UNIVERSIDAD DE PAMPLONA

Desarrollo de un sistema embebido para prácticas de procesamiento digital de audio utilizando dsPIC®

Sherlin Eutimio Hernández Contreras

Universidad de Pamplona
Facultad de Ingenierías y Arquitectura, Programa de Ingeniería Electrónica
Pamplona, Colombia
2016

Desarrollo de un sistema embebido para prácticas de procesamiento digital de audio utilizando dsPIC®

Sherlin Eutimio Hernández Contreras

Trabajo de grado presentado como requisito parcial para optar al título de:
Ingeniero Electrónico

Director:
MSc. Carlos Arturo Vides Herrera

Universidad de Pamplona
Facultad de Ingenierías y Arquitectura, Programa de Ingeniería Electrónica
Pamplona, Colombia
2016

Dedicatoria

A mis padres Ana y Eutimio
por su infinito apoyo.

Agradecimientos

A mi familia por su apoyo en todo momento.

A mi director de tesis MSc. Carlos Arturo Vides, al ingeniero Libardo Gamboa y al Msc. Jesús Eduardo Ortiz por sus valiosas observaciones y correcciones para mejorar este trabajo. También a los demás profesores de la Universidad de Pamplona que fomentaron mi aprendizaje durante mi etapa estudiantil.

A todas las personas de las que recibí apoyo en mi estancia en la ciudad de Pamplona y motivaron el cumplimiento de mis logros, entre ellas debo destacar a Claudia Gómez, Doña Socorro Vera, Don Ángel Gómez, Doña Olga Vera, la profesora Nubia Riscanevo y el profesor William Parada.

Y a todos aquellos quienes directa o indirectamente contribuyeron de forma positiva en mi crecimiento personal y profesional.

Resumen

En el presente trabajo se plantea el desarrollo de una tarjeta de procesamiento digital de audio que sirva como herramienta para la enseñanza de la tecnología de sistemas embebidos. A partir de un estudio de la actualidad de este tipo de tarjetas y del análisis de algunos productos relacionados, de la empresa Microchip, se diseñó una propuesta para solventar las necesidades detectadas o posibles mejoras con base en criterios de velocidad de procesamiento, consumo de energía, precio y posibilidades de desarrollo.

Mediante pruebas de laboratorio, se comprobó la funcionalidad del sistema con distintos algoritmos de procesamiento de audio diseñados específicamente para la tarjeta construida. El producto de esta investigación es una tarjeta de procesamiento digital de señales de audio con software de soporte para el desarrollo de los algoritmos y un manual de uso que incluye prácticas de procesamiento diseñadas para dicho sistema.

Palabras clave: Sistemas embebidos, DSP, dsPIC®, señales de audio, procesamiento digital de audio.

Abstract

In this paper it is proposed the development of a digital audio processing circuit board that serves as a tool for the teaching of embedded system technology. Starting from a study about the present of that kind of circuit boards and by the analysis of some related products by Microchip Company, it was designed a proposal to address the identified needs or possible improvements, based on criteria processing speed, power consumption, price and development possibilities.

By laboratory tests, the functionality of the system was checked with different audio processing algorithms specifically designed for the circuit board built. The product of this research is an embedded circuit board for digital audio signal processing with software for supporting the development of algorithms and user guides including processing practices designed for that system.

Keywords: Embedded systems, DSP, dsPIC®, audio signals, digital audio processing.

Contenido

Lista de Símbolos

Símbolos

Símbolo	Término
G	Ganancia de audio
W	Registro de trabajo
Wd	Registro de trabajo destino
WREG	Registro de trabajo principal
Ws	Registro de trabajo fuente
$x[n]$	Señal digital de audio de entrada
$y[n]$	Señal digital de audio de salida

Abreviaturas

Abreviatura	Término
AAC	Codificación Avanzada de Audio
ADC	Convertidor Analógico a Digital
ADPCM	Modulación por Impulsos Codificados Diferencial Adaptativo
AGU	Unidad de Generación de Direcciones
ALU	Unidad Aritmético-Lógica
AM	Amplitud Modulada
CD	Disco Compacto
CODEC	Codificador-Decodificador
CPU	Unidad Central de Proceso
DAC	Convertidor Digital a Analógico
DAT	Cinta de Audio Digital
DCI	Interfaz de Convertidor de Datos
DFT	Transformada Discreta de Fourier
DMA	Acceso Directo a Memoria
DSC	Controlador Digital de Señales
DSP	Procesador Digital de Señales
DSP	Procesamiento Digital de Señales

Abreviatura	Término
DTMF	Sistema de marcación por tonos
DVD	Disco de Vídeo Digital
ECAN™	Controlador de Red de Área Mejorado
FFT	Transformada Rápida de Fourier
FIFO	Datos salen en el orden en que entraron
FIR	Respuesta Finita al Impulso
FM	Frecuencia Modulada
IDE	Entorno de Desarrollo Integrado
IPE	Entorno de Programación Integrado
IIR	Respuesta Infinita al Impulso
LED	Diodo Emisor de Luz
LSb	Bit Menos Significativo
LSB	Byte Menos Significativo
LSW	Palabra Menos Significativa
MCU	Microcontrolador
MIPS	Millones de Instrucciones Por Segundo
MP3	Formato de audio MPEG-1(2) Audio Layer III
MSb	Bit Más Significativo
MSB	Byte Más Significativo
MSW	Palabra Más Significativa
OGG	Formato de audio de la Fundación Xiph.Org
PCB	Tarjeta de Circuito Impreso
PCM	Modulación por Impulsos Codificados
PWM	Modulación por Ancho de Pulso
QEI	Interfaz de Convertidor de Cuadratura
RAM	Memoria de Datos
SFR	Registros de Funciones Especiales
SMT	Tecnología de Montaje Superficial
SPI	Interfaz Periférica Serial
TAD	Período de trabajo del ADC
TFT	Transistores de Película Fina
UART	Receptor-Transmisor asíncrono universal
USB	Bus Serial Universal
v pk-pk	Voltaje pico a pico
VoIP	Voz sobre Protocolo de Internet
WAV	Formato de audio WAVEform
WMA	Formato de audio Windows Media

Lista de Figuras

Lista de Tablas

1. Introducción

La importancia de las señales de audio y el auge de la tecnología digital evidencian un nuevo campo de exploración que supone la necesidad de contar con equipos y herramientas actualizados para su estudio e investigación. El presente documento es el producto de una propuesta que servirá como herramienta para el aprendizaje de las técnicas básicas para el procesamiento digital de señales de audio a través de un sistema embebido, previamente se ha decidido que el sistema estará basado en un dispositivo llamado dsPIC®.

1.1. Problema

Los DSP (Procesadores Digitales de Señales) han alcanzado un alto grado de utilización en el mercado de los productos de electrónica y telecomunicaciones [9] debido a que las técnicas de procesamiento digital han demostrado grandes ventajas frente a las técnicas analógicas[32], las aplicaciones actuales son numerosas [32, 9, 2] y siguen en aumento. En conveniencia con esta realidad es muy importante el aprendizaje de esta tecnología en el ámbito universitario[9].

El desarrollo de prácticas de procesamiento de señales en un sistema embebido requiere amplio conocimiento de la arquitectura del sistema y a veces la programación resulta bastante difícil de comprender para un principiante, esto crea la necesidad de tener ejemplos de programas funcionales, sencillos y comprensibles para el que inicia el estudio de esta tecnología, de modo que cuando el aprendiz comprenda los puntos básicos, pueda luego extender su conocimiento hacia algoritmos más avanzados.

Teniendo en cuenta que los circuitos integrados llamados dsPIC® se muestran propicios para el desarrollo de estas aplicaciones[3], se plantea la pregunta ¿Cómo aprovechar la tecnología de los dsPIC® para permitir un aprendizaje eficiente sobre procesamiento digital de audio en sistemas embebidos?

1.2. Justificación

El incremento exponencial tanto en número como en complejidad de las aplicaciones en el campo del procesamiento digital de señales [3] supone la necesidad de implementar nuevas herramientas que se ajusten a la actualidad de esta rama de la ingeniería electrónica. La tecnología moderna, basada enormemente en la electrónica digital, maneja muchos algoritmos que son resueltos con DSP. Los dsPIC® son una clase de dispositivos híbridos que integran en un solo chip las bondades de los microcontroladores con la potencia de los DSP, haciéndolos ideales para agilizar el aprendizaje de aquellos que tienen experiencia en la programación de microcontroladores.

Se ha detectado que el mercado de tarjetas entrenadoras para aplicaciones actuales de procesamiento de audio no está bien cubierto o no ofrece entornos de desarrollo que motiven la iniciación en esta tecnología, esto sugiere que se debe mejorar en estos aspectos para modernizar la experiencia de los estudiantes e investigadores. Actualmente, la tendencia de los programas académicos que son afines al procesamiento digital de señales es incluir en sus planes de estudio al menos una materia prácticamente obligatoria relacionada con este tema, esto determina la necesidad de desarrollar herramientas complementarias al currículo.

Considerando que muchos kits de desarrollo actuales no dan mucha facilidad de uso, y que un curso de procesamiento de señales puede ser tedioso, es necesario buscar un enfoque hacia la facilidad de programación de un sistema embebido, la cual es muy diferente a la programación de una computadora.

Por último, en el ciclo de desarrollo de un producto se involucran varios pasos que no se contemplarían en la simulación por computadora. Adquirir experiencia, en el hardware embebido como tal, permite entrenarse para aplicaciones de tiempo real que pueden dar el aprendizaje que no se adquiriría de otra manera.

1.3. Objetivos

1.3.1. Objetivo general

- Diseñar e implementar un sistema embebido para prácticas de procesamiento digital de audio utilizando dsPIC®.

1.3.2. Objetivos específicos

- Construir una tarjeta embebida para prácticas de procesamiento digital de audio.
- Implementar, en la tarjeta embebida, prácticas de audio con enfoque en operaciones matemáticas en el dominio temporal.
- Realizar prácticas de procesamiento digital de audio aplicando filtros FIR y filtros IIR en el sistema embebido desarrollado.
- Elaborar aplicaciones básicas del procesamiento digital de audio con base en la transformada Discreta de Fourier utilizando la tarjeta embebida desarrollada.
- Validar el funcionamiento del sistema embebido desarrollado.

1.4. Metodología y organización del trabajo

El presente trabajo ha sido realizado con base en soluciones, en lo posible, con software libre y multiplataforma ya que se aumentan las posibilidades de reestructurar el proyecto en el futuro mediante herramientas que permiten prolongarse en el tiempo y que por lo tanto facilitan el mantenimiento y actualización del resultado final sin tener mayores implicaciones

económicas.

Esta tesis aporta al estado del arte muchos valores añadidos, entre ellos el uso de kiCAD para la elaboración de esquemáticos y circuitos impresos, la construcción de una tarjeta embebida con programador incorporado, el uso de la versión más actualizada del entorno de desarrollo integrado MPLAB® X, el uso del nuevo compilador de C para sistemas embebidos (XC16) y la combinación de código fuente en C con código fuente en ensamblador utilizando las herramientas más modernas proporcionadas por la empresa Microchip.

Este documento fue elaborado con el software de composición tipográfica LaTeX teniendo como base una plantilla para tesis de maestría y doctorado de la Universidad Nacional de Colombia[1]; el contenido está estructurado en siete capítulos así:

- El presente capítulo menciona las motivaciones para realizar este trabajo de grado, los objetivos y la estructura en que está organizada la información.
- El segundo capítulo introduce los conceptos teóricos básicos acerca de señales de audio, procesamiento digital de señales, la tecnología dsPIC® y el estado del arte del procesamiento digital de audio mediante dsPIC®.
- El tercer capítulo trata sobre el diseño del hardware propuesto para el procesamiento digital de audio.
- En el cuarto capítulo se muestra cómo es el método para implementar operaciones matemáticas en el dominio temporal con señales de audio y se plantean las primeras prácticas de laboratorio.
- En el quinto capítulo se exponen los fundamentos del diseño de filtros digitales y se muestran ejemplos realizados para la tarjeta desarrollada.
- El sexto capítulo está dedicado a la Transformada Discreta de Fourier y su implementación en dsPIC® con base en el algoritmo FFT.
- En el séptimo y último capítulo se hacen las respectivas conclusiones y recomendaciones para futuros trabajos.

Los objetivos son abarcados en los capítulos tercero, cuarto, quinto y sexto, al final de cada uno de esos capítulos se muestran los resultados.

Al final de este documento se ha añadido un apéndice dedicado a la creación de proyectos para dsPIC® a través del Entorno de Desarrollo Integrado MPLAB® X.

[1] Plantilla disponible en https://www.overleaf.com/latex/templates/plantilla-tesis-maestria-y-doctorado-universidad-nacional-de-colombia/bvkytfhxsskk#.V8vDVjUoNPo

2. Marco teórico y estado del arte

La mayoría de señales estudiadas en ciencias e ingeniería son de naturaleza analógica [1] y se representan como funciones continuas; sin embargo, el procesamiento digital requiere trabajar con señales de naturaleza discreta. Una señal de audio es la representación eléctrica de las ondas generadas por los sonidos, aunque en los inicios de la electrónica las señales de audio eran de naturaleza continua, la electrónica actual ha mostrado que el audio tratado en forma discreta tiene unos alcances impresionantes, por eso, actualmente la tendencia es el tratamiento de las señales de audio en forma digital. Un dsPIC®, es un microcontrolador con un núcleo DSP capaz de realizar en un solo ciclo de máquina operaciones comunes del procesamiento digital de señales, con lo cual también puede procesar señales de audio. Este capítulo muestra un panorama de las señales de audio y los dsPIC®.

2.1. Importancia de las señales de audio

El sonido es uno de los fenómenos físicos más importantes en la naturaleza, en los seres humanos, además de otros sentidos como la vista, el olfato, el gusto y el tacto está el sentido del oído, con el cual podemos comunicarnos con otros individuos mediante las señales de voz. Los sonidos emitidos por los fenómenos naturales son también elementos de comunicación de la naturaleza. Los seres humanos estamos condicionados por los fenómenos acústicos, la risa, el llanto, la música, son sólo algunos ejemplos de la influencia que tienen los sonidos sobre nosotros. La pérdida de la capacidad de oír, conocida como sordera, afecta de forma muy notable la calidad de vida de quien padece esta incapacidad. La ciencia avanza en esta rama para tratar de mejorar la calidad de vida de estas personas, por lo tanto, el estudio de las señales de audio en conjunto con el funcionamiento del sentido del oído son un campo de gran importancia para las investigaciones modernas.

2.2. Teoría de las señales de audio

El procesamiento digital de señales tiene una rama muy importante que es el procesamiento digital de audio. Las principales características de este tipo de señales son:

- Unidimensional
- Naturaleza continua
- Rango de frecuencia de 20 Hz a 20 KHz [15]

Una señal de audio analógica es una réplica fiel de la onda sonora que representa, mientras que una señal de audio digital consiste en la discretización en tiempo y amplitud de la señal de audio continua, esto significa que es una señal muestreada y la frecuencia de muestreo influye directamente en la calidad misma de la señal. Una señal discreta en amplitud significa que es una señal cuantizada y el tamaño de cada muestra impacta directamente sobre los parámetros del sistema de procesamiento tales como la memoria de almacenamiento y la arquitectura de dichos sistemas.

El procesamiento digital de la señal de audio implica que ésta deba convertirse de analógica a digital, el dispositivo que se encarga de muestrear y cuantizar la señal se llama conversor analógico-digital (ADC), este proceso implica la pérdida por cuantización y en consecuencia será imposible reconstruir una señal realmente exacta a través de los datos obtenidos por dicho conversor. El rango dinámico de una señal está estrechamente relacionado con la resolución de la misma, como punto de referencia tenemos el intervalo dinámico completo del oído humano, el cual supera los 120 dB, un sistema de procesamiento digital de audio generalmente tiene un rango dinámico mucho menor.

La señal de audio hace parte del grupo de señales a las que se puede aplicar las tres operaciones básicas en el dominio del tiempo: escalamiento, retraso y adición. También se le puede aplicar operaciones en el dominio de la frecuencia, siendo la más representativa la Transformada Discreta de Fourier.

Las señales de audio tienen subgrupos, entre los más importantes se puede destacar. Señales de voz y señales de instrumentos musicales. Un panorama general de las aplicaciones del procesamiento de audio, muestra aplicaciones en los siguientes campos:

- Grabación sonora: compresores, limitadores, expansores, compuertas de ruido, ecualizadores, filtros, reducción de ruido, efectos musicales.
- Telefonía: DTMF, cancelación de eco, compresión de voz, transmisión masiva de voz en tiempo real, voz sobre IP (VOIP), reconocimiento del habla, encriptado de voz.
- Síntesis de sonidos: Síntesis musical, música electrónica, síntesis de voz, conversión texto a voz.

En cuanto a las frecuencias de muestreo y cantidad de bits por muestra, la Tabla **2-1** resume los formatos más utilizados.

Tabla 2-1.: Frecuencias de muestreo y resoluciones típicas en los sistemas de audio.

Calidad de sonido	**Frecuencia de muestreo**	**Profundidad de sonido**
Telefónica	8000 Hz	1 x 8 bits (mono)
Calidad Radio AM	11025 Hz	1 x 8 bits (mono)
Calidad Radio FM	22050 Hz	1 x 16 bits (mono)
CD estéreo	44100 Hz	2 x 16 bits (estéreo)
Calidad DAT	48000 Hz	2 x 16 bits (estéreo)

Vemos que las señales de audio comunes suelen tener uno o dos canales y son llamadas monofónicas y estereofónicas respectivamente; también existen configuraciones con más canales.

2.3. Procesamiento analógico de señales de audio

Básicamente, un procesamiento de señales de audio se hace para mejorar la calidad de la señal, o para extraer información importante de la misma. En el auge de la electrónica analógica surgieron muchas técnicas que permitieron variados tipos de procesamiento analógico de las señales de audio con aplicaciones en música y telefonía principalmente. Los filtros analógicos son un claro ejemplo del alcance de este tipo de procesamiento, en cuanto a

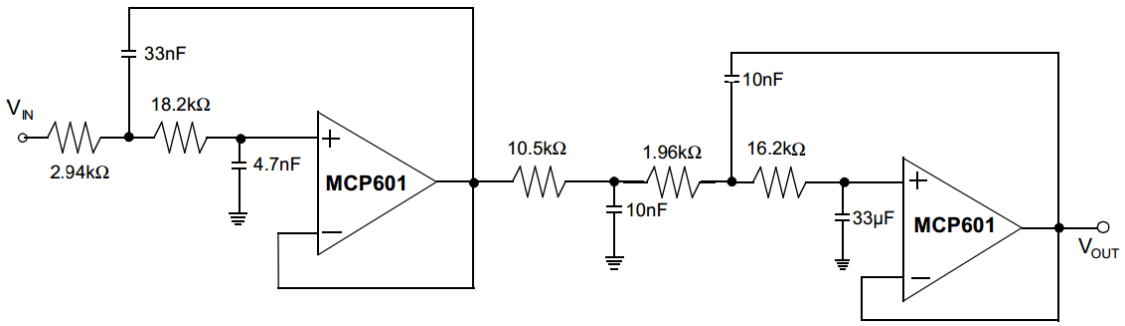

Figura 2-1.: Filtro Bessel de quinto orden como ejemplo de sistema de procesamiento analógico [5, p. 8].

la telefonía, un avance interesante es el vocoder, que es un codificador de voz que disminuye notablemente la cantidad de información necesaria para almacenar o transmitir un mensaje de voz.

En la Figura **2-1** se muestra el esquema de un típico procesador analógico llamado filtro anti-aliasing, el cual modifica el espectro de una señal.

Los elementos básicos que permiten el procesamiento analógico son los capacitores y los inductores.

2.4. Procesamiento digital de señales de audio

El procesamiento digital de señales ha demostrado su gran cantidad de ventajas frente al clásico procesamiento analógico. La mayoría de señales estudiadas en ciencias e ingeniería son de naturaleza analógica y se representan como funciones continuas; sin embargo, el procesamiento digital requiere trabajar con señales de naturaleza discreta con cierta tasa de muestreo. Este procesamiento implica hacer operaciones matemáticas discretas sobre una señal que es de naturaleza continua. Esto implica que todo sistema de procesamiento de señales en tiempo real debe tener dispositivos de conversión analógico-digital y digital-analógico para que las señales puedan entrar y salir del sistema de procesamiento digital. En la Figura **2-2** se muestra un clásico ejemplo de procesamiento de señales digitales, en este caso se trata de un filtro de respuesta finita al impulso:

Los elementos básicos que permiten el procesamiento digital son los bloques de retardo, los multiplicadores y lo sumadores.

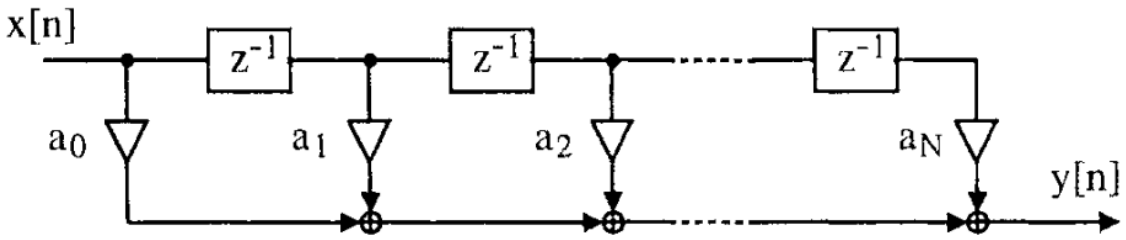

Figura 2-2.: Filtro FIR como ejemplo de sistema de procesamiento digital [15, p. 101].

2.5. Ventajas y desventajas del procesamiento digital de audio

Una gran ventaja es la precisión, las señales digitales son altamente confiables en cuanto a inmunidad al ruido, conservación de características a lo largo del tiempo, almacenamiento y operaciones matemáticas perfectamente definidas con resultados bien predecibles. La estandarización de los sistemas de procesamiento es más sencilla debido a que estos sistemas son más estables, sin las típicas desviaciones de sus parámetros por causas de envejecimiento, cambios de temperatura, tolerancia de los componentes, etc. Los sistemas digitales evitan los problemas de acople de impedancias propios de los circuitos analógicos.

La primera desventaja a tener en cuenta es la pérdida por efectos de cuantización, y por frecuencia de muestreo. Esto imposibilita la reconstrucción exacta de una señal, aunque existen técnicas de interpolación. Otra desventaja es el efecto llamado latencia, que consiste en el tiempo en que una muestra de la señal entra al sistema hasta que sale de éste luego de su procesamiento, esto es consecuencia de la velocidad de procesamiento del sistema y del tamaño de los búfer que contienen las muestras a procesar.

2.6. Procesadores digitales de señal (DSP)

Un DSP (Procesador Digital de Señales) es un dispositivo que recibe señales tomadas del mundo real (voz, audio, video, temperatura, presión, posición, etc.) para manipularlas matemáticamente mediante operaciones discretas. Un DSP recibe señales de un ADC, modifica o analiza las señales para finalmente enviar la señal modificada a una etapa de conversión digital a analógica mediante un dispositivo denominado DAC. Se puede definir un procesador digital de señales DSP como un microchip diseñado para resolver un conjunto de operaciones matemáticas sobre una señal continua o analógica pero expresada en forma digital, el esquema general de este tipo de sistemas se ilustra en la Figura **2-3** y un sistema típico de aplicación se ilustra en la Figura **2-4**.

Figura 2-3.: Esquema general de un sistema DSP [3, p. 4].

Figura 2-4.: Aplicación típica de un DSP: grabación y reproducción de audio mp3.

El procesador digital de señales es actualmente una de las herramientas tecnológicas más importantes y poderosas, cada vez son más las aplicaciones que derivan de esta tecnología, como ejemplo podemos percibir cómo los algoritmos de grabación, compresión y reproducción de audio permiten optimizar el rendimiento de los modernos dispositivos y esto se logra gracias a la tecnología DSP.

El oído humano tiene la capacidad de captar sonidos que van desde los 20 Hz hasta los 20 kHz, y este rango se toma como referencia para el diseño de procesadores de audio ya que la frecuencia de la señal que se va a tratar tiene implicaciones en las características del hardware.

La frecuencia de muestreo es la cantidad de muestras de señal tomadas en un segundo, el criterio de muestreo de Nyquist-Shannon establece que la señal a muestrear debe estar limitada en banda y que la frecuencia de muestreo debe ser al menos dos veces mayor que la frecuencia de la señal a procesar, esto implica que para que una señal de audio pueda ser procesada y reconstruida en toda su banda, su muestreo debe ser de por lo menos 40 KHz; por eso un formato común de audio utiliza tasas de muestreo de 44,1 KHz, pudiéndose encontrar tasas mayores en dispositivos de alta fidelidad o tasas menores en bandas limitadas como la de la voz humana, la cual comúnmente tiene componentes de frecuencia de máximo 4 kHz. La Figura **2-5** ilustra el proceso de muestreo de una señal analógica.

La resolución de cada muestra es también indicadora de la calidad de audio, para formatos comunes como el WAV, la resolución suele estar expresada en 16 bits para sonido monofónico y 32 bits para sonido estéreo. La Figura **2-6** ilustra cómo la señal se aproxima más a la

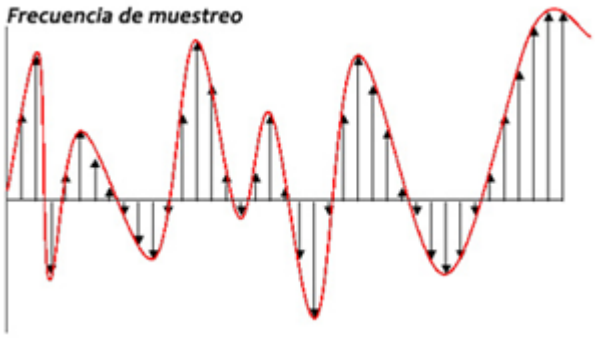

Figura 2-5.: Muestreo de una señal analógica.

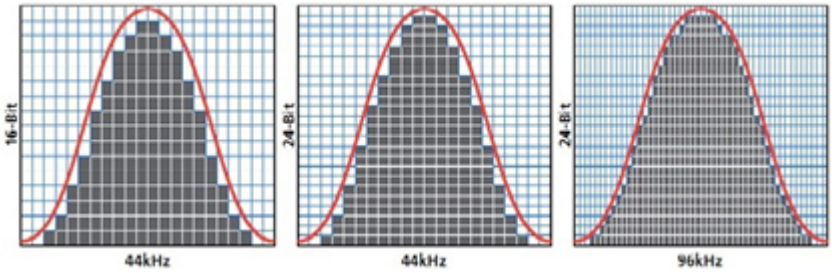

Figura 2-6.: Influencia de la frecuencia de muestreo y la resolución en la calidad de una señal.

original si se aumenta los parámetros de frecuencia de muestreo y de resolución.

La tecnología digital impone límites de velocidad de procesamiento, que, aunque cada vez menos restrictivos, determinan los anchos de banda de señales que pueden ser tratadas digitalmente. La selección de un DSP adecuado se basa principalmente en formato aritmético, velocidad, organización de la memoria, arquitectura interna, costo, entre otros.

2.7. Controladores digitales de señal (DSC)

Un DSC, en términos sencillos, es un microcontrolador que incluye hardware especializado para realizar operaciones de procesamiento digital de señales. La empresa encargada de introducir el término DSC es Microchip y sus productos comerciales son llamados dsPIC®. A continuación se explicará el funcionamiento de la tecnología dsPIC®, su estudio se puede hacer en 3 partes:

- Unidad central de proceso (CPU)
- Integración del sistema

- Periféricos

La unidad central de proceso se encarga de las características básicas esenciales para el control del dispositivo (CPU, memoria de datos, memoria de programa, interrupciones).

La integración del sistema son todas las características adicionales que dan flexibilidad al dispositivo, diferentes modos de operación y mecanismos de control en caso de eventos especiales (Programación de la memoria flash, oscilador, reset, perro guardián, modos de ahorro de energía, seguridad CodeGuard, programación y diagnóstico, configuración del dispositivo).

Los periféricos son todos aquellos circuitos auxiliares que conectan con la CPU. (Puertos E/S, temporizadores, captura de entrada, salida de comparación, conversores ADC y DAC, UART, SPI, I^2C^{TM}, DCI, QEI, $ECAN^{TM}$, DMA, etc).

Entre la documentación importante que soporta el desarrollo con dsPIC® está el manual de referencia de la familia específica, el manual de referencia del programador, la respectiva hoja de características del dispositivo específico, las notas de aplicación, los códigos de ejemplo y demás información presente en el sitio web del fabricante.

2.7.1. Unidad central de proceso (CPU)

La unidad central de proceso se basa en una arquitectura Harvard modificada de 16 bits (Figura **2-7**), la principal característica de esta CPU es el soporte para operaciones de procesamiento digital de señales. Las instrucciones tienen un tamaño de 24 bits, el contador de programa (PC) tiene 24 bits y puede manejar hasta 4M x 24 bits de memoria de programa (Figura **2-8**).

Los registros de trabajo son de 16 bits (Figura **2-9**), estos registros pueden operar como datos, direcciones o registros de desplazamiento de direcciones.

Existen 2 tipos de instrucciones manejadas por el dsPIC®: MCU y DSP. El primer tipo es el que comúnmente usan los microcontroladores el otro tipo es el especializado en el procesamiento digital de señales, estas instrucciones admiten varios modos de direccionamiento y fueron optimizadas para el lenguaje C.

El espacio de datos (Figura **2-10**) puede direccionar hasta 32k palabras ó 64 kbytes. Éste espacio está dividido en dos bloques (memoria X y memoria Y), cada uno tiene su propia unidad de generación de direcciones (AGUX y AGUY). Las instrucciones MCU únicamente pueden acceder a la memoria mediante AGUX, esto es, toda la memoria de datos es accedida

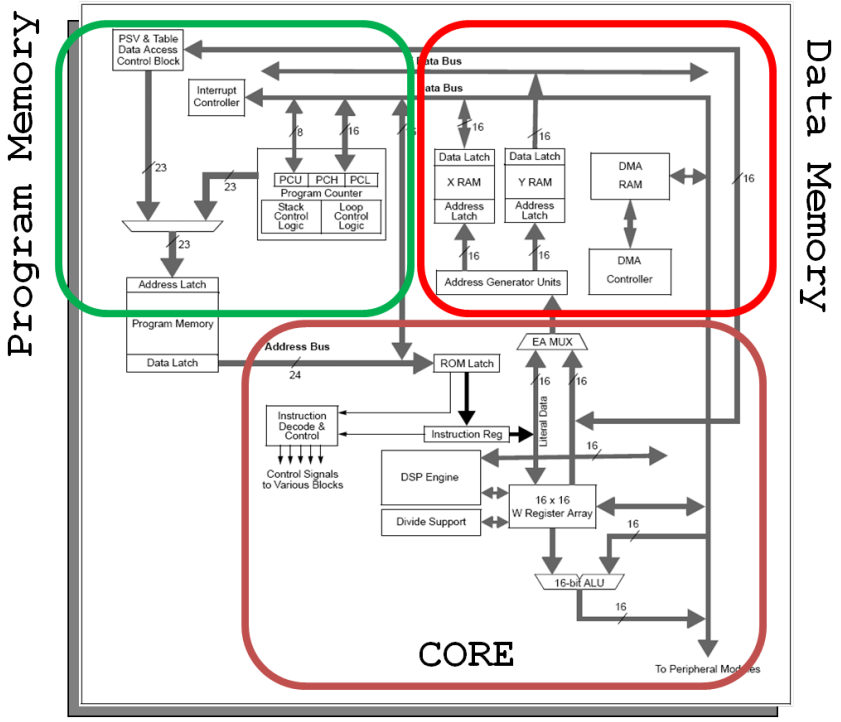

Figura 2-7.: Arquitectura del dsPIC® [17].

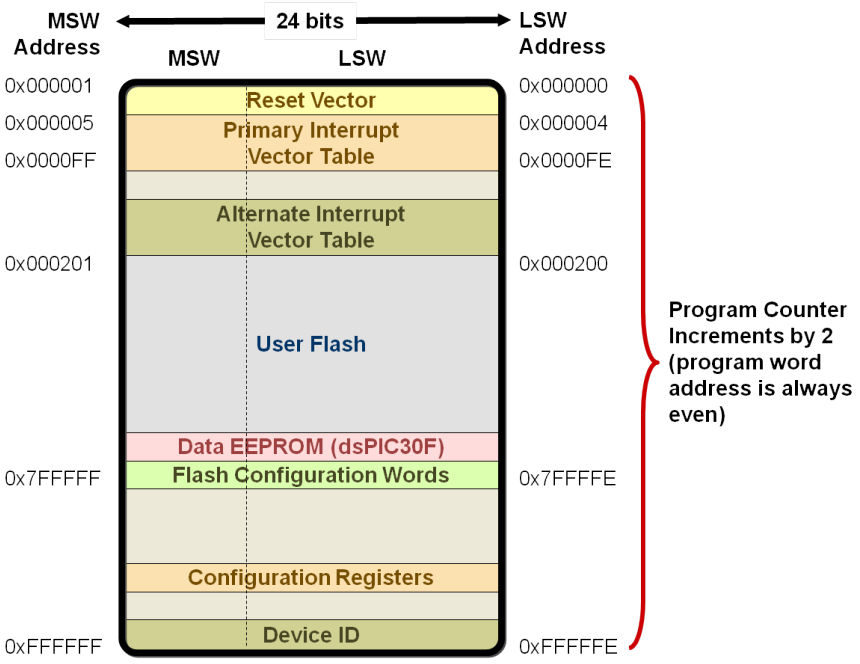

Figura 2-8.: Memoria de programa del dsPIC® [17].

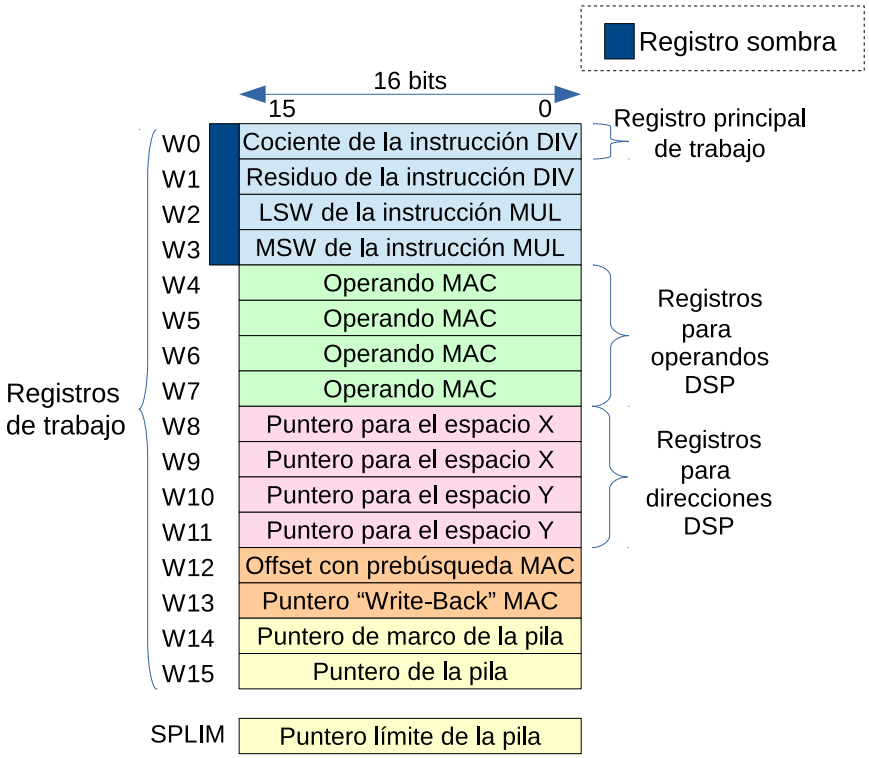

Figura 2-9.: Registros de trabajo del dsPIC® (Adaptado de [17]).

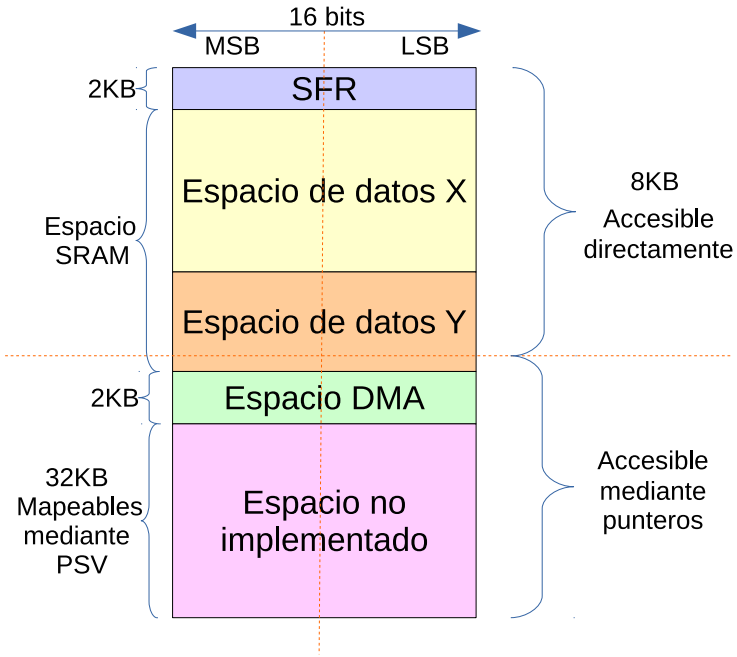

Figura 2-10.: Memoria de datos del dsPIC® (Adaptado de [17]).

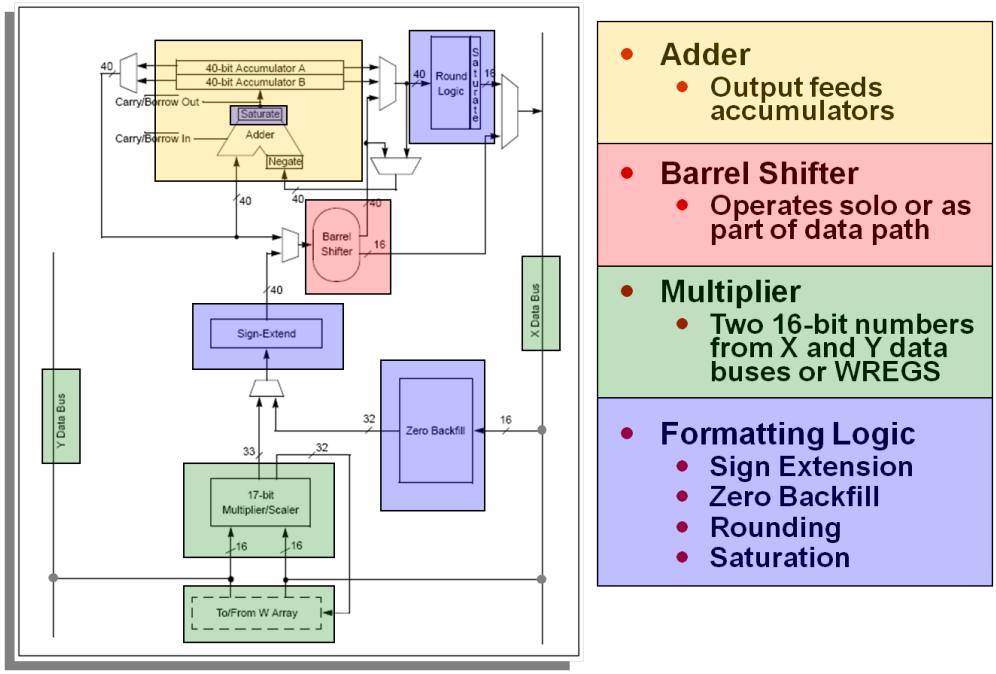

Figura 2-11.: Esquema del motor DSP [17].

como si fuera un único espacio lineal; entre tanto, las instrucciones DSP acceden a la memoria mediante las dos unidades AGUX y AGUY simultáneamente, esto divide la memoria en 2 espacios (X e Y). Un registro llamado PSVPAG permite agregar hasta 16k palabras de la memoria de programa a la memoria de datos, esto permite que los programas puedan operar con parte de la memoria de programa como si fueran espacios pertenecientes a la memoria de datos.

Otras características de la CPU son: soporte para varios modos de direccionamiento, multiplicador de 17 bit por 17 bit, unidad aritmético-lógica (ALU) de 40 bits, dos acumuladores de 40 bits, un registro de desplazamiento bidireccional de 40 bits con capacidad de desplazar hasta 16 bits en un solo ciclo. En la Figura **2-11** puede apreciarse el esquema del motor DSP.

