

**DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA**

**FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN**

Informe de Practica Final

“Sonómetro, procesamiento digital”

**Asignatura:** Complementos de Electrónica Analógica II

**Ingeniería Electrónica**

***Autor:***

*Pignataro, Juan Emanuel – Registro 24.854*

**Año 2019**

# Introducción

El sonómetro es un instrumento de medida que sirve para medir diversos descriptores asociados al sonido entre los que se encuentran los niveles de [presión sonora](https://es.wikipedia.org/wiki/Presi%C3%B3n_sonora). En concreto, el sonómetro mide el nivel de [ruido](https://es.wikipedia.org/wiki/Ruido) que existe en determinado lugar y en un momento dado. El instrumento consta de las siguientes partes fundamentales:

1. Micrófono: Convierte las variaciones de presión de las ondas sonoras en una señal eléctrica.  
  
2. Amplificador: Amplifica la señal recibida lo suficiente para permitir la medida de los niveles bajos.  
  
3. Ponderación en frecuencia: Es un conjunto de filtros eléctricos cuya respuesta intentó simular la respuesta auditiva humana. Compensa la diferencia de sensibilidad del oído humano para tonos puros según las distintas frecuencias audibles mediante las ponderaciones A, B y C. Si bien en la actualidad es sabido que los sonidos tonales aislados no representan la audición de sonidos más complejos, la ponderación A y la ponderación C se utilizan por la correlación con los efectos del ruido en el ser humano que han mostrado en investigaciones acumuladas sobre hipoacusia causada por ruido, molestias causadas por ruido entre otros efectos.

4. Ponderación temporal: Luego de la ponderación en frecuencia se realiza un promedio móvil obteniéndose una señal promediada que se actualiza en tiempos cortos. Llevan una constante de tiempo de 125ms en el caso de respuesta rápida o 1s en el caso de respuesta lenta.

5. Indicador: Muestra la señal de salida una vez atravesadas las etapas de procesado.



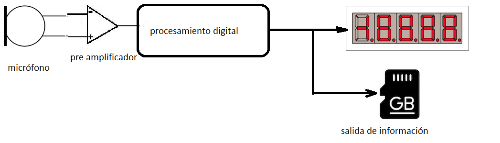


Figura : fotografía y esquema de un sonómetro

En la figura 1 vemos un esquema general de las partes que componen a grandes rasgos un sonómetro.

# Objetivo

En este informe nos centraremos en el estudio y realización de la parte digital de procesamiento, la cual debe poseer ciertas características de precisión.

El objetivo fundamental será lograr la realización del procesamiento digital de un sonómetro partiendo de una entrada de línea de sonido, proveniente de un preamplificador, para finalizar en un elemento de visualización (display) donde se podrá observar el resultado de la medición en tiempo real.

Siempre se buscará cumplir con las normas de la Comisión Electrotécnica Internacional (I.E.C. por sus siglas en inglés) la cual nos da una serie de pautas de normalización para cumplir, y así lograr una estandarización y confiabilidad en nuestro producto. Dichas especificaciones se encuentran en la norma IEC 61672

# Especificaciones

Según la norma IEC 61672, el algoritmo que debe implementarse es el que se describe en la figura 2 en forma gráfica.

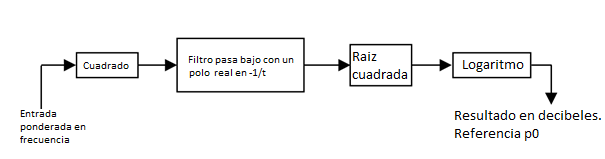


Figura algoritmo de procesamiento de un sonómetro representado en forma visual

## Normas I.E.C. utilizadas.

La Comisión Electrotécnica Internacional es una organización de [normalización](https://es.wikipedia.org/wiki/Normalizaci%C3%B3n) en los campos: [eléctrico](https://es.wikipedia.org/wiki/Electricidad), [electrónico](https://es.wikipedia.org/wiki/Electr%C3%B3nica) y tecnologías relacionadas. Esta comisión se encarga de la preparación y publicación de normas internacionales para todas las áreas mencionadas.

Todas las normas internacionales de la I.E.C. son totalmente basadas en el consenso y representan las necesidades de las partes interesadas. Cada país miembro, no importa cuán grande o pequeño, tiene un voto y una voz en lo que sucede en una Norma Internacional I.E.C.

La normativa a utilizar es la 61672 la cual se ha publicado en tres partes.

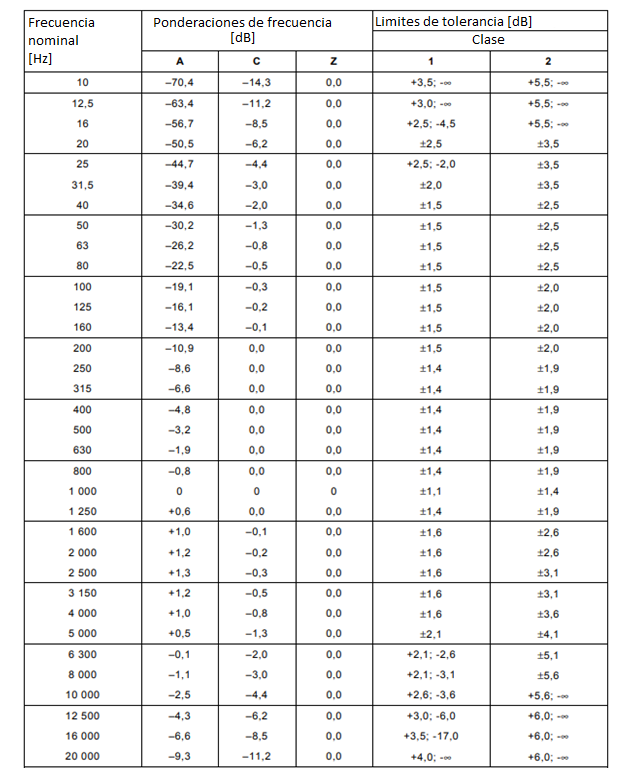
Parte 1: Detalla las especificaciones de rendimiento que un sonómetro de clase 1 o 2 debe tener.

Parte 2: Se usa en laboratorios de pruebas de calidad, como Applus en España o el PTB en Alemania, para testear los instrumentos y asegurar que cumplen con las descripciones del fabricante. Esto se conoce como certificado de examen de modelo.

Parte 3: Detalla las pruebas que pueden llevarse a cabo durante una verificación periódica o rutinaria.

De la primera parte de la norma se reproducen las tolerancias en la siguiente tabla:

Tabla Ponderación de frecuencias y límites de tolerancia, incluidos los valores máximos de la incertidumbre de medición.



Con los datos de la tabla 1 se realizó un gráfico para lograr visualizar los márgenes en los cuales debemos entrar para cumplir con la norma I.E.C. ya sea para una clase 1 o para lograr una clase 2 (ver figura 3).

Se trabajó con los valores de la norma al 10% para poder tener un error en el resto de los componentes del sonómetro, ya que se está realizando el estudio solamente de la parte digital y no del sonómetro completo y como se mencionó antes la norma hace referencia a valores referidos al sonómetro completo con todas sus partes.

Tolerancia para clase 2

Tolerancia para clase 1

Figura Limites de tolerancia de la norma para distintas clases al 10%.

## Curvas de ponderación de frecuencia

La mayoría de los criterios acústicos vinculados con la percepción o los efectos del ruido sobre el ser humano están expresados utilizando alguna forma de ponderación frecuencial. El propósito es tener en cuenta que la sensibilidad auditiva cambia con la frecuencia.

Existen varias ponderaciones para sonómetros, las cuales vamos a usar dependiendo de la aplicación que estemos realizando.

Ponderación ‘A’: La ponderación ‘A’ es la ponderación más habitual, no es muy sensible a frecuencias bajas y altas, pero sí lo es entre 500 Hz y 6 kHz.

Ponderación ‘C’: La ponderación ‘C’ se emplea para estudiar a las frecuencias bajas, y eventualmente estableciendo con ella diferencias respecto a la ponderación ‘A’.

Dichas ponderaciones se grafican en la figura 4 de acuerdo a la tabla 1.

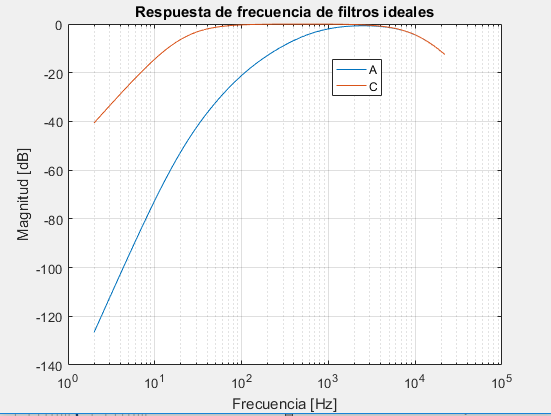


Figura : curvas de ponderación de frecuencia A y C teóricas

# Desarrollo

Para el desarrollo del algoritmo se comenzó por su implementación en Matlab fuera de línea.

