DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PROGRAMABLE, ORIENTADO A LA APLICACIÓN DE ALGORITMOS DE PROCESAMIENTO DIGITAL DE SEÑALES DE AUDIO.

PRESENTADO POR

JEFFERSON ESQUIVEL HINCAPIE

ESTUDIANTE LICENCIATURA ELECTRÓNICA

UNIVERSIDAD PEDAGOGICA NACIONAL
FACULTAD DE CIENCIA Y TECNOLOGÍA
BOGOTA, 2017

Dedicatoria

Primeramente, a Dios por permitir haber alcanzados este logro tan anhelado en mi vida, lleno de sacrificios y sufrimientos que hoy me hacen una mejor persona.

A mi madre, por ser esa mujer que me ha apoyado incondicionalmente durante toda mi vida, siempre aconsejándome en las decisiones más importantes que he tomado, gracias madre.

A mi padre, por ser es hombre que me ha demostrado que con su nobleza se puede conseguir grandes cosas, y sobre todo por creer en que podía llegar a convertirme en la persona que hoy soy

A mi hijo, que a pesar de que la vida en este momento nos tiene separados, daré lo mejor de mí para que siempre me recuerde como ese padre que lo dio todo por querer un mejor futuro, así como mi padre lo dio todo hacia mí.

A mis amigos, personas que me demostraron que se puede llegar a apoyar a un buen compañero de forma incondicional y sobre todo por ser esos hermanos que nunca tuve.

A Merilli, por ser esa mujer que me enseñó que en la vida hay que luchar por conseguir nuestros sueños, sin importar lo que piensen los demás, gracias a ella soy mejor persona.

1. Información General	
Tipo de documento	Trabajo de grado
Acceso al documento	Universidad Pedagógica Nacional. Biblioteca Central
Titulo del documento	Diseño y construcción de un prototipo programable, orientado a la aplicación de algoritmos de procesamiento digital de señales de audio.
Autor(es)	Esquivel Hincapié, Jefferson.
Director	Bonilla Camelo, Julio Giovanny.
Publicación	Bogotá. Universidad Pedagógica Nacional, 2017. 72 p.
Unidad Patrocinante	Universidad Pedagógica Nacional.
Palabras Claves	Tratamiento digital de señales de audio, filtrado digital, prototipo, sistemas microcontrolados.

2. Descripción

Trabajo de grado que propone el desarrollo de un prototipo programable donde se apliquen conceptos adquiridos en las asignaturas que involucren el tratamiento digital de señales.

Inicialmente parte desde una problemática presentada en la Licenciatura en Electrónica, donde se expone la falta de fundamentación en el área filtrado digital, proponiendo como solución la realización del proyecto donde se ponga en práctica la fundamentación teórica. Seguido de esto se expone unos requerimientos fundamentales que limitan el desarrollo del proyecto a trabajar con señales de audio dentro del espectro audible humano (20 Hz a 20 kHz), partiendo desde aquí al desarrollo de cada una de las partes del proyecto. Luego de su desarrollo, se procede a diseñar e implementar algoritmos de filtrado digital basado en arquitecturas de tipo FIR e IIR, donde se muestra de manera clara la metodología de diseño, los resultados y la comparación significativa entre los dos tipos de filtro utilizados.

Finalmente se realiza un breve ejercicio de análisis de los resultados obtenidos a partir de la aplicación del filtrado en tiempo real, relacionado directamente con la exposición de las conclusiones que llevaron a la culminación del proyecto.

3. Fuentes

MITRA, S. Digital Signal Processing, Second Edition, McGraw Hill.

PROAKIS G. J, MANOLAKIS G., D. Digital Signal Processing, Prentice Hall Internacional Inc.

ATMEL CORPORATION. (2013). 8-bit Atmel XMEGA AU Microcontroller XMEGA AU MANUAL.http://www.atmel.com

ATMEL CORPORATION. (2014). 8/16-bit Atmel XMEGA Microcontroller ATxmega128A4U /ATxmega64A4U /ATxmega32A4U / ATxmega16A4U. http://www.atmel.com

ATMEL CORPORATION. (2002). AVR223: Digital Filters with AVR. http://www.atmel.com

SHENOI B. A., Introduction to Digital Signal Processing and Filter Design, Wiley Interscience, 2006.

ATMEL CORPORATION. (2016). AVR201: Using the AVR Hardware Multiplier. APPLICATION NOTE. http://www.atmel.com

ATMEL CORPORATION. (2010). Atmel AVR1300: Using the Atmel AVR XMEGA ADC. http://www.atmel.com

ATMEL CORPORATION. (2016). The Atmel AVR Dragon Debugger. http://www.atmel.com

ATMEL CORPORATION. (2016). Atmel Studio USER GUIDE. http://www.atmel.com

BONILLA CAMELO, J. PEÑA, W. (2010). Diseño y programación de filtros digitales FIR en lenguaje de descripción de hardware HDL e implementación en FPGA. (Tesis de pregrado). Universidad Pedagógica Nacional. Bogotá Colombia.

JARRIN OCHOA, D. (2016). Diseño e implementación de filtros FIR e IIR utilizando el microcontrolador XMEGA de ATMEL para el tratamiento de señales de audio. (Tesis de pregrado). Universidad Politécnica Salesiana. Quito, Ecuador.

MESA HERNANDEZ, E. (2016). Generador de señales periódicas de forma predeterminada empleando un dispositivo chipbas8. (Tesis de pregrado). Universidad Nacional Autónoma de Mexico. Ciudad de México, México.

NATIONAL SEMICONDUCTOR. (1999). LMF100 High Performance Dual Switched Capacitor Filter. http://www.ti.com

OBERSTAR, E. BAUCH, M. (2001). Narrow Band Filter Implementation On A Low Cost Microcontroller Issues and Performance. University of Wisconsin. Madison, USA.

VIVAS GONZALES, E. RIVERA PINZÓN, D. GOMEZ, E. Implementation and Simulation of IIR Digital Filters in FPGA Using MatLab System Generator.

TEXAS INSTRUMENTS. (2017). LM386 Low Voltage Audio Power Amplifier.

TEXAS INSTRUMENTS. (2017). LMx58-N Low-Power, Dual-Operational Amplifiers.

LAND, B. (2014). DSP on 8-bit microcontroller. https://www.hackster.io/bruceland/dsp-on-8-bit-microcontroller-21220c.

JIANG, J. (2014). Teaching Digital Signal Processing Course with a Real-Time Digital Crossover System for Electrical and Computer Engineering Technology Students. Purdue Univesity, North Central, USA.

4. Contenidos

Capítulo 1. Definición del problema de investigación.
Es el primer apartado donde expone las razones fundamentales que se encontraron para determinar la realización del proyecto.
1.4 Objetivos. Se expone el objetivo general del proyecto que es el desarrollo global del prototipo, basado principalmente en los 3 objetivos específicos que sustentan su realización: diseño, construcción e implementación.

Capítulo 2. Aspectos metodológicos.
Es el apartado donde se expone el tipo de metodología que se usó para dar solución al problema de investigación.

Capítulo 3. Desarrollo del prototipo programable orientado a filtrado digital.
En este capítulo se aprecia el cuerpo del proyecto, donde se parte desde los requerimientos generales, pasando por la realización de cada una de las etapas necesarias para la construcción del prototipo y llegando finalmente a la implementación de los algoritmos de filtrado.

Capítulo 4. Análisis de los resultados y conclusiones.
Aquí es donde se realiza el ejercicio de análisis y comparación de los resultados obtenidos, relacionado directamente con la determinación de las conclusiones.

Bibliografía. Fuentes bibliográficas usadas para la elaboración del proyecto.

Anexos. Códigos de programación implementados en el microcontrolador utilizado.

5. Metodología

El tipo de metodología de diseño planteado en este proyecto es llamado TOP DOWN. La metodología TOP DOWN se encarga de estudiar el proceso de diseño desde el propósito más general y global de un sistema, partiendo del macro hasta los detalles más mínimos; es decir, se basa en la división de problema de diseño generales en subproblemas que, a la vez, se dividen en problemas más pequeños que se atiende de manera puntual. Esto garantiza que, en el caso de un prototipo de tipo electrónico el estudio del sistema se pueda dividir en módulos o etapas, cada una con una determinada función que aporta al desarrollo del producto final.

6. Conclusiones

Un sistema basado en arquitectura microcontrolada permite desarrollar la estructura del algoritmo de forma secuencial, permitiendo evidenciar el proceso en forma lineal y observar el flujo directo del programa para encontrar posibles errores y fallas en el algoritmo.

El microcontrolador ATXMEGA32A4U tiene una gran capacidad de procesamiento en cuanto a su sistema de temporizaciones y su hardware multiplicador, posibilitando realizar operaciones de multiplicaciones y acumulación de información de datos en intervalos de tiempo bastante pequeños. El caso particular de los filtros FIR mostró que se pueden realizar hasta 13 operaciones de multiplicación y acumulación en un tiempo menor a 50 us, dando por hecho que es un sistema capacitado para realizar tratamiento de señales no solo en el espectro audible, sino para señales de ultrasonido.

La implementación de filtros de condensador conmutado permite en gran medida reducir el tamaño de los circuitos en cuanto a la optimización de espacio y también respecto a la facilidad de configurarse para diferentes tipos de filtro. Característica muy relevante de su funcionamiento es la flexibilidad con relación a la variación de su frecuencia de corte directamente relacionada con una señal de reloj externa, dándole atributos muy adecuados para aplicaciones de soporte para filtrado digital, haciéndolos superiores a las implementaciones en forma discreta de filtros analógicos.

Los filtros digitales son sistemas que demuestran la gran flexibilidad ante cambios que en los filtros analógicos significaría una reestructuración total en su construcción, y no solo eso presentan una precisión bastante alta debido a que no depende directamente de valores de componentes discretos como los analógicos, si no de frecuencias de muestreo y capacidad de procesamiento, que hoy día se ha solucionado satisfactoriamente, demostrando su gran ventaja a la hora de realizar aplicaciones de tratamiento de señales en tiempo real.

Las señales muestreadas a frecuencias alrededor de 5 a 10 veces su frecuencia superior presentan distorsiones bastante significativas en su forma de onda al momento de la conversión digital a analógica, presentándose claramente una relación inversa entre la frecuencia de muestreo y la distorsión de la señal de salida, es decir que este fenómeno se corrige bastante bien al aumentar de forma significativa la frecuencia de muestreo del sistema.

La diferencia significativa de los filtros IIR respecto a los filtros FIR en cuanto al tamaño y la longitud de sus coeficientes, los hace más digeribles por sistemas con capacidades de procesamiento bajas limitados por velocidades de reloj pequeñas, puesto que se necesitan menor cantidad de iteraciones para efectuar las operaciones de suma y multiplicación; sin embargo, los filtros FIR presentan una mejor respuesta en cuanto a amplitud y fase lineal, haciéndolos más adecuados para aplicaciones donde la calidad de los resultados es más exigente.

La cantidad de coeficientes de un filtro FIR es bastante extensa dependiendo de la frecuencia de muestreo del sistema, esto permite encontrar una relación significativa entre longitud de coeficientes y la velocidad de muestreo, es decir si el sistema necesita tasas de muestreo altas esto impactará directamente en la arquitectura FIR, necesitando más coeficientes para garantizar resultados muy positivos en la señal de salida y por supuesto velocidades de procesamiento mayores.

Debido a que los efectos en fase de los filtros antialias y posfiltrado no son lineales, esto puede afectar un poco el corrimiento en fase de las señales de entrada, puesto que son de cuarto orden y la frecuencia de corte de estos filtros es de 20 kHz las señales anteriores a esta se encuentran en una banda donde el desfase en la señal no afecta significativamente su forma de onda.

Elaborado por:	Esquivel Hincapié, Jefferson.
Revisado por:	Bonilla, Julio.

Fecha de elaboración del Resumen:	20	Febrero	2018

Tabla de contenidos **Página**

Tabla de gráficas **Página**

Capítulo 1. Definición del problema de investigación

1.1 Introducción

El crecimiento y evolución de la electrónica ha permitido la realización de múltiples actividades que facilitan la vida y el trabajo de las personas, convirtiéndose en herramienta indispensable que nos caracteriza como seres tecnológicos. Esta evolución se debe a los requerimientos que conlleva suplir nuestras necesidades como consumidores de tecnología para múltiples usos desde funciones laborales, comunicación y hasta entretenimiento. Estas necesidades requieren de un desempeño bastante alto en los dispositivos electrónicos que se encargan de realizar mediciones de valores analógicos, para ser convertidos posteriormente a valores digitales. De la mano de la electrónica esta la instrumentación, la cual se encarga de la toma de valores como lo mencionamos anteriormente, y es indispensable en el campo del procesamiento digital de señales, como la temperatura, iluminación, video o audio. En el campo del procesamiento del audio, se requiere del funcionamiento de equipos que mezclan electrónica analógica y digital, puesto que los valores en el ambiente cotidiano se encuentran de forma analógica.

Los procesadores de señales más comunes y eficientes con respecto a velocidad y complejidad de funciones son los digitales, debido a las grandes ventajas que ofrecen en cuanto a la inmunidad a interferencia de señales y ruido, la versatilidad a la hora de transmitir la información, la portabilidad de la información y la complejidad de tareas que puede realizar frente a los procesos analógicos. Un sistema de procesamiento digital (como el estudiado particularmente en este trabajo) requiere también de partes que integren procesos analógicos, que en el campo del audio son indispensables, precisamente en la forma en como el ser humano percibe la información del medio.

Órgano responsable de percibir este tipo de señales, o de otra forma transformar la información presente en el medio es el oído, que posee la facultad de captar ondas sonoras de tipo mecánica que se desplazan en el medio, en impulsos eléctricos transmitidos hacia el sistema nervioso central, donde se procesa la información. Dependiendo de la edad, la perturbación, la estimulación y múltiples factores biológicos, el oído tiene un comportamiento que se transmite en una respuesta, y la calidad de esa respuesta denota la capacidad auditiva de cualquier persona. Dicha capacidad auditiva la podemos relacionar directamente con la sensibilidad a tonos de diferente frecuencia, que (según estudios científicos y médicos) va desde los 20 Hz hasta los 20000 Hz, intervalo denominado en otras palabras ancho de banda auditivo humano.

Con base al ancho de banda auditivo, son diseñados sistema de procesamiento de señales de audio, que en algunos casos son llamados ecualizadores, que constan de arreglos de filtros de diferentes tipos como los son: Filtros de paso bajo, filtros de paso alto, filtros de banda de paso y filtros de banda rechazada. Estas bandas son anteriormente mencionadas frecuencias. Procedimiento como este es realizado de forma más efectiva debido a los sistemas digitales, que gracias a su desarrollo es posible diseñar y construir artefactos que modifican las componentes principales de una señal de audio que son en muchos casos dispositivos que ayudan a mejorar la calidad auditiva de personas con limitaciones.

1.2 Planteamiento Del Problema

Analizando el crecimiento y la importancia de la electrónica, y en su defecto la Electrónica digital, es de comprender que estos efectos condicionan a los profesionales que se desempeñan en el área de la electrónica y la programación a una tarea frecuente de fundamentación de sus conceptos y técnicas.

Observando y analizando la aplicabilidad de los sistemas digitales y que el procesamiento digital de señales está en aumento, se observa que la Licenciatura en Electrónica de la Universidad Pedagógica Nacional necesita una profundización en cuanto al estudio y aplicación de temáticas relacionadas, puesto que no se llevan a cabo la realización de prácticas donde se evidencie un ejercicio de contraste con la teoría.

1.3 Justificación

Con el fin de ampliar conocimientos en un campo de estudio como es el tratamiento digital de señales, se pretende diseñar un prototipo aplicado al área del filtrado de señales de audio, que se implemente de forma sencilla y sobre todo que sirva como punto de partida para realizar proyectos de laboratorio que tengan como objetivo analizar y procesar señales en tiempo discreto, con el fin de abordar la problemática presentada anteriormente y dar pautas para poder solucionar la falta de comprobación de teoría en práctica.

La realización de este tipo de proyectos complementa el proceso de formación y desarrollo de habilidades y técnicas requeridas para desempeñarse en una disciplina como es el campo de la electrónica digital. El desenvolvimiento en este campo de acción permite poner a prueba temas relacionados con las diferentes transformadas implementadas en el desarrollo de algoritmos de filtrado, que por su aplicabilidad, son relevantes en diferentes campos de estudio como son el control, las telecomunicaciones, instrumentación e incluso los sistemas de potencia.

La forma de abordar el diseño dispositivos orientados al desarrollo de prácticas relacionadas con el filtrado digital, se puede analizar desde el uso de diferentes arquitecturas y sistemas programados. Uno de los implementados en la licenciatura en Electrónica ha sido los sistemas de arreglos lógicos programables (FPGA), en donde se desarrollaron trabajos de grado enfocados a la aplicación de diferentes Transformadas en tiempo discreto. Estos sistemas ofrecen ventajas con relación al procesamiento de información de forma paralela, pero presentan complicaciones debido su lenguaje de programación. En contraste se encuentran los sistemas microcontrolados donde su arquitectura y lenguajes de programación son más flexibles.

Puesto que en la Universidad se ha abordado poco este tipo trabajos a partir programación en microcontroladores, se hace interesante y pertinente realizar proyectos de grado enfocados al procesamiento digital de señales, a partir de estas arquitecturas. Para complementar el proceso de formación y desarrollo de habilidades, es pertinente mencionar que al diseñar un prototipo programable aplicado al tratamiento digital de señales, donde se pueda elaborar laboratorios en tiempo real, con ayuda de herramientas de simulación, permite que los estudiantes de tecnología sean capaces de fortalecer los conceptos vistos en clase.

1.4 Objetivos

1.4.1 General.

Desarrollar un prototipo programable enfocado a la aplicación de algoritmos de filtrado digital de señales de audio.

1.4.2 Específicos.

- Diseñar el circuito impreso donde se implemente una arquitectura DSP, que cumpla los requerimientos del proyecto.

- Construir el prototipo a partir de los elementos y componentes seleccionados en la etapa de diseño.

- Implementar algoritmos de programación orientado a la aplicación de filtros digitales, con el fin de comprobar el desempeño del sistema a señales de audio.

1.5 Antecedentes

1.5.1 Purdue University, North Central, Indiana EEUU. Teaching Digital Signal Processing Course with a Real-Time Digital Crossover System for Electrical and Computer Engineering Technology Students. (Curso de aprendizaje de procesamiento digital de señales en tiempo real para estudiantes de ingeniería eléctrica e informática).

Este curso estuvo a cargo del Doctor Jean Jiang, profesor de la Universidad de Purdue de

North Central en Indiana en el año 2014. En este hace énfasis en la importancia de optar por una metodología centrada en el estudio de los fenómenos vistos en el tratamiento digital de señales a partir de aplicaciones en tiempo real. Los materiales y las herramientas implementadas fueron un DSP TMS320C6713 DSK y el software de MATLAB. Menciona que las ventajas de usar Matlab como herramienta de diseño y simulación son bastante buenas, puesto que se prueba los diseños de forma óptima antes de implementarlo en las tarjetas de adquisición de datos.

Al finalizar el curso realizaron una encuesta donde se estudiaban los logros y las expectativas de curso en comparación de cursos anteriormente ofrecidos. Los resultados arrojados fueron clasificados en tres categorías con su respectivo porcentaje: 1) Excelente, Nivel de entendimiento (80%), Cumplimiento de Expectativas (85%), Trabajo (90%); 2) Bueno, Nivel de entendimiento (20%), Cumplimiento de Expectativas (15%), Trabajo (10%); 3) Aceptable, Nivel de entendimiento (0%), Cumplimiento de Expectativas (0%), Trabajo (0%); 2) Insatisfactoria, Nivel de entendimiento (0%), Cumplimiento de Expectativas (0%), Trabajo (0%). La conclusión principal de este proyecto fue que el desarrollo de actividades donde se involucra la simulación en herramientas como lo es MATLAB y el desarrollo de prácticas en tiempo real, refuerza el conocimiento adquirido en los estudiantes.

1.5.2 Diseño y Construcción de un Prototipo para Medición de Armónicos Basado en Técnicas de DSP e Implementado En FPGA.

Trabajo de grado realizado por Carlos A. Cantillo, Omar G. Roa. Licenciados en Electrónica de la Universidad Pedagógica Nacional, año 2007. Este trabajo se trata de una implementación de un algoritmo de la transformada rápida de Fourier (FFT), programado en una FPGA Spartan 3 XC3S400. Inicialmente, capturaron una señal de la red eléctrica, por medio de un circuito de adquisición de señal diseñado por los autores. La frecuencia de trabajo de este dispositivo fue de 60 Hz, y la frecuencia de muestro fue de 64 muestras por segundo.

Luego de esto, la FPGA capta la señal previamente digitalizada para proceder a aplicar el algoritmo diseñado en un en lenguaje de descripción de hardware (HDL). Posterior al procesamiento de la señal tomada de la red, se realizó una transmisión de información de la FFT de la señal, por medio de un protocolo de comunicación serial RS232 al computador, donde se realizó una visualización de los armónicos, por medio de un software diseñado en VISUAL BASIC 6.0.

1.5.3 Diseño y Programación de Filtros Digitales FIR en Lenguaje de Descripción de Hardware HDL e Implementación en FPGA.

Trabajo de grado realizado por Julio G. Bonilla, William E. Peña. Licenciados en Electrónica de la Universidad Pedagógica Nacional, año 2010. Este trabajo fue implementado en una FPGA Nexys 2. Lo que se buscó fue desarrollar un dispositivo que capturará señales de audio para ser posteriormente procesada en la con filtros digitales. Inicialmente los autores se enfocaron en el diseño y la implementación de una etapa de adquisición de la señal por medio de un PIC 18LF4523. Posteriormente se implementó una etapa de prefiltrado, para poder adecuar de una forma eficiente la señal al proceso de filtrado. La etapa de filtrado fue implementada por medio de un filtro FIR. Agregado a esto, se encontraron los coeficientes de la función de transferencia del filtro por medio del método Parks-Mcclellan.

Para la etapa de adecuación de la señal y la DAC (conversión digital a analógica), usaron un circuito integrado de referencia TLV5619 que proporciona grandes ventajas en relación a la salida con varios niveles de voltaje de acuerdo a los niveles de referencia de la señal, acompañado de un amplificador operacional en modo restador para cambiar el nivel de referencia positivo entregado por la etapa de filtrado digital.
Al final del desarrollo del dispositivo, los autores llegaron a la conclusión de que el sistema de desarrollo de la Nexys 2 permite una implementación a satisfacción de estructuras DSP en arquitectura paralela, a pesar de contar con pocos módulos multiplicadores embebidos.

1.5.4 Sistema de tiempo real embebido en un microcontrolador de alto desempeño para el procesamiento de audio.

Proyecto realizado por Moisés A. Martínez H, Ingeniero en Automatización de la Universidad Autónoma de Querétaro, México, año 2010. El autor se enfocó a la realización e implementación de un control que fuera capaz de desarrollar un procesamiento de audio de una manera más económica, e igualmente eficaz que dispositivos especializados en esa área, empleando un sistema operativo de tiempo real embebido en un microcontrolador. Inicialmente pretendió explicar las características previas al procesamiento de audio, como el correcto desarrollo de una etapa de acondicionamiento que permitiera el funcionamiento adecuado de un microcontrolador. También explica las características que debe poseer un microcontrolador para que se pueda implementar un código adecuado para el procesamiento de audio, como la memoria, las instrucciones por segundo, características de voltaje de entrada, etc. De igual manera hizo referencias para tomar la decisión de el compilador adecuado, que pueda utilizar y satisfacer las necesidades de la investigación, empezando por la parte del Sistema Operativo en Tiempo Real. Agregado e esto, demostró como un Sistema Operativo en Tiempo Real mejora el desempeño de la parte crítica de un programa, claramente realizando en base a multitareas. Finalmente se intentó demostrar que los

objetivos y la hipótesis planteados se cumplen satisfactoriamente, permitiendo dejar una investigación que abrirá oportunidades a futuros proyectos.

1.5.5 Implementación de filtro digital en tiempo real para detección de la onda R.

Proyecto realizado por Javier E. Gonzales B, Cristian V. Cárdenas y Johann Nieto C, de la Universidad Santo Tomás, Bogotá, año 2014. El objetivo de este trabajo fue presentar los resultados obtenidos a partir de la implementación de un filtro digital en tiempo real, orientado a la detección de la onda de R de una señal electrocardiográfica. La estrategia que implementaron fue basada en la teoría de filtros promediadores y los filtros derivadores, los cuales implementaron en un dispositivo programable ARM y de uso abierto. Los resultados que obtuvieron fueron la base para la implementación de un sistema que se encargara de cuantificar la frecuencia cardiaca realizada en tiempo real.

