UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

EQUALIZACIÓN DE MÚSICA EN TIEMPO REAL

Reporte Técnico de Investigación presentado por:

Alma Rivas Castillo 74046

Francisco Sánchez Mancillas 82162

Requisito para la obtención del título de

INGENIERO EN SISTEMAS COMPUTACIONALES

Profesor Responsable: *Dr. Humberto de Jesús Ochoa Domínguez*

Mayo del 2011

Autorización de Impresión

Los abajo firmantes, miembros del comité evaluador autorizamos la impresión del proyecto de titulación

EQUALIZACIÓN DE MÚSICA EN TIEMPO REAL

elaborado por los alumnos:

Alma Rivas Castillo 74046

Francisco Sánchez Mancillas 82162

Mtra. Ivonne Haydee Robledo Portillo	Dr. Humberto Ochoa Domínguez
Profesor de la Materia	Asesor Técnico

Declaración de Originalidad

Nosotros Alma Rivas Castillo y Francisco Sánchez Mancillas declaramos que el material contenido en esta publicación fue generado con la revisión de los documentos que se mencionan en la sección de referencias y que el programa de cómputo (*software*) desarrollado es original y no ha sido copiado de ninguna otra fuente, ni ha sido usado para obtener otro título o reconocimiento en otra Institución de Educación Superior.

Alma Rivas Castillo	Francisco Sánchez

Agradecimientos

A mi madre Alejandra,
mi hermano Federico,
mis padres Ricardo y José de Jesús
y a mi maestra de Proyecto de Titulación Ivonne.
Alma Rivas.

A mis padres Francisco e Isabel,
mi compañero y amigo Armando
y a mi maestra de Proyecto de Titulación Ivonne.
Francisco Sánchez.

Índice

Lista de Figuras

Capítulo 1. Introducción

El presente proyecto consiste en la ecualización de música utilizando *hardware* para tiempo real. El sistema agiliza el procesamiento de los algoritmos utilizados. Para poder elegir el mejor de éstos, se realizó una serie de comparaciones en tiempo y calidad de la voz entre dichos algoritmos. Las limitaciones a las que se enfrentaron los algoritmos fueron el espacio en memoria y el número de multiplicaciones y acumulaciones que se llevan a cabo.

La ecualización de música en tiempo real se utiliza en áreas tales como las comunicaciones, un área específica de utilización es el radio definido por *software* (SDR) donde la señal de voz debe procesarse antes de ser modulada. También, se puede encontrar la ecualización de música en tiempo real en la transmisión de señales de televisión donde el video se multiplexa con el audio previamente procesados para ser comprimidos y a su vez transmitidos.

En el medio musical, la ecualización de música es utilizada en la grabación de discos en los estudios o en los conciertos. Debido a que es difícil para un artista repetir una pieza musical múltiples veces y en el caso de un concierto es imposible poder repetir algo, por eso se utiliza la ecualización en tiempo real para suprimir y/o corregir cualquier problema de la grabación en el momento, como podría ser eliminar el ruido en el ambiente, corregir errores de dicción o desentonación del cantante. Así se protege la integridad de la voz del cantante al tener que esforzarse menos, también se disminuye el tiempo de producción del material discográfico y por ende se ahorra tiempo y dinero. En aplicaciones de Internet y servicios tales como: voz sobre IP, video llamadas, video conferencias y transmisiones de video.

Los algoritmos inicialmente se derivan de ecuaciones en diferencias. Estos se representan utilizando diagramas de flujo para visualizar las entradas y las salidas que se ven involucradas en el flujo de la señal.

La limitación principal es el tamaño del algoritmo ya que debe realizar la tarea en la menor cantidad de pasos posibles, de esta manera el tamaño de la aplicación será menor y podrá almacenarse dentro de la memoria de la tarjeta DSK 5510.

El proyecto tiene impacto en lo relacionado con las comunicaciones tales como: radio televisión, telefonía, etc. Se profundiza desde la perspectiva de la implementación, fue interés meramente académico aportar el análisis de los algoritmos existentes a un nivel de licenciatura. Y en el ámbito profesional adquirimos experiencia de un campo que es de nivel maestría o de diplomado, y conocimiento para desarrollar posteriormente en futuros estudios o en el ejercicio de nuestra profesión.

En el primer capítulo de este proyecto se podrán observar los antecedentes, en el cual se ven los usos del TMS320C55 en diferentes proyectos. En el segundo capítulo se muestra el marco teórico, explicando las partes de mayor relevancia. Pasando al capítulo 3 se presenta el diseño del ecualizador, donde viene el procedimiento y explicación sobre la elaboración de la aplicación y sus componentes. En el capítulo 4 se pueden observar los resultados aplicando el filtro FIR, junto con las gráficas y por último en el capítulo 5 se encuentran las discusiones y conclusiones.

1.1 Antecedentes

Texas Instruments, más conocida en la industria electrónica como TI, es una empresa norteamericana con sede en Dallas, Texas que desarrolla y comercializa semiconductores y tecnología para ordenadores. TI es el tercer mayor fabricante de semiconductores del mundo detrás de *Intel* y *Samsung* y es el mayor suministrador de circuitos integrados para teléfonos móviles. Igualmente, es el mayor productor de procesadores digitales de señal y semiconductores analógicos. Otras áreas de actividad incluyen circuitos integrados para módem de banda ancha, periféricos para ordenadores, dispositivos digitales de consumo y RFID (*Radio Frequency IDentification*).

El C55 (TMS320VC5510) puede usarse en aparatos personales portátiles, tales como: teléfonos celulares y estaciones base, reproductores de audio digital, cámaras digitales, libros electrónicos, el reconocimiento de voz, receptores GPS, huella digital y reconocimiento de patrones, módems inalámbricos y auriculares.

En 1993, M. Peresse, et al., realizaron un trabajo acerca de la habilitación de JPEG2000 en aparatos móviles inalámbricos 3G a través de la arquitectura OMAP ™ que describe cómo el formato JPEG2000 (códec de compresión de imágenes) en las terminales

inalámbricas 3G está habilitado gracias a la eficiencia del núcleo del TMS320C55xx o C55x incorporado en el TI OMAP ™. La arquitectura OMAP™ H / W se describe con un énfasis en cómo la compresión de imágenes y aplicaciones de descompresión se beneficiarán de esta arquitectura avanzada. El documento también describe las ventajas que ofrece OMAP ™ y su subsistema de arquitectura DSP para la transformada wavelet discreta (DWT), que es el núcleo JPEG2000 que consume una mayor parte de recursos de cómputo. Por último, se compara el tiempo de descarga de flujo de bits y los escenarios de codificación para imágenes QVGA comprimidas a color [1].

Posteriormente, en el año 2000, A. Wang, R. Hezar y W. Gass, del Instituto de Tecnología de Massachusetts (MIT), en Cambridge, MA., desarrolló una aplicación de baja potencia de un receptor de W-CDMA en una potencia ultra baja DSP, presentando los resultados de una implementación de enlace descendente y receptor de banda ancha-CDMA en el nuevo desarrollo de bajo consumo de *Texas Instruments* TMS320C55x para aplicaciones inalámbricas. El C55x incluye nuevas instrucciones y unidades funcionales, tales como la *Dual MAC* (*dual multiply-accumulate*), paralelismo implícito / explícito y una etapa de la fuente de información. También incluye el *hardware* en la unidad de dirección exclusivamente para el cálculo de direcciones de memoria. Esta nueva estructura es adecuada para la aplicación de los sistemas que tienen grandes tareas de cálculo y con requerimientos de baja potencia, tales como el procesamiento de banda base en el receptor de enlace descendente de W-CDMA. El chip disipa 0,25 mW / MIPS que es inferior a la potencia disipada en el poder de poca intensidad DSP, el C54x y en este trabajo se demuestra que el C55x tiene una mejora de 40 a 60% con respecto al C54x [2].

Como otro ejemplo, en el año 2002, se presenta en la Universidad Bonn de Alemania, un proyecto acerca de redes neuronales para el procesamiento de señales con el nombre de TIS3. El TIS3 consiste en un codificador táctil que escanea el objeto que desee en una corriente paralela de cursos de estimulación táctil, un estimulador táctil para provocar sensaciones táctiles espacio-temporales en la piel de una región seleccionada, y un módulo de aprendizaje para generar un vector cada vez mejor parámetro para la ET sobre la base de la entrada de características del sujeto humano. Como primer paso, la función TE se implementó como un conjunto de generadores 3 × 5, lo que podría ser empleado como

una función del tiempo por medio de un algoritmo basado en DSP (TMS320C55) sintonizables de señales espacio-temporal. Los resultados mostraron que el TIS3 en un circuito cerrado con los seres humanos, estos sujetos, podrían estar sintonizados para obtener las percepciones táctiles claramente distinguibles en menos de 80 ciclos de interacción [3].

Además, en junio del 2003, Byeong-Doo Choi de la Universidad de Corea, desarrolló una aplicación integrada en tiempo real de MPEG-4 de perfil simple utilizando el estándar AAC (*Advanced Audio Coding*), Utilizando chips DSP y RISC. Se optimizó los módulos de MPEG-4 con la estructura de doble almacenamiento en *buffer* de memoria, el doble MAC aplicado a la MC (*Motion Compensation*) de medio pixel, y un algoritmo de conversión en AAC de flotante a entero y el DSP TMS320C5510 [4].

Más aún, en diciembre del 2004, Jeong Hoo Lee, et al., de la Universidad de Ajou en Corea del Sur de la carrera de Ingeniería Eléctrica y Computación, realizaron el trabajo acerca de la arquitectura eficiente del DSP de decodificación de Viterbi con la latencia TB (*Trace Back*). Este trabajo propone especializadas instrucciones DSP y su arquitectura de *hardware* para el algoritmo de Viterbi. Cuando la longitud de restricción de K es 5, la arquitectura propuesta puede reducir los ciclos de decodificación de hasta un 17% en comparación con Carmel DSP y alrededor del 45% en comparación con TMS320C55x [5].

En otro caso, en octubre del 2006, Y. K. Jung de la Universidad A & M de Texas, realizó un trabajo de diseño y optimización de un decodificador de instrucciones programables para la arquitectura DSP, descubrió una técnica de reconfiguración de *hardware* y *software* para diseñar un decodificador de instrucciones programables para sistemas DSP que no emplean un *field-programmable gate-array*, que son circuitos integrados que son o pueden ser programados después de su manufactura. Esta técnica permite a los desarrolladores de *software* para reorientar de manera rápida y precisa de sus sistemas DSP. Con el fin de evaluar los procedimientos y desempeño de esta técnica de reconfiguración, se utilizó un RID (Decodificador de la Instrucción Reconfigurable) en este caso el DSP de TMS320C55 de *Texas Instruments*. Esto resultó en la optimización de dicho DSP [6].

Para el siguiente año, en abril del 2007, L. Girija y A.S. Spanias, realizaron un trabajo sobre el análisis de la arquitectura de la matriz de procesamiento (MxP, *Matrix Processing*). En este trabajo, describen y evalúan una nueva arquitectura DSP llamado el procesamiento de la matriz (MxP *trade*). Se centraron especialmente en la manipulación de la matriz y el tipo de vector incorporado en las normas de procesamiento de vídeo, por ejemplo, el filtrado, DCT, y la estimación de movimiento. Se presenta tablas de comparación de los resultados del MxP con otros DSP ampliamente utilizados como el TMS320c55xtrade TI, y también usando el TMS320c64xtradeTI [7].

En el mismo año del 2007 en septiembre, J.F. An de la Universidad Nacional Oceánica de Taiwán, realizó un trabajo acerca de la aplicación de simulador de canal MIMO para las aplicaciones de canales modelo SUI, el cual desarrolló un simulador de un transmisor Kroneckerelel. El simulador proporciona modelos reconfigurables para el canal adecuado para MIMO (múltiple entrada y múltiple salida), MISO (múltiple entrada y una salida) y SIMO (una entrada y múltiple salida) y un análisis de rendimiento de cada una de ellas. El simulador de canal dispone de frecuencia selectiva y la frecuencia de decoloración de componentes planos. En este simulador el conjunto de chips DSP TMS320C55 se utiliza como un subsistema esencial en la aplicación de tiempo real complejo de variables aleatorias gaussianas, hace los cálculos de la matriz Kronecker y las operaciones de descomposición de Cholesky [8].

El siguiente trabajo se desarrolló en febrero del 2008, G. Gammie, et al., de *Texas Instruments* en Dallas, TX., realizaron un proyecto acerca de 45 nm 3.5G de banda base y del procesador de aplicaciones multimedia, con adaptación del *body-Bias* y técnicas de ultra baja potencia *System on Chip* (SoC) de integración es el tema de la primera banda 3.5G integrada y un procesador de aplicaciones multimedia fabricado con un bajo consumo de energía plataforma de diseño digital y analógica y tecnología de proceso de 45 nm. Este SoC es compatible con los estándares móviles: HSUPA / HSDPA, WCDMA EDGE / GPRS / GSM y aplicaciones como MPEG-4 de video *streaming*, Java y audio en MP3. Multimedia de alto rendimiento, el motor de multiprocesador incluye un ARM1176 840MHz, un DSP TMS320C55x 480 MHz, y un procesador de imagen 240MHz [9].

Por último, en julio del 2010, T. Chattopadhyay, realizó un trabajo acerca de mejoras de rendimiento del codificador H.264 utilizando optimizaciones de plataformas específicas de bajo costo en plataformas DSP. Este autor presentó un documento para la realización de un codificador de línea de base QCIF H.264 en un DSP TMS320C55x, que tiene sólo 256 KB de RAM y 150 MHz de reloj. Se logró casi el 60% de reducción con respecto a MCPS codificador de referencia optimizado mediante la aplicación de estas optimizaciones específicas de la plataforma [10].

Con los anteriores antecedentes, se dieron a conocer los tipos de investigaciones en los que hacen uso del TMS320C55. Con esto se demuestra los diferentes usos y su gran importancia en el mejoramiento de la ecualización de música en tiempo real.

1.2 Descripción del Problema

Actualmente, en la industria de la región los puestos de diseño digital los ocupan personal extranjero y muy pocas plazas están ocupados por mexicanos, ya que las universidades en la región no imparten ningún tipo de clases relacionadas con el procesamiento de señales digitales, y aún menos enfocadas principalmente a la ecualización de música. Es necesario prepararse en estas nuevas tecnologías para poder incidir en la industria regional, y poder demostrar que los estudiantes de nivel licenciatura son capaces de poder comprender este tipo de estudios e implementarlos posteriormente en un ambiente laboral o académico.

1.3 Justificación

A la fecha, en la Universidad Autónoma de Ciudad Juárez (UACJ) poco se ha hecho en cuanto a proyectos de ecualización de música. Los DSP no son conocidos en la universidad por los estudiantes y egresan de nivel licenciatura sin conocimiento alguno al respecto.

Obteniendo este nivel de conocimiento se puede obtener más experiencia en el ámbito laboral y curricular. Se usarán los DSP TMSVC5510 ya que son de los más veloces que existen en el mercado para la ecualización de música y audio.

1.4 Objetivo

Analizar el algoritmo de ecualización de música, para comprobar su operación en tiempo real mediante su implementación en el DSP TMSVC5510, con el fin de adquirir un nivel de experiencia y conocimiento de nivel medio en esta tecnología.

1.5 Preguntas de Investigación

¿Cuál es el mayor problema para trabajar en tiempo real?
¿Qué ventajas tienen estos algoritmos sobre los que no funcionen en tiempo real?
¿Qué otras ventajas se puede tener al implementar esta tecnología?
¿Qué efecto produce un filtro al aplicarlo a una señal digital?

1.6 Procedimiento

Se investigaron los antecedentes y trabajos relacionados con la ecualización de música, familiarizando con el lenguaje y las herramientas de programación, en este caso el Lenguaje C y el *Code Composer Studio* (CCS).

Habiendo realizado una investigación del tipo explicativa y comparativa, lo que se buscó fue aprender y comprender el funcionamiento de los algoritmos para poder compararlos posteriormente. Se estudiaron los diferentes algoritmos disponibles y se revisaron las diferencias y similitudes entre ellos. Se realizaron pruebas con cada uno de ellos y se registraron los datos arrojados, así como el tiempo que tardaron en responder y la calidad con que se escucha la música ya procesada.

Realizando prácticas y pruebas de los algoritmos en el laboratorio del Cuerpo Académico de Instrumentación y Procesamiento de Señales (CAIPS), se utilizaron los instrumentos DSP TMS320VC5510, la tarjeta DSK 5510, en un ambiente de programación *Code Composer Studio*. De esta forma se seleccionó el algoritmo más eficiente y se escribió un reporte justificando el porqué de la decisión tomada.