El motor DSP contiene 2 acumuladores de 40 bits cada uno. Hardware para división, multiplicador de 17x17 bits, 16 registros de 16 bits y una amplia variedad de modos de direccionamiento.

2.7.2. Acceso Directo a Memoria (DMA)

Para mejorar el rendimiento del sistema, los nuevos dsPIC® incorporan módulos de acceso directo a memoria DMA (Figura **2-12**)para facilitar el traspaso de datos entre periféricos y memoria RAM, este es un mecanismo que mejora el rendimiento del programa ya que la

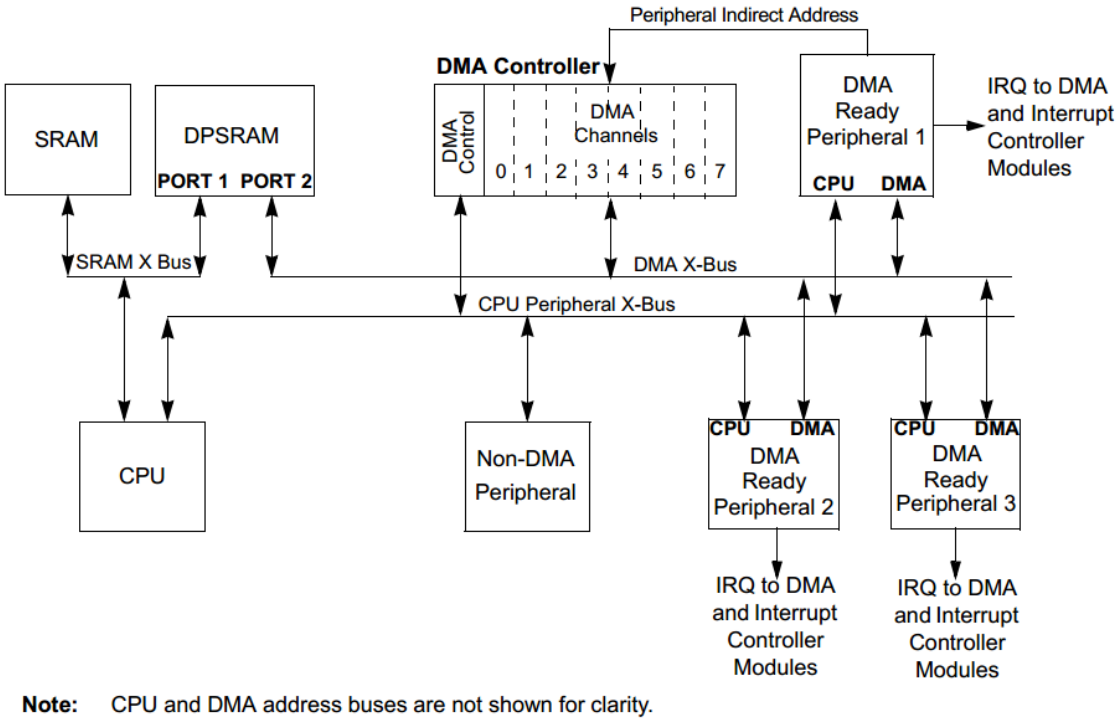

Figura 2-12.: Módulo de acceso directo a memoria [27].

transferencia de los datos se lleva a cabo con mínima intervención de la CPU, esto hace que los programas sean más fluidos por tener menos interrupciones, el módulo DMA copia automáticamente en y desde un espacio de memoria llamado DMA RAM, este mecanismo aumenta enormemente la eficiencia del sistema.

El módulo DMA puede copiar bloques enteros de datos sin requerir ayuda de la CPU, generando una sola interrupción por bloque. Para utilizar eficientemente este módulo, los bloques de información transferidos deben ubicarse en el espacio DMA RAM.

2.7.3. Puertos E/S y tecnología de mapeo de pines

Entre las principales nuevas características añadidas a los dsPIC® destaca el mapeo de pines, el cual permite que muchos de los periféricos puedan ser conectados de forma más flexible a los puertos de entrada salida mediante la técnica de multiplexado. De esta manera, el diseño de hardware se simplifica notablemente, pues ahora es posible conectar muchos periféricos a pines que sean más convenientes, esto implicará un poco más de atención en el diseño de software.

Los puertos de E/S están controlados por 4 registros:

- TRISx (Dirección)

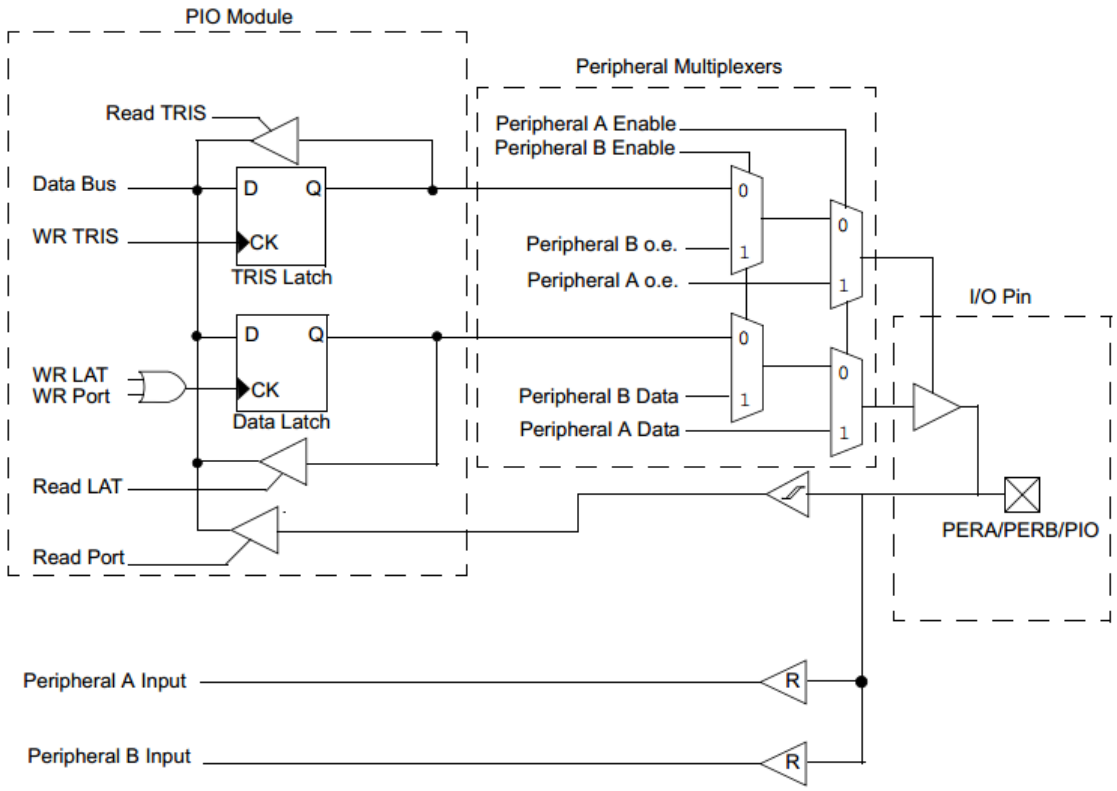

Figura 2-13.: Diagrama de control de los puertos E/S con mapeo de pines [25].

- PORTx (Puerto)
- LATx (Latch)
- ODCx (Open-Drain)

El control de los pines E/S le pertenece al registro PORTx a menos que esté activado un periférico vinculado a dicho pin. Como pueden existir múltiples pines de periféricos que se pueden multiplexar, existe una prioridad que define cuál es el que finalmente tiene el control sobre el puerto. Algunos periféricos pueden leer la salida del registro PORTx, esto causa un particular comportamiento ya que se pueden crear puentes entre la salida de PORTx y la entrada de un periférico. Esta funcionalidad puede resultar útil para comprobar manualmente algunas funciones sin necesidad de usar señales externas. Los periféricos que comúnmente admiten datos desde los registros PORTx son las interrupciones externas, el timer, la captura de entrada y los pines de fallo de PWM. Para un funcionamiento normal es recomendable que todos los pines mapeados sean configurados como entradas mediante el registro TRISx. La Figura **2-13** muestra cómo está constituido el módulo E/S del dsPIC®.

Los periféricos que tienen la posibilidad de mapeo de pines son llamados periféricos remapeables y no tienen un pin específico, sino que debe asignarse por software el pin que se

desee. La hoja de características detalla cuales pines están disponibles para mapearse con periféricos. Después de realizada la asignación correspondiente, el periférico mapeado tiene la prioridad sobre el pin asignado excepto si el pin tiene funciones analógicas. Para que un pin mapeado funcione correctamente, debe configurarse como E/S digital incluso después de un reset.

El mapeo de periféricos se realiza por medio de dos sets de registros, uno dedicado a las entradas y el otro a las salidas. El bit OSCCON¡6¿permite el mapeo de pines si su estado es 0 y lo bloquea si su estado es 1. Según el manual de referencia, el tiempo de bloqueo y desbloqueo del mapeo de pines es crítico y debería hacerse siempre con líneas escritas en ensamblador, por eso, el compilador XC16 incorpora funciones built-in que pueden ser usadas para este fin.

La prioridad de las líneas E/S está definida en el diagrama de pines a través del nombre del pin. Las funciones de la izquierda tienen mayor prioridad sobre las funciones de la derecha. Las líneas también tienen resistores internos de pull-up y pull-down que se pueden configurar usando los registros CNPUx y CNPDx respectivamente.

2.8. Software de desarrollo para dsPIC®

Aunque existen varias empresas que ofrecen software para el desarrollo de aplicaciones con dsPIC®, la tarjeta de procesamiento digital de audio será diseñada para implementar algoritmos basándose en las herramientas ofrecidas por la empresa Microchip, las principales razones para tomar esta determinación son:

- La empresa Microchip es el fabricante original del dispositivo dsPIC®, lo cual supone que sus herramientas de desarrollo están mejor enfocadas y soportadas.
- La documentación ofrecida por dicha empresa es suficientemente amplia permitiendo un desarrollo bien instruido para el diseñador tanto de hardware como de software.
- El entorno integrado de desarrollo MPLAB® contiene muchas herramientas útiles que facilitan y mejoran el desempeño del programador; además es multiplataforma, lo que permite llegar a más usuarios.
- Existe una versión estudiantil gratuita de compilador para lenguaje C, el cual es muy potente teniendo en cuenta que la arquitectura de los dsPIC® se ha diseñado de forma que las instrucciones en lenguaje C estén muy bien optimizadas.
- Hay muchas herramientas gratuitas adicionales para el desarrollo de algoritmos entre las que se encuentra, por ejemplo, software de ayuda para el desarrollo de filtros digi-

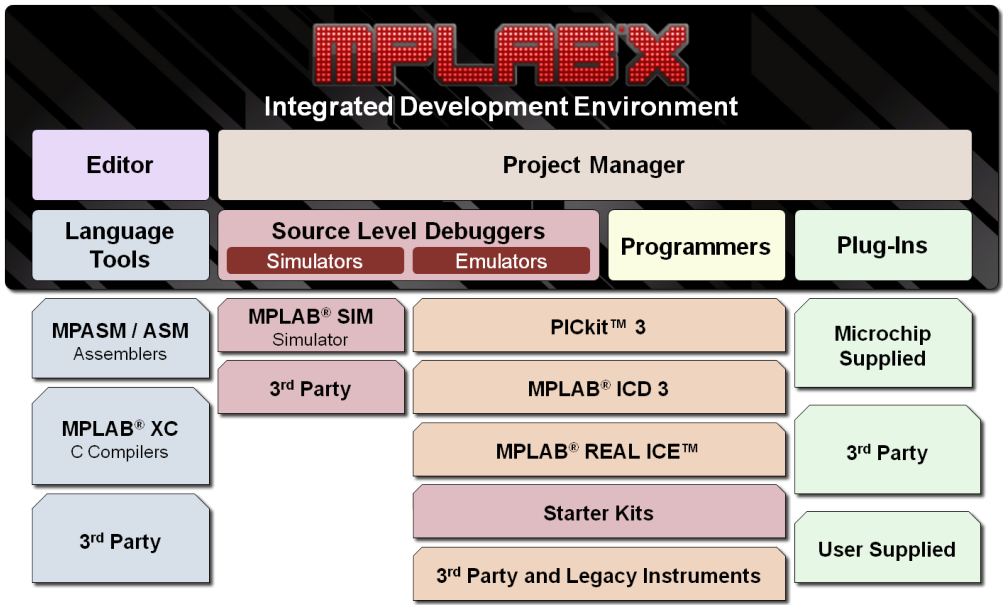

Figura 2-14.: Entorno de desarrollo integrado MPLAB® [18].

tales. Algunas tienen la desventaja de que son limitadas debido a que funciones más avanzadas se ofrecen con versiones de pago.

- Las actuales tarjetas de procesamiento de audio comercializadas por Microchip incorporan herramientas de desarrollo pertenecientes a la misma empresa, lo cual permitirá estudiar mejor la experiencia llevada por estos productos.

Los motivos anteriormente señalados se ajustan a la necesidad de que el producto final sea económico para el usuario final, que contenga un buen soporte y que el uso del mismo no sea muy complicado.

2.8.1. Software MPLAB® X y compilador XC16

El software de desarrollo es ofrecido por Microchip directamente en su página web, con posibilidad de descarga directa. MPLAB® es un entorno de desarrollo integrado gratuito, es el principal software para los productos de Microchip ya que permite desarrollar el código que se implementa en los mismos y acepta la incorporación de herramientas adicionales, actualmente cuenta con la versión denominada MPLAB® X, la cual soporta plataformas basadas en WINDOWS, MAC OS y GNU/LINUX. La Figura **2-14** ilustra el entorno de desarrollo MPLAB®.

Para complementar bien el kit de desarrollo básico, se hace necesario descargar el paquete de compiladores, el que ofrece la empresa es el compilador MPLAB® XC en sus versiones de

8, 16 y 32 bits, en este caso es de interés la versión de 16 bits como consecuencia de la arquitectura de los dsPIC®, por esta razón el compilador requerido es el denominado MPLAB® XC16 que actualmente ofrece la versión 1.24 para las mismas plataformas en que se puede instalar el IDE MPLAB® X. El compilador MPLAB® XC posee una versión estudiantil totalmente gratuita (FREE) y además ofrece una prueba de 60 días de su versión mejorada (PRO), la cual tiene un nivel de optimización muy alto, existe adicionalmente otra versión más económica pero con menos nivel de optimización (Standard).

Tras la instalación del MPLAB® X aparecen tres íconos nuevos en el escritorio: el IDE (principal), Driver Switcher (para intercambiar proyectos con el MPLAB® antiguo) y el IPE (para permitir la programación de dispositivos directamente desde el IDE). Es obligatorio instalar también el compilador ya que el IDE actualmente no incluye ningún ensamblador, el ASM30 está descontinuado, quedando como principal alternativa el ensamblador que viene incorporado con el compilador XC16.

2.8.2. Combinación de archivos C y ensamblador

A continuación se explica cómo combinar archivos .asm con archivos .c. Las funciones de C pasan sus parámetros a otras funciones, incluidas funciones en ensamblador, a través de los registros de trabajo W. Estos registros de trabajo son usados en el orden que son pasados los parámetros.

Los registros W0 a W7 son usados para almacenar los parámetros o los argumentos. Las funciones que no son llamadas durante una interrupción deben preservar el valor de los registros W8-W15.

Las interrupciones pueden ocurrir en cualquier punto del programa, por eso deben preservar todos los registros devolviéndolos al valor original del programa antes de que la interrupción fuera generada cuando la interrupción haya finalizado.

Las variables o funciones declaradas en un archivo ensamblador deben ser declaradas como globales para que puedan ser referenciadas por funciones de C. Los símbolos en ensamblador deben ser precedidos por al menos un guión de piso, estos símbolos serán referenciados desde los códigos en C quitando el primer guión de piso.

En ensamblador existen 3 tipos de secciones. .text para código, .data para variables inicializadas y .bss para variables no inicializadas.

En el lenguaje ensamblador para dsPIC® un literal es un valor numérico que consiste en un operando para la instrucción que lo está utilizando, un literal se define anteponiendo un

símbolo numeral (#).

Cuando en una instrucción, un valor numérico no está antecedido por un símbolo numeral, entonces ese valor será tratado como una dirección y el operando es el número que está apuntado por esa dirección.

Las funciones y variables declaradas en C se asumen inmediatamente en ensamblador por lo que este último no debe declararlas sino utilizarlas directamente precediendo con un guión de piso.

2.8.3. Configuración básica en MPLAB® X de un proyecto con dsPIC®

Una vez creado el proyecto en MPLAB® X, se debe agregar un archivo de código fuente haciendo clic derecho sobre la carpeta *SourceFiles* y seleccionando $New- > CMainFile...$, se da un nombre al archivo y luego se da clic en *Finish*. El IDE incluye por defecto unas bibliotecas pero se debe adicionar manualmente la correspondiente al dispositivo que se vaya a programar. Por ejemplo:

```
#include <dspic33fj128gp802.h>
```

La función main contiene por defecto unos parámetros que podemos suprimir. Una línea más que aparece en la función creada por MPLAB® X es:

```
return (EXIT_SUCCESS);  EXIT_SUCCESS equivale a 0.
```

Los principales registros para comenzar son los relacionados a los puertos, líneas de E/S. Los registros encargados de definir cada pin como entrada o salida son los registros TRISx, en la respectiva hoja de especificaciones está el detalle de los bits correspondientes. Si un bit del registro TRISx está definido como 0, su correspondiente pin asociado se comportará como salida; si dicho bit está definido como 1 entonces ese pin será una entrada. Otro registro muy importante es el AD1PCFGL, el cual define si el pin correspondiente es analógico o digital, un bit igual 0 define al correspondiente pin como analógico y si el bit tiene un valor de 1 su correspondiente pin será digital. En dispositivos con más de 16 entradas analógicas existe otro registro llamado AD1PCFGH para permitir la incorporación y manejo de más pines.

Después de un reset, todos los pines quedan definidos como entradas analógicas. O sea, por defecto todos los bits de los registros TRISx serán 1 y los del registro AD1PCFGL serán 0.

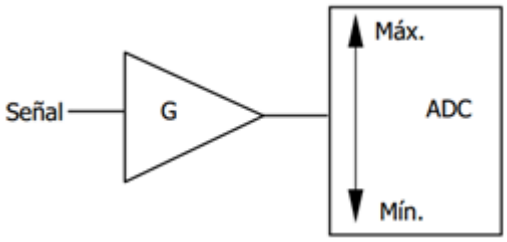

Figura 2-15.: Conexión de amplificador operacional para acondicionamiento hacia un ADC.

2.9. Acondicionamiento de señales de audio

El acondicionamiento de una señal de audio consiste en adaptar y mejorar la señal que se desea adquirir, por micrófono o línea de salida de un equipo de audio, a las características del ADC de modo que se obtenga la mayor calidad posible en la señal a convertir. Esta adaptación tiene tres parámetros principales: amplitud, offset y frecuencia de paso.

La amplitud de la señal está dada por los voltajes de referencia que admite el ADC, en este caso, se hará el diseño en correspondencia con los voltajes de alimentación.

Para lograr este objetivo, el principal dispositivo electrónico existente es el amplificador operacional, el cual permite el ajuste de los tres parámetros anteriormente mencionados. Lo que se busca entonces, es filtrar la señal, para obtener sólo la banda de interés, luego ubicar el offset o compensación para la entrada del ADC y por último ajustar la ganancia para el máximo aprovechamiento del rango sin llegar a la saturación. La conexión del amplificador con la entrada analógica del dsPIC® se hace como en la Figura **2-15**.

El amplificador operacional puede ser configurado como un filtro pasa bajas para este ajuste, diferentes topologías pueden ser aplicadas para este fin, la Figura **2-16** es un ejemplo tomado de la nota de aplicación AN682 de Microchip, consiste en un filtro pasa bajas Butterworth de segundo orden con frecuencia de corte de 10 kHz, este esquema es conocido como configuración Sallen-Key.

2.10. Tarjetas de procesamiento digital de audio con productos de Microchip

Teniendo en cuenta únicamente microcontroladores y controladores digitales de señal, la Tabla **2-2**[1] resume las aplicaciones de audio que pueden ofrecer diferentes familias de productos.

[1]Adaptada de la original encontrada en www.microchip.com

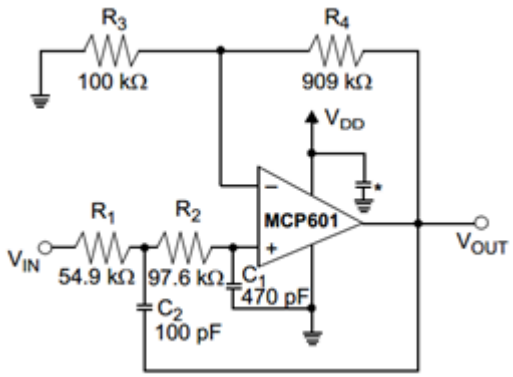

Figura 2-16.: Filtro pasa bajas Butterworth con frecuencia de corte de 10 kHz.

Tabla 2-2.: Aplicaciones de audio con productos de Microchip.

Familia	Aplicación de audio
PIC10, PIC12, PIC16	Zumbadores, Alarmas, generación de tonos
PIC16, PIC18	Grabación y reproducción (ADPCM, G.711)
PIC24	Grabación y reproducción (ADPCM, G.711)
dsPIC® DSC	Grabación y reproducción (ADPCM, G.711, G.726[a], Speex), Cancelación de eco acústico, supresión de ruido, cancelación de eco telefónico, reconocimiento de habla.
PIC32	Reproducción (Speex, ADPCM), grabación (PCM, ADPCM), decodificación MP3, decodificación de AAC, transmisión de audio (Bluetooth, USB).

Con base en la Tabla **2-2** y la Figura **2-17** puede concluirse que las mejores posibilidades están en las famiias dsPIC® DSC y PIC32, la primera tiene una arquitectura de 16 bits y la segunda es de 32 bits. Diferentes criterios permiten definir a los dsPIC® como los más apropiados.

Los PIC32 pueden tener internamente memorias flash de hasta 2 MB permitiendo que puedan almacenar segmentos de audio, es óptimo para el control de las señales de audio pero la desventaja está en la ausencia de un motor DSP que permita realizar óptimamente operaciones de procesamiento de señales, generalmente en esta familia los productos son más caros. Aunque la familia dsPIC® presenta algunas desventajas como la ausencia de memorias flash de almacenamiento de propósito general, el motor DSP les proporciona un rendimiento superior para tareas de procesamiento de señales en relación a su costo, el bus de datos es más reducido (16 bits) pero está acorde con el estándar de la mayoría de señales de au-

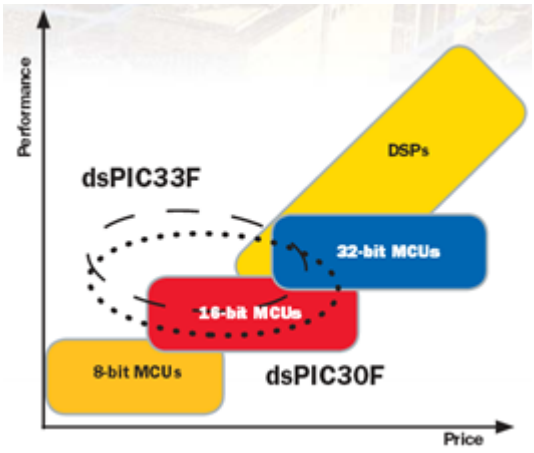

Figura 2-17.: Relación precio vs rendimiento de los productos de Microchip.

dio y finalmente lo que lo hace inmejorable para la aplicación que se busca en este trabajo es la inclusión de productos con conversores DAC de 16 bits para audio, módulos DCI para trabajar con Codecs y módulos PWM avanzados para aplicaciones de audio de bajo costo.

Es así como existen múltiples configuraciones para el tratamiento de las señales de audio resumidas en la Tabla **2-3**.

Tabla 2-3.: Diferentes arquitecturas de los convertidores ADC y DAC para sistemas DSP.

Entrada de audio	**Salida de audio**	**Ventajas**	**Desventajas**
ADC externo	Modulación PWM	Económica forma de conversión digital a analógico.	Aumenta el número de líneas E/S del procesador.
ADC externo	DAC externo	Dependiendo de las características de los conversores se pueden lograr amplias resoluciones y rangos dinámicos.	Aumenta el número de líneas E/S del procesador. Incrementa notablemente el tamaño del circuito impreso.
ADC externo	DAC interno	Es una buena alternativa para buscar mejor resolución en la señal de entrada. Permite ajustar formatos de ADC y DAC para ser compatibles.	En general, los DAC internos no tienen una calidad suficiente para justificar ADC?s con características avanzadas.
ADC interno	Modulación PWM	Es una configuración muy económica y sencilla de implementar.	La calidad del audio es baja, con mucha distorsión.
ADC interno	DAC externo	Es sencilla de implementar para el dsPIC mediante protocolos tales como el SPI.	Mayor uso de líneas E/S del procesador.
ADC interno	DAC interno	Es la configuración más autosuficiente, requiere mínima circuitería externa.	En general, un sistema de este tipo tiene limitaciones en la calidad del audio.
Codec de audio	Codec de audio	Mejor calidad de la señal, el procesador está totalmente exento del manejo de las señales analógicas.	Puede incrementar de manera significativa el costo del proyecto, no es factible si se busca un sistema de bajo costo.

El proyecto será más factible si el procesador es más autosuficiente, esto significa que minimizar el uso de componentes externos tendrá más beneficios. En el sitio web de la empresa Microchip se menciona que los conversores ADC de 12 bits que incluyen los dsPIC® pueden ser suficientes en muchos casos para el tratamiento de señales de voz pero recomiendan el uso de un CODEC si se busca mejorar la resolución. Teniendo en cuenta que se pretende hacer procesamiento de señales con enfoque didáctico, se buscará la configuración más autosuficiente posible que permita el logro de este objetivo.

Buscando exhaustivamente en internet se comprobó que el mercado de tarjetas para aplicaciones de audio con productos microchip está principalmente dominado por dos empresas:

- Microchip Technologies
- MikroElektronika

Ambas empresas emplean dispositivos de muchos fabricantes diferentes para complementar los periféricos de los microcontroladores o controladores digitales de señal, y ensayan con diferentes tipos de arquitectura.

A continuación se hace una síntesis de la oferta para aplicaciones de audio.

2.10.1. EasyPIC Fusion™v7

Esta tarjeta (Figura **2-18**)de la empresa MikroElektronika soporta dispositivos dsPIC33, PIC24 y PIC32, tiene conectores de audio para entrada de micrófono y salida para audífonos de hasta 30 ohmios de impedancia; el CODEC que maneja esta etapa es el circuito integrado VS1053 de la empresa VLSI Solution y es principalmente utilizado para la decodificación de múltiples formatos como el OGG Vorbis, mp3, WMA, WAV, entre otros. Tiene funcionalidades de grabación en distintos formatos, los ADC y DAC integrados son de tipo estéreo.

2.10.2. Mikromedia for dsPIC33

Esta tarjeta de desarrollo (Figura **2-19**) está diseñada principalmente para interactuar con una pantalla TFT de 320x240 incluida. No tiene opciones de entrada de audio, solamente de salida a través de un decodificador de audio (VS1053) con interface SPI. El procesador que utiliza es un dsPIC33FJ256GP710A con bootloader para recibir nuevos programas a través del puerto UART.

2.10.3. MIKROMEDIA for dsPIC33EP

Es básicamente el anterior pero usando un procesador mejorado dsPIC33EP512MU810.

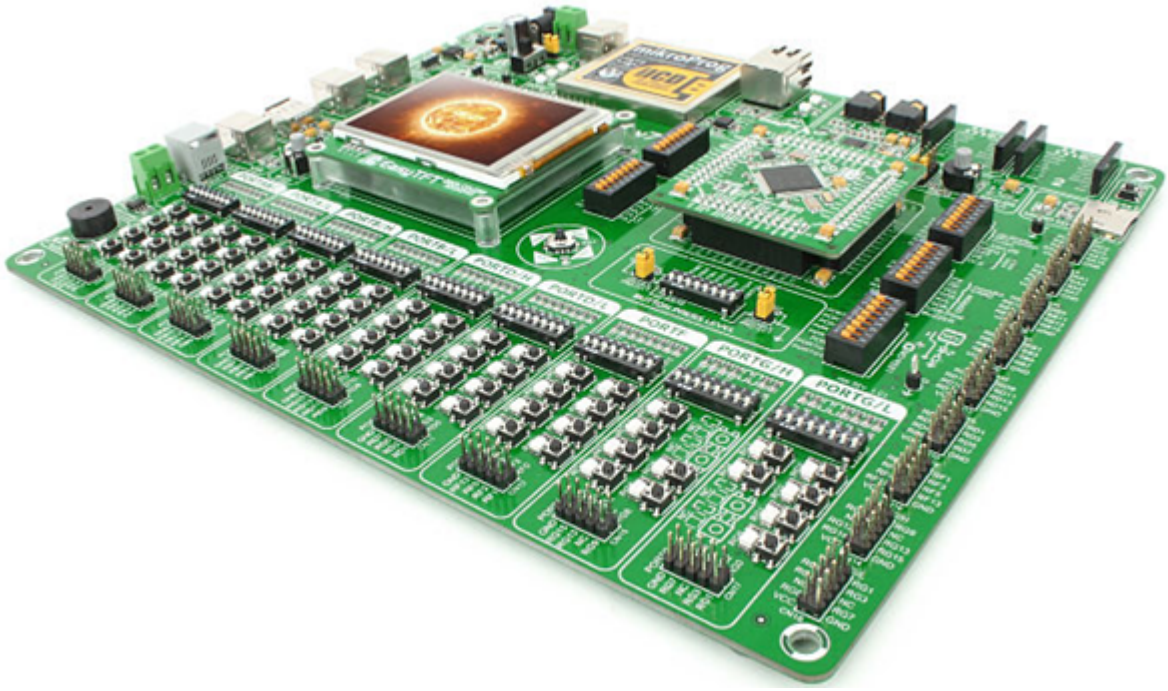

Figura 2-18.: Tarjeta EasyPIC Fusion^{TM}v7.

Figura 2-19.: Tarjeta Mikromedia for dsPIC33.

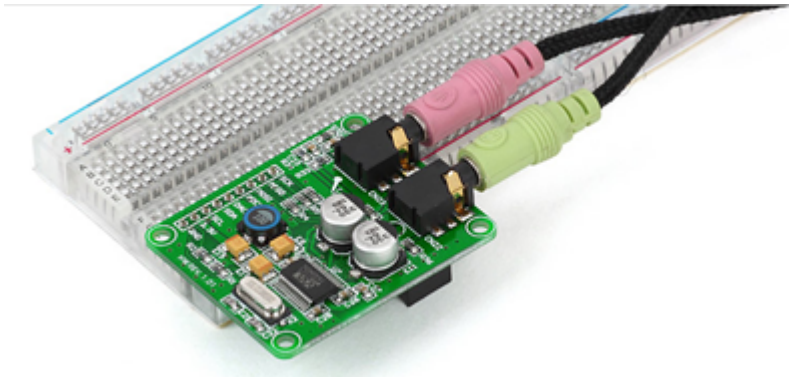

Figura 2-20.: Tarjeta Audio Codec Board - PROTO.

2.10.4. Mikromedia for PIC24 y Mikromedia for PIC24EP

En este caso, los procesadores usados son el PIC24FJ256GB110 y el PIC24EP256GU810 respectivamente.

2.10.5. Audio Codec Board - PROTO

MikroElektronika también cuenta con una serie de tarjetas complementarias especializadas en la codificación de audio, no son procesadores sino interfaces para las señales analógicas, entre ellas se destaca una tarjeta de extensión)(Figura **2-20** que usa el códec WM8731.

2.10.6. dsPICDEM 1.1 Plus

Este producto, de la empresa Microchip consiste en una tarjeta de desarrollo (Figura **2-21**) de propósito general con amplio enfoque hacia las señales de audio. Se han hecho múltiples investigaciones en este sistema, siendo muy importantes las enfocadas al reconocimiento de voz. La interfaz analógico-digital se logra mediante el códec de audio SI3000, el cual está diseñado para la banda de voz. El corazón del sistema es el dsPIC30F6014A.

2.10.7. MPLAB starter kit for dsPIC DSCs

Es un procesador digital de audio diseñado para quienes se inician en esta tecnología, tiene un enfoque principalmente en señales de voz, en los programas de ejemplo se muestra la técnica de compresión basada en el algoritmo $G,711\mu - law$. Como características principales se destacan: un códec de audio que permite que el dsPIC® sólo se preocupe por el control digital, una memoria Flash de 8 bits, bootloader. El códec usado es un WM8510, que permite trabajar con frecuencias de muestreo de hasta 48 KHz, pero las librerías escritas para esta tarjeta permiten sólo hasta 16 KHz.

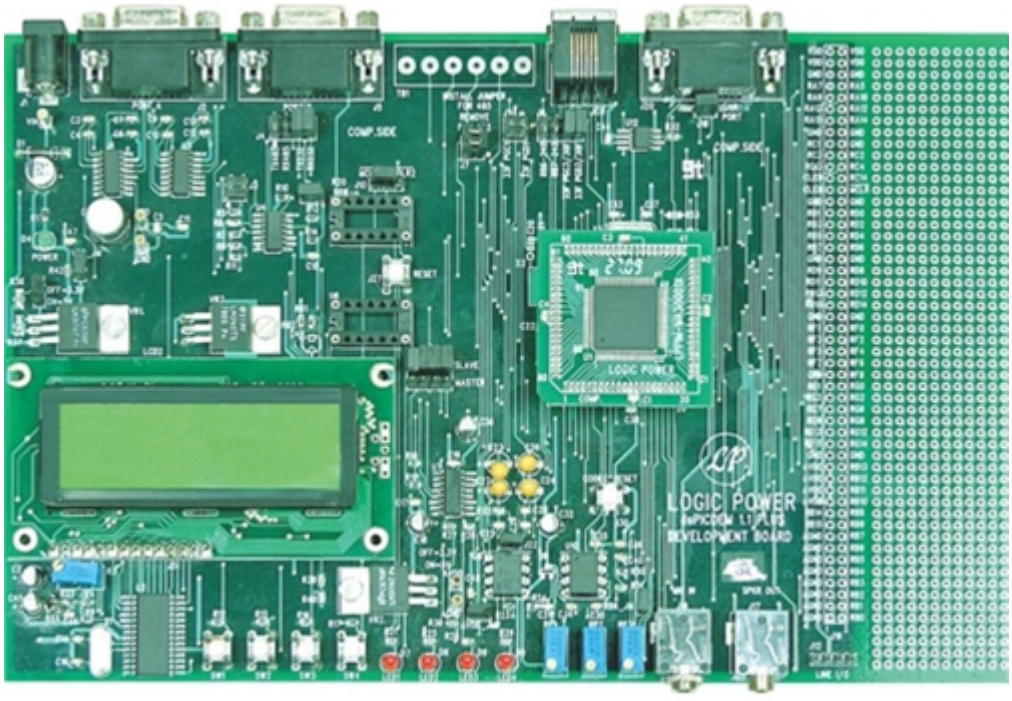

Figura 2-21.: Tarjeta dsPICDEM 1.1 Plus.

Figura 2-22.: Tarjeta MPLAB starter kit for dsPIC DSCs.

Esta tarjeta de desarrollo (Figura **2-22**) ya puede considerarse obsoleta debido a que sólo permite ser programada con el MPLAB® 8, el cual es un software que ha sido descontinuado. La principal dificultad de actualización está dada por el bootloader, el cual no tiene soporte para entornos de desarrollo diferentes. Otra característica que hace considerar que esta herramienta está desactualizada es que el procesador utilizado (dsPIC33FJ256GP506) no tiene las nuevas características tecnológicas de los dsPIC® tales como el mapeo de pines y el acceso directo a memoria.

2.10.8. Audio Development Board for dsPIC33E

Esta (Figura **2-23**) puede considerarse como una herramienta de desarrollo plenamente actualizada, está diseñada para permitir su interfaz con otra tarjeta de desarrollo llamada PICtail plus. Entre sus principales características están: soporte para audio USB, pantalla TFT de 2 pulgadas, códec de audio WM8960, procesador dsPIC33EP512MU810 y micrófono incorporado, entrada de audio y salida para audífonos.

2.10.9. Otros trabajos

Se ha podido determinar que muchas personas tienen proyectos personales con dsPIC® para procesamiento de señales de audio, algunos de estos proyectos son accesibles desde internet, sus autores publican con o sin ámbito académico. Como ejemplo está la empresa «cumbria designs»[2] que está construyendo prototipos de procesadores de audio con dsPIC®, uno de ellos es llamado «minifilter DSP Audio Processor» (Figura **2-24**) el cual utiliza como procesador el DSPIC33FJ128GP802, y otro producto es el «Eden DSP» (Figura **2-25**) con diferentes versiones diseñadas para DSC de tecnología SMT.

[2]http://www.cumbriadesigns.co.uk/

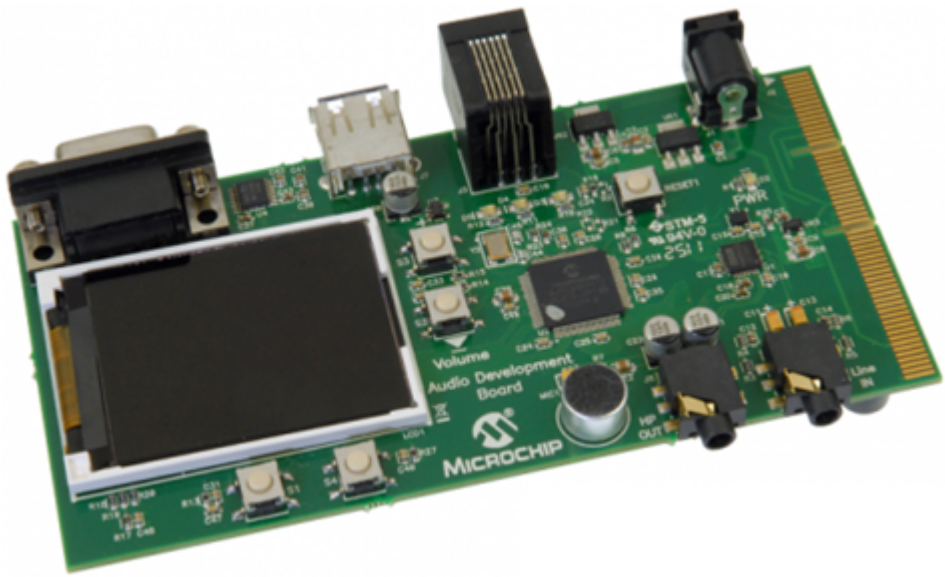

Figura 2-23.: Tarjeta Audio Development Board for dsPIC33E.

Figura 2-24.: Tarjeta Minifilter DSP Audio Processor.

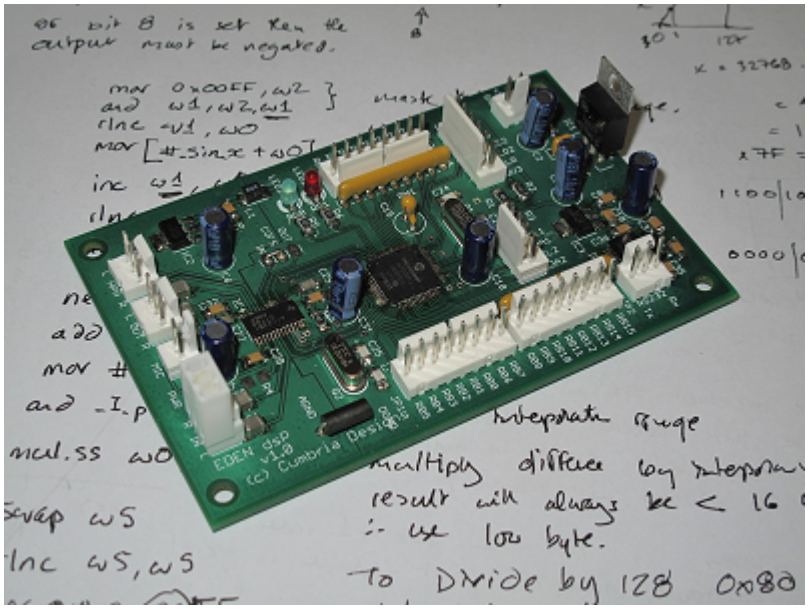

Figura 2-25.: Tarjeta EDEN dsP v1.0.

