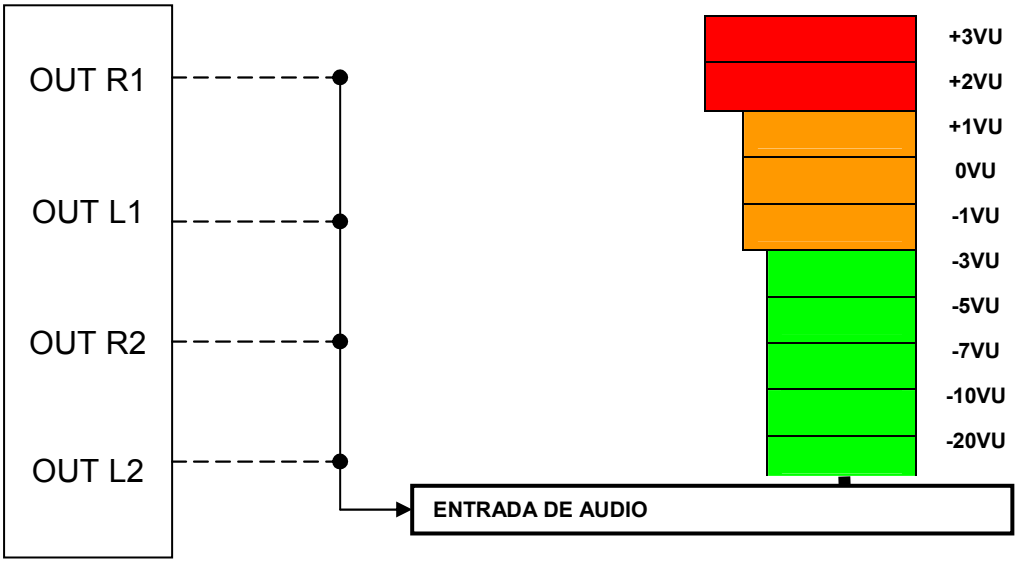


Figura 4 Tercera etapa.

3. HERRAMIENTAS DE TRABAJO.

3.1. DSP SHARC DE ANALOG DEVICES.

¿QUÉ ES UN DSP?

Un procesador digital de señales o DSP (sigla en inglés de digital signal processor) es un sistema basado en un procesador o microprocesador que posee un conjunto de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad. Debido a esto es especialmente útil para el procesado y representación de señales analógicas en tiempo real.

Fue en la década de los 80 cuando aparecieron los primeros procesadores digitales de señales lanzados por **Texas instruments,**y gracias a la aparición en el mercado de procesadores digitales lo suficientemente rápidos y potentes, como para poder implementar los algoritmos de tratamiento de señales de forma económica y sencilla estos tienen ahora un papel fundamental en el mundo del procesamiento digital de señales, ya que son de gran utilidad en distintos campos y aplicaciones, como pueden ser en campos acústicos, biomédicos y telecomunicaciones, por nombrar solo algunos.

Como se muestra en el siguiente esquema, el funcionamiento básico de un DSP se resume en: leer una nueva muestra/bloque de muestras de un dispositivo de entrada, convertir la señal analógica a digital, realizar el procesado lo mas rápido posible, convertir de nuevo la señal de digital a analógica, enviar el resultado a un dispositivo de salida, y estar preparado para la siguiente muestra/boque de entrada.

Figura 5 Esquema de funcionamiento básico del DSP.

DIAGRAMA DE BLOQUES DEL PROCESADOR ADSP-21489

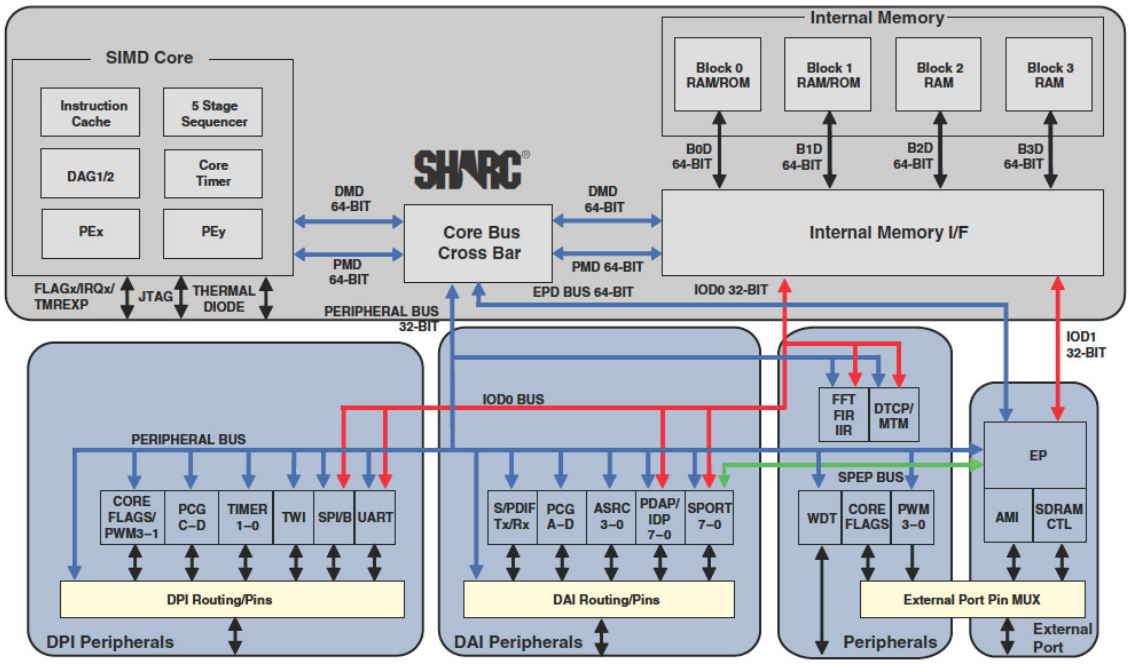


Figura 6 Diagrama de Bloques del procesador ADSP-21489.

El núcleo SIMD se compone de:

- Dos elementos de procesado (PEx, PEy), cada uno con:
  - ALU
  - Multiplicador
  - Shifter
  - Set de registros (16 primarios, 16 secundarios, para acelerar el cambio de contexto en ISRs!)
- Data address generators (DAG1, DAG2) con capacidad para gestionar hasta 32 buffers circulares.
- Caché de instrucciones y program sequencer de 5 etapas
- Temporizador
- PEx siempre está activo. PEy puede ser habilitado/inhabilitado
- Si la instrucción se lee de la caché, es posible leer hasta 4 operandos de la memoria en un ciclo (2 PM, 2 DM)

Para mas información sobre el funcionamiento interno del ADSP-21489 se adjunta el datasheet.

3.2. LA PLACA DE DESARROLLO ADSP-21489 EZ-BOARD.

Los procesadores SHARC se basan en una arquitectura de 32 bits súper Harvard (de ahí el nombre SHARC) que incluye una arquitectura de memoria compuesta por dos grandes bloques SRAM de dos puertos “on-chip”, junto con un sofisticado procesador IO, que da al procesador SHARC el ancho de banda necesario para la realización de cálculos a una velocidad muy alta.

La placa de evaluación esta diseñada para usarse junto con los entornos de desarrollo como CrossCore, Embedded Studio y VisualDSP++, este último como ya veremos mas adelante, será el que usaremos para implementar nuestro programa, el cual utilizaremos a la hora de filtrar las señales.

Para acceder al DSP desde el ordenador (PC), se puede hacer a través del Debug Agent Board, que es un emulador ICE/ICD de bajo coste para placas de desarrollo de Analog Devices y de terceros autorizados, con el que se puede acceder tanto al DSP como a los periféricos de la placa de desarrollo; este se conecta a la placa de desarrollo mediante los conectores P1 (JTAG) y ZP1 y dispone de un conector mini USB al PC de desarrollo

También se puede acceder a través de un emulador externo (denominado ICE o ICD) conectado a la placa de desarrollo que se comunica con el DSP mediante interfaz JTAG y con el PC mediante USB

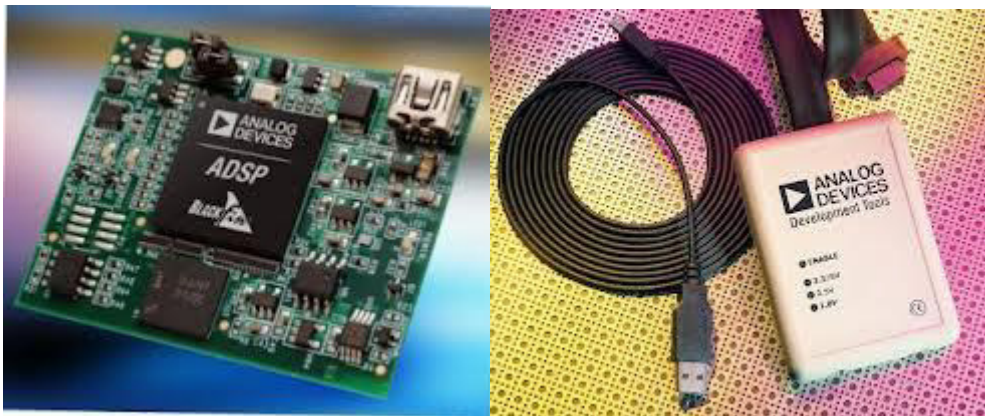


Figura 7 Debug Agent Board (izquierda) y Emulador externo ICE o ICD(Derecha).

Vista General de la placa de desarrollo ADSP-21489 EZ-BOARD.

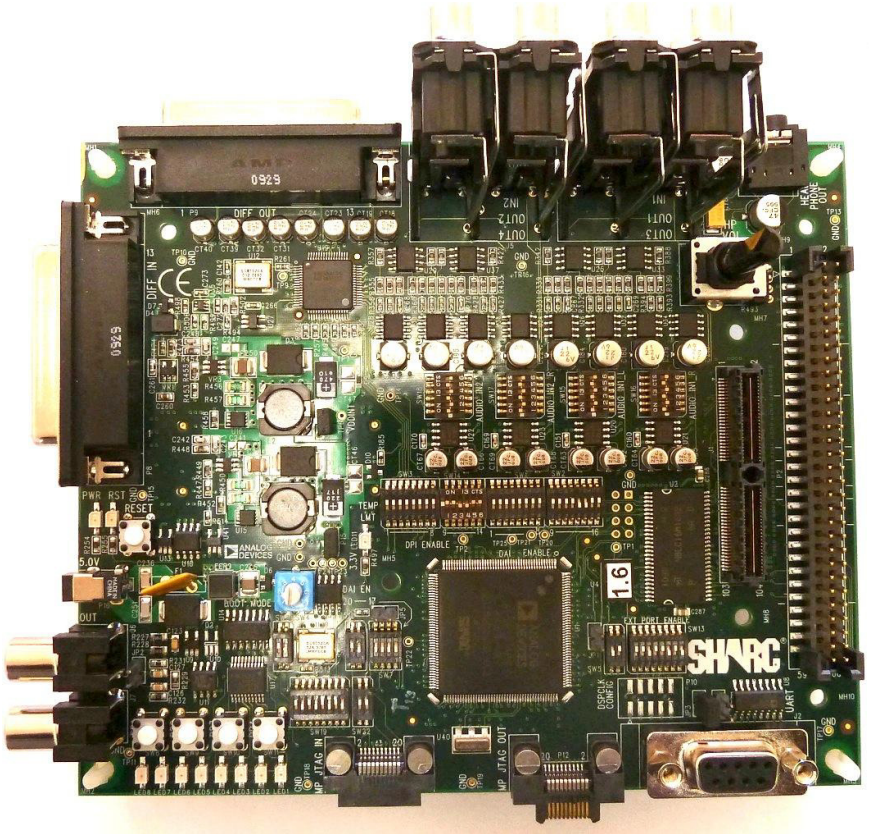


Figura 8 Vista general de la placa de desarrollo (top)

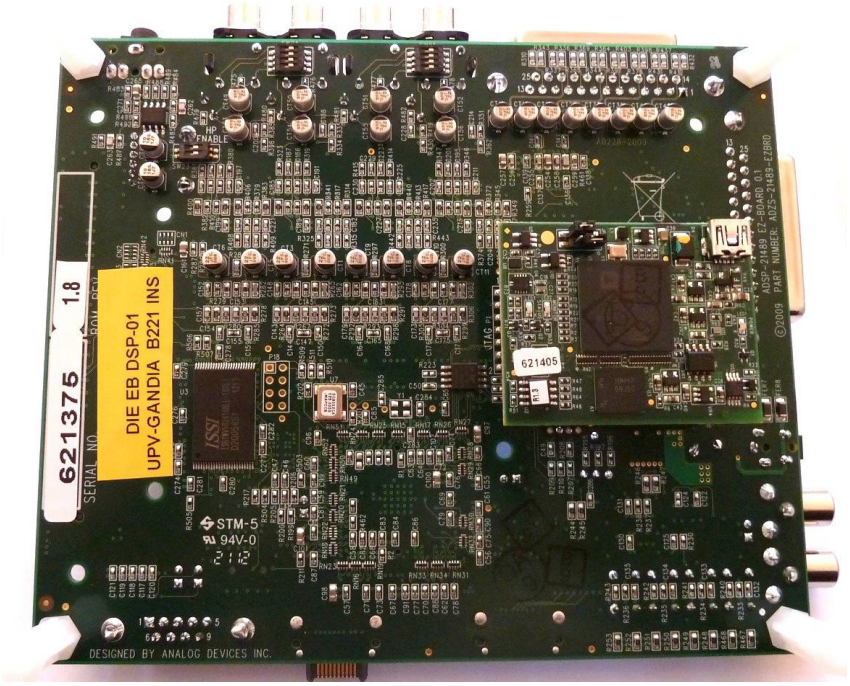


Figura 9 Vista general de la placa de desarrollo (bottom).