## Desarrollo en entorno de Matlab

### Filtros de ponderación de tiempo

Lo primero que se desarrolló fueron los filtros de ponderación de tiempo, a partir de la especificación de la norma IEC 61672 que establece que deben ser filtros de orden 1 con constantes de tiempo 1 segundo y 0.125 segundos respectivamente para Slow y Fast de la siguiente forma

tau=0.125;

num = [0, 1];

den = [tau, 1];

[numF,denF] = bilinear(num,den,Fs); % Analog to digital conversion

tau=1;

num = [0, 1];

den = [tau, 1];

[numS,denS] = bilinear(num,den,Fs); % Analog to digital conversion

La función *bilinear* de Matlab toma la función previamente definida en el dominio continuo de Laplace y la transforma al dominio discreto de la forma IIR.

Se obtuvieron los siguientes resultados:

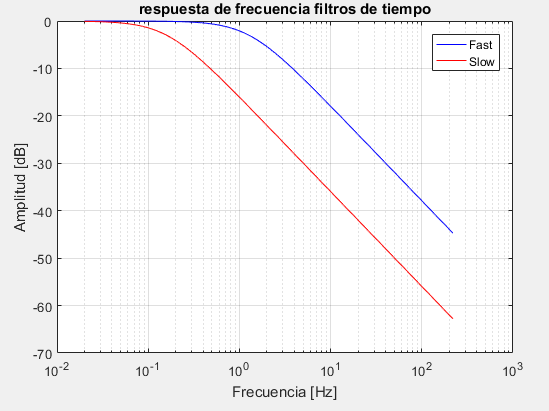


Figura respuesta de frecuencia de filtros de tiempo

Recordando que

para

Para

Por lo cual se comprueba que están bien ejecutados

### Filtros de ponderación de frecuencia

De forma similar se obtuvieron los filtros de ponderación de frecuencia.

Para ponderación A:

Ceros:

* z=0 (cuádruple)

Polos:

* P1=20.60 Hz (doble)
* P2=107.7 Hz
* P3=737.9 Hz
* P4=12194 Hz (doble)

k = 1.2588\*12200^2\*(2\*pi)^2; %ganancia de estado estacionario para A

z = [0; 0; 0; 0];

p = [-20.60\*2\*pi -20.60\*2\*pi ... %polos del filtro A

-107.7\*2\*pi -737.9\*2\*pi ...

-12194\*2\*pi -12194\*2\*pi];

p = p.';

[num,den] = zp2tf(z,p,k); % Convert to transfer function form

[numA,denA] = bilinear(num,den,Fs); % Analog to digital conversion

Para ponderación C:

Ceros:

* z=0 (doble)

Polos:

* P1=20.60 Hz (doble)
* P2=12194 Hz (doble)

k = 12194^2\*(2\*pi)^2; %ganancia de estado estacionario para C

z = [0; 0];

p = [-20.60\*2\*pi -20.60\*2\*pi... %polos del filtro C

-12194\*2\*pi -12194\*2\*pi];

p = p.';

[num,den] = zp2tf(z,p,k); % Convert to transfer function form

[numC,denC] = bilinear(num,den,Fs); % Analog to digital conversion

En la figura 6 observaremos los resultados obtenidos de la simulación. Vemos que la respuesta obtenida para los filtros digitales es fiel a la teórica exigida por la norma, excepto en las frecuencias cercanas a la de Nyquist de muestreo (48000Hz) donde la función *bilinear* de Matlab fuerza a atenuar lo más posible la respuesta para evitar problemas de aliasing. Esta opción no se puede desactivar, debido a que si se pudiese desactivar aparecería el problema del aliasing, es decir que aparecerían frecuencias espurias que contaminarían muestras adyacentes.

Habiendo hecho un pequeño estudio de las opciones comerciales que pueden encontrarse, se destaca que ninguna es sensible más allá de los 8KHz de frecuencia, estando en un rango de precio de hasta $124.000. Por lo tanto, lo anterior no es un problema grave.

Los modelos comerciales que se consultaron fueron:

* Tenmars St107
* Extech 407730
* Amprobe Sm 10
* Holdpeak Hp-882a
* Cem Dt-8852
* Unit T Ut351

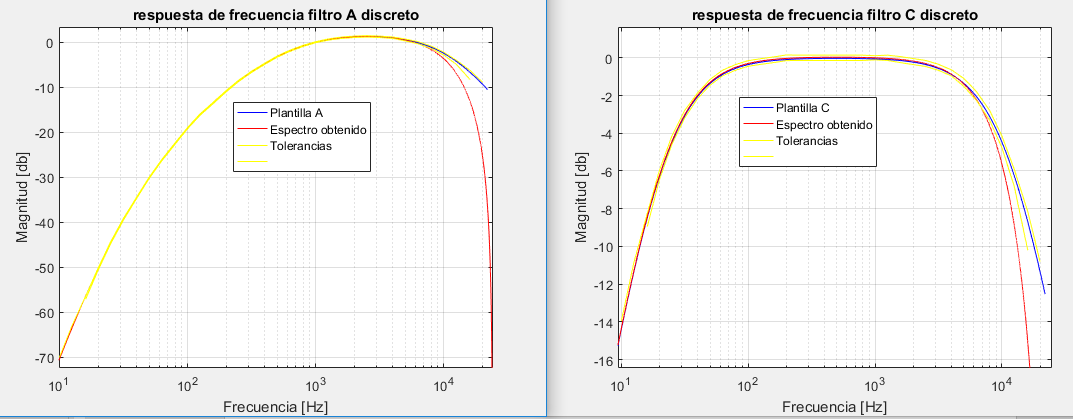


Figura Respuesta de frecuencia de filtros A y C reales digitalizados contra plantillas ideales.

### El algoritmo completo se adjunta en apéndice 7.1.

### Pruebas de implementación del algoritmo completo

Se implementó el algoritmo completo y se hicieron diversas pruebas. Las primeras en el dominio del tiempo para observar las dinámicas lentas del experimento.

Primero observamos en la figura 7 un experimento que se tomó con la función *wgn* [6.13] de 200.000 muestras. Dicha función genera muestras en forma gaussiana con espectro de frecuencias plano.

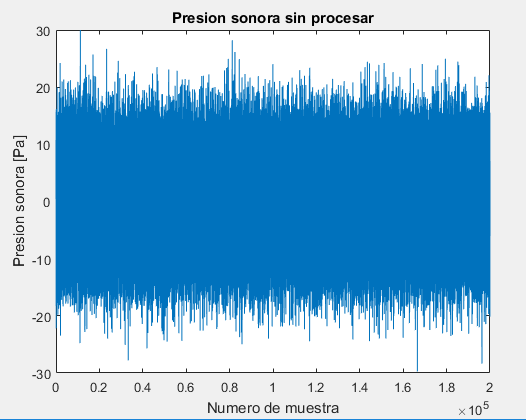


Figura Presión sonora sin procesar

Y se obtuvieron los resultados de las figuras 8 y 9. En estas imágenes es fácil sacar dos conclusiones.

1. Para una potencia constante de ruido, la medición es constante una vez superado el transitorio, lo que muestra que está bien implementado el algoritmo.
2. De las curvas sin logaritmar podemos ver claramente la diferencia entre los filtrados Fast y Slow, donde Fast tiene un transitorio más rápido, y es más reactivo a la dinámica de la señal censada.