Son numerosas las investigaciones realizadas hasta ahora acerca de las aplicaciones de los dsPIC® en cuanto al procesamiento digital de audio, en el año 2010 fue publicado un trabajo de investigación de la Universidad Nacional de Mar del Plata (Argentina) denominado «Sistema de compresión de voz portátil basado en un dsPIC®» el cual involucra procesamiento de audio en dsPIC®. En ese artículo se propone un sistema de filtros para adecuar el espectro de los sonidos que perciben personas con sordera parcial, con el objetivo de mejorar su audición.

Existe una tesis que data del año 2011 denominada «Diseño y fabricación de un módulo didáctico basado en dsPIC® como herramienta de apoyo en el aprendizaje de sistemas de control digital con énfasis en la implementación de filtros digitales» por Yesid Erasmo Meneses, de la Universidad Pontificia Bolivariana (Colombia), en dicha tesis se creó un tarjeta con base en el dsPIC33FJ128GP802, con pines externos para conexión del programador PicKit2, aunque el DAC interno es usado en esta tarjeta, también se incorporó un DAC externo, según el autor, para aplicaciones de control que no son específicas de audio, el objetivo principal de esa investigación fue proponer una herramienta didáctica para el aprendizaje de la tecnología dsPIC®, los compiladores usados para programar esa tarjeta fueron mikroC y C30, para el diseño de los filtros digitales el software Digital Filter Design de Microchip fue utilizado.

3. Diseño de la tarjeta electrónica para procesamiento digital de audio con dsPIC®

En este capítulo se explica cómo fue el proceso de diseño del hardware que será utilizado para la implementación de los algoritmos de procesamiento digital de audio, empezando desde la selección de los componentes, continuando con los esquemáticos y finalizando con el diseño del circuito impreso. Es de destacar que el sistema maneja señales tanto digitales como analógicas y eso tendrá consecuencias en el diseño de las conexiones y la ubicación de los componentes para controlar los ruidos que puedan afectar a la señal.

3.1. Parámetros de selección de componentes electrónicos para el diseño de tarjetas de audio

En primer lugar, ha de decidirse el tipo de interfaz ADC y DAC que será utilizada. Para el presente caso, por simplicidad y conociendo que existen productos dsPIC® con conversores digital-analógico diseñados específicamente para aplicaciones de audio, será acotada la selección del procesador a los dispositivos que cumplen con esta característica. Aquí, los parámetros que entran en juego son las resoluciones de los conversores y el rango dinámico de los mismos.

Otro parámetro sumamente importante es la frecuencia de muestreo, debido a que el presente trabajo tiene por objetivo abarcar toda la banda de audio, será imprescindible que el sistema pueda procesar señales de hasta 20 kHz. En este caso, el teorema de Nyquist-Shannon sugiere que la mínima frecuencia de muestreo debe ser de 40 kHz. Esto tiene repercusiones no sólo en la velocidad de procesamiento sino también en el diseño del filtro antialiasing para el acondicionamiento de la señal de entrada.

Los sistemas embebidos en general operan a bajos niveles de voltaje, esto tiene consecuencias en las especificaciones de los amplificadores operacionales, los cuales deberían ser con tecnología Rail-to-Rail para aprovechar al máximo el rango de las señales. También repercute en la selección de los amplificadores de audio.

Los conectores de entrada y salida de la señal serán ajustados a conectores específicos de audio, que normalmente son de tipo plug y jack.

En este proyecto no se incluirá herramientas para depuración in circuit, por lo tanto no se hará necesario utilizar la comunicación JTAG. Entre las ventajas de no implementarlo están que es más difícil de hackear y difícil de hacer ingeniería inversa (Considerar que un sistema embebido completo tiene la opción JTAG). Entre las desventajas se tiene que el diseñador de aplicaciones debe corregir sus códigos obligatoriamente en software sin recibir información de hardware diciendo los puntos posibles de problema.

3.2. Parámetros para el diseño de circuitos impresos dedicados a aplicaciones de audio

Este tipo de circuitos pertenece a «circuitos con señales mezcladas», lo cual significa que hay dos tipos de señal en una sola tarjeta: señales analógicas y señales digitales. Esto potencia las posibles fuentes de ruido y para contrarrestarlo se hace necesaria la aplicación de técnicas de separación de señales, diseño apropiado de tierras, separación interna de la alimentación

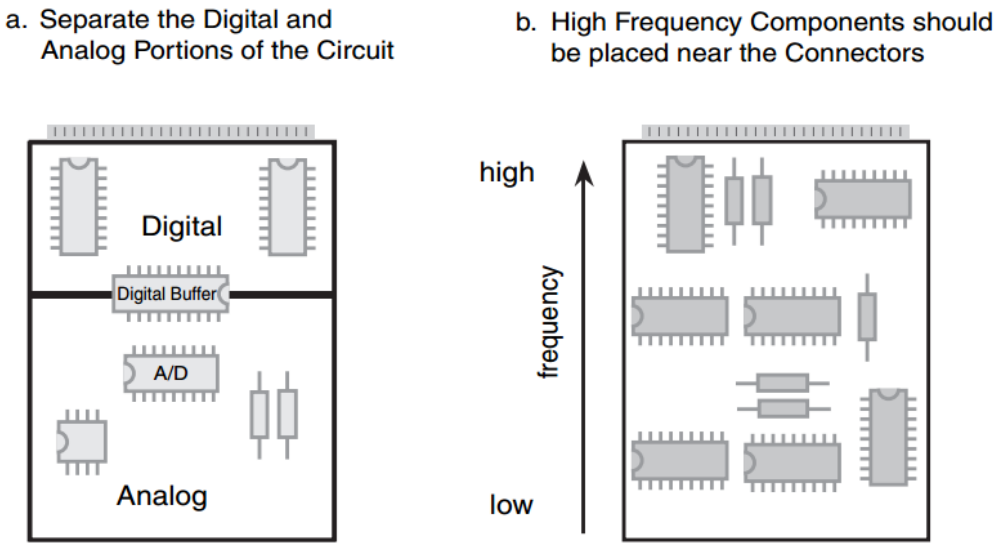

Figura 3-1.: Sugerencias para ubicación de los componentes en circuitos de audio [4, p. 267].

analógica de la digital, uso de inductores y otros desacoples. El diseño del circuito impreso juega un papel fundamental en la reducción de ruido por señales mezcladas.

Un mal diseño del circuito impreso hará que haya mala compatibilidad electromagnética entre la parte digital y la parte analógica del circuito, esto producirá ruidos en las señales. Para evitar los problemas de ruido se tendrá en cuenta unos criterios básicos de diseño de circuito impreso sugeridos en [4]: en primer lugar, los elementos digitales deberían quedar en una zona apartada de los elementos analógicos; en segundo lugar, los elementos que operan a altas frecuencias se deben ubicar tan cerca a los conectores como sea posible (ver Figura **3-1**).

Otro criterio importante es el uso de planos de tierra, el cual simplemente consiste en una porción de circuito impreso que proporciona un camino de retorno de baja impedancia para las corrientes del circuito, también teniendo en cuenta que los planos para tierra analógica y tierra digital deben estar físicamente separados aunque residan en la misma tarjeta, para que las corrientes de señales digitales no interactúen con las analógicas. Una sencilla manera de desacoplar estas señales es mediante el uso de un inductor donde ocurre la separación de las líneas de alimentación. Los dispositivos digitales son dispositivos que generan gran cantidad de ruido en las líneas de alimentación debido a los picos de corriente transitoria ocasionadas por las conmutaciones internas de los nivele lógicos. En la Figura **3-2** se tiene un típico desacople con topología estrella en la línea de alimentación.

Como el ruido no es un factor definitivo en el funcionamiento de la parte digital, se puede reducir el uso de las inductancias únicamente al acople de la parte analógica del circuito como se observa en la Figura **3-3**.

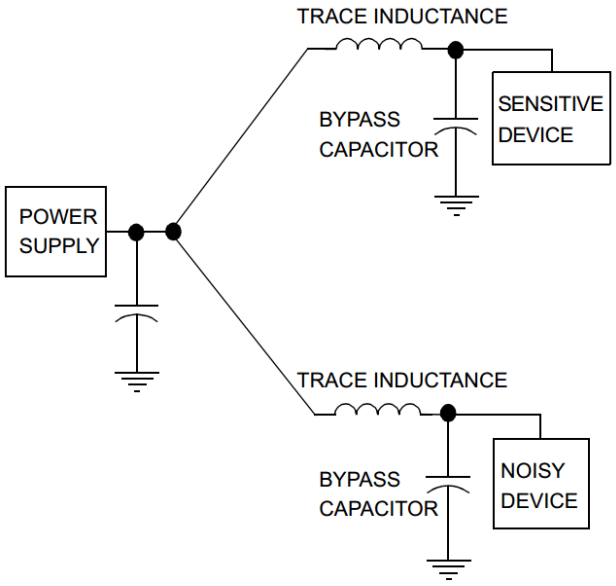

Figura 3-2.: Desacople mediante topología estrella [12, p. 5].

Otra recomendación para reducir el ruido es evitar la superposición de señales que deben estar separadas, como se muestra en la Figura **3-4**.

Como recomendaciones adicionales se tiene: no rutear líneas digitales cerca al plano de tierra analógico, mantener todas las señales analógicas encerradas dentro el plano de tierra analógico, utilizar condensadores de desacople en todos los pines de alimentación de dispositivos digitales.

3.3. Análisis de los requisitos de diseño para una nueva tarjeta

En primer lugar, se ha detectado que el primer obstáculo de una tarjeta de procesamiento digital de audio es su elevado precio, por lo tanto aquí podemos hacer un enfoque en materiales económicos pero sin disminuir en exceso las prestaciones.

Otro requisito fundamental es que en lo posible sea totalmente integrada, para minimizar el uso de herramientas externas a la tarjeta en sí, esto supone que por lo menos, el programador del dsPIC® debería estar integrado en la tarjeta diseñada.

El sistema debe contar con herramientas de desarrollo fácilmente adquiribles, actualizadas

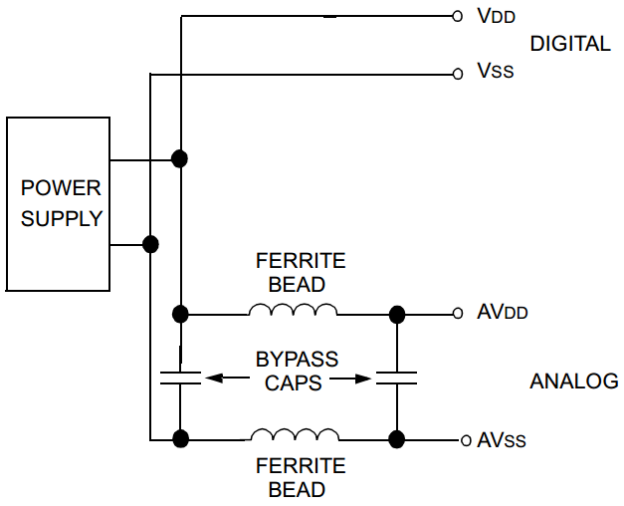

Figura 3-3.: Inductancias para acople de la parte analógica de un circuito de audio [12, p. 5].

Figura 3-4.: Superposición de planos analógicos y digitales en una PCB [14].

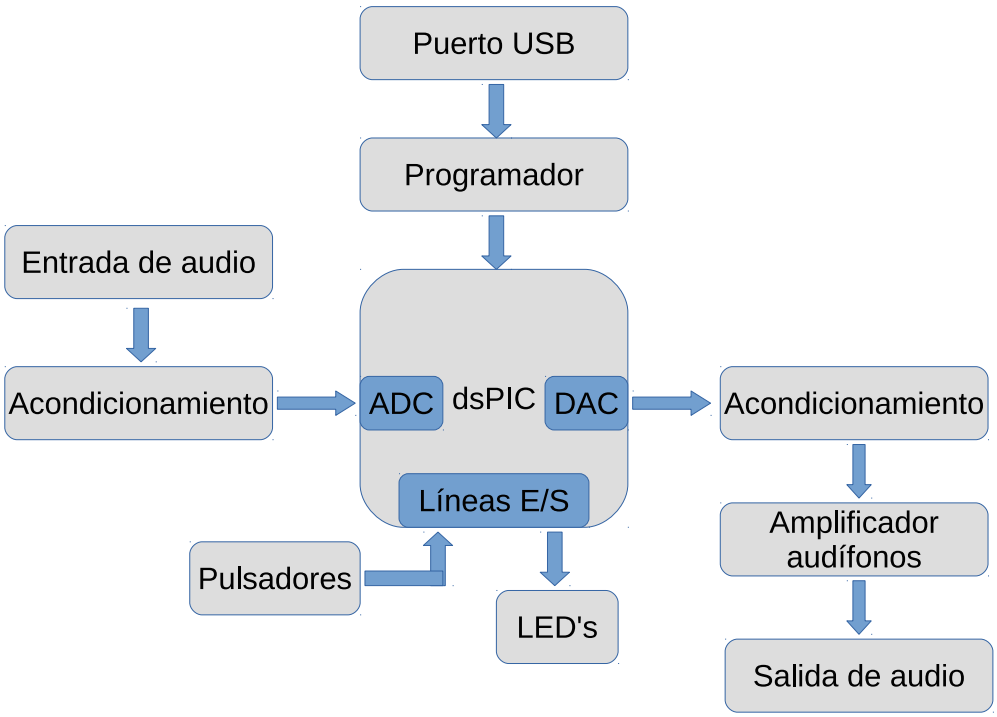

Figura 3-5.: Modelo propuesto para la tarjeta de procesamiento digital de audio.

y con herramientas que faciliten el desarrollo de forma notable. En la Figura **3-5** se puede apreciar el modelo propuesto.

3.4. Selección del dsPIC®

Un conjunto de características a tener en cuenta en la selección de un dispositivo DSP, en este caso específico dsPIC®, son los siguientes: formato de los datos, ancho de los datos, velocidad, organización de la memoria, arquitectura del procesador, consumo, coste, entorno de desarrollo.

Actualmente Microchip ofrece dos familias de Controladores Digitales de Señal, ambas con arquitectura de punto fijo de 16 bits:

- Familia dsPIC30F
- Familia dsPIC33F/E

Esta última consta de dos subfamilias que son la clásica dsPIC33F y la mejorada dsPIC33E. El formato y el ancho de los datos se ajustan a los estándares clásicos utilizados en el ámbito

del procesamiento digital de sonido, en donde la resolución suele ser de 16 bits para sonido monofónico. El entorno de desarrollo que ofrece Microchip es muy completo y documentado, incluyendo herramientas gratuitas y de prueba que aumentarán la productividad del desarrollo de aplicaciones con DSC.

La familia dsPIC30F fue la primera versión de lo que Microchip denominó Controladores Digitales de Señal (DSC). Posteriormente implementó la familia dsPIC33F/E para potenciar las capacidades y el rendimiento de los DSC en aplicaciones más complejas. En la tabla **3-1** se consignan las características más relevantes de estas dos familias.

Tabla 3-1.: Principales características de las familias de dsPIC®.

Recurso	**dsPIC30F**	**dsPIC33F/E**
Voltaje de alimentación	2.5 V a 5.5 V	Soporta $3V$ y $5V$
Velocidad de procesamiento	Hasta 30 MIPS	Hasta 70 MIPS
Memoria de programa	Hasta 144 KB	Hasta 512 KB
Memoria de datos	Hasta 8 KB	Hasta 30 KB
Número de pines	Desde 18 hasta 80	Desde 18 hasta 144
Mapeo de pines	No	Sí
ADC	12 bit, 200 ksps	12 bit, 500 ksps
DAC	No	Sí
Interfaz CODEC	Sí	Sí

Uno de los puntos más importantes a resaltar es el voltaje de alimentación, que directamente influye en el consumo de energía. Menos voltaje de operación significa menos potencia de consumo, lo cual favorece el ahorro de energía. Como se puede observar, la familia dsPIC33F/E ofrece mayores ventajas en muchos aspectos y se muestra como la más propicia para elegir un dispositivo que sirva como procesador digital de audio, la tendencia de las últimas tarjetas relacionadas con dsPIC® ya reflejan este hecho.

Con los anteriores argumentos para elegir la familia dsPIC33F/E y sin dejar de lado las velocidades de procesamiento que puedan ofrecer los diferentes dispositivos miembros de esta, se analizará principalmente los recursos de memoria, periféricos, patillaje y costos que tienen estos dispositivos para elegir uno que se ajuste a los requerimientos del proyecto. En los libros de diseño de productos con procesadores digitales de señal siempre se insiste en la importancia del costo de estos componentes, ya que la solución debe optimizar siempre este parámetro buscando que el procesador cumpla las exigencias requeridas con el menor costo posible lo cual no tiene nada que ver con elegir el dispositivo más potente.

Como punto de referencia en cuanto a memoria de programa se plantea analizar qué se ha usado en proyectos similares:

- La tarjeta de desarrollo dsPICDEM 1.1 consiste en una plataforma de desarrollo de propósito general que por supuesto incluye opciones de procesamiento de audio, aunque limitados a la banda de voz, su frecuencia de muestreo está determinada por el CODEC utilizado, el SI3000 de Silicon Labs, y va de 4 kHz a 12 kHz. En este caso el DSC utilizado es el dsPIC30F6014 el cual tiene una capacidad de memoria de programa de 144KB, y una memoria RAM de 8192 Bytes, su velocidad de procesamiento es de 30 MIPS.

- El MPLAB Starter Kit for dsPIC® DSCs en un conjunto de herramientoas hardware y software cuyo núcleo es un dsPIC® con capacidad de memoria de programa de 256KB y RAM de 16 KB. En esta tarjeta la frecuencia de muestreo también está dada por un CODEC de alta fidelidad fabricado por una empresa pionera en dispositivos para tratamiento digital de señales llamada actualmente Cirrus; este CODEC tiene la posibilidad de manejar hasta 48 KHz de muestreo, pero las librerías proporcionadas en el kit solo están escritas para implementar muestreos de 8 kHz o 16 kHz. El hardware también proporciona una forma alternativa de captura de audio a través de su conversor análogo digital interno y el método PWM como modulación de audio, esto para demostrar los sistemas de audio de bajo costo ya que se suprime el uso de un CODEC pero al mismo tiempo disminuye su fidelidad.

- El Audio Development Board for dsPIC33E es una tarjeta más ambiciosa que las anteriores, en el sentido que incluye más modernos periféricos y con más capacidades como puerto USB, LCD a color, entre otros. La memoria de programa de su dsPIC® es de 512 kB y su RAM de 52KB. Su capacidad de procesamiento es de 60 MIPS. Estas características repercuten inmediatamente en su precio.

- Otras tarjetas no se muestran muy relevantes ya que se destinan a diferentes propósitos que no colaboran a la facilidad de uso para que una persona inicie cómodamente en la implementación de algoritmos de procesamiento de audio.

Antes de continuar cabe anotar que el kit de desarrollo más barato encontrado ha sido el MPLAB® Starter Kit for dsPIC® DSCs, con un precio cercano a USD 60. Lo cual sin duda es económico pero sus prestaciones son en cierto modo deficientes en cuanto prácticamente solo permite que el audio entre, se procese y salga de nuevo sin dar lugar a visualizar datos en alguna pantalla o proporcionar puertos de control para ampliar las aplicaciones. Justamente esto es lo que se busca mejorar, ofreciendo un producto con dsPIC® que sea sencillo en su manejo pero que aun así permita obtener aplicaciones interesantes.

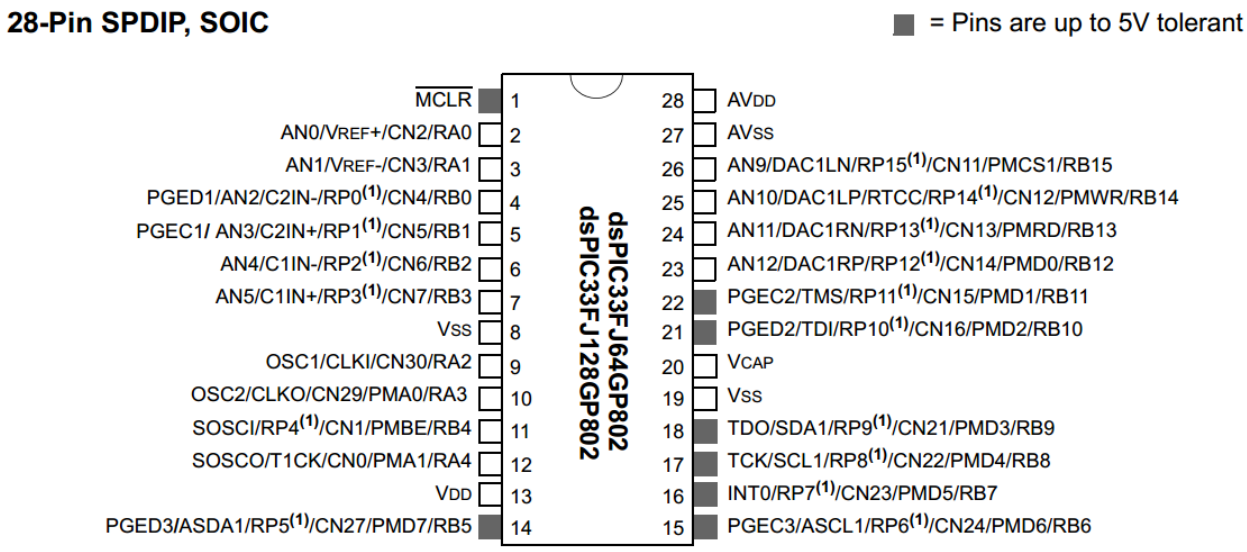

Figura 3-6.: Diagrama de pines del DSPIC33FJ128GP802 [23, p. 3].

Entre los dsPIC® ofrecidos en el mercado, debido a su versatilidad, bajo precio, facilidad de implementación y la incorporación de un DAC para aplicaciones de audio, se mostró más favorable el dsPIC33FJ128GP802, el cual ofrece, entre otras, las características de la Tabla **3-2**.

Tabla 3-2.: Principales características del DSPIC33FJ128GP802.

Parámetro	**Valor**
Arquitectura	16 bits
Velocidad de procesamiento	40 MIPS
Memoria de programa	128 kB
Memoria RAM	16284 Bytes
Voltaje de operación	$3V \sim 3{,}6V$
Pines de entrada salida	21
Acceso directo a memoria (DMA)	8 canales
Conversor analógico a digital	12 bits, 500 ksps
Conversor digital a analógico	16 bits, 100 ksps
Mapeo de pines	16 pines remapeables
Cantidad de instrucciones base	83

El diagrama de pines y las conexiones eléctricas mínimas se muestran en la Figura **3-6** y la Figura **3-7** respectivamente.
Con base en la Figura **3-7** se debe tener en cuenta las siguientes recomendaciones:

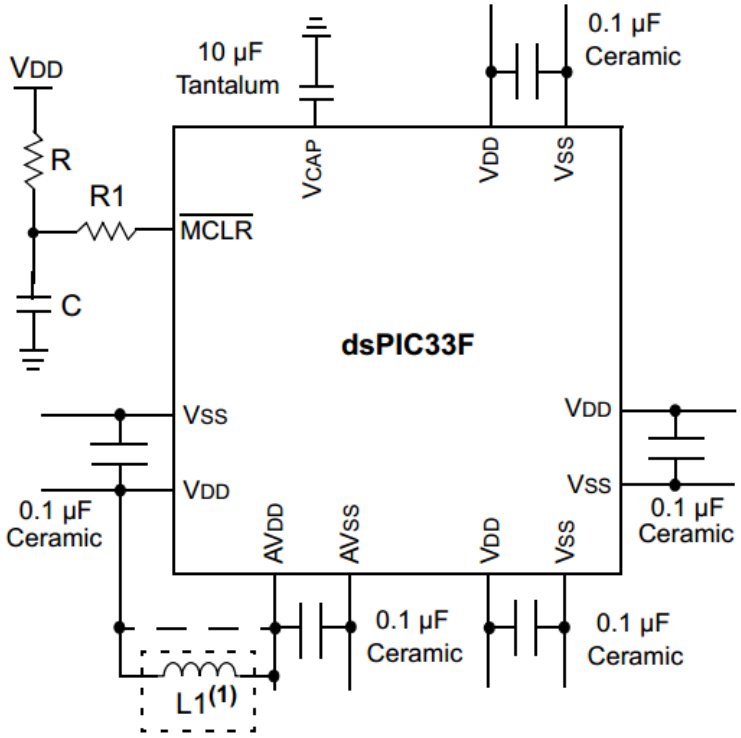

Figura 3-7.: Conexiones eléctricas mínimas del DSPIC33FJ128GP802 [23, p. 20].

- Todos los pines VDD y VSS deben ser conectados a la fuente de alimentación.
- Todos los pines AVDD y AVSS deben ser conectados aunque no se use el conversor análogo-digital.
- Todos los pines de alimentación deben tener un capacitor cerámico para el desacople con valor de 100 nF.
- VCAP debe conectarse siempre a un capacitor con baja resistencia en serie equivalente, usualmente este capacitor es de tantalio.
- MCLR debe tener un resistor hacia VDD
- PGECx y PGEDx deben conectarse para la programación ICPS
- Pines OSC1 y OSC2 se conectan cuando se usa oscilador externo
- De forma opcional, un inductor puede ser conectado entre VDD y AVDD para mejorar el rechazo al ruido en las señales analógicas.

El diagrama esquemático de la conexión eléctrica del dsPIC® se muestra en la Figura **3-8**.

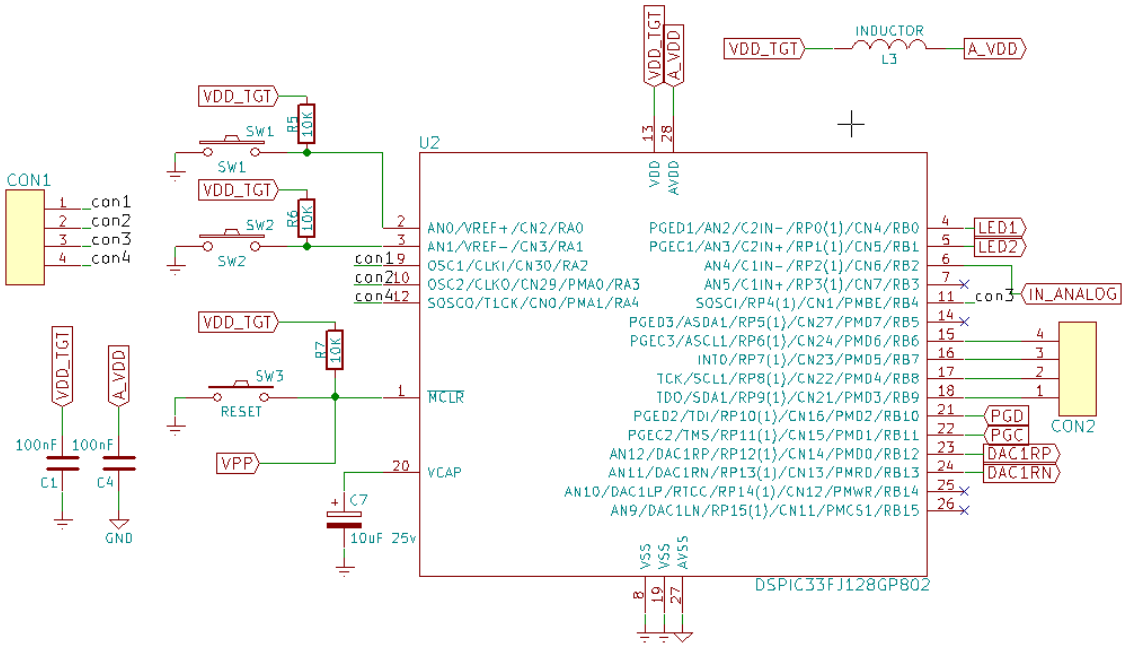

Figura 3-8.: Esquemático del DSPIC33FJ128GP802.

3.5. Interfaz de programación

El programador a ser incorporado será un clon del conocido programador pickit2, las modificaciones realizadas buscan la simplicidad del sistema. Únicamente se dejó la mínima cantidad de líneas necesarias para la programación y se conservó el LED que indica el estado ocupado del microcontrolador. La fuente de alimentación es externa, por eso aquí sólo se muestra la respectiva entrada de 3.3 voltios. El diagrama esquemático del circuito programador se ha dividido en 2 partes: la parte del control (Figura **3-9**) y la parte de generación de voltaje de programación VPP (Figura **3-10**). El inductor utilizado en el circuito elevador de voltaje tiene un valor de 680 uH.

3.6. Acondicionamiento de la señal de audio

Para la entrada de señal se ha usado como base un circuito pasabajas de segundo orden, el cual se muestra en la Figura **3-11**.

Para la salida de audio, se ha usado la configuración de amplificador operacional para señales balanceadas sugerido en la sección 33 del manual de referencia de la familia dsPIC33F, y se ha añadido un amplificador de audio a la salida para poder enviar la señal a unos audífonos como se observa en la Figura **3-12**.

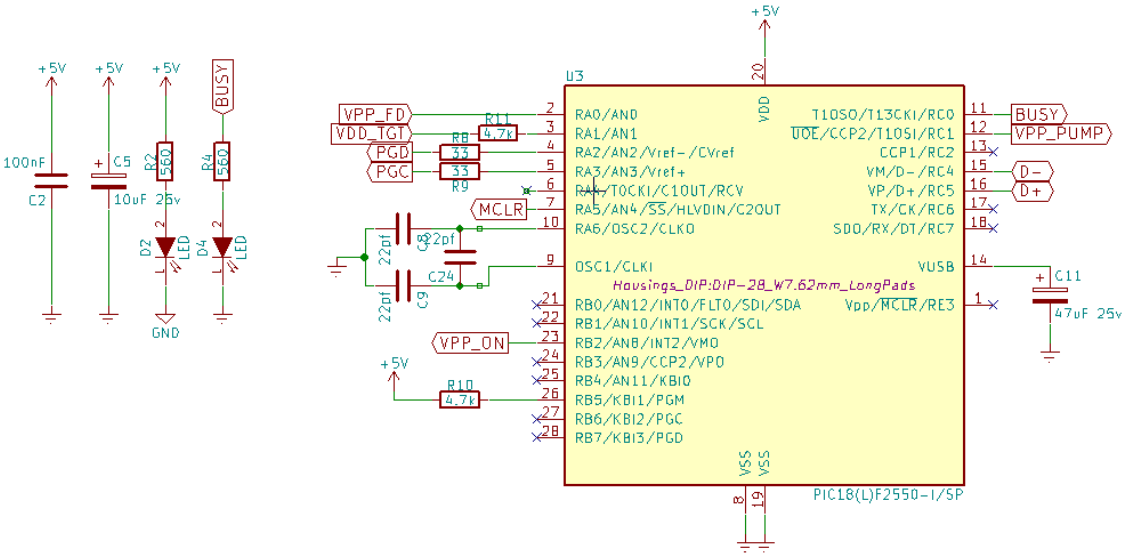

Figura 3-9.: Circuito de control del programador para el DSPIC33FJ128GP802.

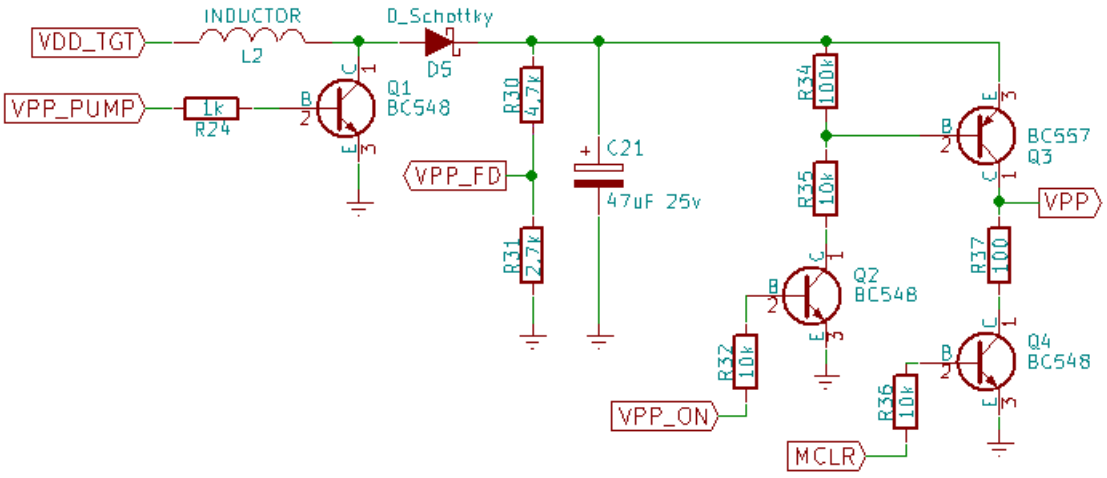

Figura 3-10.: Circuito de elevación de voltaje del programador para el DSPIC33FJ128GP802.

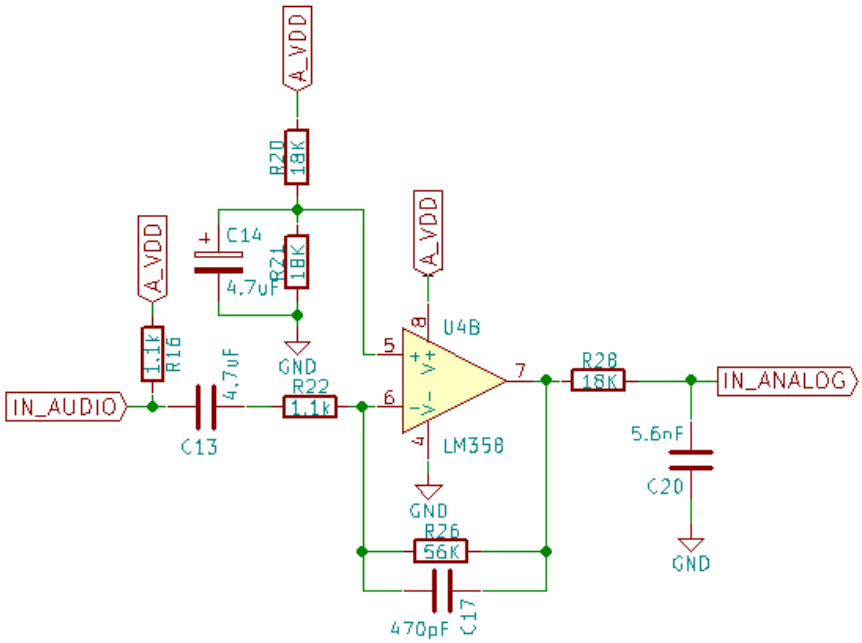

Figura 3-11.: Filtro antialiasing para la entrada analógica del DSC.

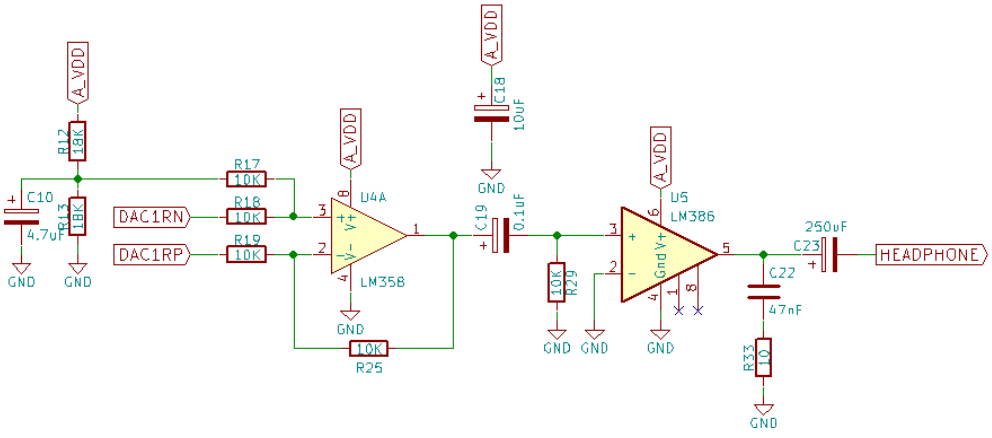

Figura 3-12.: Circuito de salida de audio.

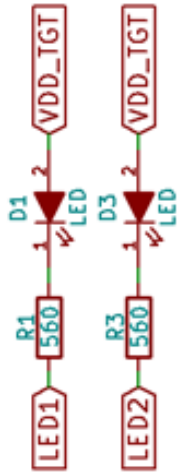

Figura 3-13.: Indicadores LED.

3.7. Componentes de interfaz con el usuario

El circuito tiene interfaces para conectarse con un ordenador a través del puerto USB. Cuenta también con regletas hacia pines desconectados para permitir la conexión de módulos adicionales para propósito general.

El circuito cuenta con pulsador para resetear y con dos pulsadores auxiliares para propósito general. Además de eso, se incluyen 2 visualizadores LED (Figura **3-13**) para propósito general, configurados en modo sumidero, esto significa que se activan con nivel lógico bajo.

Por último, el circuito acepta que el usuario conecte cualquier línea de audio proveniente de equipos tales como computadores, celulares, DVD, y demás dispositivos con salida de línea para audio. Así mismo, el circuito cuenta con driver para conectar audífonos directamente. El conector empleado tanto para la entrada como para la salida de audio es de tipo jack 3.5 mm, su esquemático está ilustrado en la Figura **3-14**.

3.8. Fuente de alimentación

Los reguladores lineales producen menos ruido que los conmutados, por eso será la primera consideración para el diseño de la fuente de alimentación.

Un correcto diseño de la alimentación es crucial para el desempeño general del sistema y especialmente para el dsPIC®. En este tipo de sistemas siempre se usa alimentación con fuentes reguladas, se asume que la entrada de voltaje es 5 voltios ya que se piensa utilizar el

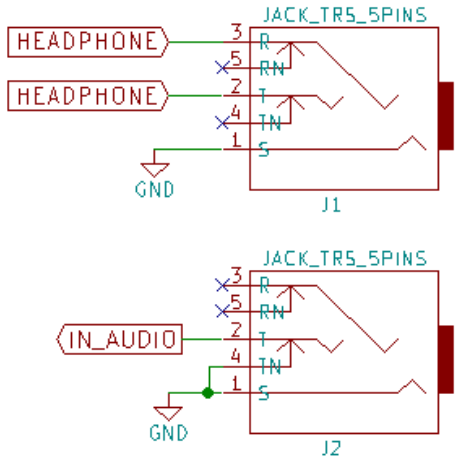

Figura 3-14.: Conectores de audio 3.5 mm.

puerto USB estándar de los computadores como puerto de programación y para optimizar conexiones se usará también como puerto de suministro de energía. El puerto USB, según su versión, ofrece distintas cantidades de corriente que deben ser tenidas en cuenta para establecer los parámetros de consumo del circuito que será conectado.

Un cable USB estándar se compone de cuatro hilos de los cuales dos son los destinados al suministro de energía: VBUS para 5 voltios y GND para tierra. En la Tabla **3-3** se evalúan las características básicas de este puerto en sus distintas versiones.

Tabla 3-3.: Características de corriente del puerto USB.

Puerto	Corriente máxima
USB 1.0	-
USB 1.1	-
USB 2.0	500
USB 3.0	900

El dsPIC® funciona con 3.3 voltios, por lo que se hace necesario implementar un circuito que reduzca los 5 voltios que proporciona el puerto USB. El primer parámetro que vemos implicado es que la caída de tensión en el regulador, en las hojas de datos de estos dispositivos este parámetro suele llamarse «dropout voltage», el cual está determinado por la siguiente expresión:

$$v_{dropout} = \quad v_{in} - v_{out}$$

Symbol	Parameter		Value	Unit
V_{IN}	DC Input Voltage		15	V
P_{TOT}	Power Dissipation		12	W
T_{STG}	Storage Temperature Range		-40 to +150	°C
T_{OP}	Operating Junction Temperature Range	for C Version	-40 to +150	°C
		for standard Version	0 to +150	°C

Figura 3-15.: Parámetros máximos de operación del LD1117 [29].

$$v_{dropout} = 5v - 3{,}3v$$
$$v_{dropout} = 1{,}7 \quad v$$

Por lo tanto, ha de seleccionarse un regulador con un voltaje dropout menor a 1.7 voltios.

Otro parámetro a tener en cuenta es la corriente que manejará, se ha decidido que el regulador de voltaje sea totalmente consistente con la corriente que puede entregar el puerto USB, pero al existir diferentes versiones con diferentes capacidades de corriente se ha optado por elegir la más común actualmente que es el estándar USB 2.0; aunque la tendencia es cada vez mayor hacia el estándar USB 3.0 no habrá inconveniente debido a que este último es totalmente compatible para facilitar la migración hacia esa tecnología. En este sentido, la corriente estará limitada a 500 mA.