Arquitectura del sistema

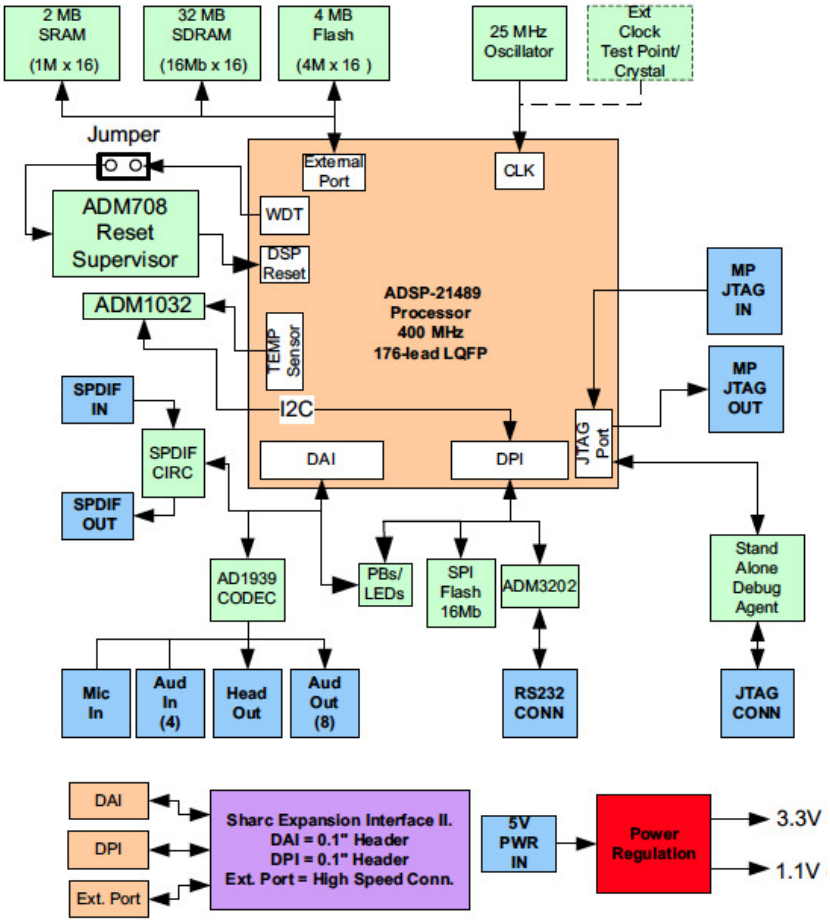


Figura 10 Arquitectura del sistema de la placa ez board

Características técnicas.

La placa cuenta con:

- **Procesador de Analog Devices ADSP-21489 SHARC.**
  - Núcleo de rendimiento hasta 400 MHz.
  - Conjunto LQFP de 176 pines.
  - Oscilador de 25MHz CLKIN.
  - 5 Mb de memoria RAM interna.
- **Memoria externa en la placa de desarrollo**
  - Memoria flash paralela.
    - Numonyx M29W320EB - 4 MB (4M x 8 bits), el Dsp arranca de esta memoria.
  - Memoria SDRAM.
    - Micron MT48LC16M16A2P6A2 - 16 Mb x 16 bits (Hasta 166Mhz).
  - Memoria asíncrona (SRAM)

        - ISSI IS61WV102416BLL-10TLI - 1M x 16 bits (2 MB)
    - Memoria flash SPI
        - Numonyx M25P16 - 16 Mb.
        - Se puede configurar la placa para que arranque desde esta memoria.
- **Interfaz de audio analógica**
    - Códec de audio Analog Devices AD1939.
        - Se conecta por bus I2s al bloque DAI del DSP.
        - Soporta hasta 4 entradas analógicas(2 pares estéreo) y 8 salidas analógicas(4 pares estéreo), con frecuencia de muestreo de hasta 192kHz.
        - El bus I2S consta como mínimo de 3 líneas:
            - Bit clock (BLCK)
            - Word clock – o Leith right clock (LRCLK)
            - Una línea de datos multiplexada (canal derecho/izquierdo)
        -

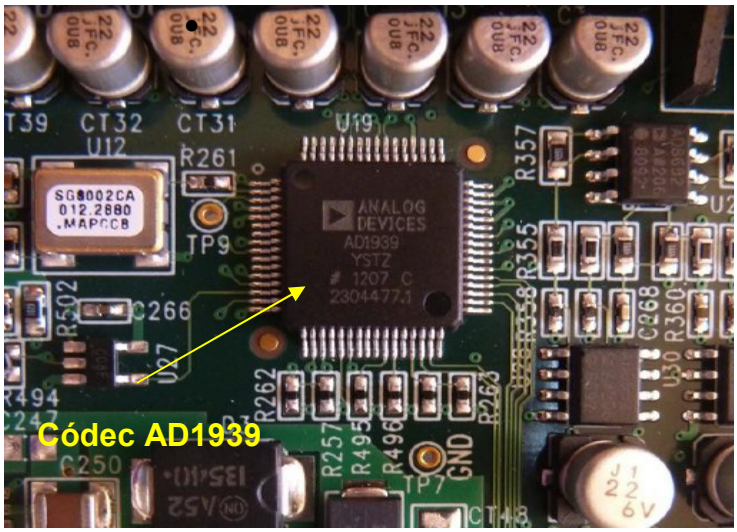


Figura 11 Códec de audio AD1939

    - 4 x 2 conectores RCA para ocho canales de salida estéreo
    - 4 x 1 conectores RCA para cuatro canales de entrada estéreo
    - Dos conectores DB25 para diferentes entradas / salidas
    - Toma de 3,5 mm para auriculares con control de volumen conectado a una de las salidas estéreo

Figura 12 Conectores entrada salida RCA

- **Interfaz de audio digital (S / PDIF)**
  - Salida de conector phono RCA
  - La entrada de jack phono RCA

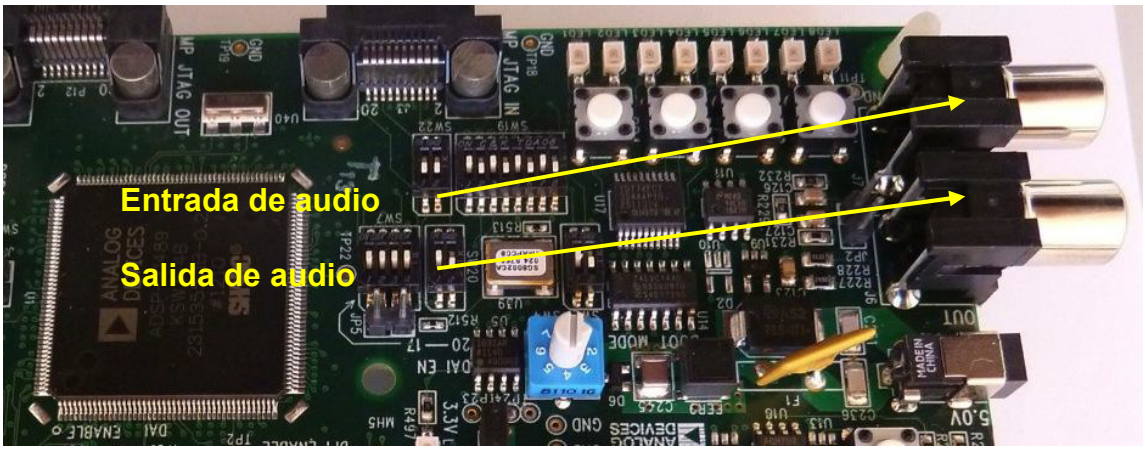


Figura 13 Interfaz digital (S/PDIF)

- Control de la temperatura
  - ON Semiconductor ADM1032
  - El sensor de temperatura local y remota
- **Puerto Serie**
  - El DSP dispone de una UART. A través de un transceiver (ADM3202) se transforma en una interfaz RS-232 (señales RX,TX,RTS Y CTS) que se lleva al exterior mediante un conector DB-.9.

Figura 14 Conector DB-9

- LEDs
  - Conectados a salidas de bloque DPI del DSP
  - Cuenta con once LEDS, uno para el reseteo de la placa (rojo), ocho de propósito general (ámbar), un sensor de temperatura (ámbar), y uno de encendido (verde)

Figura 15 Leds y Pulsadores

- Pulsadores
  - Hay 5 pulsadores (4 de propósito general y uno de reset). Dos están conectados al bloque DAI del DSP (para ser usados como entradas de interrupción) y dos a pines flag del DSP (se leen mediante el registro FLAG)
- Interfaz de ampliación II
  - El diseño de la nueva generación de la interfaz de expansión ofrece acceso a la mayoría de las señales del procesador
- Fuente de alimentación
  - 5V a 3.6 Amperios

3.3. VISUAL DSP++ 5.0

Para la realización del programa que se va a utilizar a la hora de filtrar señales, se utilizara el entorno de desarrollo Visual DSP++5.

Visual DSP++5.0 es una herramienta muy versátil, y proporciona numerosas opciones a la hora de generar un proyecto.

En este apartado se explicaran algunas de las características más importantes de dicho entorno, se nombraran las herramientas del código de desarrollo y la conexión a una sesión.

Características de Visual DSP++5.0

- Tiene amplias capacidades de edición. Se puede crear y modificar archivos de origen mediante el uso de sintaxis en varios lenguajes de programación. Puede depurar programas escritos en C, C++, o ensamblador.
- Tiene un fácil acceso a las herramientas de desarrollo de código. Analog Devices proporciona herramientas de desarrollo de código como: compilador de C/C++, ensamblador, enlazador (linker), divisor (splitter), y el cargador (loader); permite el uso de cuadros de dialogo en lugar de scripts de línea de comandos.
- Tiene una gran flexibilidad en la compilación de proyectos. Permite compilar archivos o proyectos de forma selectiva, es decir, podemos compilar archivos por separado, o todo el proyecto a la vez. Durante la compilación, permite ver el estado de esta, y en caso de error basta con un doble clic en el mensaje de error para que nos muestre donde esta el fallo.
- Gestión flexible del espacio de trabajo. Permite crear hasta diez áreas de trabajo y cambiar rápidamente entre ellas. Admite crear y depurar múltiples proyectos en una sola sesión.
- Tiene un control de depuración muy eficaz. Se pueden establecer puntos de interrupción (breakpoints) en símbolos y direcciones y, a continuación ejecutar el programa paso a paso para detectar problemas en la lógica de codificación

3.4. ORCAD CAPTURE Y ORCAD LAYOUT.

Como se ha descrito anteriormente, para la realización de la placa PCB que integrará todos los componentes electrónicos necesarios para la construcción del vúmetro, será necesario la utilización de las herramientas Orcad Capture Y Orcad Layout.

Con el editor de circuitos Orcad Capture realizaremos el esquema de conexiones de todos nuestros componentes electrónicos.

Una vez se halla realizado el esquema de conexiones y se compruebe que no hay ningún error, se procederá al diseño de la placa PCB. Para ello utilizaremos el editor de placas Orcad Layout.

Ambos procedimientos se explicarán más detalladamente en el apartado 5.2.

3.5. FILTROS FIR.

Un filtro FIR (*Finite Impulse Response* ó *Respuesta finita al impulso*) es un tipo de filtro digital, cuya respuesta a una señal impulso como entrada tendrá un número finito de términos no nulos.

Como se muestra en la siguiente imagen la estructura básica de este filtro es una línea de retardos.

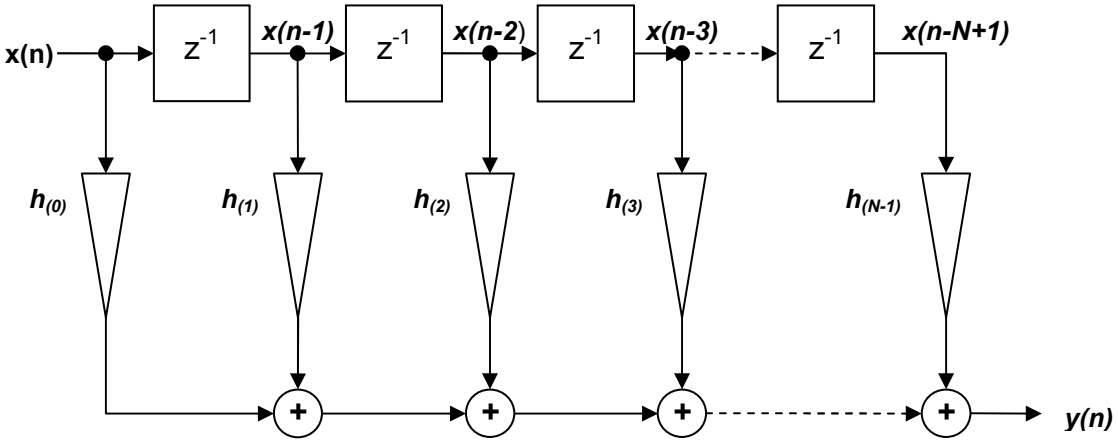


Figura 16 Esquema básico de funcionamiento de un Filtro FIR. Los términos h(n) son los coeficientes y z^{-1} los retardos.