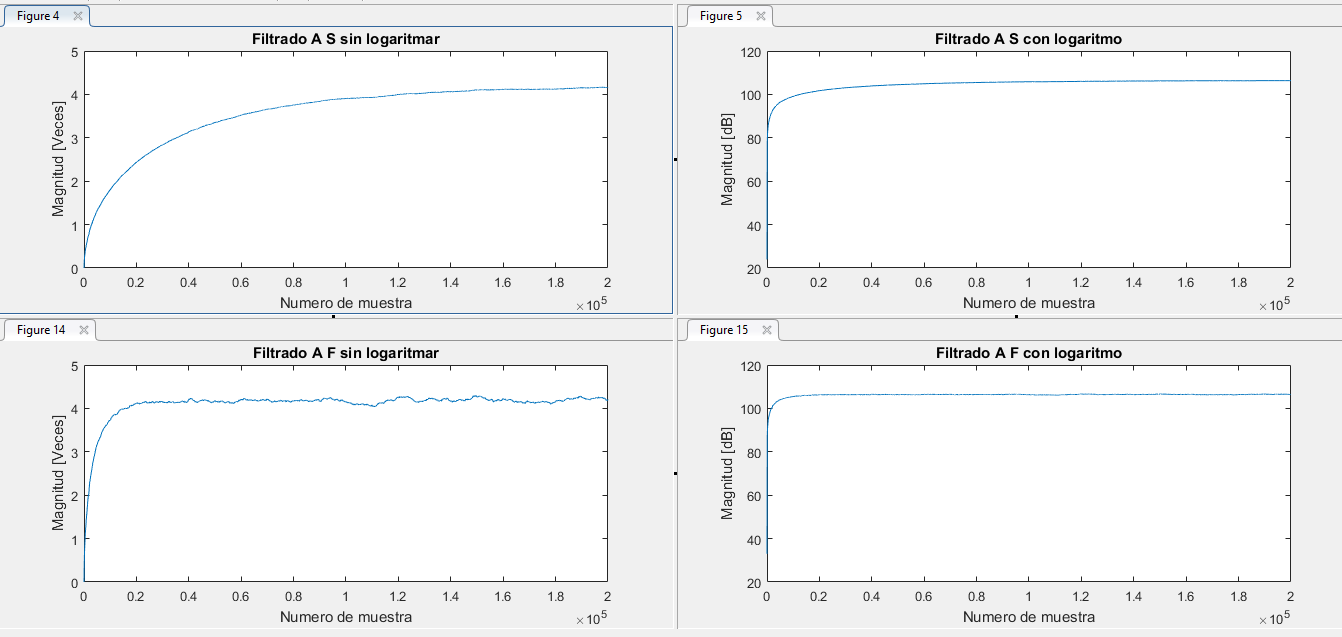


Figura Resultados del algoritmo de Matlab con filtrado A

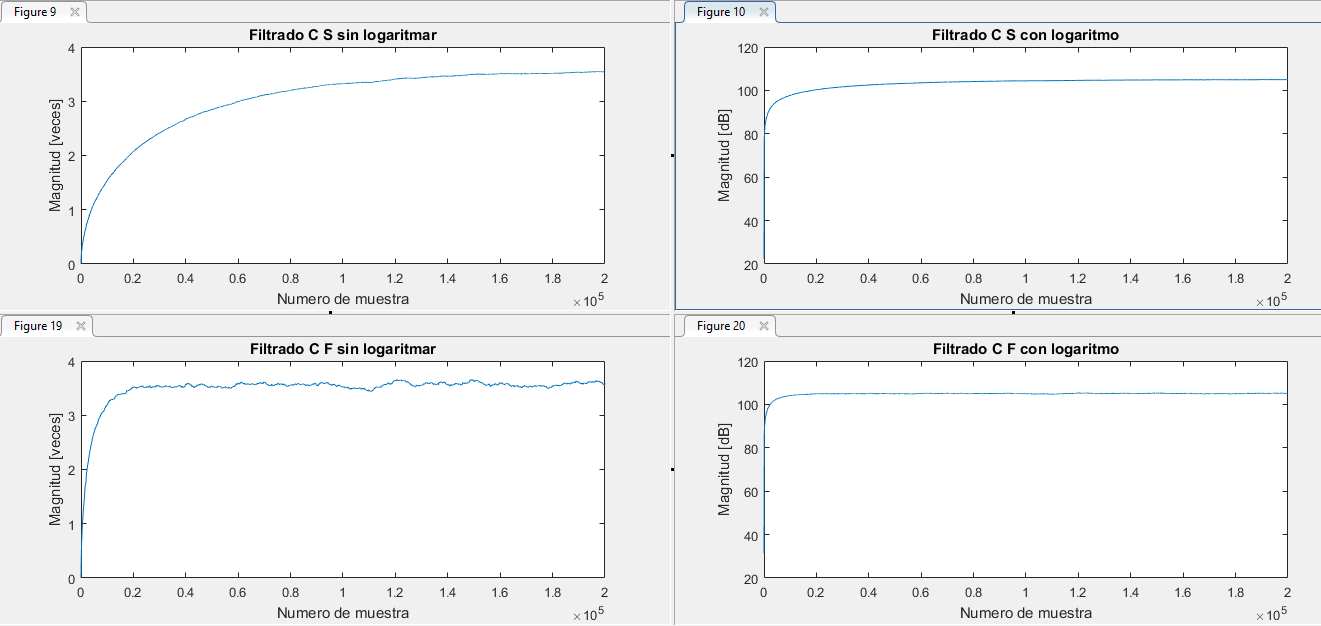


Figura Resultados del algoritmo de Matlab con filtrado C

Luego de la correcta implementación en Matlab se procedió a implementarlo en el dispositivo FPGA.

## Desarrollo en FPGA

### Introducción a FPGA

Para lograr que este desarrollo funcione en tiempo real en forma de un dispositivo real, se utilizó la placa de desarrollo DE2-115 de ALTERA. Esto posibilitó la integración de dicho proyecto con la materia “Temas Específicos de Electrónica Digital I”.

la placa utilizada cuenta como núcleo con un FPGA Cyclone IV de la compañía ALTERA, y se ilustra y esquematiza en las figuras 10 y 11 respectivamente.

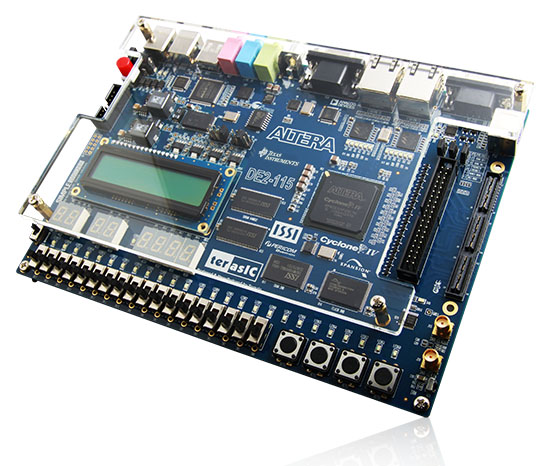


Figura placa de desarrollo DE2-115 de ALTERA

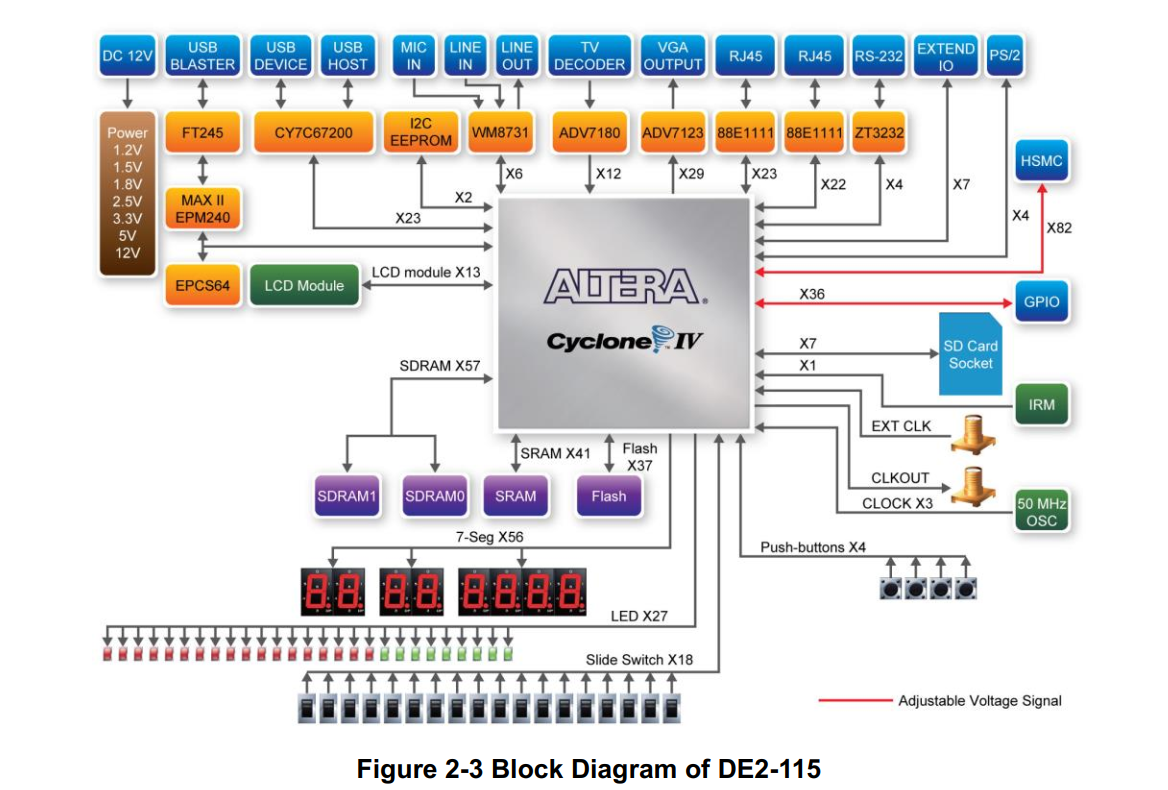


Figura diagrama de bloques de la placa DE2-115 de ALTERA

Un FPGA (del inglés field-programmable gate array), es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento, mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

Los FPGA se utilizan en aplicaciones similares a los *Circuitos Integrados de Aplicación Específica (ASIC)* sin embargo son más lentos, tienen un mayor consumo de energía y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, los FPGA tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

En el FPGA no se realiza programación tal cual como se realiza en otros dispositivos como DSP, CPLD o microcontroladores. El FPGA tiene celdas que se configuran con una función específica ya sea como memoria (FLIP-FLOP tipo D), como multiplexor o con una función lógica tipo AND, OR, XOR. La labor del programador es describir el hardware que tendrá el FPGA. Por consiguiente, la tarea del programador es definir la función lógica que realizará cada uno de los CLB (bloques de lógica configurable), seleccionar el modo de trabajo de cada IOB (bloque de entrada-salida) e interconectarlos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL o lenguajes de descripción de hardware. Los HDL más utilizados son:

* VHDL
* Verilog
* ABEL

En nuestro caso se utiliza VHDL dentro del entorno de programación de la propia marca ALTERA, llamado Quartus, cuya interfaz se muestra en la figura 12.