1.6 Marco teórica

1.6.1 Señal eléctrica:

Una señal eléctrica se puede definir como una función que varía con respecto al tiempo, la cual indica información sobre su naturaleza, en la cual puede transmitir información. Existen múltiples tipos de señales dependiendo de su naturaleza, sea analógica a digital. Dentro de estas características las señales pueden ser periódicas o aperiódicas.

1.6.2 Filtros digitales:

Un filtro es un proceso computacional o algoritmo mediante el cual una señal digital (secuencia de muestras) es transformada en una segunda secuencia de muestras o señal digital de salida. Un filtro es un proceso computacional o algoritmo mediante el cual una señal digital (secuencia de muestras) es transformada en una segunda secuencia de muestras o señal digital de salida.

1.6.3 Microcontrolador:

Los microcontroladores están presentes en cualquier lugar; en el trabajo, en casa, en la industria, en el espacio y en general, en nuestra vida. Pueden estar controlando desde la temperatura de un horno en casa, nuestro teléfono celular y hasta algún proceso industrial. Cada vez dependemos más de ellos, sin ellos no serían posibles las comunicaciones, no habría producción a grandes escalas en las fábricas, no tendríamos equipos electrónicos utilizados en hospitales, ni tampoco sistemas que nos facilitan la vida a diario como las lavadoras, o simplemente no tendríamos un sistema que se ha vuelto esencial para el desarrollo de la humanidad: la computadora.

1.6.4 Microcontrolador Xmega:

Son microcontroladores de la empresa desarrolladora de tecnología Atmel, los cuales tiene un núcleo de procesamiento con base a arquitectura de tipo AVR basada en RISC (Computador con Conjunto de Instrucciones Reducidas), los cuales los hacen óptimos para mejorar la velocidad y poder conseguir procesamiento en tiempo real. Este microcontrolador se caracteriza por incluir una gran cantidad de periféricos dentro de su encapsulado, caracterizándose al poseer módulos ADC (Convertidor Analógico Digital) y DAC (Convertidor Digital Analógico) de 12 bits, traduciéndose en términos más simples, tener gran sensibilidad a pequeños cambios de voltaje a la entrada debido a la resolución de 12 bits. Además de esto son capaz de tomar muestras de la señal de entrada a una frecuencia de muestreo de hasta 2 Ms/s, los cuales los hace ventajosos para el tratamiento digital de señales.

XMEGA como el más avanzado AVR de 8 bits de Atmel

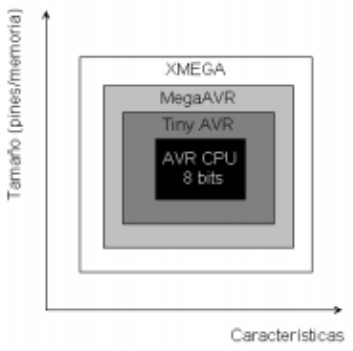


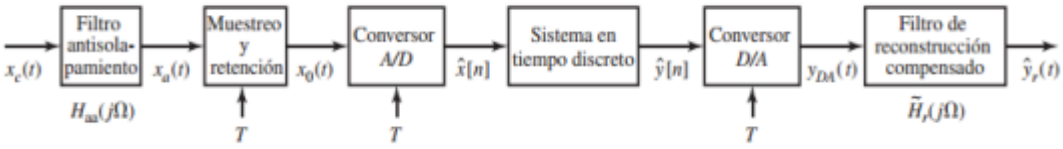
Figura 1. Microcontrolador XMEGA respecto a otros AVRs de 8 bits

Fuente: (Atmel, 2008, pág. 2)

La figura anterior compara el tamaño y características de los dispositivos de Atmel con núcleo AVR, donde se evidencia fácilmente que los XMEGA son los más desarrollados con respecto a desempeño. La programación de estos dispositivos se puede realizar por medio de lenguaje Assembler y lenguaje C, este último se caracteriza por ser de alto nivel que nos permite enfocarnos en rutinas y funciones más globalizadas dentro de nuestro programa a implementar.

1.6.5 Adquisición de Señales:

Inicialmente, para realizar el procesamiento de cualquier señal, se debe tener en cuenta una etapa de adquisición de datos, es decir un proceso de conversión analógica a digital (ADC), donde la señal pasará por un proceso de discretización para posteriormente ser enviada a un módulo que se encargue del proceso de filtrado.
Como lo menciona Oppenheim, un procesado digital de señales analógicas debe poseer por conveniencia la siguiente estructura:



Estas estructuras contienen filtros de preprocesamiento y posprocesamiento, los cuales se encargan de asegurar el ancho de banda, paras evitar componentes de frecuencia por encima de la máxima frecuencia de trabajo. Estos filtros también eliminan el ruido aditivo que puede contener inicialmente la señal y realizan restitución sin distorsión y limitación de ancho de banda.

1.6.6 Conversión analógica a digital

Como se mencionó anteriormente este proceso consta en realizar mediciones de forma periódica de una señal de tipo analógica, (es decir que está representada de forma continua y valores infinitesimales en un intervalo de tiempo), para llevarlos a valores discretos cuantizados representados por datos numéricos codificados en valores binarios. Este proceso es indispensable en la tarea de adquisición de señales, está representado por las siguientes etapas: muestreo, cuantización y codificación.

1.6.7 Muestreo:

Según el teorema de muestreo (Harry Nyquist 1928, Claude E. Shannon 1949), para que una señal analógica pueda transmitirse, procesarse por un medio digital y volver a reconstruirse, es necesario muestrear con una frecuencia fm, mayor o igual al doble del ancho de banda B, de la señal analógica. Matemáticamente está dado por la siguiente ecuación $fm \geq 2B$.

1.6.8 Cuantización:

Es el proceso en el que se convierte valores continuos en discretos. En esta parte juega un papel fundamental la resolución del sistema de cuantización, dada en términos de la ventana de trabajo de voltaje de entrada y la cantidad de bits usados en la conversión.

1.6.9 Tratamiento Digital de Señales:

Con la creación de computadoras digitales, se ha renovado el énfasis en el análisis y diseño de sistemas digitales, que representa una clase importante de sistemas de ingeniería. Esto se debe a la gran ventaja que ofrece la digitalización de datos en cuanto a la duración con el paso del tiempo de la información, ruido e interferencia. Una gran ventaja de la implementación de sistemas digitales e informáticos, es la utilización de lenguajes programación los cuales se pueden modificar y optimizar de acuerdo con las aplicaciones específicas requeridas usando el mismo hardware.

Un lenguaje de programación es un método de comunicación entre un usuario y la máquina, el cual permite proporcionar órdenes hacia el dispositivo, por medio de funciones y caracteres inteligibles para ambos. En sistemas de procesamiento de información, estos lenguajes son los encargados de proporcionar las indicaciones a realizar en dispositivos como los microprocesadores y microcontroladores.
Estos sistemas programables son herramientas de gran ayuda al momento implementar aplicaciones donde se requiera estudiar el comportamiento de sistemas analógicos puesto que permiten realizar prácticas donde se contrasta la teoría con procesos en tiempo real.

Herramienta conceptual importante al momento de estudiar las señales y los sistemas es la FFT. Es un algoritmo eficiente muy utilizado en el tratamiento de señales digitales para encontrar la transformada Discreta de Fourier de una señal en tiempo discreto, la cual proporciona información relacionada de forma directa sobre el contenido espectral de la señal, requisito importante para conocer el ancho de banda y potencia. Estos datos se relacionan de forma directa con el diseño de filtros en el momento que se desea encontrar las frecuencias de corte, las ganancias y atenuaciones en las bandas de paso y rechazo.

1.6.10 Filtrado Digital

El tema del filtrado digital se puede abordar desde 2 ámbitos. Los FIR (Respuesta Finita al Impulso) y los IIR (Respuesta Infinita al Impulso).
Los sistemas FIR también llamados de fase lineal, se utilizan para aplicaciones donde es importante que la respuesta de la fase se mantenga constante de acuerdo con el cambio de frecuencia. Los sistemas IIR, son basados en el diseño de filtros analógicos lo que determina que son físicamente construibles. La diferencia entre los dos radica en que para implementar un filtro FIR, se requiere mayor cantidad de coeficientes para los valores determinados a partir de la función de transferencia, en cambio los IIR no necesariamente. Otra diferencia entre ambos es debido a la sensibilidad de encaminarse a la inestabilidad por parte de los IIR, esto dificulta un poco su diseño e implementación.

Capítulo 2. Aspectos Metodológicos

En el proceso de desarrollo y elaboración de un proyecto donde involucre el diseño de artefactos tecnológicos, especialmente diseño de circuitos electrónicos existe un tipo de metodología adecuado para realizar un proceso que asegure la producción de resultados acorde a los propósitos. El tipo de metodología de diseño planteado en este proyecto es llamado TOP DOWN.

La metodología TOP DOWN se encarga de estudiar el proceso de diseño desde el propósito más general y global de un sistema, partiendo del macro hasta los detalles más mínimos; es decir, se basa en la división de problema de diseño generales en subproblemas que, a la vez, se dividen en problemas más pequeños que se atiende de manera puntual. Esto garantiza que, en el caso de un prototipo de tipo electrónico el estudio del sistema se pueda dividir en módulos o etapas, cada una con una determinada función que aporta al desarrollo del producto final.

Al finalizar el producto el mantenimiento es fácil debido a que, en su diseño, la división por etapas permite saber determinar la funcionalidad y desempeño característico de cada una. Esto aumenta las ventajas al momento de poder determinar la detección de fallas y posterior corrección.

Figura 2. Metodología TOP DOWN. Fuente: http://mimateriaenlinea.unid.edu.mx

Esta metodología de diseño presenta múltiples ventajas que entre las cuales se puede destacar las siguientes:

- Incrementar productividad de diseño haciendo que los tiempos de producción sean menores.

- Rápida detección de errores debido a la mayor inversión de tiempo al diseño de cada etapa, esto determina fácilmente las características puntuales de cada una.

- Brinda una mayor habilidad para administrar diseños complejos, que nace de la exploración del sistema y mayor entendimiento de como viene el diseño.

- Reorganizar las tareas del diseño, permitiendo la elaboración de labores de forma paralela y no limitándose a crear dependencias en serie.

El diseño TOP DOWN también reduce el impacto de los cambios posibles que pueden aparecer posiblemente a futuro. Si el producto necesita ser rediseñado de forma parcial, esta metodología permite realizar cambios de forma fácil y rápida.

Para que este proceso TOP DOWN sea efectivo, se parte de los siguientes parámetros básicos:

- Una representación de diseño compartida, que permita durante todo el proceso trabajar simultáneamente en el esquema.

- La involucración de pasos múltiples, empezando con una abstracción superior que vaya refinando los detalles.

- Planeación básica de la verificación del proceso de diseño, donde sean identificados prontamente los riesgos, los planes de modelación y simulación para contrarrestar los riesgos.

Capítulo 3. Desarrollo del prototipo programable orientado a filtrado digital

En este apartado se describirá la forma en cómo se procedió a realizar el proyecto, con base en los objetivos específicos del mismo y por supuesto las razones que se tuvieron para dar por hecho cada parte que integra este trabajo. Se hablará brevemente de las herramientas que se eligieron y por qué se optó por la utilización de componentes nombrados puntualmente en cada subproceso.

3.1 Diseño e implementación de etapas del prototipo que cumplan con requerimientos del proyecto, con su respectivo diseño de PCB

3.1.1 Criterios de diseño.

Dependiendo de cuál sea el propósito con el que se desarrollará un dispositivo o aplicación, se debe pensar en cómo estructurar las partes para que cumplan con la debida función principal. Debido a que este trabajo de grado se enfoca en realizar tratamiento digital de señal, específicamente para filtrar señales de audio, se explicará a continuación las características relevantes a la hora de pensar en un diseño para el prototipo.

3.1.1.1 *Características que definen el diseño:*

- La señal de audio para el espectro audible está limitada de los 20Hz a los 20kHz, por lo cual se debe pensar en implementar etapas de filtrado anteriores y posteriores al tratamiento en dominio discreto (es decir, en el dominio de los sistemas digitales).
- Un fenómeno común en el tratamiento de señales es el Aliasing, que entromete componentes espectrales a la señal de salida haciéndola completamente distinta a la de la entrada; sin embargo, esto se evita utilizando filtros pasa bajas antes de una conversión a dominio digital; para esto, es conveniente usar filtros de orden superior, es decir, que elimine en lo posible frecuencias mayores a 20kHz. Recomendable para eliminar el Aliasing, usar filtros de capacitor conmutado.
- Los sistemas digitales que se enfocan en procesar señales externas, en su mayoría sus núcleos son con base a sistemas microcontrolados, que funcionan con voltajes positivos, por lo que es necesario adecuar la señal de audio a una sola polaridad, es decir volverla unipolar.
- La señal que proporciona por defecto una tarjeta de sonido de un computador es limita a una amplitud máxima de 2Vpp (dos voltios pico a pico, es decir la diferencia entre su voltaje más positivo y el más negativo). Con respecto a estas características se debe pensar en variar la amplitud de la señal dependiendo de las etapas por las que atraviesa.

A partir de los requerimientos que acabaron de plantearse, se propone el siguiente esquema donde muestra una serie de etapas que cumplen con las condiciones necesarias para adecuar una señal de audio, antes y después de ser procesada para su posterior reproducción.

3.1.2 Estructura del prototipo.

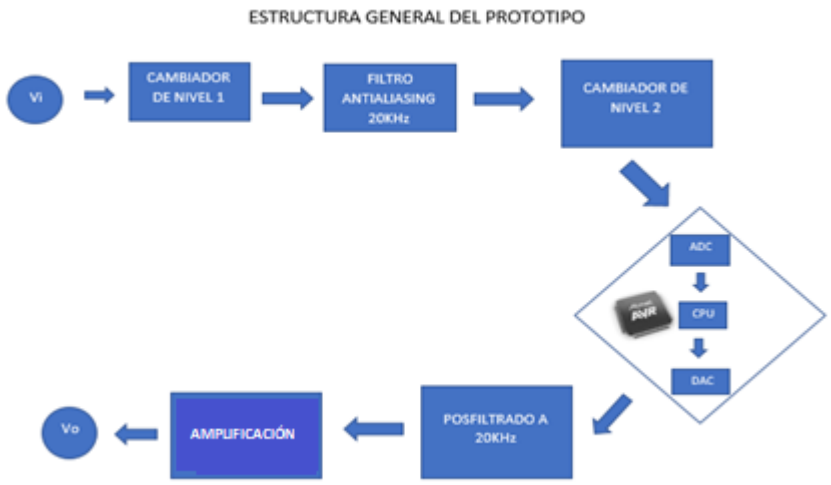


Figura 3. Estructura del prototipo

En la figura 3 se observa el diagrama donde fluye la señal de audio de entrada Vi (desde la parte superior izquierda hasta la parte inferior izquierda), que pasa por cada uno de los bloques los cuales tiene una función específica para realizar el debido tratamiento. Como se observa, el cerebro del prototipo es un dispositivo de ATMEL de referencia ATXMEGA32A4U. Este es un microcontrolador de 8 bits que posee grandes ventajas con respecto a diferentes arquitecturas opcionales para realizar variedad de procesos en forma digital.

Analizando detalladamente el esquema general del prototipo, se observa que la señal de audio de entrada es sometida inicialmente a una etapa cambiadora de nivel, luego a un filtro, posteriormente a otro cambiador de nivel antes de aplicarse a la entrada del microcontrolador. Posterior a la etapa del microcontrolador existe otra etapa cambiadora de nivel, estas etapas son fundamentales debido que cada etapa tiene características de entrada muy específicas y diferentes con respecto a las otras, y es por eso que es necesario cambiar la amplitud y el nivel DC de la señal. En su totalidad hay 2 etapas cambiadoras de nivel, una de amplificación, dos de filtrado y una de procesamiento digital. A continuación, se empezará a explicar la razón y el funcionamiento de cada una.

3.1.2.1 ***Etapas del prototipo.***

3.1.2.1.1 *Etapa 1: Cambiador de nivel 1.*

Como se mencionó con anterioridad, la señal de audio de entrada está limitada a una amplitud 2 voltios pico a pico, de naturaleza bipolar (figura 4.a, pag.13), es decir una señal AC que tiene su amplitud tanto en la parte positiva como negativa, una señal con amplitud apta para la entrada al convertidor analógico a digital del microcontrolador (que es de 3.3v), pero, por su estructura y

funcionamiento el microcontrolador no permite señales bipolares (se explicará con mayor detalle su funcionamiento en el apartado de procesamiento digital)

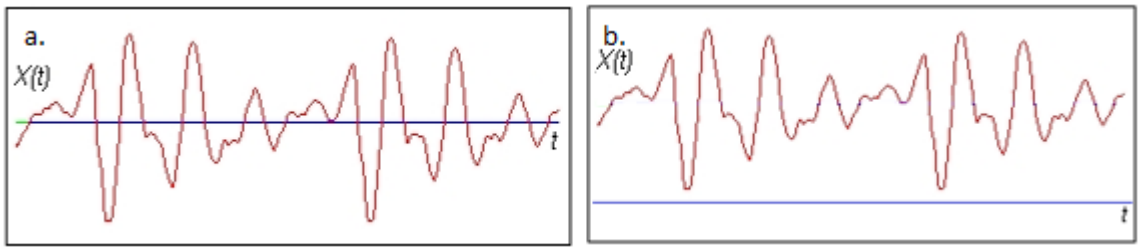


Figura 4. Señal de audio: a) señal bipolar, b) señal unipolar, es decir con offset. Fuente Andrés, Flores Espinoza (1993).

Para solucionar este inconveniente es necesario sumarle un valor positivo a la señal de entrada, que garantice que toda su forma de onda se encuentre por encima del valor 0. La mejor alternativa ante esta situación es implementar un circuito sumador con amplificadores operacionales, donde garantice una impedancia de entrada alta que no afecte la señal y una impedancia de salida baja que no afecte la siguiente etapa.

La figura 5, muestra el diagrama esquemático propuesto para solucionar el inconveniente sobre el desplazamiento de nivel

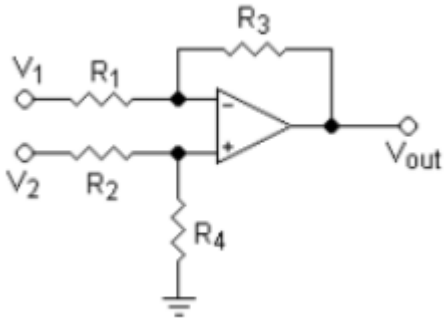


Figura 5. Amplificador operacional inversor – diferencial. Fuente User:Omegatron using Klunky schematic editor, 26 Junio 2005

Como se puede observar, el diagrama involucra 2 voltajes de entrada y su respectiva salida, 4 resistencias y un amplificador operacional. El amplificador operacional utilizado es de referencia LM 358, el cual proporciona característica adecuadas como lo son una impedancia de entrada alta, un ancho de banda de más de 1 MHz y una adecuada ventana de voltaje donde la señal se puede desplazar sin saturarse (para mayor detalle, se recomienda ver la hoja de especificaciones proporcionada por los fabricantes). El voltaje de alimentación definido para esta y la gran mayoría de las etapas del circuito es 5V.

La función de esta etapa es tomar la señal de audio V1, invertir su polaridad, multiplicarla por un factor en términos de R3 y R1, y finalmente sumarle un valor dependiendo de la señal de entrada V2 y los valores de R1, R2, R3 y R4. Matemáticamente está expresado de la siguiente manera:

$$Vout = V2\frac{R4}{R2+R4}\left(\frac{R3}{R1}+1\right) - V1\frac{R3}{R1} \quad (1)$$

Los valores de las resistencias y el valor constante de V1, se eligieron de acuerdo con las características de entrada de la etapa filtradora antialiasing que se explicará con detalle más adelante.

Los valores respectivamente son:

$$V1 = 3.3v; \quad R1 = 10K\Omega; \quad R2 = 20K\Omega; \quad \text{R3} = 10K\Omega; \quad \text{R4} = 10K\Omega$$

Reemplazando los de las resistencias y V1, y simplificando la ecuación (1), el valor correspondiente de Vout está en términos de la siguiente expresión:

$$Vout = 2.475 - 0.5V1 \ \ v \quad (2)$$

La señal de salida de esta etapa se asemeja al bosquejo proporcionado por la figura 4.b de la página 13.

3.1.2.1.2 *Etapa 2: Filtrado Antialiasing a 20KHz.*

Dicho anteriormente, el fenómeno del Aliasing proporciona problemas a la hora de reconstruir una señal. Es por esto principalmente que se propone realizar una etapa de filtrado, donde se escoge una frecuencia de 20KHz como punto clave de partida para empezar el diseño. La calidad de dicho filtro depende principalmente del tipo y el orden con que se construya. Dentro de la teoría de filtros analógicos, uno de los más utilizados por su facilidad y eficiencia son los de tipo Butterworth, que se caracterizan por tener una banda de paso plana y una ganancia de 0.707 en la frecuencia de corte, es decir la frecuencia clave del filtro. Así como se clasifica filtros por la eficiencia en sus curvas, ganancia y atenuación; es importante clasificarlos por la naturaleza de su funcionamiento. Los filtros eléctricos principalmente se clasifican según la figura 6:

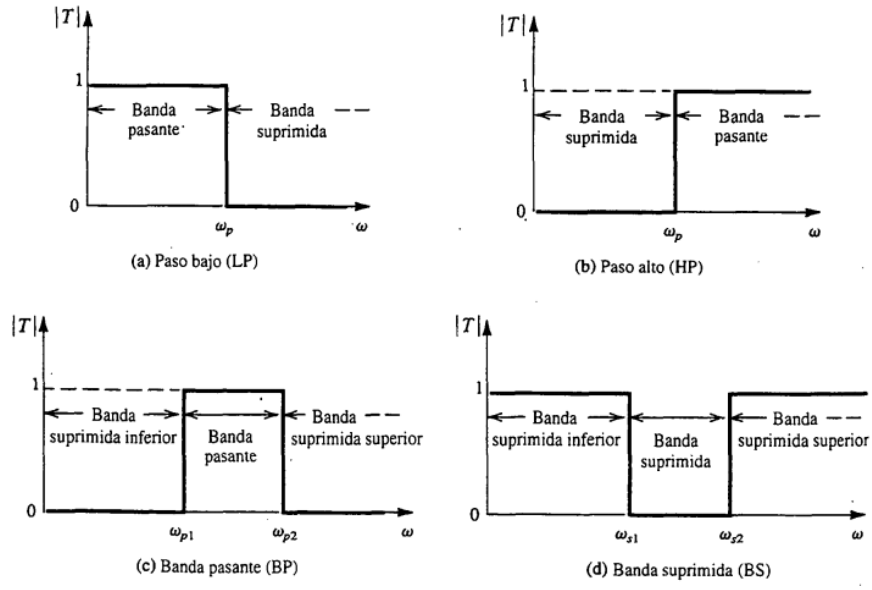


Figura 6. Características ideales de cuatro tipos principales de filtros: a) paso bajo (LP); b) paso alto (HP); c) banda de paso (BP); d)banda de rechazo (BS). Fuente Circuitos Microelectrónicos. Sedra, Adel. Smith, Kenneth. Oxford University Press. Pag 886.

El tipo de filtro más indicado para el diseño del filtro antialiasing es el de tipo paso bajo; puesto que deja pasar las frecuencias menores a 20kHz y suprime la amplitud de las frecuencias mayores.