En esta estructura la salida, *y(n),* es la suma ponderada de la entrada actual y un cierto número de entradas pasadas *x(n)*, que se puede expresar en la siguiente forma:

$$y(n) = h_0 x(n) + h_1 x(n-1) + \ldots h_{N-1} x(n-N+1) = \sum_{k=0}^{N-1} h_n \cdot x(n-k)$$

Figura 17 Ecuación 1.

donde $h_{(i)}$ son los coeficientes de la respuesta al impulso del filtro y *N* es el número de coeficientes. Dichos coeficientes representan el orden del filtro.

La transformada Z de la ecuación 1 seria:

$$H(z) = \frac{Y(z)}{X(z)} = h_0 + h_1 z^{-1} + h_2 z^{-2} + \cdots + h_{N-1} z^{-(N-1)} = \sum_{k=0}^{N-1} h_n z^{-k}$$

Figura 18 Ecuación 2.

3.6. FILTER DESIGN & ANALYSIS TOOL, "FDA TOOL" DE MATLAB.

Para el diseño e implementación de los filtros, se utilizará la herramienta FDA tool, incluida en Matlab, ya que nos permitirá configurar distintos parámetros del filtro y nos calculará los coeficientes de este rápidamente.

Podremos acceder a esta herramienta introduciendo en el cuadro de comandos de Matlab las palabras *fdatool* y pulsando intro

Una vez realizado el paso anterior se nos abrirá una ventana como la que vemos a continuación:

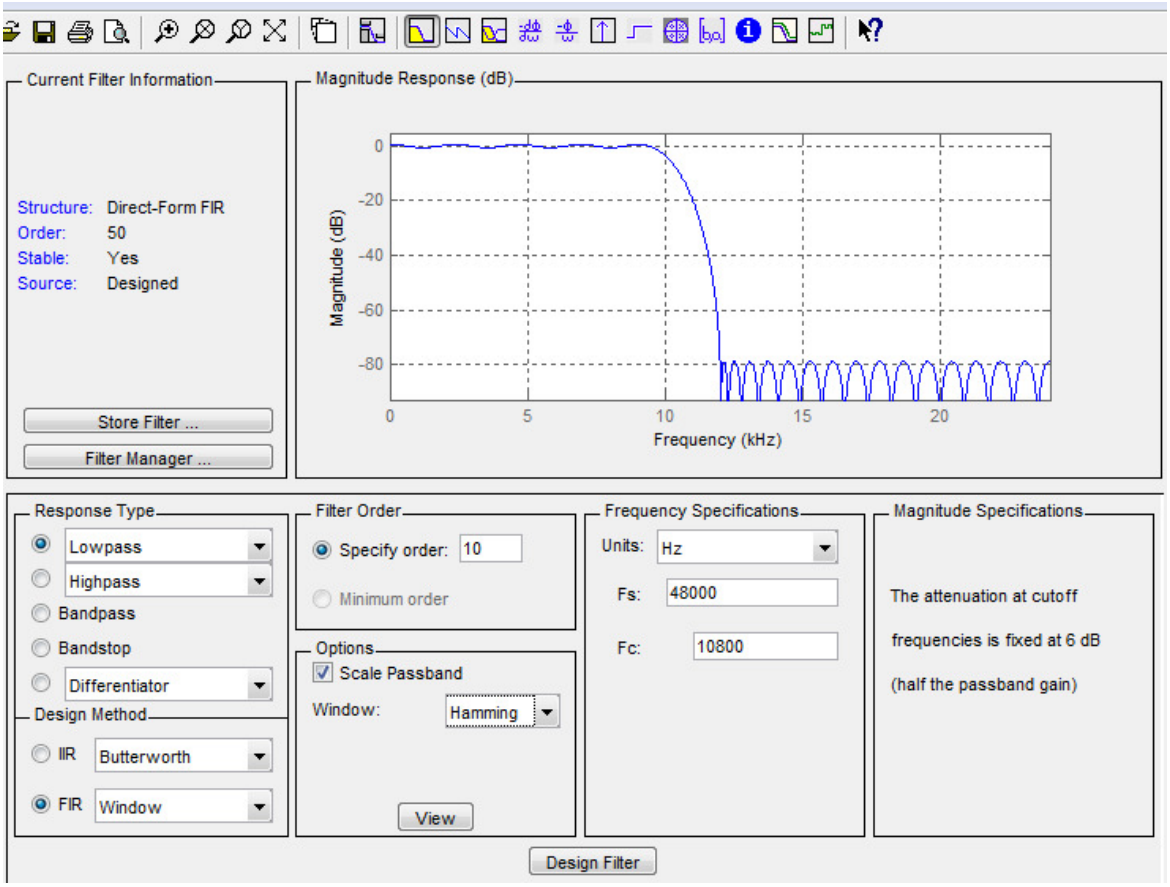


Figura 19 Cuadro de Configuración del Filtro Digital.

Como observamos en la imagen, la ventana se divide en tres bloques diferenciados:

El que aparece en la parte superior izquierda, nos proporciona tanto la información del filtro diseñado como la estructura del filtro, el orden, incluso si el filtro es o no estable.

En la parte superior derecha nos aparece un gráfico que representa la respuesta en magnitud (dB) del filtro diseñado.

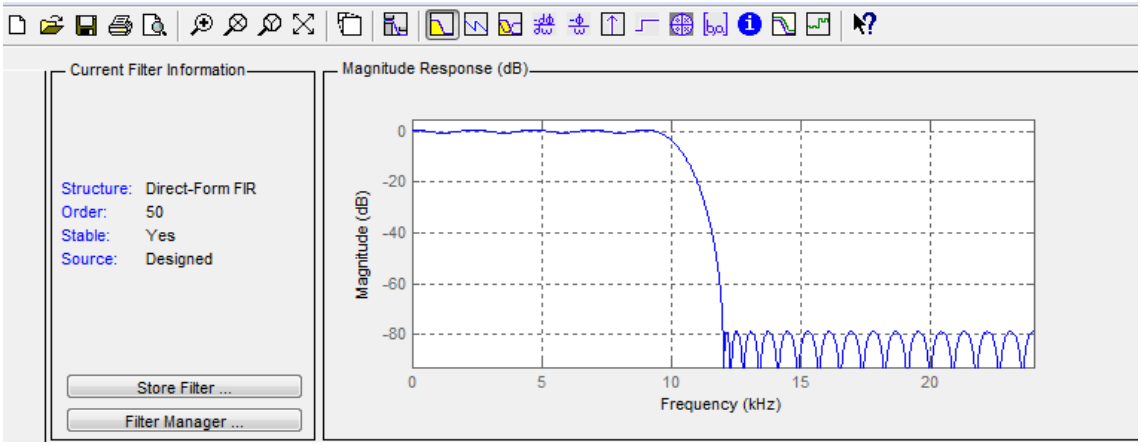


Figura 20 Información del Filtro.

El recuadro de la parte inferior de la imagen proporciona los parámetros de configuración del filtro organizados en cinco pequeñas secciones.

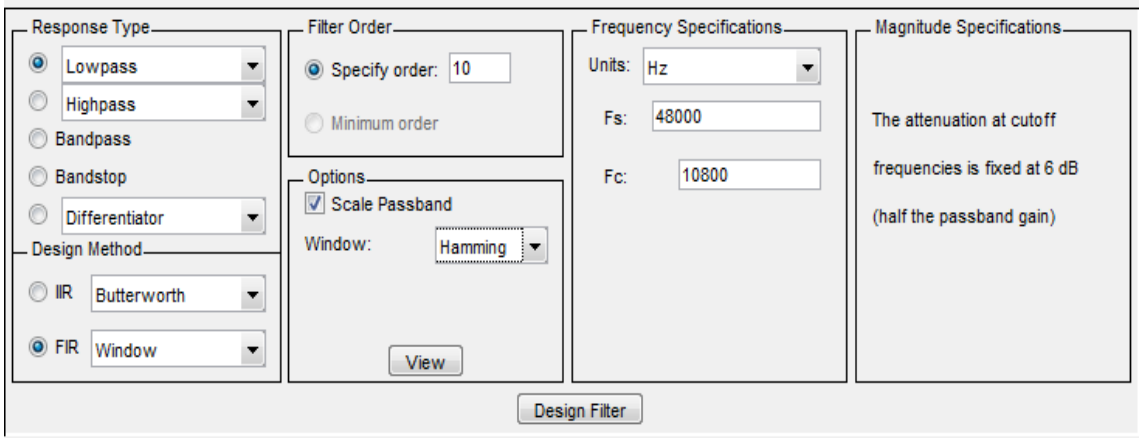


Figura 21 Parámetros de Configuración del Filtro.

En el primer recuadro de la parte inferior, podemos configurar el tipo de respuesta que deseamos para el filtro a diseñar. Permite la elección del tipo de filtro, paso-bajo, paso-alto, paso-banda, banda eliminada y diferenciador. Como hemos dicho anteriormente, este proyecto se centrará en los filtros paso-bajo, paso-alto y paso-banda.

El segundo recuadro de la parte inferior, permite la configuración del método de diseño. Podemos seleccionar entre filtros FIR e IIR, en este caso haremos uso de los filtros FIR ya que se ajusta mejor a nuestras necesidades.

Una vez seleccionada la opción de tipo de filtro a utilizar, tipo FIR, observamos que esta tiene asociada un menú desplegable que permite elegir el tipo de método que empleará dicho filtro. Este menú posee once opciones de configuración, siendo las mas comunes Equiripple, Least-squares (mínimos cuadrados) y Window (Enventanado).

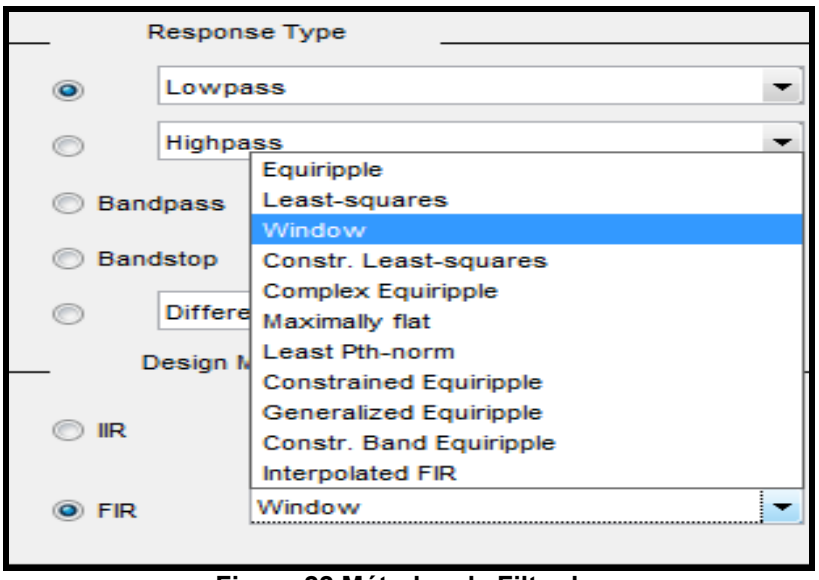


Figura 22 Métodos de Filtrado.

En este caso se ha decidido elegir el método Window, ya que se adapta mejor a nuestras necesidades y requiere de un menor número de operaciones a la hora de calcular los coeficientes. Tras seleccionar el método Window, nos aparece otro menú desplegable con 18 tipos de ventanas distintas que podemos utilizar.

Se ha decidido utilizar la ventana de Hamming, ya que esta posee un rizado en la banda de paso bastante bajo y es más simple su configuración, dado que depende básicamente del número de coeficientes, a mayor numero de coeficientes menor rizado en la banda de paso.

Una vez se ha establecido el método de diseño y el tipo de ventana, se procederá a diseñar y calcular los coeficientes de los filtros,como veremos más adelante en el apartado “4.3 Coeficientes.h”.

Para la obtención de los coeficientes, pulsaremos en la pestaña File y después en “Export”. Se abrirá un panel donde nos dejará elegir la forma de exportar los coeficientes; en el apartado “Export To”, seleccionaremos la opción: “Coefficient File (ASCII)”, y en Formato seleccionaremos Decimal. Al pulsar sobre el botón “Export” se nos generará un archivo de texto, donde encontraremos todos los coeficientes del filtro.

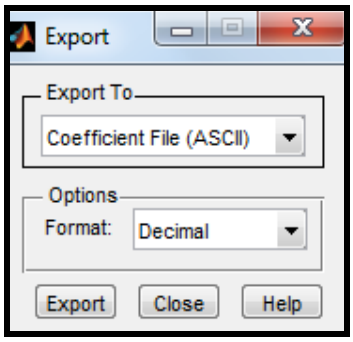


Figura 23 Exportación de los Coeficientes.

