Martínez Hernández
Moisés Agustín

Sistema de Tiempo Real Embebido en un
Microcontrolador de Alto Desempeño para el
Procesamiento de Audio

2010

Universidad Autónoma de Querétaro
Facultad de Ingeniería

Sistema de Tiempo Real Embebido en un Microcontrolador de Alto Desempeño para el Procesamiento de Audio

Tesis
Que como parte de los requisitos para
obtener el grado de

INGENIERO EN AUTOMATIZACIÓN

Presenta

Martínez Hernández Moisés Agustín

Dirigida por

M. C. Manuel Toledano Ayala

C.U. Santiago de Querétaro, Qro. Agosto 2010

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Ingeniería en Automatización

Sistema de Tiempo Real Embebido en un Microcontrolador de Alto Desempeño para el Procesamiento de Audio

TESIS

Que como parte de los requisitos para obtener el grado de

INGENIERO EN AUTOMATIZACION

Presenta:
Moisés Agustín Martínez Hernández

Dirigido por:
M.C. Manuel Toledano Ayala

SINODALES

M.C. Manuel Toledano Ayala Presidente	_____ Firma
Dr. Genaro Martín Soto Zarazúa Secretario	_____ Firma
Dr. Edgar Rivas Araiza Vocal	_____ Firma
Ing. Carlos Ricardo Luna Ortiz Suplente	_____ Firma
M.C Juvenal Rodríguez Reséndiz Suplente	_____ Firma

_____ Dr. Gilberto Herrera Ruíz Director de la Facultad	_____ Dr. Luis Hernández Sandoval Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Agosto 2010
México

RESUMEN

Esta investigación es enfocada a la realización e implementación de un control que sea capaz de desarrollar un procesamiento de audio de una manera más económica, e igualmente eficaz que dispositivos especializados en esa área, empleando un sistema operativo de tiempo real embebido en un microcontrolador. Inicialmente se pretende explicar las características previas al procesamiento de audio, como es una correcta etapa de acondicionamiento que permita el funcionamiento adecuado del microcontrolador, se muestran sugerencias para realizar una etapa que permita economizar y ahorrar espacio de diseño al usuario. Se explican las características que debe poseer un microcontrolador para que se pueda implementar un código adecuado para el procesamiento de audio, como son la memoria, las instrucciones por segundo, características de voltaje de entrada, etc. De igual manera se hacen referencias para tomar la decisión de el compilador adecuado, que se pueda utilizar y satisfaga las necesidades de la investigación, empezando por la parte del Sistema Operativo en Tiempo Real. Se intenta demostrar como un Sistema Operativo en Tiempo Real mejora el desempeño de la parte crítica de un programa, claramente realizando en base a multitareas. Finalmente se intenta demostrar que los objetivos y la hipótesis planteados se cumplen satisfactoriamente, permitiendo dejar una investigación que abrirá oportunidades a futuros proyectos.

(**Palabras clave**: Sistemas embebidos, Tiempo Real, procesamiento de audio, FFT, microcontrolador)

SUMMARY

This research is focused on the realization and implementation of a control that is able to develop an audio processing in a more economical and equally effective than specialized devices in the area, using a real-time operating system embedded in a microcontroller. Initially intended to explain the features pre-processing audio, such as proper conditioning stage to permit the proper functioning of the microcontroller, display suggestions for a stage that allows saving and space saving design to the user. The characteristics that must have a microcontroller so that it can implement an appropriate code for processing audio, such as memory, instructions per second, input voltage characteristics, etc. Similarly, references are made to make the decision of the appropriate compiler, which can be used and meets the needs of research, starting with the part of the Real Time Operating System. We encourage showing as a Real Time Operating System enhances the performance of the critical part of a program, clearly making based on multi-tasking. The intention was to demonstrate that the objectives and hypotheses are being satisfactorily implemented, allowing an investigation that will leave opportunities to future projects.

(**Key words**: Embedded Systems, Real-time audio processing, FFT, Microcontroller)

A mi abuelo y mi madre

AGRADECIMIENTOS

Inicialmente quiero agradecer a Dios por permitirme estar en donde estoy y generarme las circunstancias para llegar hasta aquí.

A mi madre Arcadia Hernández Espinosa por apoyarme durante toda mi formación estudiantil, es por ella que me esfuerzo día a día, sin ella no habría tenido la determinación de seguir adelante y mucho menos de lograr todo lo que tengo.

A mis profesores que me apoyaron en todo momento.

A mis amigos que me acompañaron a lo largo de mi formación académica, Alberto, Jorge, Genaro, Roberto y sobre todo a Fortino y Juvenal que me ayudaron a superarme y esforzarme en mis trabajos.

Í N D I C E

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCION

En la actualidad los escenarios donde los seres humanos interactúan son del tipo inteligente en su mayoría, lo que quiere decir que están dotados de un sin número de funciones para su bienestar, comodidad, entre otros.

Cuando se habla de acondicionar un espacio de trabajo o de entretenimiento incluso solo para tener una buena presentación del lugar, se pide que se maneje lo más innovador que nos pueda proporcionar una ligera idea de la capacidad humana para el desarrollo de control sobre los procesos de automatización, que no solo se encuentran dentro de una industria, o dentro del desarrollo de tecnología innovadora, también se puede representar de forma que cosas tan sencillas se puedan realizar de una manera más fácil, eficiente e interactiva. Hoy en día la gran mayoría de los procesos que se desarrollan exigen el uso de un proceso en tiempo real, sobre todo cuando se habla de trabajar con un muestreo que requiera de varios filtros de tareas, una muestra clara es el procesamiento de audio, el cual se ha desempeñado en múltiples tareas, tareas que dadas las circunstancias requieren de un procesamiento de a señal muestreada al mismo tiempo que realizan sus tareas indicadas, la interpretación de la música, es una tarea muy compleja que requiere de costos muy altos y como ya se ha repetido anteriormente una buena precisión, pero cuando se habla de todo ello se sobreentiende que la automatización se encarga de ser la solución de casos como esos.

El acondicionar con una muestra de arte y automatización, como lo son las fuentes bailarinas, es un ejemplo claro de trabajo sobre control, velocidad y precisión, la realización de esta actividad tiene una demanda muy alta de comunicación de dispositivos, que entre si realizan multitareas, a base de una comunicación, ya que los modelos actuales son más simples siempre y cuando se

les programe una secuencia, que vaya de acuerdo con la música que se toca, y que su programación está dada para seguir una rutina cada determinado tiempo, esto no nos proporciona una idea de los avances que se tiene hoy en día, el desarrollo y la formación dl ingeniero se dan para atacar problemas que no solo están presentes dentro de una industria sino dentro de procesos que requieran ese tipo de control, sabemos que podría ser más económico realizar una rutina sencilla , el obtener que un dispositivo realice un procesamiento de audio nos exigiría que dicho dispositivo fuera muy caro, en un dado caso lo cual no sería conveniente y en segunda que tenga la capacidad de realizar varias tareas dentro de un tiempo real, se podría pensar que hoy en día ya existen dispositivos que claramente podrían desarrollar este tipo de acciones, un ejemplo claro una tarjeta Spartan, que es lo que más se mueve dentro de la rama de la programación de dispositivos, si bien se sabe que la Spartan puede ejecutar tareas en paralelo, incluso de puede utilizar para acondicionamiento y procesamiento de señales, pero el poder ejecutar todas esas tareas nos implica que es un dispositivo que tiene un alto costo, claramente dependiendo de su diseño; entonces podríamos pensar en algunos otros dispositivos en los cuales se pueda hacer procesamiento de audio, que tal vez se puedan explicar que son los más aptos para la tarea y de un costo considerable.

Trabajar un procesamiento de audio nos da de inmediato una idea de lo costoso que sería implementar alguna tarea como la que se ha descrito (las Fuentes Bailarinas), incluso pensar en las fuentes que están montadas en el espectáculo de las vegas, es una muestra clara de que se debe tener una considerable suma de dinero para el desarrollo de este tipo de montajes.

Hace un par de años atrás, dos “locos científicos” (como se les nombro), conocidos como EepyBird se convirtieron en una sensación en el Internet al dejar caer, aproximadamente más de 500 mentas de “Mentos” en 200 litros de DietCoke obteniendo resultados explosivos. Gracias a una fuerza científica llamada nucleación y una manipulación cuidadosa, la coca cola saltó al aire como

un geiser y bailó alrededor asemejándose a las mundialmente famosas Fuentes del Bellagio.

El creador de las Fuentes, WET Design, se asombró tanto con este divertidísimo tributo que llamo a los creadores Fritz Grobe y Stephen Voltz a su oficina, para que hicieran una presentación en vivo.

Con numerosas rutinas musicales coreografiadas profesionalmente y agua en aceleración elevándose al cielo tanto como 460 pies desde un total de 1,214 aparatos que emiten agua, el lago de $40 millones y 8.5 acres es la instalación de agua más ambiciosa que haya creado WET Design. El show gratuito ha aparecido en numerosas películas, shows de televisión, fotografías y artículos sobre Las Vegas.

El programa computarizado que opera el show de las fuentes automáticamente cancela o disminuye la intensidad basado en condiciones climáticas.

Proporcionar que un sistema sea totalmente autónomo, es una característica que permite abrir nuevas oportunidades a la humanidad en general, no solo en la parte económica sino la posibilidad de desarrollar más áreas de oportunidad, las Fuentes de Bellagio son un ejemplo claro de que la automatización está presente en áreas que son tan cotidianas para las personas, y claramente impresionantes.

Una clara desventaja son los recursos que requiere un proceso de tal complejidad, empezando por un tendencia que últimamente ha tomado fuerza dentro de muchas aplicaciones, aunque no tan joven cómo aparenta, es la programación de aplicaciones embebidas utilizando Sistemas Operativos de Tiempo Real. Este tipo de herramientas son un poderoso auxiliar en el desarrollo y puesta a punto de sistemas complejos y se puede encontrar en las aplicaciones más comunes con las que nos relacionamos día a día. La más fácil de notar es el teléfono celular, se debe tomar en cuenta que todos los teléfonos celulares modernos utilizan algún

sistema operativo de tiempo real, sistema operativo embebido o empotrado como también se le suele llamar.

El mercado de dispositivos electrónicos, exige actualmente, el desarrollo cada vez más rápido de estos sistemas, esta carrera por poner un nuevo producto a la venta antes que la competencia obliga a los desarrolladores a programar aplicaciones cada vez más complejas en menos tiempo. Es por ello que los tradicionales métodos de programación con lenguajes ensambladores e incluso la programación de microcontroladores con lenguajes de alto nivel como C o BASIC, están quedando atrasados.