Capítulo 2. Marco Teórico

Por no estar familiarizados con este tema, se tuvo que investigar las bases y los conceptos fundamentales de todo lo que se va utilizar, desde los detalles más mínimos hasta los procesos, herramientas y temas más sofisticados que utilizamos en nuestro proyecto. Nosotros buscamos dar a entender lo que se necesita para comprender en que consiste la ecualización de música y de cuáles son las características que se deben cumplir para poder trabajar en tiempo real. Como también otros conceptos de conocimiento general acerca del sonido, las ondas sonoras y los tipos de ondas que existen y realización de muestreos.

2.1 Sonido

El sonido puede definirse como una perturbación introducida en un medio elástico que consiste en condiciones alternadas de compresión y rarefacción, también se puede definir como una transferencia de energía de un lugar a otro.

Además, puede transmitirse por medio del aire a temperatura ambiente a una velocidad de 340 m/s y en el agua a 1400m/s. Entre más sólido el medio, tiene mayor conductividad. En el caso de solidos como el hierro, el sonido puede propagarse a una velocidad de 5000 m/s pero a medida que avanza su energía se va atenuando y desaparece debido a que los medios no son perfectamente elásticos.

El sonido al propagarse adquiere forma de ondas que obedecen las leyes de la física y se representan en forma de una sinusoidal periódica (Figura 1). El intervalo de tiempo en el que se efectúa un movimiento completo se llama periodo, a la variación completa desde la cresta hasta la depresión se denominan ciclo y al número de ciclos que se producen en una unidad de tiempo se le llama frecuencia generalmente medida en *Hertz* (ciclos por segundo). Al comparar los distintos sonidos se llegaron a clasificar en agudos y graves, aquellos que poseen una mayor frecuencia de vibración son conocidos como agudos y aquellos con una menor frecuencia entran en la categoría de graves. La longitud de onda es la distancia que recorre una onda en un determinado periodo de tiempo, en otras palabras es el tiempo en que recorre un ciclo completo. Las ondas también poseen diversas amplitudes de onda o vibración que representan la diferencia de sonoridad de cada sonido. La

intensidad del sonido en un punto dado, ya sea en un micrófono o en un tímpano humano es la cantidad de energía que llega a dicho punto y es medida en decibeles (dB) [11].

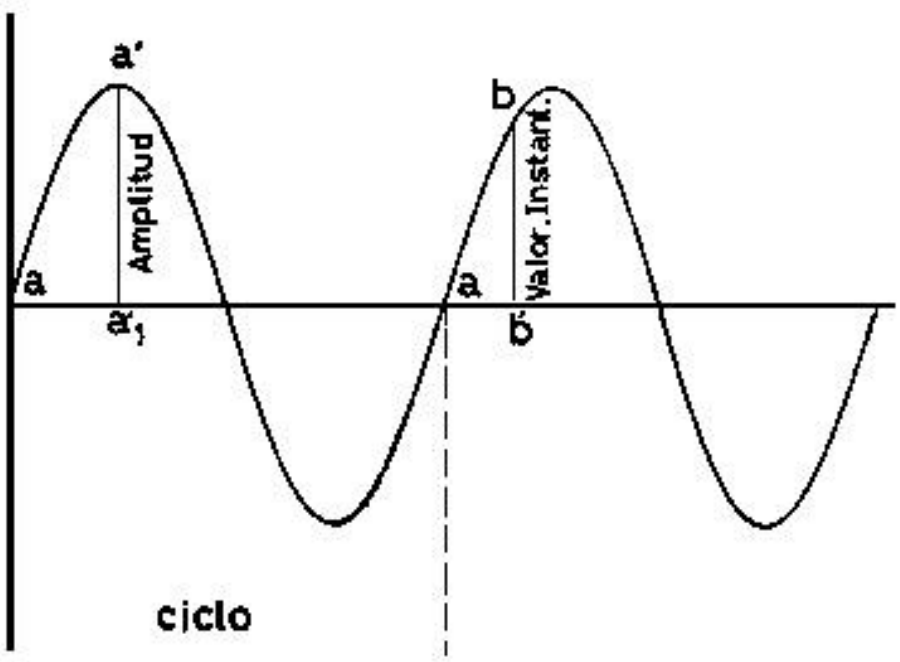

Figura 1. Diagrama de una Onda [11].

2.1.1 Ondas Sonoras

Las ondas sonoras son aquellas que transportan lo que definimos como sonido. Estas ondas se desplazan a través de todas las moléculas cercanas a la fuente, y estas a su vez las transmiten a las moléculas contiguas permitiendo así la propagación del sonido. Como ya habíamos mencionado, uno de los factores que influyen en la velocidad de transmisión es la cohesión de las moléculas, entre más cercas estén la una de la otra más fácil será la transmisión de la onda, es decir el aire es un mal conductor a diferencia del hierro. Otros factores que influyen en la transición de dichas ondas son: presión, humedad y temperatura. Estas ondas producen vibraciones o movimientos vibratorios, y debido a estos movimientos se clasifican en distintas maneras [12].

2.1.2 Tipo de Ondas

Las ondas se clasifican en dos grandes ramas que son Longitudinales y Transversales. La mayoría de las ondas quedan clasificadas en alguno de estos campos, pero estos a su vez se dividen en Ondas Planas y en Ondas Esféricas.

- Ondas Longitudinales. Son las ondas que provocan un movimiento oscilatorio paralelo a la dirección de propagación de la onda. El sonido está compuesto de este tipo de ondas.

- Ondas Transversales. Son aquellas cuyas oscilaciones ocurren de manera perpendicular a la dirección en que se propagan. Cabe mencionar que este tipo de ondas no producen sonido, solo se presentan en movimientos sísmicos y campos electromagnéticos.

- Ondas Planas. Son ondas de frecuencia constante y se propagan en una sola dirección a lo largo del espacio, si la onda se propaga en una dirección única, sus frentes de onda son planos y paralelos.

- Ondas Esféricas. Son ondas que se propagan a la misma velocidad en todas direcciones y se llamas así porque sus frentes de onda son esferas concéntricas, cuyo centro coincide con la posición de la fuente de la perturbación de todas las direcciones [13].

2.1.2.1. Señal Monoaural

Esta es una señal que toda la información de un sonido va en un solo canal, es decir que solo puede escucharse por una sola salida o un altavoz. En caso de que el sistema sea reproducido en un sistema con dos o más altavoces y todos estos reproducen la misma señal. A pesar de esto, el sonido sigue siendo monofónico siempre y cuando ambos altavoces reproduzcan la misma señal. Esta señal es la forma más básica de trasmitir señales [14].

2.1.2.2 Señal Estereofónica

Se refiere al grabado y reproducción de señales por medio de dos canales, hoy en día esta es la forma más común de transmitir sonidos y reproducir canciones. El propósito de dividir el audio en dos canales es crear una experiencia más natural que por lo menos permite apreciar los sonidos que vienen de izquierda o derecha. Aunque este término se refiere exclusivamente para las señales en dos canales, también suele empleares para aquellas transmitidas en más de dos [15].

2.1.3 Muestreo de Señales de Audio

El audio es analógico desde su origen, por lo tanto los sistemas digitales deben transformar este aspecto a través del muestreo y cuantificación, herramientas que permiten la digitalización de la señal. Para esto debe aplicarse el teorema de muestreo que es pieza fundamental de la conversión analógica - digital.

En las grabaciones digitales, la información está representada por unos y ceros que provienen de procesos de muestreo codificando la señal analógica en una secuencia de valores de amplitud. A pesar de tomar la señal digital solo de muestras, si se realiza de manera adecuada no se perderán datos. Para lograr esto la señal análoga debe capturarse uniformemente y que ruidos u otra cosa no intervengan, como esto en ocasiones suele no ser posible entonces se puede aumentar el número de muestras tomadas por segundo y así lograr una mejor fidelidad.

Para lograr un buen muestreo del sonido se debe de utilizar un filtro de paso bajo que elimina las frecuencias altas creando así una frecuencia de banda limitada y por consiguiente se evitará la pérdida de la información y se tendrá la capacidad de recuperar la señal original.

Todo esto no sería posible sin el teorema de muestreo, que establece que: “Una señal continua limitada en banda puede ser reemplazada por una secuencia discreta de muestras sin pérdida de información”, también describe como una señal puede ser reconstruida a partir de sus muestras. Por otra parte, el teorema especifica que la frecuencia

de muestreo debe ser al menos el doble de la máxima frecuencia de la señal (frecuencia Nyquist).

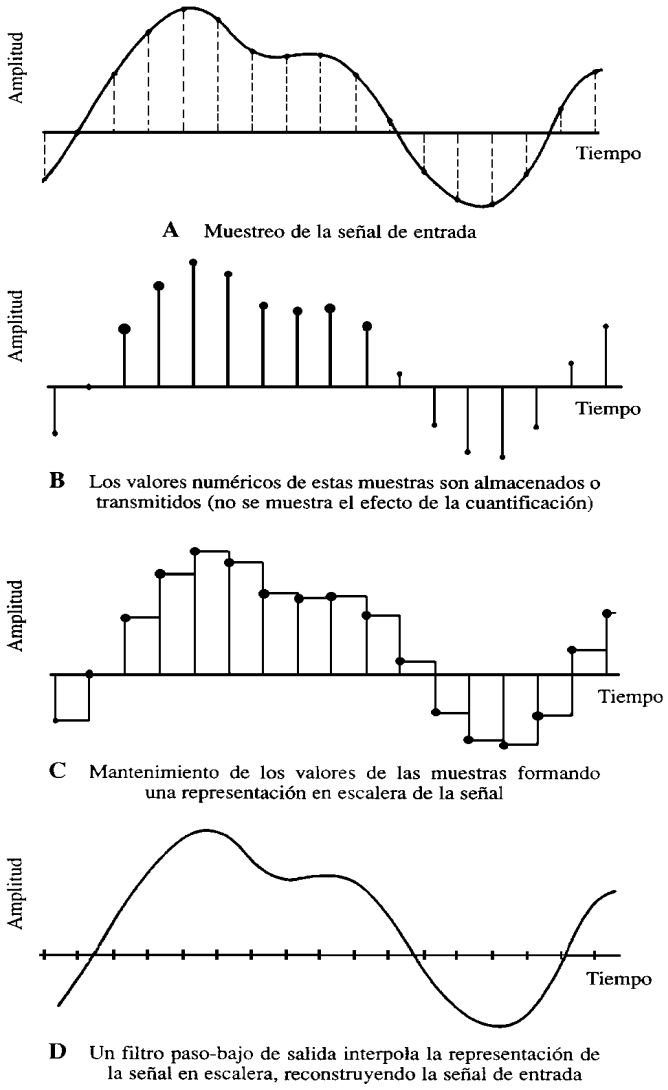

Figura 2. Digitalización y reconstrucción de una señal a partir del muestreo [16].

La figura 2 muestra cómo una señal análoga puede ser convertida a digital y como la señal original se puede reconstruir a partir de las muestras. El teorema de muestreo nos dice que se debe de limitar el ancho de banda a la mitad de la frecuencia de muestreo para evitar una distorsión conocida como "*aliasing*". El *aliasing* es un fenómeno anómalo que puede crear falsos componentes de una señal que aparecen dentro del ancho de banda y son

indistinguibles de las verdaderas. Pero este problema tiene una solución muy simple, solo se tiene que limitar la entrada de la señal al ancho de banda del receptor por medio de un filtro paso–bajo (*anti-aliasing*) que proporcione una atenuación significativa para asegurar que la frecuencia de la señal no superare el valor de la frecuencia Nyquist [16].

2.2 Herramientas y Dispositivos

En esta sección se introducen las herramientas que se utilizaron para la realización del ecualizador de música en tiempo real. Como primera herramienta se tiene el programa *Code Composer Studio*, que proporciona un entorno integrado de desarrollo para hacer uso de la segunda herramienta, que en este caso hablamos de un DSP TMS320VC5510 de *Texas Instruments*. Y la tarjeta DSK 5510 ya que está diseñado para trabajar con el *Code Composer Studio*.

Los dispositivos que nos ayudaran a realizar las pruebas para una mejor comprensión de lo que se realiza son los accesorios de audio tales como: micrófonos y bocinas [17].

2.2.1 DSK 5510

Para poder trabajar con el núcleo del TMS320VC5510 es necesario que se conecte primero a una tarjeta de audio, en éste caso el DSK 5510. El DSK 5510 es una plataforma de desarrollo que permite a los usuarios desarrollar y evaluar aplicaciones para la familia de procesadores C55. El DSK 5510 también sirve como *hardware* de referencia para el DSP TMS320VC5510 [18].

El DSK viene con dispositivos integrados que se adaptan a una amplia variedad de entornos de aplicaciones. Las características clave del 5510 incluyen:

- El DSK 5510 opera a 200 MHz.
- Un códec estéreo AIC23.
- 8 MB de memoria DRAM síncrona.
- 512 KB de memoria flash no volátil.

- 4 indicadores LED de acceso de usuario y DIP *Switches*.
- *Software* de configuración de la placa a través de registros implementados en CPLD.
- Puente de selección de opciones de arranque.
- Conectores estándar de expansión para el uso de tarjeta hija.
- Fuente de alimentación de 5V.

El DSK 5510 posee una unidad lógica programable compleja llamada CPLD y es usada para unir todos los componentes de la tarjeta. El CPLD tiene una interfaz basada en registros de usuario, permitiendo al usuario configurar la tarjeta. El DSK también incluye 4 LED's y 4 DIP *Switches* como una forma sencilla de proporcionar al usuario una retroalimentación interactiva. Se puede tener acceso a ambos mediante la lectura y escritura en los registros de CPLD [18].

El DSK 5510 está diseñado para para trabajar con el *Code Composer Studio* (CCS) de *Texas Instruments*, que se comunica con la tarjeta a través de un emulador de JTAG integrado. El *Code Composer Studio* está formado por una serie de herramientas de desarrollo y depuración de aplicaciones embebidas. Incluye compiladores para cada una de las familias de dispositivos de *Texas Instruments*, un editor de código fuente, un ambiente de construcción de proyectos, un depurador, simulador y más.

El CCS tiene una interfaz única que lleva al usuario paso a paso a través del desarrollo de la aplicación y está basado en Eclipse un *software framework* abierto, usado en muchas aplicaciones diferentes pero que fue desarrollado como un *framework* abierto para el desarrollo herramientas. Por esto se decidió basar al CCS en Eclipse y por qué provee un excelente *software framework* para ambientes de construcción y desarrollo de *software*, que además se ha convertido en el *framework* estándar más utilizado. CCS combina las ventajas del Eclipse con las capacidades de depuración embebida de *Texas Instruments* en un ambiente de desarrollo que llama la atención de los desarrolladores [19].

La familia del C55x tiene un programa y un espacio de datos unificado a un registro de periféricos especialmente dedicados a esto integrados en la misma tarjeta. El código del

programa es direccionado en bytes (8 bits) mientras que los datos se direccionan en palabras de 16 bits. El 5510 comienza sus direcciones en 0 y de ahí las va numerando hasta ocupar toda la memoria asignada a esto y continúa en caso de tener una memoria externa. La memoria del DSP ocupa los primeros bytes de los registros seguidos por la DARAM (*Dual-Access* RAM) interna que es diferente de la SARAM (*Single-Access* RAM). La memoria interna del 5510 se divide en bloques de 4 *KWord*, cada uno capaz de soportar operaciones independientes. Se puede incrementar su desempeño poniendo código y datos juntos de tal manera que las instrucciones tengan sus operandos distribuidos por distintos bloques sin la necesidad de utilizar sólo uno. Los bloques DARAM son más precisos por su naturaleza de doble puerto que permite un grado mayor de operaciones. El DSK 5510 cuenta con 8 bloques de DARAM y 32 bloques de SARAM para un total de 160 *KWords* de memoria interna. En la figura 3 se puede observar cómo las memorias del C55x y el DSK 5510 comparten todas las direcciones y qué rango ocupan cada una de ellas.

Ahora se enlistarán los componentes físicos y lógicos de la tarjeta: CPLD, Códec AIC23, SDRAM, Memorial Flash, 4 LED's, 4 DIP *Switches*, 3 interfaces de expansión, conector de micrófono, entrada de audio estéreo, salida de audio estéreo, conector para audífonos o bocinas estándar, un interruptor de *reset*, un puerto USB y 2 conectores de 5V, en la figura 4 podemos ver la ubicación de cada uno de ellos.