4. CODIGO EN VISUAL DSP++5.0.

4.1. CREACIÓN DE UNA NUEVA SESION.

Para la creación de nuestro programa, partiremos de un proyecto ejemplo que viene incluido en el paquete de instalación del ADSP-21489 EZ-BOARD, ya que así nos evitaremos problemas a la hora de realizar configuraciones internas y nos ahorraremos bastante trabajo. Una vez abierto dicho programa, procederemos a realizar las modificaciones convenientes como vamos a mostrar de aquí en adelante.

En primer lugar comenzaremos creando una nueva sesión de trabajo para conectar nuestra placa de EZ-BOARD.

Para ello pulsamos en session y seleccionamos new sesión.

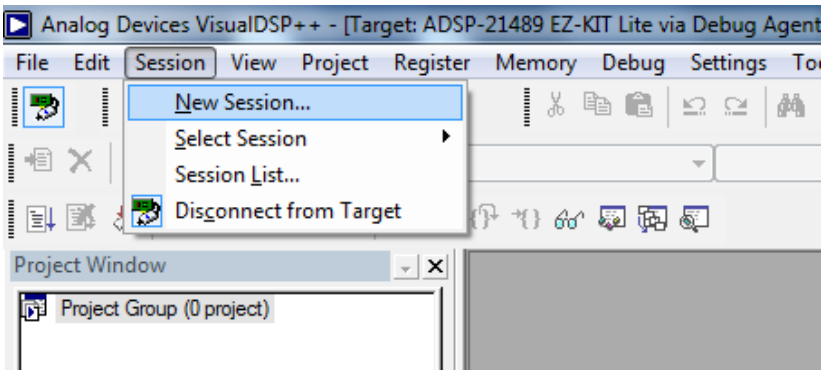


Figura 24 Creacion de una Nueva Sesión.

Tras realizar esta operación se nos abrirá una ventana en la que deberemos escoger que procesador vamos a usar, en nuestro caso el ADSP-21489.

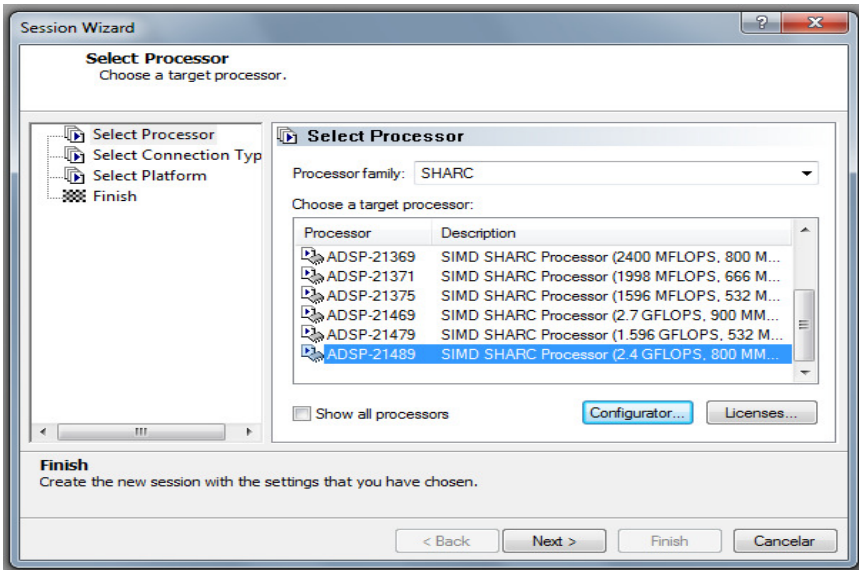


Figura 25Selección del Procesador.

Una vez seleccionado pulsaremos en next para seguir configurando nuestra sesión.

En la siguiente ventana que aparece se debe elegir el tipo de conexión que queremos utilizar, ya sea con nuestra placa EZ-Board, en modo Emulador, modo simulación o en Legacy Target. Dado que se va a trabajar con la placa, seleccionaremos la opción EZ-KIT Lite.

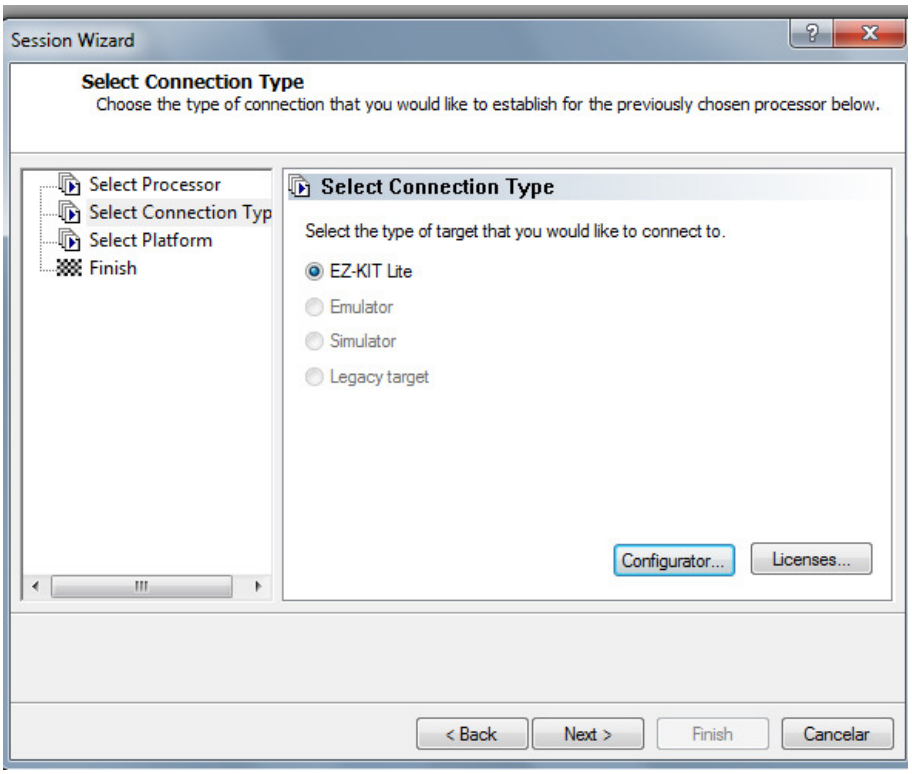
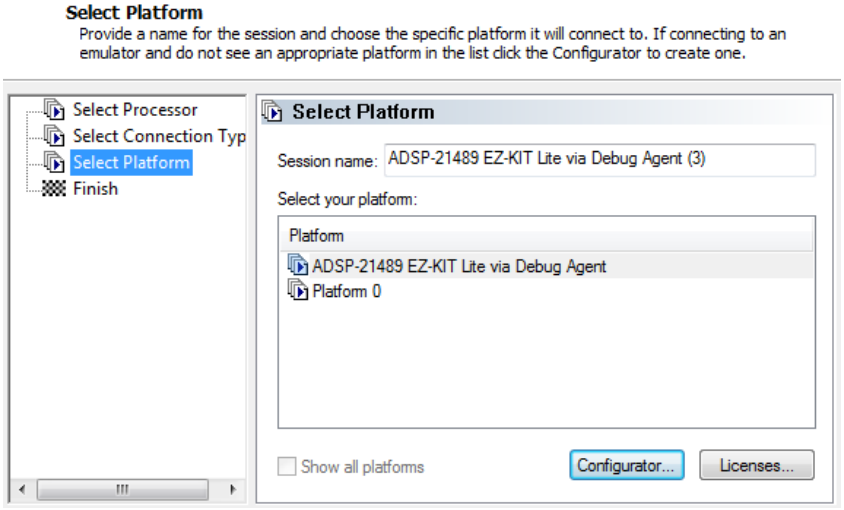


Figura 26 Tipo de Conexión.

Una vez seleccionada la opción deseada se hace clic en siguiente.

En la siguiente ventana, seleccionaremos el tipo de plataforma que vamos a utilizar para conectar nuestra placa y daremos un nombre a nuestra sesión. En este caso y como ya se comentó anteriormente, la placa se conectara mediante el Debug Agent Board. Después de seleccionar la opción deseada, se hace clic en siguiente, donde saldrá un cuadro informativo con los parámetros de nuestra sesión. Se hace clic en siguiente y ya estamos listos para comenzar.



Fiigura 27 Selección de Plataforma.

Una vez creada la sesión, se procederá a cargar el proyecto ejemplo que viene con la instalación del software VisualDSP++ 5.0.

Para ello se hará clic en open, Project y dentro de la carpeta de instalación del programa VisualDSP++ 5.0, se seleccionará la opción 214xx, dentro de esta "Examples", y posteriormente se seleccionará la carpeta "ADSP-21489 EZ-Board". En esta última carpeta se encontrarán varios ejemplos; el ejemplo a través del cual se comenzará este programa será "21489 AD1939 C Sampled-Based Talkhru 48 or 96 kHz"

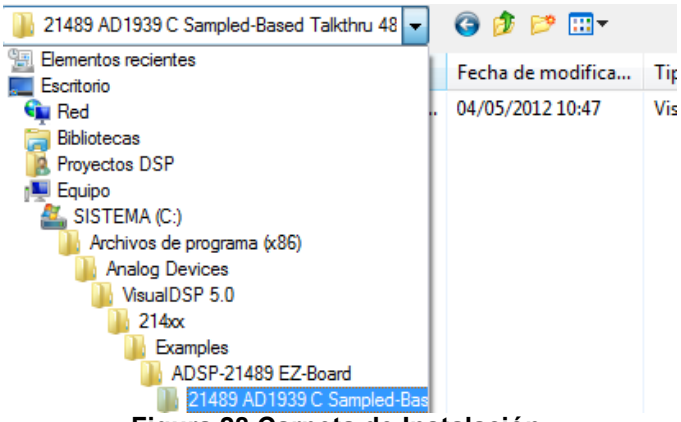


Figura 28 Carpeta de Instalación.

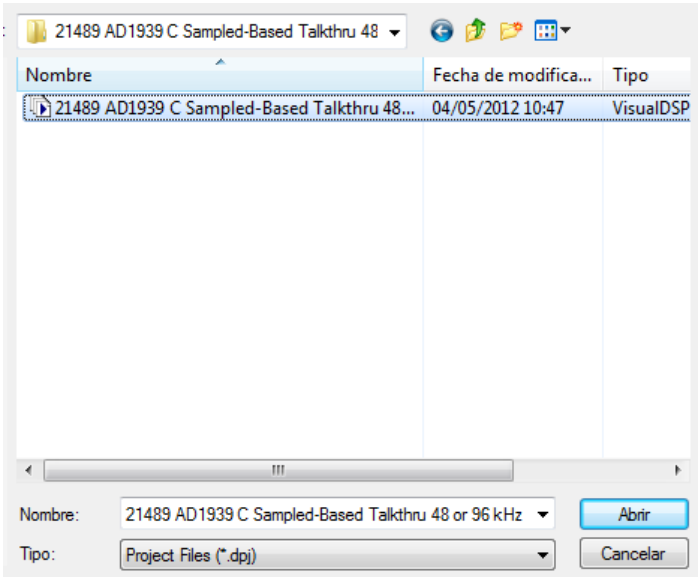


Figura 29 Selección del Proyecto.

Una vez abierto el ejemplo nos apareceran en el lado izquierdo del programa distintas carpetas y archivos, estos archvios configuran diversos parametros de nuestro DSP, los cuales vamos a explicar brevemente.

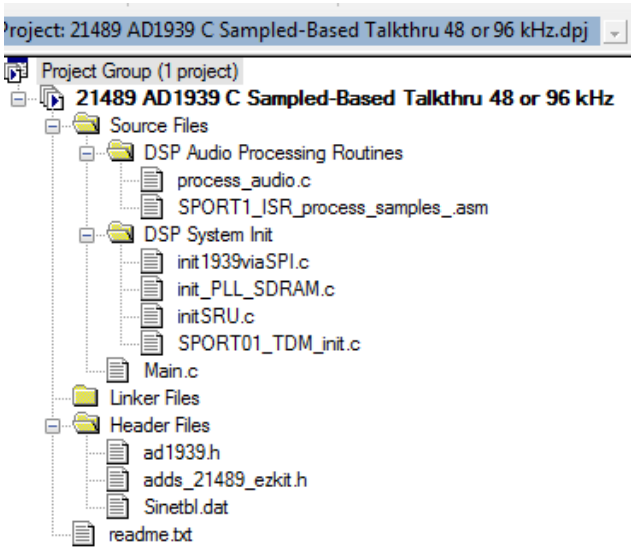


Figura 30 Archivos de Configuración Interna.

En el fichero process_audio.c se pueden configurar los canales de entrada y salida de audio de las muestras que llegan al DSP a través del códec AD1939, es decir, se puede decidir que las muestras recibidas entren por el canal 1, el canal 2 o ambos a la vez, y por que canales van a salir después de ser procesadas.

El fichero SPORT1_ISR_process_samples_.asm recibe los datos de entrada de los ADCs AD1939 a través del puerto SPORT1 y transmite los datos de audio procesado a los cuatro DAC AD1939 estéreo de salida a través del puerto SPORT0.

El fichero init1393viaSPI.c contiene las subrutinas para acceder a los registros de control del AD1939 a través del bus de datos SPI, y configurar los AD1939s para la "*comunicación serie*" (envío de un bit de información de manera secuencial) TDM (*Time Division Multiple Access)*.En este fichero también podemos configurar la frecuencia de muestreo, entre 48kHz y 96kHz.