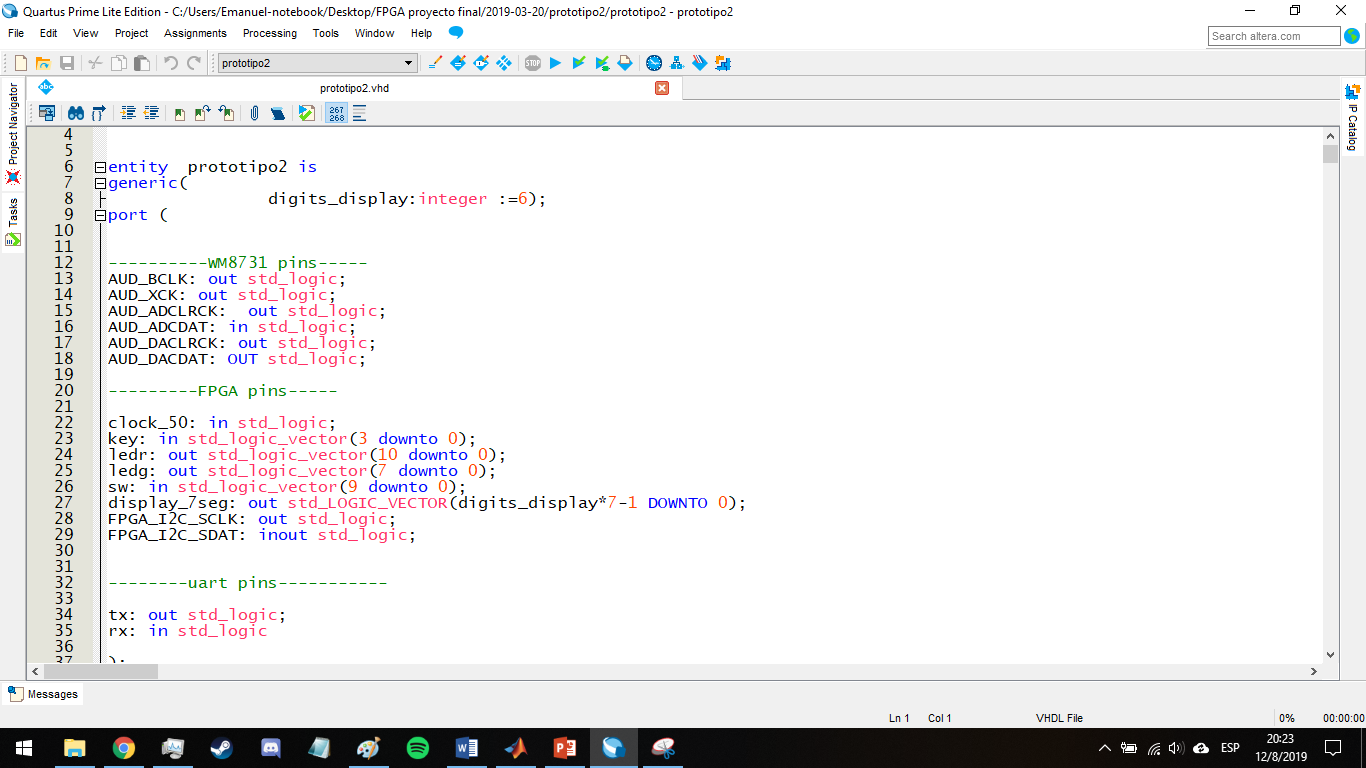


Figura interfaz de programación dentro del entorno Quartus

### Bloques utilizados

Para nuestra realización se utilizaron los siguientes módulos de la placa de desarrollo.

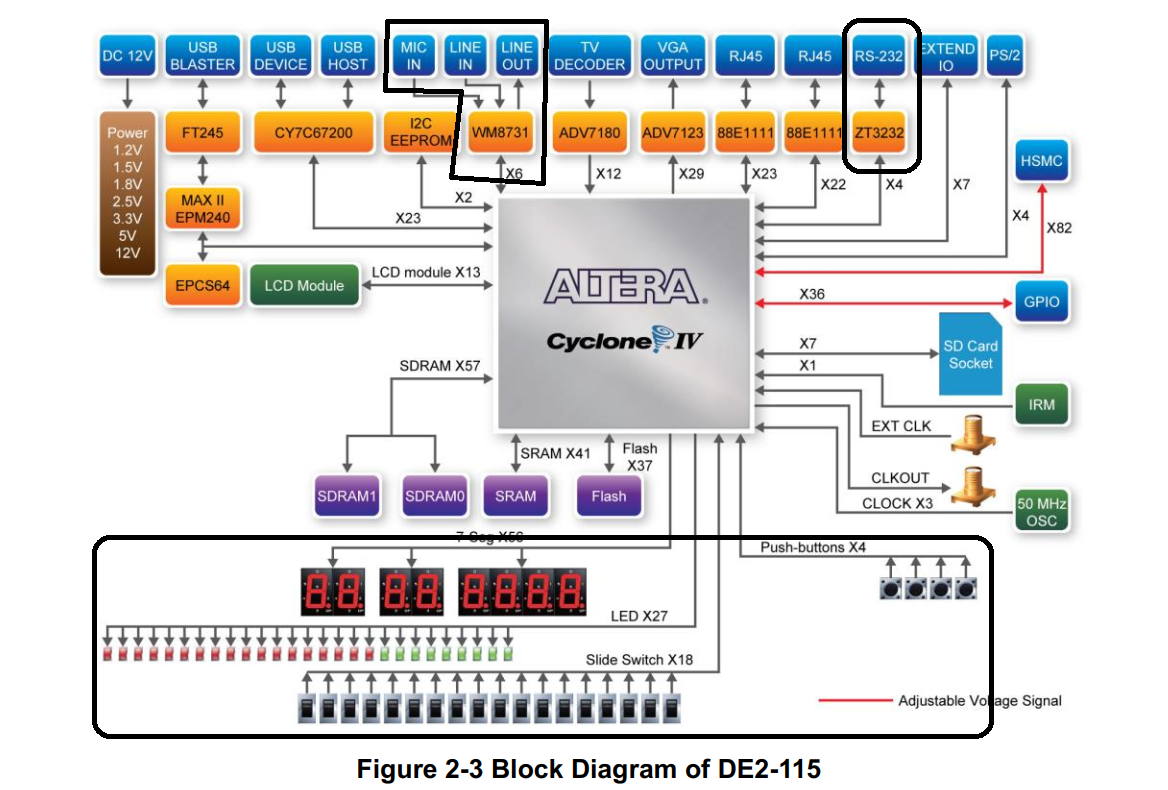


Figura bloques utilizados

* Placa de sonido
* Puerto serie
* Display 7 segmentos
* Pulsadores
* Llaves
* Ledes

#### Audio códec

El códec de audio que se utilizó es el WM8731 que está integrado en la placa de desarrollo.

Se configuró para que capture la señal de línea de entrada a 48KHz y 16bits de palabra.

Las características del códec se encuentran en el apéndice 7.3.

#### RS-232

Para la comunicación entre el FPGA y la PC se utilizó el puerto serie debido a la sencillez de configuración a pesar de su baja velocidad de transferencia. En nuestro caso se utilizó a 115200 baudios y en modo unidireccional desde el FPGA hacia la PC, ya que lo único que resultó de interés fue tener toda la información procesada por el FPGA a fin de comprobar en otro entorno como Matlab su correcta ejecución.

Algunas características se muestran en el apéndice 7.4.

### Esquema general

En la figura 14 se describe la configuración de hardware interna del FPGA en bloques.