Comenzaremos con la CPLD (*Complex Programmable Logic Device*) que permite el control de varias características de la tarjeta, como los accesos a memoria, control de la tarjeta hija, interfaces y señales. Esta unidad se divide en 4 registros y cada una tiene una función diferente. Los registros USER_REG son usados para leer el estado de los DIP *Switches* y encender los LED's para así permitir la interacción entre el DSK y el usuario. Los registros DC_REG son usados para monitorear y controlar a la tarjeta hija mediante una interfaz utilizando líneas de estado y líneas de control. Los registros VERSION contiene dos campos de solo lectura que indican la versión de la tarjeta y del CPLD, este registro le permitirá a nuestro *software* diferenciar entre las diferentes variantes del DSK.

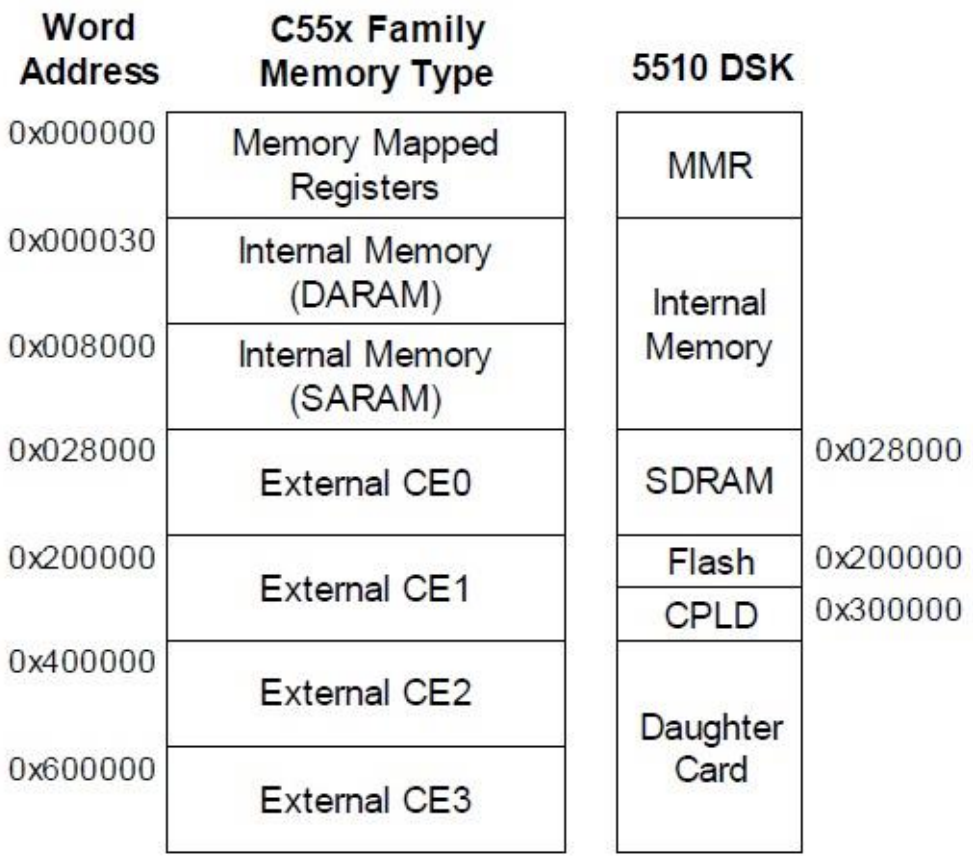

Figura 3. Direccionamientos de las memorias del C55x y del DSK 5510 [18].

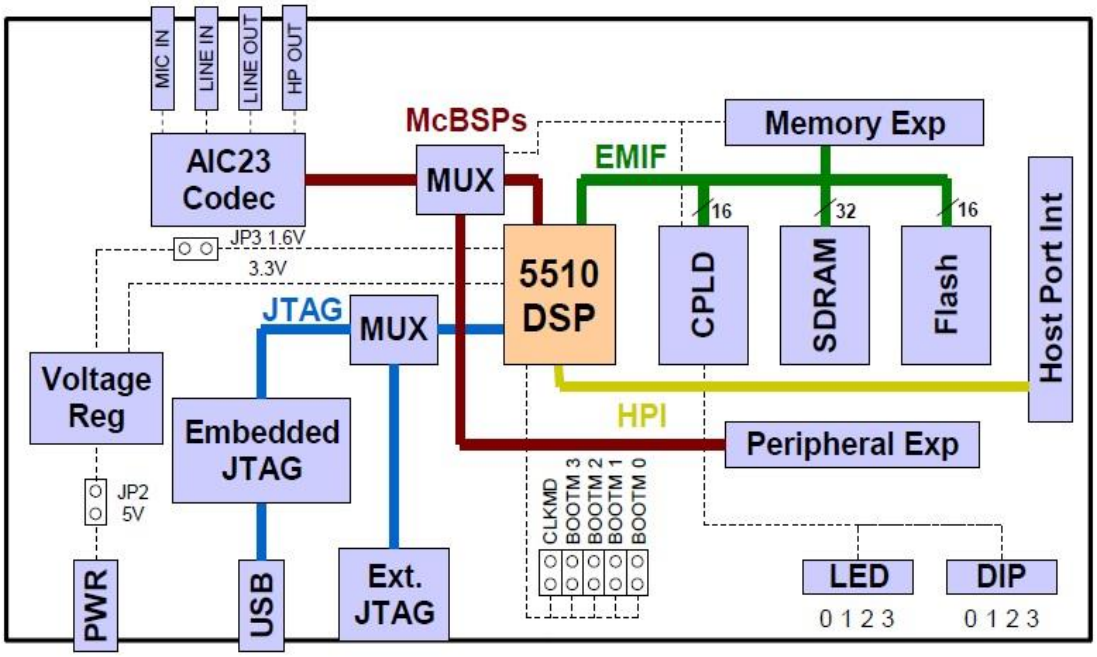

Figura 4. Diagrama de los componentes de la DSK 5510 [18].

Y finalmente el registro MISC, que proporciona control sobre las diversas funciones de la tarjeta. En el caso del DSK 5510 el registro MISC controla como las señales auxiliares son traídas desde la tarjeta hija.

La siguiente parte del DSK 5510 que veremos es el Códec AIC23. Este códec estéreo es para señales de entrada y salida, con el podemos tomar las señales análogas del micrófono y convertirlas en datos digitales para que sean procesadas por el DSP. Cuando el DSP termina de procesar los datos, el DSP utiliza este códec para convertir las señales digitales nuevamente en señales análogas para que sean transmitidas a través de la conexión o puerto para audífonos.

El DSK 5510 también cuenta con una SDRAM síncrona estándar de 64 megabits, mediante una interfaz de 32 bits y un *memory clock* externo de 100MHz. Esta memoria debe ser constantemente actualizada para asegurar la integridad de su contenido, esto es recomendado hacerlo cada 15.6 micro segundos. También posee una memoria flash del DSK que soporta palabras de 256K x 16 bits y sólo es usada por la CPLD. Como ya se había mencionado la tarjeta cuenta con 4 LED's y 4 DIP *Switches* que permiten al usuario una manera simple de introducir y leer datos a través del registro USER_REG del CPLD.

El DSK cuenta con interfaces de expansión donde se pueden conectar las tarjetas hija. El 5510 cuenta con 3 de estos puertos que le ayudan a expandir sus capacidades, construir aplicaciones más específicas de entrada y salida. Aparte de estos 3 conectores, también cuenta con un conector de micrófono, un conector de entrada de audio estéreo, una línea de salida de audio estéreo y un conector para audífono o bocinas; estos 4 últimos son de tipo estándar de 3.5 mm y dos conectores de 5V para energizar la tarjeta. También cuenta con un puerto USB que permite la emulación lógica en el DSK sin necesidad de utilizar emuladores externos. Además de los 4 LED's antes mencionados también tiene otros 4 LED's que le indican al usuario el estado de la tarjeta, como puede ser que este en emulación, energizada, en estado de *reset* o trasfiriendo datos a través del puerto USB, por último la tarjeta cuenta con un *reset switch* que interrumpe el flujo de corriente y voltaje en la tarjeta [18].

2.2.2 TMS320VC5510

Después de ver los componentes físicos del DSK 5510, pasaremos a ver las partes lógicas del TMS320VC5510. En la figura 5 podemos ver las partes como: *Instruction-buffer Unit*, *Idle Domain Register*, *Advanced Power Management*, *Dual Multiply-Accumulate*, *logic arithmetic unit*, *accumulator*, *barrel shifter*, *Address-data flow unit*, *data-computation unit*.

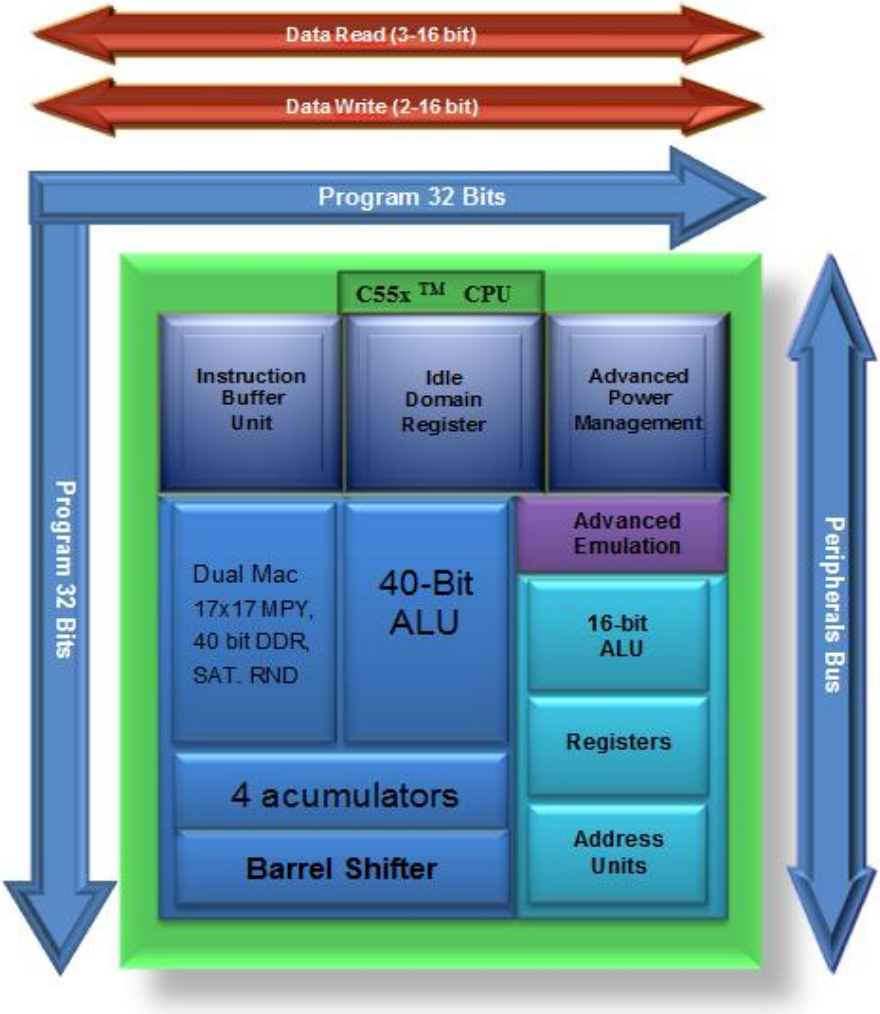

Figura 5. C55x DSP *Core* [20].

El IU (*Instruction-buffer Unit*) del TMS320VC5510 se encarga de llevar la secuencia de instrucciones de la memoria en la CPU. Durante cada ciclo de CPU, el IU recibe 4 bytes de código de programa del bus de sistema de 32 bits, y decodifica de 1 a 6 bytes de código que anteriormente fueron recibidos en la cola. El IU, pasa la información

decodificada al PU (*Program-flow Unit*), el AU (*Address-data-flow Unit*), y el DU (*Data-computation Unit*) para realizar la ejecución de las instrucciones. La cola del IU soporta instrucciones de repeticiones de bloques de código locales, que ejecutan un bloque de instrucciones dentro de la cola.

Registro de dominios inactivos (*Idle Domain Register*), para la administración de potencia existen 6 dominios programables por *software*:

- CPU
- Memoria caché
- Periféricos
- DMA
- Generador de reloj
- EMIF (*External Memory Interface*)

Cada dominio puede ponerse en modo "*low power idle*" cuando no se requieren de sus capacidades. Además de que el usuario tiene la capacidad de controlar la potencia dinámicamente y los dominios idle se configuran por el usuario y le permiten configurar que parte del *hardware* esta activa. [20]

Para el APM (*Advanced Power Management*), en los dominios de inactividad configurables el usuario puede personalizar el consumo de energía para una aplicación específica. El núcleo DSP del C55x extiende la opción de *three fixed idle* del C54x con un total de 64 combinaciones configurables por el usuario de los pasados 6 componentes vistos anteriormente.

El resultado es el uso de la energía más eficientemente y un sistema de menor costo. Al usar menos energía y disipando menos calor, el núcleo del DSP C55x da a diseñadores mayor flexibilidad respecto al diseño de la placa.

DualMAC (*Multiply-Accumulate*), en DSP, MAC es una operación común que calcula el producto de dos números y agrega ese producto a un acumulador:

$$a \leftarrow a + (b * c)$$

Cuando haya terminado con números de punto flotante podría realizarse con 2 *roundings* o con un redondeo único. Cuando se realiza con un redondeo único, se llama un FMA (*fused multiply-add*) o FMAC (*fused multiply-accumulate*).

En las computadoras una unidad MAC, consiste en un multiplicador implementado en lógica combinacional seguido de un sumador y un acumulador de registro que almacena el resultado cuando es registrado. La salida del registro es alimentada a una entrada del sumador, por lo que en cada reloj la salida del multiplicador se agrega al registro. Los multiplicadores combinacionales requieren una gran cantidad de lógica, pero pueden calcular un producto mucho más rápido que el método de desplazamiento y suma el cual es típico en las computadoras antiguas. Los primeros procesadores en equiparse con unidades MAC fueron los procesadores de señal digital, pero la técnica ahora es común en procesadores de propósito general. Las unidades de lógica aritmética (ALU) son el corazón de cualquier microprocesador. El ALU se divide en cuatro partes principales:

- Bloque de aritmética: este bloque es usado para realizar operaciones aritméticas, como suma, resta y de comparación. El núcleo del bloque de aritmética es un sumador.
- Bloque de lógica: este bloque se utiliza para realizar operaciones lógicas de *bitwise* (bit a bit) simples como AND, OR y XOR.
- Multiplexores: estos bloques se utilizan para seleccionar las entradas apropiadas para la aritmética y bloques de lógica. Por lo general más de 2 buses llegan a las entradas del ALU. A veces estos multiplexores se utilizan para realizar algunas operaciones de lógica simple. El MUX de 5-1 es un registro de corrimiento programable (*shifter*).Sus entradas contienen cambios a las versiones de salida MUX de 9-1. El MUX de 2-1 puede ser programado para invertir uno de los operandos (que puede usarse para ejecutar una resta utilizando sólo un sumador).
- Registros: estos bloques se utilizan para almacenar los operandos y los resultados. Por lo general, estos registros no forman parte del archivo de registro del microprocesador (aunque no siempre es el caso).

Se debe tomar en cuenta que hay un bus en bucle desde la salida hacia la entrada del ALU, lo que permite utilizar los resultados recién calculados como operandos en el próximo ciclo [16].

El acumulador es un registro de propósitos generales de 8 bits usado para almacenar operandos, resultados de cálculos aritméticos, y de manipulación de datos. Además, es directamente accesible a la CPU (unidad central de procesamiento) para operaciones no aritméticas. El acumulador es usado durante la ejecución de un programa donde el contenido de alguna posición de memoria es cargado en el acumulador. También, la instrucción: almacenar (sta), causa que el contenido del acumulador sea almacenado en alguna posición de memoria preestablecida [17].

Sin un registro acumulador, sería necesario escribir el resultado de cada cálculo (adición, multiplicación, desplazamiento, etc.) en la memoria principal, para tal vez ser leída nuevamente en la siguiente operación. El acceso a la memoria principal es más lento que el acceso a un registro como el acumulador, porque la tecnología usada para la memoria principal es más lenta y barata que la usada para un registro interno del CPU.

Un ejemplo para el uso del acumulador es cuando se suma una lista de números. El acumulador inicialmente se pone en cero, después cada número es sumado al valor en el acumulador. Sólo cuando se han sumado todos los números, el resultado en el acumulador es escrito a la memoria principal o a un registro no acumulador de CPU.

En una arquitectura de computadora, lo que distingue un registro acumulador de uno que no lo sea, es que el acumulador puede ser usado como operando implícito para las instrucciones aritméticas (si la arquitectura tuviera uno).