El fichero init_PLL_SDRAM.c inicializa el PLL del DSP para las tasas del CCLK Y HCLK (reloj de núcleo CCLK) necesarias.

El fichero initSRU.c inicializa el DAI (Digital Audio Interface) para acceder al ADC (Analog Digital Converter) y DAC (Digital Analog Converter).Se utiliza para enviar y recibir muestras de audio en el códec AD1939 de la placa 21489 EZ-Board usando los puertos SPORT 0/1.

El fichero SPORT01_TDM_init.c configura los puertos de control SPORT1 RX y SPORT0 TX y los registros DMA (Acceso Directo de Memoria) para las operaciones multicanal.

El fichero ad1939.h contiene los registros y las direcciones de memoria para el códec de audio AD1939.

Dado que los ficheros explicados anteriormente son configuraciones internas del programa y apenas se va a realizar ningún cambio en ellos, vamos a crear los ficheros necesarios para poder realizar nuestro programa.

La función en la que se basará el programa, será en la función fir () integrada en la librería del entorno VisualDSP++5.0.Esta función implementa una respuesta finita al impulso basándose en la suma de productos. Las características del filtro (pasa baja, pasa alta, pasa banda, etc.) vienen dadas por el número y el tipo de coeficientes que calculemos con la herramienta FDA tool.

Como podemos observar en la descripción encontrada en el manual **"Visual DSP5.0++ Run-Time Library Manual"** en esta función el número de coeficientes del filtro está definido por el parámetro **"taps"** y dichos coeficientes deben de estar almacenados en orden inverso en un array llamado **"coeffs"**, es decir en la posición "0" del array "coeffs[ ]" (coeffs[0]) deberá estar almacenado el valor del último coeficiente del filtro y en la ultima posición del array anteriormente citado (coeffs[taps-1]) deberá estar guardado el valor del primer coeficiente del filtro, dicho array ha de estar localizado en la memoria del programa (pm) para que pueda ser utilizado

Cada filtro debe tener su propia línea de retardo, que está representada por el array "state" el array contiene un puntero en la primera posición de la línea de retardo, seguido de los valores de la línea de retardo. Por tanto, la longitud del array "state", debe de tener una posición de memoria más que el array "taps"

Para que todo funcione correctamente, el array state debe ser inicializado a cero antes de llamar a la función fir () por primera vez.

La sintaxis de la función la podemos encontrar en el manual y sería: **Señal_de_salida= fir (Señal_entrada, coeffs, state, TAPS);**

Por lo tanto, y una vez que sabemos todo lo necesario para que la función fir () se ejecute correctamente, deberemos crear, un archivo donde estén declarados todos los coeficientes de cada filtro, al que se le llamara **coeficientes.h** y otro archivo en el que se inicialicen el estado de las líneas de retardo al que se le ha dado el nombre de ***EEA_initFiltros.c***. Para las interrupciones de los pulsadores y el encendido de los leds, se creará otro fichero que contenga dicho código, al que se le ha dado el nombre de ***EEA_leds.c***

Para ello se comenzará creando un fichero que contenga todos los prototipos de las futuras funciones que se utilizaran durante el proyecto. En este fichero se declararan tanto las funciones como las variables que van a ser utilizadas, a este fichero le llamaremos ***EEA_DSP_Sampled_Header.h***

Por último se creará otro fichero llamado ***EEA_DSP_Sampled.c*** en el cual se encontrará la función fir () que realizara el filtrado de nuestras señales.

A continuación se va a explicar el contenido de cada fichero nuevo que hemos generado.

4.2. EEA_DSP_SAMPLED_HEADER.H

Este fichero contiene tanto los prototipos de las funciones que vamos a utilizar como las variables que hay declaradas dentro de ellas. En el definiremos la frecuencia de muestreo que va a ser utilizada, 48 kHz., y se declararán todas las funciones que usaremos en los ficheros EEA_initFiltros.c, EEA_leds.c, EEA_DSP_Sampled.c .

```
//Frecuencia de muestreo
#define USE_48_KHZ_SAMPLE_RATE
#ifdef  USE_48_KHZ_SAMPLE_RATE
#define F_SAMPLE 48000
#endif
/*****************************************************************************************
**  Funciones para la inicializacion de los filtros,de los leds y de los pulsadores **
*****************************************************************************************/
void    EAA_DSP_Process_Samples(void); //Funcion que filtra la señal
void    LOWFIRInit(void);   //Funcion inicialia linea de retardo del Filtro pasa baja
void    BANDFIRInit(void);  //Funcion inicialia linea de retardo del Filtro pasa banda
void    HIGHFIRInit (void); //Funcion inicialia linea de retardo del Filtro pasa alta
void    Init_LEDs(void);              //Función inicializar leds.
void    Init_PushButtons(void);       //Función inicializar botones.

/*********************************************************************
**       Funciones para las interrupciones de los botones:       **
*********************************************************************/
void SW8_IRQ1_handler(int sig_int);                                 /**/
void SW9_IRQ2_handler(int sig_int);                                 /**/
void SW10_SW11_DAI_handler(int sig_int);                            /**/
/*********************************************************************/

//Constantes numericas del fichero coeficientes.h
#define LOWTAPS 301  //Numero de coeficientes del filtro Pasa Baja
#define BANDTAPS 301 //Numero de coeficientes del filtro Pasa Banda
#define HIGHTAPS 301 //Numero de coeficientes del filtro Pasa Alta

//Declaración de los arrays de las lineas de retardo
extern float statelow[];
extern float stateband[];
extern float statehigh[];

//Variables leds y botones.
volatile bool gb_sw8_pushed;
volatile bool gb_sw9_pushed;
volatile bool gb_sw10_pushed;
volatile bool gb_sw11_pushed;
int Pulsado,Pulsado2,Pulsado3,Pulsado4;
int Veces;
int Veces2;
int Veces3;
int Veces4;
extern int salir;
//Declaración de entradas y salidas de audio
extern float InputSample_L;
extern float InputSample_R;
extern float OutputSample_R1;
extern float OutputSample_L1;
extern float OutputSample_R2;
extern float OutputSample_L2;
```

Figura 31 Fichero EEA_Sampled_Header.h

4.3. COEFICIENTES.H

En este fichero definiremos los coeficientes que se utilizarán para cada filtro. Como se ha explicado anteriormente el diseño de los filtros se ha realizado con la herramienta FdaTool, de Matlab como se muestra a continuación.

4.3.1. DISEÑO DE FILTROS FIR

4.3.1.1. Pasa Baja.

Para el diseño del filtro pasa baja, se intentará abarcar todas las frecuencias que se engloban dentro de los tonos graves, es decir, las frecuencias bajas, correspondientes a las 4 primeras octavas, esto es, desde los 16 Hz a los 256 Hz.

Por lo tanto la frecuencia de corte del Filtro Pasa baja, será a 256 Hz.

Para que tenga un menor rizado se ha decidido realizar un filtro de orden 300.

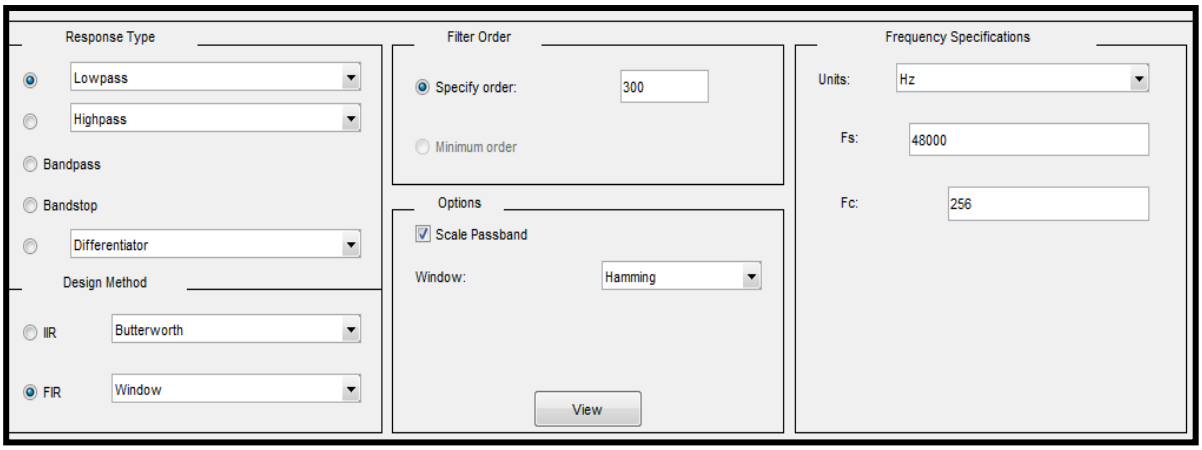


Figura 32 Configuración del Filtro Pasa Baja.

Una vez diseñado el filtro, el programa nos muestra la imagen que representa la respuesta en magnitud (dB) del filtro diseñado. En este caso seria:

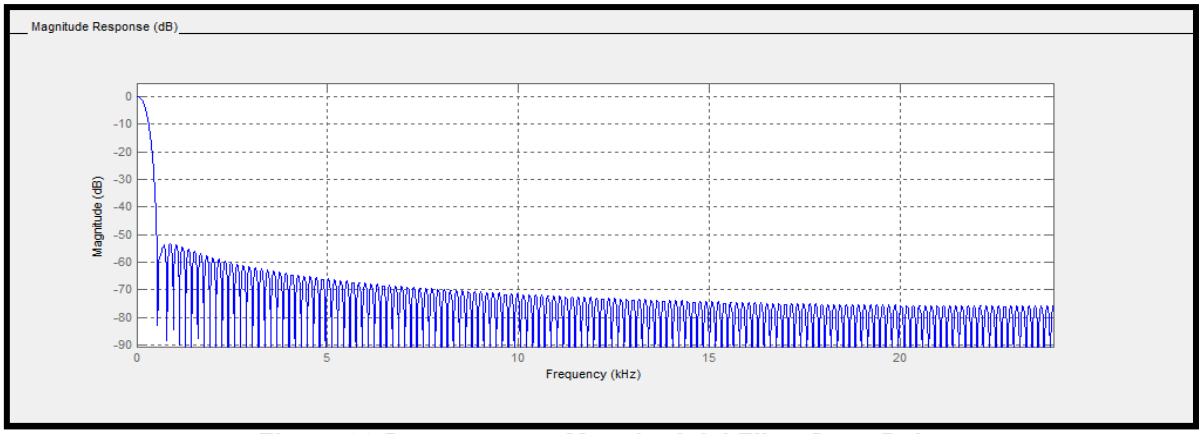


Figura 33 Respuesta en Magnitud del Filtro Pasa Baja.

Para la obtención de los coeficientes se actuaría de la misma forma que se ha explicado anteriormente en el apartado **3.6**

4.3.1.2. Pasa banda.

Para el diseño del filtro pasa banda, se intentará abarcar todas las frecuencias que se engloban dentro de los tonos medios, es decir, las frecuencias medias, correspondientes a las octavas quinta, sexta y séptima, esto es, de 256 Hz a 2 Khz.

Por lo tanto la primera frecuencia de corte del Filtro Pasa banda, será a 256 Hz y la segunda estará situada a 2Khz.

Para que tenga un menor rizado se ha decidido realizar un filtro de orden 300.

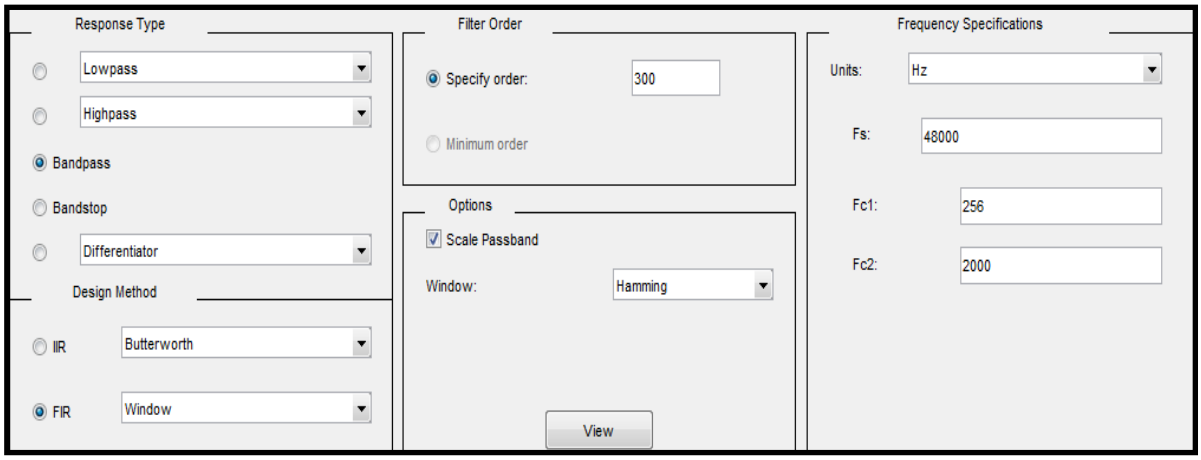


Figura 34 Configuración del Filtro Pasa Banda.