El pensar en desarrollar un sistema de tiempo real que pueda desempeñarse en tecnología se puede esperar que tenga un elevado costo, pero tal vez el tener en consideración dispositivos como lo son las tarjetas Spartan o incluso un DSP, estaríamos hablando de dispositivos costosos, pero no necesariamente se necesita utilizar dispositivos tal elevados, solo se requiere de un hardware adecuado para el desarrollo de un software apto, por ende el plantear el uso de un microcontrolador podría ser apto para una situación de realizar multitareas en tiempo real, claramente deberá ser un micro de alto rendimiento para poder atacar ese problema.

Un microcontrolador está adaptado para trabajar en tiempo real, pero a su vez tiene la capacidad de realizar diferentes tareas cada una con su respectiva prioridad, si bien otra ventaja que existe es su capacidad para comunicarse con otros dispositivos, con diferentes protocolos. El que pueda comunicarse con diferentes protocolos nos permite que se puedan conectar microcontroladores para tener una ejecución más adecuada de un proceso.

1.1 Hipótesis

El procesamiento de señales realizado por un sistema operativo en tiempo real embebido en un microcontrolador permite optimizar el tiempo de ejecución de las tareas críticas y no críticas asignadas para el procesamiento de audio y reducir costos en la unidad de procesamiento.

1.2 Objetivos

1. Implementar la realización de multitareas a través de un sistema de tiempo real embebido en un microcontrolador para el procesamiento de audio

2. Realizar la comunicación entre microcontroladores para así tener una mayor eficacia con respecto a la respuesta de las tareas asignadas.

3. Integrar una etapa de procesamiento de señales en el sistema operativo en tiempo real, minimizando los componentes empleados para la reducción de costos materiales.

2. REVISION DE LITERATURA

2.1 Marco Teórico

2.1.1 Descripción de los microcontroladores PIC de alto desempeño

Los microcontroladores están invadiendo el mundo. Están presentes en nuestra casa, en nuestro trabajo y en nuestra vida. Se pueden encontrar controlando los hornos de microondas y los televisores de nuestro hogar, en los teclados y ratones de los computadores y en los automóviles. En el bolsillo llevamos unos cuantos entre los del teléfono móvil, los que tienen las modernas llaves del coche, los mandos a distancia del garaje y la alarma doméstica.

Para el hogar inteligente es necesario contar con un dispositivo que se encargue del control de los distintos dispositivos que se hallen dentro del hogar; este dispositivo es llamado microcontrolador, que es el común denominador en cuanto a mando o accionamiento de los electrodomésticos. También es el encargado de recibir la información y procesarla de acuerdo a la petición que haga el usuario ya sea de forma local o remota. Básicamente un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador, teniendo en cuenta que un controlador es un dispositivo que se emplea para el gobierno de uno o varios procesos [12].

De manera general los microcontroladores se caracterizan por disponer normalmente de: procesador, memoria para contener datos, módulos de entradas y salidas para comunicarse con el exterior, diversos módulos para el control de periféricos (ADC, PWM, UART, etc.), generación de pulso de reloj para poder sincronizar todo un sistema. El microcontrolador es un circuito integrado que incluye todos los componentes de una PC, debido a su reducido tamaño es posible montar el microcontrolador en el propio dispositivo que gobierna.

Los microcontroladores actualmente son fáciles de conseguir, económicos, operan a altas frecuencias lo que nos da una alta velocidad de funcionamiento; cuentan con herramientas de desarrollo baratas y fáciles de usar, muchas de ellas se pueden tomar de forma libre por medio de los proveedores como son Microchip, Atmel, Motorola, Intel, etc.; también se cuenta con una gran variedad de hardware que permiten grabar, depurar, borrar y comprobar el comportamiento de los mismos.

Un microcontrolador posee todos los componentes de una computadora, pero con unas características fijas que no pueden alterase.

Se pueden emplear como coordinador o como puente entre interfaces, ya sea que el microcontrolador sea el encargado de gestionar el control y administración de los recursos, o que el mismo pueda hacer eso y a la vez ser manipulado de manera remota por algún otro dispositivo como una PC a través de internet.

Su gran variedad hace que pueda implementar funciones simples como el control On/Off, adquisición de señales analógicas tal como la temperatura, control analógico por PWM, así como cuestiones de protocolos de comunicación como el RS-232, USB o Ethernet.

Aunado a estas características se encuentra la facilidad de programarlos y hacer desarrollo en ellos mismos con poca electrónica, además de ser económicos y fáciles de conseguir en el mercado de la electrónica.

2.1.2 Descripción de un Sistema Operativo en Tiempo Real (RTOS)

El sistema operativo es el principal componente de arquitectura responsable de asegurar una ejecución oportuna de todas las tareas con algunos requisitos de tiempo. En presencia de diversas acciones concurrentes ejecutando en un único procesador, el objetivo de un kernel en tiempo real es garantizar que cada actividad completa su ejecución dentro de su plazo. Tenga en cuenta que esto es muy diferente a reducir al mínimo los tiempos de respuesta medio de un conjunto de tareas [7].

Un sistema operativo de tiempo real (SOTR o RTOS -Real Time Operating System en inglés), es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real. Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Si no las respeta, se dirá que el sistema ha fallado. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible (determinista).

Hay dos tipos de sistemas de tiempo real: reactivos y embebidos. Un reactivo en tiempo real del sistema tiene una constante interacción con su ambiente (por ejemplo, un piloto de control de una aeronave). Un sistema embebido en tiempo real se utiliza para controlar hardware especializado que se instala dentro de un del sistema más grande (como un microprocesador que controla los frenos antibloqueo en un automóvil). [1]

Usado típicamente para aplicaciones integradas, normalmente tiene las siguientes características:

- No utiliza mucha memoria.
- Cualquier evento en el soporte físico puede hacer que se ejecute una tarea.
- Multi arquitectura (puertos de código para otro tipo de UCP).

- Muchos tienen tiempos de respuesta predecibles para eventos electrónicos.

En la actualidad hay un debate sobre qué es tiempo real. Muchos sistemas operativos de tiempo real tienen un programador y diseños de controladores que minimizan los periodos en los que las interrupciones están deshabilitadas, un número llamado a veces duración de interrupción. Muchos incluyen también formas especiales de gestión de memoria que limitan la posibilidad de fragmentación de la memoria y aseguran un límite superior mínimo para los tiempos de asignación y retirada de la memoria [9].

Un ejemplo temprano de sistema operativo en tiempo real a gran escala fue el denominado “programa de control” desarrollado por American Airlines e IBM para el sistema de reservas Sabre.

Hay dos diseños básicos:

- Un sistema operativo guiado por eventos sólo cambia de tarea cuando un evento necesita el servicio.
- Un diseño de compartición de tiempo cambia de tareas por interrupciones del reloj y por eventos.

El diseño de compartición de tiempo gasta más tiempo de la UCP en cambios de tarea innecesarios. Sin embargo, da una mejor ilusión de multitarea. Normalmente se utiliza un sistema de prioridades fijas.

Uno de los algoritmos que suelen usarse para la asignación de prioridades es el Rate-Monotonic Schedule. Si el conjunto de tareas que tenemos es viable con alguna asignación de prioridades fijas, también es viable con el Rate-Monotonic Schedule, donde la tarea más prioritaria es la de menor periodo. Esto no quiere

decir que si no es viable con Rate-Monotonic Schedule no sea viable con asignaciones de prioridad variable. Puede darse el caso de encontrarnos con un sistema viable con prioridades variables y que no sea viable con prioridades fijas.

El RTOS no es exactamente un SO (sistema operativo) a pesar de que los dos se basan en un núcleo (kernel) que se encarga de controlar la ejecución de las tareas. La diferencia yace en la carga inicial, si es solo el núcleo RTOS o si además se cargan otros procesos (SO). El RTOS está pensado para trabajar con los microcontroladores. Puede utilizarse en los PIC de gama media pero donde se obtiene mayor rendimiento es en los PIC de gama alta. [6]

2.1.3 Comunicación entre Tareas

En [8] se explica que existen tres posibles problemas que pueden interferir con la comunicación confiable de los datos cuando se maneja un sistema de multi tareas. La tarea de recepción no podría estar lista para aceptar datos cuando la tarea de enviar este transfiriendo. La tarea de envió puede no estar lista cuando la tarea de recibir necesite los datos. O las dos tareas pueden estar operando a tasas significantemente diferentes, lo que significa que una de las dos tareas pueda ser abrumada en la transferencia.

Las diferentes tareas de un sistema no pueden utilizar los mismos datos o componentes físicos al mismo tiempo. Hay dos diseños destacados para tratar este problema.

Uno de los diseños utiliza semáforos. En general, el semáforo puede estar cerrado o abierto. Cuando está cerrado hay una cola de tareas esperando la apertura del semáforo.

Los problemas con los diseños de semáforos son bien conocidos: inversión de prioridades y puntos muertos.

En la inversión de prioridades, una tarea de mucha prioridad espera porque otra tarea de baja prioridad tiene un semáforo. Si una tarea de prioridad intermedia impide la ejecución de la tarea de menor prioridad, la de más alta prioridad nunca llega a ejecutarse. Una solución típica sería tener a la tarea que tiene el semáforo ejecutada a la prioridad de la tarea que lleva más tiempo esperando.

En un punto muerto, dos tareas tienen dos semáforos pero en el orden inverso. Esto se resuelve normalmente mediante un diseño cuidadoso, realizando colas o quitando semáforos, que pasan el control de un semáforo a la tarea de más alta prioridad en determinadas condiciones.

La otra solución es que las tareas se manden mensajes entre ellas. Esto tiene los mismos problemas: La inversión de prioridades tiene lugar cuando una tarea está funcionando en un mensaje de baja prioridad, e ignora un mensaje de más alta prioridad en su correo. Los puntos muertos ocurren cuando dos tareas esperan a que la otra responda.

Aunque su comportamiento en tiempo real es menos claro que los sistemas de semáforos, los sistemas basados en mensajes normalmente se despegan y se comportan mejor que los sistemas de semáforo.

2.1.4 Interrupciones

Un dispositivo periférico puede generar una señal eléctrica llamada interrupción que modifica ciertas banderas que se encuentran en el CPU. La detección de una interrupción es parte del ciclo de instrucción. En cada ciclo de instrucción, el CPU chequea las banderas para ver si algún dispositivo necesita atención [3].Las interrupciones generadas por los dispositivos periféricos son generalmente asíncronos con respecto al programa que se está ejecutando. Un evento es asíncrono a una entidad si el momento cuando ocurre no está determinado por la entidad. Las interrupciones no siempre ocurren en el mismo punto dentro de la ejecución de un programa. En contraste, un evento de error como la división por cero es síncrono en el sentido de que siempre ocurre durante la ejecución de una instrucción particular si el mismo dato es presentado a la instrucción [5].