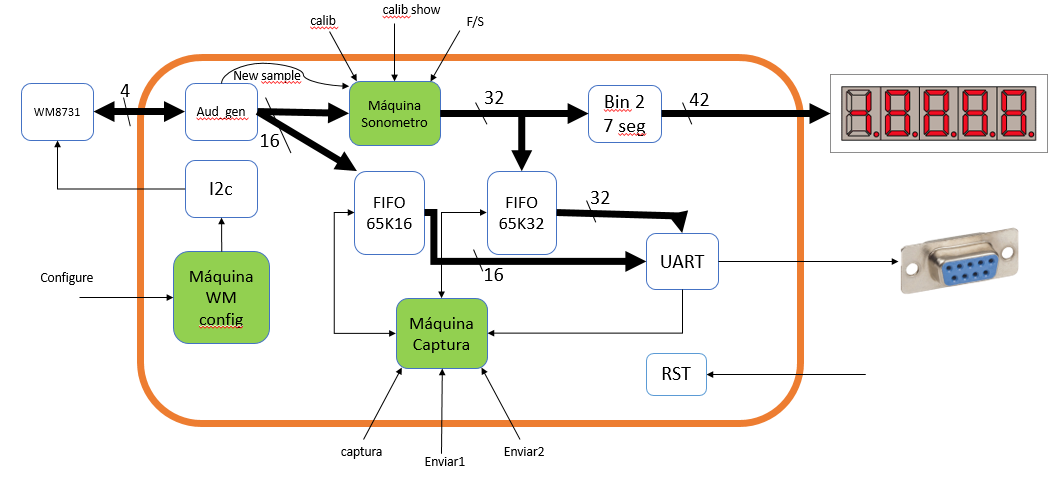


Figura diagrama de bloques de los componentes dentro del FPGA

Cabe destacar que el algoritmo de procesamiento del sonido está dentro del bloque *Máquina Sonómetro*.

En este punto se describe el método de desarrollo del prototipo. Si bien en el display se hará una representación de la medición con un tiempo de retenimiento para visualización del usuario, existe un mecanismo paralelo compuesto por dos memorias FIFO que tiene la función de transmitir por el puerto serie a la PC dos arreglos de datos. Por un lado, la señal capturada directamente por el códec de audio y por otro el resultado final del procesamiento. Esto permite fácilmente cargar dichos datos en el entorno de Matlab y comprobar si el algoritmo estuvo bien implementado.

Para ello se aplica el algoritmo en Matlab a los datos crudos de la placa de sonido y luego se compara con el resultado del FPGA.

### Componentes heredados

Para la realización de todo el procesamiento del sonido se utilizaron diversos componentes de descripción de hardware que simplificaron la tarea, algunos de ellos provistos por el entorno de programación y otros de terceros encontrados en internet (fuentes en la descripción). Esto demuestra el buen nivel de escalabilidad e integración del lenguaje de descripción de hardware. Los bloques más importantes se describen a continuación.

#### PLL

El esquema de conexión se muestra en la figura 15. Este componente es generado por el entorno de programación y se utiliza para la generación de un reloj de 12Mhz necesario para la placa de sonido a partir de 50Mhz nativos.

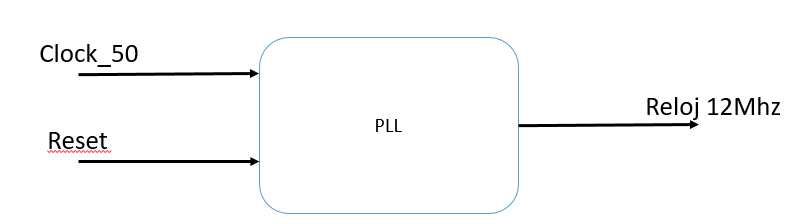


Figura PLL

#### Aud\_Gen

Este componente esquematizado en la figura 16 recibe la información en serie bit a bit de la placa de sonido y la convierte en palabras de 16 bits en complemento a 2.

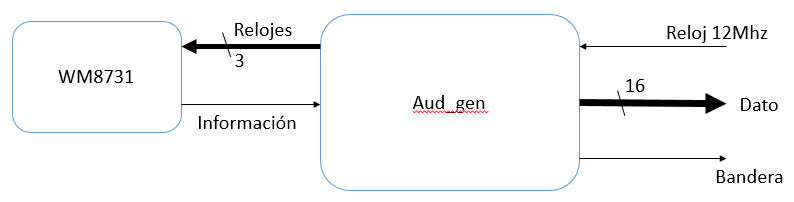


Figura Aud\_Gen

#### FIFO

Este componente se utiliza para almacenar temporalmente la información de los experimentos realizados. Se ilustra en la figura 17.



Figura FIFO 65k16

#### I2C

Este componente (Figura 18) se utiliza para la configuración de la placa de sonido. Toma datos de 16 bits y los envía en serie bit a bit.

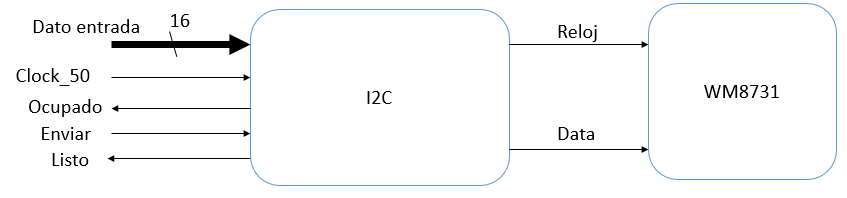


Figura I2C

#### WM config machine

En la Figura 19 se ilustra la máquina de estados que configura la placa de sonido de la siguiente forma:

La placa de sonido consta de un set de registros que, siendo escritos con diferentes valores, son cambiadas las configuraciones de funcionamiento.

Para esto se genera una máquina de estados que cuando es disparada envía un dato predefinido a cada registro para configurarlo.

La configuración que se usó se muestra en el apéndice 7.5.

El enlace a la hoja de datos del integrado se encuentra en las referencias.

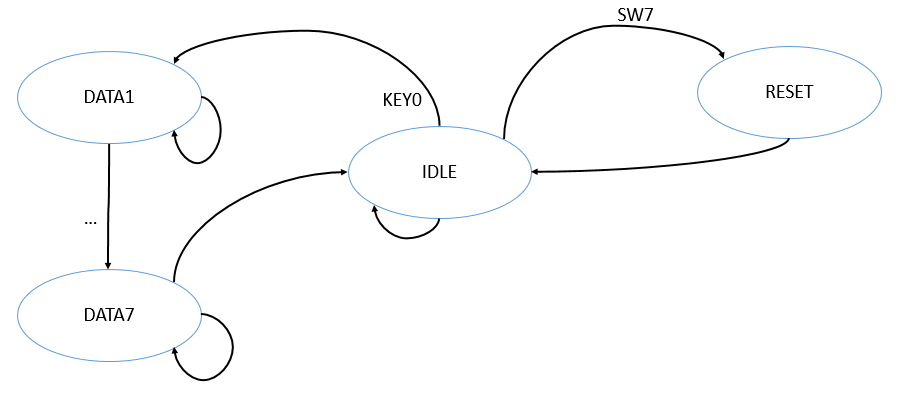


Figura esquema de la máquina de estados WM config machine

#### Floating Point

Este componente (Figura ) es provisto por el entorno de programación y tiene todas las funciones de punto flotante que se especifican en la Tabla .

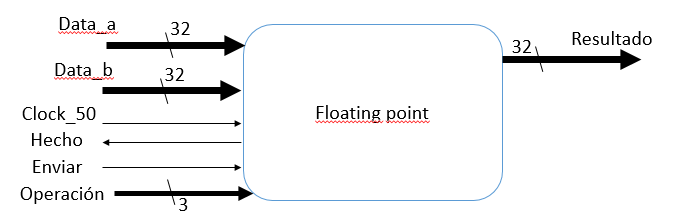
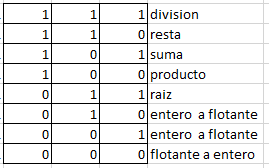


Figura 20 Floating Point

Tabla código de cada operación



Para mayor rendimiento de la aplicación y mayor sencillez de programación se instanciaron tantos de estos componentes como operaciones fueron requeridas, dejando el código de operación y los puertos de conexión conectados fijos.

### Máquina Sonómetro

En la Figura podemos observar la máquina de estados que hace el procesamiento propiamente dicho del sonido.

Es un proceso a grandes rasgos secuencial y realiza los siguientes pasos.