Una vez diseñado el filtro, el programa nos muestra la imagen que representa la respuesta en magnitud (dB) del filtro diseñado. En este caso seria:

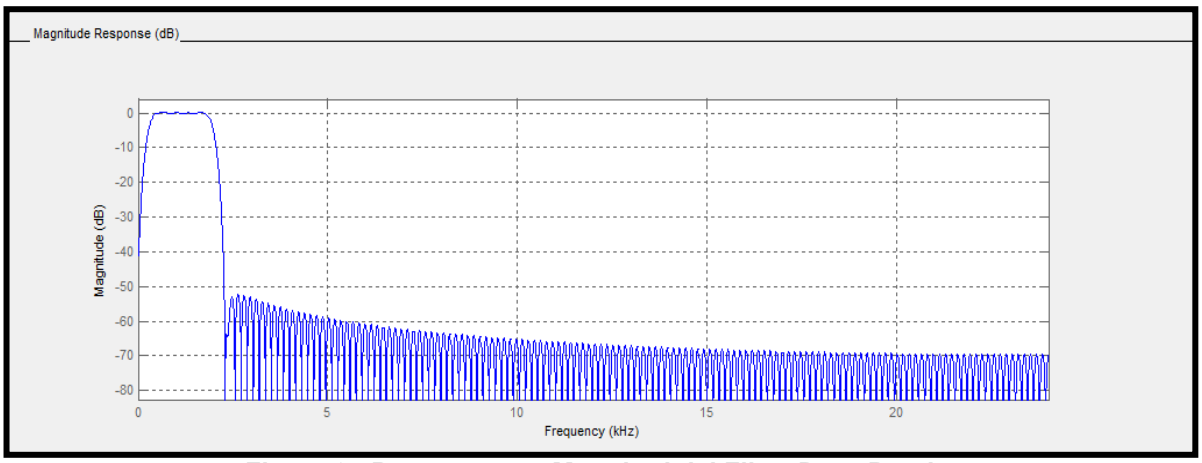


Figura 35 Respuesta en Magnitud del Filtro Pasa Banda.

Para la obtención de los coeficientes se actuaría de la misma forma que se ha explicado anteriormente en el apartado **3.6**

4.3.1.3. Pasa Alta.

Para el diseño del filtro pasa alta, se intentará abarcar todas las frecuencias que se engloban dentro de los tonos agudos es decir las frecuencias, altas, correspondientes a las tres últimas octavas, esto es, de 2 kHz hasta poco más de 16 kHz.

Por lo tanto frecuencia de corte del Filtro Pasa alta, será a 2 kHz.

Para que tenga un menor rizado se ha decidido realizar un filtro de orden 300.

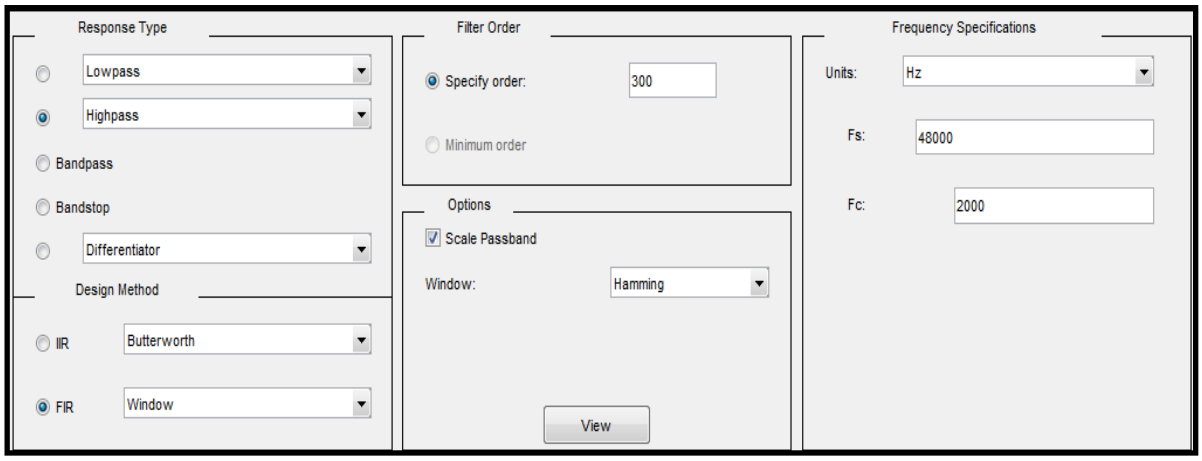


Figura 36 Configuración del Filtro Pasa Alta.

Una vez diseñado el filtro, el programa nos muestra la imagen que representa la respuesta en magnitud (dB) del filtro diseñado. En este caso seria:

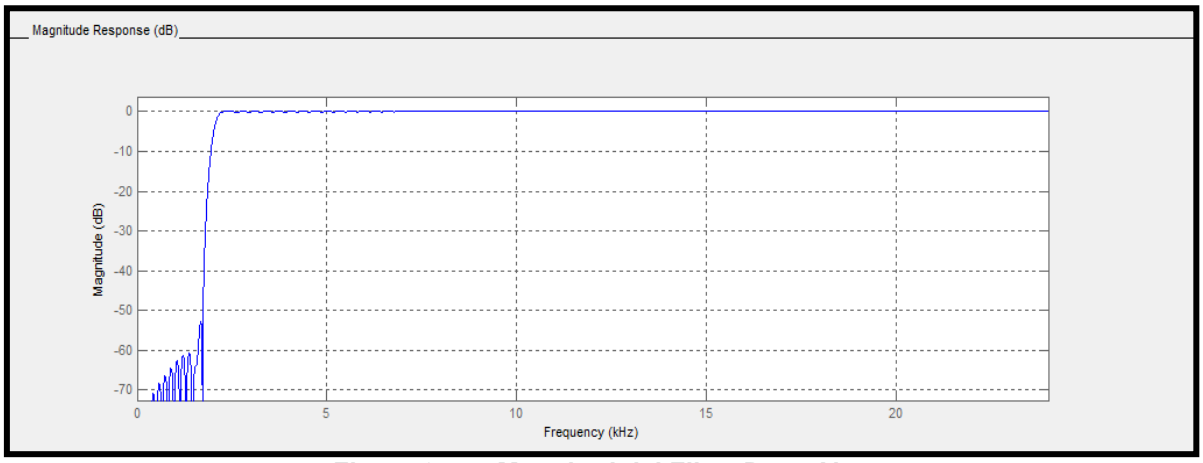


Figura 37 en Magnitud del Filtro Pasa Alta.

Para la obtención de los coeficientes se actuaría de la misma forma que se ha explicado anteriormente. En el apartado 3.6

Una vez que hemos obtenido los coeficientes de cada filtro, procederemos a definirlos dentro de nuestro fichero coeficientes.h.

Para ello primero declaramos las variables LOWTAPS, BANDTAPS, Y HIGHTAPS, cuyo valor será igual al número de coeficientes, es decir, si hemos calculado 301 coeficientes, el valor de estas variables ha de ser 301.

Los coeficientes de los filtros los guardaremos en los arrays coeefslow (para el pasa baja),coeffsband (para el pasa banda) y coeffshigh (para el pasa alta), el tamaño de estos arrays debe ser igual al numero de coeficientes del filtro.

```
////////////////////////////////////////////////////////////////////
//                                                                //
//                                                                //
//                  COEFICIENTES FILTROS                          //
//                                                                //
//                                                                //
////////////////////////////////////////////////////////////////////

//================================================================//
//                                                                //
//                       FILTRO PASA BAJA                         //
//                                                                //
//================================================================//


#define LOWTAPS 301 //Definimos la variable LOWTAPS que es el número
                    //de coeficientes para el filtro pasa baja.

float pm coeffslow[LOWTAPS] = { //Declaramos el array coeffslow que contiene
                                // los coeficientes del filtro Pasa Baja
-0.00016286103582181718   ,
-0.00016585573292492099   ,
```

Figura 38 Almacenamiento de Coeficientes Pasa Baja.

Para los filtros pasa banda y pasa alta se realizará la misma operación con sus coeficientes correspondientes.

```
//================================================================//
//                                                                //
//                       FILTRO PASA BANDA                        //
//                                                                //
//================================================================//

#define BANDTAPS 301  //Definimos la variable BANDTAPS que es igual al
                      //numero de coeficientes del filtro pasabanda.

float pm coeffsband[BANDTAPS] = {   //Declaramos el array coeffsband que contiene
                                    // los coeficientes del filtro Pasa Banda
0.00033132394867912491        ,
 0.00032981646012149215       ,
```

Figura 39 Almacenamiento de Coeficientes Pasa Banda.

```
//================================================================//
//                                                                //
//                       FILTRO PASA ALTA                         //
//                                                                //
//================================================================//


#define HIGHTAPS 301 //Definimos la Variable HIGHTAPS,
                     //que es el numero de coeficientes del filtro pasa alta.

float pm coeffshigh[HIGHTAPS] = {      //Declaramos el array coeffhigh que contien
-0.00016979400007409044              , // los coeficientes del filtro Pasa Alta
-0.00016531735158632O5               ,
```

Figura 40 Almacenamiento de Coeficientes Pasa Alta.

4.4. EEA_INITFILTROS.C

En este archivo inicializaremos el estado de las líneas de retardo de los filtros,se ha de incluir el archivo de cabecera "EEA_DSP_Sampled_Header.h" para que el programa reconozca las variables y las funciones declaradas dentro de este.

```
#include "EAA_DSP_Sampled_Header.h"

/**Estas Funciones inicializan las lineas de retardo de los filtros **/
//=============================================================//
//                      INICIO FILTRO PASA BAJA                //
//=============================================================//
float statelow[LOWTAPS+1];

void    LOWFIRInit(void){

    int i=0;
    for(i=0;i<LOWTAPS+1;i++)
        statelow[i]=0;

}
//=============================================================//
//                      INICIO FILTRO PASA BANDA               //
//=============================================================//
float stateband[BANDTAPS+1];

void    BANDFIRInit(void){

    int i=0;
    for(i=0;i<BANDTAPS+1;i++)
        stateband[i]=0;

}
//=============================================================//
//                      INICIO FILTRO PASA ALTA                //
//=============================================================//
float statehigh[HIGHTAPS+1];


void    HIGHFIRInit(void){

    int i=0;
    for(i=0;i<HIGHTAPS+1;i++)
        statehigh[i]=0;

}
```

Figura 41 Inicialización de las Líneas de Retardo.

4.5. EEA_LEDS.C

En este fichero incluiremos las interrupciones para el uso de los botones, y el encendido o apagado de los leds, se ha programado de tal forma que cada vez que se pulse un botón, se encienda un led distinto dependiendo del tipo de filtrado que se esté realizando en cada momento.

Cada botón tiene asociados dos leds, de este modo al ser pulsado por primera vez, se encenderá el primer led indicándonos que se esta realizando un filtrado paso bajo, al ser pulsado una segunda vez se apagará el led anterior y se encenderá el led contiguo indicándonos que se esta realizando un filtrado paso banda y por ultimo al ser pulsado una tercera vez se encenderán los dos leds indicándonos que se esta realizando un filtrado paso alto, si pulsamos una cuarta vez el programa volverá a empezar desde el principio, realizando así la misma acción que si se hubiese pulsado una sola vez. Como el código es bastante extenso se ha decidido no introducir capturas de pantalla, ya que este se adjunta en un archivo anexo.

4.6. EEA_DSP_SAMPLED.C

En este fichero se encuentra la función EAA_DSP_Process_Samples, dentro de esta función se encuentra todo lo necesario para poder realizar el filtrado de las señales.

Como se muestra en la figura 42 el primer paso será incluir el fichero EAA_DSP_Sampled_Header.h y todas las librerías necesarias, entre ellas la librería <filters.h>, la cual contiene la función fir().

Después se declararán, cinco variables donde se guardarán las diferentes señales. En la variable “Senal” se realizará la suma del canal izquierdo mas el canal derecho de nuestra señal de entrada, con el fin de convertir nuestra señal de entrada de estéreo a mono, para realizar posteriormente los filtrados correspondientes. También se declararán las salidas donde obtendremos nuestras señales.

En las variables “Senal2”,”Senal3” y “Senal4” se guardaran las señales procesadas mediante la función fir(), a las cuales se les realizará un filtrado Paso Bajo, Paso Banda y Paso Alto, respectivamente.

En la variable “Senal5” se realizará la suma de las tres señales anteriores, con la intención de obtener la señal ecualizada.