Por simplicidad, economía y facilidad de adquisición se ha decidido trabajar con el circuito integrado LD1117. Con base en la hoja de datos del fabricante (Figura **3-15** y Figura **3-16**) se obtiene el esquemático de la Figura **3-17**.

3.9. Resultados

Con los parámetros mencionados en la sección 3.2. se diseñó el prototipo de circuito impreso usando el software kiCAD. El circuito se resume así:

- Circuito impreso con tecnología Through and Hole.
- Dimensiones: 121.9 mm x 97.2 mm
- Una cara de cobre.
- 20 puentes de alambre
- 24 capacitores

Symbol	Parameter	Test	Min.	Typ.	Max.	Unit
V_O	Output voltage	V_{in} = 3.2 V, I_O = 10 mA, T_J = 25°C	1.188	1.20	1.212	V
V_O	Reference voltage	I_O = 10 to 800 mA V_{in} - V_O= 1.4 to 10 V	1.140	1.20	1.260	V
ΔV_O	Line regulation	V_{in} - V_O = 1.5 to 13.75 V, I_O = 10 mA		0.035	0.2	%
ΔV_O	Load regulation	V_{in} - V_O = 3 V, I_O = 10 to 800 mA		0.1	0.4	%
ΔV_O	Temperature stability			0.5		%
ΔV_O	Long term stability	1000 hrs, T_J = 125°C		0.3		%
V_{in}	Operating input voltage				15	V
I_{adj}	Adjustment pin current	V_{in} ≤15 V		60	120	µA
ΔI_{adj}	Adjustment pin current change	V_{in} - V_O = 1.4 to 10 V I_O = 10 to 800 mA		1	5	µA
$I_{O(min)}$	Minimum load current	V_{in} = 15 V		2	5	mA
I_O	Output current	V_{in} - V_O = 5 V, T_J = 25°C	800	950	1300	mA
eN	Output noise (%V_O)	B =10Hz to 10KHz, T_J = 25°C		0.003		%
SVR	Supply voltage rejection	I_O = 40 mA, f = 120Hz, T_J = 25°C V_{in} - V_O = 3 V, V_{ripple} = 1 V_{PP}	60	75		dB
V_d	Dropout voltage	I_O = 100 mA		1	1.1	V
		I_O = 500 mA		1.05	1.15	
		I_O = 800 mA		1.10	1.2	
	Thermal regulation	T_a = 25°C, 30ms Pulse		0.01	0.1	%/W

Figura 3-16.: Características eléctricas del LD1117 [29].

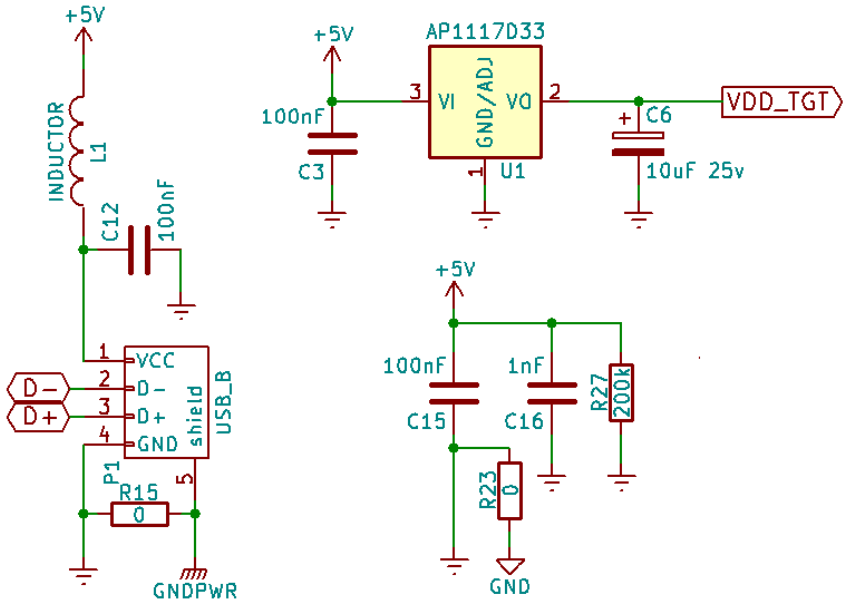

Figura 3-17.: Circuito de alimentación.

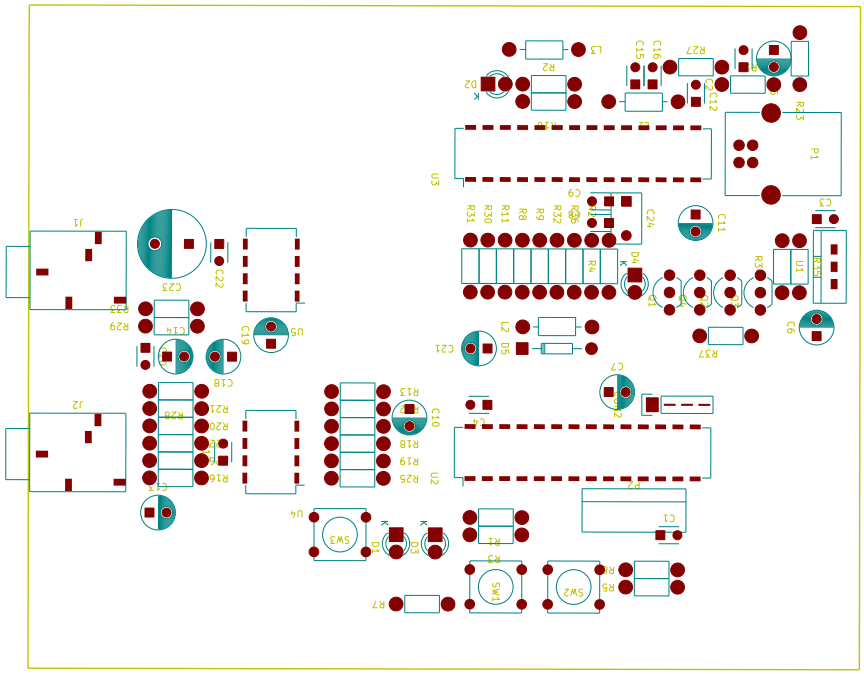

Figura 3-18.: Vista superior de la ubicación de los componentes de la PCB.

- 4 diodos LED
- 2 conectores jack de 3.5 mm
- 3 inductores
- 1 conector USB tipo B
- 4 transistores
- 3 interruptores tipo pulsador
- 1 regulador de tensión
- 4 circuitos integrados, incluidos el DSC y el microcontrolador de programación
- 37 resistores
- 2 conectores tipo regleta; uno con 6 pines y el otro con 4 pines

La Figura **3-18** muestra la disposición de los componentes a lo largo de la PCB; la **3-19** es la vista superior del ruteo, este ha sido llevado a cabo de forma manual.

En la Figura **3-20** se tiene una versión 3D del circuito generado y en la Figura **3-21** una fotografía del prototipo real.

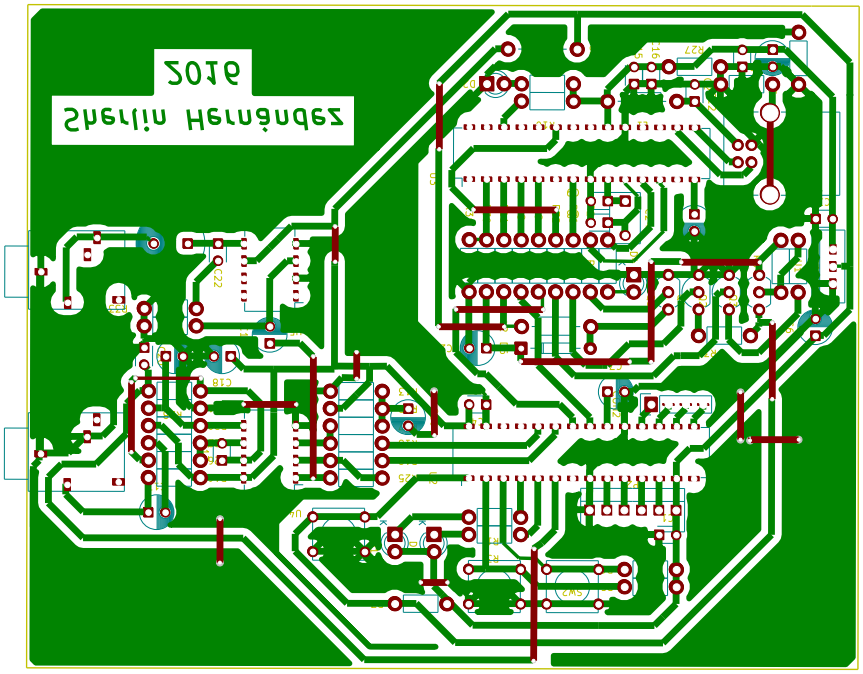

Figura 3-19.: Vista superior del ruteo de la PCB.

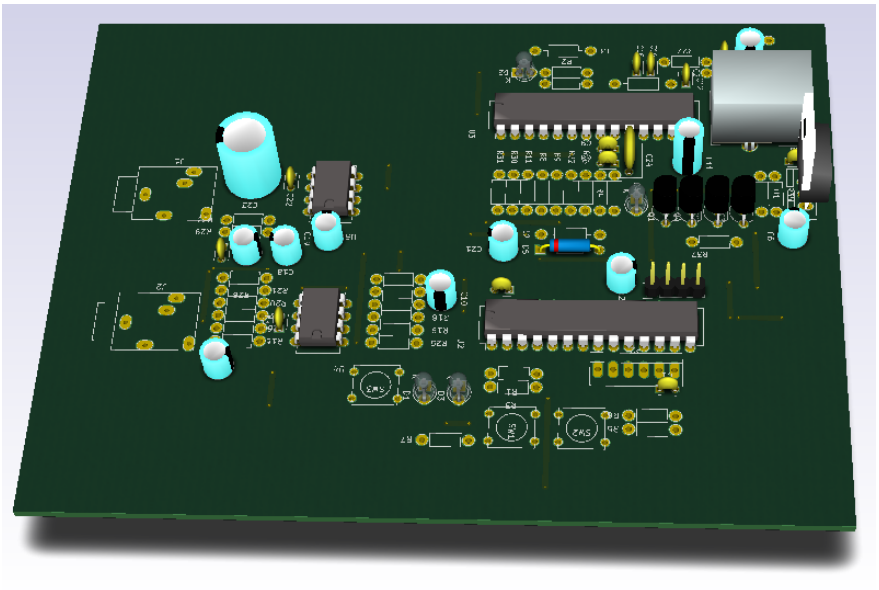

Figura 3-20.: Versión 3D del circuito.

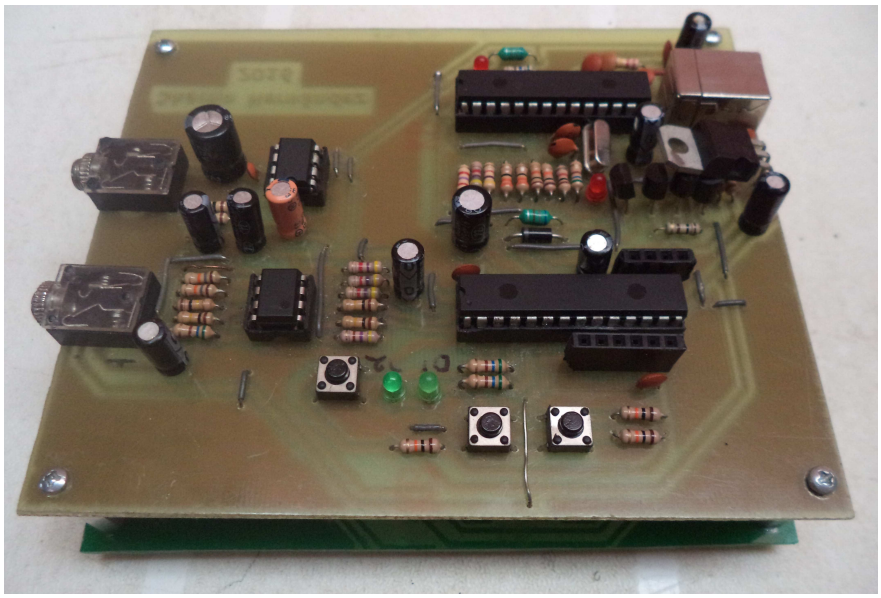

Figura 3-21.: Fotografía del prototipo final.

4. Procesamiento digital de audio con operaciones en el dominio temporal

Para la implementación de prácticas primero se hará un recorrido por el proceso de desarrollo de código que será grabado al dsPIC®, la metodología de programación está basada principalmente en el lenguaje C y para la parte de procesamiento de las señales serán usadas funciones escritas en lenguaje ensamblador, este enfoque pretende implementar programas modulares que permitan que las personas que lean el código puedan desarrollar aplicaciones rápida y eficazmente mediante el tratamiento del código como bloques con funciones específicas. El software para el desarrollo de estos programas es el MPLAB® X en su versión 3.30 y el compilador es el XC16 en su versión 1.26.

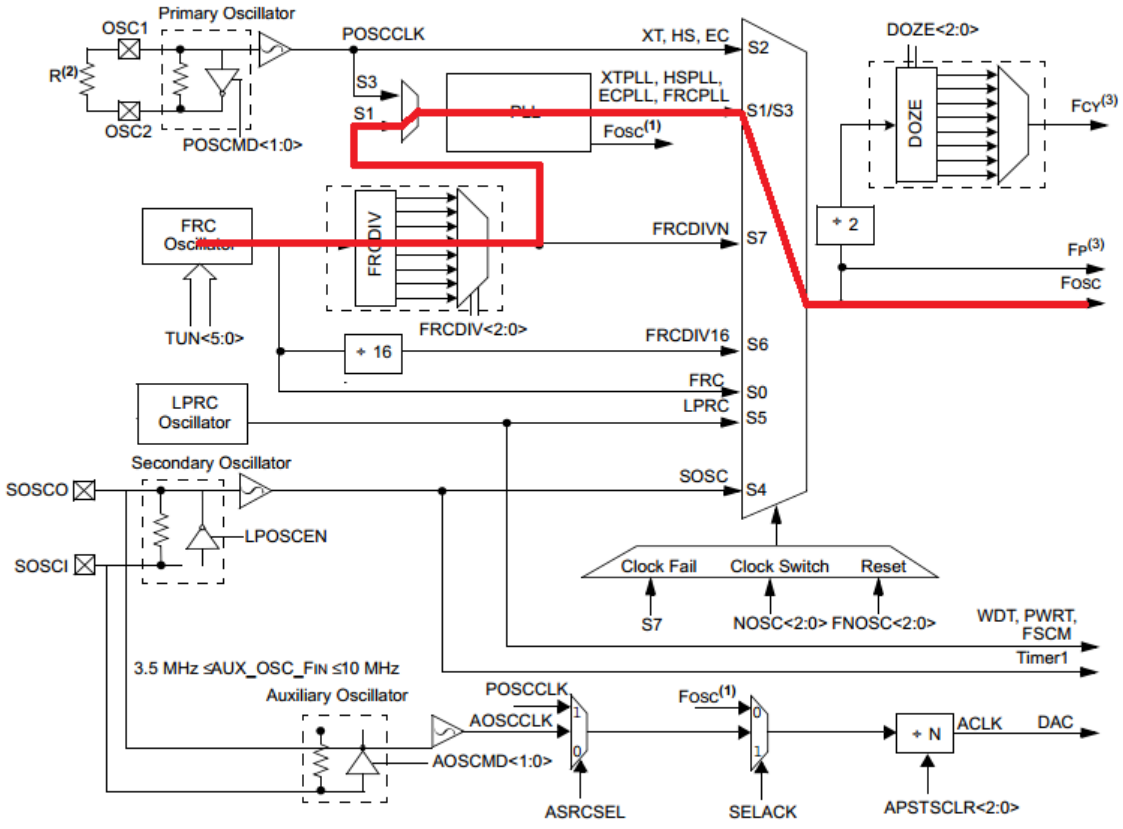

Figura 4-1.: Sistema del oscilador y camino de configuración propuesto (Adaptado de [28]).

4.1. Configuración del dsPIC®

4.1.1. Configuración del oscilador

El oscilador del dsPIC33FJ128GP802 tiene muchas características que no son comunes en tecnologías anteriores, entre ellas, un divisor de realimentación en el PLL que permite obtener una gran cantidad de frecuencias diferentes mediante este módulo. Otra característica es la posibilidad de usar un oscilador RC interno de 7,3728 MHz que permite ser configurado mediante PLL para trabajar con la máxima velocidad de reloj permitida por el procesador, la cual es 80 MHz. El uso del oscilador interno minimiza la cantidad de componentes externos y deja a disposición más líneas de entrada/salida, la principal desventaja está en la menos estabilidad y menos exactitud de frecuencia respecto a otros tipos de osciladores como los basados en cristales de cuarzo. En la Figura **4-1** del sistema oscilador del dsPIC33F mostrando el camino que debe ser tomado para usar el oscilador interno a la máxima frecuencia sugerida.

Se puede observar que en primer lugar tenemos un registro llamado $TUN < 5 : 0 >$, el cual permite desviar mediante software el valor nominal de la frecuencia del oscilador en un rango de $\pm 12\,\%$, con el objetivo de ajustar a un valor apropiado. Su valor por defecto es 0 el cual

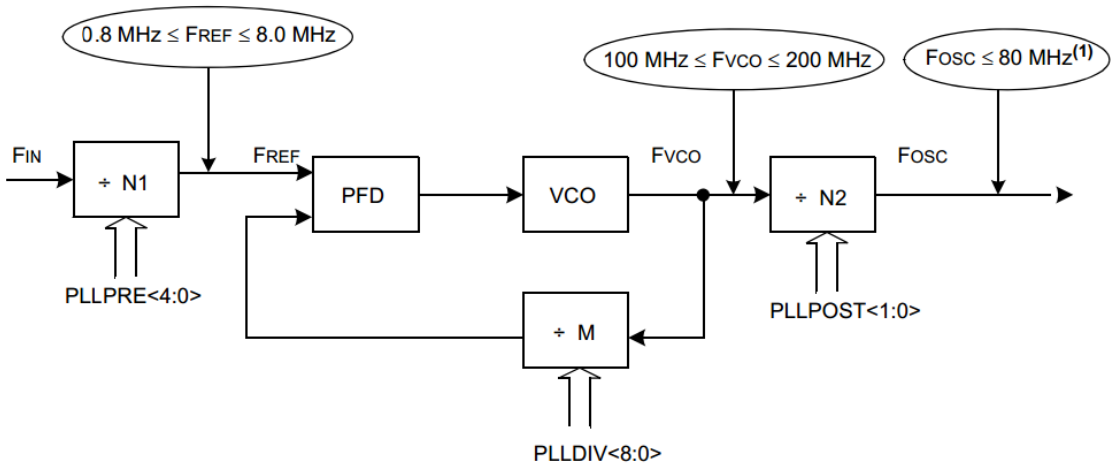

Figura 4-2.: Esquema del circuito PLL [28].

significa que trabaja con la frecuencia nominal de 7,3728 MHz.

Luego se llega a un divisor multiplexado por el registro $FRCDIV < 2 : 0 >$, por defecto dicho registro está configurado para oscilador interno sin división y no será modificado.

Después se tiene el multiplexor que selecciona la entrada al circuito PLL, esta configuración no está en los registros de configuración especiales sino en la memoria de programa, por lo tanto debe ser especificado en los fusibles o palabra de configuración.

El circuito PLL requiere especial cuidado en su configuración, para entender mejor su funcionamiento la Figura **4-2** resume los registros que lo configuran y los parámetros de frecuencia en cada etapa del circuito.

Vemos que el sistema PLL está controlado por tres registros divisores, el registro $PLLDIV < 8 : 0 >$ al estar en modo realimentación se comporta en realidad como un multiplicador de la frecuencia. Los registros $PLLPRE < 4 : 0 >$ y $PLLPOST < 1 : 0 >$ por defecto tienen un valor de cero y según su hoja de características, esto equivale a que cada uno divide entre 2 la frecuencia. El valor de M se calcula de forma sencilla mediante la expresión:

$$M = PLLDIV + 2$$

Asumiendo $N1 = 2$ y $N2 = 2$, se puede calcular M y PLLDIV para que el dsPIC® trabaje a la máxima velocidad. Así el valor propicio para esto es: $PLLDIV = 41$, por lo tanto $M = 43$. La siguiente ecuación muestra el valor de la frecuencia si se aplica esta configuración:

$$F_{osc} = F_{in}\left(\frac{M}{N1 \times N2}\right)$$
$$F_{osc} = 7{,}3728MHz\left(\frac{43}{2 \times 2}\right)$$
$$F_{osc} = 79{,}2576MHz$$

Esa será la frecuencia nominal de trabajo que se programará en el dsPIC®, como se puede apreciar, es un valor un poco menor a 80 MHz que es la frecuencia máxima sugerida por el fabricante. La velocidad de procesamiento quedaría así:

$$F_{CY} = \frac{F_{osc}}{2}$$
$$F_{CY} = \frac{79{,}2576 \quad MHz}{2}$$
$$F_{CY} = 39{,}6288 \quad MHz$$

El último paso es seleccionar la frecuencia de salida del PLL como frecuencia que utilizará el dispositivo, esto se hace desde los fusibles de configuración.

Debido a que el dsPIC® no inicia su operación con el oscilador FRC en modo PLL, es necesario hacer una conmutación para cambio de reloj a través de software. Existen unas funciones especiales en C que permiten hacer este cambio. La función que configura el oscilador queda así:

```
void ConfigOscilador (void)
{
    OSCTUNbits.TUN      = 0;  // Sintoniza FRC a la frecuencia nominal
    CLKDIVbits.FRCDIV   = 0;  // Divide entre 1 la frecuencia de FRC
    CLKDIVbits.PLLPOST  = 0;  // N1 = 2
    CLKDIVbits.PLLPRE   = 0;  // N2 = 2
    PLLFBDbits.PLLDIV   = 41; // M = 43 (M = PLLFBD + 2)

    // Cambio de reloj para incorporar PLL
    __builtin_write_OSCCONH(0x01);          // Inicia el cambio de reloj a
       FRCPLL    (NOSC=0b001)
    __builtin_write_OSCCONL(0x01);          // Inicia el cambio de reloj
    while(OSCCONbits.COSC != 0b001);    // Espera a que ocurra cambio de reloj
    while(OSCCONbits.LOCK != 1);        // Espera a que el PLL se bloquee
}
```

4.1.2. Configuración de los puertos

En cuanto a la configuración de los puertos, o líneas de E/S, los registros encargados de definir cada pin como entrada o salida son los registros TRISx, en la respectiva hoja de especificaciones está el detalle de los bits correspondientes. Si un bit del registro TRISx está definido como «0» su correspondiente pin asociado se comportará como salida; si dicho bit está definido como «1» entonces ese pin será una entrada. Otro registro muy importante es el AD1PCFGL, el cual define si el pin correspondiente es analógico o digital, un bit igual «0» define al correspondiente pin como analógico y si el bit tiene un valor de «1» su correspondiente pin será digital. En dispositivos con más de 16 entradas analógicas existe otro registro llamado AD1PCFGH para permitir la incorporación y manejo de más pines.

Después de un reset, todos los pines quedan definidos como entradas analógicas. O sea, por defecto todos los bits de los registros TRISx serán «1» y los del registro AD1PCFGL serán «0».

Antes de continuar, se definirán los puertos con un archivo de cabecera para tener un manejo de código más sencillo, los LED?s serán conectados a RB0 y RB1, los pulsadores a RA0 y RA1. Debido a la configuración en modo sumidero de los LEDs se indicará que un «0» encenderá el LED y un «1» lo apagará. Por último, se define también que un «0» significa salida o pin analógico y un «1» significa entrada o pin digital según el registro al que se aplique, el archivo queda de la siguiente manera:

```
// Macros para los puertos
#define LED1      _LATB0
#define LED2      _LATB1

#define BOTON1    _RA0
#define BOTON2    _RA1

#define ENCENDER          0
#define APAGAR        1

#define SALIDA        0
#define ENTRADA       1

#define ANALOGICO         0
#define DIGITAL       1
```

La siguiente función configura los puertos para la tarjeta de prácticas diseñada, vemos que el único pin analógico es AN4, el cual corresponde a la entrada de la señal de audio. Se definen los pines para los pulsadores y los LED, esto últimos se configuran para que inicien apagados:

```
void ConfigPuertos(void)
{
    AD1PCFGL           = 0xFFFF;     // Todos los pines son digitales
    AD1PCFGLbits.PCFG4 = ANALOGICO; // AN4 es un pin analógico

    _TRISA0 = ENTRADA;   // Botón 1
    _TRISA1 = ENTRADA;   // Botón 2

    _TRISB0 = SALIDA;    // LED1
    _TRISB1 = SALIDA;    // LED2

    _TRISB10 = ENTRADA; // PGD
    _TRISB11 = ENTRADA; // PGC

    LED1 = APAGAR;
    LED2 = APAGAR;
}
```

4.1.3. Configuración de ADC y DMA

El convertidor analógico-digital presente en el dsPIC33FJ128GP802 tiene una resolución configurable de 10 bits o 12 bits, con velocidades de muestreo máximas de 1.1 Msps y 500 ksps respectivamente. El búfer de este conversor solamente contiene un espacio de una palabra pero debido a la posibilidad de Acceso Directo a Memoria (DMA) puede ampliarse sin intervención de la CPU hacia el espacio DMA RAM.

La entrada de este conversor puede ser conectada a 4 canales diferentes de muestreo y retención, el canal CH0 será usado para adquirir la señal de audio ya que es el único canal que admite el modo 12 bits. Este canal se selecciona con la siguiente instrucción:

```
        AD1CON2bits.CHPS = 0;
```

Los voltajes de referencia se configurarán internamente a AVDD y AVSS. En la Figura **4-3** se muestra el esquema que se quiere configurar, en rojo está la señal que se va a convertir y en azul las señales de referencia, en dicho esquema se han suprimido los otros tres canales por simplicidad.
Hay que configurar los dos multiplexores que controlan la entrada hacia el amplificador diferencial, primero seleccionamos AN4 como entrada de señal hacia la línea no inversora con la siguiente instrucción:

```
        AD1CHS0bits.CH0SA = 4;
```

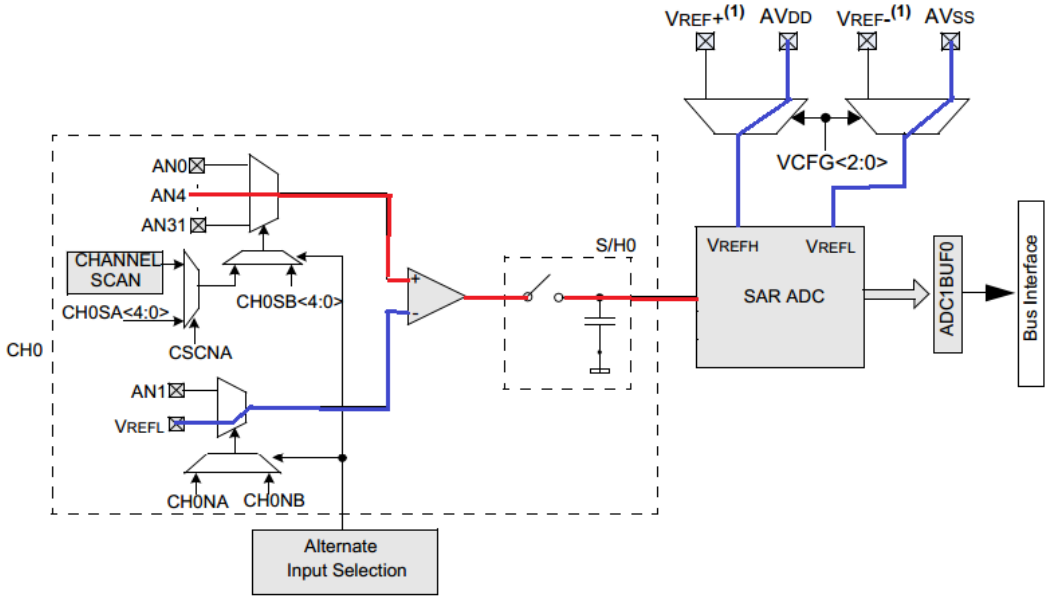

Figura 4-3.: Esquema del ADC (Adaptado de [24]).

Como no es de interés utilizar el diferencial de la señal, la entrada inversora se dejará conectada a VREFL con la siguiente instrucción:

```
AD1CHS0bits.CH0NA = 0;
```

Con el registro VCFG seleccionamos los voltajes de referencia, en este caso AVDD y AVSS así:

```
AD1CON2bits.VCFG = 0;
```

De esta forma quedan configurados los multiplexores. Con la siguiente instrucción se selecciona el modo 12 bits:

```
AD1CON1bits.AD12B = 1;
```

Otro aspecto importante a configurar es el formato de conversión, de los cuatro formatos posibles el que interesa es el fraccional con signo ya que además de dar signo a la conversión también hace un escalamiento automático hacia 16 bits, llenando con ceros los 4 bits menos significativos. Este formato será muy importante porque es en el que se hace el procesamiento de señales en el dsPIC®, la siguiente instrucción configura este formato que también es llamado Q15:

```
AD1CON1bits.FORM = 3;
```

Signed Fractional (1.15)	$\overline{d11}$	d10	d09	d08	d07	d04	d03	d02	d01	d00	d01	d00	0	0	0	0

Figura 4-4.: Formato de conversión de 12 bits fraccional con signo [24, p. 74].

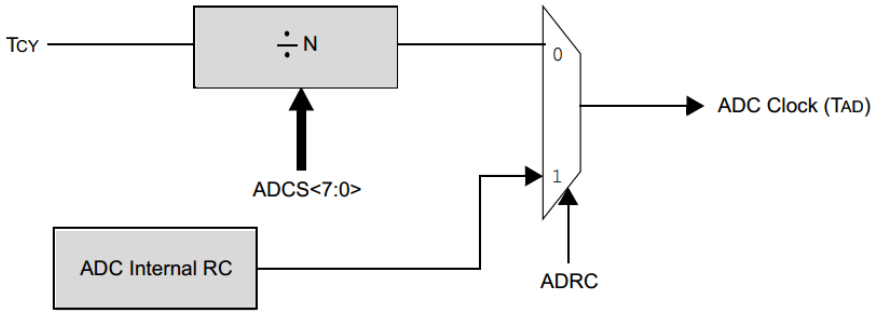

Figura 4-5.: Selección del reloj el ADC [24, p. 28].

Así, el formato en que se llenará el búfer del ADC se ilustra en la Figura **4-4**.

El muestreo será configurado para ser automático, de modo que el ADC estará muestreando siempre que no esté realizando una conversión en ese momento. La instrucción para configurarlo es la siguiente:

```
AD1CON1bits.ASAM = 1;
```

Se indicará al dsPIC® que el momento en que se empieza una conversión será manual, esto se especifica a través de la instrucción:

```
AD1CON1bits.SSRC = 0;
```

De esta manera, una conversión será realizada cada vez que se escriba un cero en el bit sample.
El ADC tiene dos posibilidades de selección de reloj: derivado del sistema o su reloj interno. Este último, según la hoja de características, tiene un período típico de 250 ns. Del otro reloj se puede obtener un rango de frecuencias mediante el registro $ADCS < 7 : 0 >$ como se muestra en la Figura **4-5**.

El TAD es un parámetro que se debe respetar, para el ADC de este dsPIC® debe ser de mínimo 117.6 ns. Esto lo cumple el reloj interno del ADC pero si se desea usar el reloj del sistema entonces debe ajustarse mediante el divisor controlado por $ADCS < 7 : 0 >$. La fórmula del TAD está definida a continuación:

$$T_{AD} = \frac{ADCS + 1}{F_{CY}}$$

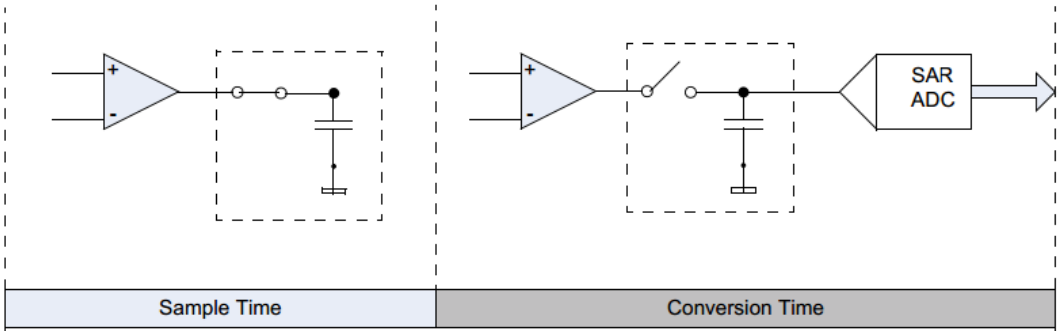

Figura 4-6.: Etapas de muestreo y retención en el ADC [24, p. 16].

Con lo cual podemos calcular un valor mínimo de ADCS para una frecuencia del sistema dada.

$$ADCS_{min} = (T_{ADmin})\,(F_{CY}) - 1$$
$$ADCS_{min} = \left(117{,}6 \times 10^{-9}\right)\left(39{,}6288 \times 10^{6}\right) - 1$$
$$ADCS_{min} = 3{,}66$$
$$ADCS_{min} \approx 4$$

La aproximación debe hacerse hacia arriba para no obtener un valor inferior al mínimo requerido en el TAD. Al reemplazar el valor de 4 en ADCS se obtiene que el valor del TAD es 126.17 ns, cumpliendo así el requisito mínimo del reloj del conversor. Este será el parámetro que será configurado mediante las dos siguientes líneas:

```
AD1CON3bits.ADRC = 0;
AD1CON3bits.ADCS = 4;
```

La Figura **4-6** ilustra el manejo del capacitor del circuito de muestreo y retención durante los momentos de muestreo y de conversión.

En modo 12 bit, el ADC requiere de 14 TAD para efectuar una conversión.

$$T_{C12bit} = 14 \times T_{AD}$$
$$T_{C12bit} = 14 \times 126{,}17 \quad ns$$
$$T_{C12bit} = 1{,}77 \quad us$$

Un período de conversión TC no puede ser muy largo porque el voltaje en el capacitor de retención puede decaer lo suficiente como para alterar el resultado. Otro tiempo mínimo que debe respetarse es el tiempo de muestreo TSAMP que debe ser de por lo menos 3 TAD, o sea 352.8 ns para que el capacitor se cargue hasta un valor acertado.

El tiempo de muestreo en la aplicación depende de la frecuencia de muestreo de audio y está dado por:

$$T_{SAMP} = T_S - T_{C12bit}$$

Asumiendo que la frecuencia de muestreo de audio es la máxima del DAC, o sea 100 KHz, se obtiene:

$$T_{SAMP} = 10 \quad us - 1{,}77 \quad us$$

$$T_{SAMP} = 8{,}23 \quad us$$

El cual es unas 23 veces mayor al necesario, este parámetro no tiene inconveniente teniendo en cuenta que el ADC del dsPIC® está diseñado para llegar a velocidades de 500 ksps con resolución de 12 bits.

Sólo resta configurar las características DMA para el ADC, hay un bit de interés que configura el modo como el ADC llena el espacio DMA, en este caso es preferible que esto se haga en el mismo orden de conversión, para ello se usa la siguiente instrucción:

```
AD1CON1bits.ADDMABM = 1;
```

También es muy necesario controlar que el incremento de la dirección DMA RAM se haga una sola vez por cada conversión, este parámetro llamado razón de incremento de la dirección DMA se configura con el registro $SMPI < 3:0 >$ así:

```
AD1CON2bits.SMPI    = 0;
```

Ya está totalmente configurado el módulo ADC, ahora puede encenderse:

```
AD1CON1bits.ADON = 1;
```

A continuación vemos cómo quedó la función completa:

```
void ConfigADC(void)
{
    // Multiplexores del canal 0
    AD1CON2bits.CHPS = 0;       // Indica que el canal 0 es el canal de
        conversión
    AD1CHS0bits.CH0SA = 4;      // AN4 se conecta a la entrada +ve
    AD1CHS0bits.CH0NA = 0;      // VREFL se conecta a la entrada -ve

    // Multiplexor de los voltajes de referencia
    AD1CON2bits.VCFG = 0;   // Los voltajes de referencia son AVDD y AVSS
```

```
    // Formato de los datos
    AD1CON1bits.AD12B = 1;        // Indica resolución de 12 bits
    AD1CON1bits.FORM  = 3;        // Formato de datos: Fraccional con signo (Q15
        )

    // Modo de muestreo y conversión
    AD1CON1bits.ASAM  = 1;        // Inicia muestreo: al terminar la última
        conversión
    AD1CON1bits.SSRC  = 0;        // Inicia conversión: limpiando el bit sample

    // Reloj de conversión
    AD1CON3bits.ADRC = 0;         // El reloj del ADC se deriva del reloj del
        sistema
    AD1CON3bits.ADCS = 4;         // TAD = (ADCS+1)/FCY = 126.17 ns, TC = 1.77
        us

    // Características DMA
    AD1CON1bits.ADDMABM = 1;// Los búfer DMA se llenan en orden de conversión
    AD1CON2bits.SMPI    = 0;// Por cada conversión incrementa una dirección
       DMA

    AD1CON1bits.ADON = 1;         // Enciende el módulo ADC
}
```

A continuación se creará otra función, esta vez para configurar el canal 0 del módulo DMA para que transfiera datos entre el búfer del ADC y el espacio DMA RAM.

```
void ConfigDMA0(void)
{
    DMA0CONbits.AMODE = 0;      // Registro indirecto con post incremento
    DMA0CONbits.MODE  = 0;      // Modo continuo sin Ping-Pong

    DMA0PAD = (int)&ADC1BUF0;  // Dirección del búfer del ADC (0300)
    DMA0CNT = (TamBufer-1);  // Número de transferencias DMA (N_MUESTRAS-1)

    DMA0REQ =  13;       // La interrupción del ADC provoca transferencia DMA

    DMA0STA = __builtin_dmaoffset(BuferA); // Dirección inicio para el búfer A

    IFS0bits.DMA0IF  = 0;                 // Limpia la bandera de interrupción
        el DMA
    IEC0bits.DMA0IE  = 1;                 // Activa la interrupción de DMA

    DMA0CONbits.CHEN = 1;                 // Activa el canal 0 del módulo DMA
}
```

Donde $N\_MUESTRAS$ es un valor que debe ser definido con anterioridad y equivale al tamaño de cada búfer.

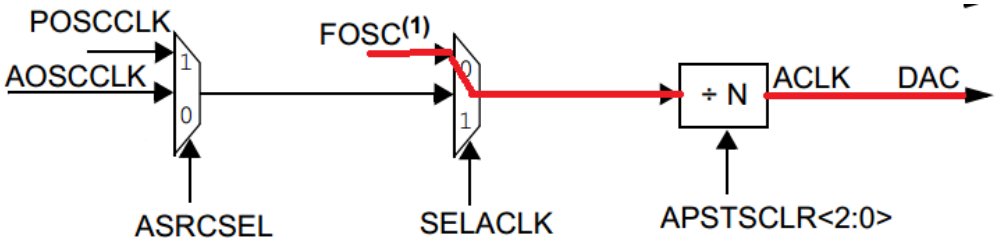

Figura 4-7.: Configuración del oscilador de entrada al DAC (Adaptado de [28]).

4.1.4. Configuración del DAC

La principal característica del DAC del dsPIC33FJ128GP802 es que está diseñado específicamente para aplicaciones de audio. Este dispositivo debe configurarse para una frecuencia de muestreo constante, esta frecuencia se deriva del oscilador auxiliar (ACLK) a través de un divisor de frecuencia. La salida se actualiza de forma automática tomando los datos del búfer, si el búfer está vacío en el momento de la actualización la salida será el valor por defecto guardado en el registro DACDFLT. La máxima frecuencia de muestreo recomendada es 100 kHz, y la máxima frecuencia de audio permitida es 45 kHz.

Este módulo tiene en total 5 registros directamente asociados. De los cuales 2 son para el envío de datos (izquierdo y derecho), uno es para guardar el valor por defecto (en caso de búfer vacío) y los otros dos son los registros de control y de estado. En el registro de control están los bits del divisor de frecuencia, el de forma de los datos, el de activación del amplificador, operación en modos de bajo consumo y el de activación del módulo. La figura **4-7** muestra cómo será tomada la frecuencia del DAC.