Otro ejemplo, una computadora puede tener una instrucción como:

:: Addmemaddress

Esta instrucción agregaría el valor leído en la posición de memoria indicada en memaddress al valor del acumulador, poniendo el resultado en el acumulador. El acumulador no es identificado en la instrucción por un número del registro; es implícito en la instrucción y ningún otro registro puede ser especificado en la instrucción. Algunas arquitecturas utilizan un registro particular como acumulador en algunas instrucciones, pero

en otras instrucciones usan números de registros como especificación explícita del operando.

Un *barrel shifter* es un circuito digital combinacional que puede desplazar una palabra por un determinado número de bits en un sólo ciclo de reloj, ya que es construido con lógica combinacional.

Por ejemplo, un *barrel shifter* de 4 bits, con entradas A, B, C y D. El *shifter* puede ciclar el orden de los bits ABCD, DABC, CDAB o BCDA; en este caso, ningún bit se pierde. Es decir, pueden cambiar todas las salidas hasta tres posiciones hacia la derecha (y hacer cualquier combinación cíclica de A, B, C y D). El *barrel shifter* tiene varias aplicaciones, además de ser un componente útil en microprocesadores (al lado de la ALU) [18].

El UA (*Address-data flow unit*), sirve como administrador de acceso de datos, para la lectura de datos y escritura de buses. Además de que genera las direcciones de espacio de datos, para la escritura y lectura de estos datos. El UA se compone de 8 registros auxiliares extendidos de 23 bits (XAR0-XAR7), 4 registros temporales de 16 bits (T0-T3), un puntero extendido de 23 bits del coeficiente de datos (XCDP) y un puntero de pila extendida de 23 bits (XSP). El ALU tiene 16 bits adicionales que pueden utilizarse para operaciones aritméticas simples. Los registros temporales pueden utilizarse para expandir el compilador eficientemente minimizando la necesidad de acceso a la memoria. El AU permite 2 registros de la dirección y un puntero de coeficiente para ser usadas juntas para el procesamiento de información doble en un ciclo de reloj único. El AU admite hasta cinco *buffers* circulares.

El DU (*data-computation unit*), maneja el procesamiento de datos de la mayoría de las aplicaciones del C55x. El DU consiste en 2 unidades MAC, ALU de 40 bits, 4 acumuladores de 40 bits (AC0, AC1, AC2 y AC3), un *barrel shifter*, saturación y redondeo de control lógico. Tiene 3 puertos de lectura de datos que habilitan dos caminos que se simultáneamente conectan con las unidades MAC. En un solo ciclo cada unidad MAC puede hacer una multiplicación de 17 bits y una suma de 40 bits o una resta con opción de saturación. El ALU puede hacer operaciones aritméticas, lógicas, de redondeo y de saturación de una tamaño no mayor a 40 bits, también puede archivar 2 operaciones de 16

bits en su dos acumuladores (alto y bajo) al mismo tiempo. El ALU puede aceptar valores inmediatos del IU como datos o con registros de otra AU o PU [19].

2.2.3 *Code Composer Studio*

Code Composer Studio (*CCStudio*) es el entorno de desarrollo integrado para los DSP's de *Texas Instruments* (TI), micro controladores y procesadores de aplicación. *CCStudio* incluye una serie de herramientas que se utilizan para desarrollar y depurar aplicaciones incorporadas. Incluye compiladores para cada una de las familias de dispositivos de TI, editor de código fuente, entorno de generación de proyecto, depurador, perfiles, simuladores y otras características. *CCStudio* proporciona una interfaz de usuario única llevando a los usuarios a través de cada paso en el desarrollo de la aplicación. Las interfaces y herramientas familiares permiten a los usuarios empezar a trabajar más rápido que nunca y agregar funcionalidad a su aplicación gracias a herramientas de productividad sofisticados.

El depurador integrado de *CCStudio* tiene capacidades específicas de DSP y puntos de interrupción avanzados para simplificar el desarrollo. Puntos de interrupción condicional o *hardware* basados en expresiones completas en C, variables locales o registros. La ventana de memoria avanzada le permite inspeccionar cada nivel de la memoria para poder depurar problemas complejos de coherencia caché. *CCStudio* apoya el desarrollo de sistemas complejos con múltiples procesadores o núcleos. Puntos de interrupción globales y operaciones sincrónicas proporcionan control sobre múltiples procesadores y núcleos.

Los perfiles interactivos del *CCStudio* facilitan la medición del rendimiento del código y aseguran el uso eficiente de los recursos del objetivo del DSP durante las sesiones de depuración y desarrollo. El analizador permite a los desarrolladores fácilmente perfilar todas las funciones de C y C++ en su solicitud de ciclos de instrucción o de otros eventos, como errores y aciertos de caché. Los rangos del perfil pueden utilizarse para concentrar esfuerzos en áreas de alto uso de código durante la optimización, ayudando a los desarrolladores a producir códigos finamente ajustados. Los perfiles están disponibles para rangos de ensamble, para cualquier combinación en código C o C++. Para aumentar la

productividad, todas las instalaciones de perfiles están disponibles durante todo el ciclo de desarrollo.

Algunas tareas como lo son las pruebas, necesitan ejecutar por horas o días sin interacción del usuario. Para realizar esta tarea, el IDE debe ser capaz de automatizar tareas comunes. *CCStudio* tiene un completo entorno de secuencias de comandos (*scripting*) que permite la automatización de tareas repetitivas, como pruebas y evaluaciones del rendimiento. Una consola separada de secuencias de comandos le permite escribir comandos o ejecutar secuencias de comandos dentro del IDE.

TI ha desarrollado los compiladores de C y C++ específicamente diseñados para maximizar el rendimiento y uso del DSP. Los compiladores de TI utilizan una amplia gama de clásicos DSP orientados y dispositivos específicos de optimizaciones sofisticadas que se adaptan a la arquitectura del DSP.

Los compiladores de TI también realizan programas de nivel óptimo que evalúan el rendimiento del código a nivel de aplicación. Con la vista de nivel de programa, el compilador es capaz de generar un código similar a un desarrollador de programa ensamblador que tiene vista de todo el sistema. Este punto de vista de nivel de aplicación es aprovechada por el compilador para hacer compensaciones que aumentan significativamente el rendimiento del DSP.

Los simuladores proporcionan una forma, en la cual los usuarios comienzan un previo desarrollo para tener acceso al desarrollo de la tabla. Los simuladores también tienen la ventaja de proporcionar una mejorada visibilidad del rendimiento y el comportamiento de las aplicaciones. Varias variantes del simulador están disponibles permitiendo a los usuarios compensaciones fuera del ciclo de precisión, velocidad y simulación periférica, con algunos simuladores siendo ideales para el algoritmo de evaluación comparativa y otros para la simulación de sistemas más detallados.

DSP/BIOS 6.x DSP/BIOS 6.x es un sistema avanzado en tiempo real compatible con ARM926, ARM Cortex M3, C674x, C64x +, C672x y dispositivos basados en 28 x. DSP/BIOS 6.x ofrece numerosas mejoras en kernel y depuración que no las tiene el DSP/BIOS 5.x, incluyendo más rápidos y flexibles eventos, gestiones de memoria y prioridad de herencia de exclusiones mutuas (*Mutexes*, del inglés *Mutual exclusion*).

DSP/BIOS 6.x incluye una capa de compatibilidad DSP/BIOS 5.x para admitir la fácil migración del código fuente de la aplicación. El DSP/BIOS 5.x proporciona servicios de multitarea preventiva para dispositivos DSP. Los servicios del DSP/BIOS 5.x incluyen interrupciones de *software*, semáforos, mensajes, E/S de dispositivos, administración de memoria y de energía. Además, el DSP/BIOS 5.x también incluye instrumentación de depuración y herramientas, junto con la impresión de baja sobrecarga y recopilación de estadísticas [21].

2.3 Tiempo Real

Un sistema en tiempo real es aquel que requiere que reaccione a estímulos del ambiente, incluso el paso del tiempo, en intervalos dictados según el entorno. El diccionario Oxford define a un sistema de tiempo real como: cualquier sistema en el que el tiempo que tarde el sistema en producir una respuesta es significativo. Es todo se debe a que generalmente los datos de entraba provienen del mundo físico, por la tanto la salidas o respuestas del sistema tienen que estar relacionadas con el mismo. La diferencia entre el tiempo de entrada y tiempo de respuesta debe de ser lo suficientemente pequeña para que sea aceptable. Otra manera de definir un sistema en tiempo real puede ser, cualquier actividad de procesos de información o sistema que tiene que responder a entradas de estímulos generados externamente en un periodo corto y finito de tiempo. Generalmente los sistemas en tiempo real son sistemas que mantienen una interacción de tiempo continuo con su ambiente.

La certeza con que un sistema funciona no depende en solo los resultados sino en el tiempo que este se generan. Un sistema en tiempo real debe de ofrecer una respuesta satisfactoria o tener consecuencias significantes. Si las consecuencias dan como resultado una degeneración o un mal desempeño se dice que el sistema es suave. Si las consecuencias se refieren a un fallo del sistema se dice que es un sistema duro [22].

2.4 Filtros Digitales

El filtrado digital es una operación de convolución de la imagen original con la función filtro. La operación de convolución entre las funciones $f_1(t)$ y $f_2(t)$ se define mediante una tercera la función $f(t)$ del siguiente modo:

$$f(t) = f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(x)\, f_2\ (t-x)\ dx$$

Considerando en general una imagen de entrada finita y discreta caracterizada por la función de luminancia z(i, j) – en la cual i, j representan las coordenadas de las celdillas-, una imagen de salida z'(i, j) y un sistema de filtrado W, cuya función de respuestas es w(m, n) –siendo m, n las coordenadas matriciales de la función de filtrado, que en general diferirán del sistema de referencia de la imagen-, puede describirse el proceso de filtrado como la siguiente operación de convolución:

$$z'(i,j) = w(m,n) * z\ (i,j)\ = \sum_{m=-k}^{k} \quad \sum_{n=-1}^{1} \quad w(m,n)\ z(i-m,\ j-n)$$

Es la ecuación del filtrado especial de una imagen digital, donde el operador w representa la matriz deslizante de dimensión (2k+1) x (21+1) utilizada en el filtro y donde i y j son respectivamente las líneas y columnas de la imagen. Los elementos $w_{m,\ n}$ constitutivos de la matriz son denominados coeficientes de peso, y el entorno [-k, k] x [-l, l] ventana del filtro. Usualmente k = l, es decir, la ventana de filtrado es cuadrada. Así, el filtrado de una imagen mediante la matriz deslizante:

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix}$$

Dará a una imagen de la forma:

$$z'\ (i, j) = w_{11}\, z\ (i-1, j-1) + w_{12}\, z\ (i-1, j) + w_{13}\, z\ (i-1, j+1) +$$
$$w_{21}\, z\ (i, j-1) + w_{22}\, z\ (i, j) + w_{23}\, z\ (i, j+1) +$$
$$w_{31}\, z\ (i+1, j-1) + w_{32}\, z\ (i+1, j) + w_{33}\, z\ (i+1, j+1)$$

A medida que k y l son mayores, la influencia del entorno de cada celdilla en el nivel digital asignado a la celdilla resultante será progresivamente mayor, modificándose en consecuencia la apariencia de la imagen filtrada. Habitualmente los filtros utilizados son de 3x3 elementos como se muestra en la figura 6 [23].

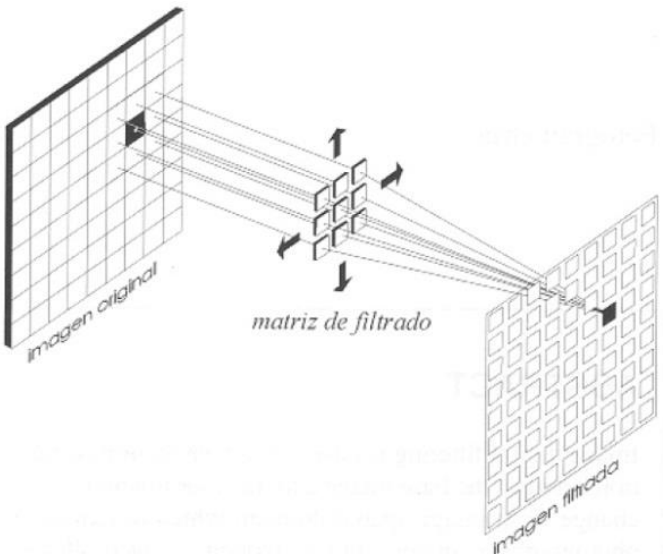

Figura 6. La matriz deslizante de filtrado en el dominio espacial [23].

2.4.1 Filtro FIR

En un filtro con respuesta al impulso finita o FIR, los coeficientes a_k son cero, lo que significa que la respuesta del filtro depende solamente de la entrada y no de valores pasados de la salida. Este tipo de filtros tiene una respuesta finita ya que no exhiben recursión.

La figura 7 muestra la implementación de un filtro FIR. Una de las propiedades más interesantes de este tipo de filtros es la simetría de sus coeficientes y el hecho que pueden ser diseñados de tal forma de exhibir una respuesta de fase lineal.

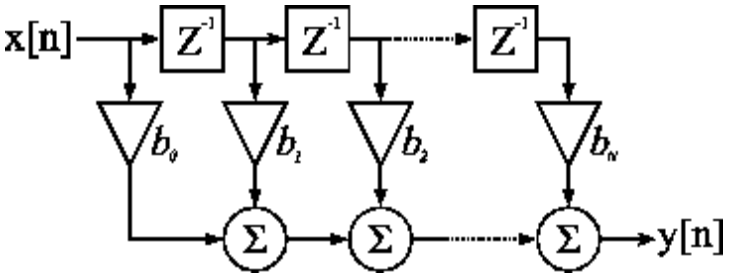

Figura 7. Esquema de implementación de un filtro FIR [24].

La función de transferencia de un filtro FIR está dada por:

$$H(z) = \sum_{i=-\infty}^{\infty} h_n z^{-n} = \sum_{n=0}^{M} b_n z^{-n}$$

Los filtros FIR tienen las siguientes propiedades:

- Respuesta de fase lineal (si se diseñan adecuadamente)
- Estabilidad con coeficientes cuantizados
- En general, requieren de un mayor orden que los filtros IIR.

El diseño de filtros FIR se basa usualmente en una aproximación directa de la magnitud de la respuesta deseada a la cual se le agrega la condición de una respuesta de fase lineal [24].

$$h(N-1-n) = \pm h(n)$$

Capítulo 3. Diseño del ecualizador

En esta investigación se realizó la comparación de los algoritmos de Filtros FIR, los cuales fueron Pasa Bajas, Pasa Altas y Pasa Bandas. Para lograr esto además de la tarjeta y el procesador se utilizó el *Code Composer Studio* (CCS) y *Matlab*. Se comparó cómo son afectadas las señales de audio por los diferentes tipos de filtros al aplicarlos sucesivamente utilizando la aplicación de gráficas en el CCS, se pudo observar los diferentes efectos de cada filtro, además de que se pudo percibir estas diferencias de manera audible. A continuación se presentaran los materiales y métodos utilizados en esta investigación.

3.1 Descripción del área de estudio

Este proyecto está enfocado especialmente a lo que es el procesamiento digital de señales en tiempo real, que servirá como un punto de inicio para futuras investigaciones o proyectos del Cuerpo Académico de Instrumentación y Procesamiento de Señales. Además de servir como una herramienta para demostrar los principios básicos de tiempo real y ecualización.

3.2 Materiales

Los materiales para poder realizar el proyecto son los siguientes:

1. DSP TMS320VC5510.- Fue primordial ya que se encargó para el procesamiento de las señales así como de ejecutar el programa que controla la tarjeta.
2. Tarjeta DSK 5510.- Esta la interface principal ya que todos los dispositivos están conectados a ella.
3. *Code Composer Studio* (CCS).- Se utilizó para programar el DSP y visualizar el resultado de las gráficas al comparar las señales de entrada y de salida de los filtros.
4. *Matlab*.- Esta herramienta fue de utilidad para realizar el diseño de los filtros a comparar.
5. Cable de audio de 3.5mm macho a macho.- Sirvió para realizar la conexión mandando las señales desde una computadora hacia la interfaz de entrada del *Stereo códec* AIC23 de la tarjeta.

6. Bocinas y auriculares.- Fue de utilidad para observar los resultados de los filtros de manera audible.

3.3 Métodos

Después de familiarizarse con el equipo y herramientas se continuó con la investigación sobre el filtro. Los filtros están formados por un arreglo de enteros signados de 16 bits, que conforman los coeficientes, la cantidad de coeficientes determina el orden del filtro, entre más grande sea el orden del filtro más eficiente será este. Los coeficientes fueron calculados usando *Matlab* según los parámetros estándar que se utilizan en las bandas ecualizadoras de: 31, 63, 125, 250, 500, 1000, 2000, 4000, 8000 y 16000 Hz. Cada filtro requería de su propio arreglo para poder filtrar una parte del ancho de banda de las frecuencias, que consta de todas la frecuencias que son audibles, que van desde 0 hasta los 16000 Hz. Por tanto, el ancho de banda debió de ser dividido en 10 secciones denominadas bandas, donde cada sección debería de abarcar el doble de la magnitud de frecuencia que la sección anterior. Como ya se había mencionado, para poder calcular los coeficientes se utilizaron funciones de *Matlab* (ver anexo 1), a partir de las secciones calculadas anteriormente convertidas a radianes/muestra (unidades entre cero y uno), siendo el 16000 equivalente a 1.