Como se ha explicado, la eficiencia de un filtro depende directamente del orden; pero, al momento de implementar configuraciones en cascada para aumentar la calidad de funcionamiento, nos vemos condicionados a los efectos que genera una gran cantidad componentes, que, en su naturaleza, generan costos excesivos, aumento de espacio y posiblemente se entrometa ruido a la señal de salida. Solución conveniente a estas problemáticas de diseño, es la implementación de filtros de condensador conmutado, que son circuitos integrados multiconfigurable, por lo tanto, se acaba el problema de tamaño con respecto al orden, son menos sensibles al ruido y la temperatura; la frecuencia de corte principal del filtro depende directamente de una señal de reloj externa, en otras palabras, reduce y simplifica en gran medida el tamaño del diseño. El filtro de capacitor conmutado elegido para cumplir con los requerimientos es el mostrado en la figura 6:

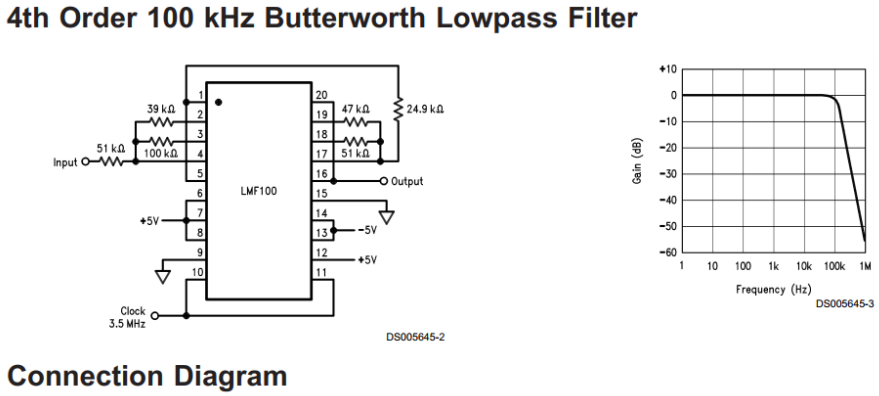


Figura 7. Filtro pasabajos de 4° orden implementado con un LMF100. Fuente: High Performance Dual Switched Capacitor Filter. National Semiconductor, julio de 1999.

Como se puede observar, la figura 7 muestra en la parte izquierda el diagrama de conexión, y el diagrama de bode para una frecuencia de 100 KHz en su parte derecha. Por información del fabricante, este dispositivo se puede configurar para ajustar la frecuencia de corte a 20 KHz, gracias a la frecuencia de reloj que se aplica a las terminales 10 y 11. En el diagrama de bode, se observa las ventajas que ofrece con respecto a la banda de paso, antes de los 100KHz. Las características de banda de paso son ideales para nuestra aplicación puesto que su ganancia es plana y de 0 dB, es decir unitaria.

La alimentación de este circuito es dada por fuente dual o fuente simple, es decir con voltajes de 5 y -5 voltios o de 5 y 0v. Para mayor conveniencia, se decidió aplicar fuente sencilla de 5 voltios, adecuada para energizar sistemas digitales y sistemas a baja potencia. En la hoja de especificaciones del LMF100, hay una tabla de características eléctricas de funcionamiento donde se menciona características de la ventana de trabajo para el voltaje de salida, es decir los voltajes permitidos en la entrada, para que el voltaje de salida no sufra saturaciones. Según estos datos, los voltajes permitidos en ella son de ± 1.4 v con respecto a la tierra analógica (AGND que es 2.5v), para aclarar esto, el voltaje de entrada debe estar entre 3.9v y 1.1v, bastante limitada, un 56% de excursión de voltaje útil en relación con los 5v de la fuente de alimentación. Esto explica las razones por las cuales a la primera etapa la señal de entrada se le sumó un voltaje de 2.47v y una atenuación del 50%, resumida en la ecuación (2).

La señal de salida del filtro antialiasing, es una señal montada sobre 2.47v la cual es aplicada a la etapa cambiadora de nivel 2. El diagrama proporcionado para la etapa de filtrado antialiasing de este proyecto, es el sugerido por el fabricante en la figura 6, pero con una diferencia de la fuente de alimentación simple de 5v, una frecuencia de reloj de 1.4142 MHz calculada a partir de la ecuación (3), las mismas resistencias y cambios sugeridos a partir de la hoja de especificaciones para alimentación simple.

$$fo = \frac{fclk}{100}\sqrt{2} \qquad (3)$$

Donde f_{clk} es igual a 1.4142 MHz, para finalmente obtener fo = 20KHz.

3.1.2.1.3 Etapa 3: Cambiador de nivel 2.

Posterior a la etapa de filtrado se debe pensar en cómo adecuar la señal de audio antes de entregarla a microcontrolador. Sabiendo que la señal que recibe el convertidor analógico a digital está limitada por el valor de alimentación y la referencia de conversión analógica (AREF), que para este caso es el voltaje Vcc del microcontrolador (3.3v), la señal de salida del cambiador de nivel 2 tiene que cumplir con estos requerimientos para asegurar una adecuada conversión, especialmente debe estar montada sobre la mitad del voltaje de referencia (AREF/2 siendo 1.65v), y su amplitud debe abarcar gran cantidad de la excursión de voltaje de entrada, asegurando una adecuada resolución en el proceso de conversión. Este proceso se explicará detalladamente en la siguiente sección.

Retomando la información proporcionada por la etapa 2, la señal de salida está montada sobre 2.47v y la amplitud es de 1v pico a pico y según los requisitos para la conversión, la señal debe estar tener un offset de 1.65v y hay que tenerse en cuenta aumentar su amplitud. Inicialmente su amplitud era de 2v pico a pico, pero fue modificándose por criterios de diseño de las etapas anteriores, ahora es momento de restaurar la señal a sus parámetros iniciales, pero con un voltaje offset de 1.65v. El circuito propuesto para cumplir los requerimientos es el de la figura 8.

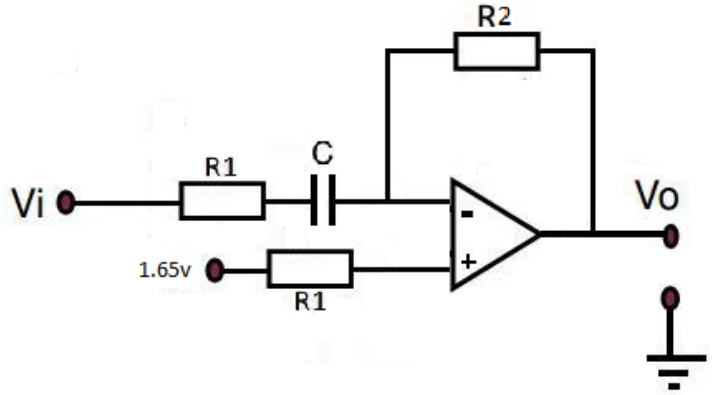


Figura 8. Amplificador y eliminador de offset. Fuente: www.electronicaengeneral.wordpress.com

La figura 8 muestra un circuito en relación a un amplificador operacional encargado de eliminar inicialmente el voltaje offset de 2.47v gracias al condensador que está en serie a la resistencia de entrada del voltaje Vi. La resistencia R2 proporciona una ganancia respecto al valor R1 de valor negativo, debido a que la entrada de la señal es por la terminal inversora. Finalmente, por su naturaleza de sumador inversor, a la señal de entrada se le suma el voltaje de 1.65v que ingresa por la terminal no inversora. Matemáticamente se puede deducir en términos de respuesta en frecuencia, gracias a la ecuación (4):

$$Vo = -\frac{R2}{R1}\frac{jw * Vi}{jw + \frac{1}{R1 * C}} + 1.65v \quad (4)$$

El operador "jw" es un número complejo que define la variable independiente de la función. Para valores de frecuencia altos en el orden de los cientos y los miles, la ecuación se puede simplificar aplicando el principio de límite cuando "jw" tiende a infinito. El procedimiento es el siguiente

$$Vo = \lim_{jw \to \infty} \left(-\frac{R2}{R1}\frac{jw * Vi}{jw + \frac{1}{R1 * C}} + 1.65v \right) \quad (5)$$

Simplificando (5) obtenemos:

$$Vo = -\frac{R2}{R1} * Vi + 1.65v \quad (5)$$

Con valores de R2 = 20KΩ, R1= 10KΩ, la ecuación (5) queda:

$$Vo = -2Vi + 1.65v \quad (5)$$

Gracias a la ecuación (5), podemos concluir que la señal proporcionada a la siguiente etapa cumplirá con los requisitos, entregando una señal con parámetros similares a la recibida por una tarjeta de audio, pero con un voltaje offset de 1.65v ideal para una conversión adecuada.

3.1.2.1.4 Etapa 4: Procesamiento de la señal.

En términos más claros, en este punto se encuentra la parte central y la más relevante en el proceso que hemos estado llevando, puesto que aquí es donde se procesa digitalmente la información proporcionada por las fuentes de audio. Esta etapa, como se mencionó con antelación, es con base a un sistema microcontrolado, especialmente diseñado para trabajar con este tipo de señales. La referencia del chip es ATXMEGA32A4U del fabricante de tecnología ATMEL. Se prefirió este dispositivo debido a las grandes ventajas que ofrece con respecto a múltiples dispositivos conocidos comercialmente, principalmente por incluir en su arquitectura encapsulada los módulos de conversión analógico a digital (ADC) y digital a analógico (DAC). Este último, está ausente en la mayoría de dispositivos de funcionamiento similar, por lo que en muchos casos es necesario incluir una etapa externa que se encargue de suplir esta función. Otra ventaja adicional, es que internamente contiene un módulo oscilador que se encarga de sincronizar la CPU y los periféricos, característica que no poseen demás dispositivos debido a que en sus casos particulares es necesario incluir un cristal externo. A continuación, se mostrará el diagrama de puertos y módulos internos del microcontrolador en la figura 9.

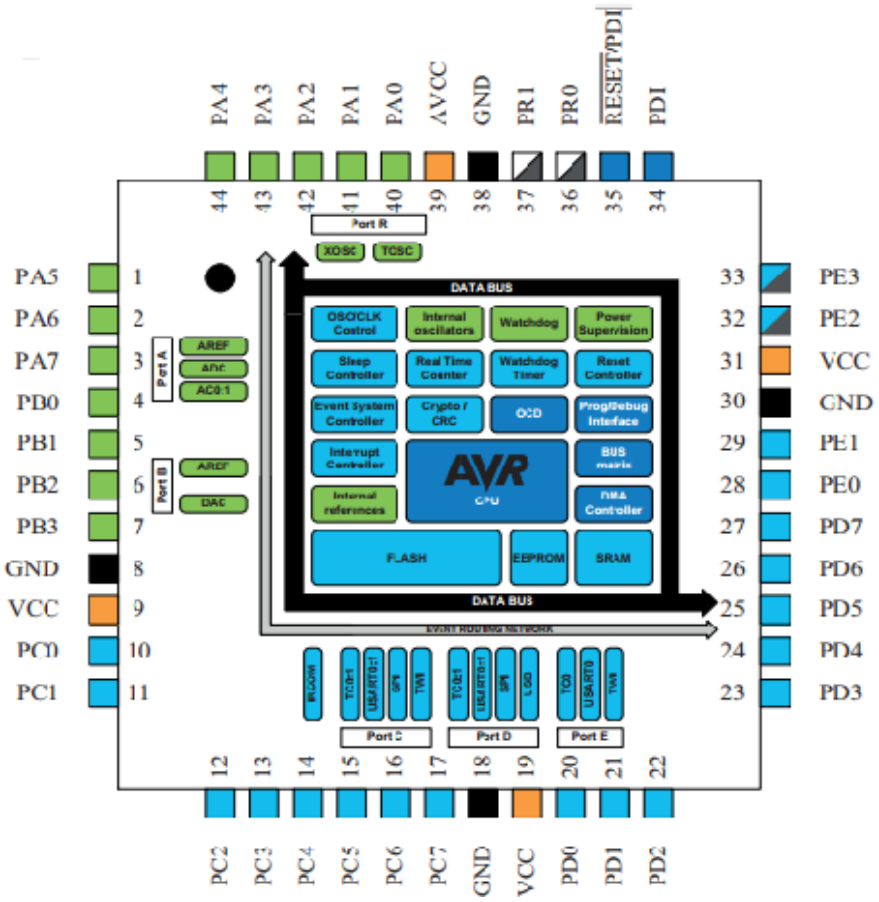


Figura 9. Microcontrolador AT XMEGA32A4U. Fuente: 8/16-bit Atmel XMEGA Microcontroller. Atmel Corporation 2014. Pag 4.

La figura 9 muestra los módulos y la relación con cada puerto. Se evidencia que tiene 34 pines programables y los módulos anteriormente mencionados. Se explicará las características y la funcionalidad de los módulos relevantes para este proyecto.

3.1.2.1.4.1 Convertidor Analógico a Digital (ADC).

Como el estudio principal de este proyecto es el tratamiento digital y las señales que inicialmente tomamos están en el dominio continuo, es necesario el uso de un dispositivo que convierta voltajes continuos en valores discretos.

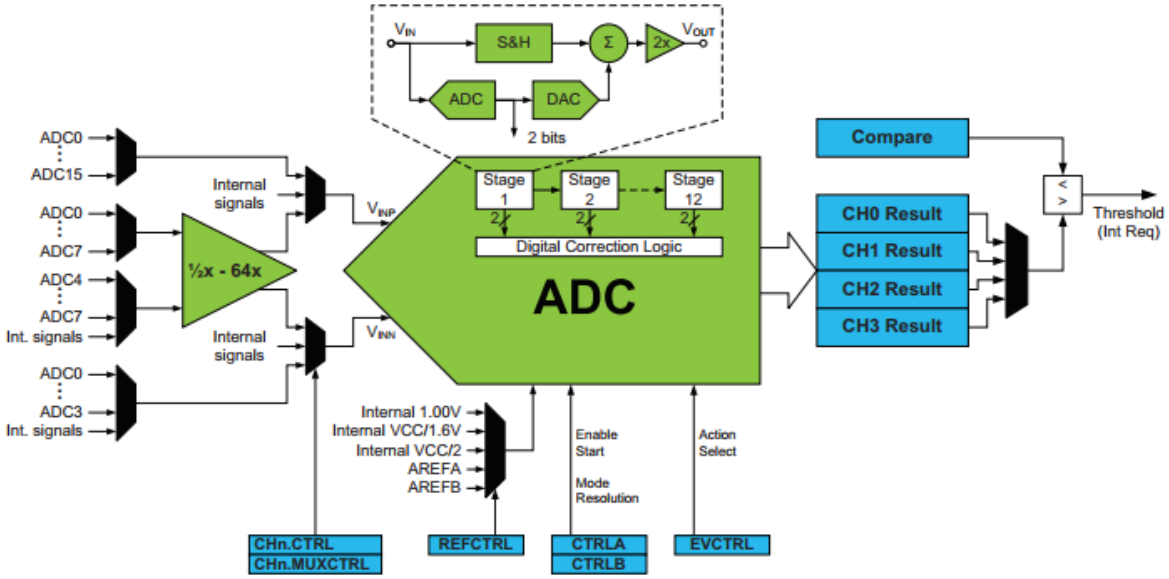


Figura 10. Convertidor analógico a digital. Fuente: 8-bit Atmel XMEGA AU manual. Atmel Corporation 2014. Pag 340.

El convertidor analógico a digital es el encargado de cumplir dicha función. El proporcionado para este caso tiene una resolución de 12 bits, es decir que podemos dividir el rango de valores de entrada en 2^12, que son 4096 palabras, desde 0 hasta 4095. Esta gran cantidad de valores suministra una excelente sensibilidad a pequeños cambios en el voltaje de entrada. Si se aclara que el voltaje de referencia para la conversión es de 3.3v, la resolución está dada por la ecuación (6).

$$Res = \frac{3.3v}{(2^{12} - 1)bit} \quad (6)$$

$$Res = 0.805\frac{mV}{bit} \quad (6)$$

Lo que nos quiere decir la ecuación (6) es que la entrada tiene sensibilidad de pequeños cambios de hasta 0.805 mV. Otra característica relevante es la frecuencia de muestreo. Según la hoja de especificaciones del fabricante, la señal de reloj proporcionada al ADC es la misma proporcionada a la CPU, que en este caso es de 32MHz. Según el fabricante, está señal de reloj pasa por un preescalador (figura 11), o divisor de frecuencias antes de sincronizar el ADC.

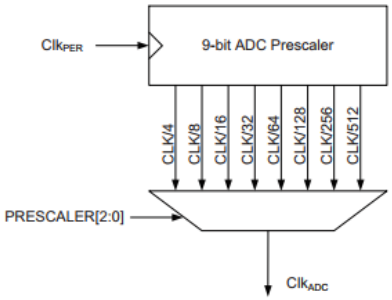


Figura 11. Preescalador de frecuencias. 8-bit Atmel XMEGA AU manual. Atmel Corporation 2014. Pag 346

Como se observa, el preescalador divide la frecuencia de entrada Clk_{PER} en 8 valores, dividido en potencias de a 2, donde un multiplexor o seleccionador elige el valor suministrado por software. La frecuencia máxima posible para el ADC Clk_{ADC} es 8 MHz. Según las especificaciones, la frecuencia máxima de muestreo del ADC es de 2 Ms/s (2 millones de muestras por segundo) para el caso de 8 bits de resolución; sin embargo, en el caso de resolución de 12 bits la frecuencia se reduce a unos 1.33 Ms/s (1 millón trescientos treinta mil muestras por segundo aproximadamente). Estos datos se explican fácilmente gracias a las especificaciones, puesto que el ADC demora un ciclo de reloj (Clk_{ADC} = 8 MHz), por cada dos bits de la muestra, para un total de 2 Ms/s para el caso de 8 bits, y 1.33 Ms/s para el de 12 bits, datos expuestos anteriormente. El ADC tiene otras cualidades como: 4 canales de conversión con 12 entradas compartidas, referencias de conversión interna, modo de conversión diferencial, entre otros; para este proyecto se utilizó el modo de conversión simple. Esta información se puede ver de forma detallada en la hoja de especificaciones proporcionada por el fabricante. Siguiendo el proceso, los datos convertidos son almacenados en un registro de 16 bits llamado RESH – RESL, este registro está a disposición de la CPU y se actualiza cada 4 ciclos de $Clk_{ADC.}$

3.1.2.1.4.2 CPU.

La función principal de la CPU (figura 12) es ejecutar el código y desempeñar todos los cálculos proporcionados. Su arquitectura AVR RISC (conjunto de instrucción reducidas), proporciona grandes facultades que posibilitan la segmentación y el paralelismo en la ejecución de instrucciones y reducen los accesos a memoria. Adicionalmente, posee un hardware multiplicador que reduce en gran medida la ejecución de operaciones que involucran multiplicaciones, ideal para este proyecto. Estas operaciones usan registros de 8 bits, proporcionando operaciones de números enteros y fraccionarios con signo y sin signo de hasta resultados numéricos de 32 bits. Una operación simple de 2 números de 8 bits tarda dos ciclos de máquina.

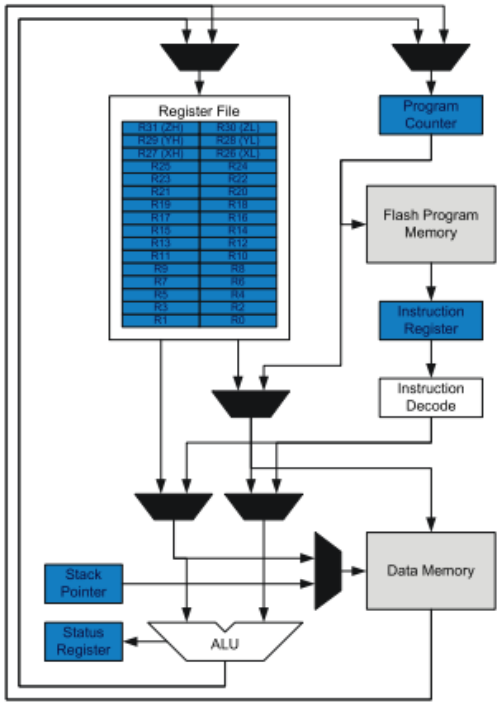


Figura 12. Diagrama de bloque de la CPU. 8-bit Atmel XMEGA AU manual. Atmel Corporation 2014. Pag 7

La figura 12 muestra el diagrama de bloques de la CPU, donde se observa la forma en cómo se estructura y como es el flujo de datos. La ALU, Unidad Lógico Aritmética es la encargada de realizar operaciones aritméticas básicas entre datos proporcionados, ya sea por memoria RAM o Register File o un dato proporcionado Flash Program Memory (Memoria flash). La ALU cuenta con 142 instrucción proporcionadas en lenguaje ensamblador. Se determinó usar un lenguaje de programación C por su facilidad a la hora de implementar algoritmos.

Así como mencionó, la señal de reloj para los módulos periféricos como el ADC y la CPU también requiere una señal que sincronice los procesos y defina la velocidad de procesamiento. Las señales de reloj son proporcionadas por el System Clock and Options Clock (Sincronizador del sistema y opciones de reloj, figura 13)

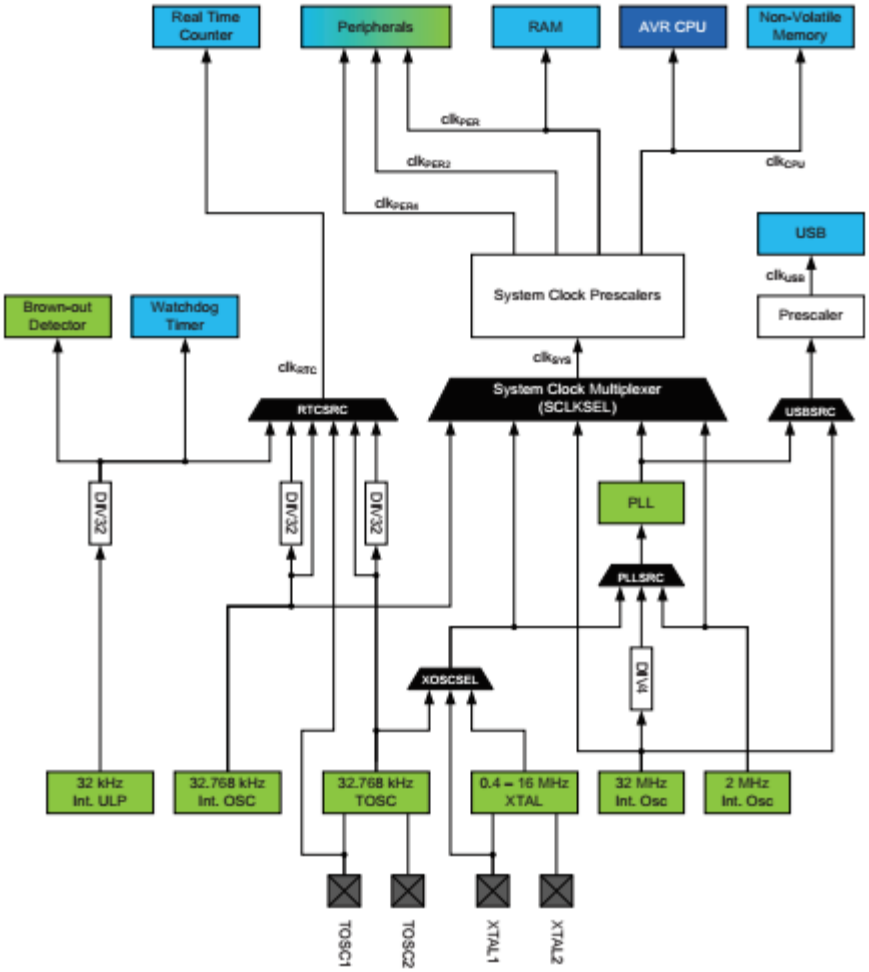


Figura 13. Sincronizador del sistema y distribuidor de reloj. 8-bit Atmel XMEGA AU manual. Atmel Corporation 2014. Pag 83.

El sincronizador del sistema es un módulo del microcontrolador encargado de la velocidad de la CPU y todos los periféricos. Explicando brevemente desde la parte inferior a la superior, los 6 módulos inferiores proporcionan señales de reloj de diferentes frecuencias, donde los dos internos (TOSC y XTAL) son los encargados de gestionar señales de reloj a partir de cristales externos. Los 4 de los extremos (Int. ULP, Int. OSC, 32 MHz, 2 MHz), son osciladores internos. Todos a excepción de Int ULP, pueden utilizarse para sincronizar la CPU y los periféricos, por medio del System Clock Multiplexer (Sistema Multiplexador del reloj), donde se elige la señal principal que utilizara el sistema. Se observa a costado izquierdo el RTCSRC (Fuente Reloj para el Contador en Tiempo Real), el cual es un módulo contador para aplicaciones que requieran una alta precisión en tiempo. Al costado derecho está el USBSRC (Fuente de reloj para USB), donde se sincroniza la velocidad de funcionamiento USB, si está activado. Para finalizar, en la parte superior derecha se encuentran los destinos principales de la señal de reloj elegida para sincronizar el sistema. La configuración de las opciones que brinda este módulo es realizada por software.