Para obtener en cada salida un filtrado distinto, mediante el uso de los botones se ha incluido en el programa un switch, el cual cambiará la señal de salida dependiendo de las veces que pulsemos el botón.

```
#include "EAA_DSP_Sampled_Header.h"
#include <stdio.h>
#include <math.h>
#include <filters.h>
#include <coeficientes.h>
float Senal2,Senal3,Senal4,Senal5;
float Senal;
float InputSample_L;
float InputSample_R;
float OutputSample_L1;
float OutputSample_R1;
float OutputSample_R2;
float OutputSample_L2;

void    EAA_DSP_Process_Samples(void){

        Senal=(InputSample_R+InputSample_L);//Sumamos el calan Derecho e Izquierdo
                                            //de la señal de entrada

        Senal2= fir(Senal, coeffslow, statelow , LOWTAPS);//Realizamos un Filtrado
                                            //Pasa baja y lo guardamos en Senal2

        Senal3= fir(Senal, coeffsband, stateband , BANDTAPS);//Realizamos un Filtrado
                                            //Pasa Banda y lo guardamos en Senal3

        Senal4= fir(Senal, coeffshigh, statehigh , HIGHTAPS);//Realizamos un Filtrado
                                            //Pasa Alta y lo guardamos en Senal4

        Senal5= (Senal2+Senal3+Senal4);     //Sumamos las 3 señales filtradas
                                            //y las guardamos en Senal5
```

Figura 42 Declaracion de variables y filtrado de señales de la Función EAA_DSP_Process_Samples

```
////////////////////////////////////////////////////////////////////////
//            FILTRADO MEDIANTE EL PULSADO DE BOTONES                 //
////////////////////////////////////////////////////////////////////////
    //Dependiendo de las veces que pulsemos el boton "SW8"//
    //Pasará a un caso u otro del switch obteniendo a la  //
    //Salida L1 la señal con su filtrado correspondiente. //
    ////////////////////////////////////////////////////////

    if( Pulsado==1){ //Pulsado = 1, cuando es pulsado el boton "SW8"

                switch (Veces)  {
                    case 1:
                            OutputSample_L1= Senal2;
                            break;
                    case 2:
                            OutputSample_L1= Senal3;

                            break;
                    case 3:
                            OutputSample_L1= Senal4;
                            break;
                }//switch
         }//if ( Pulsado==1)

    ////////////////////////////////////////////////////////////
    //Dependiendo de las veces que pulsemos el boton "SW9"//
    //Pasará a un caso u otro del switch obteniendo a la  //
    //salida R1 la señal con su filtrado correspondiente. //
    ////////////////////////////////////////////////////////

    if(Pulsado2 == 2) {  //Pulsado = 2, cuando es pulsado el boton "SW9"

                switch (Veces2) {
                    case 1:
                            OutputSample_R1= Senal2;
                            break;
                    case 2:
                            OutputSample_R1= Senal3;
                            break;
                    case 3:
                            OutputSample_R1= Senal4;
                            break;
                    }//switch

          }//if  (Pulsado2 ==2)
```

Figura 43 Código Funcion EAA_DSP_Process_Samples Parte 1

```
    ////////////////////////////////////////////////////////
    //Dependiendo de las veces que pulsemos el boton "SW10"//
    //Pasará a un caso u otro del switch obteniendo a la   //
    //salida L2 la señal con su filtrado correspondiente.  //
    ////////////////////////////////////////////////////////

        if (Pulsado3 == 3) { //Pulsado = 3, cuando es pulsado el boton "SW10"

                switch (Veces3){
                    case 1:
                            OutputSample_L2= Senal2;
                            break;
                    case 2:
                            OutputSample_L2= Senal3;
                            break;
                    case 3:
                            OutputSample_L2= Senal4;
                            break;
                }//switch

        }//if(Pulsado == 3)

    ////////////////////////////////////////////////////////
    //Dependiendo de las veces que pulsemos el boton "SW11"//
    //Pasará a un caso u otro del switch obteniendo a la   //
    //salida R2 la señal sin procesar o la señal ecualizada//
    ////////////////////////////////////////////////////////
    if (Pulsado4==4){ //Pulsado = 4, cuando es pulsado el boton "SW11"

            switch (Veces4){

                    case 1:
                            OutputSample_R2= Senal;
                            break;
                    case 2:
                            OutputSample_R2= Senal5/2;
                            break;
            }//switch

        }//if(Pulsado==4);


}//EEA_SAMPLED
```

Figura 44 Código Funcion EAA_DSP_Process_Samples

5. DISEÑO DEL VUMETRO OPTICO.

En este apartado se explicaran tanto los elementos que han sido necesarios para la construcción del vúmetro, como las herramientas de trabajo utilizadas.

5.1. EL CIRUITO INTEGRADO LM 3916.

5.1.1. FUNCIONAMIENTO.

El LM3916 contiene en su interior diez comparadores de tensión los cuales tienen las entradas positivas conectadas a un divisor de tensión interno. Este divisor esta conectado a una tensión de referencia que viene dada por el integrado y que depende de la tensión de alimentación y del valor de la resistencia R2 (como se puede observar en la figura 47), la cual debemos calcular previamente mediante la formula:

$$Vref = 1{,}25V\left(1+\frac{R2}{R1}\right)+R2\cdot 80\mu A$$

Figura 45 Ecuacion3

Como en nuestro caso vamos a trabajar con diferentes niveles de entrada y dado que el DSP a partir de valores de salida superiores a 1,6V, comienza a saturar la señal, esta resistencia se ha sustituido por un potenciómetro con el fin de calibrar el vúmetro según nuestras necesidades.

Como se muestra en el diagrama de bloques, la señal de audio accede a un buffer, el cual tiene su salida conectada a las entradas negativas de los comparadores, de esta forma la señal de entrada será comparada entre los valores de 0V y el valor de tensión de referencia que hallamos fijado, con lo cual el nivel de entrada de la señal de audio habilitará a una cantidad de comparadores proporcional a su amplitud, obteniéndose así niveles de tensión bajos en las salidas de los mismos con el consecuente encendido de los leds conectados a estos.

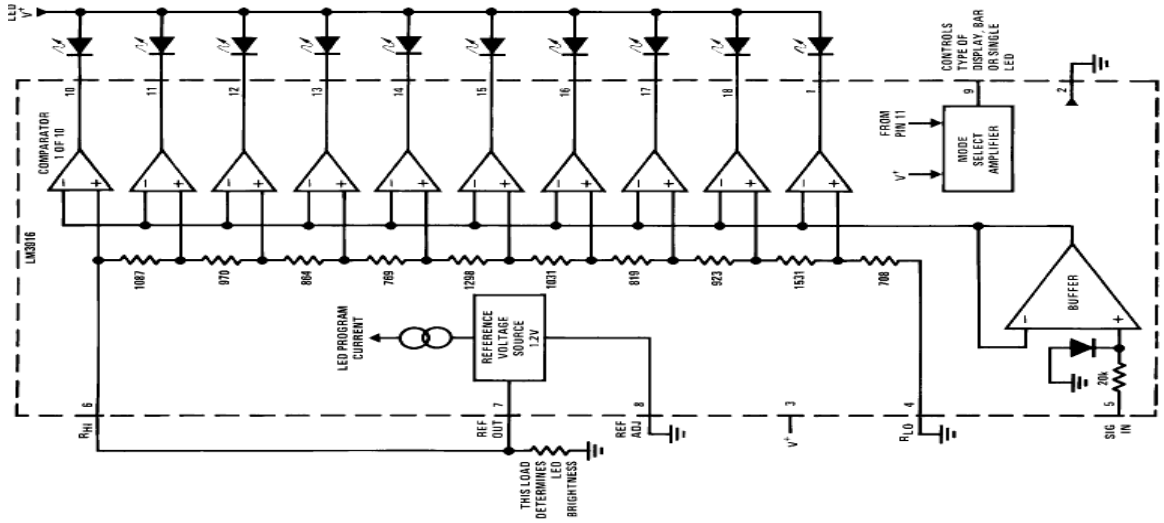


Figura 46 Diagrama de Bloques del Circuito Integrado LM3916

Este integrado posee limitación de corriente interna en cada uno de los comparadores, por lo que es innecesario el empleo de resistencias en forma externa en serie con los Leds.

La intensidad de los leds también depende de la tensión de referencia y de la resistencia R1 y se calcula mediante la fórmula:

$$Iled = \frac{12{,}5V}{R1} + \frac{Vref}{2{,}2k\Omega}$$

Figura 46 Ecuación 4

Pero como en nuestro caso también vamos a variar el brillo de los leds se ha decidido sustituirla por un potenciómetro.

5.2. DISEÑO DE LA PLACA PCB

Para el diseño de la placa, en primer lugar se realizará el circuito esquemático mediante la herramienta Orcad Capture.

El circuito a diseñar será el que viene definido por el fabricante en el datasheet al cual se le han hecho una serie de modificaciones que se explican continuación.

Con el fin de acondicionar la señal de entrada y seguir mas de cerca el valor de pico de la señal, a la configuración del circuito se le ha agregado un diodo 1N4148, un condensador de 1µF y una resistencia de 560kΩ que van conectados al pin numero cinco del integrado(entrada de audio).

Dado que el nivel de salida de la señal del DSP no supera los 1,6 Voltios las resistencias R1 y R2, se han sustituido por dos potenciómetros de 10kΩ con el fin de poder ajustar la sensibilidad y el brillo del vumetro según nuestra necesidad. El potenciómetro P1 es el encargado de determinar el brillo de los led, y el potenciómetro P2 se encarga de ajustar la sensibilidad del vúmetro

También se ha incluido un interruptor en el pin 9 con el fin de cambiar el modo de visualización entre modo punto (solo se enciende un led) y modo barra se encienden todos los leds.

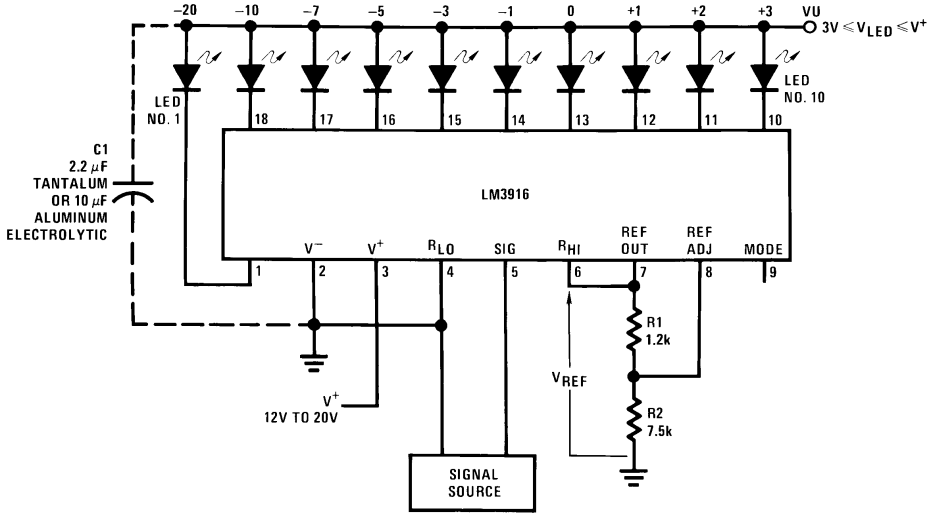


Figura 47 Esquema sin modificar

5.2.1. ESQUEMA CON ORCAD CAPTURE.

A continuación se muestra el esquema diseñado en Orcad Capture y ya modificado de la placa.

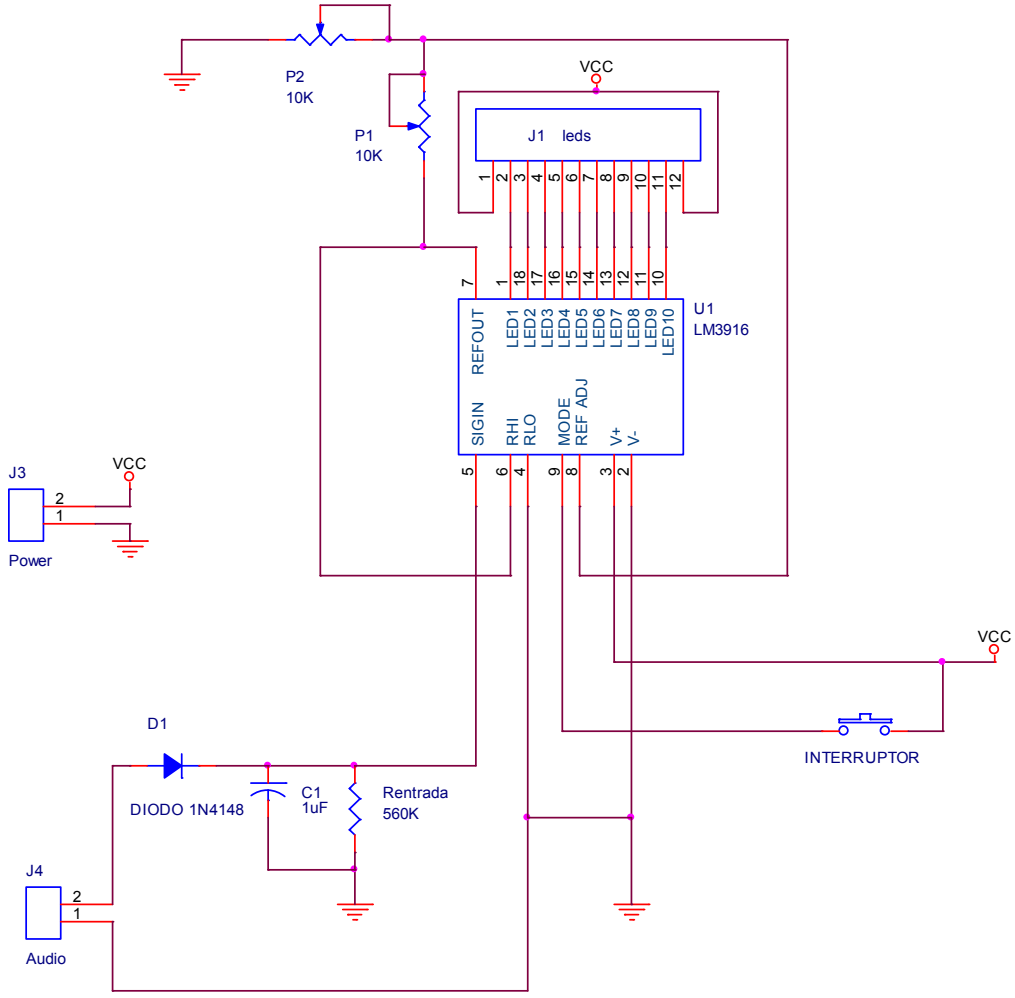


Figura 48 Esquema en Orcad Capture

5.2.2. TRAZADO DE PISTAS MEDIANTE LAYOUT

Una vez diseñado el circuito con Orcad Capture y se procederá a realizar el diseño de la placa PCB.