Las rutinas del Sistema de Operación llamadas manejadores de dispositivos usualmente manejan las interrupciones generadas por el dispositivo. Los Sistemas de Operación usan interrupciones para implementar el tiempo compartido. Tienen un dispositivo llamado timer que genera una interrupción después de un intervalo específico de tiempo. El Sistema de Operación inicializa el timer antes de actualizar el Program Counter pare ejecutar un programa de un usuario. Cuando

el timer expira, genera una interrupción causando que el CPU ejecute la rutina de servicio de la interrupción timer.

Una señal es la notificación por software de que un evento ocurrió. Por lo general es la respuesta del Sistema de Operación. Por ejemplo, ctrl-C genera una interrupción para el manejador de dispositivo que maneja el teclado. El manejador notifica al proceso apropiado mandando un signal. El Sistema de Operación también puede enviar signals a un proceso para notificar la finalización de una E/S o de un error.

Las interrupciones pueden ser producidas por Hardware o por Software.

- Las interrupciones por hardware son producidas por un dispositivo y viajan por el mismo bus del sistema.

- Las interrupciones por software son producidas por medio de la ejecución de una operación especial que se conoce como "llamada al sistema" (system call) o por errores producidos dentro de un proceso, también conocidas como excepciones.

Hay muchos tipos de interrupciones y pare cada uno de estas existe una rutina en el sistema de operación que le da servicio. Los sistemas de operación actuales permiten a los dispositivos tales como E/S o reloj del sistema interrumpir el CPU asíncronamente.

Las interrupciones son la forma más común de pasar información desde el mundo exterior al programa y son, por naturaleza, impredecibles. En un sistema de tiempo real estas interrupciones pueden informar diferentes eventos como la presencia de nueva información en un puerto de comunicaciones, de una nueva muestra de audio en un equipo de sonido o de un nuevo cuadro de imagen en una videograbadora digital.

Para que el programa cumpla con su cometido de ser tiempo real es necesario que el sistema atienda la interrupción y procese la información obtenida antes de que se presente la siguiente interrupción. Como el microprocesador normalmente solo puede atender una interrupción a la vez, es necesario que los controladores de tiempo real se ejecuten en el menor tiempo posible. Esto se logra no procesando la señal dentro de la interrupción, sino enviando un mensaje a una tarea o solucionando un semáforo que está siendo esperado por una tarea. El programador se encarga de activar la tarea y esta se encarga de adquirir la información y completar el procesamiento de la misma.

El tiempo que transcurre entre la generación de la interrupción y el momento en el cual esta es atendida se llama latencia de interrupción. El inverso de esta latencia es una frecuencia llamada frecuencia de saturación, si las señales que están siendo procesadas tienen una frecuencia mayor a la de saturación, el sistema será físicamente incapaz de procesarlas. En todo caso la mayor frecuencia que puede procesarse es mucho menor que la frecuencia de saturación y depende de las operaciones que deban realizarse sobre la información recibida.

2.1.5 Sistema Embebido

La mayor parte de las actividades de control, tales como adquisición de señal, filtrado, procesamiento de datos sensoriales, la planificación de acciones, y el actuador de control, suelen implementarse como tareas periódicas activado en las tasas específicas impuestas por los requisitos de aplicación. Cuando un conjunto T de n tareas periódicas al mismo tiempo tiene que ser ejecutado en el mismo procesador, el problema es comprobar si todas las tareas pueden completar su ejecución dentro de su calendario restricciones [7].

Un sistema integrado, empotrado o embebido es un sistema informático de uso específico construido dentro de un dispositivo mayor. Los sistemas integrados se utilizan para usos muy diferentes a los usos generales a los que se suelen someter a las computadoras personales. En un sistema integrado la mayoría de los componentes se encuentran incluidos en la placa base (la tarjeta de vídeo, audio, módem, etc.).

Dos de las diferencias principales son el precio y el consumo. Puesto que los sistemas integrados se pueden fabricar por decenas de millares o por millones de unidades, una de las principales preocupaciones es reducir los costes. Los sistemas integrados suelen usar un procesador relativamente pequeño y una memoria pequeña para reducir los costes. Se enfrentan, sobre todo, al problema de que un fallo en un elemento implica la necesidad de reparar la placa íntegra.

Lentitud no significa que vayan a la velocidad del reloj. En general, se suele simplificar toda la arquitectura del ordenador o computadora para reducir los costes [2]. Por ejemplo, los sistemas integrados emplean a menudo periféricos controlados por interfaces síncronos en serie, que son de diez a cientos de veces más lentos que los periféricos de un ordenador o computadora personal normal. Los primeros equipos integrados que se desarrollaron fueron elaborados por IBM en los años 1980.

2.1.6 Descripción del procesamiento de Señales

¿Qué significan el "procesamiento de señal"?, realmente es un conjunto de algoritmos para el procesado de señal en el dominio digital, hay equivalencias analógicas en estos algoritmos pero procesarlos digitalmente ha sido provisto para ser más eficiente. Esta ha sido una tendencia de muchos, muchos años los algoritmo para procesamiento de señales son los bloques básicos para la

construcción de muchas aplicaciones en el mundo, desde teléfonos celulares hasta reproductores de MP3, cámaras digitales, entre otros [1].

El procesamiento digital de señales (en inglés digital signal processing, DSP) es un área de la ingeniería que se dedica al análisis y procesamiento de señales (audio, voz, imágenes, video) que son discretas. Aunque comúnmente las señales en la naturaleza nos llegan en forma analógica, también existen casos en que estas son por su naturaleza discretas, por ejemplo, las edades de un grupo de personas, el estado de una válvula en el tiempo (abierta/cerrada), etc.

Cada vez tiene mayor importancia en la ingeniería el procesado de señales digitales, en ella, sobre todo, se ha catalizado por el aumento de potencia de los ordenadores con una bajada drástica de los precios. Tanto es así que hay en el mercado desde hace algunos años procesadores específicos para el procesamiento digital, denominados DSP. Pero hasta hoy en día, los diseños de procesadores genéricos son ampliados en registros e instrucciones para poderlas destinarlas a tareas de procesamiento de señales, sea el caso más llamativo del conjunto de instrucciones MMX insertadas en la familia INTEL a partir del mítico PENTIUM.

Por sólo hablar de las aplicaciones industriales que se sustentan en procesamiento digital de señales se podrían citar algunas tales como:

Instrumentación electrónica:

- Filtrado de señales
- Osciloscopios digitales
- Analizadores de espectro

Electrónica de Potencia:

- Señales de disparo sobre SCRs, IGBTs, MOSFET, etc.

Control:

- Reguladores discretos
- Controladores de robots

Procesamiento de imágenes:

- Filtrado de imágenes
- Reconocimiento de Formas
- Compresión y descompresión de imágenes

Procesamiento de sonido

- Identificación de fonemas
- Voz sintética

Se puede procesar una señal para obtener una disminución del nivel de ruido, para mejorar la presencia de determinados matices, como los graves o los agudos y se realiza combinando los valores de la señal para generar otros nuevos.

3. METODOLOGIA

3.1 Introducción

Los RTOS, son una herramienta bastante fácil de utilizar, la metodología de trabajo no es complicada, es más importante programar una parte crítica adecuada para la función que se necesite, además de cuestionar los demás elementos que entran en función para un fin.

Este capítulo muestra un panorama general de la investigación, así como las etapas que se desarrollaron en el proyecto a fin de obtener el mejor funcionamiento en el desarrollo del mismo.

Describe las funciones de cada etapa, y las demandas que se debe hacer a cada una, así como sus características y los materiales que se pueden utilizar en el desarrollo de las mismas, explicara el uso de cada dispositivo que se empleó, haciendo hincapié al uso de dispositivos diferentes con el fin de encontrar el más apto para el funcionamiento.

3.2 Esquema general del proyecto

En esta sección se describe el esquema general con el que se trabajó, de igual manera explica la función que ejecuta cada bloque, sus entradas y salidas. Para de esta manera permitir el entendimiento de los pasos que se siguieron posteriormente. El diagrama a bloques que describe el proyecto se muestra en la Figura 3.1.

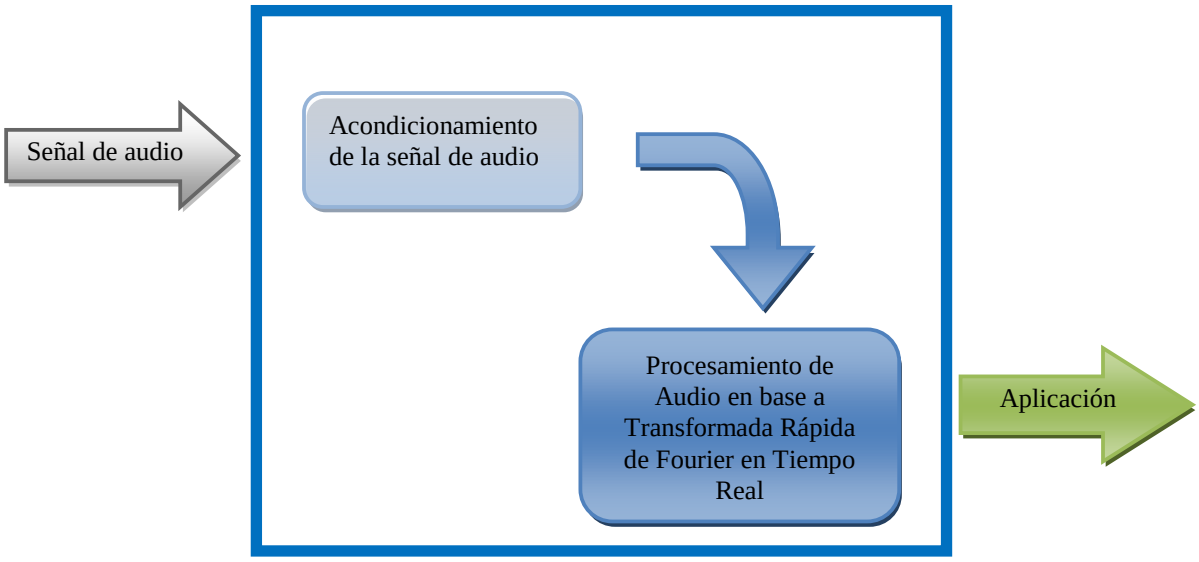

Figura 3.1. Esquema general del proyecto

Es fácil apreciar que el esquema general solo nos muestra 2 etapas en las cuales solo una de ellas se enfoca al tema que se trata en este documento. La etapa de acondicionamiento dependerá de las características de la aplicación, haciendo especial énfasis a las frecuencias que se desean filtrar, ya sea el caso de audio en específico o a voz.