3.1.2.1.4.3 Convertidor Digital a Analógico (DAC).

Como el dominio del tiempo el cual es requerido para la señal de salida del sistema es el continuo, se debe pensar en convertir los datos procesados digitalmente. El DAC hace el proceso

inverso del ADC, convierte valores digitales a voltajes. Este módulo genera una gran ventaja por contenerse bajo el mismo chip, posibilitando una optimización de espacio y portabilidad en el circuito. Explicando las características del DAC, este permite una gran resolución similar a la del ADC, es decir 12 bits, es decir, tiene la facultad de generar pequeños cambios de voltaje a la salida, a partir de cambios pequeños en los valores internos.

Siguiendo con las especificaciones, el DAC del ATXMEGA32A4U, tiene la facultad de convertir datos hasta una tasa de 1 Ms/s (1 millón de muestras por segundo). Para nuestro caso, que es el espectro audible, es más que suficiente ese valor. Así como el ADC tiene hasta 12 entradas para conversión que se pueden usar de forma simultánea, el DAC posee dos canales de conversión, posibilitando aplicaciones en formato estéreo. Para este caso, las aplicaciones se realizan en un formato mono (un solo canal). La figura 14 muestra el diagrama esquemático del DAC.

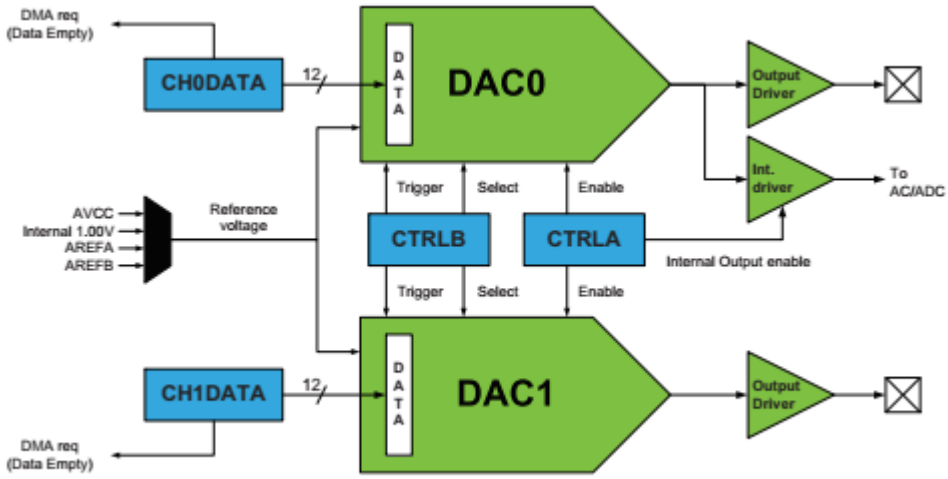


figura 14. Esquema del DAC. 8-bit Atmel XMEGA AU manual. Atmel Corporation 2014. Pag 367.

Se observa claramente los dos canales de conversión del DAC con salidas independientes. Los voltajes de referencia para este módulo son compartidos por ADC, que proporciona ventajas al momento de realizar el cambio del dominio discreto al dominio del tiempo continuo. La conversión de este tipo de dispositivos, en muchos casos es con base a circuitos de PWM (Modulación por ancho de pulso), que, según la teoría de Fourier, contiene una cantidad de espectral bastante alta. Esta condición en la señal de salida genera inconvenientes, puesto que la forma de onda presenta características de discontinuidad. Este problema se puede solucionar implementando un filtro posterior a esta etapa.

3.1.2.1.5 Etapa 5: Posfiltrado a 20 KHz.

Como se explicó anteriormente, una etapa de filtrado elimina componentes espectrales indeseadas en una señal, y este es el caso, la señal de salida del DAC tiene un gran contenido de estas, y es por esa razón que se implementa un filtro con características idénticas a la primera etapa de filtrado. El filtro de la etapa anterior, como se explicó en su momento, tiene características

apropiadas para este tipo de aplicaciones, con ganancia y atenuación necesarias para eliminar interferencias.

3.1.2.1.6 Etapa 6: Amplificación.

Las aplicaciones de audio necesariamente requieren una etapa ganancia, que soporte dependiendo del caso cargas con baja o media impedancia. Los sistemas digitales están diseñados específicamente para ser controlados por medio de voltaje, las corrientes inducidas no son relevantes para estos; en cambio, las etapas de ganancias se encargan principalmente de entregar medias y altas ganancias de corriente, aumentado directamente la potencia suministrada a la carga.

Para esta aplicación es más que suficiente una etapa de ganancia para baja potencia, puesto que este proyecto se enfoca principalmente en el procesamiento digital. La figura 15 muestra el diagrama de un amplificador basado en un LM386.

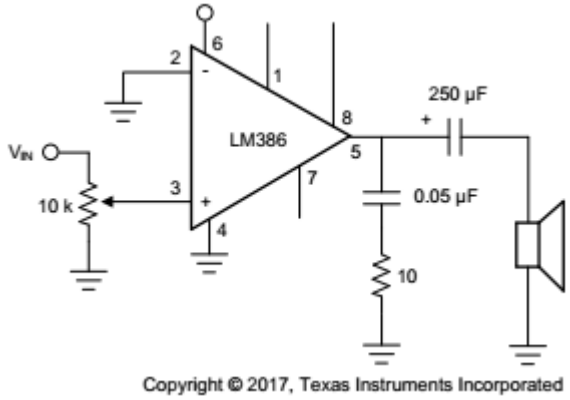


Figura 15. Amplificador de audio con Lm386. Texas Instruments 2017.

Este circuito se encarga de aumentar el voltaje de entrada hasta 20 veces y aumentar significativamente la corriente de carga. La ganancia está ajustada internamente. La salida va conectada a un Jack de audio, que será el puerto de salida del sistema.

3.1.3 Circuito esquemático general del Prototipo.

Anteriormente se explicó con detalle las razones principales y la importancia que cumplen las diferentes etapas en el desempeño general. A continuación, se muestra en la figura 15 el esquema general del proyecto.

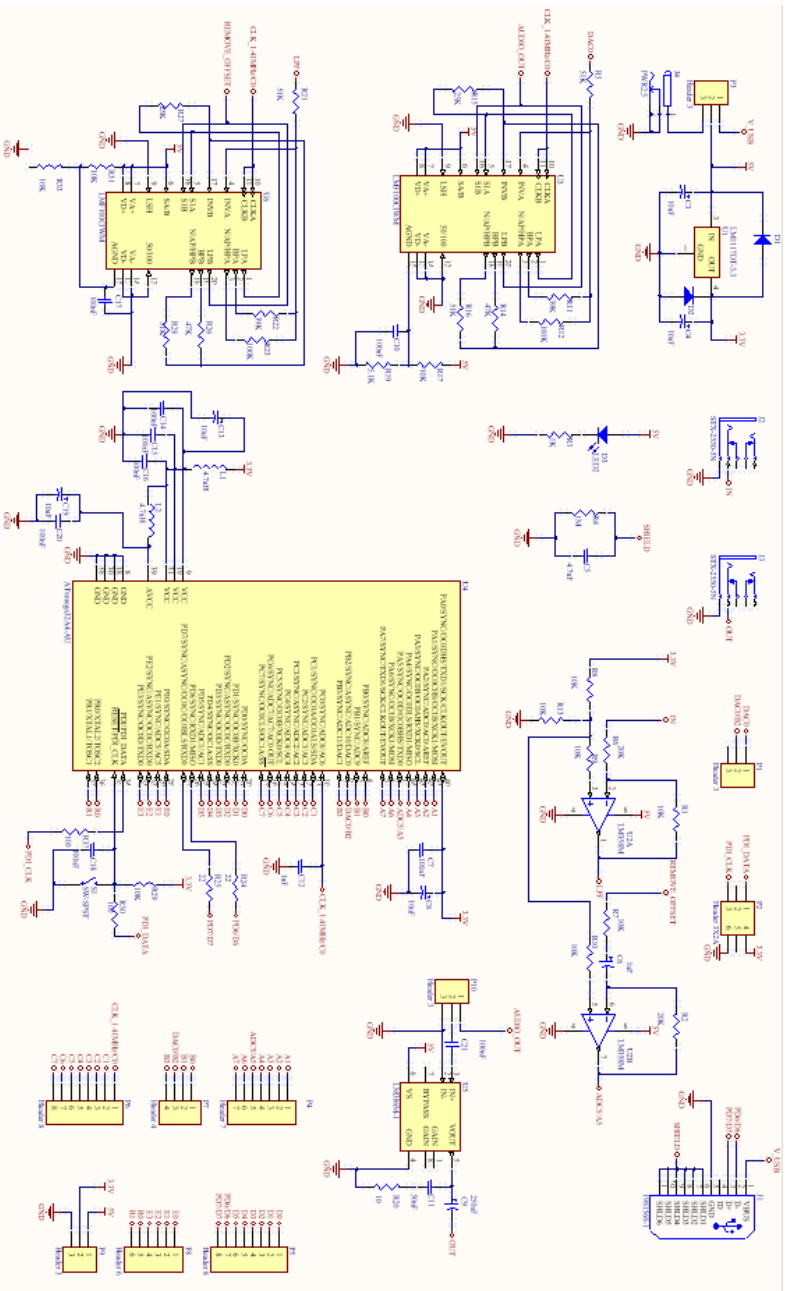


Figura 16. Esquema general del Prototipo

En la figura 16 se observa varios diagramas esquemáticos de los diferentes circuitos necesarios para cumplir con los requerimientos del proyecto. A demás de las etapas mencionadas, se le adiciono un puerto USB, para realizar posibles aplicaciones con envío y recepción de información a partir de diferentes dispositivos externos. A continuación, se explicará brevemente la estructura general del prototipo:

- Un Jack de potencia alimentado con un voltaje mayor a 5v, que suministra una de las entradas de alimentación del sistema, ubicado en la parte superior izquierda del diagrama.

- Un microcontrolador ATXMEGA32A4U de 44 pines, en la mitad.
- Dos circuitos integrados de referencia LMF100, que son los filtros de condensador conmutado correspondientes a las etapas de prefiltrado y posfiltrado, en la parte izquierda.
- Un regulador de voltaje a 3.3v con sus respectivos condensadores de acople y desacople y diodos de protección contra polarizaciones inversas, alimentado a con un voltaje mayor o igual a 5v para un adecuado proceso de regulación, ubicado en la parte superior izquierda.
- Un circuito integrado de referencia LM386, un amplificador de audio de baja potencia con sus respectivos componentes externos de funcionamiento, al lado derecho del microcontrolador.
- Dos circuitos de amplificador operacional que corresponde a las etapas cambiadoras de nivel, en la parte superior derecha del microcontrolador.
- Un conector micro USB que además de utilizarse para respectiva comunicación, proporciona un voltaje de suministro de 5v provenientes de los puertos USB del PC.
- Un puerto externo de 3 pines al lado superior del Jack de potencia, conectada en sus terminales externas a los dos voltajes de suministro del sistema y su pin interno conectado a la fuente de alimentación de 5v del sistema, utilizada principalmente para elegir el tipo de alimentación externa por medio de un jumper.
- Un led y una resistencia conectados al 5v, usados como test de voltaje de entrada, ubicados en la parte superior izquierda del microcontrolador.
- Un filtro pasa bajas RC en paralelo usado para eliminar frecuencias parásitas que se pueden filtrar por la conexión de tierra del puerto USB, ubicado a la derecha del led y la resistencia de test.
- Un puerto externo de 3x2, usado para la conexión de programadores externos, ubicado en la parte superior.
- Un puerto externo de 3 pines, conectado un su pin interno a la salida del DAC y en su parte superior a la entrada de la etapa de posfiltrado. Usado principalmente para aislar la salida del DAC, y utilizarse para aplicaciones de propósito general, ubicado a la izquierda del puerto de programación externa.
- Seis puertos de conexión externa conectados a todos los pines del microcontrolador, principalmente para aplicaciones de propósito general, ubicados al lado derecho del microcontrolador.

La explicación anterior corresponde a la ubicación de los circuitos que integran el diagrama general del prototipo.

3.1.4 Implementación del diagrama general y Circuito impreso del Prototipo (PCB).

Luego de completar la parte del diseño y la estructura del proyecto, es necesario llevarlo a un circuito impreso que garantice un óptimo funcionamiento y por supuesto ayude a la portabilidad del dispositivo. El circuito se implementó inicialmente en una protoboard para verificar el

funcionamiento de cada etapa y el funcionamiento acoplado de todo el sistema. La figura 16 muestra el prototipo implementado inicialmente.

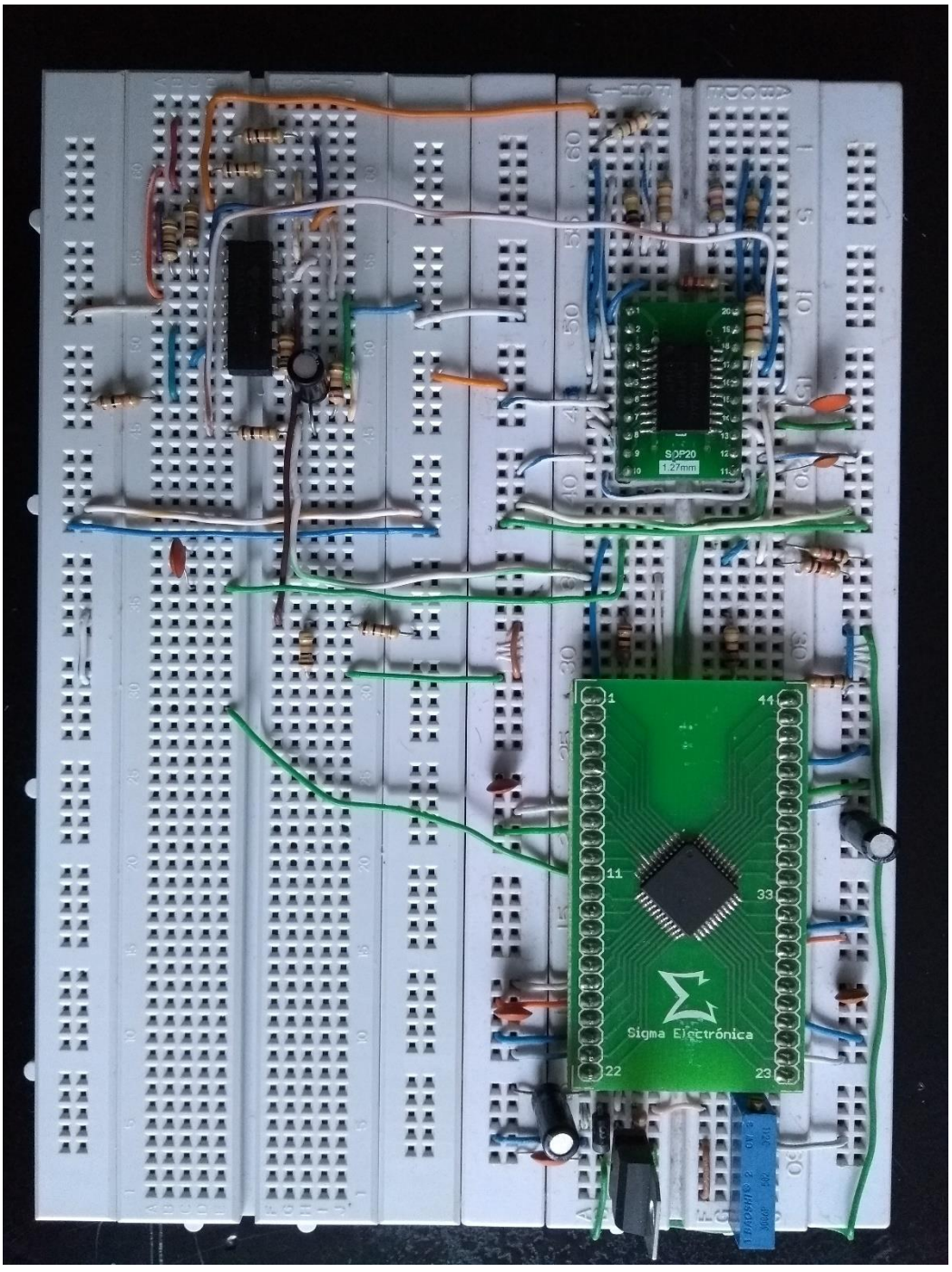


Figura 17. Implementación en protoboard del diagrama general. Fuente: Autor.

Se observa que en la parte superior de está ubicado el microcontrolador, algo particular es que es un dispositivo de montaje superficial (SMD), lo cual proporciona grandes ventajas con respecto al tamaño y la cantidad de potencia consumida para su funcionamiento. Otro integrado con estas mismas condiciones es el LMF100, mostrado en la parte superior izquierda. Las ventajas que

ofrece este tipo de tecnología de encapsulamiento con respecto a la tradicional de tecnología de agujeros pasantes (THT), son bastantes buenas debido al aprovechamiento del espacio en una PCB, que ahora se diseñan a ambas caras. Estas condiciones se usaron para decidir implementar todos los dispositivos con tecnología SMD que son más económicos y por supuesto un tamaño mucho menor.

A partir de este criterio se procede a diseñar el circuito impreso del proyecto, se decide usar la versión gratuita de ALTIUM DESIGNER, la cual se puede descargar desde la página oficial. Este software facilitó en gran medida el diseño del PCB. La figura 17 evidencia el circuito impreso terminado.

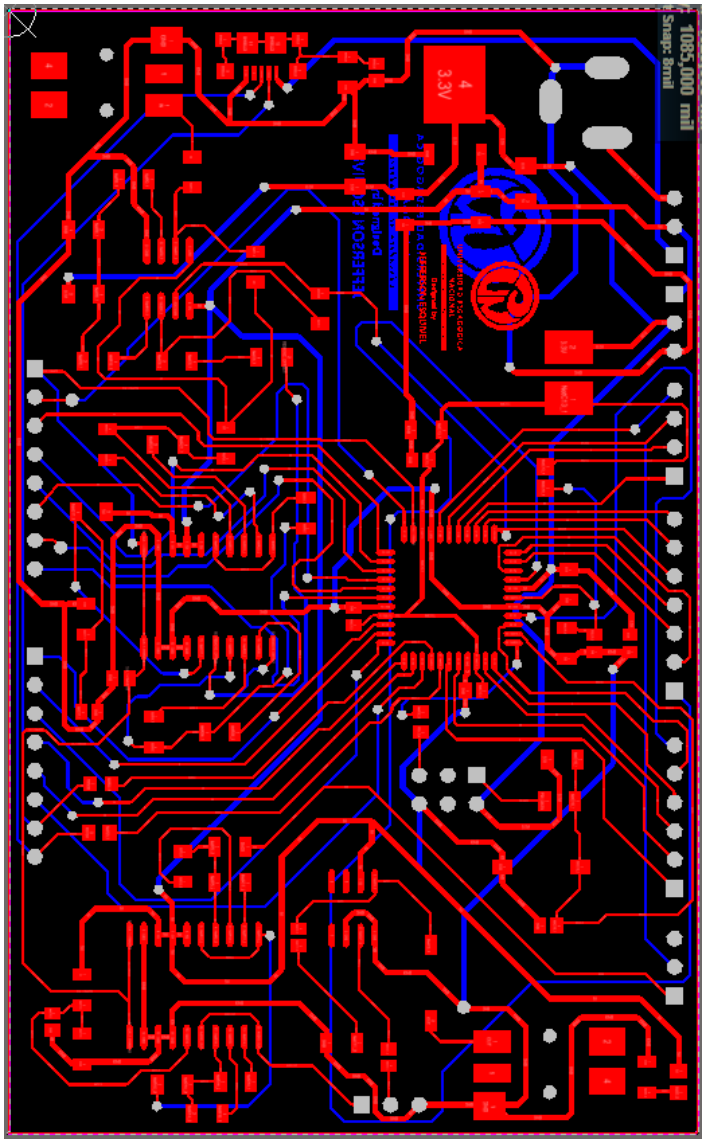


Figura 18. PCB del prototipo.

Se observa dos capas en el PCB, una de color rojo y otra de color azul, las cuales corresponde a la Top Layer y Bottom Layer respectivamente. La Top Layer es la capa superficial donde va la

ubicación de todos los componentes, desde los más pequeños (resistencias y condensadores), hasta los más grandes (circuitos integrados). La Bottom Layer es la capa inferior que se ven color azul, esta se usa principalmente para realizar conexión entre puntos que en la parte superior no es posible debido a cruces de caminos, todo esto por medio de un "pad via" en color gris. Las dimensiones de la placa son 6 x 10 cm^2. Con este apartado, se da por terminado la etapa de diseño del prototipo, para posteriormente pasar a la etapa de construcción.

3.2 Construcción y prueba del prototipo a partir de los elementos y componentes seleccionados en la etapa de diseño

3.2.1 Construcción.

Como se indicó anteriormente, el tipo de componentes seleccionados son de tipo SMD, que brinda facilidades ya mencionadas. Para construir el dispositivo hay que realizar un listado de los elementos necesarios:

- Una placa de fibra de vidrio de 6 x 10 cm^2 donde esté impreso por ambas caras las respectivas Top y Bottom Layer.
- Una regleta macho.
- Dos regletas hembras.
- Un pulsador.
- Un Jack de potencia.
- Dos Jack de audio.
- Una regleta de 2x3.
- Un regulador de voltaje a 3.3v.
- Un amplificador operacional LM358.
- Dos filtros de condensador conmutado LMF100.
- Un amplificador de audio de baja potencia LM386.
- Un microcontrolador ATXMEGA32A4U.
- Un conector micro USB.
- Un led rojo 0805 (2 x 1.25 mm^2).
- Resistencias y condensadores definidos en la etapa de diseño, con tamaños de encapsulador 1206 (3 x 1.5 mm^2) y 0805 respectivamente.

A continuación, mostraremos la PCB terminada lista para ensamblar los componentes.

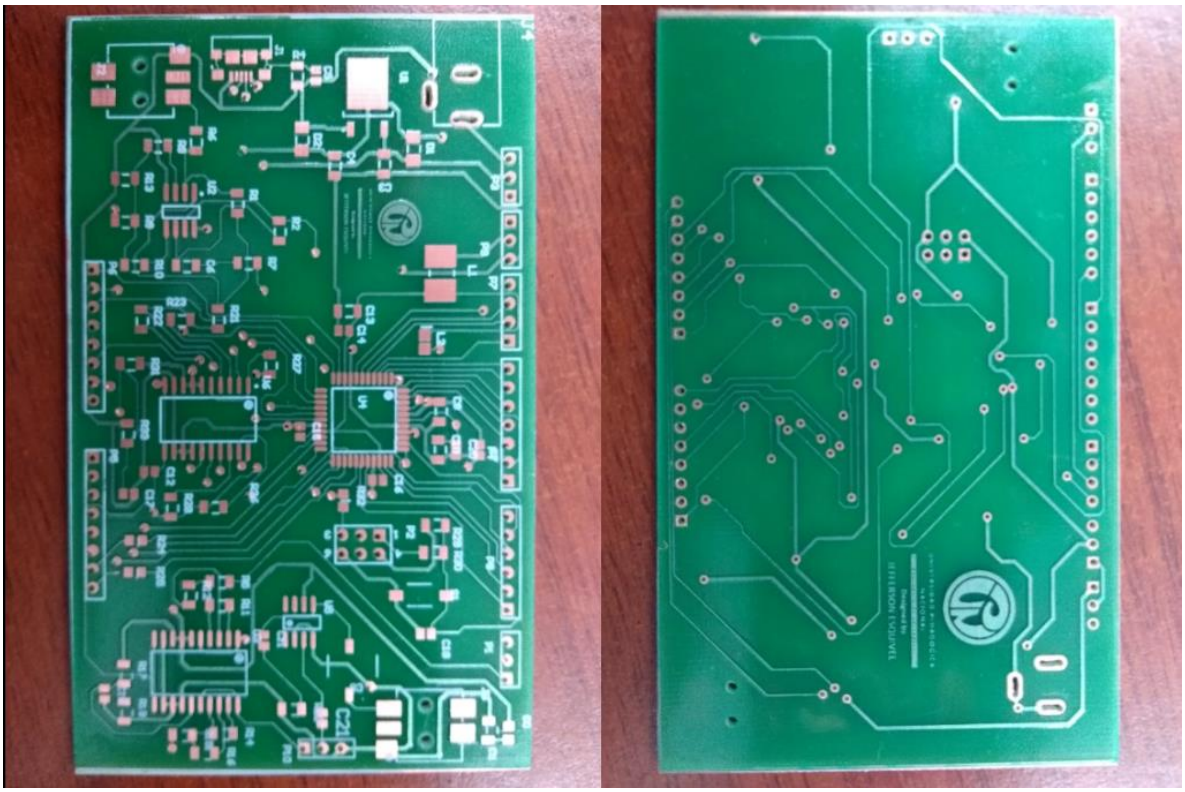


Figura 19. PCB del prototipo impreso. Lado izquierdo, Top Layer; lado derecho, Bottom Layer

En la figura 19 se observa las dos capas de la PCB que corresponde a la Top Layer y la Bottom Layer de la placa. Hay una capa de recubrimiento que ayuda a la preservación y evitar la oxidación del cobre; además, esta capa es muy conveniente al momento de soldar los componentes, puesto que evita que la soldadura se difunda por los caminos de la placa.