Vemos que son dos los elementos de interés, el bit SELACK y el registro $APSTSCLR < 2:0 >$, el primero se usa para seleccionar la fuente de reloj, en este caso será FOSC que está conectado a la salida del circuito PLL, y con el segundo configuramos el divisor de esta frecuencia, la señal obtenida a la salida es llamada ACLK (Reloj Auxiliar). Antes de ser aplicada al DAC, esta señal pasa por otro divisor controlado por el registro DACFDIV, de esta forma se obtiene la frecuencia de trabajo del DAC, la cual no puede superar los 12.8 MHz, que es el valor para que la frecuencia de muestreo se establezca en 100 ksps. Un muestreo ocurre cada 128 ciclos de reloj de ACLK. A continuación se muestra cómo se programa la frecuencia del DAC, téngase en cuenta que el valor DIV_DAC debe haberse definido con anterioridad y no debe ser menor a 6 para garantizar un óptimo funcionamiento del sistema con las condiciones de reloj que estamos configurando.

```
ACLKCONbits.SELACLK  = 0;
ACLKCONbits.APSTSCLR = 7;
DAC1CONbits.DACFDIV  = DIV_DAC;
```

Otra configuración importante que se debe hacer es indicar el formato de los datos que estamos enviando al DAC, de los dos tipos de datos el más conveniente es el entero con signo, ya que es compatible con el formato fraccional que se maneja en el procesamiento de señales.

```
        DAC1CONbits.FORM     = 1;
```

El registro de estado contiene los bits de estado del DAC, el byte más significativo aplica para el canal izquierdo y el menos significativo para el canal derecho. Cada canal tiene 5 bits de en el registro de estado: activar o desactivar canal, activar o desactivar voltaje central, tipo de interrupción, búfer lleno y búfer vacío.

Una característica de este periférico es que genera interrupción en uno de dos casos, según se configure: cuando el búfer del DAC está lleno o cuando dicho búfer está vacío. El búfer de cada canal tiene una profundidad de 4 palabras. Este periférico también tiene la capacidad de transferir datos mediante DMA. La siguiente figura muestra la función de configuración de DAC propuesta.

```
void ConfigDAC(void)
{
    ACLKCONbits.SELACLK  = 0;    // Salida del PLL como fuente de reloj
    ACLKCONbits.APSTSCLR = 7;    // Divide la frecuencia de entrada entre 1

    DAC1CONbits.DACFDIV = DIV_DAC;  // DACCLK = ACLK/(DACFDIV + 1)
    DAC1CONbits.FORM    = 1;        // Formato de datos: entero con signo
    DAC1CONbits.AMPON   = 0;        // Amplificador activo en modos Sleep/Idle
    DAC1DFLT            = 0x8000;   // Valor de salida por defecto para el DAC

    DAC1STATbits.ROEN   = 1;    // Activa el canal derecho del DAC
    DAC1CONbits.DACEN   = 1;    // Activa el DAC
}
```

A continuación se muestran los macros construidos para obtener diferentes tasas de muestreo en el DAC, en este caso seleccionado para muestrear a 44100 Hz:

```
#define Fs_100K      5        // fs real = 103200 Hz
#define Fs_48000     12       // fs real = 47631 Hz
#define Fs_44100     13       // fs real = 44229 Hz
#define Fs_32000     18       // fs real = 32589 Hz
#define Fs_22050     27       // fs real = 22114 Hz
#define Fs_16000     38       // fs real = 15877 Hz
#define Fs_11025     55       // fs real = 11057 Hz
#define Fs_8000      76       // fs real = 8042 Hz

#define DIV_DAC      Fs_44100     // Establece frecuencia de muestreo (Fs)
```

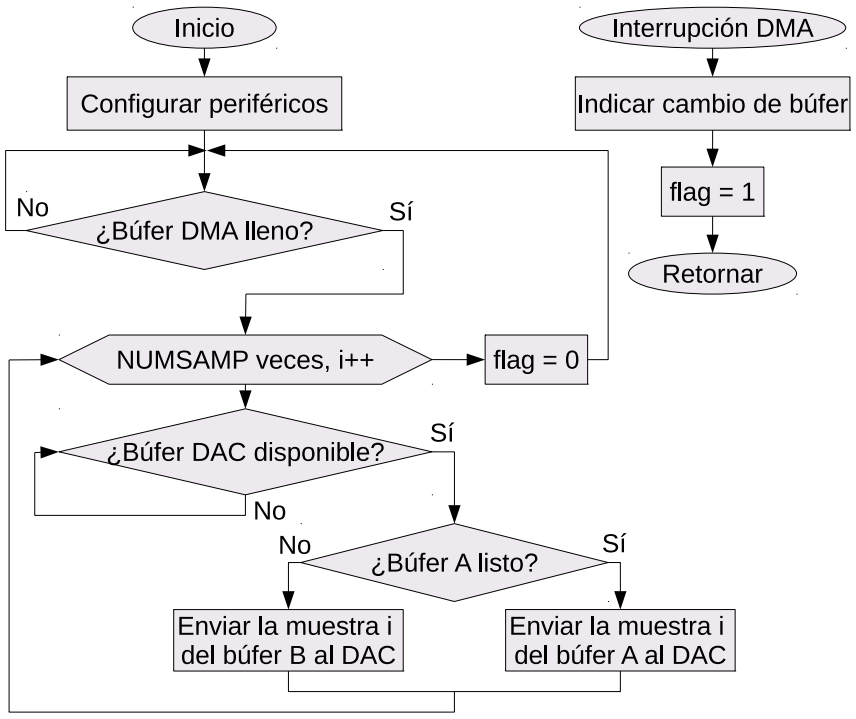

Figura 4-8.: Diagrama de flujo del código de ejemplo CE154.

4.2. Muestreo y reconstrucción de señales de audio

Estando definidas las funciones que configuran el dispositivo ahora es posible muestrear una señal de audio mediante el ADC y enviar esas muestras al DAC, creando así un puente por software entre los convertidores.

Como punto de partida se ha tomado el código de ejemplo CE154 de la página de Microchip, su diagrama de flujo se muestra en la Figura **4-8**.

El programa se resume de la siguiente manera: el ADC es configurado para realizar conversiones disparadas por el temporizador 3 cada 9.69 us, esto con el objetivo de igualar a la frecuencia del DAC que es de 103.2 KHz, los datos son almacenados mediante DMA en 2 búfer de 256 palabras cada uno, estos búfer se llenan en modo ping-pong, cada vez que uno de los búfer se llena es generada una interrupción que cambia el búfer desde donde el DAC toma las muestras, la escritura de los datos en el búfer del DAC se hace desde un bucle infinito que está constantemente leyendo el estado vacío del búfer del DAC para saber cuándo puede enviar otra muestra. El modo ping pong se resume en la Figura **4-9**.

El código de ejemplo CE154 fue probado y se pudo evidenciar que no funciona correctamente, pues la señal aplicada a la entrada no se reconstruía de forma satisfactoria. La Figura

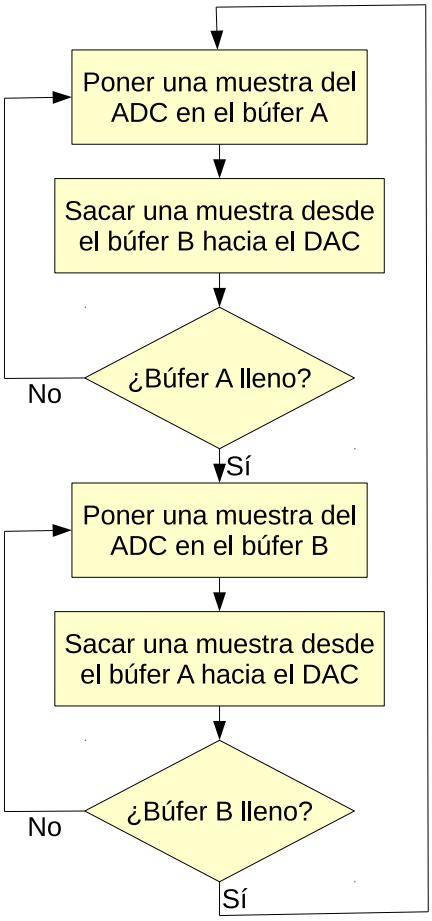

Figura 4-9.: Diagrama de flujo del algoritmo ping-pong.

Figura 4-10.: Comportamiento de la señal implementada con el código de ejemplo CE154.

4-10 muestran el comportamiento del dsPIC® con el código de ejemplo implementado.

En la búsqueda del problema se estableció que el código había sido modificado con el objetivo de adaptarlo de la versión MPLAB® 8 a MPLAB® X, en esta adaptación se cometió el error de calcular mal la fórmula del divisor de frecuencia del DAC. Además de eso, se encontró un problema en el tiempo TAD, el cual fue calculado para 100 ns por el programador del ejemplo mientras que no debía ser menor a 117.6 ns.

Al corregir el valor del registro DACFDIV a 5, se obtiene efectivamente la frecuencia de muestreo indicada en el ejemplo (103 kHz) sincronizando así con el timer y ahora la onda puede ser reconstruida satisfactoriamente por el DAC como se muestra en la Figura **4-11**.
En el análisis del código de ejemplo anterior se encuentra que el enfoque empleado no es el más adecuado ya que se basa en la configuración de dos fuentes de reloj complicando así la sincronización entre el ADC y el DAC, teniendo en cuenta que las frecuencias de muestreo de estos dos módulos deberían estar perfectamente sincronizadas se ha buscado mejorar el código mediante un enfoque hacia una sola fuente de reloj, para este fin se hace necesario prescindir del Timer 3 y hacer de forma manual el disparo del ADC.

El primer objetivo para solucionar el problema fue comprobar el funcionamiento punto a punto, luego se procedió a reescribir el código incorporando el módulo DMA. Con los resultados, se ha propuesto cuatro algoritmos que aseguran que las frecuencias de trabajo de los dos convertidores coinciden perfectamente.

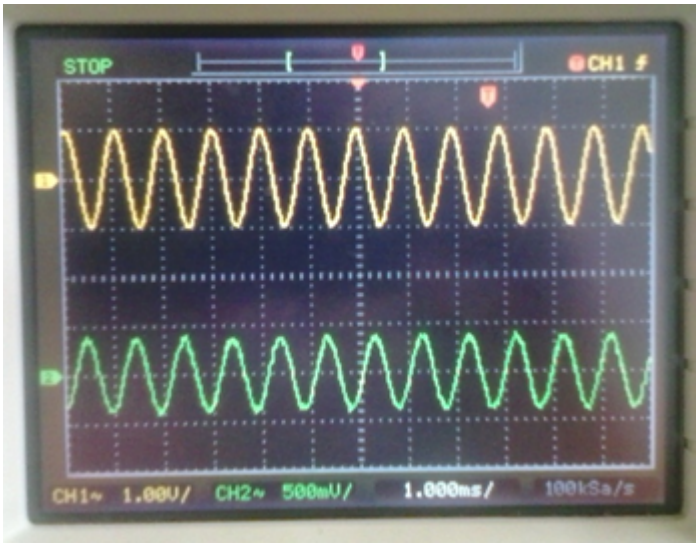

Figura 4-11.: Comportamiento de la señal implementada con el código de ejemplo CE154 modificado.

4.2.1. Muestreo y reconstrucción punto a punto

En esta configuración, ilustrada en la Figura **4-12**, no hace uso de interrupciones sino que se deja un flujo libre del programa a través de un bucle infinito, el cual comprueba repetitivamente si el búfer del DAC está disponible para recibir una nueva muestra.

4.2.2. Muestreo y reconstrucción punto a punto con interrupción del DAC

En esta ocasión el programa principal sólo configura los periféricos, la interrupción del DAC evita la necesidad de comprobar el estado del búfer de este último, dicha interrupción contiene la orden de muestreo, cuando la señal muestreada es finalmente convertida se da la orden de enviar la muestra al búfer del DAC.

4.2.3. Muestreo y reconstrucción mediante búfer DMA

La incorporación del acceso directo a memoria (DMA) permite prescindir del búfer del ADC para obtener los datos, pues estos son colocados directamente en el espacio de memoria RAM sin intervención de la CPU luego de cada conversión, de esta manera se puede tratar a las muestras por bloques de tamaño especificado en la configuración del módulo DMA, se ha propuesto el uso de dos bloques para hacer uso del algoritmo ping-pong, el cual es soportado por el dsPIC®. La interrupción generada por el DMA es usada para intercalar el búfer del que se toman las muestras de salida y para reinicializar la variable i, usada para recorrer

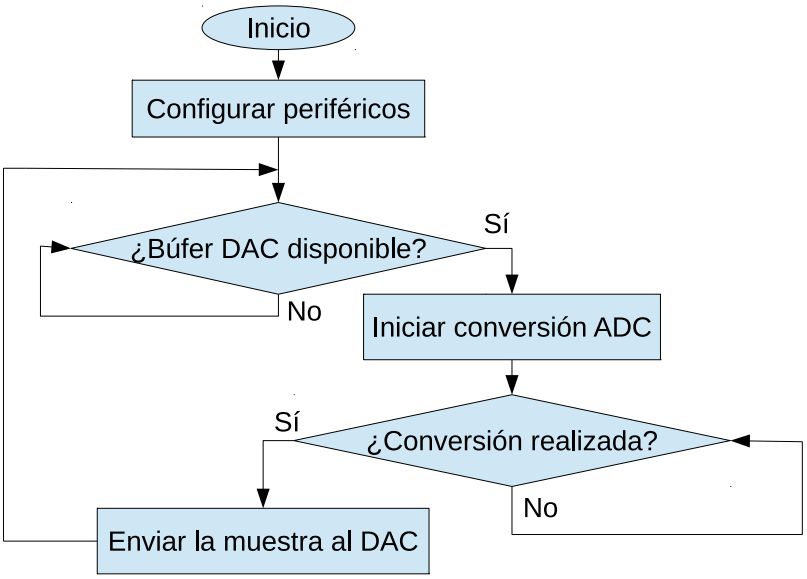

Figura 4-12.: Muestreo y reconstrucción punto a punto.

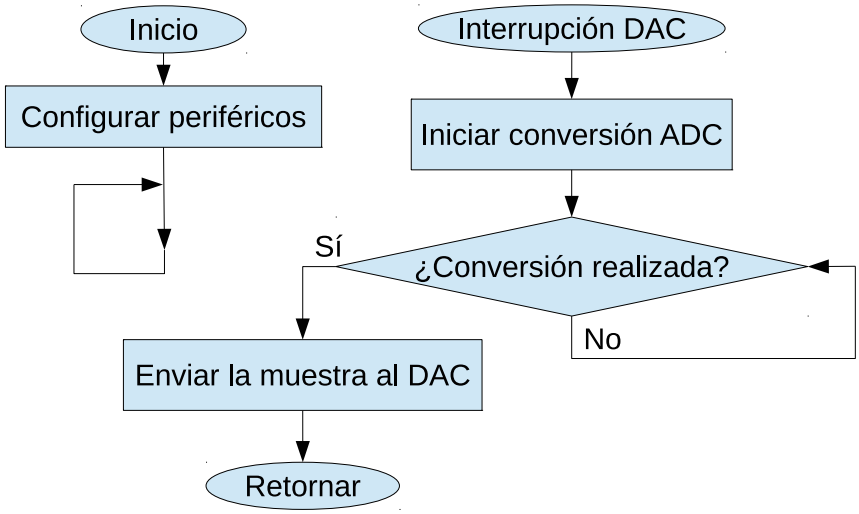

Figura 4-13.: Muestreo y reconstrucción punto a punto con interrupción del DAC.

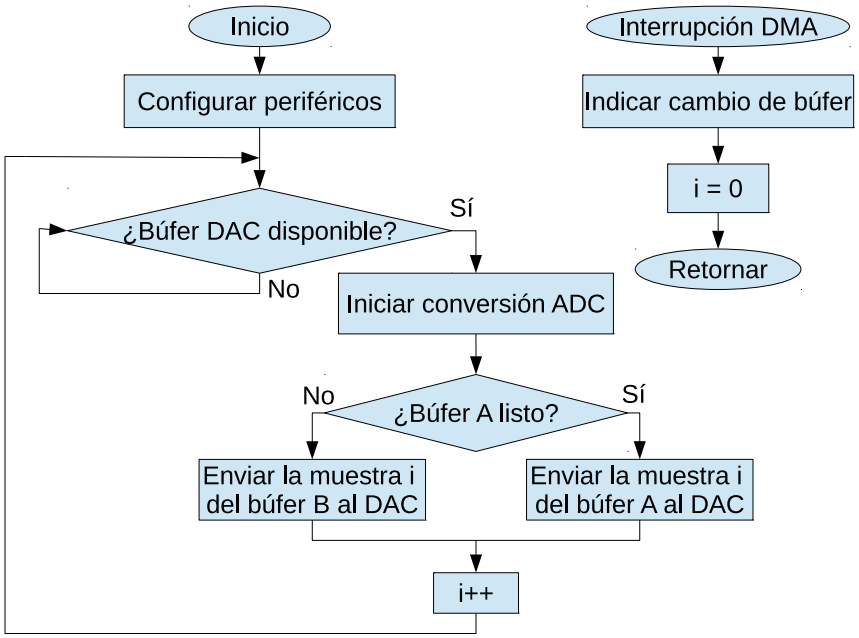

Figura 4-14.: Muestreo y reconstrucción mediante búfer DMA.

cada bloque. Véase la Figura **4-14**.

4.2.4. Muestreo y reconstrucción mediante búfer DMA con interrupción del DAC

Este algoritmo es bastante similar al anterior, pero dejando que la interrupción del DAC controle la obtención de las muestras del ADC y el envío de las muestras de salida. En resumen el programa principal no tiene control del muestreo, todo se controla mediante interrupciones, de esta manera, se puede implementar procesamiento por bloques desde el programa principal de una forma eficiente. Véase la Figura **4-15**.

4.3. Generación de señales periódicas mediante tablas

4.3.1. Búfer circular

Un búfer circular (Figura **4-16**) es un arreglo de una cantidad fija de datos que considera a la última dirección como adyacente a la primera, de este modo se puede tratar a las direcciones como un bucle sin fin permitiendo así un movimiento circular de datos.

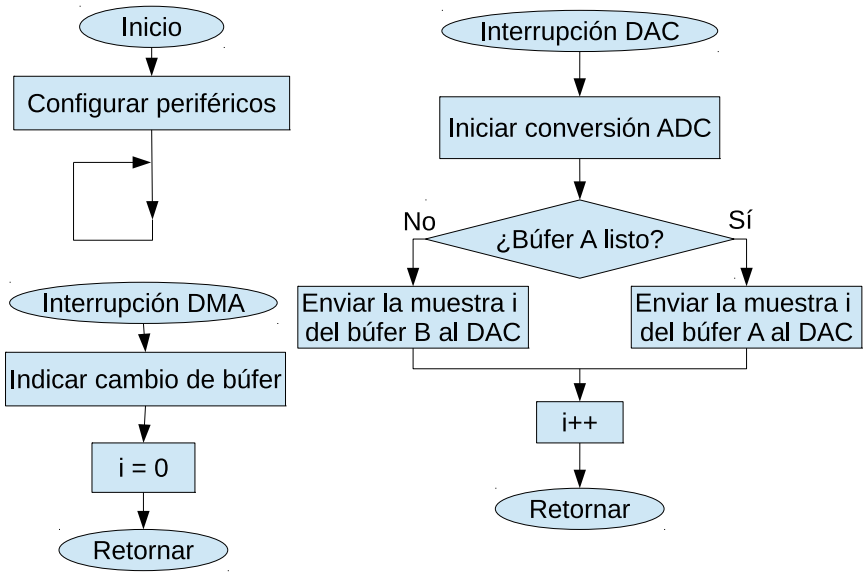

Figura 4-15.: Muestreo y reconstrucción mediante búfer DMA con interrupción del DAC.

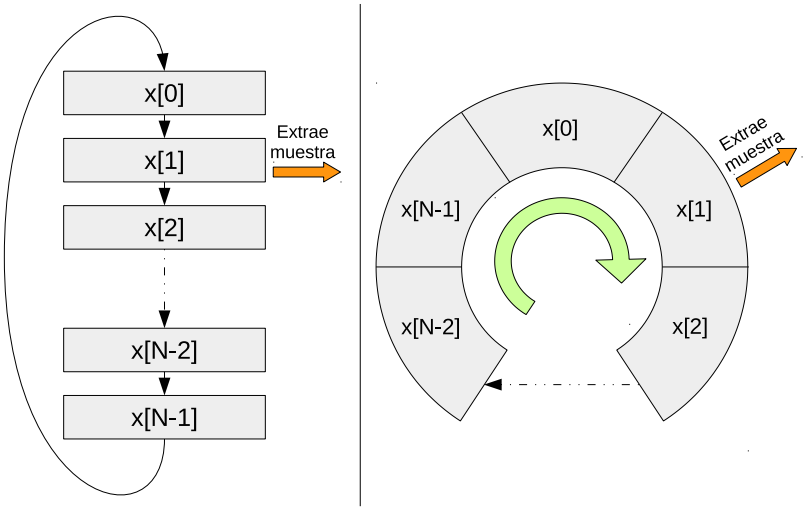

Figura 4-16.: Búfer circular de tamaño N.

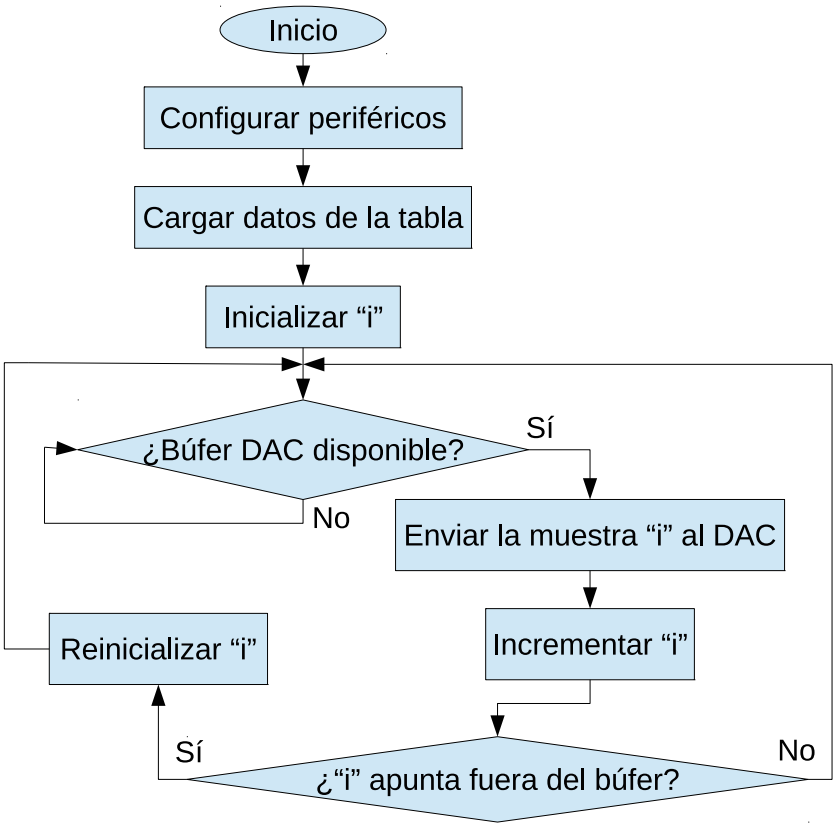

Figura 4-17.: Generación de señales periódicas mediante búfer circular.

Uno de los usos más comunes del búfer circular en cuanto al audio es la generación de señales periódicas mediante tablas que guarden muestras de cualquier señal, una tabla entonces se puede convertir en un búfer circular de sólo lectura. La Figura **4-17** ilustra un ejemplo de este tipo en donde se envían muestras desde una tabla con tamaño igual a $TamOnda$ hacia el DAC, la variable i actúa como puntero. A continuación se muestra un código que implementa la parte encerrada en el bucle principal de dicho ejemplo:

```
while(DAC1STATbits.REMPTY != 1);// Espera a que el DAC esté disponible

DAC1RDAT = Onda[i];             // Escribe la muestra de la onda en el DAC

i++;                            // Incrementa el puntero del búfer
if(i==TamOnda)                  // Si el puntero alcanza el tamaño del
    búfer
    i = 0;                      // Lo reinicializa
```

Como se observa en el anterior código, es necesario incrementar manualmente el puntero y comprobar constantemente el límite del búfer para corregir la dirección si es necesario, estas operaciones se pueden realizar automáticamente mediante una tecnología llamada direccio-

namiento modular.

4.3.2. Direccionamiento modular

Esta característica de los dsPIC® permite un tratamiento eficiente de búfer circulares ya que las direcciones son corregidas a través del hardware.

El direccionamiento modular se configura a través de 5 registros, el registro de control MODCON configura el modo de operación, los registros XMODSRT y XMODEND configuran las direcciones de inicio y de fin respectivamente para el bus de datos X, mientras que YMODSRT y YMODEND hace lo mismo con el bus de datos Y. No debe realizarse una operación que involucre direccionamiento indirecto justo después de escribir en alguno de los anteriores registros, se sugiere el uso de la operación NOP si el programa presenta este caso.

Los registros W actúan como punteros cuando hay direccionamiento modular, estos pueden ser usados con direccionamiento indirecto con post incremento y la dirección contenida en dicho registro se ajustará automáticamente si traspasa el límite del búfer.

El búfer circular puede diseñarse para incremento o para decremento, esto depende de si los registros de inicio contienen direcciones más grandes o más pequeñas que los registros de fin.

A continuación se muestra un código para configurar direccionamiento modular con incremento para el búfer llamado *Onda* y con $W1$ como puntero:

```
XMODSRT = (int)&Onda;                    // Dirección de inicio de la tabla
XMODEND = (int)&Onda+(TamOnda*2-1); // Dirección de fin de la tabla
MODCON = 0x8001;                         // Configura W1 como puntero
Nop();
```

La Figura **4-18** implementa otro sistema de generación de señales pero esta vez utilizando direccionamiento modular. Las siguientes líneas de código muestran un ejemplo de cómo se puede enviar muestras desde una tabla hacia el DAC utilizando esta técnica, el puntero es salvado en la variable *_Puntero*:

```
asm("mov _Puntero, w1");
asm("mov [w1++], w2");
asm("mov w2, _DAC1RDAT");
asm("mov w1, _Puntero");
```

Como se observa, las instrucciones están programadas en ensamblador porque es necesario programarlas a nivel de los registros de trabajo. En este ejemplo se ha sugerido que en cada

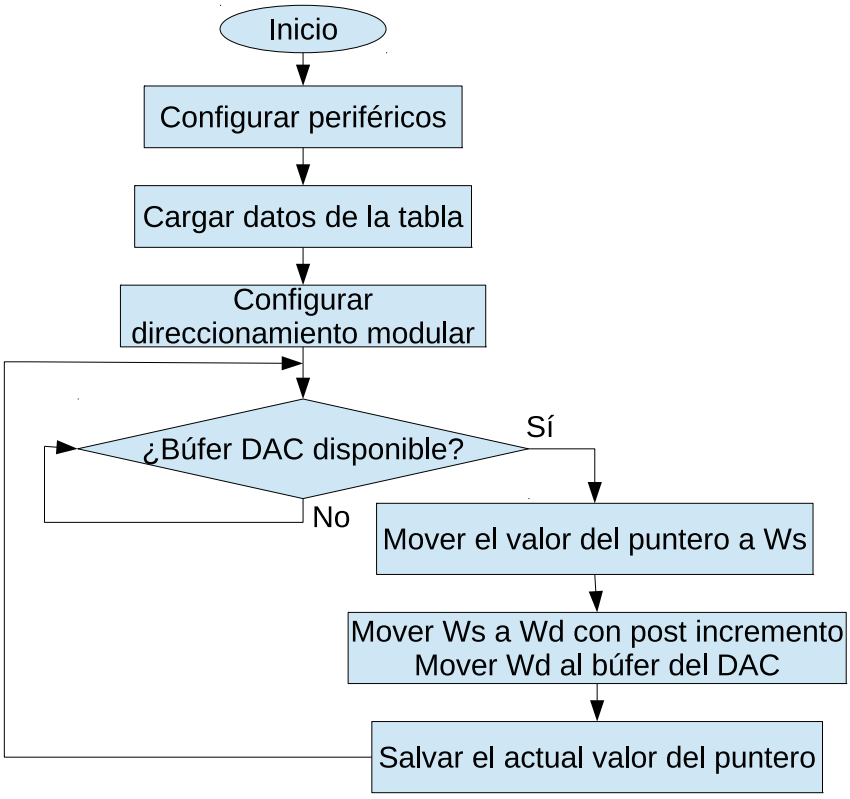

Figura 4-18.: Generación de señales periódicas mediante búfer circular con direccionamiento modular.

Figura 4-19.: Peso de los bits en el formato fraccional.

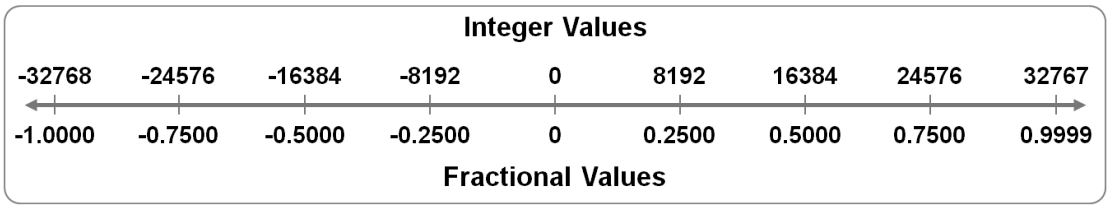

Figura 4-20.: Equivalencia entre el formato entero y el formato fraccional [17].

iteración se guarde en una variable el valor del puntero ya que los registros de trabajo son diseñados para cargar operandos y no para almacenar resultados.

4.4. Operaciones de procesamiento digital en el dominio temporal

La arquitectura del dsPIC® está diseñada para procesamiento digital con datos en formato Q15, también conocido como formato *fraccional*. Este es un formato de aritmética de punto fijo que consiste en 15 números fraccionarios y un bit de signo (MSb). El formato *fraccional* es compatible con el formato *entero* pero interpretando los datos de una manera diferente. La Figura **4-19** muestra qué valor tiene cada bit de acuerdo a su posición es el formato *fraccional*. la Figura **4-20** hace una equivalencia entre los formatos *entero* y *fraccional*.

Esto implica que los datos expresados en formato *fraccional* tienen un rango que va desde -1 hasta 0,999969482422 en el dsPIC®.

Son 3 los tipos de operaciones que se pueden aplicar en el dominio temporal a una señal digital de audio: escalamiento, retraso y adición. El escalamiento consiste en multiplicar la señal por un factor llamado ganancia, el retraso de la señal está asociado al almacenamiento de muestras durante cierto tiempo y la adición se realiza entre señales presentes o pasadas generalmente multiplicadas por un coeficiente.

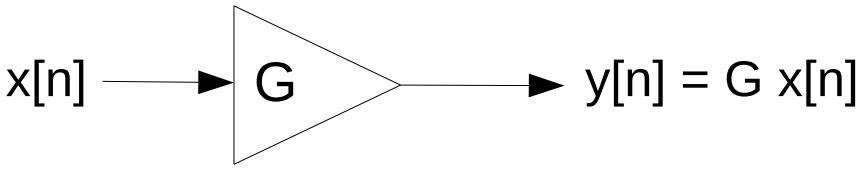

Figura 4-21.: Representación de la operación escalamiento.

4.4.1. Escalamiento de la señal de audio

La señal se puede escalar de dos maneras: mediante multiplicación por un valor de ganancia o mediante desplazamiento de bits. Si el resultado es una señal de mayor amplitud la operación se conoce como amplificación, si el resultado tiene menor amplitud entonces se conoce como atenuación. La Figura **4-21** es la representación de esta operación en un diagrama de bloques.

Multiplicación DSP

El multiplicador del dsPIC® acepta operandos de 17 bits, como los datos de entrada son de 16 bits existe una conversión interna que agrega el bit faltante, el resultado de esta operación tiene un tamaño de 32 bits. Las multiplicaciones DSP se consideran como resultados atenuados debido al formato $fraccional$.

Existen 4 instrucciones DSP que permiten realizar multiplicaciones, estas se listan en la Tabla **4-1**.

Tabla 4-1.: Instrucciones DSP que permiten realizar multiplicaciones.

Instrucción DSP	**Operación que realiza**
MPY	$a = b \times c$
MPY.N	$a = -b \times c$
MAC	$a = a + b \times c$
MSC	$a = a - b \times c$

La instrucción MPY es la multiplicación directa; MPY.N es la misma operación pero invirtiendo el signo del resultado. Las otras dos instrucciones son análogas a las anteriores pero en este caso el resultado es sumado a un acumulador.

Multiplicación MCU

Las multiplicaciones MCU, al igual que las multiplicaciones DSP, son de 16x16 bits pero se diferencian en que la multiplicación MCU permite elegir si el resultado se almacenará en un acumulador DSP o directamente en los registros W, además permite seleccionar si el resultado será expresado en 16 o 32 bits. El registro fuente puede ser cualquiera de los registros W y el registro destino debe ser un registro W de dirección par. Cuando la multiplicación está indicada para una respuesta de 32 bits, el byte más significativo quedará almacenado en la dirección impar adyacente mayor a la del registro destino. Una de las ventajas de esta operación es que el segundo operando admite direccionamiento inmediato de un literal de 5 bits, la multiplicación MCU permite seleccionar el signo con el que se interpretarán los operandos.

La instrucción *MUL* dará una respuesta de 32 bits mientras que la instrucción *MULW* dará una respuesta de 16 bits, esta última se usa en casos donde se sabe que la magnitud a multiplicar dará un resultado pequeño. La Tabla **4-2** resume las operaciones de multiplicación MCU, cada una se ejecuta en un ciclo de máquina.

Tabla 4-2.: Instrucciones MCU que permiten realizar multiplicaciones.

Instrucción MCU	**Operación que realiza**
MUL [1]	Multiplica W0 por el valor de la memoria de datos indicado, almacena la respuesta en W2 y W3.
MUL.SS ; MULW.SS	Multiplicación entre enteros con signo.
MUL.SU ; MULW.SU	Entero con signo por entero sin signo.
MUL.US ; MULW.US	Entero sin signo por entero con signo.
MUL.UU ; MULW.UU	Multiplicación entre enteros sin signo.

[1] Se ha ignorado la operación MUL.B considerando que las multiplicaciones de tipo byte no son relevantes para el procesamiento digital de audio.

Desplazamiento de bits

El desplazamiento de bits se realiza mediante el registro de desplazamiento, el cual puede desplazar un máximo de 16 bits por ciclo. De esta forma es posible tanto amplificar como atenuar una señal mediante operaciones de desplazamiento a la izquierda y desplazamiento a la derecha respectivamente. La amplificación en un sistema digital es muy impredecible respeto a las posibles saturaciones, por eso generalmente las operaciones de escalamiento de la señal se limitan a atenuarla mediante desplazamientos a la derecha. En la Tabla **4-3** se consignan las instrucciones que permiten realizar desplazamientos, las tres primeras aplican a la memoria de datos y las otras tres operan sobre los acumuladores de 40 bits.

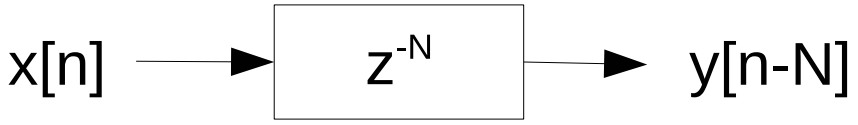

Figura 4-22.: Representación de la operación retardo de señal.

Tabla 4-3.: Instrucciones DSP que permiten realizar desplazamientos.

Instrucción	**Operación que realiza**
ASR	Desplazamiento aritmético hacia la derecha.
LSR	Desplazamiento lógico hacia la derecha.
SL	Desplazamiento hacia la izquierda.
SFTAC	Desplazamiento del acumulador.
SAC; SAC.R	Almacena el acumulador con desplazamiento opcional.
LAC	Carga el acumulador con desplazamiento opcional.

Además de estas, la instrucción ADD también permite desplazamientos opcionales. El desplazamiento de bits más común en el procesamiento digital de audio es el aritmético hacia la derecha, el cual permite conservar el signo de la muestra y nunca resulta en saturación.

4.4.2. Retardo de la señal de audio

El retraso de una señal digital está asociado al almacenamiento en memoria de las muestras. Para la tarjeta diseñada, las muestras obtenidas por el ADC se pueden almacenar mediante el módulo DMA o manualmente copiando el valor del búfer para almacenarlo en alguna parte de la memoria RAM. El primer método permite un almacenamiento de máximo 2 kb, y el segundo un almacenamiento máximo equivalente a la RAM del dsPIC® (16 kb para el DSPIC33FJ128GP802). En algún momento del programa, las muestras almacenadas serán requeridas para cumplir el procesamiento que se esté llevando a cabo. Esta operación tiene diferentes tipos de direccionamiento que permiten optimizar el almacenamiento y la obtención de las muestras retrasadas, un direccionamiento muy común en este caso es el direccionamiento modular, usado para el tratamiento de búfer circulares. La Figura **4-22** es la representación de esta operación en un diagrama de bloques.

4.4.3. Adición de señales de audio

La adición de señales es la operación de suma o resta entre dos o más señales, un ejemplo típico es la adición de una señal de audio actual con una señal de audio atenuada y retrasada,

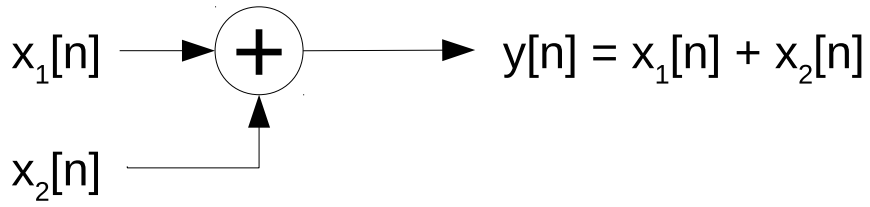

Figura 4-23.: Representación de la operación adición de señales.

el resultado es conocido como eco y es el más sencillo ejemplo de procesamiento donde se involucran las 3 operaciones del dominio temporal: retrasa la señal, escala la señal retrasada y la suma a la señal actual. La Figura **4-23** es la representación de esta operación en un diagrama de bloques. Las instrucciones de la tabla **4-4** involucran adición de señales.

Tabla 4-4.: Instrucciones DSP que permiten realizar adición de señales.

Instrucción	**Operación que realiza**
ADD	$a = a + b$
SUB	$a = a - b$
MAC	$a = a + b \times c$
MSC	$a = a - b \times c$

4.5. Efectos de audio con base en escalamiento de la señal

4.5.1. Ganancia

Éste es el procesamiento digital más sencillo que se puede efectuar con una señal, consiste en tomar las muestras de dicha señal y aplicarle cualquiera de las operaciones mencionadas en la sección 4.4.1 siguiendo un diagrama de flujo como el de la Figura **4-24**, el cual contiene una subrutina llamada *Procesar muestra* encargada de realizar el respectivo procesamiento de la señal de audio.