3.2.1 Etapa de acondicionamiento de la señal

Empezando por la señal de entrada sabemos que es una señal de audio, y como tal debe de pasar por una etapa de acondicionamiento para que de esta manera la segunda etapa pueda realizar el procesamiento del mismo. Para proseguir vale la pena recordar que se está trabajando con un microcontrolador, para explicar que el acondicionamiento de la señal deberá cumplir las demandas de los canales analógicos del microcontrolador, la más importante de ellas será que realice un nivel de DC al voltaje de la señal de audio (Figura 3.3), ya que el microcontrolador no puede recibir señales negativas a la entrada de los canales analógicos, no en

este caso. La segunda demanda del microcontrolador hacia la etapa de acondicionamiento es con respecto a los niveles de voltaje de la señal, dependiendo del tipo de microcontrolador, por ejemplo un PIC 18F4550 puede recibir niveles de voltaje de 0 a 5 volts en cualquiera de sus canales analógicos, sin embargo un PIC 24FJ64GA102 preferentemente debe trabajar con niveles de voltaje de 0 a 3.2 volts en sus canales analógicos, aunque también podría resolverse ese problema programando al PIC para que trabaje con un voltaje de referencia. Pero lo más conveniente para esta situación es tratar de que el microcontrolador este enfocado solo en la parte crítica del procesamiento del audio.

En la Figura 3.2 se puede apreciar las funciones principales de la etapa de acondicionamiento.

Es importante también tomar en cuenta que la etapa de acondicionamiento está encargada de realizar el filtrado de la señal, se debe tener en consideración el tipo de aplicación para el cual este enfocado el procesamiento de audio, por ejemplo si la aplicación está determinada para solo utilizar el espectro de voz entonces se debe de aplicar un filtro pasa bandas que permita el paso de frecuencias mayores de 3Khz pero menores a 10Khz, puesto que no tienen información relevante (con respecto a la voz) más allá de estos rangos.

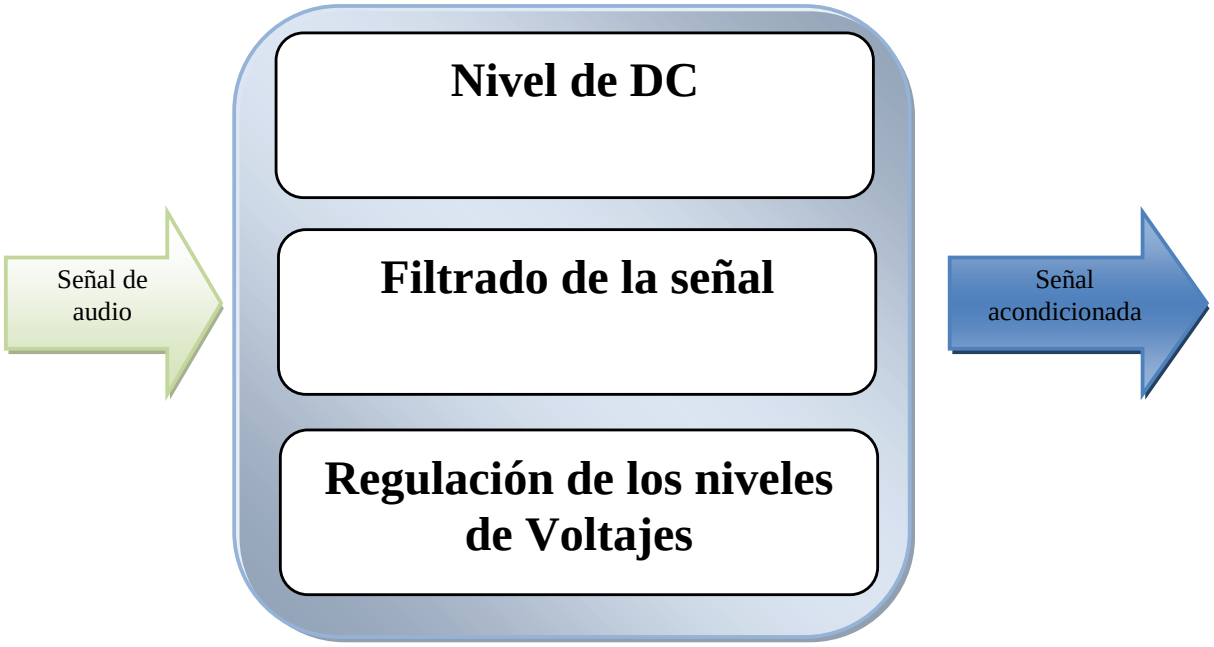

Figura 3.2. Esquema de la etapa de acondicionamiento.

Dependiendo del tipo de aplicación se debe idear una etapa de acondicionamiento, sin embargo en este caso es conveniente usar una etapa que cumpla con las características planteadas en la Figura 3.2.

En la Figura 3.3 se muestra una señal de una secuencia musical con un nivel de DC de 1.5 volts, esto solo como referencia para apreciar lo que es requisito para entrada a los canales analógicos del microcontrolador.

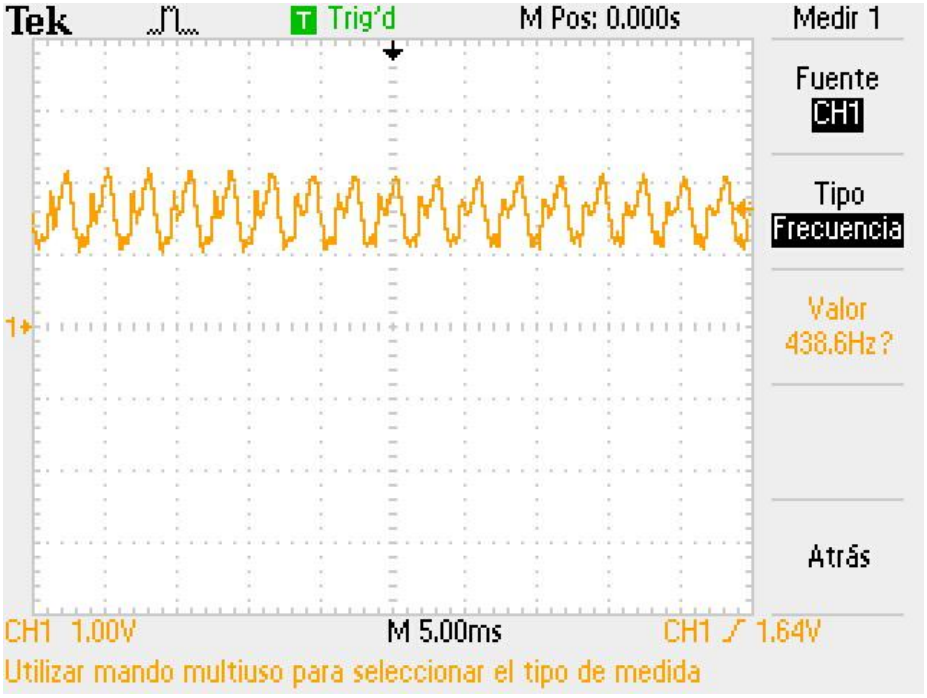

Figura 3.3. Señal de secuencia musical con un nivel de DC de 1.5 volts.

3.2.2 Etapa del Procesamiento de audio

Como anteriormente se había mencionado el procesamiento de señales es la rama de la ingeniería que se encarga de analizar y procesar señales discretas como: audio, voz, imágenes y video. Sabemos que en la actualidad existen métodos para el procesamiento de las mismas.

Inmediatamente al trabajar con procesamiento de señales pensamos en usar la Transformada Rápida de Fourier, puesto que en la mayoría de los procesos en los que se pretende manipular señales, es muy práctico además de eficiente y conciso; conciso en el sentido de presentar brevedad y economía con respecto a los recursos que se pueden utilizar.

Claramente para esta etapa se presentan 3 sub-etapas, como se ilustra en la Figura 3.4, que son lo más fundamental, pero es notorio observar que todas ellas deberán trabajar con el Sistema de Tiempo Real empotrado en el

microcontrolador. Por ello de la importancia de reducir la parte critica del programa, además de poder obtener mayor eficiencia en el procesamiento del audio.

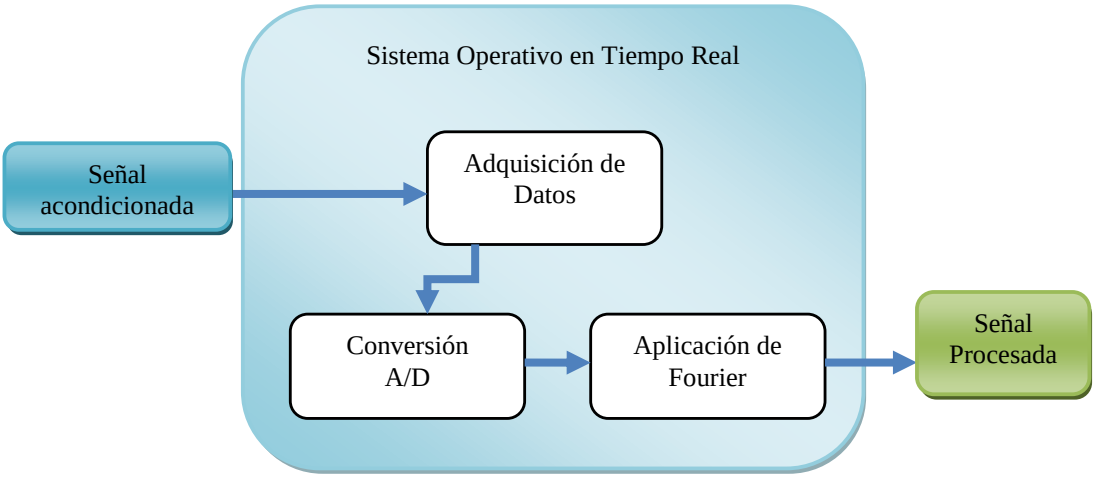

Figura 3.4. Esquema de la Etapa “Procesamiento de audio en base a FFT en tiempo real”.

Es de suma importancia que la señal de entrada a la etapa acondicionada, de caso contrario podrá afectar al dispositivo (PIC), que se encargara de procesarla. Por los motivos que se mencionaron anteriormente con respecto a los requisitos de voltaje de entrada a los canales analógicos.

El primer pasó de un sistema de procesamiento de señales que está recibiendo la información del mundo real en el sistema. Es realizar la transformación de una señal analógica a una representación digital apta para ser transformada por el sistema digital. Esta señal pasa a través de un dispositivo llamado un convertidor analógico-digital (A / D o ADC). El ADC convierte la señal analógica a una representación digital por muestreo o de medición, la señal con una tasa periódica [1]. Cada la muestra se le asigna un código digital (Figura 3.5).