Figura 20. Prototipo terminado

Posterior al terminado de la PCB, procede soldar los componentes electrónicos que mostramos en la lista, para dar por terminado la construcción del prototipo. La figura 20 muestra el PCB soldado y terminado en su totalidad.

3.2.2 Prueba del prototipo.

Al terminar la construcción del prototipo, se procede a realizar las primeras pruebas a partir de señales de varias frecuencia y amplitud. El propósito en gran medida es corroborar el funcionamiento de todas las etapas del prototipo en conjunto, a partir de la aplicación de señales de audio y osciloscopio.

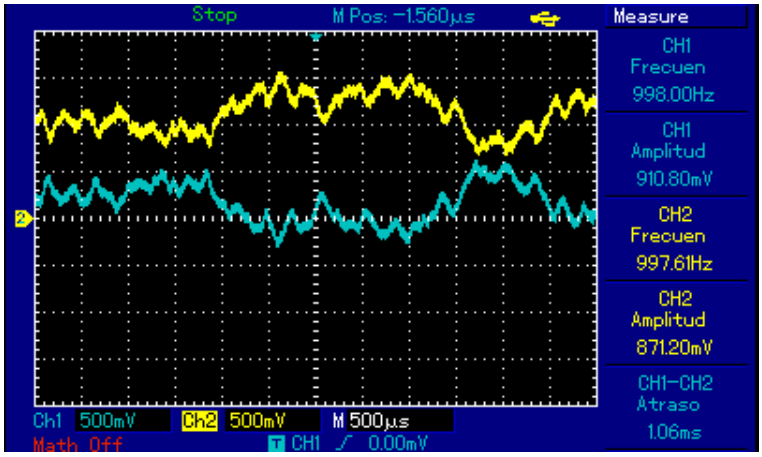


Figura 21. Señal de audio de entrada y salida del sistema. Osciloscopio U-NIT UPN.

En la figura 21, color azul señal de entrada y color amarillo salida del microcontrolador, se observa con claridad que la señal de salida está montada sobre la mitad de referencia de conversión del DAC (1.65v), e invertida en su forma. Se evidencia fácilmente que la forma de la señal se mantiene en gran proporción con respecto a la entrada. Con este resultado se comprueba un buen funcionamiento del prototipo en relación con la velocidad de muestreo y la capacidad de convertir la muestra de vuelta a la salida antes de presentarse el próximo intervalo de muestra.

3.3 Implementación de algoritmos de programación orientado a la aplicación de filtros digitales, con el fin de comprobar el desempeño del sistema a señales de audio.

En este apartado, se pondrá a prueba el funcionamiento del sistema terminado a señales de audio en tiempo real, implementando filtros digitales de tipo FIR e IIR, mostrando los resultados en cuanto al diseño y formulación matemática, y por supuesto las características y ventajas. Antes de proceder a empezar con la explicación del diseño de los algoritmos, se explicará cómo fue el proceso para realizar la programación del microcontrolador y la herramienta que se utilizó para dicho proceso.

3.3.1 Ambiente de programación del microcontrolador.

El microcontrolador ATXMEGA32A4U se programa por una interfaz de tipo Interfaz de Programación y Depuración (PDI), de propiedad de Atmel para dispositivos externos. Soporta grandes velocidades de programación para memorias de tipo no volátil (NVM), memorias Flash, EEPROM y Fusibles internos de dispositivos. Existen varios dispositivos de programación que soportan este tipo de interfaz. Atmel sugiere en gran medida dos dispositivos para realizar esta función: Atmel ICE y ATAVR Dragon. Para este proyecto se usó la tarjeta de programación ATAVR Dragon, por su relación en cuanto a costo y beneficio. Para entrar al modo de programación, Atmel sugiere desde su página oficial el uso de un compilador libre con grandes facilidades y múltiples recursos en cuanto a la programación y la depuración en tiempo real, este software tiene como nombre Atmel Studio 7, que fue en su totalidad la herramienta utilizada para implementar los algoritmos de programación necesarios para verificar el funcionamiento de las diferentes etapas referentes al prototipo.

3.3.1.1 *Herramientas de programación.*

3.3.1.1.1 *ATAVR Dragon.*

La tarjeta ATAVR Dragon es un dispositivo que soporta todas las interfaces de programación para la familia de dispositivos AVR, incluida la PDI. La figura 22 muestra una vista simple del dispositivo.

Figura 22. ATAVR Dragon. Fuente: The AVR Dragon User Guide, pag 1.

Como se explicó anteriormente, la programación PDI es realizada por medio de dos pines, CLK y DATA mostrados en la figura 23.

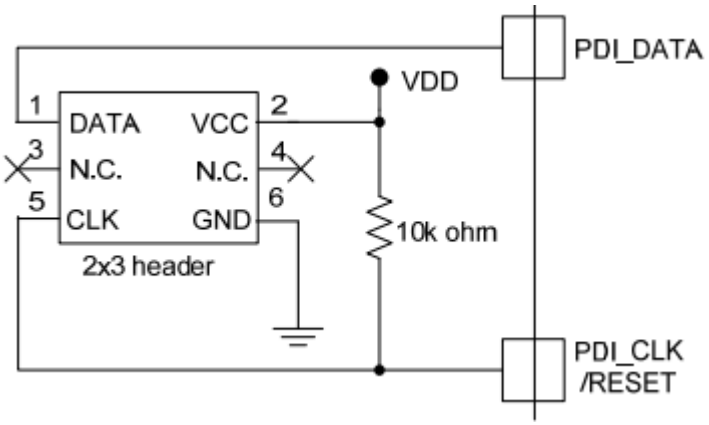


Figura 23. Pines de la interfaz de programación. Fuente: The AVR Dragon User Guide.

Se observa la ubicación de los pines de conexión para la programación del microcontrolador, PDI DATA y PDI CLK corresponde a los pines 34 y 35 respectivamente.

3.3.1.1.2 *Atmel Studio 7.*

Atmel Studio 7 es un compilador de uso libre desarrollado por Atmel para la programación de sus dispositivos. Este software se puede descargar de forma gratuita en la página principal de la compañía. La figura 24 muestra el área de trabajo.

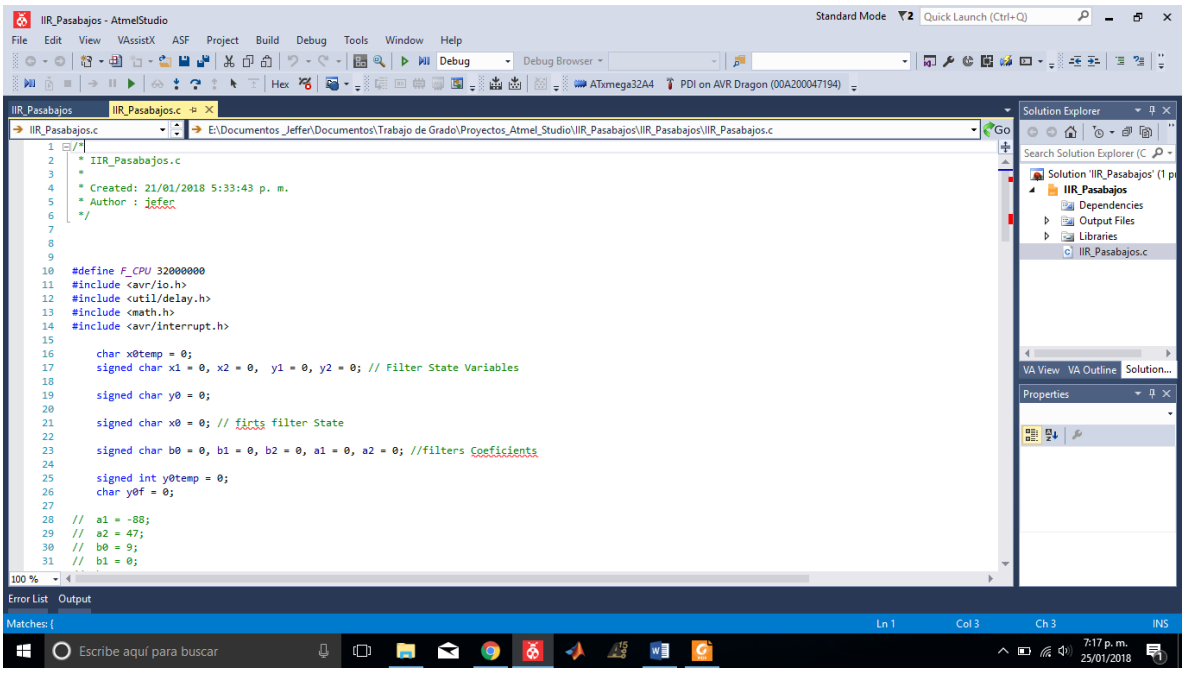


Figura 24. Área de trabajo Atmel Studio 7.

En la parte central se observa el área donde se modifica el código fuente de un proyecto, al lado derecho una ventana de exploración de los documentos abiertos. Junto a esta área existe otra interfaz donde se realiza la lectura y escritura de los dispositivos a programar. La figura 25 expone la interfaz de programación.

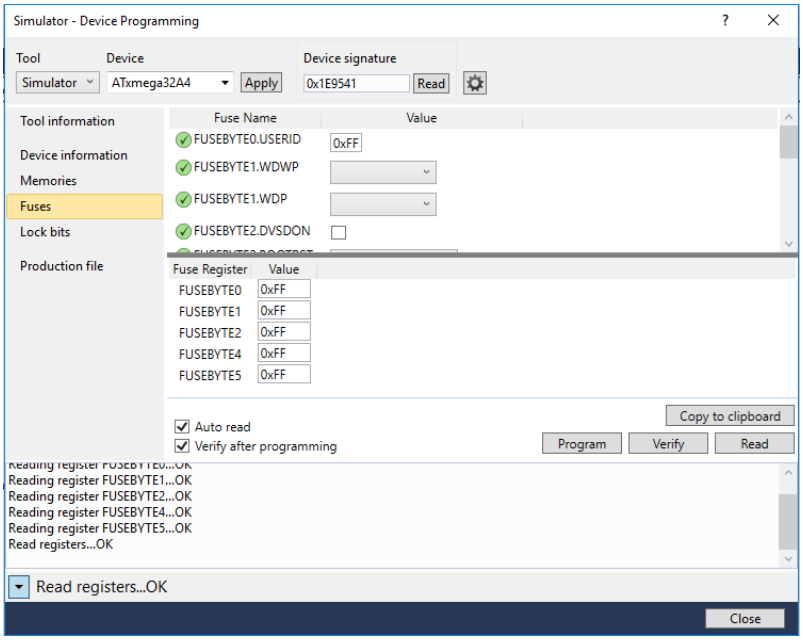


Figura 25. Interfaz de programación Atmel Studio 7

Se observa en la parte superior una barra con varias opciones desplegables, donde se muestra de izquierda a derecha la herramienta de programación, el tipo de dispositivo y el código

identificador. En la parte izquierda están las opciones para seleccionar que tipo de memoria y archivo donde está los códigos a programar del proyecto.

3.3.2 Algoritmos de programación.

Los algoritmos de programación de este trabajo están orientados a la aplicación de señales de audio, específicamente filtros digitales. Como se mencionó con anterioridad un filtro es un sistema que se encarga de suprimir señales con características de frecuencia establecidas. Específicamente, un filtro digital tiene una función de transferencia en términos del dominio de "Z", específicamente el operador de la trasformada Z. La transformada Z de una señal en tiempo discreto está mostrada en la ecuación (7):

$$Y(z) = \sum_{n=0}^{\infty} y(n)z^n \quad (7)$$

Siendo $y(n)$ la función de salida de un sistema discreto, escrito por medio de la ecuación (8):

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) \ldots . + a_mx(n-m) \quad (8)$$

Siendo $x(n)$ la señal de entrada al sistema y "m" indica las unidades de retardo de la señal de entrada y el orden de la ecuación. La función de transferencia del sistema anterior está descrita por medio de la ecuación (9):

$$H(z) = \frac{y(z)}{x(z)} \quad (9)$$

La ecuación (9) es la forma apropiada para representar los sistemas estudiados posteriormente.

Para el caso de este trabajo se procederá a diseñar de forma digital filtros de tipo pasabajas, pasaaltos, pasabanda y rechazabanda. Inicialmente tenemos el filtro IIR pasabajas.

3.3.2.1 *Diseño de un filtro IIR Pasabajos.*

El diseño del filtro pasabajas es muy relacionado al diseño en tiempo continuo de un filtro pasabajas de tipo Butterworth. Los filtros Butterworth se caracterizan por ser los más parecidos al filtro ideal en la banda de paso y por ser también los más sencillos de implementar. La función de transferencia para un filtro pasabajas de segundo orden está descrita por ecuación (10):

$$H_c(s) = \frac{GW_n^2}{S^2 + 1.4142W_nS + W_n^2} \quad (10)$$

Siendo G la ganancia del filtro y W_n la frecuencia de corte. Como el filtro está diseñado en el dominio del tiempo continuo, hay que buscar un método para levarlo al tiempo discreto. Existen muchas formas para realizar este procedimiento, pero el método más sencillo y claro escogido en este trabajo es el llamado Transformación Bilineal. La transformación Bilineal es un método

algebraico entra las variables "S" y "Z", que reemplaza todo el plano complejo de S en un plano de circunferencia unitaria de Z. Siendo $H_c(s)$ la función de transferencia del filtro en tiempo continuo, debemos llevarlo a una representación en tiempo discreto $H(z)$, esto se consigue aplicando la ecuación (11):

$$S = \frac{2}{T}\frac{(1 - Z^{-1})}{(1 + Z^{-1})} \quad (11)$$

Donde T es el periodo de muestreo. Reemplazando la ecuación (11) en (10) y simplificando obtenemos la (12):

$$H(z) = \frac{b_0 + b_1 Z^{-1} + b_2 Z^{-2}}{1 + a_1 Z^{-1} + a_2 Z^{-2}} \quad (12)$$

Donde $b_0 = \frac{GW_n^2T^2}{4+2T\sqrt{2}W_n+T^2W_n^2}$; $b_1 = \frac{2GW_n^2T^2}{4+2T\sqrt{2}W_n+T^2W_n^2}$; $b_2 = \frac{GW_n^2T^2}{4+2T\sqrt{2}W_n+T^2W_n^2}$;

y $a_1 = \frac{2(T^2W_n^2-4)}{4+2T\sqrt{2}W_n+T^2W_n^2}$; $a_2 = \frac{4-2T\sqrt{2}W_n+T^2W_n^2}{4+2T\sqrt{2}W_n+T^2W_n^2}$;

Cómo la Transformación Bilineal no produce una relación perfecta entra las frecuencias analógicas y digitales, especialmente para altas frecuencias, es recurrente realizar la siguiente transformación. Sabiendo que

$$Wn = 2\pi f_n \quad (13)$$

y f_n es la frecuencia de corte del filtro, encontramos el valor de f_d en las siguientes ecuaciones:

$$S = j2\pi f_d \quad (14)$$

$$Z = e^{2\pi f_n} \quad (15)$$

Reemplazando (14) y (15) en la ecuación (11), obtenemos (16):

$$j2\pi f_d = \frac{2}{T}\frac{(1 - e^{-j2\pi f_n})}{(1 + e^{-j2\pi f_n})} \quad (16)$$

Simplificando y despejando el valor de f_d, tenemos:

$$f_d = \frac{\tan(\pi T f_n)}{\pi T} \quad (16)$$

Ahora, a partir de (16) hallamos nuestro nuevo valor de frecuencia de corte discreta f_d. Reemplazando el valor de (16) en (13), nos da como resultado (17):

A partir de este nuevo valor de W_n encontramos los valores respectivos para los coeficientes del filtro.

Posterior al proceso del cálculo de los coeficientes, nos es recurrente hallar una representación de la función de salida en el tiempo discreto.

Llevando la ecuación (12) al dominio del tiempo discreto obtenemos (19):

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \quad (19)$$

Podemos observar que la salida del sistema está en términos de sus valores anteriores, el valor actual y los valores pasados de la entrada. Claramente se evidencia una realimentación de la salida hasta dos valores pasados, característica principal de los sistemas de tipo IIR.

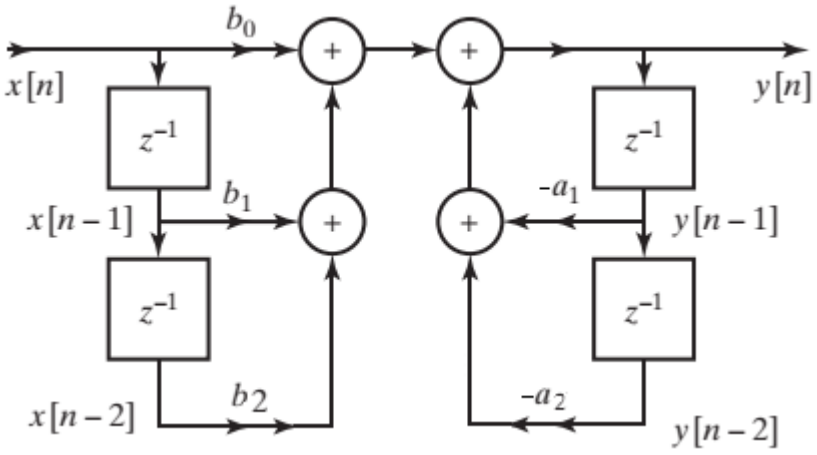


Figura 26. Diagrama de bloque de la ecuación 18 en forma directa 1

En el diagrama de bloque se observa claramente que el sistema tiene una realimentación negativa que lo caracteriza como un filtro de tipo IIR. La representación final del filtro pasabajas de segundo orden se implementa gracias a la ecuación en diferencia. En secciones posteriores presentaremos el proceso para implementar el filtro en el microcontrolador, este proceso requiere de unos pasos y condiciones que se explicarán en la sección de implementación de los filtros.

3.3.2.1.1 *Simulación del filtro IIR pasabajos.*

En esta sección se procederá a realizar la simulación del filtro pasabajas diseñado anteriormente. La frecuencia de corte f_n es 1000 Hz y la frecuencia de muestreo f_s es 20 kHz para un periodo de muestreo T igual a 50 microsegundos. Reemplazamos estos valores en las ecuaciones anteriores y obtenemos:

$$H_c(s) = \frac{6283.18}{S^2 + 8885.76\,S + 6283.18} \quad (10)$$

Que es la función de transferencia del filtro en tiempo continuo. La función de transferencia del filtro de tiempo discreto es la siguiente:

$$H(z) = \frac{0.0201 + 0.0402\,Z^{-1} + 0.0201\,Z^{-2}}{1 - 1.561\,Z^{-1} + 0.6414\,Z^{-2}} \qquad (12)$$

Y su correspondiente ecuación en diferencia es:

$$y(n) = 1.561y(n-1) - 0.6414y(n-2) + 0.02016x(n) + 0.0402x(n-1) + 0.0201x(n-2)$$

Para realizar la simulación se utiliza una versión de prueba gratuita de 30 días de Matlab la cual fue descargada desde https://la.mathworks.com/programs/trials/trial_request.html. Inicialmente probamos la respuesta del sistema a una función de entrada con 100 Hz, luego a una función de 1000 Hz y luego a una de 3000 Hz. Los respectivos códigos implementados se encuentran como anexos a este documento. Usando la función plot obtenemos:

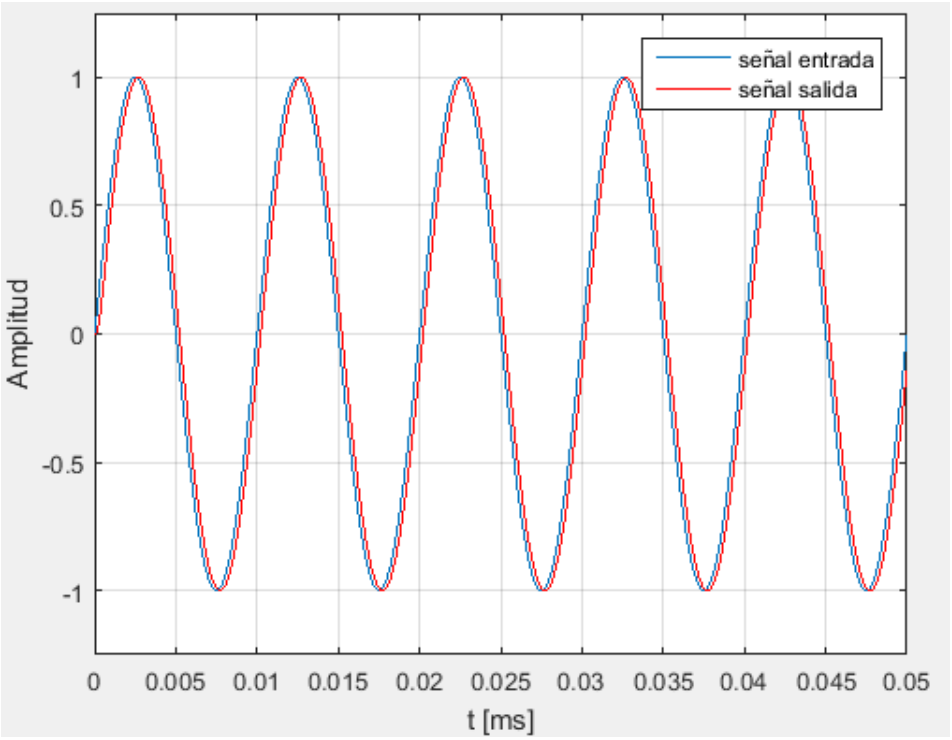


Figura 27. Respuesta del filtro a una señal de 100 Hz.

Se observa fácilmente en la figura 27, para una frecuencia de entrada de 100 Hz, que la señal de salida es casi idéntica a la de entrada a excepción de una pequeña diferencia en amplitud y fase. La ganancia y la forma de onda se mantiene puesto que la frecuencia de la señal está muy por debajo de la frecuencia de corte.

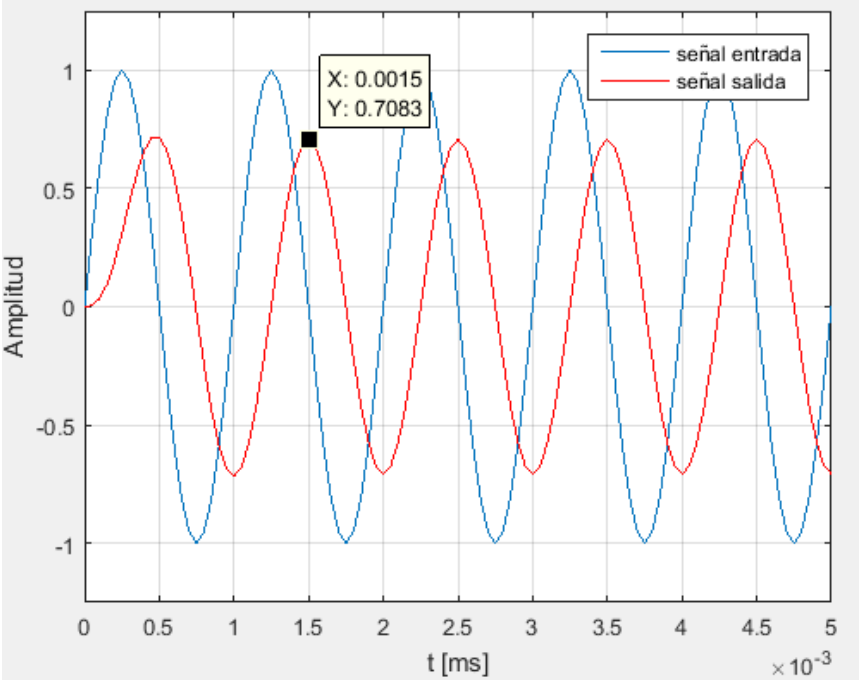


Figura 28. Respuesta del filtro a una señal de 1000 Hz.

La amplitud de la señal de salida ha cambiado significativamente (para una frecuencia de entrada de 1000Hz) con respecto a su entrada, puesto que su amplitud cae del 70 %, característica principal de los filtros de tipo Butterworth, en su frecuencia de corte.