Matlab requiere de tres parámetros para calcular los coeficientes: el orden, en este caso 208 (N), la frecuencia de la banda pasante y la frecuencia de rechazo en radianes/muestra. En la figura 8 se muestra que un filtro FIR está compuesto de una banda pasante, conformada por las frecuencias toleradas por este filtro, las cuales no se verán afectadas de ninguna forma. La banda de rechazo, contraria a la pasante está formada por aquellas frecuencias que se verán reducidas o atenuadas. Las frecuencias de la banda pasante y de rechazo (W_P y W_{St} respectivamente) son solo los puntos que dividen el ancho de banda en las bandas de rechazo y pasante, dejando en medio de estas una banda de transición. Finalmente *Matlab* arrojaba un arreglo de números en punto flotante al cual se le aplicaron corrimientos de punto fijo para facilitar su manejo en *Code Composer Studio* y poder realizar las operaciones correspondientes. En la aplicación, se utilizó la función FIR2 de *Code Composer Studio* la cual toma los coeficientes del filtro para procesar los datos del

buffer haciendo múltiples operaciones con ellos y depositando estos resultados en un buffer distinto (ver anexo 2).

Las bandas finales fueron diseñadas para abarcar el mayor número de frecuencias posibles para cada una y finalmente quedaron de la siguiente manera: de 0 a 840Hz, de 1160 a 1840Hz, de 2160 a 3840 Hz, 4160 a 7840Hz y de 8160 a 16000Hz.

Nótese que los valores de las bandas no son continuos, y esto se debe a que se creó una sexta banda que resulta en la suma de las 5 bandas anteriores, para lograr esto es necesario que sea de esta forma ya que de lo contrario al efectuar la suma se podría crear un empalme entre las bandas y producir resultados distorsionados.

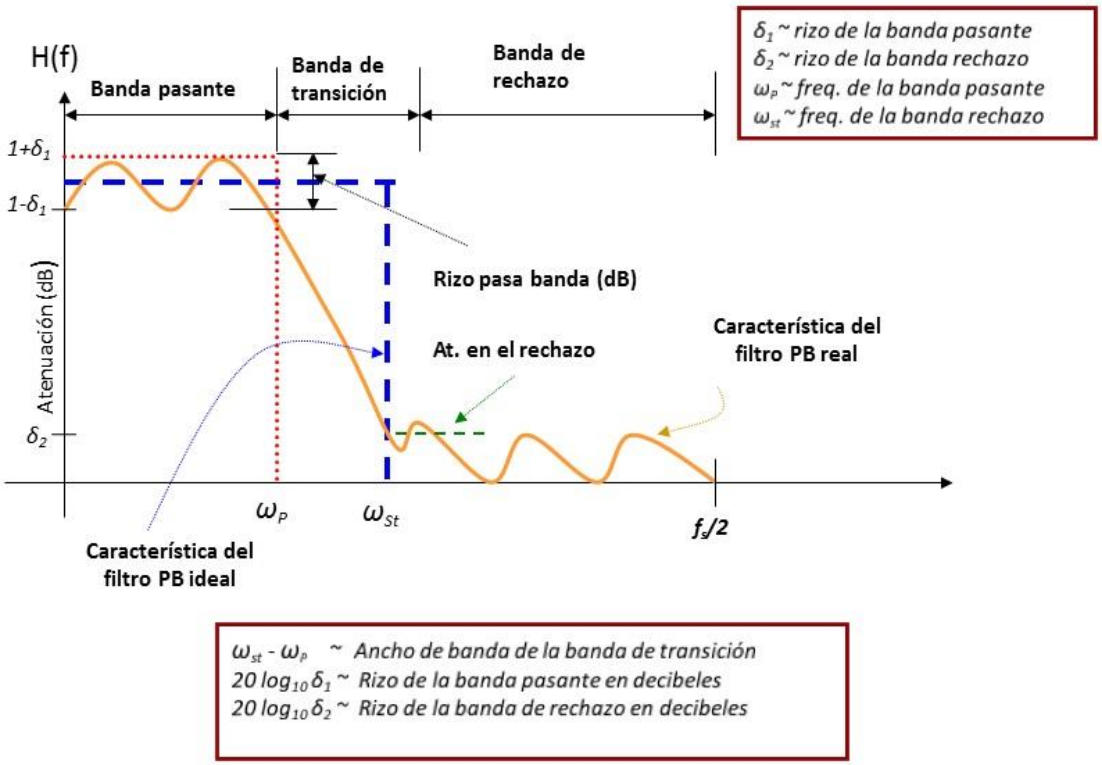

Figura 8. Parámetros del filtro FIR [17].

El programa del proyecto consiste en una aplicación que de descarga dentro de la memoria del DSK 5510 y se ejecuta en este mismo, que reproduce la música y aplica los filtros programados utilizando los DIP *switches*, mismos que deben ser activados en numeración binaria para aplicar el filtro indicado. Las funciones del DIP *switch* son:

Posición del DIP *switch* Función Producida

- 0000 ⟶ La música no pasa por el ecualizador.
- 1000 ⟶ Se activa el filtro #1 (0 a 840Hz).
- 0100 ⟶ Se activa el filtro #2 (1160 a 1840Hz).
- 1100 ⟶ Se activa el filtro #3 (2160 a 3840 Hz).
- 0010 ⟶ Se activa el filtro #4 (4160 a 7840Hz).
- 1010 ⟶ Se activa el filtro #5 (8160 a 16000Hz).
- 0110 ⟶ Se activa la suma de los 5 filtros anteriores.

Capítulo 4. Resultados

Al aplicar el filtro FIR a las distintas frecuencias producidas por la música se pudieron apreciar los efectos ocasionados, obteniendo una atenuación audible, y visible por medio de gráficas. Dichas gráficas fueron de utilidad para notar la atenuación que resultaba del filtro cuando estas no podían ser captadas fácilmente por el oído humano. En una canción se pueden encontrar distintas frecuencias producidas por los instrumentos musicales o por el cantante, al atenuar alguna frecuencia o rango de frecuencias en este caso, podemos identificar a que instrumento pertenece al percatarse de que el instrumento que la produce no puede ser escuchado.

4.1 Diseño del ecualizador

Después de realizadas las pruebas suficientes, se utilizaron 208 coeficientes para obtener los siguientes resultados de las gráficas que a continuación se presentan.

Para el filtro #1 de pasa bajas se implementó usando una frecuencia de rechazo (W_P) y pasante (W_{St}) de 0.0525 a 0.0725 respectivamente. A continuación se muestran los valores reales de los coeficientes en la figura 9 y la frecuencia en la figura 10.

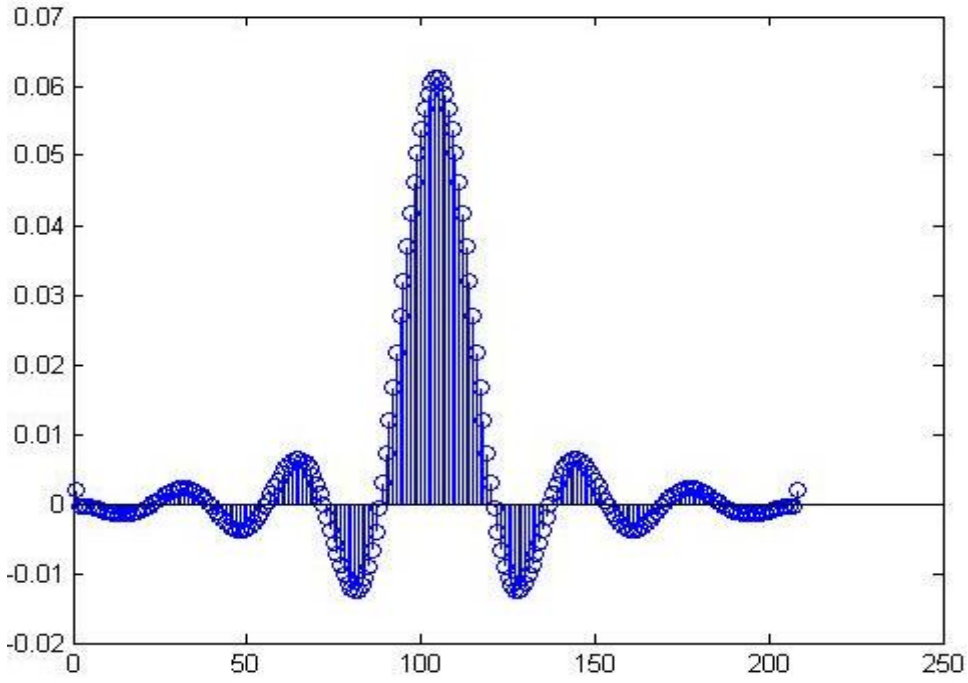

Figura 9. Valores de los coeficientes del filtro pasa bajas.

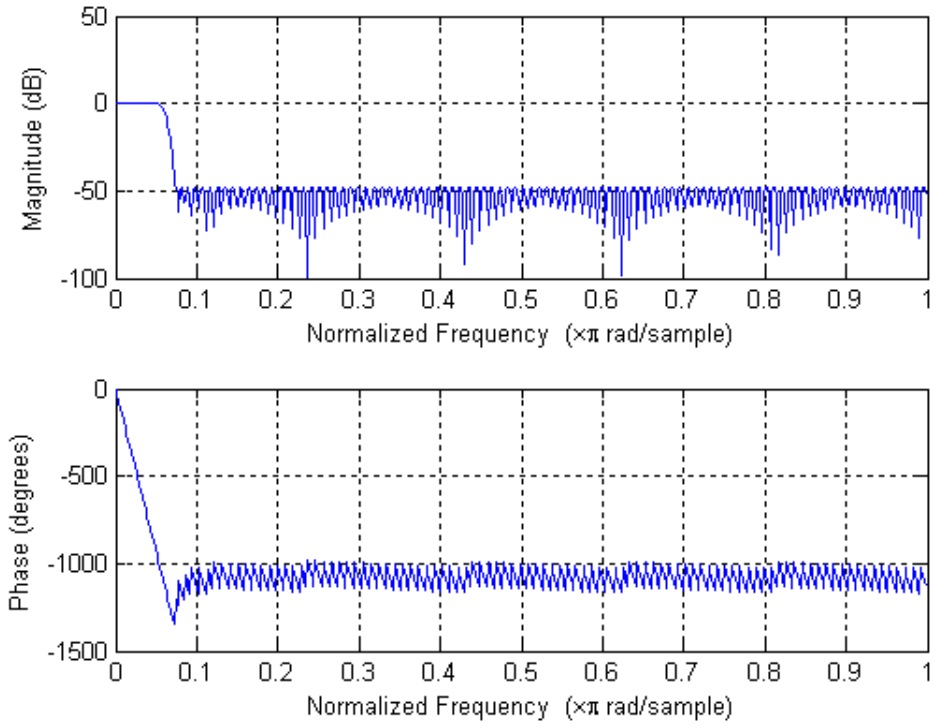

Figura 10. Frecuencia del filtro en un pasa bajas.

Para el filtro #2 de pasa bandas requieren que se implementen dos frecuencias pasante y de rechazo las cuales fueron: 0.0525 (W_{St}), 0.0725 (W_P), 0.115 (W_P) y 0.135 (W_{St}) mostrando los valores reales de los coeficientes en la figura 11 y la frecuencia en la figura 12.

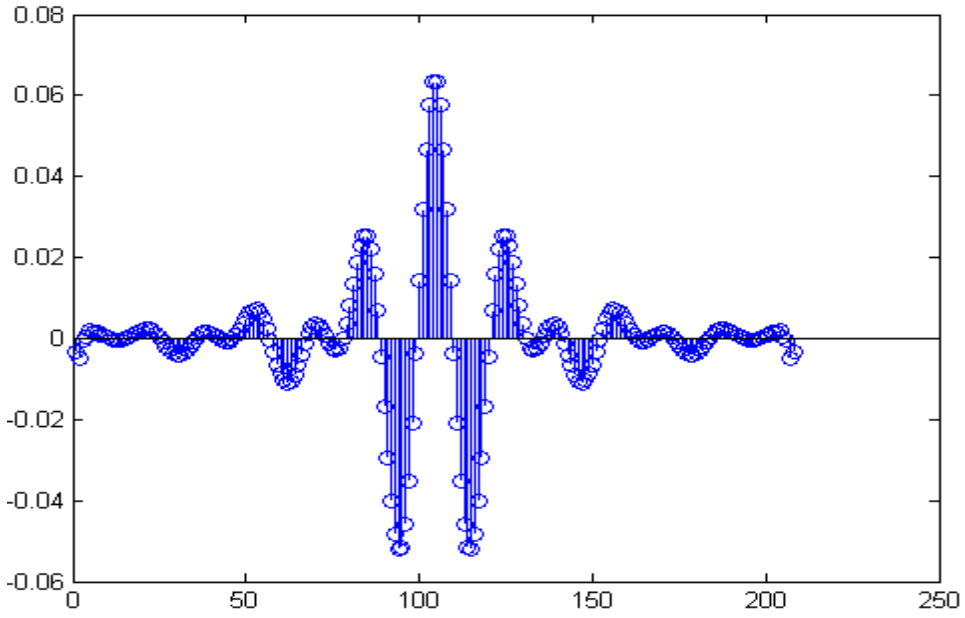

Figura 11. Valores reales de los coeficientes del filtro en un pasa bandas.

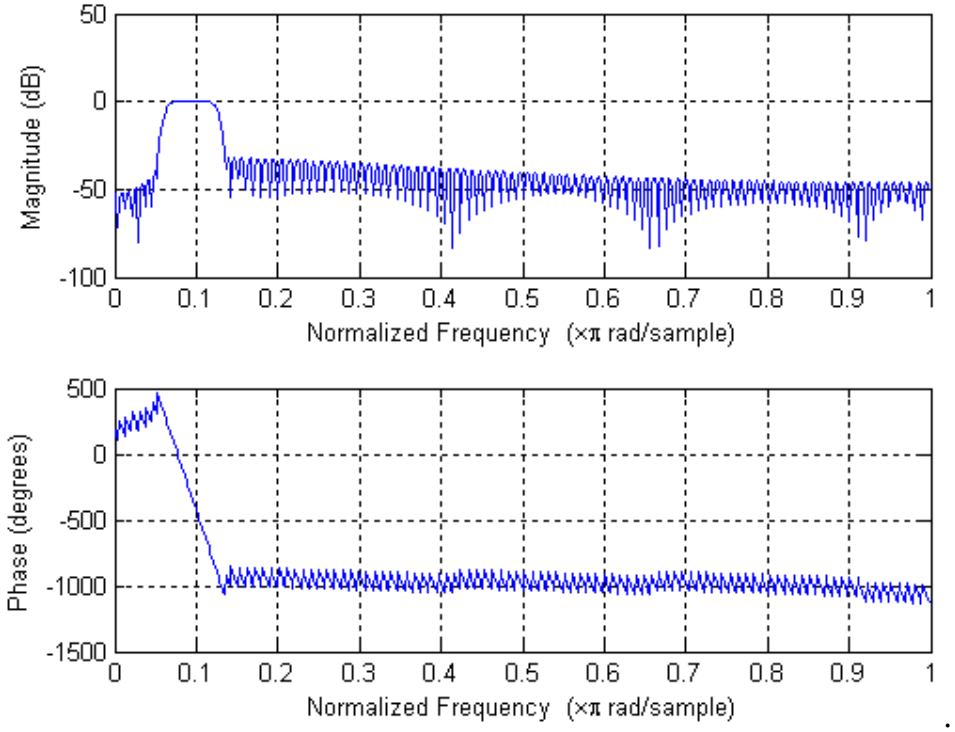

Figura 12. Frecuencia del filtro en un pasa bandas.

Para el filtro #3 de pasa bandas se implementaron las frecuencias de 0.115 (W_{St}), 0.135 (W_P), 0.24 (W_P) y 0.26 (W_{St}) mostrando los valores reales de los coeficientes en la figura 13 y la frecuencia en la figura 14.