4.5.2. Trémolo

Las técnicas de modulación de amplitud pueden aplicarse a señales de audio para obtener un efecto llamado trémolo. Esta idea consiste en aplicar una señal de baja frecuencia, menor a 20 Hz, que se encargará de modular a la señal de audio, la primera se comportará como señal moduladora y la última como señal portadora. Este tipo de procesamiento escala la

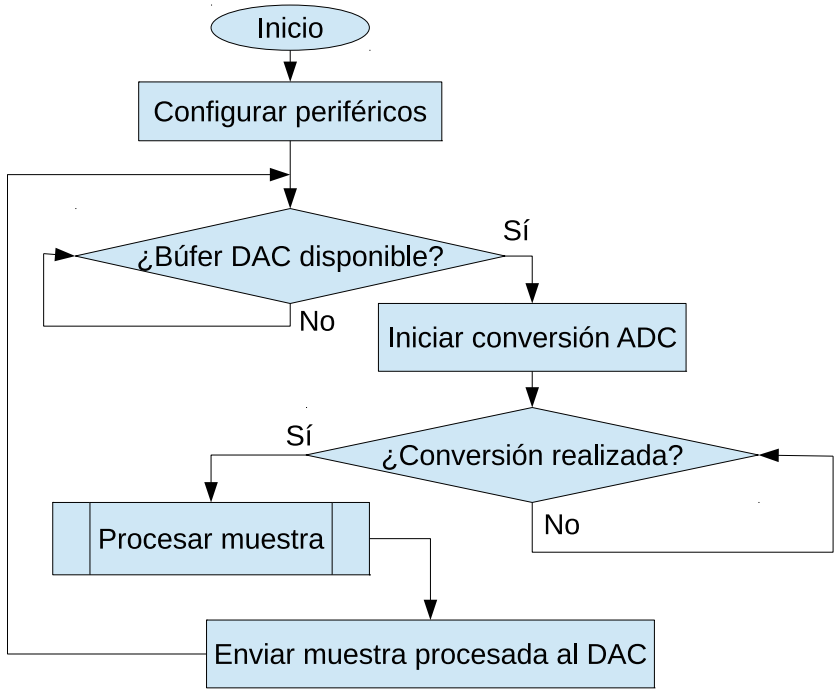

Figura 4-24.: Procesamiento muestra a muestra.

señal de audio de acuerdo a una envolvente creada por la señal moduladora. Según [11, pp. 303-304] la ecuación que representa una señal AM está dada por:

$$s\,(t) = A_c\,[1 + m\,(t)]\,cosw_c t$$

Donde A_c expresa la potencia de la señal, $m\,(t)$ la señal moduladora y $cosw_c t$ la señal portadora, el comportamiento de esta modulación se puede apreciar en la Figura **4-25**. Ignorando la potencia de la señal y generalizando la señal portadora a señales con cualquier forma $x\,(t)$ se obtiene la siguiente ecuación:

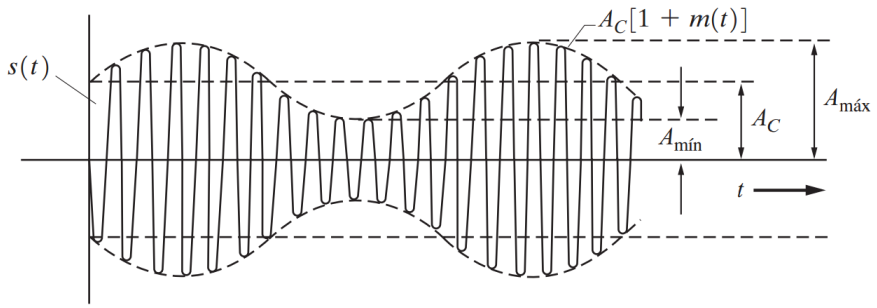

Figura 4-25.: Gráfico general de una modulación AM [11, p. 304].

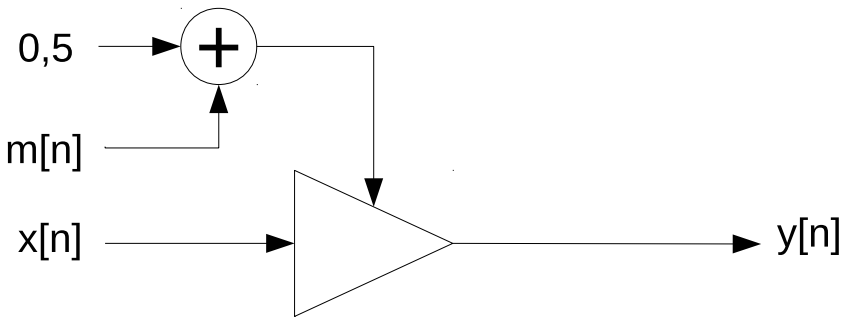

Figura 4-26.: Sistema de trémolo mediante modulación AM.

$$s(t) = [1 + m(t)]\, x(t)$$

Por último, se debe expresar esta ecuación en forma digital. Teniendo en cuenta que una modulación AM del 100 % implica que la amplitud de la señal de audio aumentará al doble, se utilizará una ecuación normalizada donde el factor de modulación tenga un máximo de 1, con esto se previene la saturación en el dsPIC®, esta ecuación entonces quedará de la siguiente manera:

$$y[n] = (0{,}5 + m[n]) \times x[n]\,; \qquad |m[t]| \leq 0{,}5$$

Donde $x[n]$ es la señal portadora (señal de audio) y $m[n]$ es la señal moduladora (cualquier señal de baja frecuencia), esta última puede ser, por ejemplo, una señal periódica guardada en un búfer circular. El diagrama de bloques de este sistema es mostrado en la Figura **4-26** y es implementado con el código a continuación:

```
.include "p33fj128gp802.inc"

.equ AmpPort,    0x4000   ; Amplitud de portadora normalizada a 0.5

.global _ModAmplitud      ; y[n] = (AmpPort + _FactorMod * _Moduladora)*(
   _Muestra)

_ModAmplitud:
    mov     _Muestra, w4     ; Señal de audio
    mov     _FactorMod, w5   ; Factor de escalado de señal moduladora
    mov     _Moduladora, w6  ; Parámetros para escalar la señal moduladora

    mpy     w5*w6, A     ;   Escala la señal moduladora
    sac.r   A, w6        ;   Señal moduladora escalada

    mov     #AmpPort, w5;   Amplitud de la portadora
    lac     w5, A
```

```
    add     w6, A       ;   Suma amplitud de portadora con señal moduladora
    sac.r   A, w7       ;   Extrae el resultado en W7

    mpy     w4*w7, A    ;   Multiplica la muestra por el factor de modulación
    sac.r   A, w7       ;   Extrae el resultado del acumulador A en W7

    mov     w7, _MuestraModulada;  y lo envía a MuestraModulada

    return
.end
```

Nótese que el factor de modulación es el principal parámetro en esta operación, este incide directamente en la envolvente de la señal de audio, la forma de la envolvente está dada por la muestra guardada en la variable $\_Moduladora$:

Es posible usar pocas muestras de la señal moduladora si el búfer circular se recorre más lentamente a través de la repetición de una misma muestra antes de pasar a la siguiente como se observa en el diagrama de flujo de la Figura **4-27** donde la variable C se encarga de contar las veces que se repite una muestra de la señal moduladora antes de obtener una muestra nueva desde un búfer circular.

4.6. Efectos de audio con base en retardo de la señal

4.6.1. Eco

La Figura **4-28** representa un sencillo modelo de generación de eco. De forma artificial, el efecto de eco se logra mezclando la señal actual con la misma señal pero retrasada, esta última multiplicada por un factor de atenuación, la ecuación que describe este efecto es la siguiente:

$$y[n] = x[n] + Gx[n - N], \qquad |G| < 1$$

Y su función de transferencia es:

$$H(z) = 1 + Gz^{-N}$$

El parámetro N indica la cantidad de muestras de retraso, esto representa el tiempo de reflexión; el parámetro G es la ganancia de la señal reflejada, la cual debe ser siempre menor a 1. La Figura **4-28** da como resultado una única reflexión de la señal, para dar mayor realismo al eco se debe agregar más reflexiones las cuales continúen atenuándose hasta desaparecer, para este fin, al sistema se puede agregar más líneas de retardo iguales pero con factores de

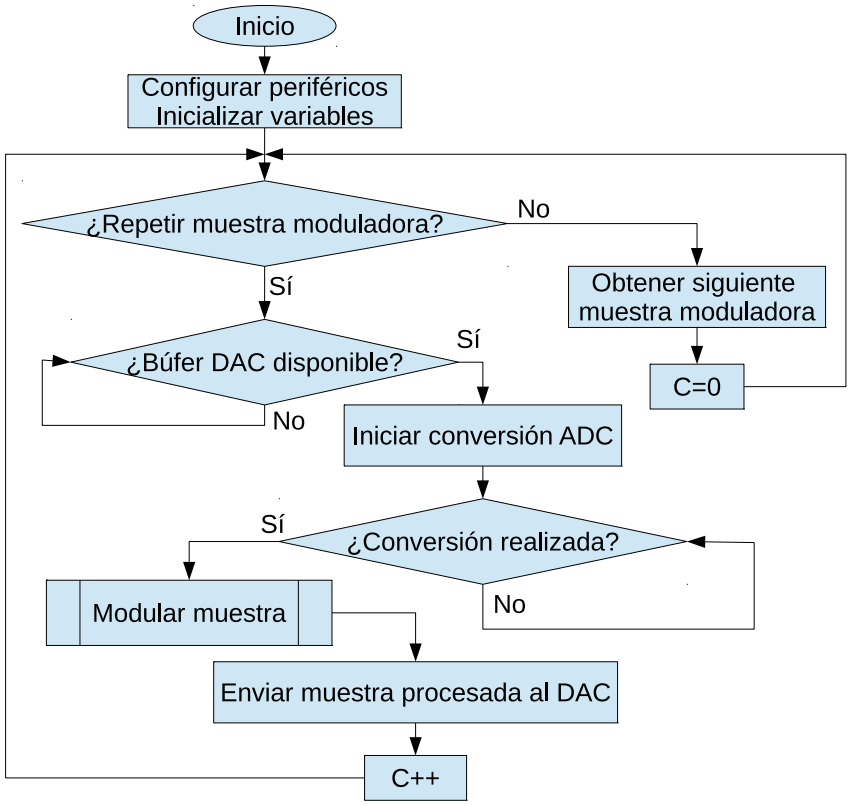

Figura 4-27.: Diagrama de flujo propuesto para aplicar efecto de trémolo.

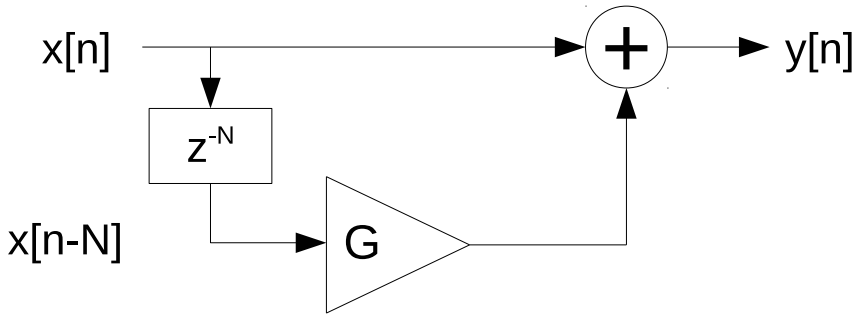

Figura 4-28.: Sistema de eco con una sola reflexión por muestra.

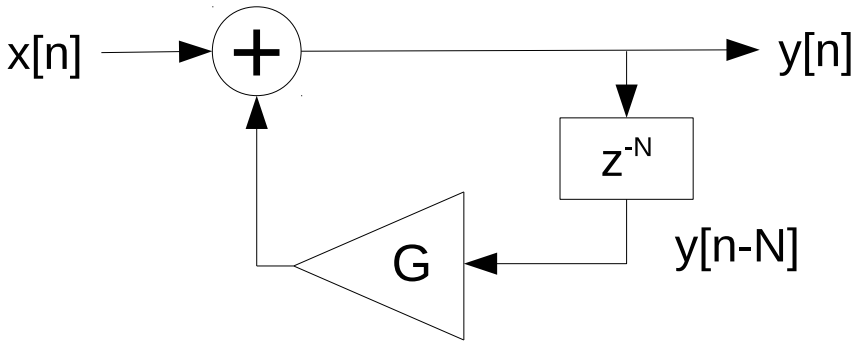

Figura 4-29.: Sistema de eco con múltiples reflexiones por muestra.

atenuación exponenciales al primer factor, de este modo se obtiene más cantidad de reflexiones finitas. Un modo sencillo de implementar un efecto de eco realista es mediante un sistema recurrente, esto se logra conectando la línea de retardo en modo realimentación como en la Figura **4-29** para prolongar infinitamente la cantidad de reflexiones por muestra de sonido. Aunque esto es un sistema de respuesta infinita al impulso, en la práctica la respuesta es finita debido a las características de los sistemas digitales.

A continuación se ilustra una rutina de generación de eco escrita en ensamblador la cual implementa los modelos anteriormente señalados, la diferencia entre ellos está en si el búfer guardará la entrada o la salida, esto se hará desde otra parte del programa. La línea de retardo es más eficiente si se maneja como búfer circular.

```
.include "p33fj128gp802.inc"

.global _Eco
_Eco:
    mov     _MuestraActual,w4;  Parámetros para los registros de trabajo
    mov     _Factor,w5;
    mov     _MuestraRetrasada,w6

    lac     w4,A;
    mac     w5*w6,A
    sac.r   A,w7;       Extrae el resultado del acumulador en W7
    mov     w7,_MuestraConEco;

    return
.end
```

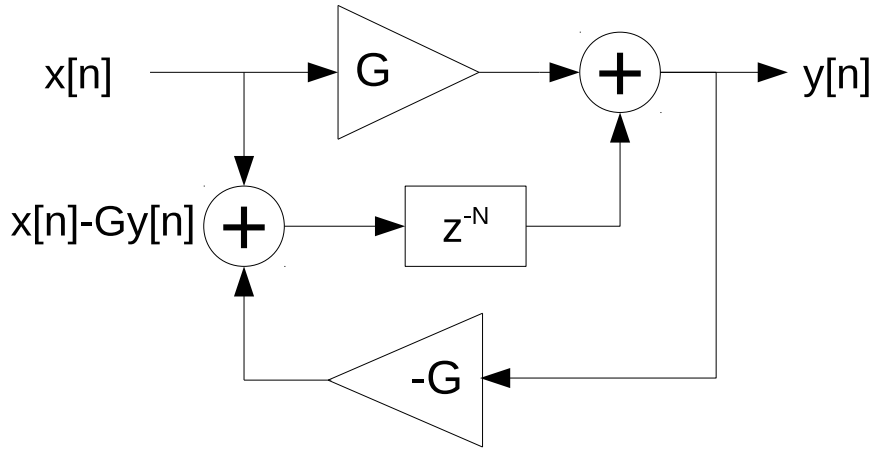

Figura 4-30.: Sistema de reverberación con respuesta en frecuencia plana.

4.6.2. Reverberación

El tiempo de retraso define si la señal se considera eco o reverberación, para el primer caso el tiempo debe ser superior a 50 ms, lo que equivale a unos 8.5 metros de distancia entre la fuente de sonido y el obstáculo que lo refleja. Aunque el mismo sistema de la Figura **4-29** puede utilizarse para generar reverberación, esta no sonará muy natural porque este sistema crea una respuesta en frecuencia de tipo peine. Para generar reverberaciones más realistas se han propuesto modelos [15, Cap. 3] como el de la Figura **4-30**, el cual consiste en un sistema con estructura pasa todas. Su ecuación en diferencias está dada por:

$$y[n] = Gx[n] + x[n-N] - Gy[n-N]$$

Y su función de transferencia es:

$$H(z) = \frac{G + z^{-N}}{1 + Gz^{-N}}, \qquad |G| < 1$$

La siguiente rutina escrita en ensamblador implementa el efecto de reverberación con base en el modelo de la Figura **4-30**:

```
.include "p33fj128gp802.inc"

.global _Reverberacion

_Reverberacion:
    mov     _MuestraRetraso,w4;  Carga los parámetros en los registros W
    mov     _Factor,w5;
    mov     _MuestraEntrada,w6
    mov     _MuestraSalida,w7
```

```
    lac     w4,A;       ; Carga el valor de retraso en B
    mac     w5*w6,A     ; Atenúa la entrada y la resta a B

    lac     w6,B;       ; Carga el valor de la entrada en A
    msc     w5*w7,B     ; Atenúa la salida y la suma a A

    sac.r   B,w4        ; B tiene un nuevo valor que irá a la linea de retraso
    mov     w4,_MuestraRetraso; lo guarda para llevar al búfer circular

    sac.r   A,w7        ; A contiene la nueva salida
    mov     w7,_MuestraSalida; la guarda para utilizar en la siguiente rutina

    return
.end
```

4.6.3. Flanger

Este efecto se obtiene al sumar a la señal original una copia retrasada, atenuada y con un búfer de retardo de tamaño variable. Por lo tanto, el sistema de la Figura **4-28** puede emplearse para este fin haciendo que el búfer que almacena las muestras varíe su tamaño. El siguiente código se encarga de las operaciones para llevar a cabo este efecto:

```
.include "p33fj128gp802.inc"

.global _Flanger

_Flanger:
    mov     _MuestraActual,w4;  Carga los parámetros en los registros de
        trabajo
    mov     _Factor,w5;
    mov     _MuestraRetrasada,w6

    lac     w4,A;
    mac     w5*w6,A
    sac.r   A,w7;       Extrae el resultado del acumulador en W7
    mov     w7,_MuestraConFlanger;

    return
.end
```

En este caso se observa que este código es equivalente al del efecto de eco, los ajustes entonces deben hacerse desde el programa principal.

4.7. Resultados

De este capítulo se han derivado cuatro prácticas de laboratorio, resumidas en la Tabla **4-5**. Cada una de estas prácticas contiene unos ejemplos y cada ejemplo consiste en un proyecto creado con el IDE MPLAB® X.

Tabla 4-5.: Prácticas de laboratorio diseñadas para procesamiento digital de audio en el dominio temporal.

Práctica	Proyecto	Descripción
1. Muestreo y reconstrucción de señales de audio.	Ejemplo_1.1	Tratamiento de las muestras punto a punto.
	Ejemplo_1.2	Tratamiento de las muestras punto a punto con interrupción del DAC.
	Ejemplo_1.3	Tratamiento de las muestras mediante bloques DMA.
	Ejemplo_1.4	Tratamiento de las muestras mediante bloques DMA con interrupción del DAC.
2. Generación de señales periódicas mediante tablas.	Ejemplo_2.1	Señal sinusoidal, señal triangular, señal diente de sierra, señal cuadrada.
	Ejemplo_2.2	Direccionamiento modular.
3. Operaciones de escalamiento de la señal de audio.	Ejemplo_3.1	Escalamiento mediante desplazamientos.
	Ejemplo_3.2	Escalamiento mediante multiplicación.
	Ejemplo_3.3	Modulación de amplitud.
4. Operaciones de retraso de la señal de audio.	Ejemplo_4.1	Eco.
	Ejemplo_4.2	Reverberación.
	Ejemplo_4.3	Flanger.

4.7.1. Validación de las prácticas de muestreo y reconstrucción de señales de audio

Para medir la frecuencia de muestreo del sistema el método empleado ha sido configurar un pin del DSC como salida y enviar un pulso cada vez que el DAC acepta una nueva muestra, este pin se conectó a un osciloscopio para observar y medir esta frecuencia. La Tabla **4-6** [1] resume los resultados de las mediciones de las frecuencias de muestreo que han

[1]Datos calculados para el dsPIC33FJ128GP802 con frecuencia de trabajo de 79,2576MHz

sido implementadas en el dsPIC®, en dicha tabla también se incluye la cantidad máxima de ciclos de máquina disponible para realizar procesamiento muestra a muestra. La Figura **4-11** también sirve como referencia del correcto funcionamiento de los algoritmos de muestreo y reconstrucción.

Tabla 4-6.: Implementación en dsPIC® de las frecuencias de muestreo típicas en el audio digital.

Frecuencia a obtener (Hz)	**Registro DACDIV**	**Frecuencia teórica (Hz)**	**Frecuencia medida (Hz)**	**Núm. máximo de instrucciones** [1]
100000	5	103200	102497	374
48000	12	47631	47332	821
44100	13	44229	43952	885
32000	18	32589	32387	1206
22050	27	22114	21979	1782
16000	38	15877	15779	2485
11025	55	11057	10989	3574
8000	76	8042	7993	4917

[1] Cantidad máxima por muestra sin tener en cuenta el TAD del ADC.

4.7.2. Validación de las prácticas de generación de señales periódicas

Cuatro tipos de señales, cada una con 120 muestras, han sido implementadas en el sistema embebido trabajando con una frecuencia de muestreo teórica de $44229Hz$. Las señales implementadas son de las siguientes formas: sinusoidal, triangular, diente de sierra y cuadrada. Se han diseñado de tal forma que las 120 muestras sean equivalentes a 1 período y que su amplitud no exceda $(-1, 1)$ para el formato $fractional$, con esto, la frecuencia teórica de cada una de ellas está dada por:

$$f_{salida} = \frac{f_s}{N_{muestras}}$$

$$f_{salida} = \frac{44229 \quad Hz}{120}$$

$$f_{salida} \approx 368{,}58 \quad Hz$$

La tabla **4-7** compara la frecuencia medida con la frecuencia esperada.

Tabla 4-7.: Medición de la frecuencia de la señal generada con el dsPIC®.

Frecuencia esperada	**Frecuencia medida**	**Porcentaje de error**
$368{,}58 \quad Hz$	$366{,}40 \quad Hz$	$0{,}59\,\%$

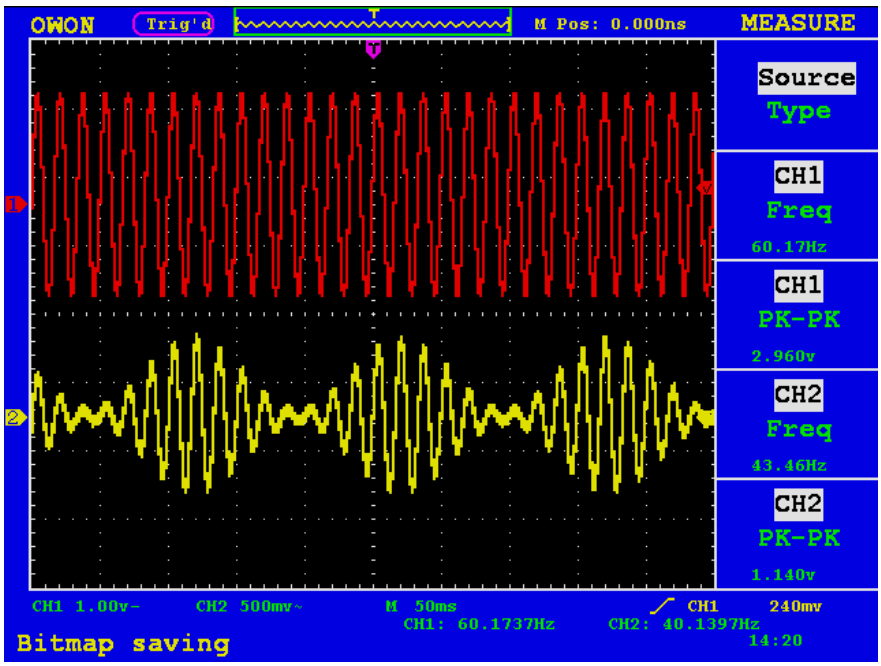

Figura 4-31.: Modulación de amplitud mediante el dsPIC®.

4.7.3. Validación de las prácticas con operaciones de escalamiento de la señal de audio

La Figura **4-31** es una captura tomada del osciloscopio en donde se tiene una señal de entrada de 3 v pk-pk; la salida, que consiste en la modulación de dicha señal, ha sido medida en el pin positivo del DAC1 (DAC1RP). El comportamiento de este procesamiento es suficiente para demostrar que el sistema desarrollado puede realizar cualquier tipo de escalamiento de la señal, incluyendo la modulación AM.

5. Implementación de filtros FIR y filtros IIR para procesamiento digital de audio

La operación más común del procesamiento de señales es el filtrado, el cual consiste en la modificación del espectro de frecuencia de una señal. En el campo del audio, esta modificación es usualmente conocida como ecualización. Los filtros digitales están definidos por ecuaciones en diferencias, las cuales implican repetidas operaciones de multiplicación y acumulación, este tipo de operaciones son llamadas operaciones MAC y la tecnología dsPIC® permite realizar una operación MAC en un solo ciclo de máquina, otra ventaja de la tecnología dsPIC® para este procesamiento es la posibilidad de manejar direccionamiento modular para la implementación de estos sistemas.

5.1. Introducción a los filtros digitales

Los filtros son la forma más común del procesamiento de señales, se usan para modificar el espectro de una señal, pudiendo alterar su magnitud o su fase en las frecuencias deseadas, con esto se consigue mejorar dicha señal según el objetivo buscado. Los filtros digitales han sido impulsados por el desarrollo de la computación, actualmente los procesadores digitales ofrecen muchas ventajas de los filtros digitales sobre los analógicos.

La siguiente ecuación describe matemáticamente a la mayoría de los filtros digitales:

$$y[n] + \sum_{k=0}^{N-1} a_k y[n-k] = \sum_{k=0}^{N-1} b_k x[n-k]$$

Donde a_k y b_k son los coeficientes del filtro, éstos ponderan los valores tomados de la salida y de la entrada de la señal respectivamente. Los filtros expresados con la anterior ecuación son llamados filtros IIR (Infinite Input Response) porque al aplicar un impulso unitario al sistema éste tendrá una respuesta infinita debido a la realimentación.

Si en la anterior ecuación los valores de a_k valen cero entonces la ecuación se reduce a la siguiente:

$$y[n] = \sum_{k=0}^{M-1} b_k x[n-k]$$

Esta ecuación representa un filtro que no depende de la salida del sistema y es llamado filtro FIR (Finite Input Response) debido a que al aplicar un impulso unitario al sistema la respuesta tendrá una longitud finita, la cual coincide con los coeficientes.

Los coeficientes de un filtro digital definen el comportamiento del sistema, si se quiere modificar estos coeficientes bastará un simple cambio de su valor guardado en memoria, esto permite que el rediseño de un filtro digital sea muy sencillo y sólo requiere cambios en software: cualquier cambio en los coeficientes cambia la función de transferencia del filtro.

En un filtro digital se puede cambiar fácilmente los parámetros como la frecuencia de muestreo, frecuencias de corte, rizado, bandas de transición, incluso se puede cambiar la estructura del filtro y todo esto sin alterar el hardware. Estos filtros Son mucho más precisos y confiables que los analógicos y se puede obtener mucha más precisión haciendo cambios en la resolución de los coeficientes o en la cantidad de ellos. Son más económicos para la implementación, la adaptación y el mantenimiento, sus datos pueden ser almacenados y a lo largo del tiempo no sufrirán degradación como ocurriría con los componentes analógicos. Otra ventaja es la posibilidad de implementar sistemas con duración del impulso finita y linealidad de fase, características imposibles de lograr con filtros analógicos. El ancho de banda puede ser incluso

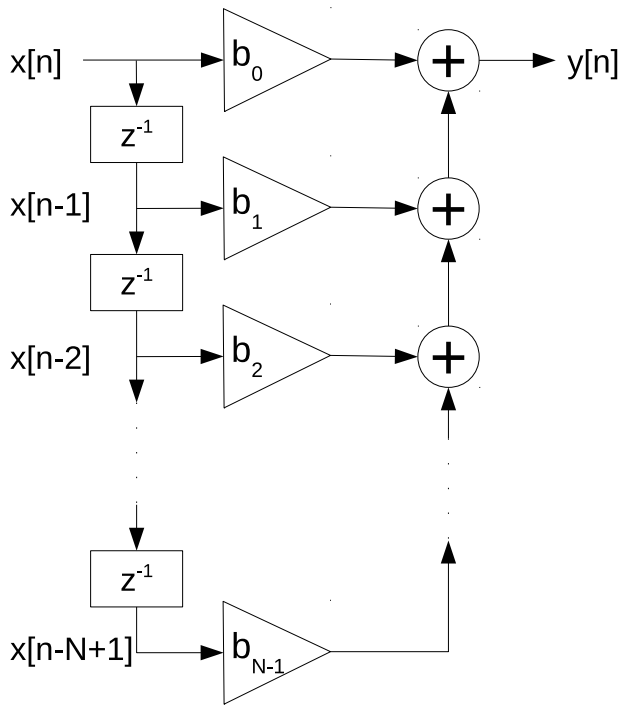

Figura 5-1.: Esquema general de un sistema de filtro FIR.

menos de 5 Hz, muy útil en señales biomédicas o sísmicas.

Las principales desventajas están en los efectos de la cuantización producidos por el tipo de aritmética de los sistemas digitales, pues el número de bits para almacenar los números es finito y esto produce no-linealidades lo que conlleva a ruidos inherentes en estos sistemas. Los coeficientes cuantificados desvían el comportamiento real del sistema del comportamiento ideal con coeficientes teóricos. Otra desventaja de los filtros digitales es que aún no son eficientes para el uso en altas frecuencias.

Un único sistema DSP puede implementar muchos filtros analógicos equivalentes.

5.2. Diseño de filtros FIR

Un sistema típico de filtro FIR es como el mostrado en la Figura **5-1**, esta estructura es llamada estructura directa porque es la implementación inmediata de la ecuación en diferencias que describe al filtro.

Para diseñar un filtro FIR hay básicamente tres métodos:

- Método de las ventanas
- Muestreo en frecuencia
- Rizado constante

El método más difundido es el método de las ventanas. Existen diversos tipos de ventanas que han sido desarrolladas para la implementación de filtros FIR, son dos los parámetros más importantes para analizar una ventana en el dominio frecuencial:

- Ancho del lóbulo principal: a mayor anchura del lóbulo mayor será la banda de transición, esta característica se controla con el parámetro M. Si M es mayor, será menor la banda de transición pero el filtro tendrá mayor complejidad.
- Atenuación del primer lóbulo secundario: la relación entre la amplitud el lóbulo principal y el primer lóbulo secundario permite estimar la razón de rizado.

La ventana rectangular es la más sencilla pero la menos efectiva en atenuación, la ventana triangular trata de minimizar el fenómeno de Gibbs causado por la naturaleza discontinua de la ventana rectangular, consecuentemente han sido creadas más ventanas con sus respectivas características.

Los filtros FIR son siempre estables y se puede comprobar fácilmente analizando la transformada z de su función de transferencia, pues como se puede observar se trata de un sistema sin polos:

$$H\left[z\right] = \sum_{k=0}^{M-1} b_k z^{-k}$$

Otra característica de los filtros FIR es que estos son los únicos con capacidad de tener una respuesta lineal en fase, lo cual es imposible con un sistema analógico. Una gran desventaja es que normalmente se necesita un orden muy alto comparado con los filtros IIR.

El método más común para el diseño de filtros FIR es conocido como método de las ventanas, otro método para el mismo fin es el muestreo en frecuencia, pero este último tiene una baja atenuación de la banda de rechazo.

Para diseñar un filtro digital, se hace necesario conocer las especificaciones del filtro, debe tenerse en cuenta que la frecuencia está normalizada en radianes y no en Hertz, siendo el valor de πrad la máxima frecuencia.

En la Figura **5-2** se tiene una gráfica que sirve como base para especificar un filtro pasa bajas. En el eje de las abscisas, el cual corresponde a la frecuencia, se puede identificar tres intervalos importantes:

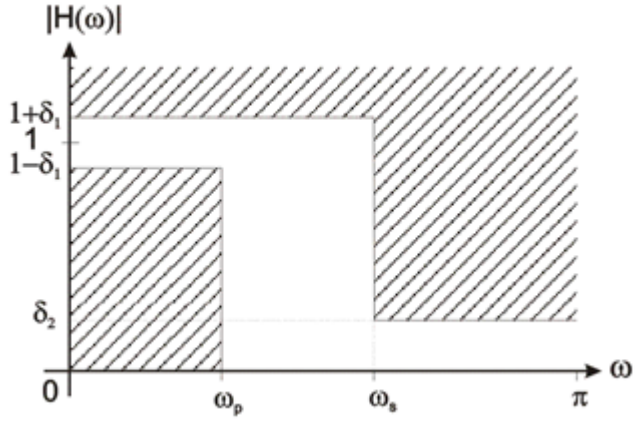

Figura 5-2.: Especificaciones de un filtro pasa bajas digital [30].

- Banda de paso $(0, w_p)$
- Banda de transición (w_p, w_s)
- Banda de atenuación (w_s, π)

Donde:

- w_p: frecuencia de corte normalizada en la banda de paso
- w_s: frecuencia de corte normalizada en la banda de rechazo

En el eje de las ordenadas, correspondiente a la amplitud, tenemos dos parámetros clave:

- δ_1: rizado máximo en la banda de paso
- δ_2: rizado máximo en la banda de rechazo

Las especificaciones de un filtro pasa altas son las mismas que el del filtro pasa bajas pero intercambiando la banda de paso con la de rechazo como se observa en la Figura **5-3**.
De forma similar, son las especificaciones de los filtros pasa banda (Figura **5-4**) y rechaza banda (Figura **5-5**), pero en este caso tenemos dos frecuencias de corte para banda de paso y dos frecuencias de corte para banda de rechazo: w_{p1}, w_{p2}, w_{s1}, w_{s2}.

Para utilizar el método de diseño con ventanas el primer paso es asumir que se diseñará un filtro ideal, el cual debe tener una respuesta como cualquiera mostrada en la Figura **5-6**.

Utilizando la transformada inversa de Fourier se obtiene la Tabla **5-1** con los valores de los 4 tipos típicos de filtros ideales.

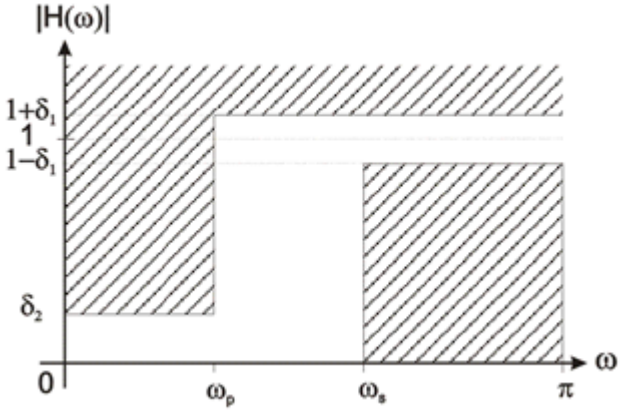

Figura 5-3.: Especificaciones de un filtro pasa altas digital [30].

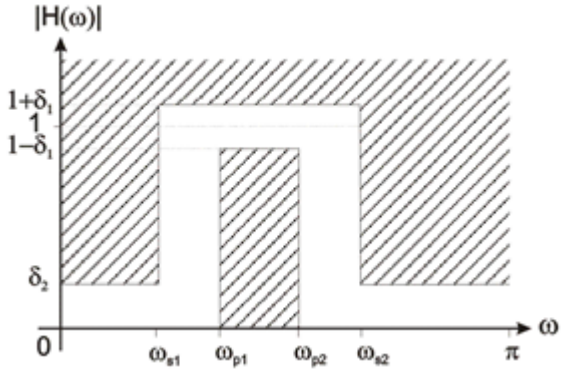

Figura 5-4.: Especificaciones de un filtro pasa banda digital [30].

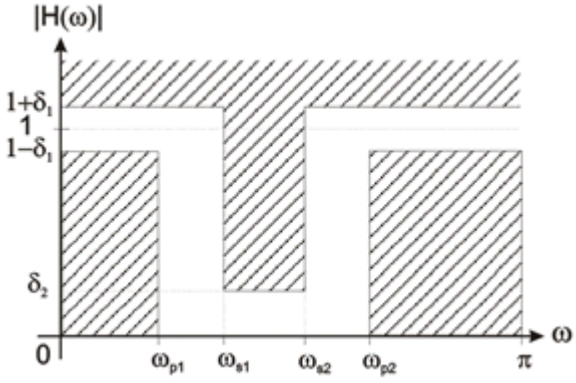

Figura 5-5.: Especificaciones de un filtro rechaza banda digital [30].

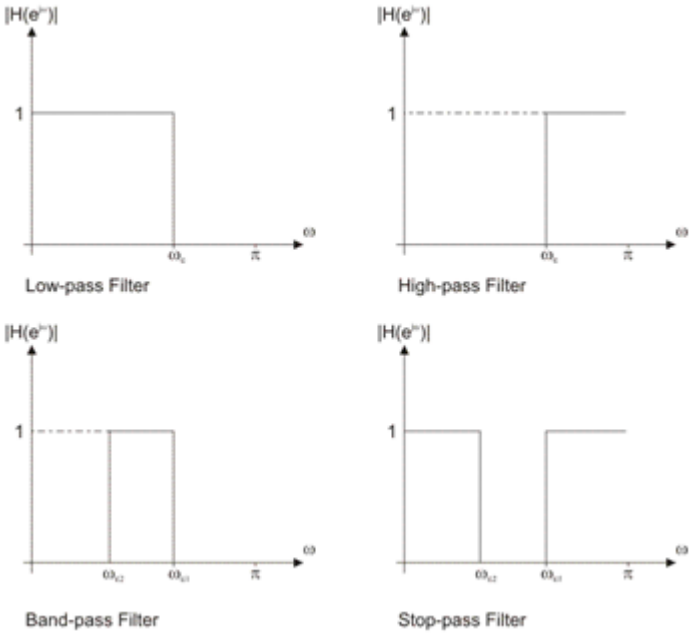

Figura 5-6.: Respuestas ideales de los filtros.

Tabla 5-1.: Fórmulas de la Transformada Inversa de Fourier de los filtros ideales.

Tipo de filtro	**Respuesta en frecuencia hd[n]**
Pasa bajas	$h_d[n] = \begin{cases} \frac{sen[w_c(n-M)]}{\pi(n-M)}; & n \neq M \\ \frac{w_c}{\pi}; & n = M \end{cases}$
Pasa altas	$h_d[n] = \begin{cases} 1 - \frac{w_c}{\pi}; & n \neq M \\ -\frac{sen[w_c(n-M)]}{\pi(n-M)}; & n = M \end{cases}$
Pasa banda	$h_d[n] = \begin{cases} \frac{sen[w_{c2}(n-M)]}{\pi(n-M)} - \frac{sen[w_{c1}(n-M)]}{\pi(n-M)}; & n \neq M \\ \frac{w_{c2}-w_{c1}}{\pi}; & n = M \end{cases}$
Rechaza banda	$h_d[n] = \begin{cases} \frac{sen[w_{c1}(n-M)]}{\pi(n-M)} - \frac{sen[w_{c2}(n-M)]}{\pi(n-M)}; & n \neq M \\ 1 - \frac{w_{c2}-w_{c1}}{\pi}; & n = M \end{cases}$

Donde N es el orden el filtro y M es una constante equivalente a $N/2$, de aquí se deduce que si el filtro es de orden par, M será un número entero y la respuesta en frecuencia será simétrica alrededor del valor M.

Las ventanas son funciones preestablecidas que se aplican a las funciones de transferencia de los filtros para suavizar su salida, la ventana más sencilla de implementar es la de tipo rectangular, pero debido a sus características la señal de salida sufre una distorsión conside-

rable, por lo tanto a lo largo del tiempo se han desarrollado otro tipo de ventanas como la triangular y las que buscan eliminar las discontinuidades abruptas, entre ellas, las ventanas más conocidas son: Hamming, Blackman, Hanning.

Con la información teórica aquí consignada ya es posible el diseño de un filtro FIR, el siguiente ejemplo muestra el diseño de un filtro FIR pasa bajas utilizando la ventana rectangular y las siguientes especificaciones:

- Orden del filtro: $N = 10$
- Frecuencia de muestreo: $44100Hz$
- Frecuencia de corte: $4{,}41kHz$

El número de coeficientes es $N + 1 = 11$. En la función de la ventana rectangular todos los coeficientes tienen valor 1, quedando dicha función así:

$$w\left[n\right] = 1; \quad 0 \leq n \leq 10$$

Es necesario tener presente la ecuación para el filtro pasa bajas ideal de la Tabla **5-1**. Recordando que M equivale a $N/2$ se tiene:

$$M = \frac{N}{2}$$
$$M = \frac{10}{2}$$
$$M = 5$$

La frecuencia de corte debe ser normalizada con la siguiente expresión:

$$w_c = \frac{2\pi f_c}{f_s} = \frac{2\pi\left(4410\right)}{44100}$$

$$w_c = 0{,}2\pi$$

Tanto M como w_c se reemplazan en $h_d[n]$, los coeficientes del filtro son obtenidos multiplicando la función del filtro ideal por la función de la ventana:

$$h\left[n\right] = w\left[n\right] \cdot h_d\left[n\right]; \qquad 0 \leq n \leq 10$$

La Tabla **5-2** resume los resultados, en la columna fractional se ha escrito la conversión de los coeficientes al formato que maneja el dsPIC®.

Tabla 5-2.: Coeficientes calculados para un filtro FIR.

n	**h[n]**	$x_2{}^{15}$	**Fractional**
0	0	0	0x0000
1	0.04677446419	1533	0x05FD
2	0.1009102305	3307	0x0CEB
3	0.1513653457	4960	0x1360
4	0.1870978568	6131	0x17F3
5 (M)	0.2	6554	0x199A
6	0.1870978568	6131	0x17F3
7	0.1513653457	4960	0x1360
8	0.1009102305	3307	0x0CEB
9	0.04677446419	1533	0x05FD
10	0	0	0x0000

El mismo método puede ser utilizado para el cálculo de los demás tipos de filtro, utilizando la respectiva ecuación de filtro ideal, calculando los coeficientes de la ventana que se va a utilizar, multiplicando estas dos funciones y finalmente reemplazando los valores de M y w_c se obtiene la fórmula que depende de n, con la cual finalmente se puede calcular cada coeficiente del filtro. Para facilitar el diseño de filtros existen diversos programas de cómputo, uno de ellos es el dsPIC® FD Lite, el cual consiste en un asistente para el cálculo de coeficientes de filtros dados ciertos parámetros, su versión de prueba tiene algunas limitaciones pero es suficiente para el enfoque educativo.