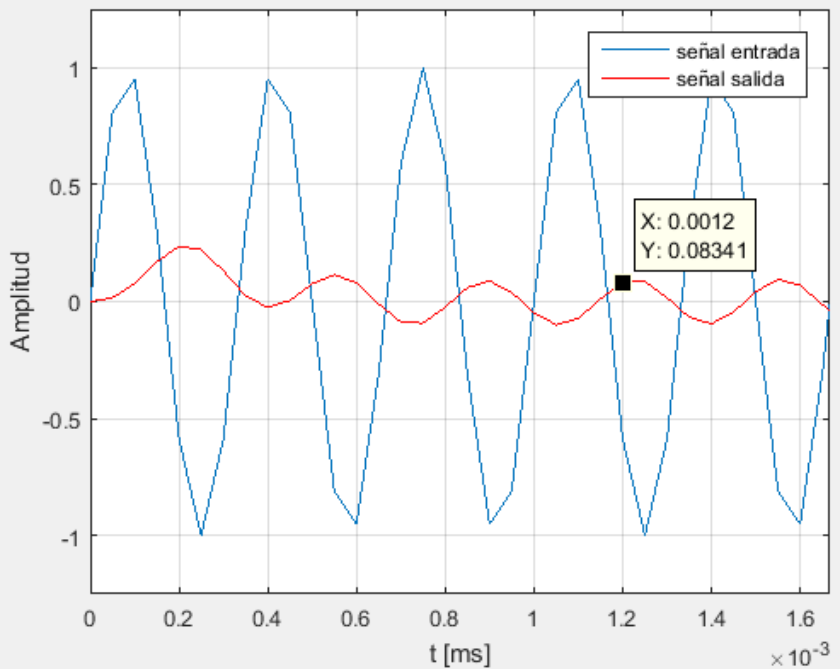


Figura 29. Respuesta del filtro a una entrada 3000 Hz.

Se evidencia de forma clara que la amplitud de la señal de salida disminuye significativamente (con una frecuencia de entrada de 3000 Hz), debido a que el filtro es de segundo orden y la

frecuencia de entrada se encuentra en la banda de rechazo del filtro. También se observa que la forma de onda de la señal cambia un poco, debido a los efectos de la frecuencia de muestreo de 20 kHz. La figura 30 muestra la respuesta en frecuencia del filtro

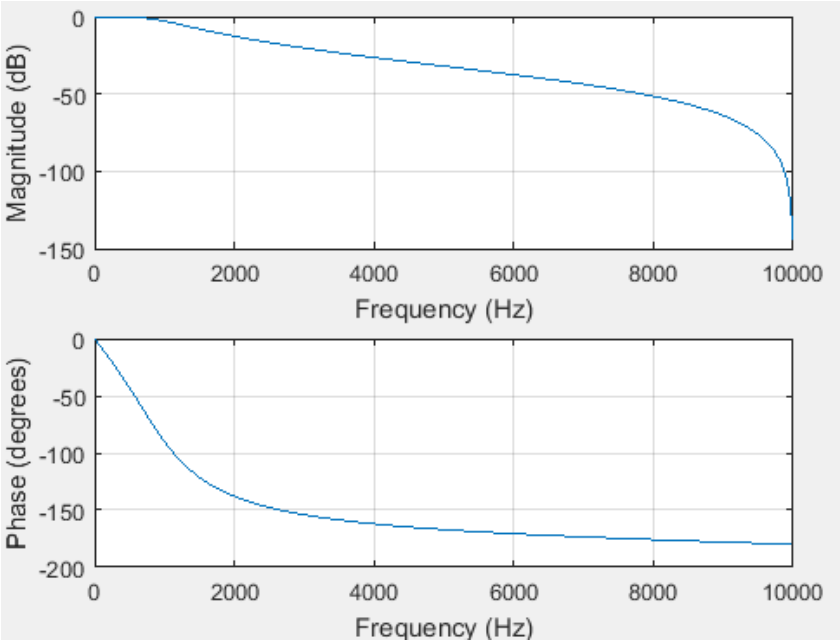


Figura 30. Respuesta en frecuencia para el IIR pasabajos de 1kHz

La figura 30 evidencia el comportamiento con respecto al cambio de la frecuencia de la entrada. Se evidencia un buen comportamiento en la simulación, datos que se compararán significativamente con los resultados en la implementación.

3.3.2.2 *Diseño de un filtro FIR pasabajos.*

Metodologías de diseño para filtros de tipo FIR existen muchísimas, y ellas varían con respecto al tipo de ventana que se quiere implementar. El diseño para este proyecto será con base a un método llamado Respuesta al Impulso del Sistema, el cual como su nombre lo indica, se le aplica una señal de entrada de tipo impulso a un filtro diseñado de forma discreta, es decir, se implementa a partir del filtro IIR pasabajas realizado anteriormente, y se toman muestras de la señal de salida para posteriormente encontrar los coeficientes del filtro.

Ahora, partimos de la función de transferencia en tiempo discreta encontrada para el filtro IIR pasabajas.

$$H(z) = \frac{0.0201 + 0.0402\,Z^{-1} + 0.0201\,Z^{-2}}{1 - 1.561\,Z^{-1} + 0.6414\,Z^{-2}} \qquad (12)$$

Con ayuda de la función impulse de MATLAB, encontramos la respuesta al impulso de este sistema y graficamos la señal de salida para un tiempo de 2 milisegundos.

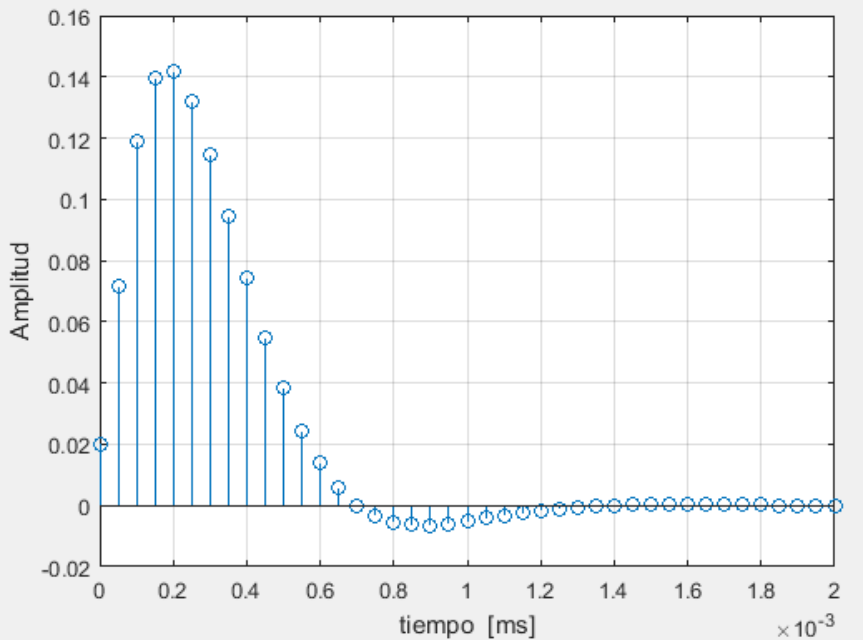


Figura 31. Respuesta al impulso del filtro IIR pasabajas.

De la gráfica se observa que los valore tomados para t > 1ms son muy aproximados a 0. Se tomarán los primeros 20 valores de la señal para posteriormente ser analizados en la implementación. Los valores de salida hasta el instante de muestreo 20 son:

y = [0.0201 0.0715 0.1188 0.1396 0.1418 0.1317 0.1147 0.0946 0.0741 0.0550 0.0383 0.0245 0.0137 0.0057 0.0001 -0.0035 -0.0055 -0.0064 -0.0064 -0.0059]

Los valores de y son los respectivos coeficientes del filtro FIR pasabajas para un periodo de muestreo de 20 kHz. La función de transferencia del filtro se representa de con la ecuación (19):

$$H(z) = b_0 + b_1 Z^{-1} + b_2 Z^{-2} + b_3 Z^{-3} + b_4 Z^{-4} + b_5 Z^{-5} + b_6 Z^{-6} + b_7 Z^{-7} + b_8 Z^{-8} + b_9 Z^{-9} + b_{10} Z^{-10} + b_{11} Z^{-11} + b_{12} Z^{-12} + b_{13} Z^{-13} + b_{14} Z^{-14} + b_{15} Z^{-15} + b_{16} Z^{-16} + b_{17} Z^{-17} + b_{18} Z^{-18} + b_{19} Z^{-19} \quad (19)$$

Donde los valores respectivos del filtro son los representados en el vector y. La representación en el dominio del tiempo discreto de la función de transferencia está dada por la ecuación (20):

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) + b_3x(n-3) + b_4x(n-4) + b_5x(n-5) + b_6x(n-6) + b_7x(n-7) + b_8x(n-8) + b_9x(n-9) + b_{10}x(n-10) + b_{11}x(n-11) + b_{12}x(n-12) + b_{13}x(n-13) + b_{14}x(n-14) + b_{15}x(n-15) + b_{16}x(n-16) + b_{17}x(n-17) + b_{18}x(n-18) + b_{19}x(n-19) \quad (20)$$

La ecuación 20 muestra claramente que un filtro de tipo FIR no contiene realimentación de su salida. Esta ecuación puede ser llevada claramente a una representación en diagramas de bloque.

En la sección de implementación de este filtro, se verificará si es posible la simplificación de esta ecuación puesto que muchos coeficientes son valores muy aproximados a 0.

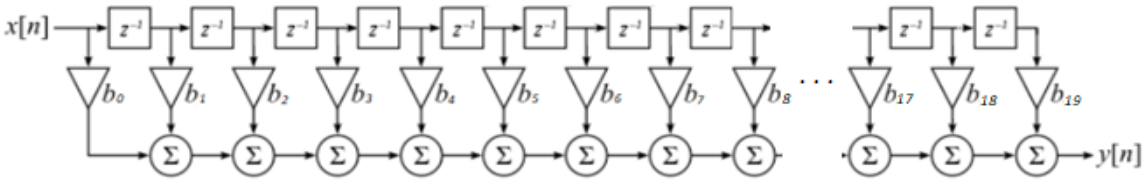


Figura 32. Diagrama de bloque para la ecuación (20).

En el diagrama existe 20 bloques de multiplicación, los cuales representan los coeficientes del filtro y los otros representan los retrasos temporales de la entrada.

3.3.2.2.1 *Simulación del filtro FIR pasabajos*

Al tener la función de transferencia y la ecuación en diferencias referente al filtro FIR, se procede a realizar la simulación del filtro pasabajas. El filtro tendrá las mismas características que el IIR diseñado anteriormente, con una frecuencia de corte f_n igual a 1000 Hz y la frecuencia de muestreo f_s de 20 kHz para un periodo de muestreo T igual a 50 microsegundos. Diseñamos el programa de simulación en Matlab y usamos la función plot para graficar la salida del sistema.

A continuación, se mostrará la respuesta del filtro FIR para 3 señales con frecuencia de 500 Hz, 1000 Hz, 3000 Hz.

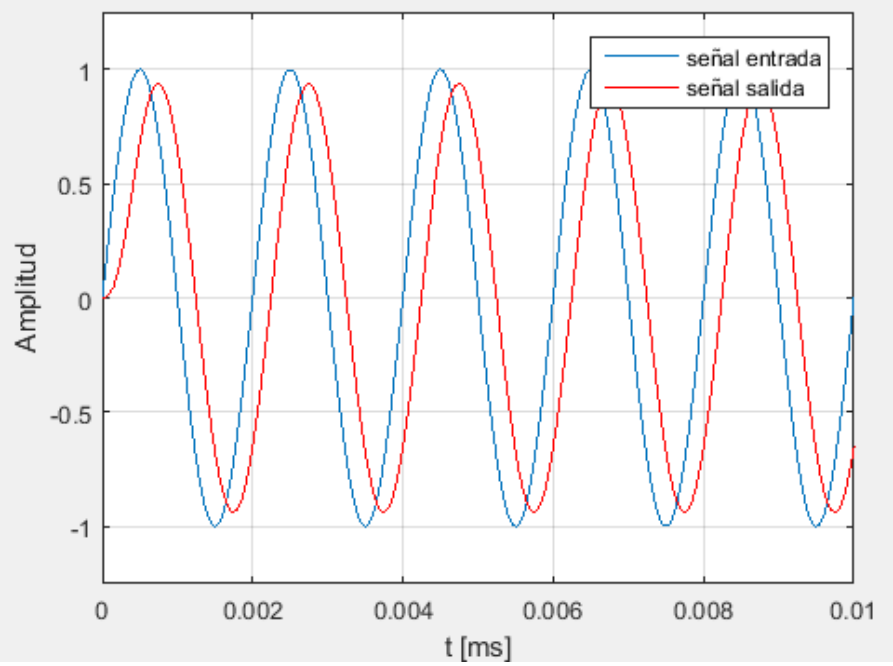


Figura 33. Respuesta del filtro a una frecuencia de 500 Hz.

Se observa que la amplitud de la señal no varía mucho con respecto a la entrada, manteniendo las características del filtro pasabajas.

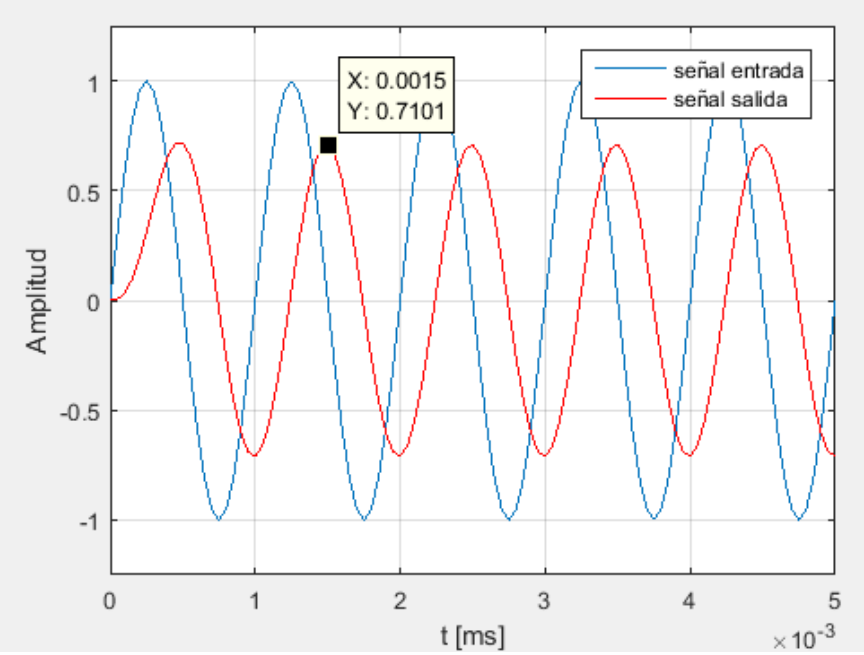


Figura 34. Respuesta del filtro FIR a una frecuencia de 1000 Hz.

Cómo es de esperar, la amplitud de la señal cae al 70% aproximadamente por ser un filtro Butterworth en su frecuencia de corte.

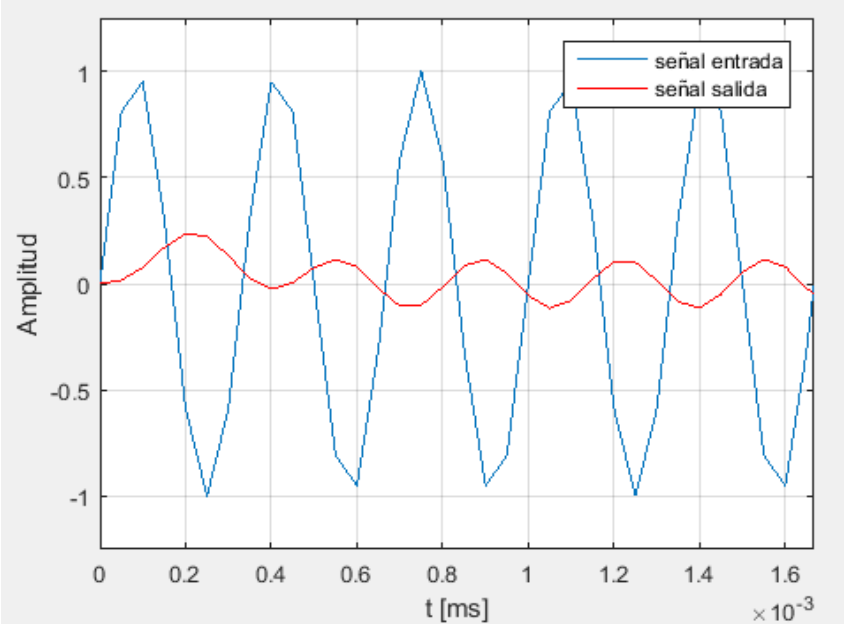


Figura 35. Respuesta del filtro a un señal de 3000 Hz.

La atenuación de la señal es evidente por tener una frecuencia que en la banda de rechazo. A continuación, se muestra el digrama de respuesta en frecuencia para el filtro FIR pasabajas a 1000 Hz.

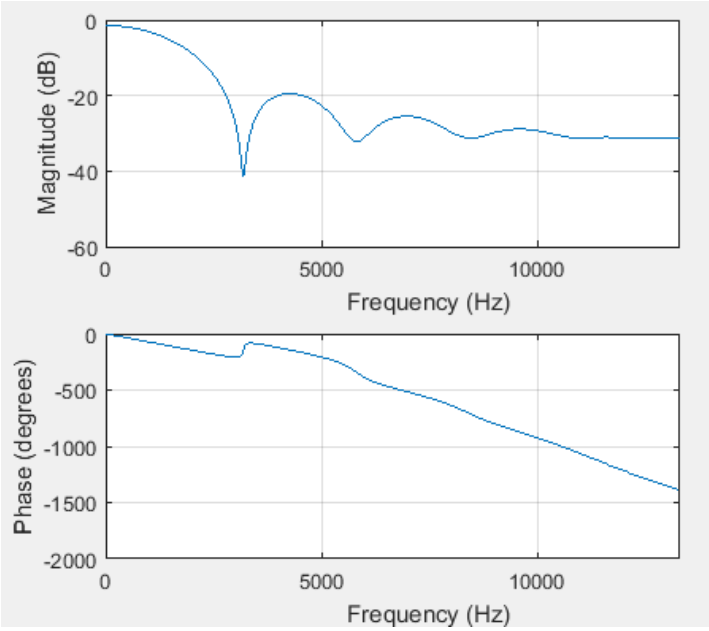


Figura 36. Respuesta en frecuencia de amplitud y fase para el filtro FIR

Característica importante que vale la pena resaltar, es la fase lineal que presenta este tipo de filtros, debido a que la señal de salida tiene un corrimiento en su fase a una razón constante, siempre y cuando la señal de entrada se encuentre en la banda de paso, es decir antes de la frecuencia de corte para este filtro pasabajos.

3.3.3 Implementación de los algoritmos de programación.

El apartado anterior evidenciaba el proceso y los resultados simulados para dos filtros pasabajos de tipo Butterworth con arquitectura IIR y FIR respectivamente. El prototipo de este trabajo se diseñó principalmente para implementar los algoritmos realizados en la simulación y así poder contrastar los resultados teóricos con los prácticos. Antes de proceder realizar los archivos de programación en el microcontrolador, es necesario aclarar unos conceptos respecto al tipo de representación numérica de los datos.

Existe dos tipos de representación para trabajar datos de forma digital, uno es la representación en **punto fijo** y la otra es la representación en **punto flotante.** Las arquitecturas de microcontroladores de 32 y 64 bits usan la representación de punto flotante, debido a su poderosa estructura en cuanto a resolución y precisión de datos al tener buses y CPU con un tamaño relativamente grande. En cambio, para arquitecturas de menos capacidad, puntualmente para el caso del ATXMEGA32A4U que es de 8 bits, es conveniente implementar una representación de punto fijo.

Los registros donde se albergan los datos que se usan para realizar las sumas y las multiplicaciones de los coeficientes con los datos de entrada, son de 8 bits. La resolución de un registro de este tamaño permite un conjunto de valores que pueden variar desde 0 hasta 255, es decir, datos de tipo entero. Cómo se vio anteriormente los coeficientes del filtro dan resultados de tipo decimal, por eso es conveniente realizar una conversión a entero.

3.3.3.1 Representación punto fijo.

La representación en punto fijo aumenta el rendimiento matemático y la tasa de ejecución (es decir, aumentar la velocidad de procesamiento), los cálculos se realizan utilizando dos representaciones de puntos fijos con signo por complemento a2. Se eligieron representaciones de punto fijo Q7 y Q6 de 8 bits para muestras de entrada y coeficientes de filtro, respectivamente, debido a la arquitectura de 8 bits. Los números Q7 pueden representar números de punto fijo que van de -1 a 0.9921875 en incrementos de 0.0078125 (-1 a 1 - 1/128). A continuación, se muestra la ponderación de bits de número Q7 de 8 bits. El lugar decimal está entre los bits 6 y 7. Las muestras de entrada están en un formato Q7.

s.	x	x	x	x	x	x	x
-1	1/2	1/4	1/8	1/16	1/32	1/64	1/128

La representación de 8 bits Q6 puede representar números de punto fijo que van desde -2 a 1.984375 en incrementos 0.015625 (-2 a 2 - 1/64). La ponderación del bit de representación Q6 se muestra a continuación. El lugar decimal está entre los bits 5 y 6. Los coeficientes de filtro están en un formato Q6.

s	x.	x	x	x	x	x	x
-2	1	1/2	1/4	1/8	1/16	1/32	1/64

Cuando un número en representación Q7 y Q6 son multiplicados (ambos de 8 bits), el resultado es un número de 16 bits representado en Q13. La representación Q13 tiene un rango de -4 a 3.999987 en incrementos de 1/8192. La representación Q13 se muestra a continuación.

s	x	x.	x	x	x	x	x	x	x	x	x	x	x	x
-4	2	1.	1/2	1/4	x	x	x	x	x	x	x	x	x	1/8192

El número Q13 de 16 bits se reduce a una representación Q7 para la salida del filtro digital. En este número Q7 está contenido en el bit de signo justo a la izquierda del lugar decimal y los siete bits justo a la derecha del lugar decimal. Estos bits Q7 se extraen desplazando el número Q13 de 16 bits hacia la derecha y luego seleccionando solo el byte inferior del valor de 16 bits. La representación numérica resultante de 8 bits se muestra a continuación.

s	x	x.	x	x	x	x	x	x	x	x	x	x	x
-4	2	1.	1/2	1/4	x	x	x	x	x	x	x	x	x	1/8192

8-bit Q7

Fácilmente este proceso se explica de la siguiente manera. Sabiendo que los coeficientes inicialmente están representados en forma decimal, se multiplican por un factor de 64 desplazándose el producto en 6 bits hacia la izquierda. Al final, el resultado se divide por 64, es decir se trunca 6 bits hacia la derecha, enviándose el valor finalmente al DAC.

3.3.3.2 Implementación del filtro IIR pasabajos a 1 kHz.

3.3.3.2.1 *Cálculo de los coeficientes en punto fijo.*

Por medio de Matlab encontraremos el respectivo valor de los coeficientes del filtro IIR pasabajos de segundo orden. En Matlab existe una función que convierte los valores decimales de los coeficientes en valores representados en punto fijo.

$fi\ (b0, 1, 8, 6)$, esta es la función fi, la cual en este caso toma el valor del coeficiente b_0 y convierte a número decimal (8 bits con 6 número para la parte decimal) más cercano al valor original, que se puede representar con 8 bits.

Luego usamos la función $int\ ()$, que entrega de forma entera el valor decimal para la representación en el formato establecido. Acompañado de este, está la función $bin()$ que muestra su respectivo valor equivalente en binario. Tomando estas funciones, procedemos a calcular el valor en representación en punto fijo.

Los valores decimales de los coeficientes son:

$$b_0 = 0.0201;\ \ b_1 = 0.0402;\ \ b_2 = 0.0201$$

$$a_1 = -1.561;\ \ a_2 = 0.6414$$

Aplicando las funciones

$b_0 = 0.0201;$ $b_0fp = fi(b_0, 1, 8, 6)$ $b_0fp = 0.0156;$ $bin(b_0fp) = 00000001;$ $int(b_0fp) = 1$, respectivo valor en punto fijo.

$b_1 = 0.0402;$ $b_1fp = fi(b_1, 1, 8, 6)$ $b_1fp = 0.0156;$ $bin(b_1fp) = 00000011;$ $int(b_1fp) = 3$, respectivo valor en punto fijo.