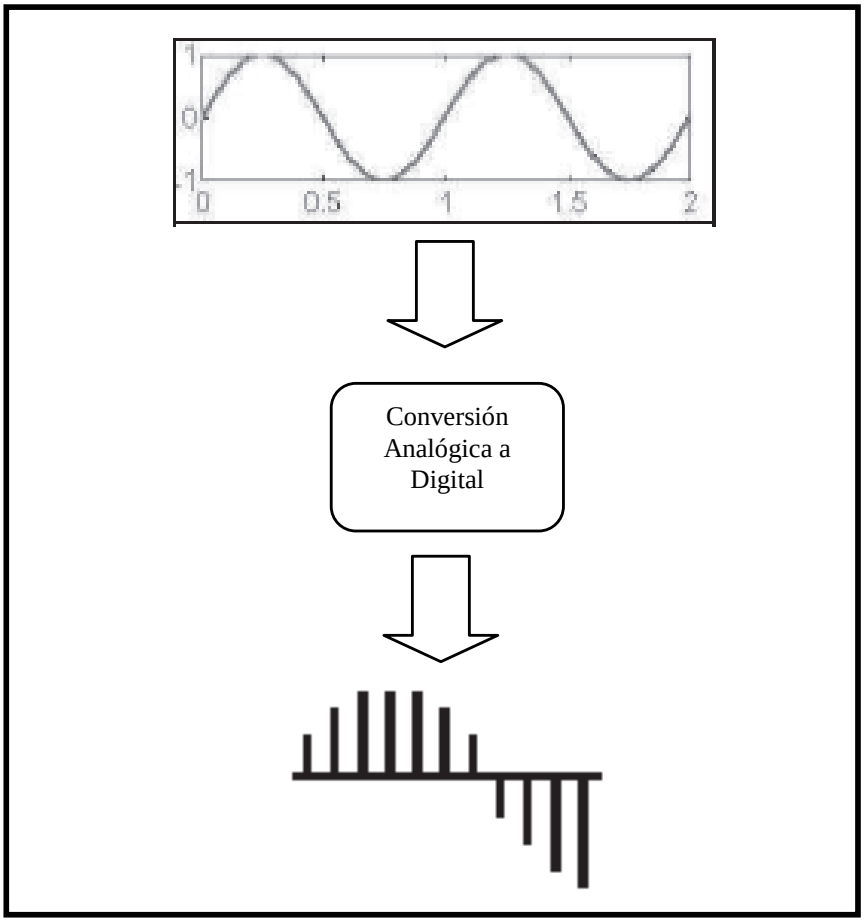

Figura 3.5 Conversión Analógica a Digital para el procesamiento de la señal.

Una vez que la entrada a la etapa “Procesamiento de audio en base a FFT en tiempo real” este disponible y sea introducida en el canal analógico, se deberá digitalizar (Figura 3.5) puesto que hoy en día la mayoría de los procesamientos de audio son de manera digital, el implementar esto en el proyecto permitirá acoplarlo o estandarizarlo a otros sistemas; otra justificación seria que el digitalizar las señales permite una mayor facilidad de procesamiento de las mismas [1]. La señal digitalizada pasara por la implementación del algoritmo de FFT, dicho algoritmo estará embebido o empotrado en el microcontrolador. La salida del algoritmo de Fourier no estregara valores correspondientes a su espectro, que a su vez estos datos serán contemplados para utilizarlos con un propósito en específico.

Al inicio de este capítulo se explicó que dependía mucho la aplicación a la que se enfoque el procesamiento de audio. Debido a la demanda realizada, la parte del Espectro de Fourier estará enfocado en ciertas tareas, mas sin embargo, sea cual sea la aplicación, se deberá tomar del espectro y realizar comparaciones con los datos que nos entrega para así decidir las acciones a ejecutar con respecto a la aplicación.

Por ejemplo si se desea tomar el procesamiento de audio para realizar un afinador de instrumentos, entonces se deben establecer ciertos parámetros en el programa implementado para que sean detectados por la entrada con respecto a tonos erróneos. Aprovechando el ejemplo del afinador instrumental, en la parte de acondicionamiento de señal, (como anteriormente se menciona que depende de la aplicación), se tomaría en cuenta implementar un filtro que elimine el ruido y las frecuencias de la voz, para que solo se pueda apreciar las notas musicales.

Esta etapa dependerá en un ligero porcentaje del objetivo final, mas sin embargo se debe tener en cuenta la condición de manejar en Tiempo Real el procesamiento de la señal. El Tiempo Real es fundamental dentro del proyecto debido a su peso en la comprobación de la hipótesis.

La parte del Sistema Operativo en Tiempo Real dentro de la etapa permitirá mejoras en la eficiencia del procesamiento de audio, la intención es que al tener un microcontrolador con un RTOS embebido se pueda mejorar en cuestión a tiempo y recursos (del microcontrolador).

El RTOS tiene un alto peso dentro de la investigación, empezando por la parte de comprobar los objetivos planteados.

En [6] se describe como un RTOS permite la optimización de múltiples tareas complejas, así como mostrar ejemplos los cuales permiten apreciar la facilidad de implementar el código con RTOS.

En la Figura 3.6 se aprecia la comparación de un PID programado en un microcontrolador sin RTOS y con RTOS.

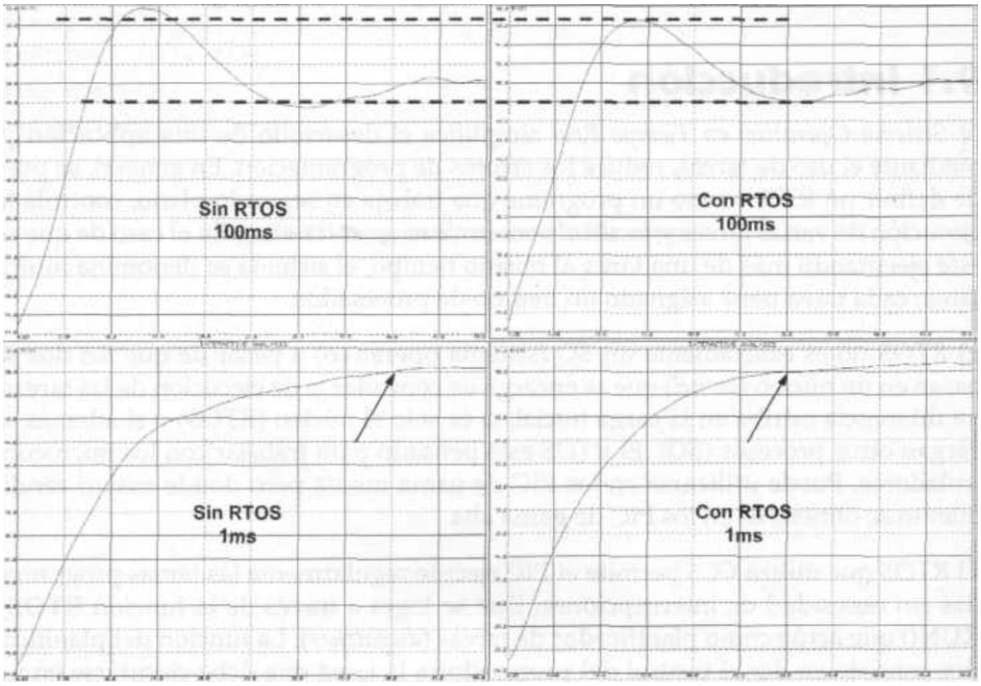

Figura 3.6. Respuesta del PID con y sin RTOS. [6]

Se puede observar claramente que el programa con RTOS permite generar mejoras en la respuesta de las tareas, la Figura 3.6 muestra como la respuesta del PID se establece más rápido, tal vez solo unos milisegundos más rápido, pero se debe estar consiente que en la mayoría de los procesos, el ganar unos milisegundos permite tener mejores resultados a largo plazo. El código del PID implementado se muestra en el anexo del escrito.

A través de investigaciones se ha comprobado que un RTOS permite generar mayor eficiencia en cualquier proceso, simplifica el desarrollo de una aplicación y,

mediante el uso de tareas, reduce los errores de programación. En general, se puede definir un RTOS como un programa que trabaja en segundo plano, controla la ejecución de varias tareas y facilita la comunicación entre ellas. En el caso de que se esté ejecutando más de una tarea al mismo tiempo, el sistema se denomina multitarea; cada tarea tiene asignado un tiempo del procesador.

En esta investigación conociendo la cantidad de memoria que consume la implementación de una FFT es más que ideal el uso de un RTOS.

3.3 Materiales y Recursos Empleados

El criterio de selección de los recursos empleados permite ampliar el panorama de los usos y aplicaciones para este proyecto, principalmente por los ahorros que se pudiesen generar.

Inicialmente se debe plantear la idea de que cada etapa constará de materiales diferentes.

Empezando por la etapa de acondicionamiento, que en si no es un tema que se tratara a profundidad ya que no es parte de la investigación, se deberá tomar en cuenta que estará constituido por tres partes fundamentales (de acuerdo a lo planteado en la Figura 3.2). Vale la pena recordar cuales de estas partes son:

- Filtrado.
- Nivel de DC
- Regulación de niveles de voltaje (amplificación).

Para la decisión de los materiales a emplear, es preferible empezar por la parte del filtrado de la señal, para así tomar el criterio de los otros requisitos, se pueden tomar los siguientes criterios:

- Empezando por plantear si sería un filtro digital, pasivo o activo. Desde este punto podemos descartar una opción debido a que un filtro digital causaría mayor cantidad de tareas en el microcontrolador y la intención es evitar usar los recursos del procesador en tareas que se puedan realizar externamente.
- Entonces si el filtrado se debe hacer pasivo o activo, se tendría que pensar en lo más flexible para el usuario, empezando por el espacio y la exactitud que se podría esperar de dicho filtro. En cuestión de espacio se plantearía un filtro pasivo pero la desventaja seria la cuestión de la exactitud de los dispositivos, si se plantea usar un filtro activo entonces el espacio seria lo que provocaría dudas, en este punto dependerá mucho del usuario.
- Para fines prácticos se pensaría en un filtro activo, además de existir integrados que facilitarían el diseño tanto del filtro, los niveles de DC (sumador), y la regulación de voltajes (amplificación de la señal) en un solo integrado, como es el caso del TL084 solo por presentar un ejemplo.

Los anteriores criterios son solo como recomendación, mas sin embargo no se podrían usar como las referencias más aptas para el diseño de la etapa de acondicionamiento.

En lo que se realizara especial énfasis es un la etapa de procesamiento (que es la justificación de la investigación). A continuación se explicara por qué usar cada herramienta en el desarrollo del proyecto.

El lenguaje de programación elegido para este proyecto ha sido el C, se ha elegido por su facilidad para la realización de tareas complejas, cada vez los compiladores de C están más optimizados, de acuerdo a [10] y [11], llegando en ocasiones a generar códigos tanto o más compactos que los desarrollados puramente en ASM.

Una vez determinado el lenguaje de programación con el que se va a trabajar, en este caso el C, el siguiente paso es seleccionar el compilador. Podemos encontrar una amplia gama de compiladores, como puede ser el C18 de la empresa Microchip y otros como PICC de la empresa Hi-Techó CSCC de la empresa CCS. El compilador elegido es el de la empresa CCS, en concreto el PCWH, soporta a toda la familia de PICs y también los DsPIC. Ha sido elegido por su gran compactación del código generado y por su alta difusión en el mundo de la enseñanza.