5.3. Diseño de filtros IIR

Un sistema de filtro IIR en estructura directa es como el mostrado en la Figura **5-7**, aunque la implementación de esta estructura permite un procesamiento rápido, es más común la implementación de la estructura mostrada en la Figura **5-8**, la cual es llamada *estructura Traspuesta de tipo II*, su particularidad es que funciona muy bien cuando los polos están cercanos a la circunferencia unidad.

El diseño de un filtro IIR no es tan intuitivo como el de un filtro FIR, uno de los métodos más utilizados para realizar este diseño se basa en tomar como referencia un filtro analógico y mediante su función de transferencia se hace una conversión hacia el dominio Z; esto es posible mediante una operación llamada *Transformada Bilineal*.

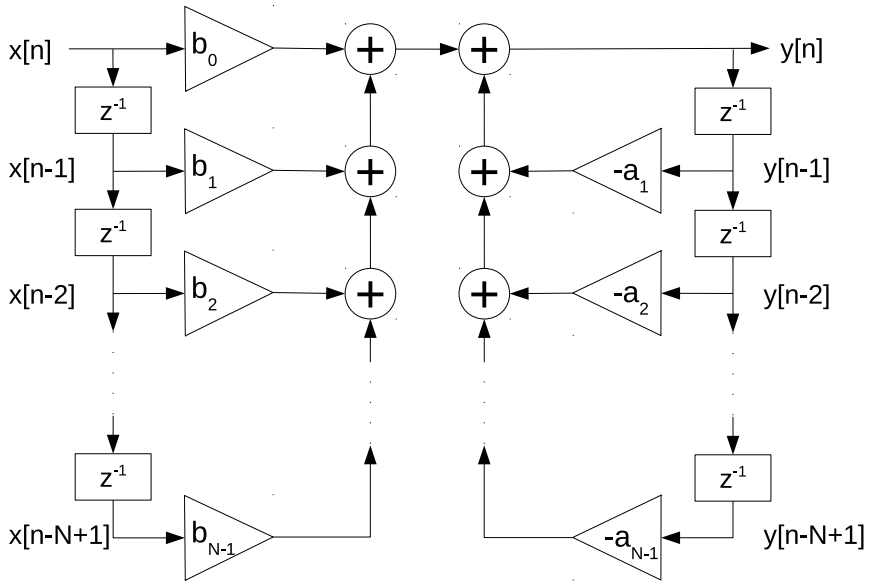

Figura 5-7.: Esquema general de un sistema de filtro IIR con estructura Directa tipo I.

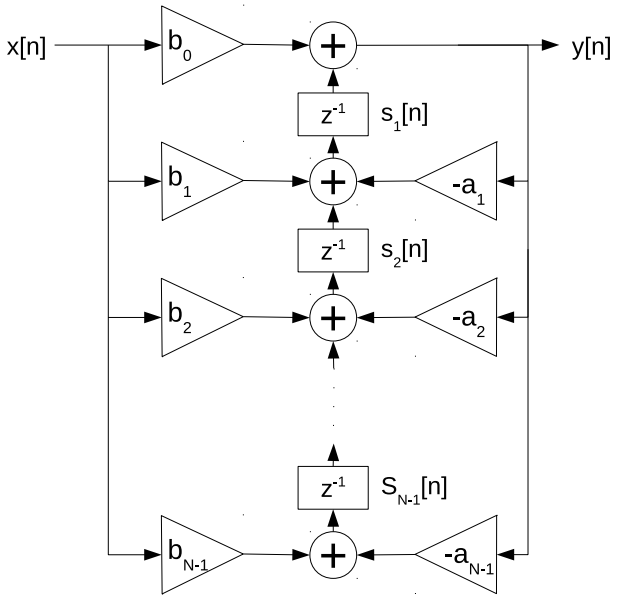

Figura 5-8.: Esquema general de un sistema de filtro IIR con estructura Traspuesta tipo II.

5.4. Biblioteca DSP del compilador XC16

Es una biblioteca desarrollada por Microchip para permitir el uso de operaciones DSP mediante el lenguaje de programación C. El objetivo de esta herramienta es que el uso del lenguaje de programación sea lo más eficiente al implementar las funciones más comunes del procesamiento digital de señales, dicha biblioteca abarca en total 52 funciones.

Para el uso de esta biblioteca se necesitan 2 archivos: el archivo de cabecera **dsp.h** y el archivo **libdsp-omf.a**, el cual contiene todos los objetos individuales de cada función. El primer archivo es usado por el compilador y el segundo por el enlazador.

Para incrementar las bondades de las instrucciones DSP, el tipo de datos preferido para trabajar este tipo de funciones es el *fractional*, este tipo de dato se caracteriza por tener el bit más significativo como indicador de signo, y los 15 bits restantes son fraccionarios. Este formato es conocido como Q1.15 (Representación binaria con punto fijo y magnitud con signo). Básicamente, esto representa números desde $-1{,}00$ hasta 1,00) Cuando en las funciones hay multiplicaciones, los cálculos se hacen usando el acumulador de 40 bits y el formato utilizado es el 9.31. Esto es: 31 bits fraccionarios, 8 bits de magnitud entera y un bit de signo. Después de utilizar el acumulador, el resultado es revertido a formato Q1.15 mediante redondeo.

Un punto importante es que en las operaciones de esta biblioteca no se hace uso de la memoria RAM, por lo que los valores que se quieran asignar a variables deben hacerse manualmente por el programador, malos usos de la memoria RAM pueden provocar resultados inesperados, por lo que hay que ser muy cuidadosos en este punto. Para la memoria RAM se recomienda el uso de operandos en los espacios x e y para lograr mejor rendimiento. Muchas funciones DSP cambian el modo de operación del dsPIC® a través del registro CORCON (Control del núcleo), regresando a su estado anterior después de la ejecución de dicha función.

Para hacer filtrado con dsPIC® se deben tener organizados los datos como un vector *fractional* (fraccionario), el hardware de los dsPIC® está optimizado para la realización de operaciones matemáticas discretas que en principio son ecuaciones en diferencias.

En cuanto a filtrado, la biblioteca DSP cuenta con 10 funciones diseñadas para el tratamiento con filtros FIR y 5 para el tratamiento con filtros IIR:

- FIRStruct
- FIR
- FIRDecimate

- FIRDelayInit
- FIRInterpolate
- FIRInterpDelayInit
- FIRLattice
- FIRLMS
- FIRLMSNorm
- FIRStructInit
- IIRCanonic
- IIRCanonicInit
- IIRLattice
- IIRLatticeInit
- IIRTransposed

5.5. Diseño de un oscilador digital para dsPIC®

A continuación se muestra el código para desarrollar un oscilador digital, será diseñado para una frecuencia de audio de $880Hz$ pero puede ser fácilmente ajustado a otro valor de frecuencia cambiando el valor de un solo coeficiente. Este tipo de oscilador se basa en las propiedades de estabilidad de los filtros IIR, cuando el diagrama de polos está dentro de la circunferencia unidad el sistema IIR se comporta como un filtro, si los polos están por fuera son un sistema inestable, pero si los polos se ubican en la circunferencia obtendremos un sistema oscilatorio.

La ecuación en diferencias de este sistema es la siguiente:

$$y[n] = -a_1 y[n-1] - y[n-2]$$

En donde el factor a_1 es el único parámetro que variará la frecuencia, los requerimientos computacionales son: 2 espacios de memoria para los coeficientes, dos espacios de memoria para las señales retrasadas y un espacio de memoria para la señal actual, así:

```
// Coeficientes del filtro IIR de segundo orden...
fractional a1 = 0x7EFF;// Este valor representa 0.992 (para obtener 880 Hz con
    fs = 44100)
fractional a2 = 0xC000; // Este valor representa -0.5

// Condiciones iniciales del filtro IIR...
fractional y = 0;       // Salida del sistema
fractional z1 = 0x0000; // Valor retrasado 1 muestra
fractional z2 = 0x0A46;//0x0A46; // Valor retrasado 2 muestras
```

Se empezará analizando las condiciones iniciales, tanto los valores pasados como el actual deben ser inicializados en cero pero en este caso es un oscilador que requiere una excitación inicial, por eso el valor de $z2$ no es cero, puede ser usado cualquier valor arbitrario en este caso.

El cálculo de los coeficientes se hizo asumiendo la frecuencia de muestreo del DAC en $44100Hz$ que es una frecuencia típica de audio. Las siguientes fórmulas permiten calcular el valor de $a1$:

$$a_1 = -2\cos(w_0)$$

$$w_0 = 2\pi\frac{f_{deseada}}{f_{muestreo}}$$

Reemplazando se tiene:

$$a_1 = -2\cos\left(2\pi\frac{f_{deseada}}{f_{muestreo}}\right)$$

$$a_1 = -2\cos\left(2\pi\frac{880}{44100}\right)$$

$$a_1 = -1{,}984$$

La ecuación en diferencias queda entonces:

$$y[n] = 1{,}984y[n-1] - y[n-2]$$

Como el valor de este coeficiente es mayor al rango del formato $fractional$, es necesario escalar el valor del coeficiente, en este caso lo más sencillo es dividir entre 2 con lo cual se afecta también al otro coeficiente. La ecuación en diferencias escalada queda:

$$\frac{y[n]}{2} = 0{,}992y[n-1] - 0{,}5y[n-2]$$

Los dos coeficientes deben ser convertidos a formato *fraccional* indicado con números hexadecimales, esto se hace multiplicando cada coeficiente por 2^15.

$$0{,}992 \times 2^{15} = 32506 = 0X7EFA$$

$$-0{,}5 \times 2^{15} = -16384 = 0XC000$$

Con estos datos ya se puede implementar el programa:

```
extern void Procesamiento(void);

int main (void)
{
    ConfigOscilador();
    ConfigPuertos();
    ConfigDAC();

    while (1)              // Bucle infinito
    {
        LED1 = APAGAR;  // Este LED muestra mediante pulsos la frecuencia de
            muestreo real
        while(DAC1STATbits.REMPTY != 1);// Espera a que el búfer del DAC quede
             vacío
        LED1 = ENCENDER;

        DAC1RDAT = y;         // Actualiza la muestra anteriormente procesada

        Procesamiento();

    }

    return (EXIT_SUCCESS);
}
```

Como se observa, ha sido definida una función externa, esta función llamará a una rutina en ensamblador que se encargará del procesamiento, o sea, de resolver la ecuación en diferencias. A continuación se muestra dicha rutina escrita en ensamblador:

```
.include "p33fj128gp802.inc"

.global _Procesamiento

_Procesamiento:

    mov     _z1,w5
    mov     _a1,w6
    mpy     w5*w6,a
```

```
    sac.r    a,w10

    mov      _z2,w5
    mov      _a2,w6
    MPY      W5*W6, A
    sac.r    a,w11

    add      w10,w11,w4
    mul.su   w4,#2,w6

    mov      _z1,w0
    mov      w0,_z2
    mov      w6,_z1
    mov      w6, _y

    return
.end
```

Nótese que en la operación *mul.su* hay una multiplicación por dos, esto se debe hacer debido al escalamiento que se hizo con los coeficientes anteriormente. De esta forma ha quedado desarrollado el oscilador.

5.6. Resultados

Como resultado de este capítulo se han diseñado dos prácticas de laboratorio, resumidas en la Tabla **5-3**. Cada una de estas prácticas contiene tres ejemplos para los tres tipos de filtrado más comunes.

Tabla 5-3.: Prácticas de laboratorio diseñadas para filtrado digital de señales de audio.

Práctica	**Proyecto**	**Descripción**
5. Ecualización con filtros FIR.	Ejemplo_5.1	Filtro FIR pasa bajas.
	Ejemplo_5.2	Filtro FIR pasa altas.
	Ejemplo_5.3	Filtro FIR pasa banda.
6. Ecualización con filtros IIR.	Ejemplo_6.1	Filtro IIR pasa bajas.
	Ejemplo_6.2	Filtro IIR pasa altas.
	Ejemplo_6.3	Filtro IIR pasa banda.

Para validar los filtros implementados se ha usado el montaje de laboratorio observado en la Figura **5-9**.

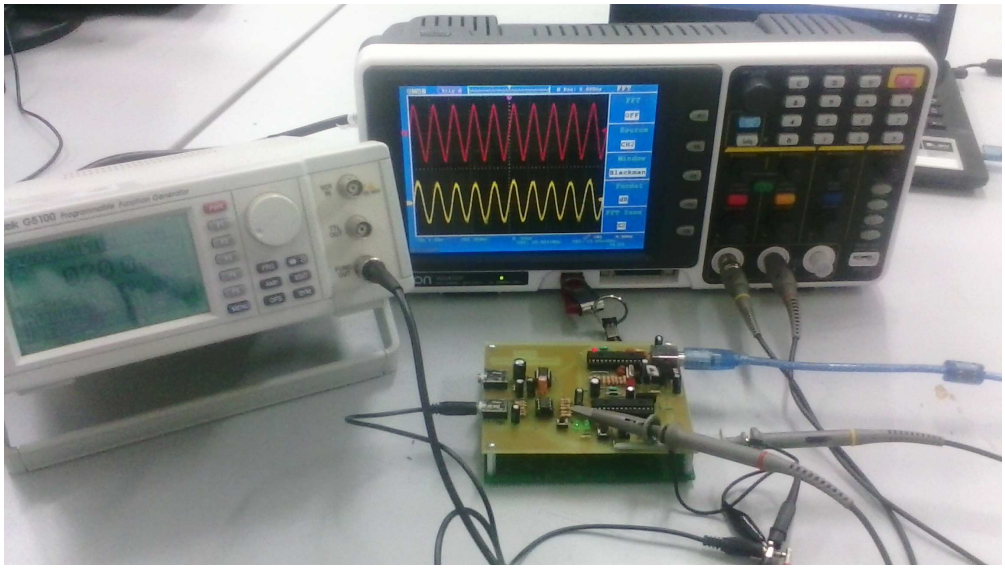

Figura 5-9.: Montaje de laboratorio para validación de filtros.

5.6.1. Validación de los filtros

Mediante el software dsPICfdLite se han calculado los coeficientes para un filtro FIR pasa bajas con las características de la Tabla **5-4**. Posteriormente han sido implementados en la tarjeta desarrollada y los resultados medidos[1] se muestran en la Figura **5-10**.

Tabla 5-4.: Especificaciones del filtro validado.

Especificación	**valor**
Frecuencia de muestreo	44229 Hz
Banda de paso	(0,1000) Hz
Banda de transición	(1000,2000) Hz
Banda de rechazo	(2000,22114.5) Hz
Rizado de la banda de paso	1 dB
Rizado de la banda de rechazo	20 dB

[1]Las medidas fueron tomadas en el pin DAC1RP del dsPIC®

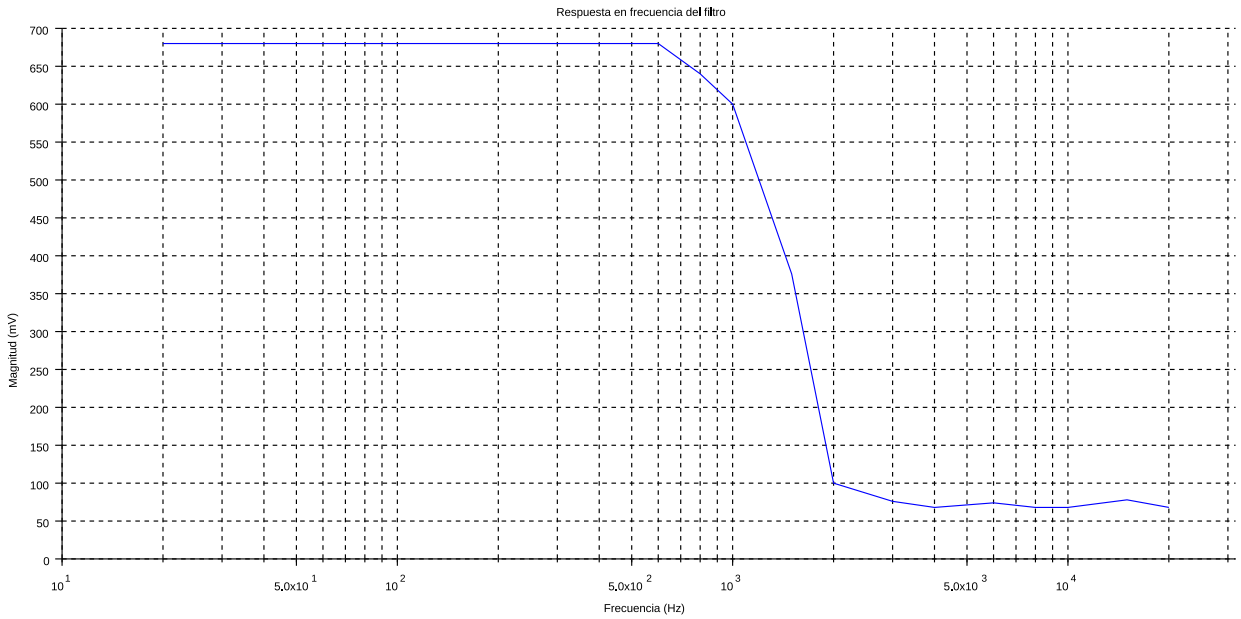

Figura 5-10.: Gráfica de la tabulación de mediciones.

6. Implementación en dsPIC® de la Transformada Discreta de Fourier

La Transformada Discreta de Fourier es una operación matemática que cambia la manera de representar las señales, esta transformada tiene como resultado un vector de sinusoidales complejas las cuales representan el espectro de la señal, en este caso, se dice que la señal está representada en el dominio de la frecuencia. Esta nueva forma de ver a las señales es muy poderosa puesto que al realizar análisis en este dominio se puede obtener más información relevante. Una de las múltiples aplicaciones de esta transformada es el reconocimiento de la componente frecuencial con más energía de una señal, o sea, la frecuencia de mayor magnitud. Para la implementación de esta operación en dispositivos embebidos como el dsPIC® se ha sugerido el algoritmo FFT (Fast Fourier Transform) para optimizar el cálculo.

6.1. Introducción a la Transformada Discreta de Fourier

La transformada de Fourier de una señal en tiempo continuo $x(t)$ está definida como:

$$X(w) = \int_{-\infty}^{\infty} x(t)\, e^{-jwt} dt, \qquad w \in (-\infty, \infty)$$

La Transformada Discreta de Fourier (DFT) reemplaza la integral infinita por una suma finita, tanto los valores en tiempo como en frecuencia son discretos:

$$X(w_k) = \sum_{n=0}^{N-1} x(t_n)\, e^{-jw_k t_n}, \qquad k = 0, 1, 2, ..., N-1$$

Donde:

$$t_n = nT$$

$$w_k = k\Omega$$

$$\Omega = \frac{2\pi}{NT}$$

La transformada inversa discreta de Fourier es:

$$x(t_n) = \frac{1}{N} \sum_{k=0}^{N-1} X(w_k)\, e^{jw_k t_n}, \qquad n = 0, 1, 2, ..., N-1$$

Es posible obtener formas más puras de representar la transformada y su inversa reemplazando los valores de frecuencia y tiempo, teniendo en cuenta que T no es relevante para el cálculo respectivo:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{\frac{-j2\pi nk}{N}}, \qquad k = 0, 1, 2, ..., N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{\frac{j2\pi nk}{N}}, \qquad n = 0, 1, 2, ..., N-1$$

Estas dos últimas ecuaciones están enfocadas en el cálculo a través de procesadores digitales, analizando su estructura se puede ver que cada una de estas ecuaciones tiene dos sumatorias, una dedicada a recorrer los valores de n y otra dedicada a recorrer los valores de k, esto significa que para obtener la transformada de una señal con N muestras, se requerirá N^2.multiplicaciones complejas y $(N-1)^2$ sumas complejas.

Para calcular la Transformada Discreta de Fourier de una señal en tiempo real habrá que almacenar en un espacio de memoria una cantidad limitada de muestras de esa señal, por lo que el procesamiento se hará por bloques de muestras, la consecuencia de esto es que

la resolución de frecuencia obtenida de la transformada estará afectada directamente por la cantidad de muestras por bloque, además de la frecuencia de muestreo. La siguiente ecuación expresa la resolución de frecuencia de dicha transformada:

$$f_{res} = \frac{f_s}{N}$$

Donde f_s es la frecuencia de muestreo y N es la cantidad de muestras almacenadas en el búfer. En la práctica, los valores típicos de N son 64, 128, 256 y 512.

Como la cantidad de cálculos para realizar esta transformada aumenta cuadráticamente respecto al tamaño del búfer, los valores de N muy grandes pueden tener problemas para implementarse en dispositivos como los dsPIC®, es por eso que lo más sensato es acudir a algoritmos que simplifican estos cálculos, como los que se comentarán en la siguiente sección.

6.2. La Transformada Rápida de Fourier (FFT)

La Transformada Rápida de Fourier (FFT por sus siglas en inglés) es el nombre que recibe un conjunto de algoritmos que buscan disminuir la cantidad de cálculos necesarios para obtener la DFT, básicamente el objetivo es disminuir de alguna manera el valor de N generando bloques más pequeños que luego se puedan combinar para obtener la respuesta correcta. El algoritmo más extendido ha sido el de *Cooley-Tukey*, el cual tiene dos variantes: diezmado en tiempo y diezmado en frecuencia, los cuales pueden ser en base 2 o base 4.

El algoritmo de diezmado en tiempo en base 2 consiste en dividir el bloque de tamaño N en dos sub bloques de tamaño $N/2$, esto ya nos dice que N debe ser par. Uno de los bloques contiene las muestras pares y el otro las muestras impares, teniendo en cuenta que las muestras pares están definidas como $x(2n)$ y las muestras impares como $x(2n+1)$, la DFT puede entonces representarse como:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, e^{\frac{-j2\pi(2n)k}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\, e^{\frac{-j2\pi(2n+1)k}{N}}$$

El exponencial del segundo término puede desarrollarse más para obtener:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, e^{\frac{-j2\pi(2n)k}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\, e^{\frac{-j2\pi(2n)k}{N}} e^{\frac{-j2\pi k}{N}}$$

Reorganizando se obtiene:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{par}(n)\, e^{\frac{-j2\pi nk}{N/2}} + e^{\frac{-j2\pi k}{N}} \sum_{n=0}^{\frac{N}{2}-1} x_{impar}(n)\, e^{\frac{-j2\pi nk}{N/2}}$$

Nótese que el primer término es la DFT del bloque de las muestras pares, y el segundo término es la DFT del bloque de las muestras impares pero multiplicadas por un factor exponencial. Más claramente:

$$X(k) = X_{par}(k) + e^{\frac{-j2\pi k}{N}} X_{impar}(k)$$

Los factores generados por este exponencial son llamados factores de giro (twiddle factors).

Teniendo en cuenta que el período de esta representación es $N/2$, y que por propiedades $k = k + T$, se puede reemplazar $k = k + N/2$ donde sea conveniente sin alterar la operación, así la ecuación puede escribirse de la siguiente manera:

$$X\left(k + \frac{N}{2}\right) = X_{par}(k) + e^{\frac{-j2\pi\left(k+\frac{N}{2}\right)}{N}} X_{impar}(k)$$

6.3. Implementación de la FFT con base en la biblioteca dsp del compilador XC16

En primer lugar se requiere saber cuál será el número de muestras que serán procesadas en cada operación de transformada, o sea, el tamaño del búfer. Si se quiere usar DMA en modo ping-pong el número de muestras por búfer está limitado a 512. Teniendo establecido el tamaño del búfer, usualmente se diseña con un tamaño que sea potencia de 2, se calcula el vector de los factores de giro. En la página de Microchip es posible encontrar archivos con códigos de factores de giro para 64, 128, 256 y 512 muestras.

Para la realización de la FFT es necesario declarar dos vectores, cada uno con tamaño igual al número de muestras, uno de estos vectores almacenará el resultado de la parte real de la transformada y el otro almacenará la parte imaginaria[1]. Esta operación se lleva a cabo a través de la función «FFTComplexIP» la cual tiene como argumentos el logaritmo base 2 del número de muestras, la dirección del vector de salida complejo, la dirección del vector de factores de giro y la página de los factores de giro en caso de que estos estén guardados en la memoria de programa. El vector resultante se irá guardando en las mismas posiciones del vector de entrada.

Es necesario utilizar la función«BitReverseComplex» para reorganizar los datos del vector complejo.

[1]La biblioteca dsp considera que estos dos vectores de formato *fractional* conforman una sola estructura de tipo *fractcomplex*.

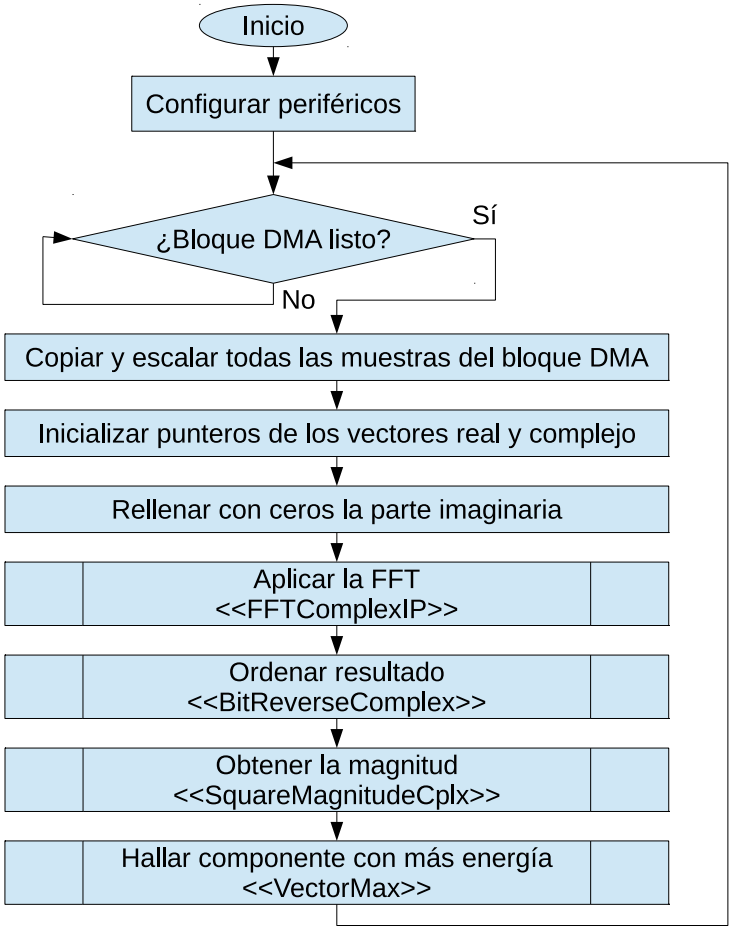

Figura 6-1.: Diagrama de flujo de la implementación de la FFT en dsPIC®.

Para calcular la magnitud de la FFT es necesario calcular la raíz cuadrada de la suma de las muestras equivalentes entre la parte real y la parte imaginaria. La función «SquareMagnitudeCplx» implementa esta operación.

Por último, y como una aplicación de la Transformada de Fourier, puede calcularse la posición del valor máximo en el vector magnitud con la función «VectorMax»; con esta operación es posible determinar cuál es la frecuencia de mayor amplitud que compone la señal a la que se la aplicó la FFT, esta frecuencia generalmente equivale a la frecuencia fundamental de dicha señal. La Figura **6-1** resume todo este proceso.

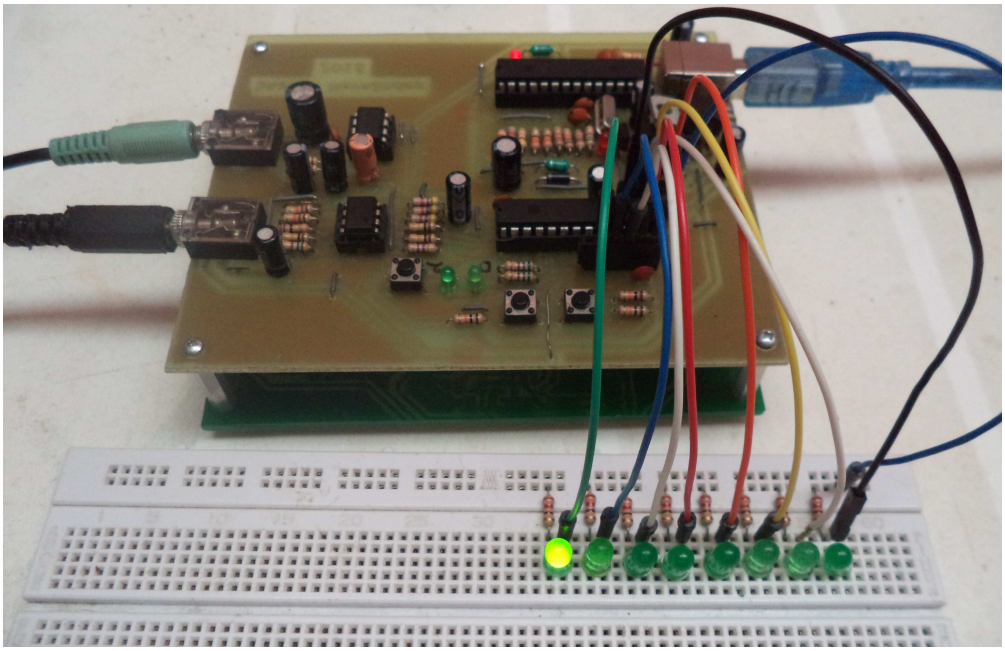

Figura 6-2.: Montaje para validación de la FFT en dsPIC®.

6.4. Resultados

Como resultado de este capítulo se tiene una práctica de laboratorio (Tabla **6-1**), la cual es una aplicación básica de la DFT que sirve para detectar la frecuencia fundamental de una señal de audio; el algoritmo FFT ha sido usado para tal fin.

Tabla 6-1.: Práctica de laboratorio diseñada para aplicaciones de la DFT.

Práctica	**Proyecto**	**Descripción**
7. Transformada Discreta de Fourier.	Ejemplo_7.1	Detección de la frecuencia fundamental en una señal de audio.

Para validar el funcionamiento de la FFT se ha realizado el montaje de la Figura **6-2**, el cual consiste en un arreglo de ocho LED conectados a diferentes pines del dsPIC® que permitirán visualizar si la frecuencia fundamental de una señal se encuentra en las primeras 8 frecuencias discretas dadas por la FFT. La frecuencia de muestreo f_s implementada ha sido de $16kHz$ con un búfer de tamaño $N = 256$ muestras. Con esto se obtendrá una resolución indicada por la siguiente ecuación:

$$f_{res} = \frac{f_s}{N} = \frac{16 \quad kHz}{256}$$

$$f_{res} = 62{,}5Hz$$

Donde f_{res} es la resolución con la que la FFT genera las muestras; con los 8 LED se pudo identificar con éxito frecuencias fundamentales de máximo $500Hz$.

7. Conclusiones y recomendaciones

7.1. Conclusiones

El dsPIC33FJ128GP802 se puede utilizar como dispositivo de procesamiento digital de audio capaz de operar la banda de 20 Hz a 20 kHz Utilizando su ADC y DAC internos se puede desarrollar un sistema donde la tarjeta electrónica es de bajo costo y fácil construcción, lo cual permite que el producto final pueda llegar a más personas y por lo tanto fomenta el aprendizaje de esta tecnología.

Los programas en C de este trabajo fueron desarrollados con el compilador XC16 en su versión estudiantil. Se pudo observar que este compilador, el cual es gratuito, funcionó sin problemas con los algoritmos desarrollados. Esto significa que sin necesidad de compiladores optimizados, en versiones de pago, se puede procesar señales de audio con buenos resultados; además la versión gratuita se puede potenciar significativamente al integrarlo con programación en ensamblador y sin tener códigos fuente complicados.

La memoria RAM del dsPIC® es apta para guardar cierto número de muestras que dependiendo de la frecuencia de muestreo no puede extenderse a grabaciones de más de 1 segundo.

En el circuito desarrollado, el amplificador de audio LM386 requiere de una etapa previa que disminuya la ganancia por lo menos 20 veces para que la salida de audio no salga distorsionada o con ruido.

Las frecuencias fundamentales en una señal de audio por lo general se encuentran en una banda baja, por eso para la búsqueda de frecuencias fundamentales en una señal de audio la FFT es más eficiente si se usa con una frecuencia de muestreo menor.

7.2. Recomendaciones

Con enfoque en futuros trabajos se recomienda el uso de un códec de audio el cual permita prescindir de la parte analógica en el dsPIC® haciendo que este último se encargue sólo de el tratamiento digital, esto también permitirá mejorar la calidad de las muestras de audio pero se debe tener en cuenta que ineludiblemente esto aumentará el costo de los prototipos.

Es recomendable la realización de un circuito para programación mediante *bootloader*.

Si se quiere ampliar la cantidad de muestras de la línea de retardo puede añadirse al sistema una memoria RAM externa en configuración FIFO.

El procesamiento digital de señales de audio es un campo muy amplio y las prácticas se pueden ampliar a otras con mayor complejidad, en la literatura se puede observar que un sólo efecto de audio puede tener muchos modelos diferentes, pueden derivarse más trabajos relacionados con la implementación de modelos no mencionados aquí.

A. Creación de proyectos con MPLAB® X y compilador XC16

La creación de un proyecto con MPLAB® sigue el ciclo de desarrollo mostrado en la Figura **A-1**.

EL primer paso del ciclo de desarrollo es la creación de un proyecto. Para tal fin se ejecuta la aplicación MPLAB® X, para el ejemplo será utilizada la versión 3.30. Luego se debe dar clic en *File-NewProject* como se muestra en la Figura **A-2**.

En la ventana que aparece (Figura **A-3**) se selecciona «Standalone Project» para la creación de un proyecto vacío, después de eso se hace clic en *Next*.

En ventana de la Figura **A-4** se selecciona la familia y el nombre específico del dispositivo para el que se hará el proyecto. Luego clic en *Next*.

La ventana de la Figura **A-5** pregunta el tipo de herramienta con la que se desea programar y/o depurar la aplicación. Por lo general es recomendable seleccionar el simulador, nótese que aunque el simulador es totalmente software está listado en las herramientas de hardware.

En la siguiente ventana (Figura **A-6**) se pregunta qué compilador se usará de la lista de compiladores previamente instalados, en este caso será usado el compilador XC16 versión 1.26.

Para finalizar se debe dar un nombre al proyecto, en este caso se ha llamado «Ejemplo» (**A-7**). Nótese que está seleccionada la opción «Set as main Project». Es importante dejar esta casilla seleccionada.

Después de dar clic en Finish, será creada una carpeta de proyecto, por defecto dentro de la carpeta MPLABXProjects, la cual termina con .X, todas las carpetas que tengan esta terminación serán tratadas por MPLAB® X como carpetas de tipo proyecto. El aspecto de la interfaz queda como se ilustra en la Figura **A-8**.

Al crear un proyecto, se crean también unos archivos y carpetas dentro de la ruta especifi-

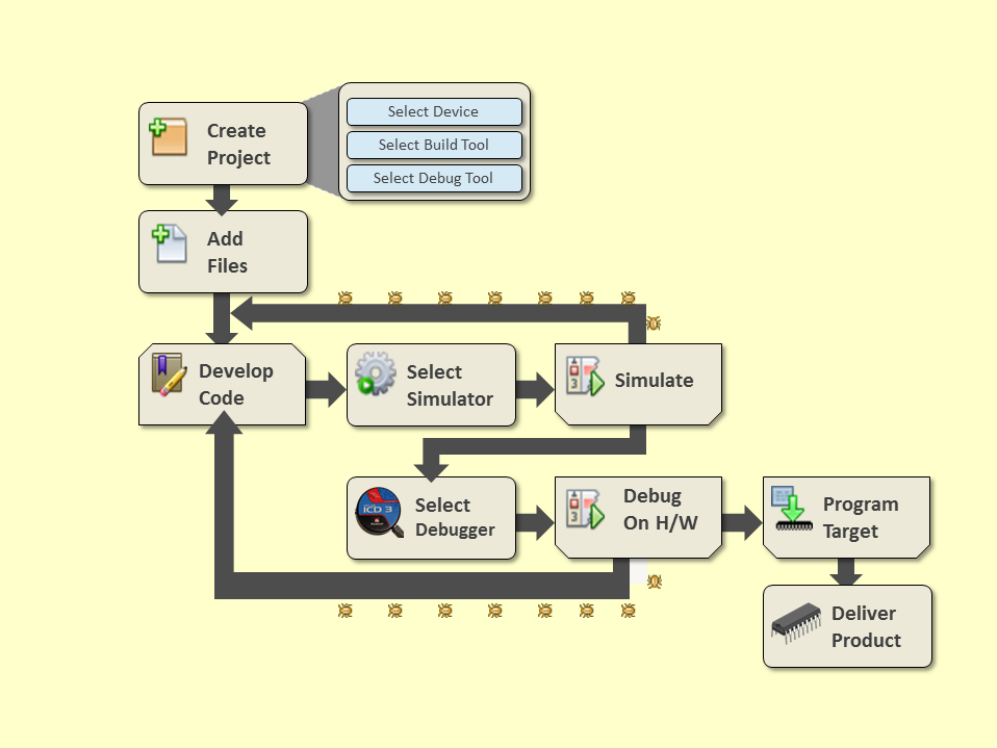

Figura A-1.: Ciclo de desarrollo de un proyecto con MPLAB® [18].

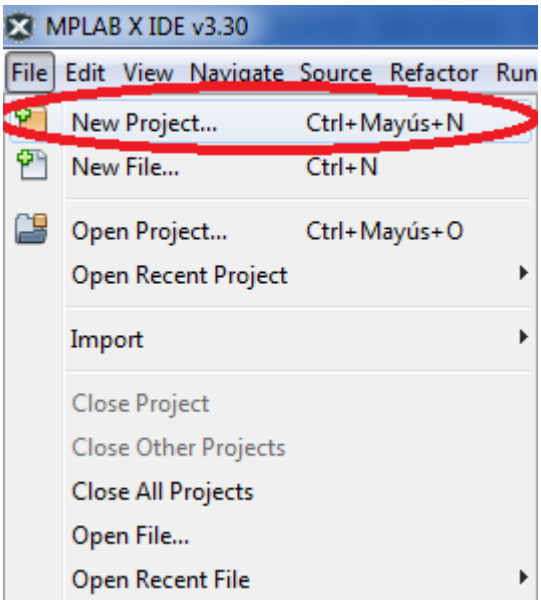

Figura A-2.: Creación de un nuevo proyecto.

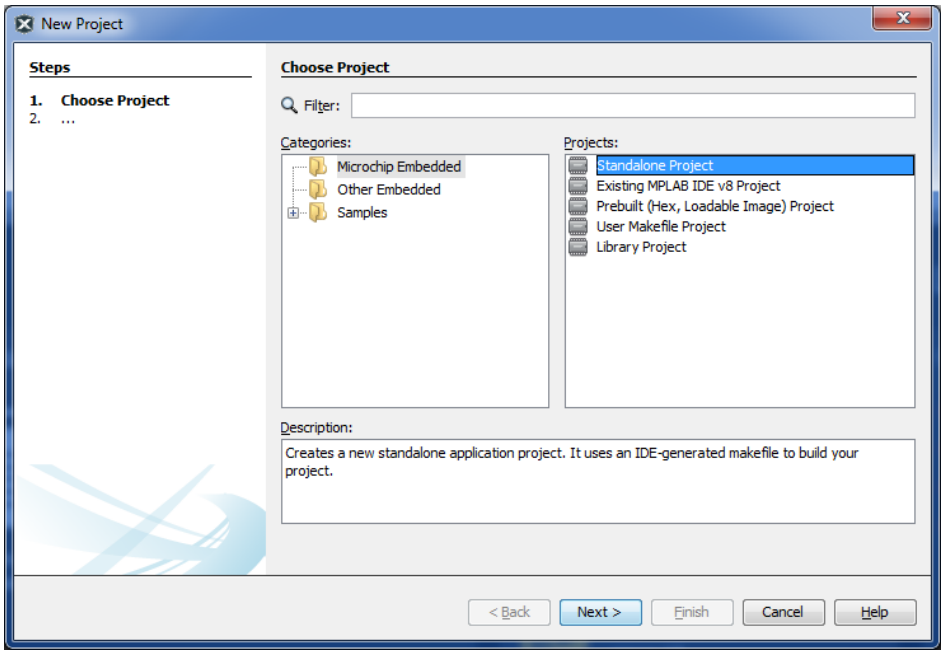

Figura A-3.: Selección del tipo de proyecto.