$b_2 = 0.0201;$ $b_2fp = fi(b_2, 1, 8, 6)$ $b_2fp = 0.0156;$ $bin(b_2fp) = 00000001;$ $int(b_2fp) = 1$, respectivo valor en punto fijo.

$a_1 = -1.561;$ $a_1fp = fi(a_1, 1, 8, 6)$ $a_1fp = -1.5625;$ $bin(a_1fp) = 10011100;$ $int(a_1fp) = -100,$ respectivo valor en punto fijo.

$a_2 = 0.6414;$ $a_2fp = fi(a_2, 1, 8, 6)$ $a_2fp = 0.6406;$ $bin(a_2fp) = 00101001;$ $int(a_2fp) = 41,$ respectivo valor en punto fijo.

Después de realizar la conversión a punto fijo los coeficientes quedan de la siguiente manera:

$$b_0fp = 1;\ \ b_1fp = 3;\ \ b_2fp = 1$$

$$a_1fp = -100;\ \ a_2fp = 41$$

Como se observa, todos los coeficientes quedaron convertidos a valores enteros fáciles de operar en el microcontrolador, puesto que al realizar operaciones de multiplicación y suma se gasta una buena cantidad de tiempo y recursos del hardware.

Al modificarse los valores de los coeficientes, la ecuación en diferencia del filtro cambia, quedando de la siguiente manera:

$$y(n) = -a_1fpy(n-1) - a_2fpy(n-2) + b_0fpx(n) + b_1fpx(n-1) + b_2fpx(n-2) \quad (21)$$

Para realizar la implementación del filtro en el microcontrolador, se debe asegurar un intervalo de muestreo bastante preciso. La frecuencia de muestreo para este caso se definió con un valor de 20khz. Puesto que la conversión del ADC se estableció de manera automática con una frecuencia máxima de muestreo de 1.33Ms/s para el caso de 12 bits de resolución, se obtiene una muestra del ADC por medio de una rutina de interrupción. Esto garantiza que se tome, se procese y se vuelva a convertir muestras en tiempos constantes. El respectivo código de programación para este filtro se anexo en las últimas páginas de este documento.

3.3.3.2.2 *Verificación de los resultados.*

Para realizar este proceso, se utilizó una herramienta proporcionada por la Universidad Pedagógica Nacional, la tarjeta de adquisición de datos My Daq de Nacional Instruments. Esta tarjeta tiene las facilidades de poder incluir funciones de osciloscopio y generador de funciones, ideal para verificar los resultados de los filtros. La implementación consta de aplicar una función de tipo sinusoidal a la entrada del circuito, por medio de un Jack de audio, el cual dirige la señal por las respectivas etapas del sistema mostradas en la figura 3. La salida es proporcionada por medio de otro Jack de audio, diseñado específicamente para aplicarse a amplificadores externos.

El ejercicio es el mismo realizado en la simulación, especialmente para comparar los resultados teóricos con los prácticos. Empezaremos con aplicar al sistema una señal de 300 Hz.

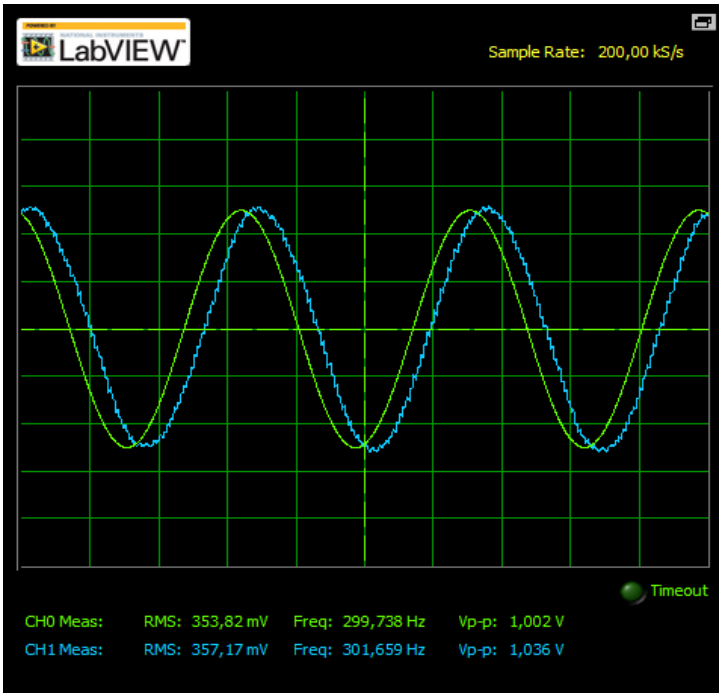


Figura 37. Señales de entrada y salida del filtro pasabajos IIR con fn 1kHz. Entrada a 300 Hz.

La Figura 37 evidencia claramente la respuesta del filtro, siendo la señal de entrada la función en color verde y la señal de salida la función en color azul. Se observa un desfase proporcionado por la respuesta en frecuencia del filtro, y la amplitud no presenta cambios significativos debido a que la frecuencia se encuentra en la banda de paso.

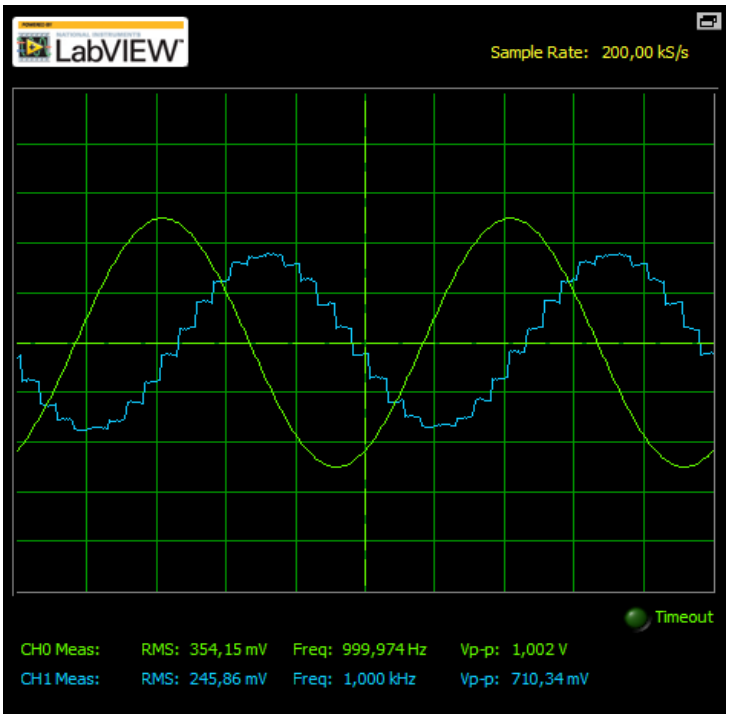


Figura 38. Señales de entrada y salida del filtro pasabajos IIR con fn 1kHz. Entrada a 1 kHz.

La figura 38 muestra información respecto a la amplitud de salida para una señal de 1 kHz, es fácil de observar que la amplitud es de 71% aproximadamente, con un error de 1.42 % con relación al valor teórico de 70%. También se aprecia los efectos de la frecuencia de muestreo de 20 kHz, efectos producidos por la conversión digital a analógica.

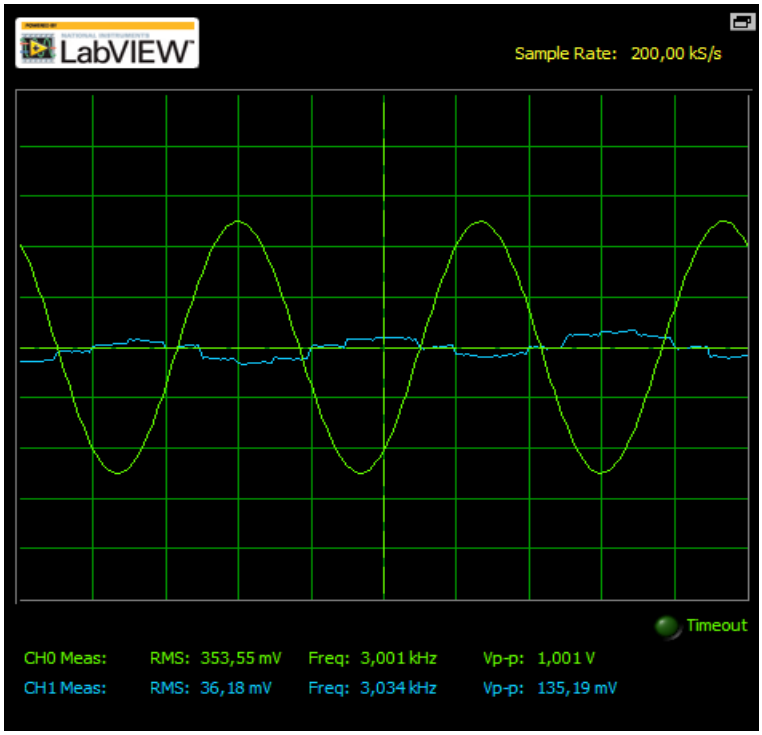


Figura 39. Señales de entrada y salida del filtro pasabajos IIR con fn 1kHz. Entrada a 3 kHz.

En la figura 39 es más evidente la atenuación de la señal de salida, puesto que tiene una amplitud de 13.5% respecto a la entrada, característica relevante de la banda de rechazo.

Los resultados del filtro IIR ante las señales de entrada de diferentes frecuencias son bastante aceptables respecto a los mostrados en la simulación. Los efectos de la frecuencia de muestreo generan cambios relevantes a la señal de salida, esto se puede mejorar aumentado la tasa de muestreo.

3.3.3.3 *Implementación del filtro FIR pasabajos a 1 kHz.*

La implementación de este filtro se realizará de la misma manera que el ejemplo anterior. Se hará una comparación en los resultados del filtro FIR con relación a los resultados del IIR. El cálculo de los coeficientes del filtro se realiza siguiendo los pasos de la sección **3.3.3.2.1.** Los coeficientes en punto fijo para el filtro FIR son los siguientes:

$$b_0fp = 1;\ b_1fp = 5;\ b_2fp = 8;\ b_3fp = 9;\ b_4fp = 9;\ b_5fp = 8;\ b_6fp = 7;\ b_7fp = 6;$$

$$b_8fp = 5;\ b_9fp = 4;\ b_{10}fp = 2;\ b_{11}fp = 2;\ b_{12}fp = 1;\ b_{13}fp = 0;\ b_{14}fp = 0;\ b_{15}fp = 0;$$

$$b_{16}fp = 0;\ b_{17}fp = 0; b_{18}fp = 0;\ b_{19}fp = 0;$$

Tal y como se mencionó en la sección de diseño para el filtro FIR, muchos coeficientes en punto fijo son 0, por lo que la ecuación (20) se simplifica bastante, eliminándose gran parte de sus factores. la ecuación 20 finalmente queda:

$$\begin{aligned} y(n) = {} & b_0fp\ x(n) + b_1fp\ x(n-1) + b_2fp\ x(n-2) + b_3fp\ x(n-3) + b_4fp\ x(n-4) \\ & + b_5fp\ x(n-5) + b_6fp\ x(n-6) + b_7fp\ x(n-7) + b_8fp\ x(n-8) \\ & + b_9fp\ x(n-9) + b_{10}fp\ x(n-10) + b_{11}fp\ x(n-11) + b_{12}fp\ x(n-12) \end{aligned}$$

De la misma manera que se implementó el filtro IIR, se decide realizar la misma rutina de interrupción para el filtro FIR. El respectivo código de este filtro para programar el microcontrolador se encuentra en los anexos a este documento.

A continuación, veremos la respuesta del filtro implementado para 3 señales diferentes:

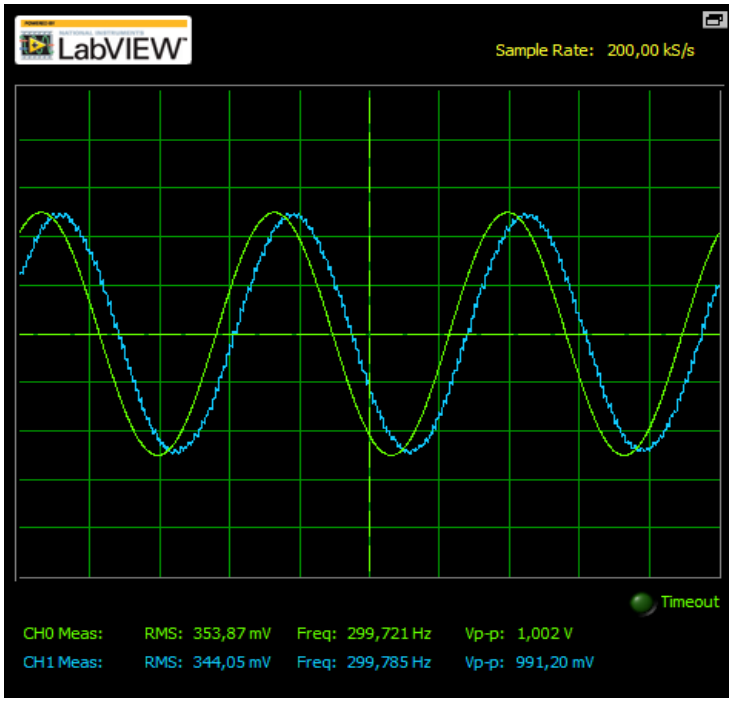


Figura 40. Señal de entrada y salida del filtro pasabajos FIR a 1 kHz. Entrada a 300 Hz.

La figura 40 muestra la entrada en color verde y la salida en color azul. La señal de salida presenta una pequeña atenuación de menos del 1%, casi irrelevante, garantizando mantener su amplitud por estar en la banda de paso. Se aprecia una pequeña distorsión natural por la frecuencia de muestreo.

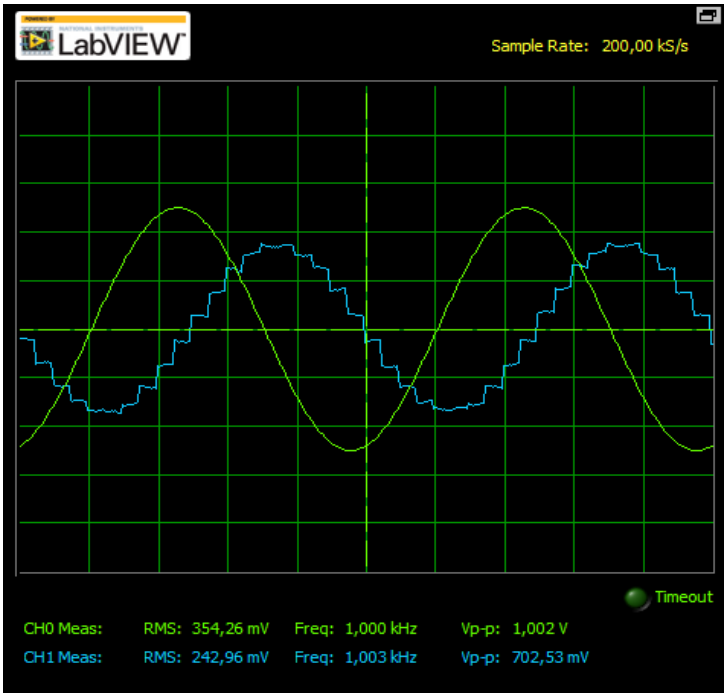


Figura 41. Señal entrada y salida del filtro pasabajos FIR a 1 kHz. Entrada a 1 kHz

Ante una entrada a 1 kHz frecuencia en el punto de corte, la respuesta es positiva debido a que la amplitud es del 70.2%, una aproximación bastante buena debido a que el error es menor que el 0.5%.

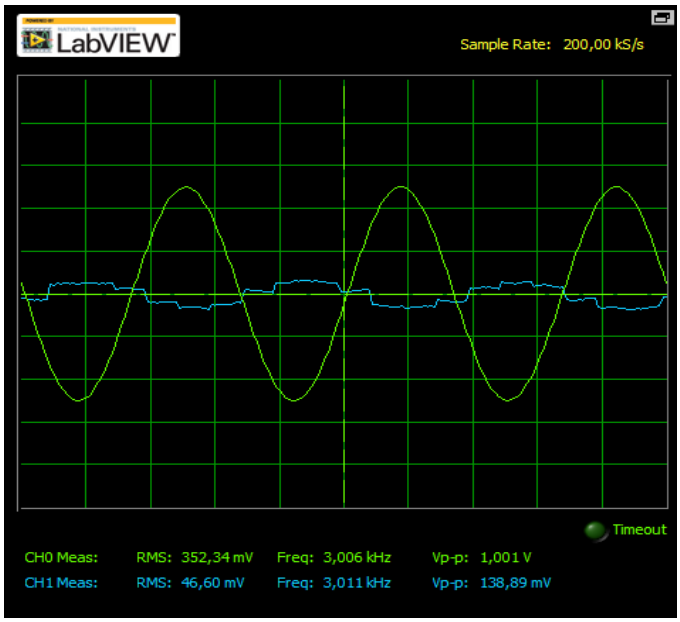


Figura 42. Señal entrada y salida del filtro pasabajas FIR a 1 kHz. Entrada a 3 kHz.

Claramente se observa la atenuación de la señal de salida. Se aprecia que la señal se atenúa significativamente con un valor final de 13.8% con relación a la entrada.

Los resultados de los valores del filtro FIR con la respectiva simulación son bastante aproximados. En comparación con los resultados del filtro IIR, los valores son mucho más acordes en la frecuencia de corte a los valores ideales del filtro.

Capítulo 4. Análisis de los resultados y conclusiones

A pesar de las dificultades que conllevaron en la realización de este proyecto, se logró conseguir resultados importantes y bastante interesantes en relación con trabajos de grado que implementaron propósitos similares al este. Como se observó, la teoría del tratamiento digital de señales es bastante compleja, pero de la misma manera bastante relevante. La evolución de este campo depende directamente del avance que se tenga en el desarrollo de sistemas que sean capaces de procesar datos a velocidades bastante altas. Por los resultados vistos en este trabajo se evidencia de forma positiva el desempeño de los sistemas microcontrolados en aplicaciones como esta.

Las siguientes conclusiones están directamente relacionadas con los resultados en los procesos de diseño, construcción, diseño de algoritmos, simulación e implementación de los algoritmos de los filtros para analizar señales de audio en tiempo real.

4.1 Conclusiones

- Un sistema basado en arquitectura microcontrolada permite desarrollar la estructura del algoritmo de forma secuencial, permitiendo evidenciar el proceso en forma lineal y observar el flujo directo del programa para encontrar posibles errores y fallas en el algoritmo.

- El microcontrolador ATXMEGA32A4U tiene una gran capacidad de procesamiento en cuanto a su sistema de temporizaciones y su hardware multiplicador, posibilitando realizar operaciones de multiplicaciones y acumulación de información de datos en intervalos de tiempo bastante pequeños. El caso particular de los filtros FIR mostró que se pueden realizar hasta 13 operaciones de multiplicación y acumulación en un tiempo menor a 50 us, dando por hecho que es un sistema capacitado para realizar tratamiento de señales no solo en el espectro audible, sino para señales de ultrasonido.

- La implementación de filtros de condensador conmutado permite en gran medida reducir el tamaño de los circuitos en cuanto a la optimización de espacio y también respecto a la facilidad de configurarse para diferentes tipos de filtro. Característica muy relevante de su funcionamiento es la flexibilidad con relación a la variación de su frecuencia de corte directamente relacionada con una señal de reloj externa, dándole atributos muy adecuados para aplicaciones de soporte para filtrado digital, haciéndolos superiores a las implementaciones en forma discreta de filtros analógicos.

- Los filtros digitales son sistemas que demuestran la gran flexibilidad ante cambios que en los filtros analógicos significaría una reestructuración total en su construcción, y no solo eso presentan una precisión bastante alta debido a que no depende directamente de valores de componentes discretos como los analógicos, si no de frecuencias de muestreo y capacidad de procesamiento, que hoy día se ha solucionado satisfactoriamente, demostrando su gran ventaja a la hora de realizar aplicaciones de tratamiento de señales en tiempo real.

- Las señales muestreadas a frecuencias alrededor de 5 a 10 veces su frecuencia superior presentan distorsiones bastante significativas en su forma de onda al momento de la conversión digital a analógica, presentándose claramente una relación inversa entre la frecuencia de muestreo y la distorsión de la señal de salida, es decir que este fenómeno se corrige bastante bien al aumentar de forma significativa la frecuencia de muestreo del sistema.

- La diferencia significativa de los filtros IIR respecto a los filtros FIR en cuanto al tamaño y la longitud de sus coeficientes, los hace más digeribles por sistemas con capacidades de procesamiento bajas limitados por velocidades de reloj pequeñas, puesto que se necesitan menor cantidad de iteraciones para efectuar las operaciones de suma y multiplicación; sin embargo, los filtros FIR presentan una mejor respuesta en cuanto a amplitud y fase lineal, haciéndolos más adecuados para aplicaciones donde la calidad de los resultados es más exigente.

- La cantidad de coeficientes de un filtro FIR es bastante extensa dependiendo de la frecuencia de muestreo del sistema, esto permite encontrar una relación significativa entre longitud de coeficientes y la velocidad de muestreo, es decir si el sistema necesita tasas de muestreo altas esto impactará directamente en la arquitectura FIR, necesitando más coeficientes para garantizar resultados muy positivos en la señal de salida y por supuesto velocidades de procesamiento mayores.

- Debido a que los efectos en fase de los filtros antialias y posfiltrado no son lineales, esto puede afectar un poco el corrimiento en fase de las señales de entrada, puesto que son de cuarto orden y la frecuencia de corte de estos filtros es de 20 kHz las señales anteriores a esta se encuentran en una banda donde el desfase en la señal no afecta significativamente su forma de onda.

5. Bibliografía

MITRA, S. Digital Signal Processing, Second Edition, McGraw Hill.

PROAKIS G. J, MANOLAKIS G., D. Digital Signal Processing, Prentice Hall Internacional Inc.

ATMEL CORPORATION. (2013). 8-bit Atmel XMEGA AU Microcontroller XMEGA AU MANUAL.http://www.atmel.com

ATMEL CORPORATION. (2014). 8/16-bit Atmel XMEGA Microcontroller ATxmega128A4U /ATxmega64A4U /ATxmega32A4U / ATxmega16A4U. http://www.atmel.com

ATMEL CORPORATION. (2002). AVR223: Digital Filters with AVR. http://www.atmel.com

SHENOI B. A., Introduction to Digital Signal Processing and Filter Design, Wiley Interscience, 2006.

ATMEL CORPORATION. (2016). AVR201: Using the AVR Hardware Multiplier. APPLICATION NOTE. http://www.atmel.com

ATMEL CORPORATION. (2010). Atmel AVR1300: Using the Atmel AVR XMEGA ADC. http://www.atmel.com

ATMEL CORPORATION. (2016). The Atmel AVR Dragon Debugger. http://www.atmel.com

ATMEL CORPORATION. (2016). Atmel Studio USER GUIDE. http://www.atmel.com

BONILLA CAMELO, J. PEÑA, W. (2010). Diseño y programación de filtros digitales FIR en lenguaje de descripción de hardware HDL e implementación en FPGA. (Tesis de pregrado). Universidad Pedagógica Nacional. Bogotá Colombia.

JARRIN OCHOA, D. (2016). Diseño e implementación de filtros FIR e IIR utilizando el microcontrolador XMEGA de ATMEL para el tratamiento de señales de audio. (Tesis de pregrado). Universidad Politécnica Salesiana. Quito, Ecuador.

MESA HERNANDEZ, E. (2016). Generador de señales periódicas de forma predeterminada empleando un dispositivo chipbas8. (Tesis de pregrado). Universidad Nacional Autónoma de Mexico. Ciudad de México, México.

NATIONAL SEMICONDUCTOR. (1999). LMF100 High Performance Dual Switched Capacitor Filter. http://www.ti.com

OBERSTAR, E. BAUCH, M. (2001). Narrow Band Filter Implementation On A Low Cost Microcontroller Issues and Performance. University of Wisconsin. Madison, USA.

VIVAS GONZALES, E. RIVERA PINZÓN, D. GOMEZ, E. Implementation and Simulation of IIR Digital Filters in FPGA Using MatLab System Generator.

TEXAS INSTRUMENTS. (2017). LM386 Low Voltage Audio Power Amplifier.

TEXAS INSTRUMENTS. (2017). LMx58-N Low-Power, Dual-Operational Amplifiers.

LAND, B. (2014). DSP on 8-bit microcontroller. https://www.hackster.io/bruceland/dsp-on-8-bit-microcontroller-21220c

JIANG, J. (2014). Teaching Digital Signal Processing Course with a Real-Time Digital Crossover System for Electrical and Computer Engineering Technology Students. Purdue Univesity, North Central, USA.