* Recibe los datos en complemento a 2 de la placa de sonido y los convierte a punto flotante para ser procesados.
* Es filtrado por los perfiles A o C de ponderación de frecuencia.
* A continuación, se eleva la señal al cuadrado, con lo que queda positiva. Y después se filtra con los perfiles F o S de ponderación de tiempo.
* Luego se suman 12.000 de estas muestras y se promedian.
* Se calcula la raíz cuadrada y el logaritmo para llevar el resultado a dB.
* Se multiplica x 86.8588 para hacer una corrección de unidades (Figura 22).
* Finalmente se ajusta con un valor de calibración y se muestra el resultado.

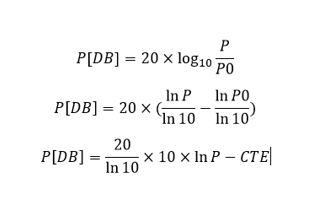


Figura 21 simplificación del cálculo en dB

Con esto se explica porque se multiplica por 86.8588 el resultado del logaritmo.

Este proceso tiene algunas pequeñas variaciones respecto al algoritmo de Matlab que detallaremos a continuación.

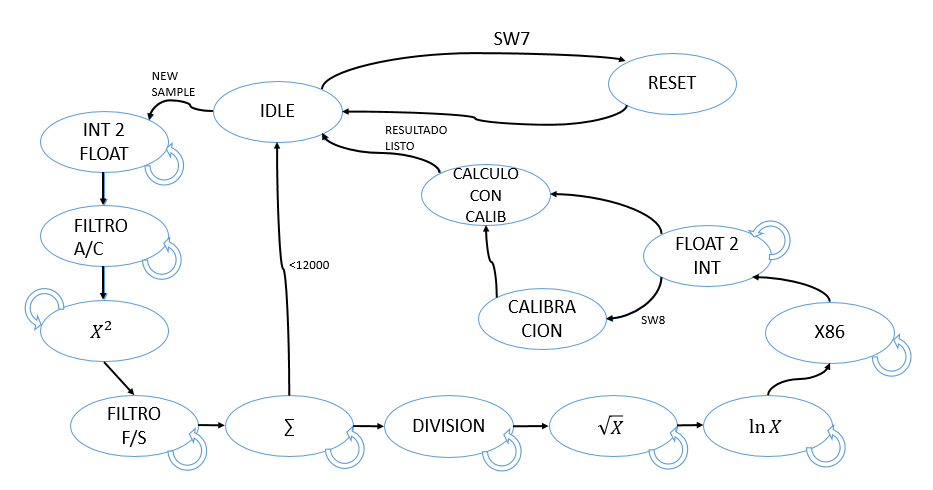


Figura 22 máquina de estados Máquina Sonómetro

Debido a que el usuario puede visualizar una limitada cantidad de información, es decir que no puede visualizar el resultado 48.000 veces por segundo, en lugar de mostrar el resultado instantáneo, se realizó una retención de 12.000 muestras, que produce 4 cambios de valor por segundo.

Para realizar el promedio de esas 12.000 muestras se tomaron los datos antes de las operaciones de raíz cuadrada y logaritmo, ya que no son operadores lineales y el resultado no sería correcto.

El código completo de descripción de hardware se adjunta en apéndice 7.2.

### Calibración

El único parámetro variable que debe tener el sonómetro es la amplitud de la señal de entrada, debido a que se está midiendo potencia, y para ello es necesario que la amplitud de la señal digitalizada sea la correspondiente con la realidad.

Para ello en un sistema analógico se utilizaría un amplificador de entrada, y mediante la ganancia ajustar la señal. Dicha ganancia representa una multiplicación de la señal con un valor constante. Una primera opción sería generar un amplificador en la entrada del proceso. Pero gracias a las propiedades de la operación logaritmo, la multiplicación por un escalar de la señal original se traduce en la suma de otro escalar una vez realizado el logaritmo.

Por lo tanto, se optó por la última opción de simplemente sumar un valor al resultado del proceso. Este valor se encuentra de forma automática de la siguiente forma.

Una vez que se ha escogido el micrófono con el pre amplificador y se han conectado a nuestro prototipo, se inyecta una señal de calibración conocida de 94db a 1KHz de presión sonora constante, y se coloca el sonómetro en modo calibración. El dispositivo a partir de aquí en cada medición calculará la diferencia entre la medición actual y los 94db de presión sonora que deberían medirse. Al desactivar el modo calibración esa diferencia queda almacenada en el dispositivo y es la constante que se adiciona en cada medida para encontrar el valor real de presión sonora.

## Comparación entre los resultados del FPGA y Matlab

Como se describió anteriormente, el proceso de comprobación de funcionamiento se basa en capturar la información cruda que proviene del FPGA, procesarla con Matlab y compararla con el resultado del procesamiento en FPGA.

Para ello se realizaron diversos experimentos en repetidas ocasiones, y se obtuvieron los siguientes resultados.

En primer lugar, se muestran los datos crudos obtenidos por la placa de sonido en la figura 23.

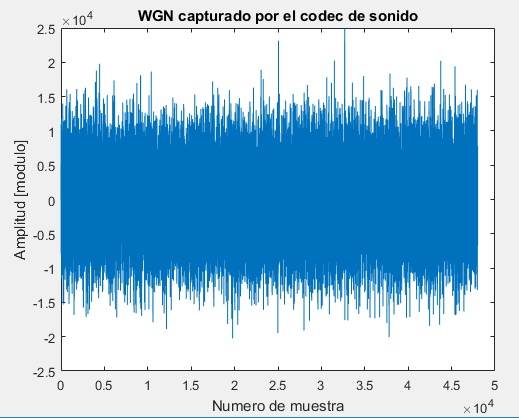


Figura 23 WGN capturado por el códec de sonido

Luego se filtraron con el filtro A en Matlab y se compararon con los enviados por el FPGA ya filtrados. El resultado se muestra en las figuras 24, 25 y 26.

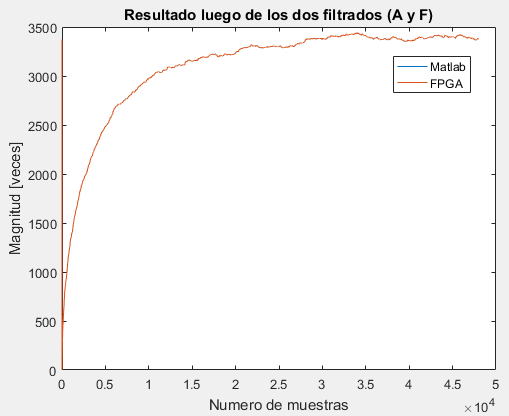


Figura 24 Resultado luego de filtrado A y F.

Si bien a simple vista parece observarse que los resultados son idénticos, cuando se hace un acercamiento a la gráfica se observa lo siguiente.

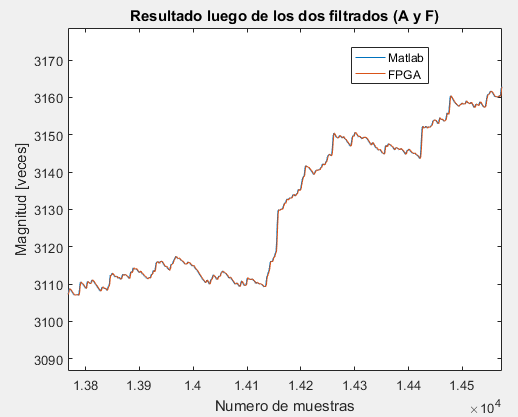


Figura 25 primer acercamiento a la gráfica de la figura 24

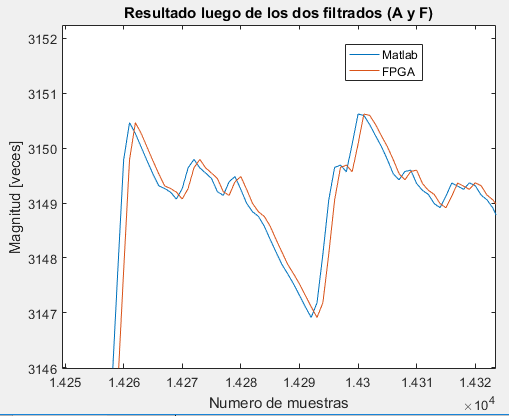


Figura 26 segundo acercamiento a la gráfica de la figura 24

El resultado del procesamiento es idéntico con la salvedad de que en el caso del FPGA está retrasado una muestra debido a que es necesario tener cada dato para calcular el resultado. Matlab encuentra el resultado en forma inmediata porque trabaja offline con toda la información ya capturada.

A continuación, se muestra en la figura 27 otro ejemplo, pero esta vez tomando el resultado final con la media por intervalos para la visualización del usuario.