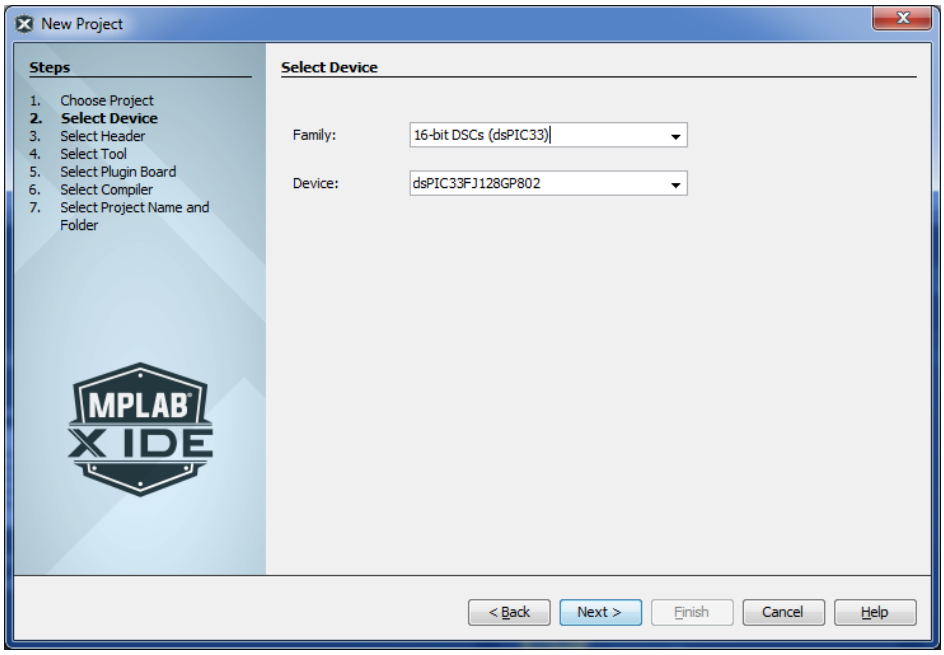

Figura A-4.: Selección del dispositivo a programar.

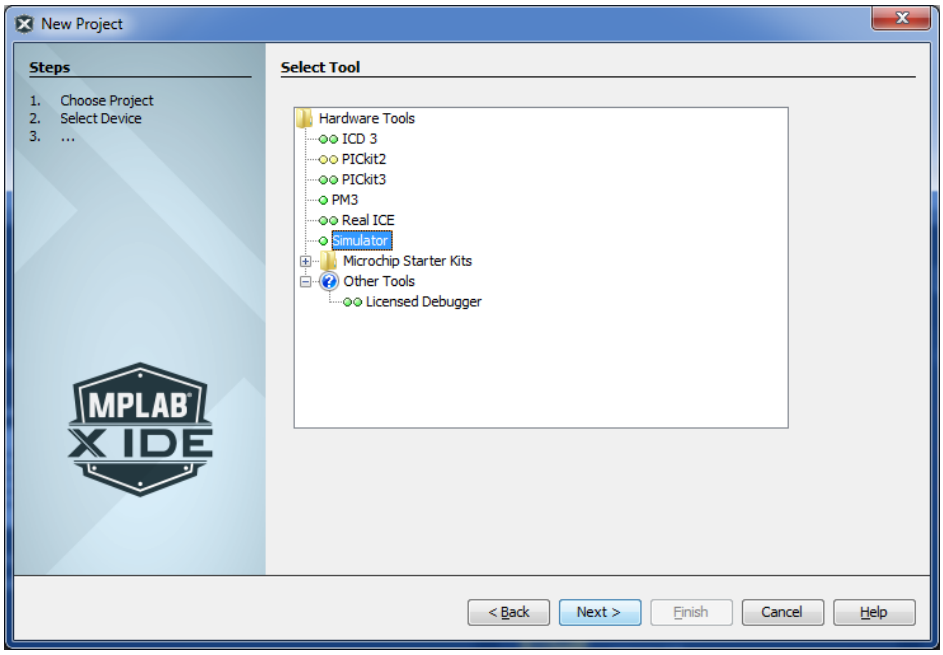

Figura A-5.: Selección de la herramienta de depuración.

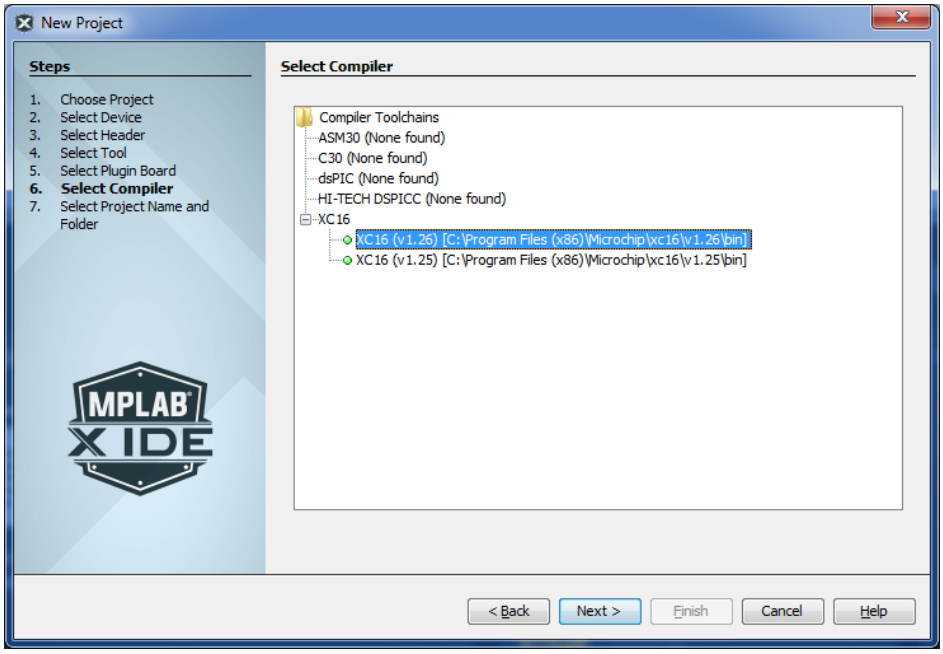

Figura A-6.: Selección del compilador.

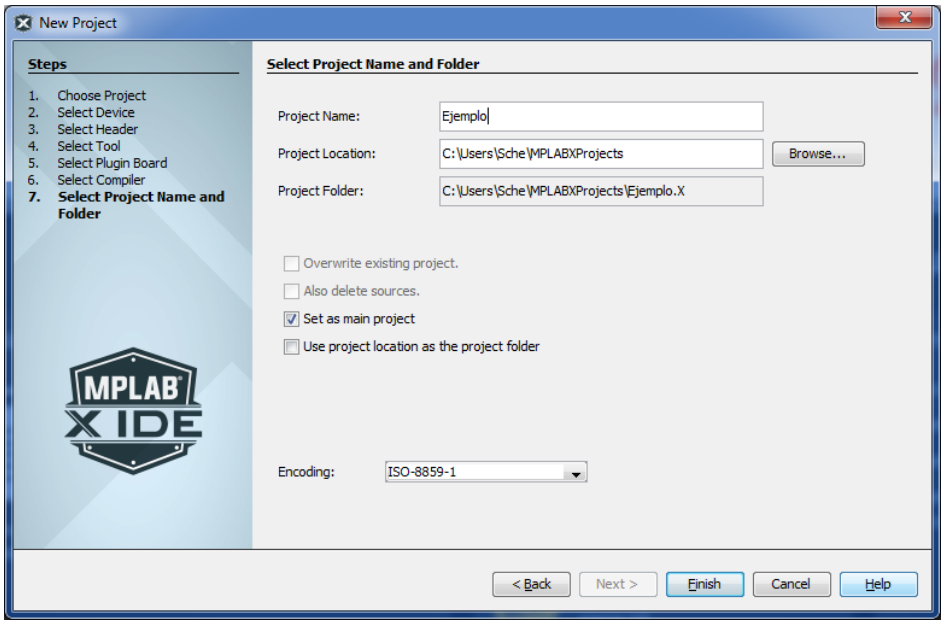

Figura A-7.: Finalización de la creación de un proyecto.

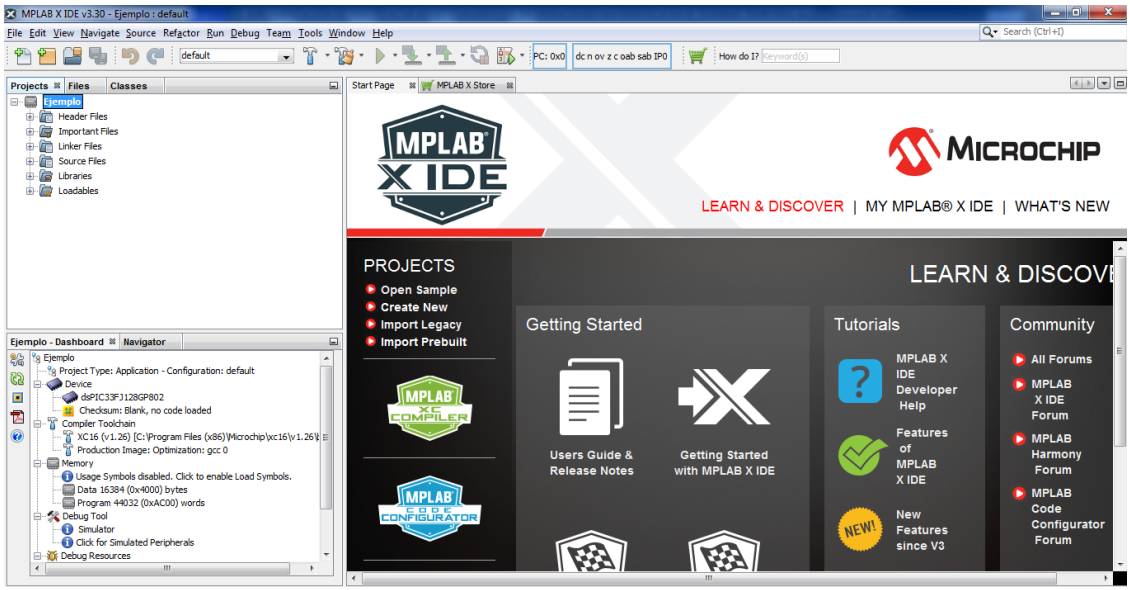

Figura A-8.: Aspecto de la interfaz de MPLAB® luego de crear un proyecto nuevo.

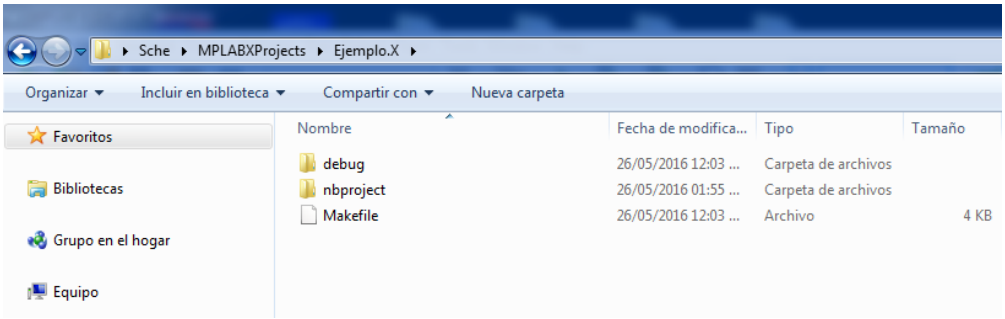

Figura A-9.: Carpeta del proyecto.

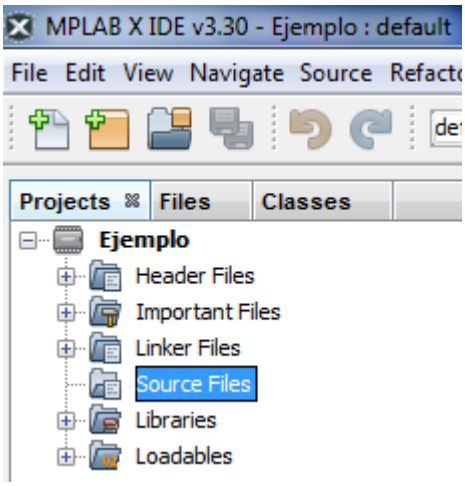

Figura A-10.: Abriendo la carpeta de códigos fuente.

cada tal y como se muestra en la Figura **A-9**.

La carpeta *debug* alberga los archivos que son creados cuando el proyecto se compila en modo *debug*. La carpeta *nbproject* contiene los archivos que definen las configuraciones del proyecto, tales como archivos incluidos, compilador seleccionado, herramienta hardware seleccionada, entre otros. Este es el aspecto de un proyecto vacío, a medida que se agregan archivos de código fuente y se compila serán generados más carpetas y archivos.

Intentando abrir la carpeta Source File, como se observa en la Figura **A-10**, puede constatarse que no existen archivos de código fuente.

El siguiente paso entonces es agregar un archivo de código fuente. Para agregar un archivo de c nuevo, se da clic derecho sobre la carpeta «Source Files» luego se selecciona $New- > Other...$ (Figura **A-11**).

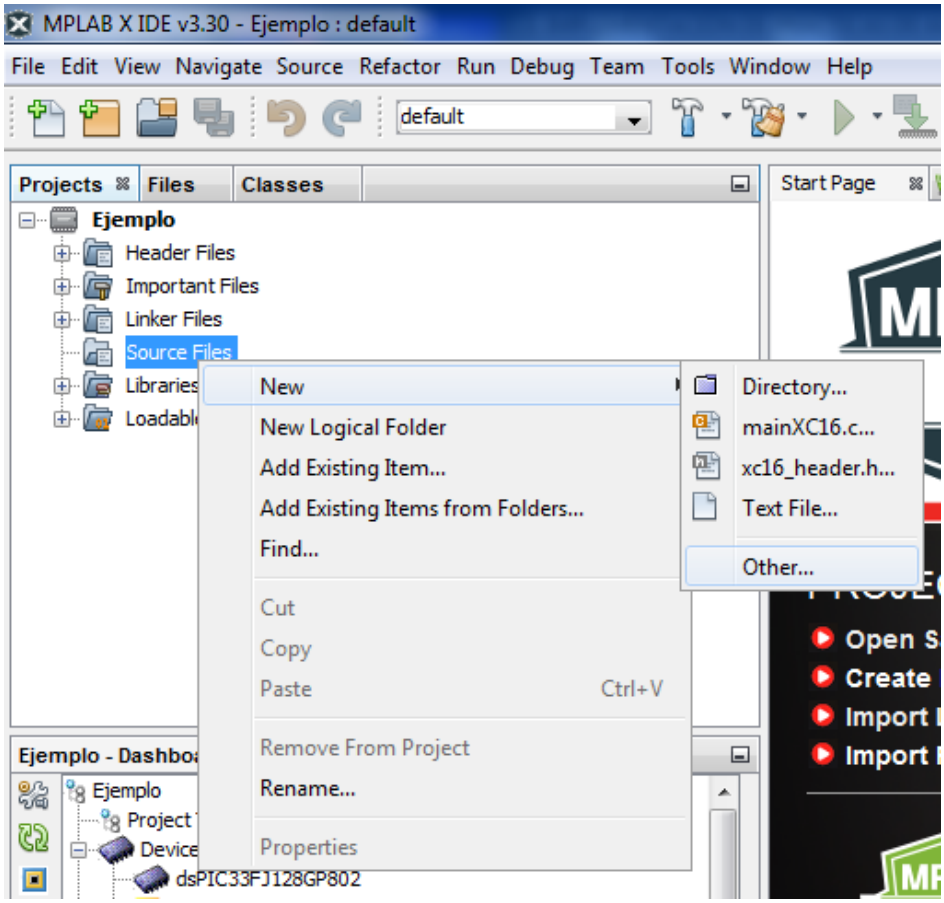

Figura A-11.: Añadiendo un nuevo código fuente al proyecto.

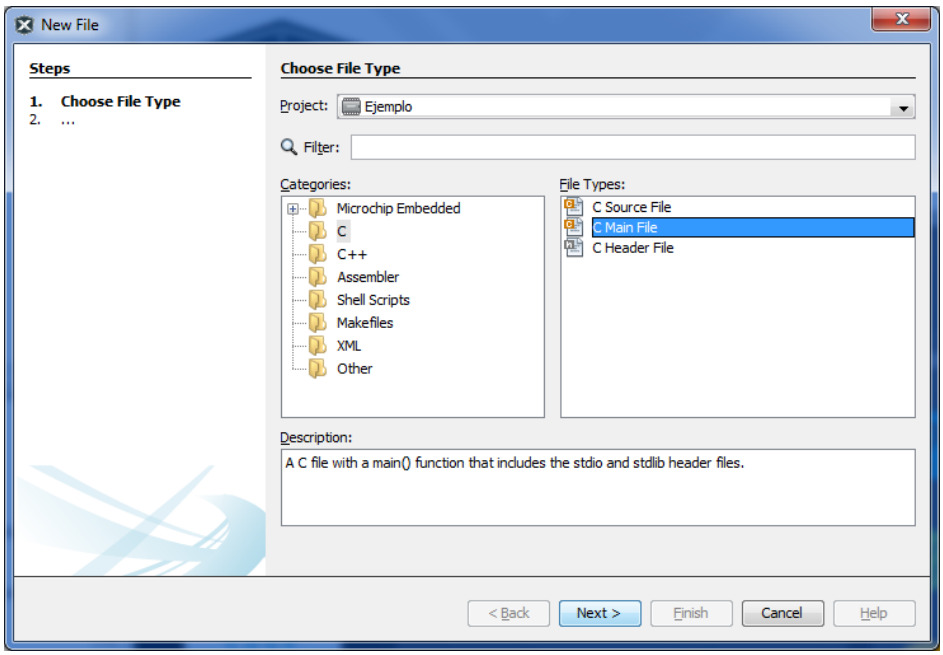

Figura A-12.: Añadiendo una función principal del lenguaje C.

En la ventana que se abre (**A-12**) se buscas en la región de categorías la opción C, a la derecha están las opciones de tipo de archivo de C, se selecciona la opción «C Main File» para crear el archivo principal del proyecto, luego clic en *Next*.

En la siguiente ventana se da un nombre al archivo en el campo «File Name» y clic en *Finish*.

De esta forma se crea una plantilla de C vacía para escribir el código principal, el aspecto del código es como el de la Figura **A-14**. De ahí se puede notar los siguientes puntos:

- El IDE incluye por defecto unas bibliotecas pero la correspondiente al dispositivo que va a ser programado, se debe adicionar manualmente. Por ejemplo:

```
#include <dspic33fj128gp802.h>
```

- También es importante añadir la biblioteca DSP, ya que en adelante será utiizada como soporte para el procesamiento de señales.

```
#include <dsp.h>
```

- La función main contiene por defecto unos parámetros que pueden suprimirse, con esos pequeños cambios el código queda de la siguiente manera:

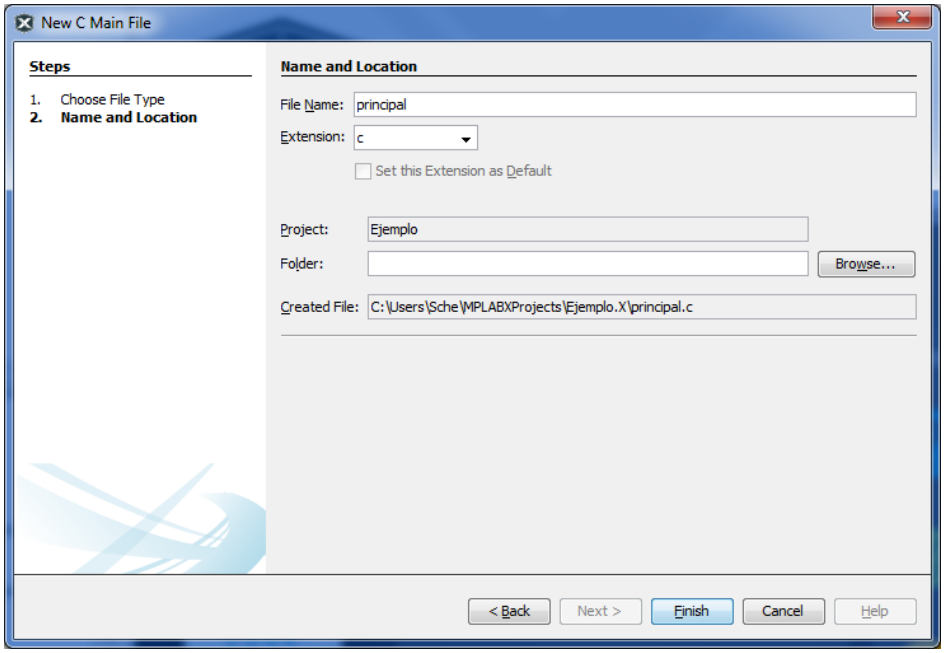

Figura A-13.: Dando nombre al archivo de la función principal.

```
#include <stdio.h>
#include <stdlib.h>
#include <p33fj128gp802.h>
#include <dsp.h>

int main()
{
        return(EXIT_SUCCES);
}
```

De forma similar a los archivos de C, se añaden los archivos de ensamblador (Figura **A-15**).

Otro tipo de archivo importante es el archivo de cabecera, el cual contiene las definiciones o macros que se usarán en los archivos fuente, son muy útiles para asignar nombres a partes de código como registros u operaciones para que el análisis del código principal sea más entendible. Para el caso de ensamblador la extensión de este tipo de archivos es .inc y para el caso de c la extensión es .h.

Para crear un archivo de cabecera usando el software MPLAB® X primero se debe hacer clic derecho en la carpeta Header Files luego seleccionar New-¿Other... (Figura **A-16**).

La Figura **A-17** muestra la misma ventana usada para agregar el archivo principal de C

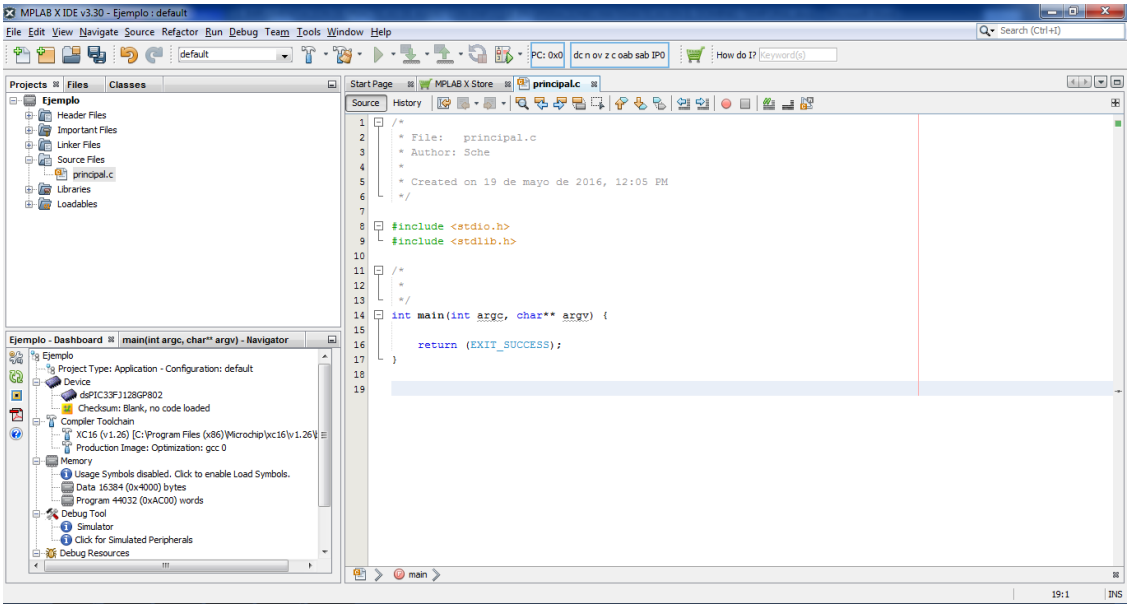

Figura A-14.: Aspecto del código del programa principal.

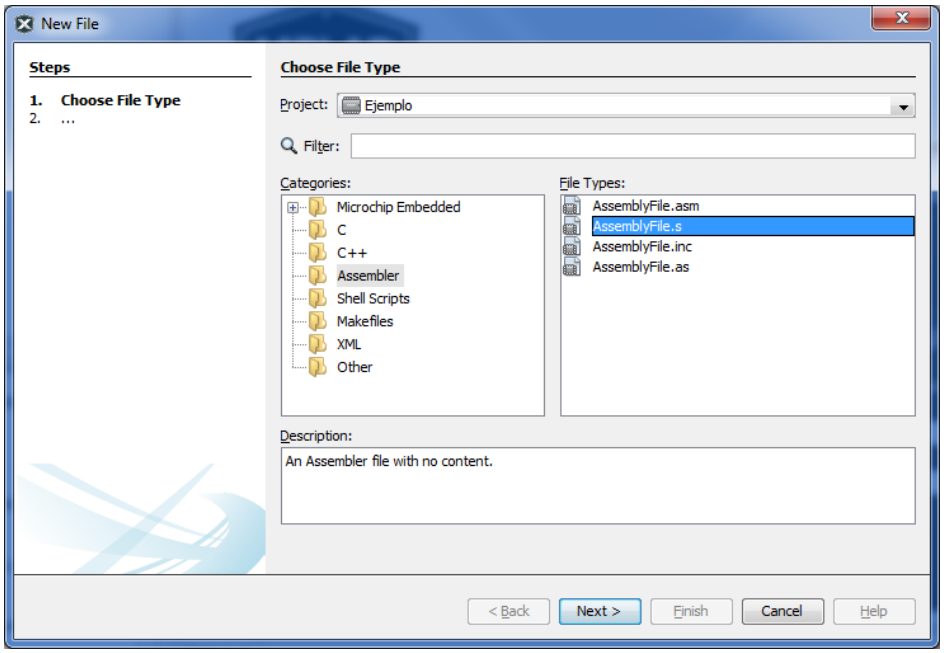

Figura A-15.: Añadiendo un archivo de ensamblador al proyecto.

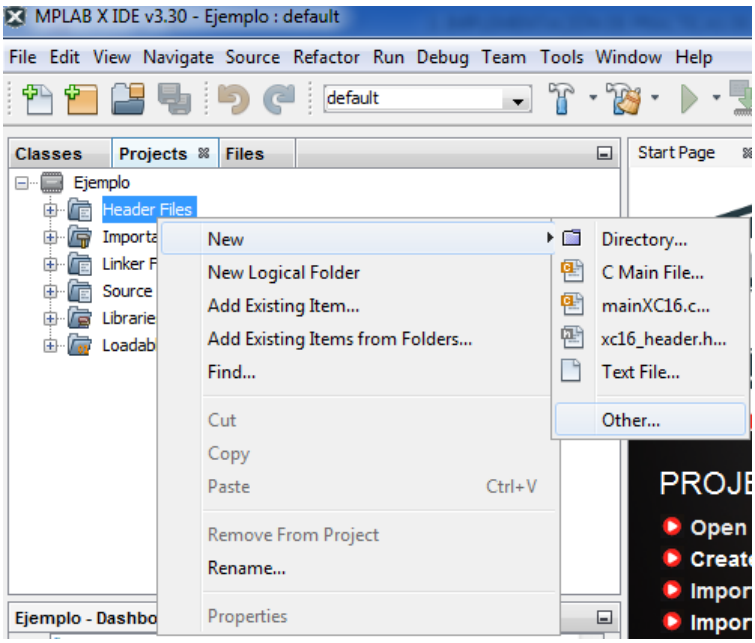

Figura A-16.: Creación de un archivo de cabecera.

pero esta vez ha sido seleccionada la opción C Header File.

Luego se procede a darle un nombre al archivo y se da clic en Finish (Figura **A-18**), el resultado será un archivo de texto con el siguiente aspecto:

```
/*
 * File:   macros.h
 * Author: Sche
 *
 * Created on 19 de mayo de 2016, 12:35 PM
 */

#ifndef MACROS_H
#define MACROS_H

#ifdef __cplusplus
extern "C" {
#endif




#ifdef __cplusplus
}
#endif
```

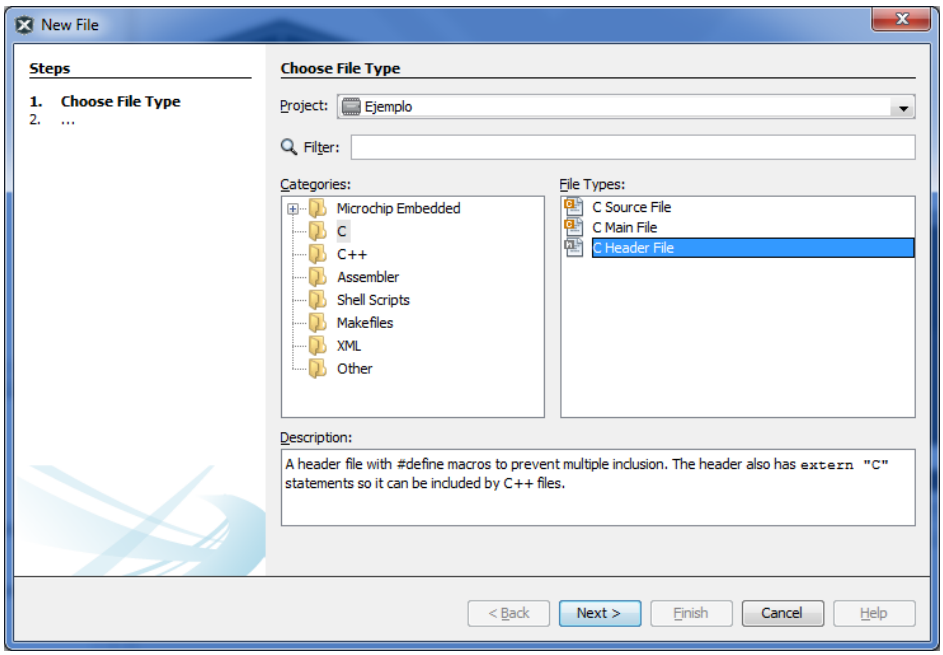

Figura A-17.: Añadiendo un archivo de cabecera de C.

```
#endif  /* MACROS_H */
```

Eliminando algunas líneas que son para extender funciones hacia el lenguaje C++ se puede simplificar el archivo quedando de la siguiente manera:

```
/*
 * File:   macros.h
 * Author: Sche
 *
 * Created on 19 de mayo de 2016, 12:35 PM
 */

#ifndef MACROS_H
#define MACROS_H

/*
 *  En este espacio se deben escribir las definiciones o macros
 */

#endif  /* MACROS_H */
```

En el espacio indicado se debe escribir el contenido como tal del archivo de cabecera, ahí se pueden definir nombres de registros, declarar funciones, invocar otros archivos de cabecera, entre otras cosas; aunque también se puede declarar variables, lo más recomendable es hacer esto en el respectivo código fuente.

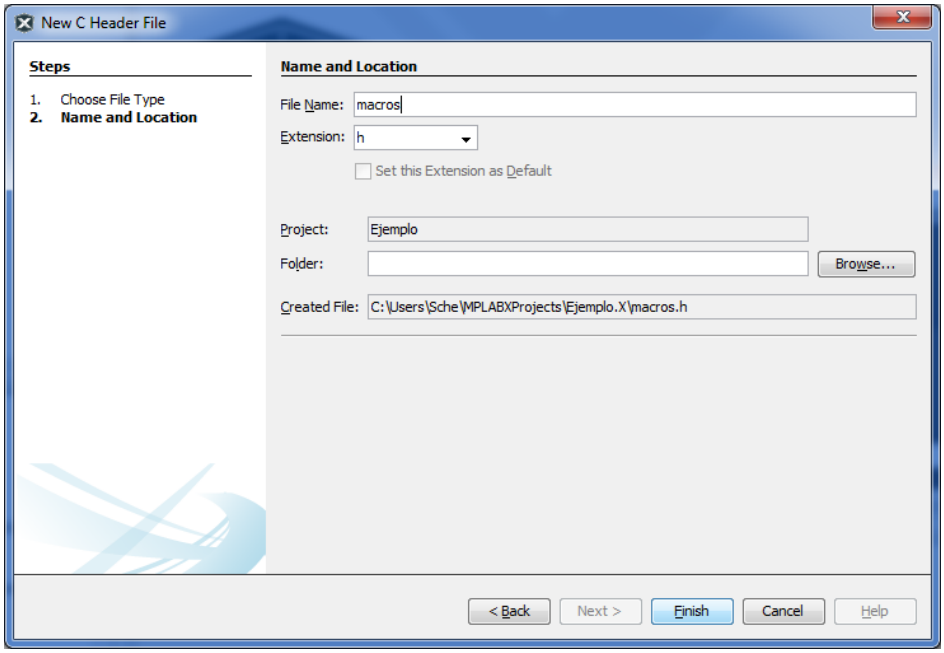

Figura A-18.: Nombrando el archivo de cabecera de C.

Para que un archivo de cabecera sea visible para el código fuente, debe declararse igual que las demás bibliotecas, pero en lugar de los signos menor y mayor (<>) deberá usarse las comillas dobles (“”), estas últimas le indican al compilador que la biblioteca declarada está contenida en la carpeta del proyecto, mientras que los otros signos indican que la biblioteca está integrada al compilador mismo. Entonces la declaración del archivo de cabecera recientemente creado se hace así:

```
#include <macros.h>
```

Incluyéndolo en el proyecto de ejemplo queda de la siguiente manera:

```
#include <stdio.h>
#include <stdlib.h>
#include <p33fj128gp802.h>
#include <dsp.h>
#include <macros.h>
```

De aquí en adelante todo consiste en programar, para esto es muy importante conocer el dispositivo a través de la hoja de características y el manual de referencia para poder escribir las funciones que controlarán el funcionamiento del dsPIC®.

Bibliografía

[1] Alvarado Moya, José P.: *Procesamiento Digital de Señales*. Cartago, Costa Rica : Notas de clase, 2011 5

[2] Angulo Usategui, José M. ; García Zapiraín, Begoña ; Angulo Martinez, Ignacio ; Sáez, Javier V.: *Microcontroladores avanzados dsPIC: Controladores Digitales de Señal. Arquitectura, programación y aplicaciones*. Madrid : Thomson-Paraninfo, 2006 2

[3] Angulo Usategui, José M. ; Etxebarría Ruiz, Aritza ; Angulo Martinez, Ignacio ; Trueba Parra, Iván: *dsPIC Diseño Práctico de Aplicaciones*. McGraw-Hill-Interamericana de España, 2006 xvii, 2, 10

[4] Baker, Bonnie: *A Baker's Dozen: Real analog solutions for digital designers*. Elsevier, 2005 xvii, 35

[5] Baker, Bonnie C.: Anti-aliasing, analog filters for data acquisition systems. En: *AN699, Microchip Technology Inc., DS00699* (1999) xvii, 8

[6] Baker, Bonnie C.: Operational Amplifier Topologies and DC Specifications. En: *AN722, Microchip Technology Inc., DS00722* (1999)

[7] Baker, Bonnie C.: Operational Amplifier AC Specifications and Applications. En: *AN723, Microchip Technology Inc., DS00723* (2000)

[8] Baker, Bonnie C.: Using Operational Amplifiers for Analog Gain in Embedded System Design. En: *AN682, Microchip Technology Inc.* (2000)

[9] Barrero García, Federico J. ; Toral Marín, Sergio L. ; Ruiz González, Mariano: *Procesadores Digitales de Señal de altas prestaciones de Texas Instruments ™: De la familia TMS320C3x a la TMS320C6000*. Madrid : McGraw-Hill-Interamericana de España, 2005 2

[10] Clavijo Mendoza, Juan R.: *Diseño y simulación de sistemas microcontrolados en lenguaje C: programación con el compilador MikroC y simulación en Proteus ISIS*. Colombia : ISBN 978-958-44-8619-6, Mayo 2011

[11] Couch, León W León W: *Sistemas de comunicación digitales y analógicos*. 7 Ed. México : Pearson Educación,, 2008 xix, 81

[12] Curtis, Keith: Analog Design in a Digital World Using Mixed Signal Controllers. En: *AN823, Microchip Technology Inc., DS00823A* (2002) xvii, xvii, 36, 37

[13] Di Jasio, Lucio: *Programming 16-bit Microcontrollers in C Learning to Fly the PIC 24*. 1 Ed. E.U.A. : Newnes, 2007

[14] e2v: Design Considerations for Mixed-Signal: How to Design a PCB Layout. En: *AN, e2v semiconductors SAS, 0999A-BDC-07/09* (2009) xvii, 37

[15] Kahrs, Mark ; Brandenburg, Karlheinz: *Applications of digital signal processing to audio and acoustics*. Vol. 437. Springer Science & Business Media, 1998 xvii, 6, 9, 86

[16] Microchip. *Audio and speech: Development Tools*. {En línea} Disponible en: (http://www.microchip.com/pagehandler/en-us/technology/audio/home.html?tab=t2). {27 marzo de 2015}

[17] Microchip. *DSP Features of the Microchip dsPIC®DSC*. {En línea} Disponible en: (http://microchip.wikidot.com/dsp0201:start). {6 Junio de 2016} xvii, xvii, xvii, xvii, xvii, xviii, 13, 14, 15, 76

[18] Microchip. *Get Started with MPLAB®X IDE and Microchip Tools*. {En línea} Disponible en: (http://microchip.wikidot.com/tls0101:start). {6 Junio de 2016} xvii, xix, 19, 120

[19] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 33. Audio Digital-to-Analog Converter (DAC)*. DS70211B. 2009

[20] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 3. Data Memory*. DS70202C. 2010

[21] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 4. Program Memory*. DS70203D. 2010

[22] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 2. CPU*. DS70204C. 2011

[23] Microchip Inc.: *16-bit Digital Signal Controllers (up to 128 KB Flash and 16K SRAM) with Advanced Analog*. Datasheet, DS70292G. 2012 xviii, xviii, 41, 42

[24] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 16. Analog-to-Digital Converter (ADC)*. DS70183D. 2012 xviii, xviii, xviii, xviii, 59, 60, 61

[25] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 30. I/O Ports with Peripheral Pin Select (PPS).* DS70190E. 2012 xvii, 17

[26] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 32. Interrupts (Part III).* DS70214C. 2012

[27] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 38. Direct Memory Access (DMA) (Part III).* DS70215C. 2012 xvii, 16

[28] Microchip Inc.: *dsPIC33F/PIC24H Family Reference Manual: Section 39. Oscillator (Part III).* DS70216D. 2012 xviii, xviii, xviii, 54, 55, 64

[29] Microelectronics, ST: *LD1117 series: Low drop fixed and adjustable positive voltage regulators.* Datasheet xviii, xviii, 48, 49

[30] Milivojevic, Zoran. *Digital Filter Design.* Libro digital disponible en (http://learn.mikroe.com/ebooks/digitalfilterdesign/) xix, xix, xix, xix, 95, 96

[31] Mitra, Sanjit K.: *Digital signal processing applications*

[32] Mitra, Sanjit K.: *Procesamiento de señales digitales: Un enfoque basado en computadora.* 3 Ed. México D.F. : McGraw-Hill-Interamericana de México, 2006 2

[33] Salazar, Jordi. *Procesadores digitales de señal (DSP): Arquitectura y criterios de selección.* En línea Disponible en (http://arantxa.ii.uam.es/~taao1/teoria/tema1/pdf/Procesadores_dig.pdf). {27 marzo de 2015}

[34] Shenoi, B. A.: *Introduction to digital signal processing and filter design.* New Jersey : Wiley-Interscience, 2006

[35] Smith, J.O. *Introduction to Digital Filters with Audio Applications.* http://ccrma.stanford.edu/~jos/filters/, online book, 2007 edition. accessed <2016-06-05>

[36] Smith, J.O. *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition.* http://ccrma.stanford.edu/~jos/mdft/, online book, 2007 edition. accessed <2016-06-05>

[37] Soria ; Martínez ; Francés ; camps: *Tratamiento digital de señales: problemas y ejercicios resueltos.* Madrid : Pearson Prentice Hall, 2003

[38] Zator Systems. *Tratamiento digital del sonido.* {En línea} Disponible en: (http://www.zator.com/Hardware/H10_2.htm). {6 junio de 2016}