Para ello se seleccionará de las librerías predefinidas por Orcad los footprints de los componentes que podamos, y aquellos que no existan, deberán ser diseñados, en este caso se han diseñado los footprints del condensador, el diodo, los dos potenciómetros, el interruptor y el conector de leds.

Una vez los footprints estén diseñados, se procederá a la conexión de todos los componentes, la disposición podemos elegirla como mejor se adecue a nuestras necesidades, en este caso en particular se ha decidido elegir la siguiente disposición de componentes.

Como se trata de un diseño bastante sencillo se ha decidido incluirlo todo en una misma capa, en este caso la capa BOTTOM.

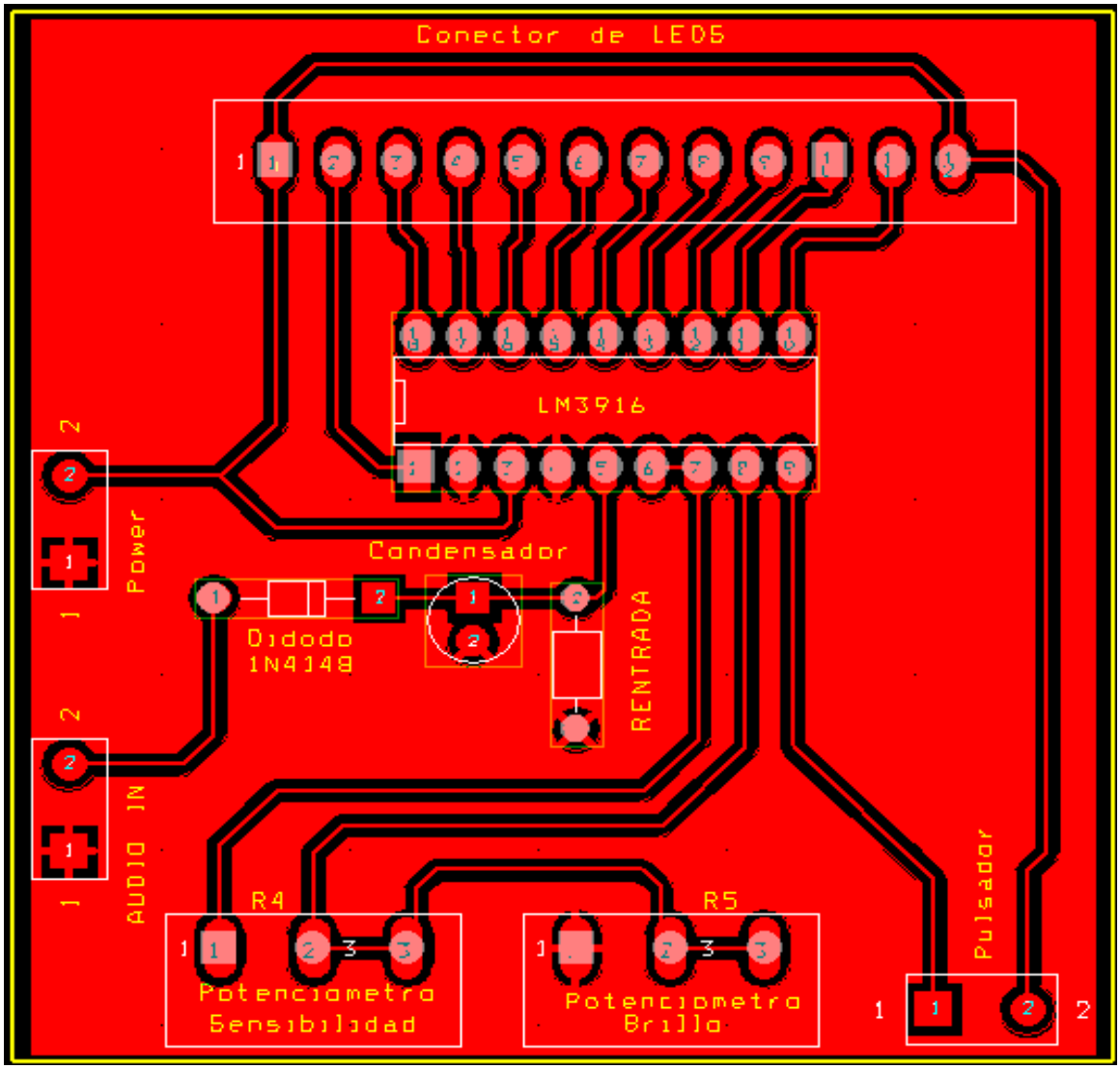


Figura 49 capa BOTTOM PCB

5.3. MONTAJE.

Una vez que tenemos la placa PCB impresa, procederemos a soldar los componentes.

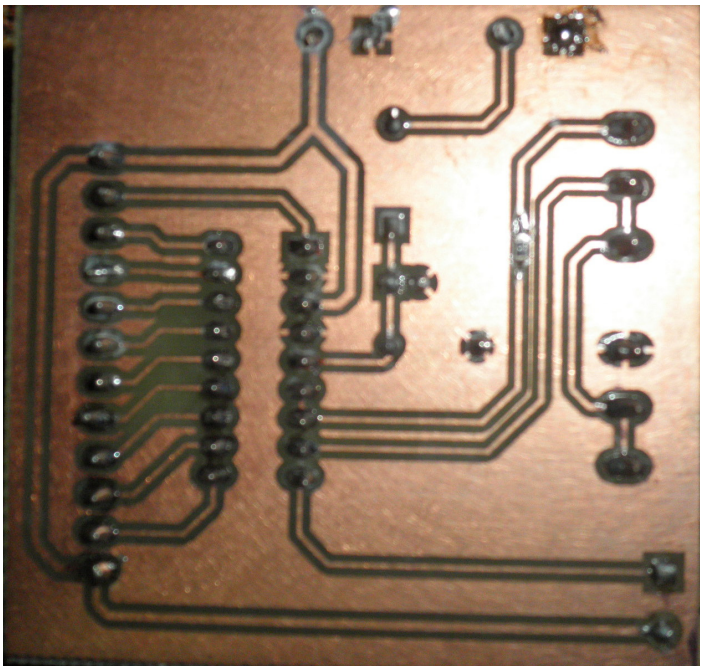


Figura 50 Placa PCB impresa.

Para la fuente de alimentación utilizaremos dos conectores de banana hembra, para la entrada de audio usaremos un conector RCA hembra, los conectores de leds elegidos, son unos conectores de tornillo de esta forma se podrá conectar y desconectar cómodamente cualquier cable.

Una vez definidos los conectores se procederá a soldarlos a nuestra placa. Una vez soldados este es el resultado.

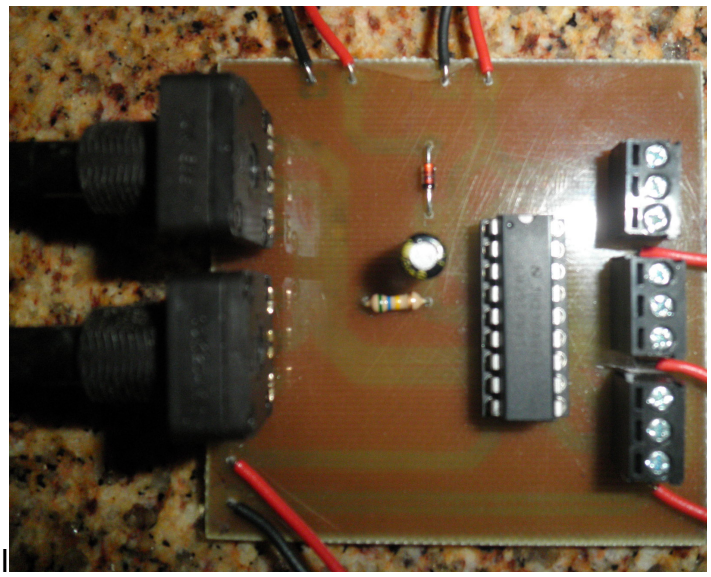


Figura 51 Placa PCB con los componentes soldados

Los leds irán incrustado en 10 placas de metacrilato de 5mm de grosor, y de ahí serán conectados mediante cables hacia el conector de nuestra PCB

Figura 52 Leds incrustados en el metacrilato y conectores banana-hembra y conector RCA-hembra

5.3.1. FUNCIONAMIENTO

Para calibrar la sensibilidad del vúmetro, se deberá introducir la señal al máximo nivel de potencia posible, y mediante el potenciómetro P2 se ira ajustando hasta que, la ultima fila de leds quede encendida, una vez hallamos conseguido esto, tendremos el vúmetro calibrado para la señal de entrada

Para comprobar el correcto funcionamiento del vúmetro, realizaremos una serie de pruebas con diferentes tonos y sonidos para comprobar que todo funciona correctamente, comenzaremos introduciendo un tono cualquiera y modificando su amplitud hasta sobrepasar los 1,6 voltios, de esta forma si todo funciona correctamente podemos comprobar como al sobrepasar dicho limite se encenderán los últimos leds de nuestro vúmetro dándonos a entender que la señal está saturando.

Comprobaremos también el modo punto y el modo barra, si todo esta bien se deberán encender todos los leds (modo barra) o un único led (modo punto).

Figuras 53 y 54
A la izquierda Vúmetro con una señal de entrada de 1,7V. A la derecha Vúmetro con una señal de entrada de 400mV

6. CONCLUSIONES

En la realización de este proyecto de fin de carrera se pretendía poner a prueba los conocimientos adquiridos durante la carrera. Dado que este proyecto consta de una parte electrónica, el diseño del vúmetro, y de una parte digital como es la programación de los filtros FIR mediante el DSP, el uso de herramientas de trabajo utilizadas durante la carrera ha sido de gran ayuda.

También se han adquirido nuevos conocimientos sobre las herramientas de trabajo ya conocidas, como en el caso de Matlab y su herramienta para el diseño de filtros, FDA TOOL, y el caso de Orcad Layout para diseño de placas PCB

El uso del DSP y del entorno de trabajo Visual DSP 5.0++ han permitido mejorar los conocimientos sobre el lenguaje de programación aplicado al audio.

Como conclusión propia puedo añadir que ha de tenerse muy claro la selección de todos los componentes utilizados, desde las características de los leds que se van a elegir, hasta el funcionamiento básico del circuito integrado, de esta forma nos ahorraremos bastantes contratiempos y dificultades que se puedan producir.

Para finalizar este Trabajo Final de Carrera y ha modo de conclusión se puede decir que como se ha podido comprobar el funcionamiento final del conjunto (Vúmetro y DSP) ha sido totalmente satisfactorio, por lo que se afirma que se han conseguido todos los objetivos fijados al inicio de este Trabajo Final de Carrera, consiguiendo así la implementación y construcción física del Vúmetro óptico de leds.

7. BIBLIOGRAFIA

LIBROS

- JOYANES AGUILAR, L. (2006). *Programación en c++, algoritmos, estructuras de datos y objetos*. Madrid: McGraw-Hill.
- CEBALLOS SIERRA, F. (2009). *Enciclopedia del lenguaje c++.* Paracuellos del Jarama, Madrid: Ra-ma.
- B.JACKSON, L. (1996). *Digital filters and signal processing.* Boston: Kluwer Academic.
- A Lee, E. et al (1997). *DSP processor fundamentals: Architectures and features*. New York : IEEE
- Analog Devices. (2012). *C/C++ Library Manual for SHARC Processors.* Nordwood, Massachusetts: Analog Devices, Inc.
- Analog Devices. (2009). *Run-Time Library Manual for SHARC Processors.* Nordwood, Massachusetts: Analog Devices, Inc.
- Analog Devices. (2007). *Visual dsp++ 5.0 Getting Started Guide.* Nordwood, Massachusetts: Analog Devices, Inc.
- Ledger D., Tomarakos J. (1998). *Using The Low-Cost, High Performance ADSP-21065L Digital Signal Processor For Digital Audio Applications*. Nordwood, Massachusetts: Analog Devices, Inc.

WEBS CONSULTADAS

- WIKIPEDIA. Procesador Digital de la Señal. <http://es.wikipedia.org/wiki/Procesador_digital_de_señal> [Consulta: 13 de mayo de 2014]
- WIKIPEDIA. Filtro FIR <http://es.wikipedia.org/wiki/Finite_impulse_response> [Consulta: 16 de mayo de 2014]
- UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA. Diseño de filtros Digitales. <http://www2.dis.ulpgc.es/~obolivar/apuntes/tema5/tema5.htm#DFFIR> [Consulta: 23 de mayo de 2014]