Otra característica importante del compilador es que esta proporcionado con comandos, librerías y ejemplos para trabajar un Sistema Operativo en Tiempo Real, esto permitirá tener acceso a comandos fáciles para el uso de un RTOS, el compilador CCS tiene una variedad de comandos que introducidos en el programa son capaces de mejorar el procesamiento de audio, incluso el uso de 2 o 3 comandos ayuda mucho a que el microcontrolador resulte ser más eficiente, también otorgara a cada tarea la posibilidad de “dormirse”, mientras que otra tarea consumen todos los recursos del procesador.

El RTOS que utiliza CCS permite que el PIC ejecute regularmente las tareas programadas sin necesidad de interrupciones. Este se logra a través de la función RTOS_RUN() que actúa como planificador de tareas. La función del planificador consiste en dar el control del procesador a la tarea que debe ejecutarse en un momento dado. Esto permitirá al proyecto proporcionar el rendimiento requerido.

En cuestión a las herramientas de ayuda como simuladores, es conocido por cualquier estudiante, Ingeniero, Maestro o Doctor que un simulador no es exacto en la gran mayoría de los casos, más sin embargo, es una herramienta útil que permite ampliar el panorama de los resultados; el simulador “Isis” de Proteus, permite simular la programación del microcontrolador con otros dispositivos conectados.

La falta de dominio de otros softwares para programación de microcontroladores, y los recursos flexibles que se manejan en los programas antes mencionados, elimina la opción de usar a un proveedor que no sea Microchip. Además de tomar la decisión que por su interacción con el compilador, la herramienta de simulación será “Isis”.

Una vez que se ha planteado las herramientas para generar el procesamiento de audio, queda pendiente el definir la cuestión del microcontrolador.

Microchip tiene diferentes tipos de familias de microcontroladores, por ejemplo: los estudiantes de preparatorias técnicas comienzas el uso de los mismos con PICs de la familia 16, estudiantes de la carrera de ingeniería realizan proyectos complejos con PICs de la familia 18. De hecho los micros de la familia 18 son excelentes dispositivos por sus características, en la Tabla 3.1 se mencionan algunas características importantes del PIC 18F4550 (uno de los mejores de la familia 18), además de ser dicho dispositivo uno de los más usados para fines prácticos; más sin embargo la investigación realizada requiere de ciertos recursos que no se pueden apreciar en la siguiente tabla.

Dispositivo	Memoria del programa		Memoria de los Datos		I/O
	Flash (bytes)	Número de Instrucciones por ciclo	SRAM (bytes)	EEPROM (bytes)	
PIC18F4550	32K	16384	2048	256	35

Tabla 3.1. Principales características del microcontrolador PIC18F4550.

En realidad el microcontrolador 18F4550 es excelente dispositivo, en la tabla 3.1 solo se muestran unas de sus múltiples características, pero como anteriormente se mencionó este microcontrolador carece de un requisito esencial para el desarrollo del procesamiento, la memoria RAM es insuficiente para el propósito que se necesita.

Es inevitable que si se desea realizar el procesamiento de audio a través de un algoritmo de la Transformada Rápida de Fourier implementado en un microcontrolador, el código consuma demasiada memoria RAM. Ahora, se está hablando solo de la parte del algoritmo de Fourier, ¿Qué pasara cuando se implemente el Sistema en Tiempo Real?

El RTOS es una herramienta bastante eficiente del CCS, pero el costo es notorio, la cantidad de memoria que consume dependiendo de las funciones que se empleen podría estar desde el 20% para tareas simples, hasta 60% en tareas más complejas. Por ello lo mejor es pensar en cuidar la parte crítica del programa (como se ha venido mencionando anteriormente), aunque en este caso sería bastante complicado que la parte critica del programa sea reducida o que se eliminen operaciones.

La necesidad de mayor cantidad de memoria elimina la posibilidad de usar el microcontrolador 18F4550. Aunque en la familia de los PICs 18 existe el 18F4620, el cual es un microcontrolador de alto desempeño y es uno de los que maneja la mayor cantidad de memoria de esta familia.

Sus principales característica se muestran en la Tabla 3.2.

Dispositivo	Memoria del programa		Memoria de los Datos		I/O
	Flash (bytes)	Número de Instrucciones por ciclo	SRAM (bytes)	EEPROM (bytes)	
PIC18F4620	64K	32768	3986	1024	36

Tabla 3.2. Principales características del PIC18F4620.

Si se comparan las tablas del 18F4550 con el 18F4620, se notaría a simple vista que el microcontrolador 18F4620 es mejor con respecto a las memorias y las entradas y salidas, aunque cada uno tiene características que el otro no posee, para la investigación es más adecuado el PIC18F4620.Se podría pensar en usar

dicho microcontrolador, sin embargo existe la cuestión de la cantidad de muestras que se pueden obtener al momento de adquirir la señal por el canal analógico.

Si se decide usar cierta cantidad de variables, se debe tener en cuenta la capacidad del PIC de ofrecer el rendimiento requerido. Por ejemplo, si se utiliza el PIC18F4620 y se programara una captura de 128 datos, el microcontrolador sería capaz de soportarlos incluso no habría problema con el procesamiento de la señal, el contra de usar dicha cantidad de variables residiría en poder obtener la señal analógica lo más completa posible, puesto que la falta de información generaría falta de cálculos en la Transformada de Fourier, incluso que el espectro de Fourier sea inexacto por mal interpretar la señal de audio. Ahora bien, si se aumentara la cantidad de variables al doble que se planteó, se podría pensar que no habrá problema con las señales obtenidas, mas sin embargo estaría la cuestión de la capacidad del PIC para procesarlas, ya anteriormente se mencionó que el PIC18F4620 es un microcontrolador que posee una de las más extensas memorias de la familia 18 de Microchip, mas sin embargo el solicitar que procese 255 datos saturaría la memoria de dicho microcontrolador. Este motivo en una desventaja para poder experimentar con los microcontroladores de la familia 18.

Anteriormente se mencionó que Microchip tenía una extensa gama de microcontroladores, entonces esto sería un motivo para empezar la búsqueda de algún dispositivo que cumpla con los requerimientos de la investigación.
En la Tabla 3.3 se muestran las características principales del PIC24HJ256GP206, el cual nos permite trabajar con mayor memoria y a velocidades favorables.

Se podría pensar que posee demasiados recursos que solo serán desperdiciados, como las entradas y salidas, los módulos, PWM, canales analógicos, entre otros; mas sin embargo el uso de los recursos ofrecidos dependerá de la aplicación que se le exija ejecutar.

Dispositivo	Flash (bytes)	Millones de Instrucciones por segundo MIPS	SRAM (bytes)	Resolución de canales Analógicos (bits)	I/O
24HJ256GP206	256 K	32768	16 K	10-12	53

Tabla 3.3. Principales características del PIC24HJ256GP206.

Si se compara este dispositivo con los anteriormente mencionados, se podría pensar que no justifica su uso y que tal vez con los microcontroladores de la familia 18 se pueda lograr un procesamiento de audio apto para fines prácticos, esta decisión dependerá mucho de la aplicación que se maneje, posiblemente existan situaciones que no requieran un 100% de exactitud, pero la intención es plantear y exponer los resultados que se obtuvieron a través de la investigación, para que futuras generaciones la utilicen como base a otros proyectos.

En el capítulo 4 se exponen los resultados obtenidos al utilizar los diferentes dispositivos que se plantearon, él porque de los comentarios que se mencionaron anteriormente y parte de la metodología que se explicó en este capítulo. Se apreciara con mayor detalle que vale la pena trabajar más sobre este proyecto y que puede ser utilizada para fines prácticos y proyectos más complejos.

4. RESULTADOS Y DISCUSION

4.1 Introducción

En este capítulo se presentan los detalles más importantes de la experimentación realizada así como los resultados obtenidos. Esto con el fin de que el lector tenga las herramientas necesarias para futuras implementaciones.
Se ilustran las gráficas del espectro de Fourier obtenidas desde el microcontrolador, así como los circuitos desarrollados tomando en consideración las cuestiones planteadas en el capítulo anterior.

De igual manera se comparan los resultados con los objetivos y las propuestas de la investigación, a fin de eliminar fallos en otras aplicaciones.

4.2 Hardware

En el desarrollo del hardware se dividió en 2 etapas fundamentales, tal como se mencionó en el capítulo 3.

- Acondicionamiento de la señal.
- Procesamiento de audio en base a Transformada Rápida de Fourier en Tiempo Real.

En la Figura 4.1 se muestra un diagrama de la “estación” con la que se realizó pruebas de los códigos de Fourier. A continuación se describe el funcionamiento de cada parte.

- Inicialmente se recibe la señal de audio la cual se va a tratar, esta señal entra directa a la amplificación, el dispositivo se configuro para obtener una amplificación de 7.5.

- Una vez que la señal fue amplificada pasa por el filtro pasabanda para eliminar ciertas frecuencias que se puedan mostrar como ruido y vibraciones.

- Por ultimo en la etapa de acondicionamiento la señal pasa por un sumador para obtener un nivel de DC de 1.5 volts, como anteriormente se mencionó esto es para proteger al µC de voltajes negativos.

- Una vez que la señal de audio ha pasado por toda la etapa de acondicionamiento esta entrara directamente al µC, será adquirida por uno de los canales analógicos para posteriormente ser procesada por el código de Fourier.

- El código de la FFT entregará los valores del espectro para que estos datos sean enviado vía serial a una interfaz en una PC.

- Toda la etapa del procesamiento se encontrara trabajando en el Sistema Operativo en Tiempo Real, este sistema operativo es proporcionado por el mismo compilador (CCS).

- Finalmente en la PC se apreciaran los datos enviados por el microcontrolador, ya sea de forma gráfica o imprimiendo los valores.