6. Anexos

6.1 Cálculo de coeficientes y simulación de filtro IIR pasabajas de segundo orden a 1 kHz en MATLAB

```
%%%% Filtro Pasabajos IIR segundo orden

fs = 20000;  % Frecuencia de muestreo
T = 1/fs;     % Periodo de muestreo
fn = 1000;    % Frecuencia de corte

fd = tan(pi*T*fn)/(pi*T);  %% frecuencia de corte digital
Wn = 2*pi*fd;       %% velocidad angular digital
G = 1;      % ganancia del filtro
 %%% Coeficientes del filtro IIR

b0 = G*Wn^2*T^2/(4+2*T*1.4142*Wn+T^2*Wn^2);
b1 = 2*G*Wn^2*T^2/(4+2*T*1.4142*Wn+T^2*Wn^2);
b2 = G*Wn^2*T^2/(4+2*T*1.4142*Wn+T^2*Wn^2);
a1 = 2*(T^2*Wn^2-4)/(4+2*T*1.4142*Wn+T^2*Wn^2);
a2 = (4-2*T*1.4142*Wn+T^2*Wn^2)/(4+2*T*1.4142*Wn+T^2*Wn^2);

H = tf([b0 b1 b2],[1 a1 a2],T);   % función de transferencia del filtro digital

%%   Simulación del filtro

y1 = 0;
y2 = 0;
y3 = 0;
x1 = 0;
x2 = 0;
N = 1000; % Number of Samples
f = 1000; % Frequency of Sine Wave
n = 0:1:N;
t = n/fs;
x = sin(2*pi*f*n/fs);

for k=1:N+1
   y(k) = -a1*y1 - a2*y2 + b0*x(k) + b1*x1 + b2*x2;
   y3 = y2;
   y2 = y1;
   y1 = y(k);
   x2 = x1;
   x1 = x(k);
end

plot(t,x,t,y,'r')
```

```
xlim([0 5/f])
ylim([-1.25 1.25])
ylabel('Amplitud');
xlabel('t [ms]');
legend('señal entrada','señal salida');
grid on


%%  Conversión de coeficientes decimales a punto fijo
B = [b0 b1 b2];   %% Coeficientes del numerador de la función de transferencia
A = [a1 a2];      %% Coeficientes del denominador de la función de transferencia

Bfp = fi(B,1,8,6);  %% Aproxima el valor decimal al más cercano representado con 8 bits y
6 para la parte decimal
Afp = fi(A,1,8,6);

bin(Bfp)    %% Convierte a binario el valor proporcionado por en punto fijo
bin(Afp)

int(Bfp)    %% Convierte e entero el resultado final de la conversión
int(Afp)
```

6.2 Cálculo de coeficientes y simulación del filtro FIR de segundo orden pasbajos a 1 kHz en MATLAB.

```
%%% Filtro FIR pasabajas

fs = 20000;
T = 1/fs;
fn = 1000;

%%% Filtro IIR del cual partirá el diseño

fd = tan(pi*T*fn)/(pi*T);  %% frecuencia de corte digital
Wn = 2*pi*fd;       %% velocidad angular digital
G = 1;      % ganancia del filtro

b0 = G*Wn^2*T^2/(4+2*T*1.4142*Wn+T^2*Wn^2);
b1 = 2*G*Wn^2*T^2/(4+2*T*1.4142*Wn+T^2*Wn^2);
b2 = G*Wn^2*T^2/(4+2*T*1.4142*Wn+T^2*Wn^2);
a1 = 2*(T^2*Wn^2-4)/(4+2*T*1.4142*Wn+T^2*Wn^2);
a2 = (4-2*T*1.4142*Wn+T^2*Wn^2)/(4+2*T*1.4142*Wn+T^2*Wn^2);

H = tf([b0 b1 b2],[1 a1 a2],T);   % función de transferencia del filtro digital
%%  Hallar respuesta al impulso

[v,t1] = impulse(H,0.5e-3);
v = v/fs;
stem(t1,v);
%% Encontrar función de transferencia para los primeros 12 periodos T

[v,t1] = impulse(H,12*T);
v = v/fs;
Hz = tf(v',1,T);

b0 = v(1); b1 = v(2); b2 = v(3); b3 = v(4); b4 = v(5); b5 = v(6); b6 = v(7); b7 = v(8); b8 =
v(9); b9 = v(10); b10 = v(11); b11 = v(12); b12 = v(13);

%% respuesta del filtro

x1 = 0; x2 = 0; x3 = 0; x4 = 0; x5 = 0; x6 = 0; x7 = 0; x8 = 0; x9 = 0; x10 = 0; x11 = 0; x12 =
0;

N = 1000; % Number of Samples
f = 1000; % Frequency of Sine Wave
n = 0:1:N;
t = n/fs;
x = sin(2*pi*f*n/fs);
```

```
    for k=1:N+1
        y(k) = b0*x(k) + b1*x1 + b2*x2 + b3*x3 + b4*x4 + b5*x5 + b6*x6 + b7*x7 + b8*x8 +
b9*x9 + b10*x10 + b11*x11 + b12*x11;
        x11 = x10;
        x10 = x9;
        x9 = x8;
        x8 = x7;
        x7 = x6;
        x6 = x5;
        x5 = x4;
        x4 = x3;
        x3 = x2;
        x2 = x1;
        x1 = x(k);
    end

    plot(t,x,t,y,'r')
    xlim([0 5/f])
    ylim([-1.25 1.25])
    ylabel('Amplitud');
    xlabel('t [ms]');
    legend('señal entrada','señal salida');
    grid on

    %%  Conversión de coeficientes decimales a punto fijo

    vpf = fi(v',1,8,6); %% convertir los valores de los coeficientes a punto fijo

    bin(vfp)   %%representar en binario los valores convertidos
    int(vfp)    %% representar en enteros los valores de la conversión
```

6.3 Código en lenguaje C del filtro IIR pasabajas a 1 kHz, para la implementación en el microcontrolador

```
/*
 * IIR_Pasabajos.c
 *
 * Author : jefer
 */

#define F_CPU 32000000
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>
#include <avr/interrupt.h>

    char x0temp = 0;
    signed char x1 = 0, x2 = 0,  y1 = 0, y2 = 0; // variables de estado
    signed char y0 = 0;
    signed char x0 = 0; // primer estado del filtro
    signed char b0 = 0, b1 = 0, b2 = 0, a1 = 0, a2 = 0; //Coeficientes filtro
    signed int y0temp = 0;
    char y0f = 0;

void clock_setup(){

        OSC.CTRL |= OSC_RC32MEN_bm | OSC_RC32KEN_bm;  // habilitar interno
    32MHz &                                                    //32KHz osciladores
        while(!(OSC.STATUS & OSC_RC32KRDY_bm));      // esperar 32Khz oscilador
        DFLLRC2M.CTRL = DFLL_ENABLE_bm ;           // habilitar DFLL
                                          //calibrar reloj interno 32Khz
    CCP = CCP_IOREG_gc;
    while(!(OSC.STATUS & OSC_RC32MRDY_bm));// Esperar hasta que se estabilice
    CCP = CCP_IOREG_gc;      // Desabilitar registros de seguridad para cargar cambios
    OSC.PLLCTRL = OSC_PLLSRC_RC32M_gc | 4;
    OSC.CTRL |= OSC_PLLEN_bm;
    while(!(OSC.STATUS & OSC_PLLRDY_bm));

// Esperar hasta que el PLL se estabilice
    CCP = CCP_IOREG_gc;
    CLK.CTRL = CLK_SCLKSEL_PLL_gc;
    OSC.CTRL &= ~OSC_RC2MEN_bm;               // desabilitar 2Mhz oscilador

//   CLK OUT Port C
            PORTC_DIR = 0x03;
```

```
    // Timer_0_configuración

    TCC0.CTRLA = TC_CLKSEL_DIV1_gc;
    TCC0.CTRLB = 0x10 | TC_WGMODE_FRQ_gc;
    TCC0.CCA = 0x000B;                                  // equivale a 1600

    // Timer 1 configuración

    TCC1.PER = 0x0683;  // para 20kHz
    TCC1.CTRLA = TC_CLKSEL_DIV1_gc;
    TCC1.INTCTRLA = TC_OVFINTLVL_MED_gc;
    PMIC.CTRL = PMIC_MEDLVLEN_bm;
    sei();

}

void adc_setup(){

    ADCA.CTRLB = ADC_FREERUN_bm | ADC_RESOLUTION_8BIT_gc;
    ADCA.REFCTRL = ADC_REFSEL_AREFA_gc;
    ADCA.PRESCALER = ADC_PRESCALER_DIV8_gc;
    ADCA.INTFLAGS = ADC_CH0IF_bm;
    ADCA.CH0.CTRL = ADC_CH_INPUTMODE_SINGLEENDED_gc;
    ADCA.CH0.MUXCTRL = ADC_CH_MUXPOS_PIN5_gc;
    ADCA.CTRLA = ADC_ENABLE_bm ;
}

void dac_setup(){
    DACB.CTRLB = DAC_CHSEL_SINGLE_gc;
    DACB.CTRLC = DAC_REFSEL_AREFA_gc | DAC_LEFTADJ_bm;
    DACB.CTRLA = DAC_CH0EN_bm | DAC_ENABLE_bm;
}


ISR(TCC1_OVF_vect){

    x0temp = ADCA.CH0RESL;
    x0 =  x0temp - 127;                    // remover offset
    y0temp = -(signed int)a1*y1 - (signed int)a2*y2 + (signed int)b0*x0 + (signed
int)b1*x1 + (signed int)b2*x2; // Multiplicación y acumulación para convertir a Q13
    y0 = (signed char)((y0temp) >> 6);
    y2 = y1;
    y1 = y0;
    x2 = x1;
    x1 = x0;
```

```
    DACB.CH0DATAH = y0 + 127;
    PORTC.OUTTGL = PIN1_bm; // señal para verificación velocidad muestreo
}

int main(void){

    clock_setup();
    adc_setup();
    dac_setup();



    // Coeficientes del Filtro
    a1 = -100;
    a2 = 41;
    b0 = 1;
    b1 = 3;
    b2 = 1;

    while(1){
    }
}
```

6.4 Código en lenguaje C del filtro FIR pasabajas a 1 kHz, para la implementación en el microcontrolador

```
/*
 * FIR_Pasabajas.c
 *
 * Author : jefer
 */

#define F_CPU 32000000
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>
#include <avr/interrupt.h>

char x0temp = 0;
signed char x0 = 0, x1 = 0, x2 = 0, x3 = 0, x4 = 0, x5 = 0, x6 = 0, x7 = 0, x8 = 0, x9 = 0, x10 = 0, x11 = 0, x12 = 0; // variables de estado filtro

signed char y0 = 0;
signed char b0 = 0, b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, b6 = 0, b7 = 0, b8 = 0, b9 = 0, b10 = 0, b11 = 0, b12 = 0; //Coeficientes del filtro

signed int y0temp = 0;
```

```
void clock_setup(){

    OSC.CTRL |= OSC_RC32MEN_bm | OSC_RC32KEN_bm;  //
    while(!(OSC.STATUS & OSC_RC32KRDY_bm));      //
    DFLLRC2M.CTRL = DFLL_ENABLE_bm ;            //
    CCP = CCP_IOREG_gc;
    while(!(OSC.STATUS & OSC_RC32MRDY_bm));      //
    CCP = CCP_IOREG_gc;                                                    //
    OSC.PLLCTRL = OSC_PLLSRC_RC32M_gc | 4;
    OSC.CTRL |= OSC_PLLEN_bm;
    while(!(OSC.STATUS & OSC_PLLRDY_bm));
    CCP = CCP_IOREG_gc;
    CLK.CTRL = CLK_SCLKSEL_PLL_gc;
    OSC.CTRL &= ~OSC_RC2MEN_bm;                 //

    PORTC_DIR = 0x03;

                                    // equal to 3200
    TCC0.CTRLA = TC_CLKSEL_DIV1_gc;
    TCC0.CTRLB = 0x10 | TC_WGMODE_FRQ_gc;
    TCC0.CCA = 0x000B;                                                  // Equal to 1600

    // Timer 1 configuración
    TCC1.PER = 0x683;  // para 20 kHz
    TCC1.CTRLA = TC_CLKSEL_DIV1_gc;
    TCC1.INTCTRLA = TC_OVFINTLVL_MED_gc;
    PMIC.CTRL = PMIC_MEDLVLEN_bm;
    sei();

}

void adc_setup(){

    ADCA.CTRLB = ADC_FREERUN_bm | ADC_RESOLUTION_8BIT_gc;
    ADCA.REFCTRL = ADC_REFSEL_AREFA_gc;
    ADCA.PRESCALER = ADC_PRESCALER_DIV8_gc;
    ADCA.INTFLAGS = ADC_CH0IF_bm;
    ADCA.CH0.CTRL = ADC_CH_INPUTMODE_SINGLEENDED_gc;
    ADCA.CH0.MUXCTRL = ADC_CH_MUXPOS_PIN5_gc;
    ADCA.CTRLA = ADC_ENABLE_bm ;
}

void dac_setup(){
    DACB.CTRLB = DAC_CHSEL_SINGLE_gc;
    DACB.CTRLC = DAC_REFSEL_AREFA_gc | DAC_LEFTADJ_bm;
```

```
    DACB.CTRLA = DAC_CH0EN_bm | DAC_ENABLE_bm;
}


ISR(TCC1_OVF_vect){


    x0temp = ADCA.CH0RESL;
    x0 = x0temp - 127;                    // Remover offset
    y0temp = (signed int)b0*x0 + (signed int)b1*x1 + (signed int)b2*x2 + (signed int)b3*x3
+ (signed int)b4*x4 +(signed int)b5*x5 + (signed int)b6*x6 + (signed int)b7*x7 + (signed
int)b8*x8 + (signed int)b9*x9 + (signed int)b10*x10 + (signed int)b11*x11 + (signed
int)b12*x12; // Multiplicación y acumulación para Q13
    y0 = (signed char)((y0temp) >> 6);//desplazar derecha 6 bit menos significativos
    x12 = x11;
    x11 = x10;
    x10 = x9;
    x9 = x8;
    x8 = x7;
    x7 = x6;
    x6 = x5;
    x5 = x4;
    x4 = x3;
    x3 = x2;
    x2 = x1;
    x1 = x0;

    DACB.CH0DATAH = y0 + 127;
    PORTC.OUTTGL = PIN1_bm;
}

int main(void){

    clock_setup();
    adc_setup();
    dac_setup();

    // Coeficientes del filtro en punto fijo
    b0 = 1;
    b1 = 5;
    b2 = 8;
    b3 = 9;
    b4 = 9;
    b5 = 8;
    b6 = 7;
    b7 = 6;
```

```
    b8 = 5;
    b9 = 4;
    b10 = 2;
    b11 = 2;
    b12 = 1;

    while(1){
    }
}
```

MANUAL TÉCNICO PROTOTIPO DE FILTRADO DIGITAL

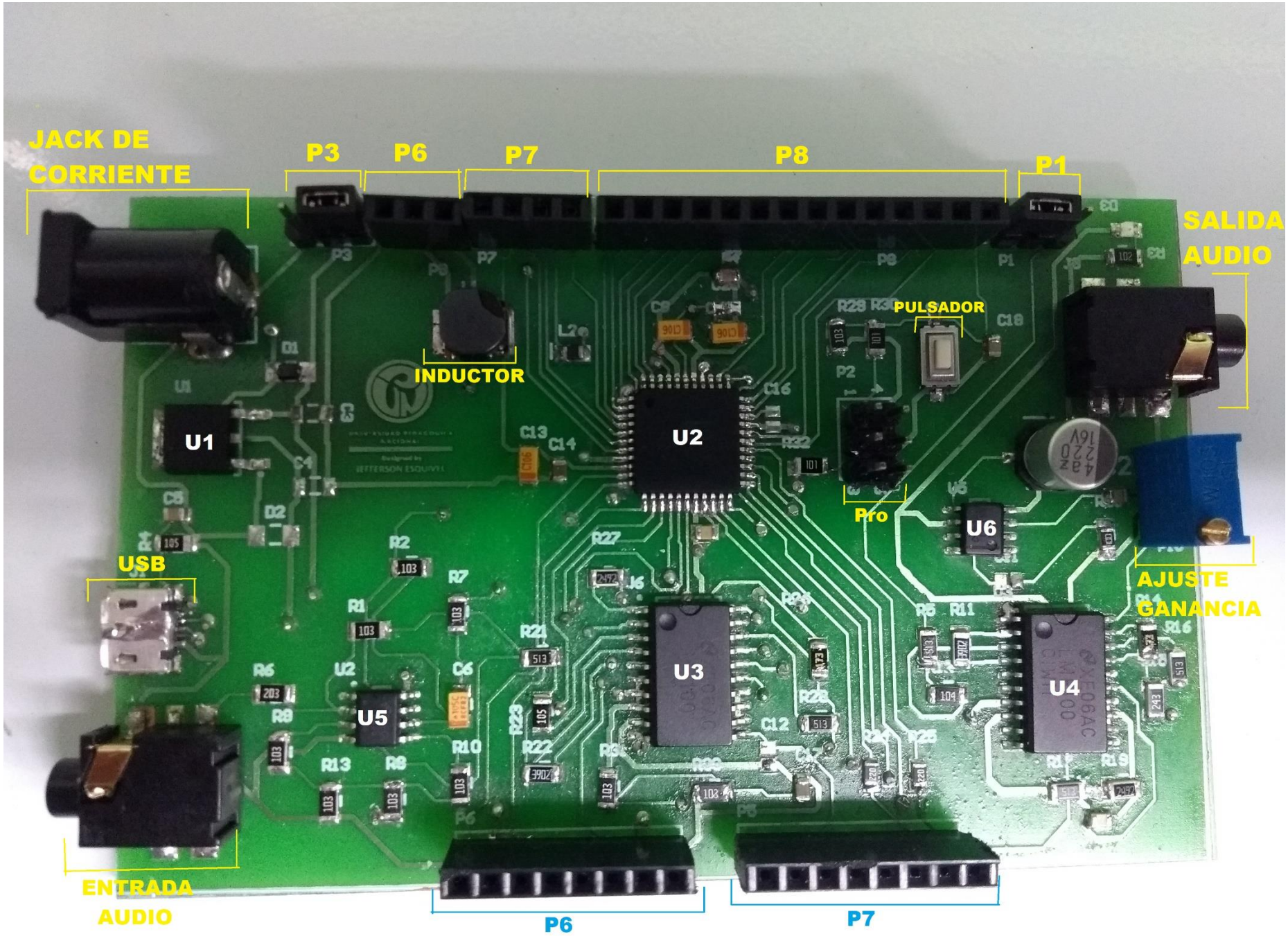
Diseñado por: Jefferson Esquivel Hincapie.

Cod: 2011203024

Correo: jefer.esqui93@gmail.com

Licenciatura en Electrónica, Universidad Pedagógica Nacional.

Este manual es diseñado especialmente para dar a conocer la estructura y las partes principales de funcionamiento del producto final del trabajo de grado que tiene como nombre **DISEÑO Y CONTRUCCIÓN DE UN PROTOTIPO PROGRAMABLE, ORIENTADO A LA APLICACIÓN DE ALGORITMOS DE PROCESAMIENTO DIGITAL DE SEÑALES DE AUDIO.**


JACK DE CORRIENTE
P3
P6
P7
P8
P1
SALIDA AUDIO
INDUCTOR
PULSADOR
U1
U2
Pro
U6
USB
AJUSTE GANANCIA
U3
U4
U5
ENTRADA AUDIO
P6
P7

ESPECIFICACIONES TÉCNICAS

El propósito de este prototipo es realizar tratamiento digital de señales, especialmente señales de audio. Es diseñado para alimentarse con voltajes mayores o iguales a **5 voltios**, pero no superiores a **7 voltios** por precaución. Aunque la arquitectura sea especialmente para el tratamiento digital de audio, se observa se observa que cuenta con 5 regletas de las cuales, 4 están conectados a los pines del microcontrolador, especialmente diseñadas para utilizarse con aplicaciones de propósitos general.

Descripción de los elementos seleccionados:

Jack corriente: Está especialmente diseñado para conectar fuentes externas de 5 voltios.

USB: Puerto micro USB implementado para realizar aplicaciones de comunicación con PC. Este puerto tiene un pin de alimentación a 5v, especialmente diseñado para utilizar el voltaje proporcionado por el puerto USB del PC.

P3: Este puerto de 3 terminales permite elegir por medio de un jumper cual será la fuente de alimentación proporcionada para el sistema. De izquierda a derecha el pin 1 está conectado al Jack de corriente, el pin 2 está conectado a la alimentación interna del sistema y el pin 3 está conectado al puerto de alimentación del puerto USB.

P6: Es un puerto de 3 terminales que de izquierda a derecha: 5v, 3.3v y GND.

P7: Puerto de 4 pines que están conectados de derecha a izquierda a las terminales: 4, 5, 6 y 7 del microcontrolador.

P8: Puerto de 10 pines conectadas de derecha a izquierda a las terminales: 28, 29 ,32, 33, 36, 37, 40, 41, 42, 43, 44, 1, 2 y 3 del microcontrolador.

P1: Puerto de 3 pines que permite separar por medio de un jumper la salida del DAC0 del micro.

P6: Puerto de 8 pines conectadas de izquierda a derecha a las terminales: 10, 11, 12, 13, 14, 15, 16, 17 del microcontrolador.

P7: puerto de 8 pines conectados de izquierda a derechas a las terminales: 20, 21, 22, 23, 24, 25, 26, 27 del microcontrolador.

U1: Regulador de 3.3v de salida que se diseñó para alimentar las terminales de Vcc ,AVcc y Aref del microcontrolador, debido a que está diseñado para funcionar con niveles de voltaje no mayores a 3.6v

U2: Microcontrolador ATXMEGA32A4U.

U3: LMF100, filtro de condensador conmutado usado como etapa del filtrado antialiasing a 20 kHz.

U4: LMF100, filtro de condensador conmutado usado como etapa de posfiltrado de la señal de salida del DAC0.

U5: LM358, dual amplificador operacional implementado para realizar las etapas cambiadoras de nivel del prototipo.

U6: LM386, amplificador de audio de baja potencia usado para alimentar etapas de amplificación externas.

Ajuste de ganancia: Este es un trimmer de 10 k ohms usado para ajustar la ganancia de salida de la etapa de amplificación.

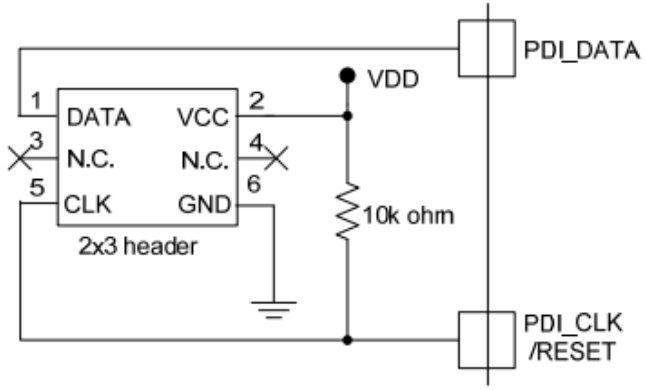
Inductor: Implementado específicamente por sugerencia del fabricante para eliminar el ruido digital presentado a altas frecuencias.

Pulsador: Está directamente conectado con el pin de RESET del microcontrolador, especialmente usado para reiniciar las aplicaciones que se estén implementando.

Entrada de audio: Es un Jack de audio diseñado especialmente para conectarse a fuentes de sonido. El prototipo está diseñado para ser alimentado con señales de audio que no superen una amplitud de 2 voltios pico a pico.

Salida de audio: Es el segundo Jack implementado para conectarse a dispositivos reproductores de audio externos.

Pro: Es un puerto de 6 pines donde que van conectados en de acuerdo con el siguiente esquema proporcionado por el fabricante.



Conexión de pines de programación.

Donde VDD es el voltaje de alimentación de 3.3v del circuito.

Programación del prototipo

Existen varios dispositivos sugeridos por el fabricante para programar el microcontrolador ATXMEGA32A4U de Atmel Corporation. El escogido para el proyecto es la tarjeta AT AVR DRAGON. Es mostrado a continuación.

Programador AT AVR DRAGON

El software para realizar la programación es de uso libre y es proporcionado por Atmel desde su página oficial www.atmel.com. La versión utilizada para el proyecto fue Atmel Studio 7.