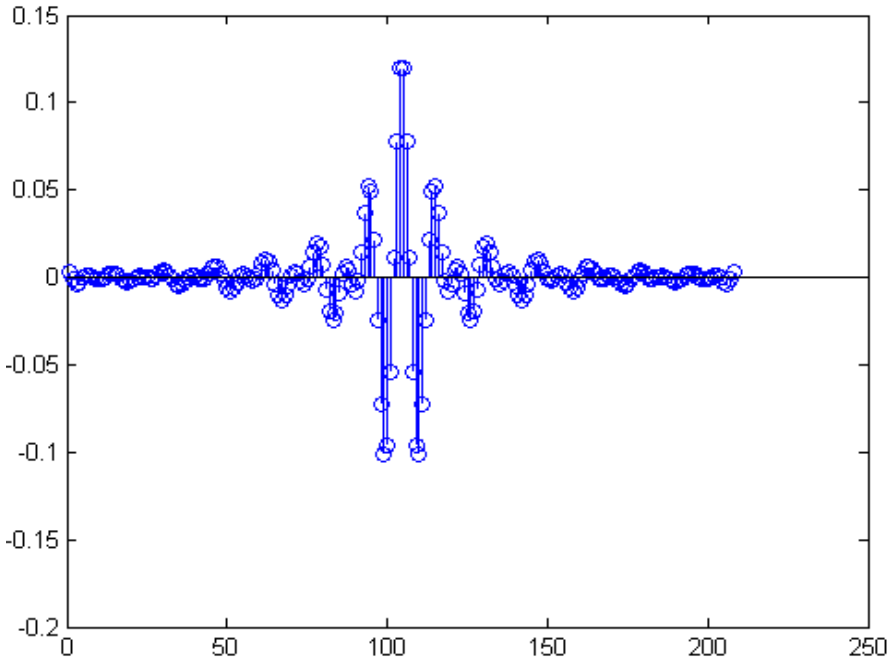

Figura 13. Valores reales de los coeficientes del filtro en un pasa bandas.

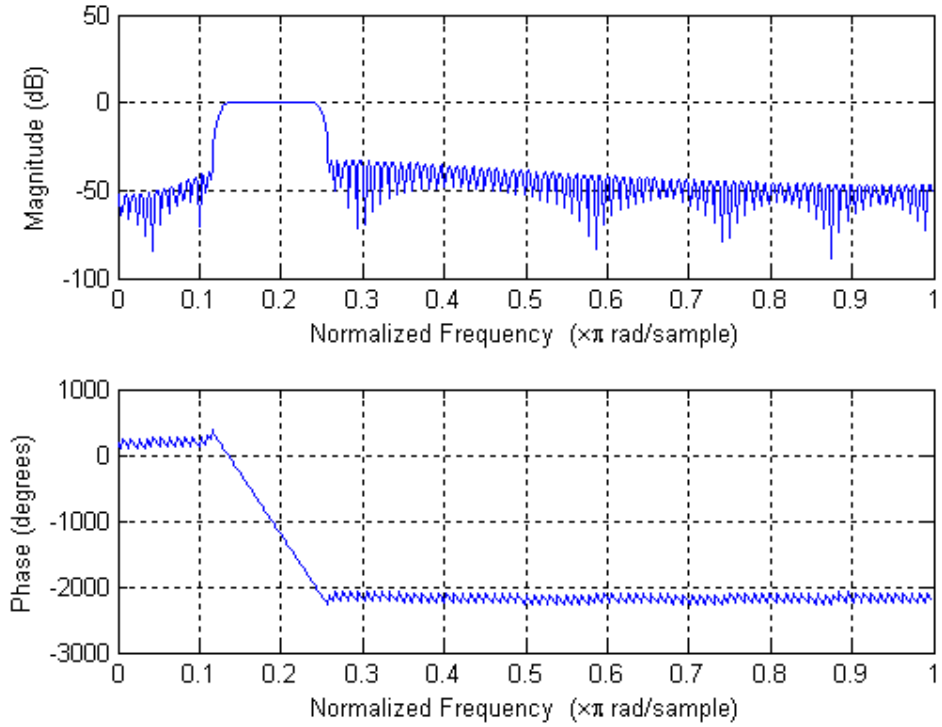

Figura 14. Frecuencia del filtro en un pasa bandas.

Para el filtro #4 de pasa bandas se implementaron las frecuencias de 0.24 (W_{St}), 0.26 (W_P), 0.49 (W_P) y 0.51 (W_{St}) mostrando los valores reales de los coeficientes en la figura 15 y la frecuencia en la figura 16.

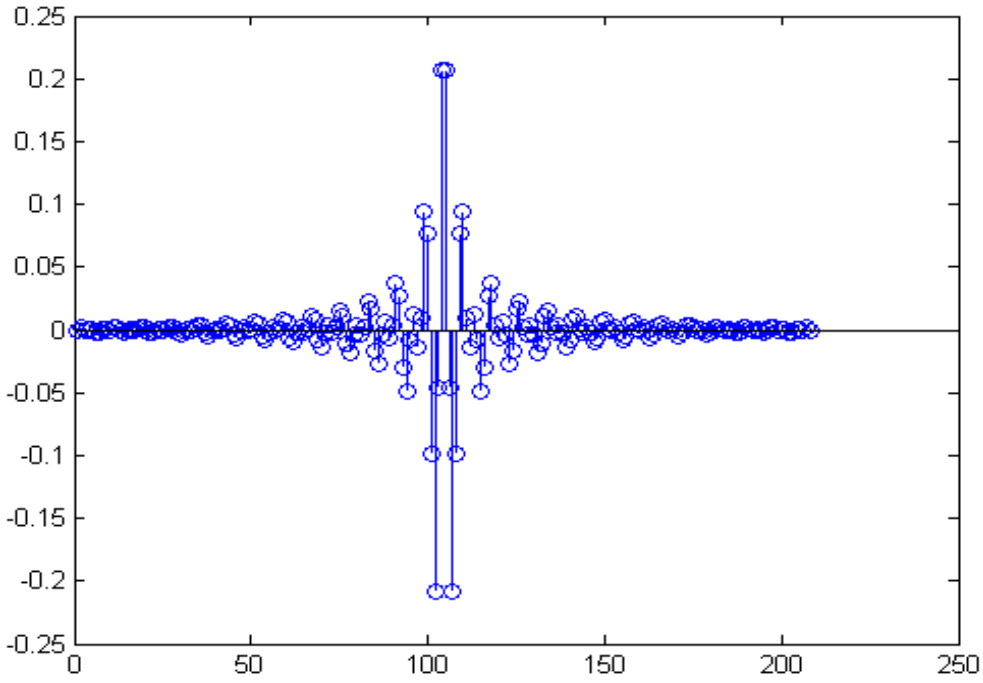

Figura 15. Valores reales de los coeficientes del filtro en un pasa bandas.

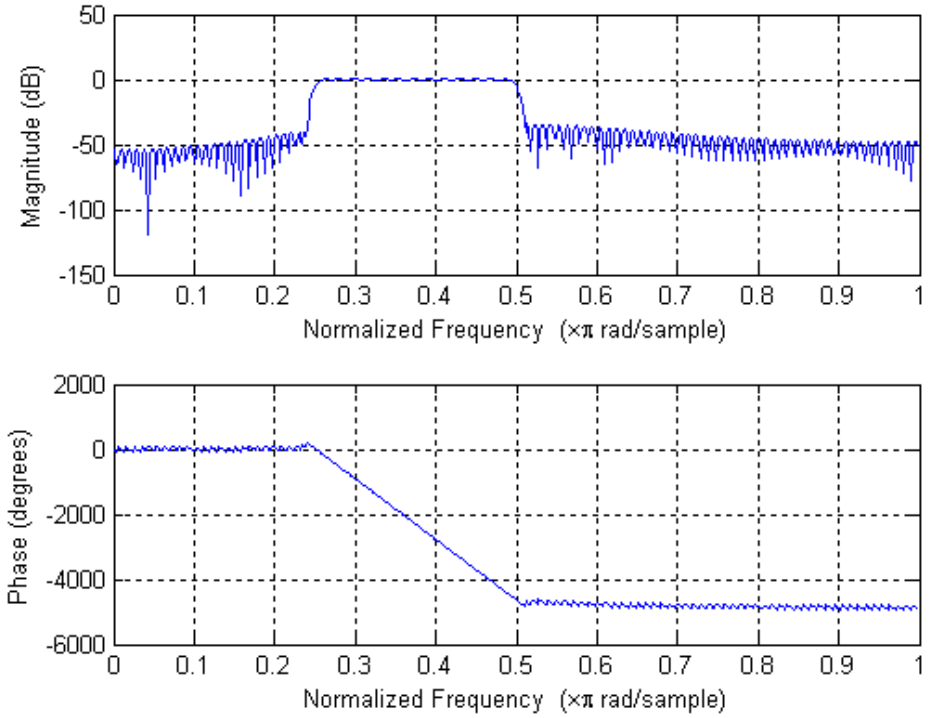

Figura 16. Frecuencia del filtro en un pasa bandas.

Para el Filtro #5 de pasa altas se utilizaron las frecuencias de 0.49 (W_{St}) y 0.51 (W_P) mostrando los valores reales de los coeficientes en la figura 17 y la frecuencia en la figura 18.

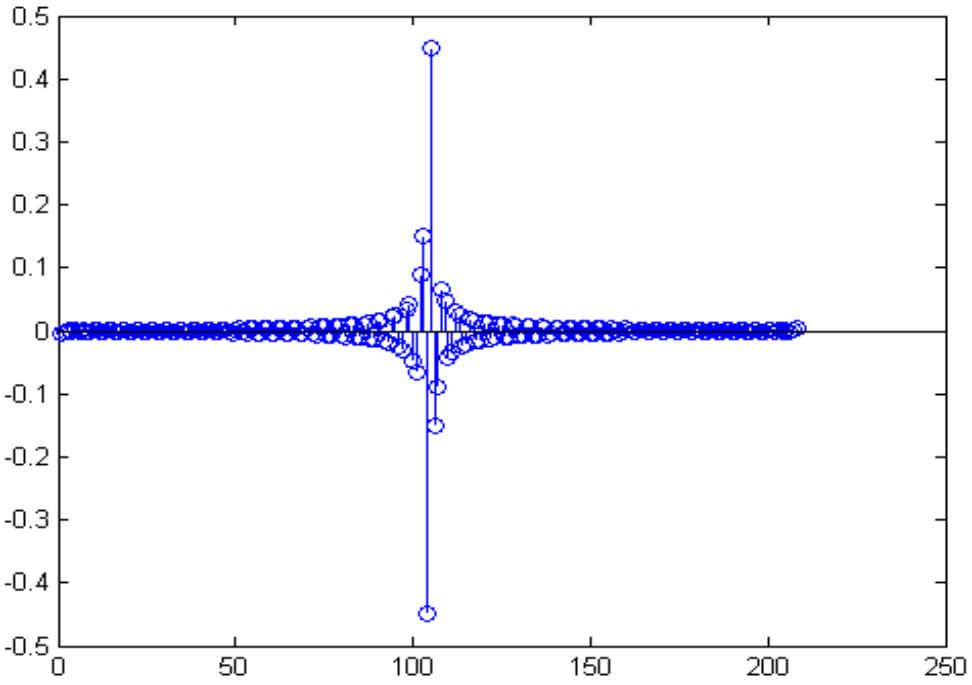

Figura 17. Valores reales de los coeficientes del filtro en un pasa altas.

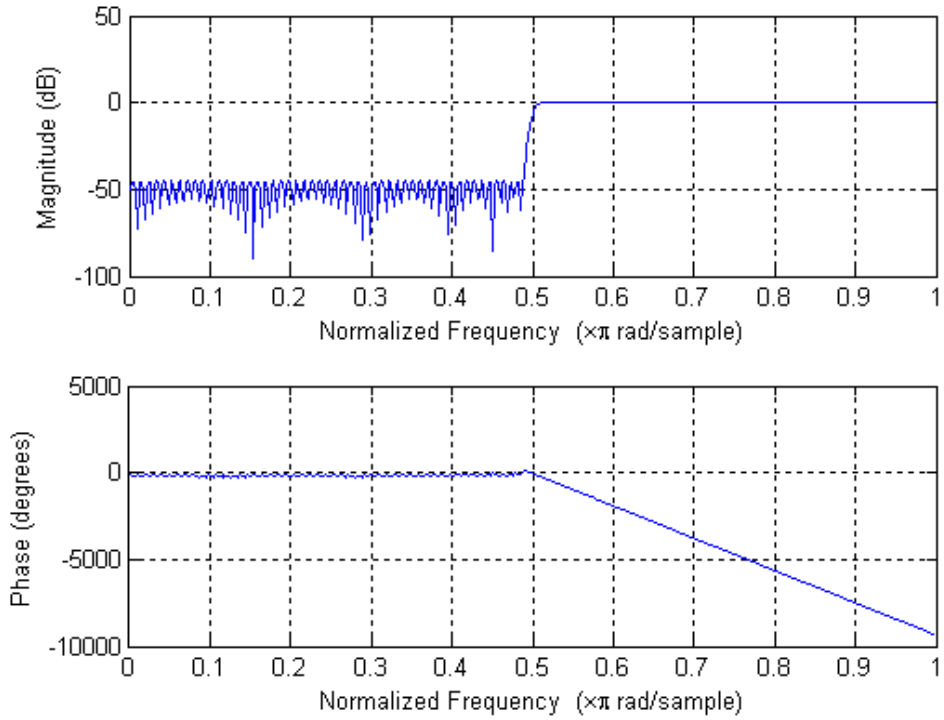

Figura 18. Frecuencia del filtro en un pasa altas.

Finalmente en la siguiente gráfica se presenta los resultados de la suma de todos los filtros usados del 1 al 5 mostrando en la figura 19 la frecuencia.

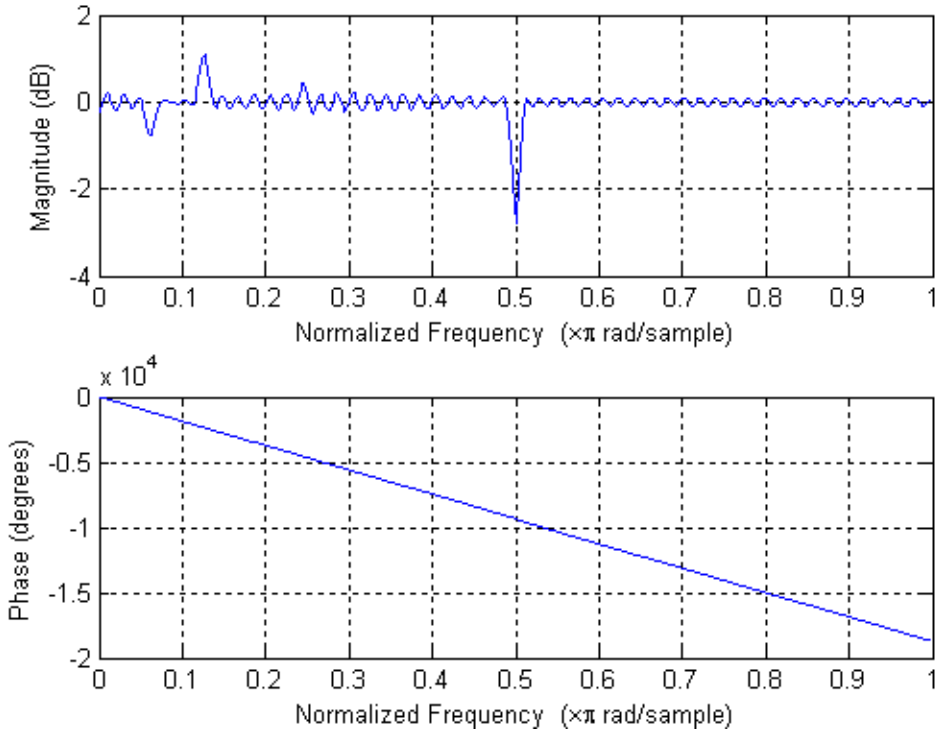

Figura 19. Suma de frecuencias de los filtros.

A continuación se presentan las gráficas de los resultados en la aplicación de los filtros, del lado izquierdo se encuentra un segmento de la señal sin filtro y en seguida, se muestra el mismo segmento de música ya procesado al cual se le aplicó el filtro. En estas gráficas se contemplan como parámetros: la posición en la muestra de 1024 bits y el la magnitud del sonido, dada en milésimas de decibel.

En la figura 20 se puede apreciar como utilizando este primer filtro de pasa bajas como se atenúan las frecuencias altas, permitiendo así el paso a las frecuencias bajas.