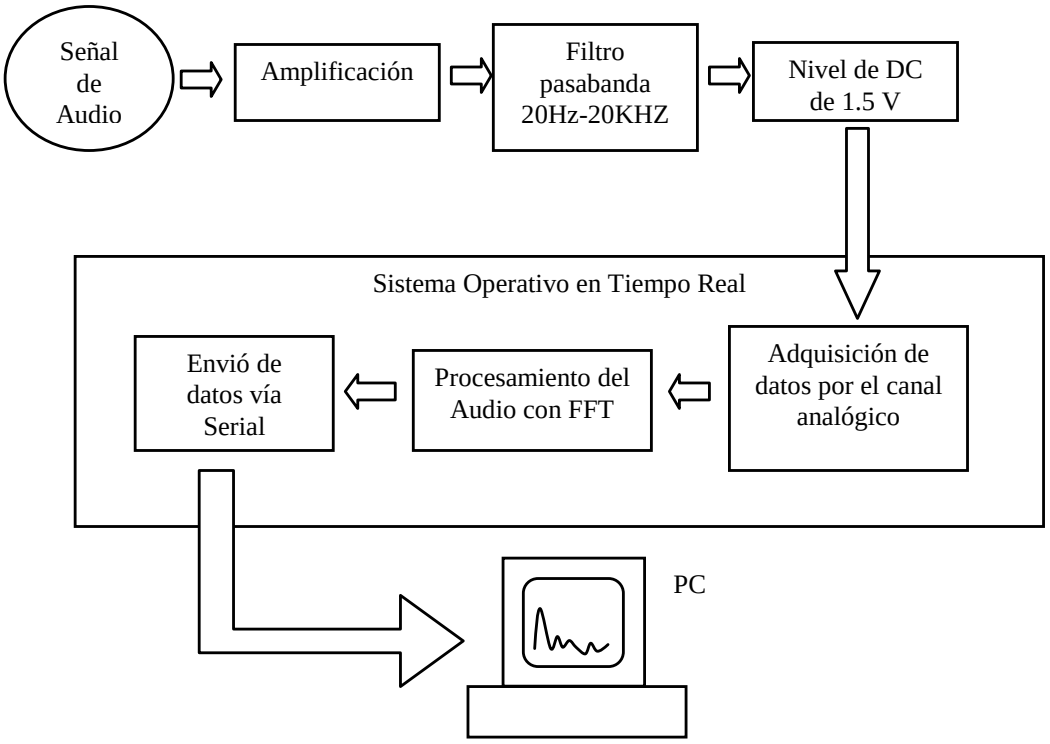

Figura 4.1. Diagrama general de la estación para pruebas.

En la imagen 4.2 se puede apreciar el circuito de acondicionamiento de la señal, para su elaboración se utilizó un TL084, la razón para el uso de este integrado es para reducir espacio en la etapa de acondicionamiento, ya que posee 4 amplificadores operacionales, y esto permite implementar el amplificador, el filtro pasabanda y el sumador en un solo integrado.

Se alimenta con +15 y -15 Volts, pero la tierra es preferible que se conecte en común al circuito del µC y del circuito de la conexión serial (max232).

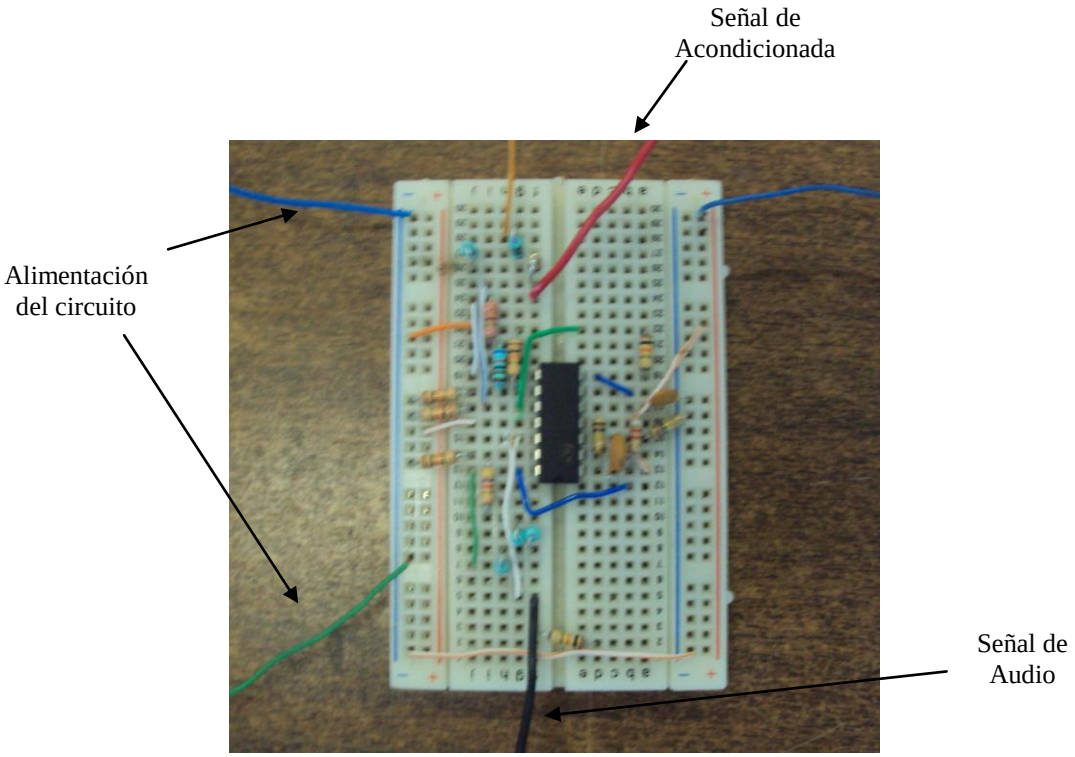

Figura 4.2. Circuito de la etapa de acondicionamiento.

En la Figura 4.3 y 4.4 se puede observar el esquemático realizado de la etapa de acondicionamiento, en la imagen 4.3 se aprecia una etiqueta de salida denominada “X1”, esta salida entra a la parte del sumador en la Figura 4.4.

Estas imágenes permiten representar lo extenso que hubiese sido el circuito de acondicionamiento de la señal, si se empleara el amplificador operacional más común (LM741), ya que cada uno debe ser alimentado y se desperdiciaría más espacio.

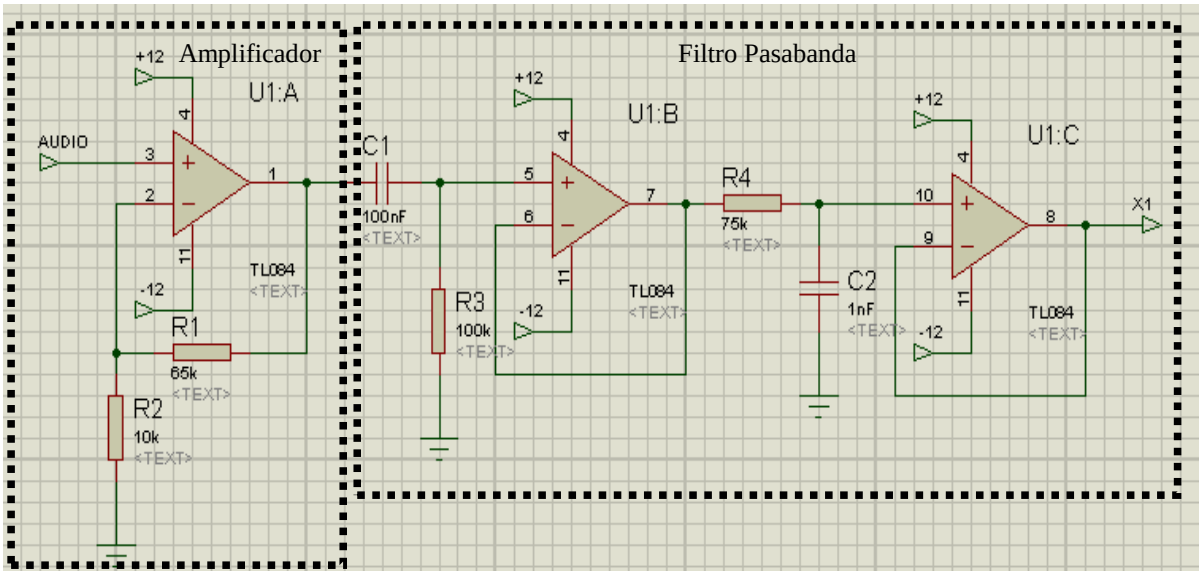

Figura 4.3. Esquemático del Amplificador y Filtro.

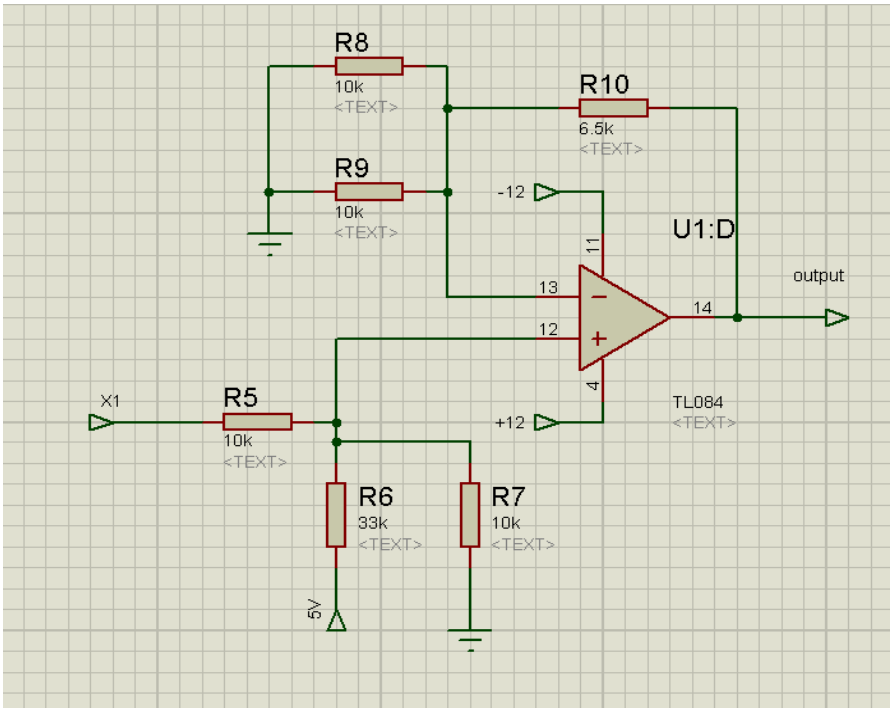

Figura 4.4. Esquemático del Sumador.

En la Figura 4.5 se aprecia la estación de trabajo, en realidad no es muy extensa, sin embargo satisface las necesidades de la investigación.

En la fotografía se puede apreciar que la señal de audio se toma directamente de la computadora portátil, misma computadora que se emplea para adquirir los datos del espectro de Fourier. Los datos como anteriormente se menciona son enviados vía RS232, para ello se ocupa un adaptador serial-usb, para que de esta manera puedan observarse.