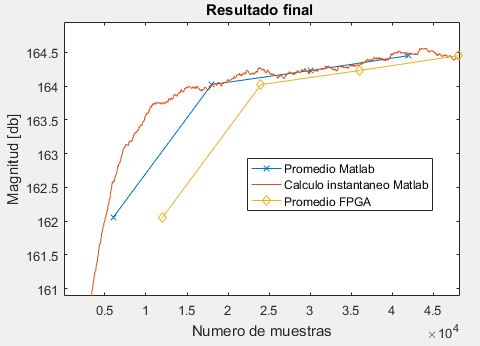


Figura 27 Comparación entre Matlab y FPGA con datos tomados por intervalos

Observamos en la figura 27:

* En rojo se muestra el cálculo de la presión sonora instantáneo hecho por Matlab a partir de los datos crudos capturados por la placa de sonido.
* En azul encontramos el cálculo hecho por Matlab, pero tomando una media móvil de 12.000 muestras.
* En amarillo está el mismo cálculo hecho por el FPGA.

Cabe destacar que al igual que antes, el FPGA necesita los 12.000 datos para poder hacer la cuenta y por eso se aprecia un retraso. A diferencia de esto, en Matlab se coloca el resultado de cada promedio de 12.000 muestras en el medio del intervalo, porque todos los datos ya se conocen y esto permite ver que la curva es la misma que la instantánea promediada.

Por último, se realizó el análisis frecuencial de la figura 24 a fin de analizar el comportamiento del filtro A implementado en el FPGA. El resultado se muestra en la figura 28, donde podemos observar que el comportamiento es satisfactorio, estando siempre dentro de las tolerancias definidas en los objetivos de este proyecto, que son el 10% de las que define la norma IEC61672.

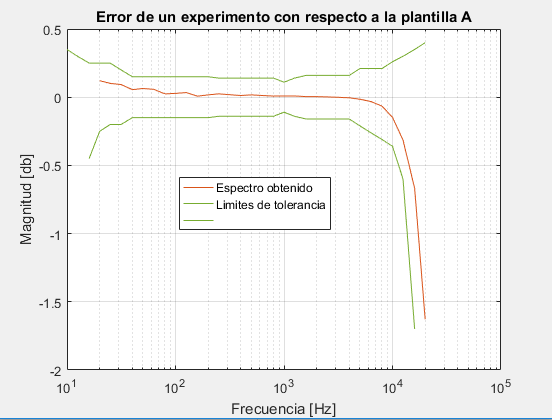


Figura 28 Error obtenido de un experimento con respecto a la plantilla A

En la figura 28 observamos que en altas frecuencias el espectro se aparta notoriamente del ideal debido a la atenuación mencionada anteriormente, generada por Matlab para evitar el aliasing, la cual no se puede evitar con el periodo de muestreo elegido.

Debido a que la respuesta se encuentra dentro de las tolerancias permitidas, comprobamos que el prototipo funciona correctamente.

De igual forma se evaluó el filtro de ponderación C. los resultados se muestran en la figura 29.

De igual forma que antes, los límites de tolerancia son el 10% de los especificados por la norma IEC61672, como se especifica al comienzo del informe.

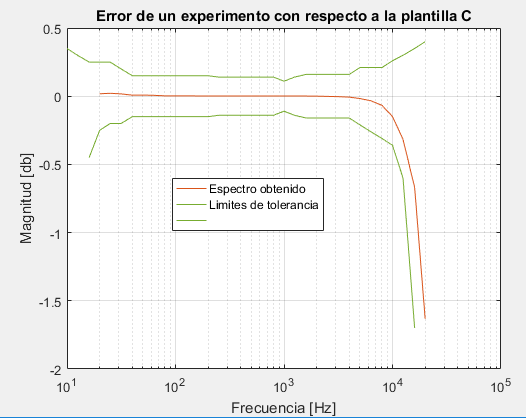


Figura 29 Error obtenido de un experimento con respecto a la plantilla C

# Conclusiones.

Fue posible poner en funcionamiento la parte digital del dispositivo cumpliendo con las exigencias de la norma, teniendo aun un 90% de margen en cuanto las ponderaciones de frecuencia. Este margen es utilizado por lo general por las imperfecciones de respuesta del micrófono. A pesar de aquello cabe destacar que los requerimientos de hardware necesarios son relativamente elevados.

Otra conclusión que se saca de esta experiencia es la alta complejidad del lenguaje VHDL. El cual es de mucho más bajo nivel que el C++ con que se programan microcontroladores convencionales. Por un lado, se deben tener en consideración los tiempos de propagación de las distintas señales dentro del dispositivo y, por otro lado, las simulaciones deben programarse en el mismo lenguaje VHDL, lo que hace al proceso de simulación del código, notablemente largo, quizás más que el de generación del mismo. Se enumeran propuestas de trabajos futuros en la siguiente sección.

Como contrapartida a lo anterior, este tipo de plataformas permite el desarrollo de sistemas altamente complejos, con relativa facilidad siempre y cuando se trabaje en forma ordenada. Esto es debido a la alta escalabilidad del código, que hace posible simular detalladamente cada pequeño componente por separado del resto.

Por último, se destaca que, si bien esta investigación se realizó en una placa de desarrollo, es decir una placa de propósito educativo, de gran volumen, peso y costo; los componentes que allí se utilizaron es posible adquirirlos en el mercado extranjero e importarlos. Esto hace que sea una gran oportunidad para fabricar un producto en el ámbito nacional sin tener competidores. Además, abre un universo de posibilidades de mejora, debido al alto potencial de rendimiento que tienen los FPGA.

# Propuestas de trabajos futuros

Fuera de la plataforma FPGA, se propone:

* El estudio de la misma temática implementada en dispositivos FPAA (Arreglos analógicos programables por campo).
* El estudio de la misma temática implementada en microcontroladores de alto rendimiento (por ejemplo, Raspberry PI).

Dentro de la plataforma FPGA, se propone:

* Realizar la comprobación de funcionamiento correcto de cada componente del hardware, bajo diversas situaciones anormales (test bench).
* Incrementar la frecuencia de muestreo, mejorando la respuesta a alta frecuencia
* Incrementar la resolución del conversor, permitiendo un rango dinámico de medición más amplio.
* Implementar la posibilidad de guardar los datos capturados en un dispositivo extraíble como una tarjeta SD.
* Mejorar la interfaz de visualización para el usuario. Por ejemplo:
  + Menús de configuración
  + Interfaz gráfica.
  + Interfaz a monitor.
  + Interfaz a aplicación de Android.
* Implementar el modo de funcionamiento “sonómetro integrador”.
* Implementar el análisis de contenido espectral por bandas de frecuencia.
* Implementar conectividad a Internet para guardado de datos (Wi-Fi, Ethernet, etc).

# Referencias.

**Conversores de datos binarios:**

1. Grondin, François (2009-2019). Float (IEEE754 Single precision 32-bit). Online Binary-Decimal Converter. <http://www.binaryconvert.com/convert_float.html?decimal=048046048049049050050053056055056052055054055051054> (revisado por última vez el 27 de agosto de 2019).
2. Exploring Binary (2008-2019). Decimal/Two’s Complement Converter. Exploring Binary. <https://www.exploringbinary.com/twos-complement-converter/> (revisado por última vez el 27 de agosto de 2019).

**Librería para placa de sonido:**

1. AntonZero (16 May 2017). WM8731-Audio-codec-on-DE10Standard-FPGA-board. GitHub. <https://github.com/AntonZero/WM8731-Audio-codec-on-DE10Standard-FPGA-board> (revisado por última vez el 27 de agosto de 2019).

**Hoja de datos de la placa de sonido:**

1. The Rockbox Crew (Sep 20 2017). Portable Internet Audio CODEC with Headphone Driver and Programmable Sample Rates. Rockbox. <https://www.rockbox.org/wiki/pub/Main/DataSheets/WM8731_8731L.pdf> (revisado por última vez el 27 de agosto de 2019).

**Librería para el manejo de puerto serie:**

1. Marquette University (2019). Uart test impl1. Marquette Embedded SystemS Laboratory. <http://www.dejazzer.com/eece4740/lectures/uart_test_impl1.zip> (revisado por última vez el 27 de agosto de 2019).

<http://www.dejazzer.com/eece4740/lectures/example_uart_echo.pdf> (revisado por última vez el 27 de agosto de 2019).

**Hoja de datos del componente para punto flotante:**

1. Intel Corporation (2019). Nios II Custom Instruction User Guide. Intel. <https://www.intel.com/content/www/us/en/programmable/documentation/cru1439932898327.html#shd1506201174145> (revisado por última vez el 27 de agosto de 2019).

**Componente para el cálculo del logaritmo:**

1. Altera Corporation. (2008). Floating Point Natural Logarithm (ALTFP\_LOG). Intel. <https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/ug/ug_altfp_log.pdf> (revisado por última vez el 27 de agosto de 2019).

**Conversión de binario a 7 segmentos:**

1. Atlassian. (Feb 13, 2019). 7-Segment Display Driver for Multiple Digits (VHDL). digikey. <https://www.digikey.com/eewiki/pages/viewpage.action?pageId=60031550> (revisado por última vez el 27 de agosto de 2019).