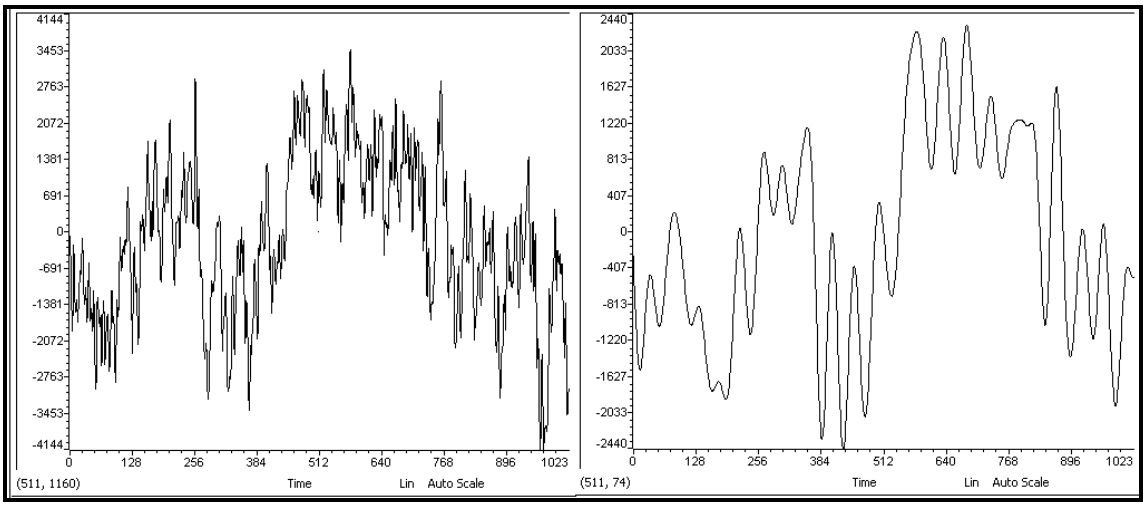

Figura 20. Aplicación del filtro 1 pasa bajas.

Al utilizar este segundo filtro de pasa bandas se puede observar cómo en la figura 21 se toma un rango de frecuencias el cual se deja pasar libremente y atenuando todas las demás.

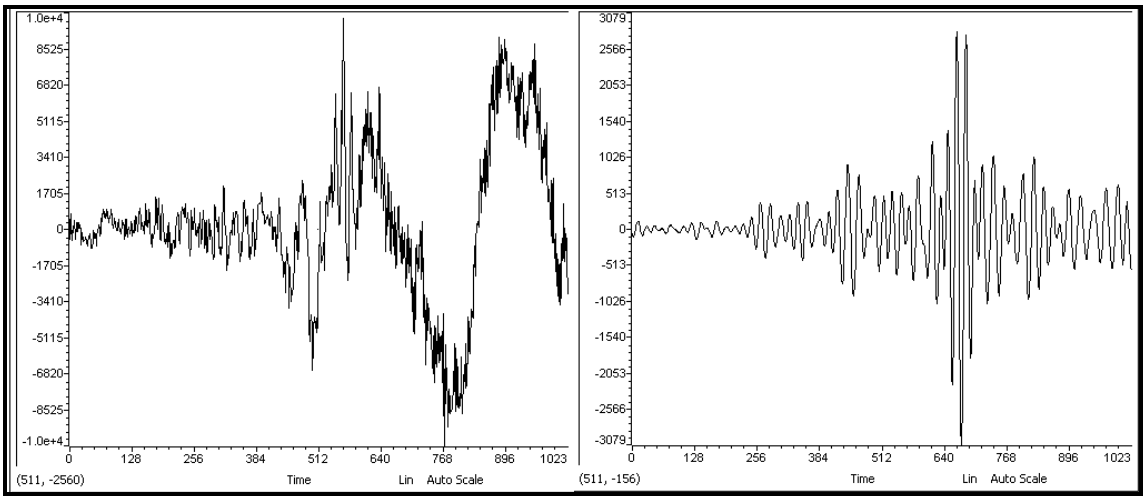

Figura 21. Aplicación del filtro 2 pasa bandas.

En el tercer filtro de pasa bandas se puede observar cómo se toma un distinto rango de frecuencias y como el filtro atenúa las frecuencias fuera de este rango, véase en la figura 22. Aunque la gráfica muestra un comportamiento errático o deficiente se debe de tomar en cuenta la magnitud de frecuencia de la señal procesada, mientras que la frecuencia de la señal original tiene una magnitud máxima de 3048 milésimas de decibel, la frecuencia de la señal procesada solo alcanza una magnitud de 992 milésimas de decibel.

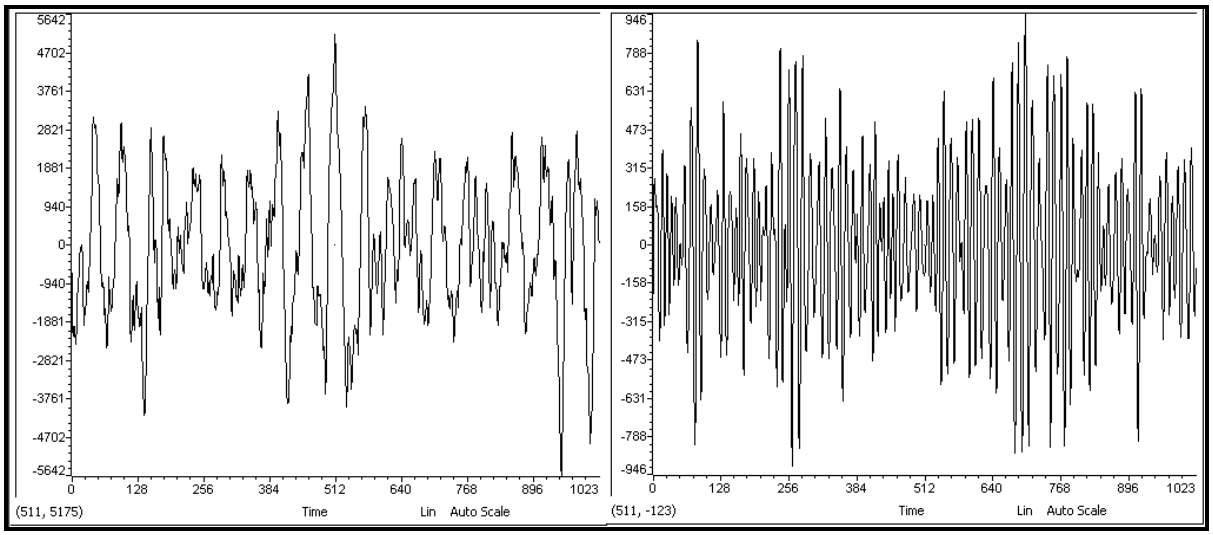

Figura 22. Aplicación del filtro 3 pasa bandas.

El cuarto filtro de pasa bandas que se muestra en la figura 23, deja pasar un diferente rango de frecuencias (4169 – 1840Hz) y atenúa a las demás frecuencias.

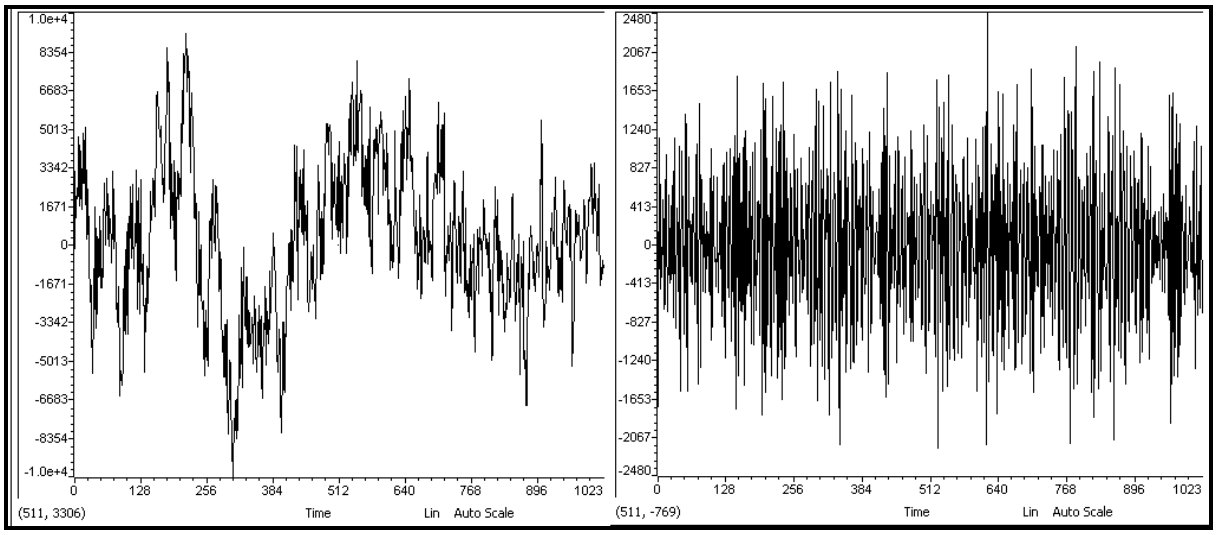

Figura 23. Aplicación del filtro 4 pasa bandas.

Utilizando este quinto filtro, un pasa altas, que atenúa las frecuencias bajas permitiendo así el paso a las frecuencias altas. Es decir, todas aquellas mayores a 8160 Hz, esto se puede observar en la figura 24.

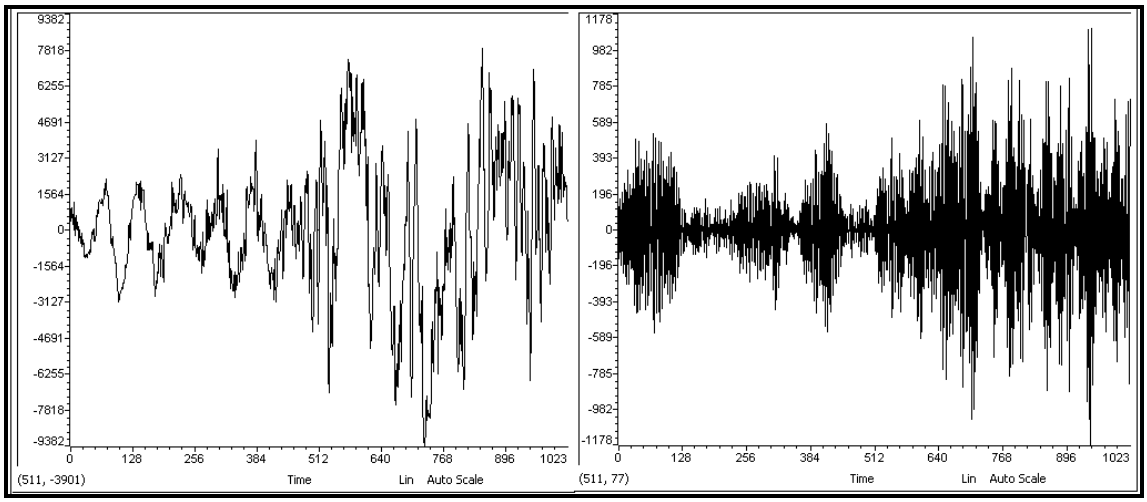

Figura 24. Aplicación del filtro 5 pasa altas.

Por último, en la figura 25 se muestra una gráfica que presenta la suma de todos los filtros. Al sumar filtros que ocupen la mayor parte del ancho de banda se produce un filtro pasa todo y como resultado que se debe obtener de la señal procesada es que tiene que ser una señal similar o igual a la señal original.

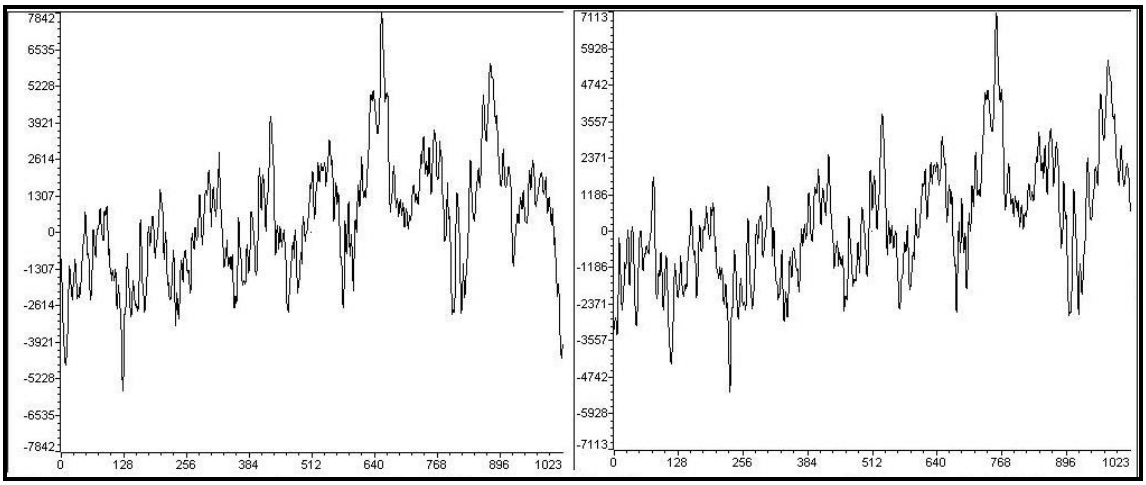

Figura 25. Suma de todos los filtros utilizados.

Capítulo 5. Discusiones y Conclusiones

El problema original en este proyecto fue el desconocimiento que tienen los egresados de nivel licenciatura en esta área importante en la tecnología, concluyendo así sus estudios sin si quiera saber o haber escuchado acerca de la ecualización de música en tiempo real.

Para la realización de este proyecto, se empezó a leer e investigar desde los temas más sencillos como el cuidado y manejo de la tarjeta, sus componentes e inclusive la compatibilidad en algunos de los sistemas operativos; hasta los temas más complejos como entender el código con el que se trabajaba en la tarjeta, aprender a utilizar el *Code Composer Studio*, conocer algunas funciones de *Matlab* para poder trabajar con los coeficientes que fueran necesarios y obtener un filtro óptimo para la realización de las pruebas.

Los resultados fueron favorables, reafirmamos que el principal problema de trabajar en tiempo real es la limitación al código, ya que si se extendía mucho el código presentaba un retardo para procesar la señal. Las ventajas que tienen estos algoritmos a diferencia de los que no funcionan en tiempo real, es que para tener una buena comunicación por cualquier medio, lo ideal es que debe llegar tal cual se vaya emitiendo la señal. Ahorrando el mayor tiempo posible en la llegada de la señal es necesario y de gran conveniencia para la comunicación y de un mayor interés en casos de emergencia o en trabajos donde la comunicación debe ser fluida. Las ventajas de implementar este tipo de tecnología son propicias para los medios de comunicación en los cuales hoy en día y a futuro serán siempre provechosos ya sea en la oficina, trabajo a distancia, comunicación con familiares o amistades lejanas, casos de urgencia y programas de interés en medios de comunicación masivos. El efecto que produce el filtro digital a una señal resulta en la atenuación de una cierta frecuencia haciendo que esta sea casi inaudible para el oído humano, sin embargo este filtro no elimina la dicha frecuencia.

La única desventaja que presentó la tarjeta DSK320C5510 fue que no se pudo crear un ecualizador completo debido a que solo se cuentan con 4 DIP *switches* para controlar la aplicación, por tal motivo no se podía activar más de un filtro a la vez, ni incrementar o reducir el grado de atenuación del filtro. Para lograr esto, se podría cambiar a una tarjeta

con mayores capacidades de hardware o agregar más dispositivos de control por medio de alguno de los puertos de expansión integrados.

En conclusión, trabajando con el filtro FIR se logró observar las diferentes frecuencias obtenidas, percibiendo como se atenúan las señales a diferentes rangos al ir intercambiando los filtros, quedando así que un filtro pasa bajas atenúa las frecuencias altas, el filtro pasa bandas solo selecciona un cierto número de frecuencias y atenúa a las demás, y el filtro pasa altas atenúa a las frecuencias bajas.

Recomendaciones y Trabajo a Futuro

Realizadas las pruebas utilizando el Filtro FIR, se observó, documentó y capturaron las gráficas de los resultados obtenidos, quedando así una oportunidad para realizar otro proyecto utilizando este filtro, ya que existe la posibilidad de incrementar la investigación añadiendo un cancelador de eco, para así obtener resultados fructíferos en la ecualización de música.

A continuación se presentan las siguientes recomendaciones:

- Seguir trabajando con el sistema operativo de *Windows XP,* por resultar de gran compatibilidad con la tarjeta. Por lo mismo que no pide tantos requerimientos de hardware como sucede con *Windows Vista* y *Windows 7*.
- Tener en buenas condiciones la toma de corriente, ya que al desconectar la tarjeta, no te permite volverla a conectar hasta que reinicies todo el equipo de nuevo. Y aquí la pérdida de tiempo es mayor, por tener que cerrar y volver a abrir los otros programas como el *Matlab* o documentos abiertos.
- El *Code Composer Studio* tiende a congelarse inesperadamente, por lo que las modificaciones hechas al código se deben estar guardando continuamente. Ya que al no responder el programa, este mismo no guarda cambios realizados. Además el *Code Composer Studio* se congela cuando cortas la comunicación desconectando la entrada de la señal a la tarjeta. A cualquier problema de congelamiento que presente el *Code Composer Studio* se tiene que reiniciar el ordenador, de lo contrario no te permitirá volver a conectar la tarjeta con dicho programa.
- Seguir la secuencia con la numeración binaria que se le agregó al código, esto es para dar un mejor aprovechamiento al tiempo. Dando así una mejor observación al realizar las distintas pruebas utilizando los filtros, y no tener que estar cambiando el código cada vez que se requiera probar un filtro distinto.

Referencias

[1] M. Peresse, K. Djafarian, J. Chaoui, D. Mazzocco and Y. Masse, “Enabling JPEG2000 on 3G wireless mobiles through OMAP™ architecture”, in *IEEE International Conference in Acoustics, Speech, and Signal Processing*, vol. 4, pp. 27-30, 1993.

[2] A. Wang, R. Hezar and W. Gass, “A low power implementation of a W-CDMA receiver on an ultra low power DSP”, in *IEEE Global Telecommunications Conference*, vol. 1, pp. 241-244, 2000.

[3] T. Schieder, C. Wilks, T. Rontzek and R. Eckmiller, “Towards a tunable tactile communication system: concept and first experiments”, in *Proceedings of the 12th IEEE Workshop in Neural Networks for Signal Processing*, vol. 12, pp. 767-776, 2002.

[4] Byeong-Doo Choi, Kang-Sun Choi, Sung-Jea Ko and A. Morales, “Efficient real-time implementation of MPEG-4 audiovisual decoder using DSP and RISC chips”, in *IEEE International Conference in Consumer Electronics*, pp. 246-247, June 2003.

[5] Jeong Hoo Lee, Weon Heum Park, Jong Ha Moon, and M. Sunwoo, “Efficient DSP architecture for Viterbi decoding with small trace back latency”, in *IEEE Asia-Pacific Conference in Circuits and Systems,* vol. 1, pp. 129-32, December 2004.

[6] Y. Jung, “Design and Optimization of a Programmable Instruction Decoder for DSP Architecture”, in *IEEE Workshop in Signal Processing Systems Design and Implementation*, pp. 333-338, October 2006.

[7] L. Girija y A.S Spanias, “Analysis of the Matrix Processing (MxP) Architecture”, in *IEEE International Conference in Acoustics, Speech and Signal Processing*, vol. 2, pp. 377-380, April 2007.

[8] J.F. An and Vie-Jier Jung, “Implementation of MIMO Channel Simulator for SUI Channel Model Applications”, in *the 18th IEEE International Symposium in Personal, Indoor and Mobile Radio Communications*, pp. 1-5, September 2007.

[9] G. Gammie, et al., “A 45 nm 3.5G Baseband-and-Multimedia Application Processor using Adaptive Body-Bias and Ultra-Low-Power Techniques”, in *IEEE International Digest of Technical Papers in Solid-State Circuits Conference*, pp. 258-611, February 2008.

[10] T. Chattopadhyay, “Enhancements of H.264 Encoder Performance Using Platform Specific Optimizations in Low Cost Dsp Platforms”, in *Second International Conference in Computational Intelligence, Communication Systems and Networks*, pp. 285-288, July 2010.

[11] M. L. Facal, *La Voz Del Cantante*, 1era. ed. Buenos Aires, Argentina: Akadia Editorial, 2005.

[12] M. Recuero, *Ingeniería Acústica*, Madrid, España: Editorial Paraninfo, 2005.

[13] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, 3rd. ed. Santa Barbara, CA: McGraw-Hill, 2006.

[14] Enciclopedia Electrónica Duiops, “Mono, monofónico y monaural”, diciembre 2007. http://www.duiops.net/hifi/enciclopedia/mono.htm

[15] M. Romero, “Introducción al Audio Digital”, Carrera de Composición, Facultad de Bellas Artes, UNLP (s/f).

[16] K. C. Pohlmann., *Principios de Audio Digital*, 4ta ed. Madrid, España: McGraw-Hill, 2005.

[17] H. Ochoa, “Procesadores Digitales de Señales II”, apuntes de clase, Departamento de Ingeniería Eléctrica y Computación, Universidad Autónoma de Ciudad Juárez, enero 2010.

[18] Digital Spectrum Incorporated, *TMS320VC5510 DSK Technical Reference*, DSP Development Systems, 2002.

[19] Texas Instruments, “Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v4.x”, (s/f). [Online]. Available: http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html?DCMP=dsp_ccs_v4&HQS=Other+OT+ccs

[20] S.M. Kuo y B.H. Lee, “*Real-Time Digital Signal Processing*”, 1st. edition, John Wiley & Sons Ltd, 2001.

[21] Digital Signal Processors & ARM Microprocessors, Texas Instruments, “IDEs including CCStudio”, (s/f). [Online]. Available: http://focus.ti.com/dsp/docs/dspsupportatn.tsp?familyId=44&sectionId=3&tabId=415&toolTypeId=30#debugger

[22] R. Oshana, *DSP Software Development Techniques for Embedded and Real-Time Systems.* Burlington, MA Estados Unidos: Elsevier, 2006.

[23] C. Pinilla, A. Alcalá y F.J. Ariza, "Filtrado de imágenes en el dominio de la frecuencia", Universidad de Jaén, Revista de Teledetección, no. 8, pp 1-5, diciembre 1997.

[24] Centro de Investigación en Tecnologías de Audio, Pontificia Universidad Católica de Chile "Filtros FIR", 2008. [Online]. Disponible: http://www.rodrigocadiz.com/imc/html/Filtros_FIR.html

Anexos

Anexo 1. Funciones y Coeficientes de *Matlab*.

Filtro #1 Pasa bajas.

```
N = 207;                                % Filter order
F = [0 0.0525 0.0725 1];                % Frequency vector
A = [1 1 0 0];                          % Magnitude vector
W = [1 5];                              % Weight vector
[b,err,res]=firgr(N,F,A,W);
hfvt = fvtool(b);
stem(b)
freqz(b)
x = fix(b.*30000)
x =
```

58	-8	-9	-12	-15	-18	-22	-27	-31	-35	-39	-43
-45	-46	-46	-45	-43	-38	-33	-26	-18	-8	1	11
22	32	42	50	58	63	66	67	65	61	53	44
31	17	1	-15	-32	-50	-66	-81	-94	-104	-111	-114
-113	-107	-96	-81	-62	-39	-13	14	43	73	102	129
153	172	186	194	195	188	173	151	121	84	40	-7
-59	-112	-166	-218	-266	-307	-340	-363	-374	-371	-353	-320
-269	-202	-119	-19	93	220	357	503	654	808	961	1111
1254	1388	1508	1613	1700	1767	1812	1835	1835	1812	1767	1700
1613	1508	1388	1254	1111	961	808	654	503	357	220	93
-19	-119	-202	-269	-320	-353	-371	-374	-363	-340	-307	-266
-218	-166	-112	-59	-7	40	84	121	151	173	188	195
194	186	172	153	129	102	73	43	14	-13	-39	-62
-81	-96	-107	-113	-114	-111	-104	-94	-81	-66	-50	-32
-15	1	17	31	44	53	61	65	67	66	63	58
50	42	32	22	11	1	-8	-18	-26	-33	-38	-43
-45	-46	-46	-45	-43	-39	-35	-31	-27	-22	-18	-15
-12	-9	-8	58								

Filtro #2 Pasa bandas.

```
N = 207;                                % Filter order
F = [0 0.0525 0.0725 0.115 0.135 1];    % Frequency vector
fresp = {@taperedresp, [0 0 1 1 0 0]};  % Frequency response function
W = [2 2 1];                            % Weight vector
[b,err,res]=firgr(N,F,fresp,W);
hfvt = fvtool(b);
stem(b)
freqz(b)
x = fix(b.*30000)
x =
```

-94	-152	-19	8	54	44	45	29	21	8	0	-7
-9	-8	-2	8	23	39	54	65	70	67	55	34
7	-24	-57	-86	-107	-118	-118	-106	-83	-54	-23	6
30	44	48	41	26	8	-9	-19	-19	-6	20	59
105	151	189	213	215	193	145	74	-11	-103	-191	-264
-312	-330	-314	-269	-200	-118	-34	37	87	110	103	70
20	-32	-74	-90	-68	-2	105	248	409	565	692	763
759	665	477	203	-135	-509	-879	-1204	-1444	-1565	-1543	-1371
-1057	-624	-111	433	956	1404	1731	1904	1904	1731	1404	956
433	-111	-624	-1057	-1371	-1543	-1565	-1444	-1204	-879	-509	-135
203	477	665	759	763	692	565	409	248	105	-2	-68
-90	-74	-32	20	70	103	110	87	37	-34	-118	-200
-269	-314	-330	-312	-264	-191	-103	-11	74	145	193	215
213	189	151	105	59	20	-6	-19	-19	-9	8	26
41	48	44	30	6	-23	-54	-83	-106	-118	-118	-107
-86	-57	-24	7	34	55	67	70	65	54	39	23
8	-2	-8	-9	-7	0	8	21	29	45	44	54
8	-19	-152	-94								

Filtro #3 Pasa bandas.

```
N = 207;                                % Filter order
F = [0 0.115 0.135 0.24 0.26 1];        % Frequency vector
fresp = {@taperedresp, [0 0 1 1 0 0]};  % Frequency response function
W = [2 2 1];                            % Weight vector
[b,err,res]=firgr(N,F,fresp,W);
hfvt = fvtool(b);
stem(b)
freqz(b)
x = fix(b.*30000)
x =
```

89	-33	-112	-78	-3	42	41	13	-17	-29	-16	16
53	74	64	25	-27	-71	-86	-65	-24	14	29	16
-11	-28	-15	28	83	118	106	45	-41	-115	-142	-112
-46	16	43	25	-16	-41	-19	52	140	194	173	72
-69	-188	-232	-185	-80	20	64	37	-26	-63	-24	93
236	322	284	117	-116	-314	-387	-311	-139	28	101	57
-46	-105	-33	175	430	584	518	214	-219	-595	-745	-611
-282	48	201	113	-106	-235	-67	443	1110	1574	1473	646
-728	-2164	-3041	-2892	-1633	352	2348	3589	3589	2348	352	-1633
-2892	-3041	-2164	-728	646	1473	1574	1110	443	-67	-235	-106
113	201	48	-282	-611	-745	-595	-219	214	518	584	430
175	-33	-105	-46	57	101	28	-139	-311	-387	-314	-116
117	284	322	236	93	-24	-63	-26	37	64	20	-80
-185	-232	-188	-69	72	173	194	140	52	-19	-41	-16
25	43	16	-46	-112	-142	-115	-41	45	106	118	83
28	-15	-28	-11	16	29	14	-24	-65	-86	-71	-27
25	64	74	53	16	-16	-29	-17	13	41	42	-3
-78	-112	-33	89								

Filtro #4 Pasa bandas.

```
N = 207;                                % Filter order
F = [0 0.24 0.26 0.49 0.51 1];          % Frequency vector
fresp = {@taperedresp, [0 0 1 1 0 0]};  % Frequency response function
W = [2 2 1];                            % Weight vector
[b,err,res]=firgr(N,F,fresp,W);
hfvt = fvtool(b);
stem(b)
freqz(b)
x = fix(b.*30000)
x =
```

-6	69	-6	4	19	-57	-48	26	13	-11	44	58
-31	-76	-14	23	-9	0	67	57	-46	-86	-16	23
-10	6	85	69	-58	-109	-23	28	-12	10	111	89
-76	-141	-29	35	-16	14	146	115	-101	-182	-36	45
-24	19	191	148	-133	-234	-44	57	-35	26	252	192
-177	-304	-54	73	-49	35	336	252	-238	-402	-68	95
-71	50	461	341	-332	-552	-89	130	-107	75	671	494
-496	-821	-128	195	-178	127	1114	829	-877	-1477	-229	375
-394	303	2807	2319	-2956	-6239	-1393	6210	6210	-1393	-6239	-2956
2319	2807	303	-394	375	-229	-1477	-877	829	1114	127	-178
195	-128	-821	-496	494	671	75	-107	130	-89	-552	-332
341	461	50	-71	95	-68	-402	-238	252	336	35	-49
73	-54	-304	-177	192	252	26	-35	57	-44	-234	-133
148	191	19	-24	45	-36	-182	-101	115	146	14	-16
35	-29	-141	-76	89	111	10	-12	28	-23	-109	-58
69	85	6	-10	23	-16	-86	-46	57	67	0	-9
23	-14	-76	-31	58	44	-11	13	26	-48	-57	19
4	-6	69	-6								

Filtro #5 Pasa altas.

```
N = 207;                                % Filter order
F = [0 0.49 0.51 1];                    % Frequency vector
A = [0 0 1 1];                          % Magnitude vector
W = [2 1];                              % Weight vector
[b,err,res]=firgr(N,F,A,W,'4');
hfvt = fvtool(b);
stem(b)
freqz(b)
x = fix(b.*30000)
x =
```

-101	-2	67	-64	11	-5	42	-34	-12	9	34	-26
-24	17	34	-25	-31	23	37	-28	-38	28	41	-32
-44	33	47	-36	-50	39	53	-42	-57	45	60	-48
-64	52	68	-56	-73	60	77	-64	-82	69	87	-73
-92	79	98	-84	-104	90	111	-97	-118	104	125	-111
-133	119	142	-128	-152	138	162	-149	-174	161	187	-174
-201	189	217	-205	-235	225	256	-247	-280	273	308	-303
-342	340	383	-386	-434	444	500	-520	-589	624	714	-777
-905	1023	1234	-1486	-1937	2688	4511	-13493	13493	-4511	-2688	1937
1486	-1234	-1023	905	777	-714	-624	589	520	-500	-444	434
386	-383	-340	342	303	-308	-273	280	247	-256	-225	235
205	-217	-189	201	174	-187	-161	174	149	-162	-138	152
128	-142	-119	133	111	-125	-104	118	97	-111	-90	104
84	-98	-79	92	73	-87	-69	82	64	-77	-60	73
56	-68	-52	64	48	-60	-45	57	42	-53	-39	50
36	-47	-33	44	32	-41	-28	38	28	-37	-23	31
25	-34	-17	24	26	-34	-9	12	34	-42	5	-11
64	-67	2	101								

Filtro #6 Pasa todo.

x = b1 + b2 + b3 + b4 + b5

x =

58	-8	-9	-12	-15	-18	-22	-27	-31	-35	-39	-43
-45	-46	-46	-45	-43	-38	-33	-26	-18	-8	1	11
22	32	42	50	58	63	66	67	65	61	53	44
31	17	1	-15	-32	-50	-66	-81	-94	-104	-111	-114
-113	-107	-96	-81	-62	-39	-13	14	43	73	102	129
153	172	186	194	195	188	173	151	121	84	40	-7
-59	-112	-166	-218	-266	-307	-340	-363	-374	-371	-353	-320
-269	-202	-119	-19	93	220	357	503	654	808	961	1111
1254	1388	1508	1613	1700	1767	1812	1835	1835	1812	1767	1700
1613	1508	1388	1254	1111	961	808	654	503	357	220	93
-19	-119	-202	-269	-320	-353	-371	-374	-363	-340	-307	-266
-218	-166	-112	-59	-7	40	84	121	151	173	188	195
194	186	172	153	129	102	73	43	14	-13	-39	-62
-81	-96	-107	-113	-144	-111	-104	-94	-81	-66	-50	-32
-15	1	17	31	44	53	61	65	67	66	63	58
50	42	32	22	11	1	-8	-18	-26	-33	-38	-43
-45	-46	-46	-45	-43	-39	-35	-31	-27	-22	-18	-15
-12	-9	-8	58								

Anexo 2. Función FIR2

A continuación se presenta un fragmento del código fuente donde se emplea la función FIR2.

```
switch(switchf)
        {
          case 0:
                copyData(gBufferRcvPingL, gBufferXmtPingL, BUFFSIZE/2);
                copyData(gBufferRcvPingR, gBufferXmtPingR, BUFFSIZE/2);
           break;
           case 1:
                fir2(gBufferRcvPingL, COEFFS21, gBufferXmtPingL,
                delayBufferL, BUFFSIZE/2, ORDER);

                fir2(gBufferRcvPingR, COEFFS21, gBufferXmtPingR,
                delayBufferR, BUFFSIZE/2, ORDER);
            break;
        }
```