**Uso de coma flotante y transformada Z:**

1. Tabeni Couvert, Guillermo M. (Jul 2, 2007). Ejercicios sobre transformada Z. Scribd. <https://es.scribd.com/doc/55150928/Transformada-z-en-matlab-para-Series-y-Transformadas> (revisado por última vez el 27 de agosto de 2019).
2. Villanueva, Wladimiro Diaz (May 11, 1998). Representación de los números en punto flotante. Universitat de Valencia. <https://www.uv.es/~diaz/mn/node11.html> (revisado por última vez el 27 de agosto de 2019).
3. Borgwardt, Michael. Números de punto flotante. LA GUÍA DEL PUNTO FLOTANTE. <http://puntoflotante.org/formats/fp/> (revisado por última vez el 27 de agosto de 2019).
4. Matematicas Discretas (Nov 22,2017). Capítulo 3: PUNTO FLOTANTE. Medium. <https://medium.com/@matematicasdiscretaslibro/cap%C3%ADtulo-3-punto-flotante-c689043db98b> (revisado por última vez el 27 de agosto de 2019).

**Función generadora de ruido blanco en Matlab:**

1. The MathWorks, (1994-2019). Wgn. MathWorks. <https://la.mathworks.com/help/comm/ref/wgn.html> (revisado por última vez el 27 de agosto de 2019).
2. Miyara, F.M., 2013, Mediciones acústicas basadas en software, Rosario, Argentina, Asociación de Acústicos Argentinos

# Apéndices

## Código completo de implementación en Matlab

El código completo se encuentra en el siguiente enlace de GitHub

<https://github.com/picna1/Sonometro.git>

## Código de implementación en FPGA

El código completo se encuentra en el siguiente enlace de GitHub

<https://github.com/picna1/Sonometro.git>

## Codec de audio

En la Figura 30 Y en la Tabla Se especifican las conexiones del códec con el FPGA.

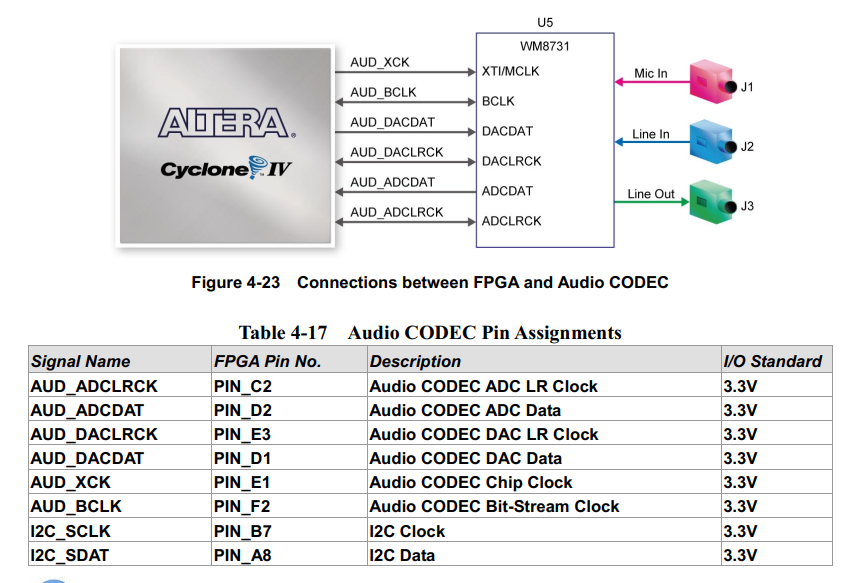
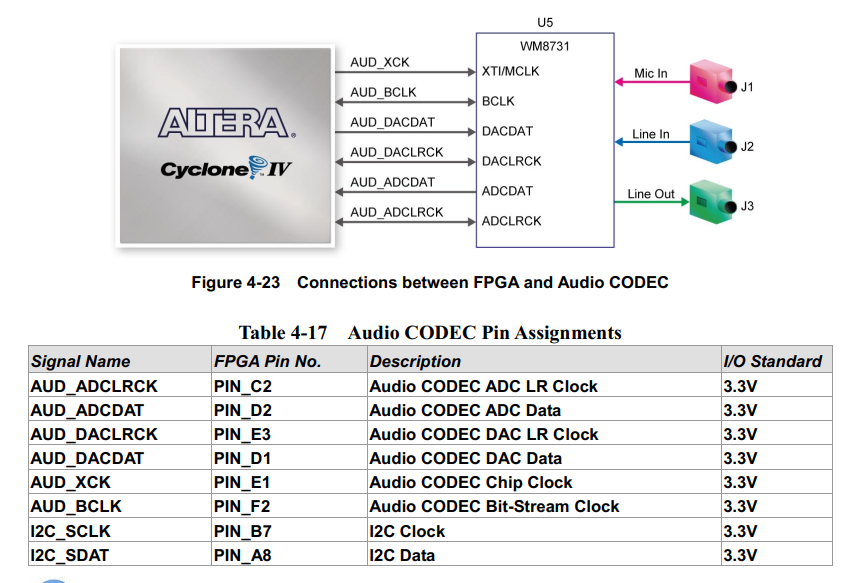


Figura 30 esquema de conexión del códec de audio con el FPGA

Tabla 4 pines de conexión entre el FPGA y el codec



Las características principales del códec son las siguientes:

* 2 líneas de programación por i2c.
* 4 líneas de comunicación de datos.
* Frecuencia de muestreo 48kHz configurable.
* Conversor, tamaño de palabra 16bits configurable.
* Modos de operación configurables.

## RS-232

En la Figura 31 y en la Tabla 5 se especifican las conexiones del chip de comunicación serie con el FPGA.

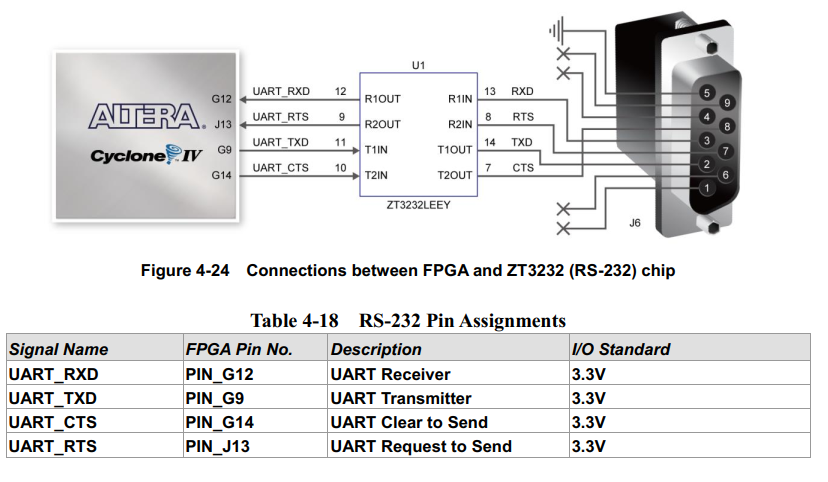
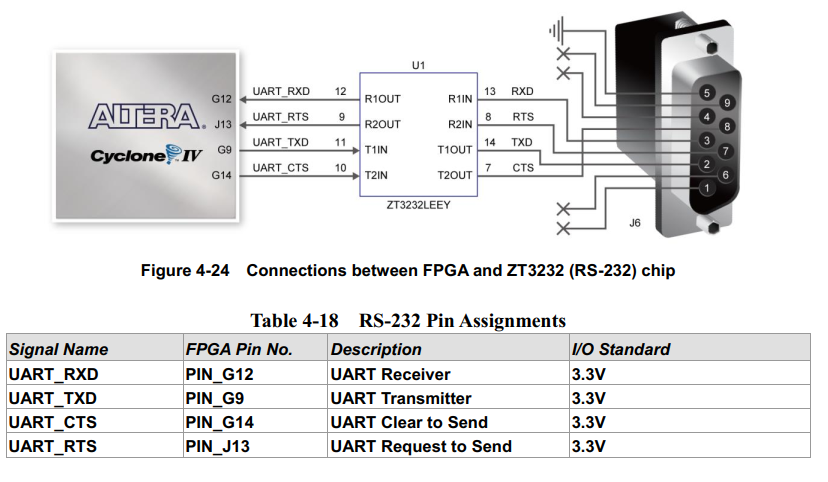


Figura 31 esquema de conexión entre el chip ZT3232 y el FPGA

Tabla 5 pines de conexión entre el FPGA y el chip ZT3232



## Configuración utilizada para la placa de sonido WM8731

R0: 100011111

R4: 000000010

R5: 000001000

R6: 000001010

R7: 000000011

R8: 000000001

R9: 111111111

r15:000000000

para mayor información sobre cada uno de los registros y configuraciones referirse a la hoja de datos del integrado (ver referencias).