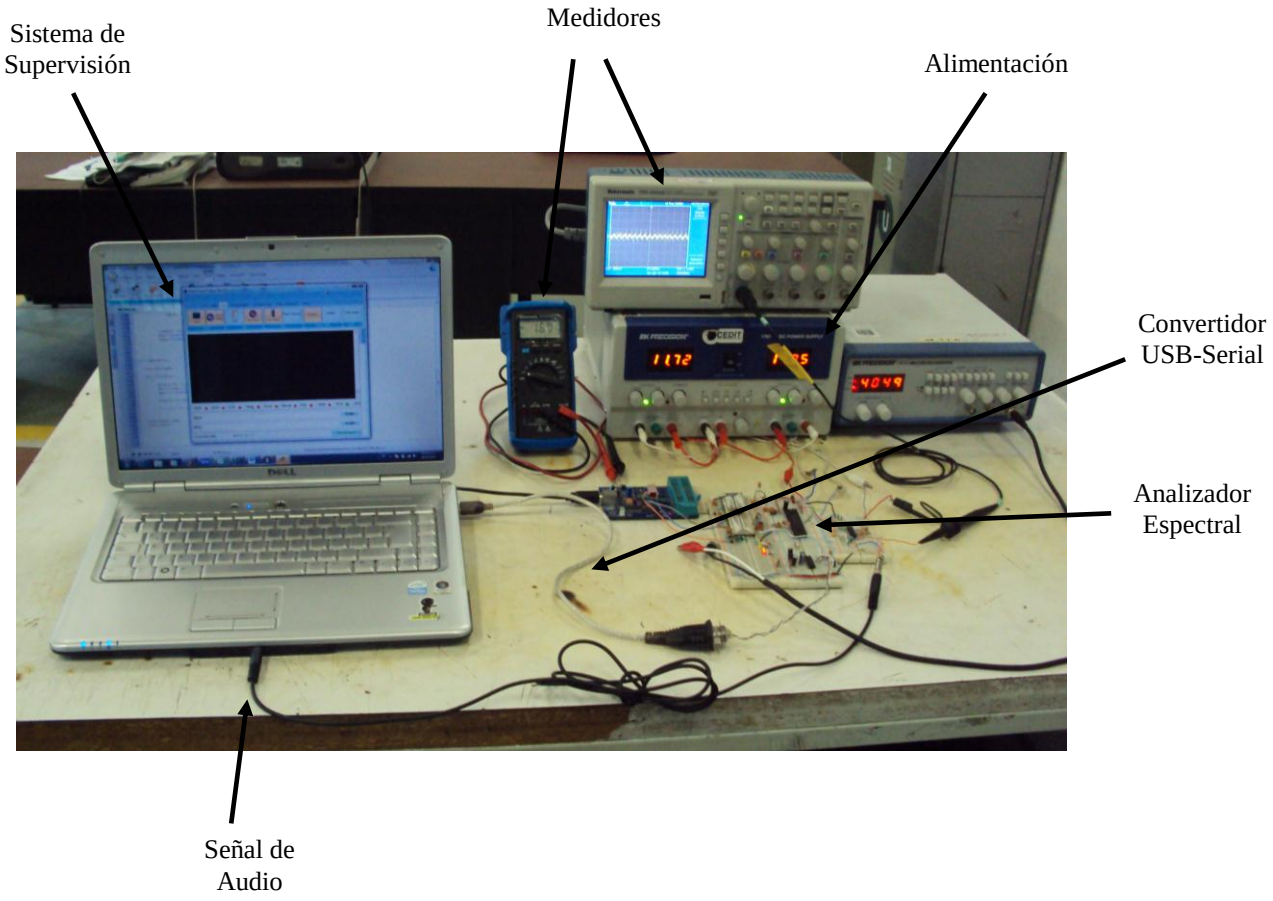

Figura 4.5. Fotografía de la estación de trabajo desarrollada.

4.3 Software

En la parte del software se utilizaron múltiples recursos, desde la producción del audio, hasta el muestreo del espectro de Fourier. En esta sección se describe cada uno y la función que desempeño.

Inicialmente para obtener la señal de audio, se desarrolló un Patch[1] en el software Max 5.1, esta aplicación permitiría la generación constante en un ciclo infinito de determinadas notas musicales. Las notas musicales permitirán observar el espectro de Fourier determinado, si se hubiese utilizado audio más complejo causaría mayor dificultad en la comprobación del correcto funcionamiento de la Transformada Rápida de Fourier.

En la Figura 4.6 se puede observar un Patch elaborado para la generación de notas musicales.

En la parte derecha de la imagen se aprecia una tabla la cual muestra de manera gráfica las notas que reproducirá el Patch. La ventaja de esta aplicación es que se pueden modificar las notas establecidas al mover los segmentos de la tabla.

[1]Patch: Nombre que recibe la aplicación desarrollada en el software Max 5.1.

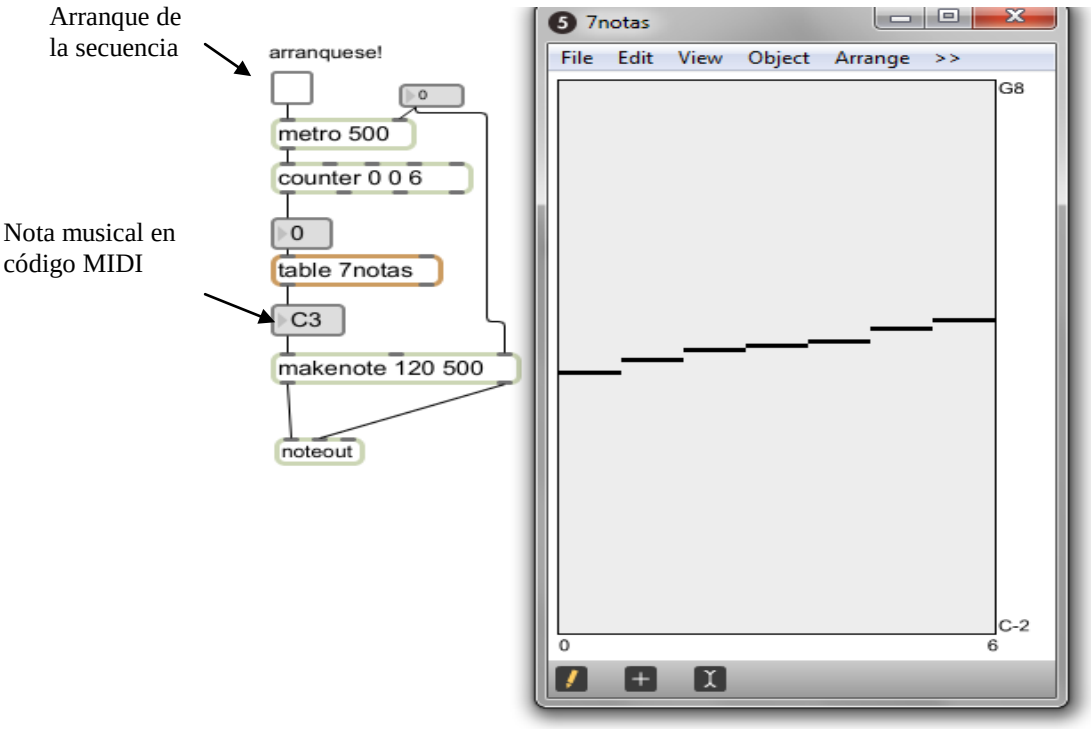

Figura 4.6. Patch implementado para producir notas musicales.

Para apreciar las respuestas de Fourier, después de que el µC realiza el procesamiento estos datos son enviados a la PC vía RS232, estos datos pueden apreciarse en una aplicación desarrollada en C# (Figura 4.7), en la imagen se muestra una ventana cuya función principal es graficar los valores adquiridos por el puerto.

Cabe mencionar que la aplicación no se desarrolló específicamente para este proyecto, es una interfaz visual aportada por otros proyectos, motivo por el cual se puede visualizar datos extras a la aplicación, por ejemplo en la parte superior muestra varias etiquetas que dicen ADC1, ADC2, etc. Solo se tomaron en cuenta los valores que desplegados.

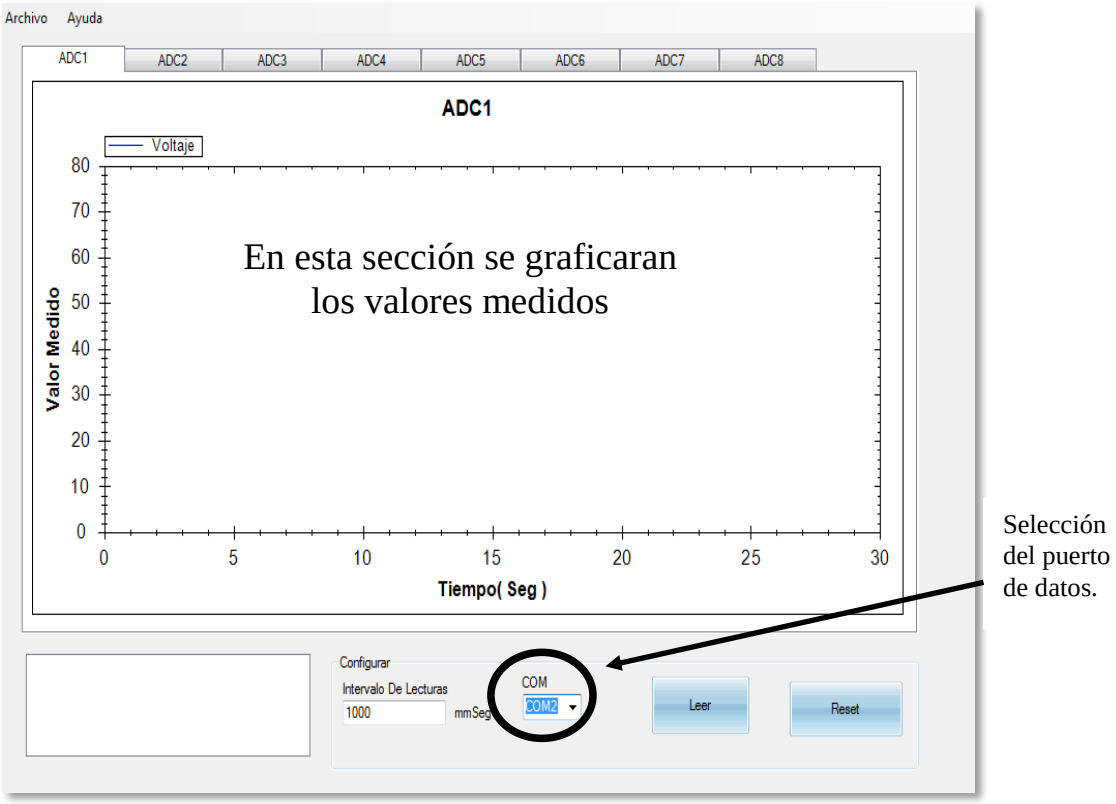

Figura 4.7. Interfaz gráfica desarrollada en C#.

Al principio de las pruebas no se utilizaba la interfaz de la figura 4.7, del mismo compilador CCS es posible obtener una terminal, que interactúa directamente con el puerto Serial, la desventaja es que solo muestra los valores fijos, pero para pruebas es ideal y práctica.

En la Figura 4.8 se aprecia la interfaz que proporciona el compilador CCS, en ella solo es necesario configurar los parámetros del puerto a trabajar.

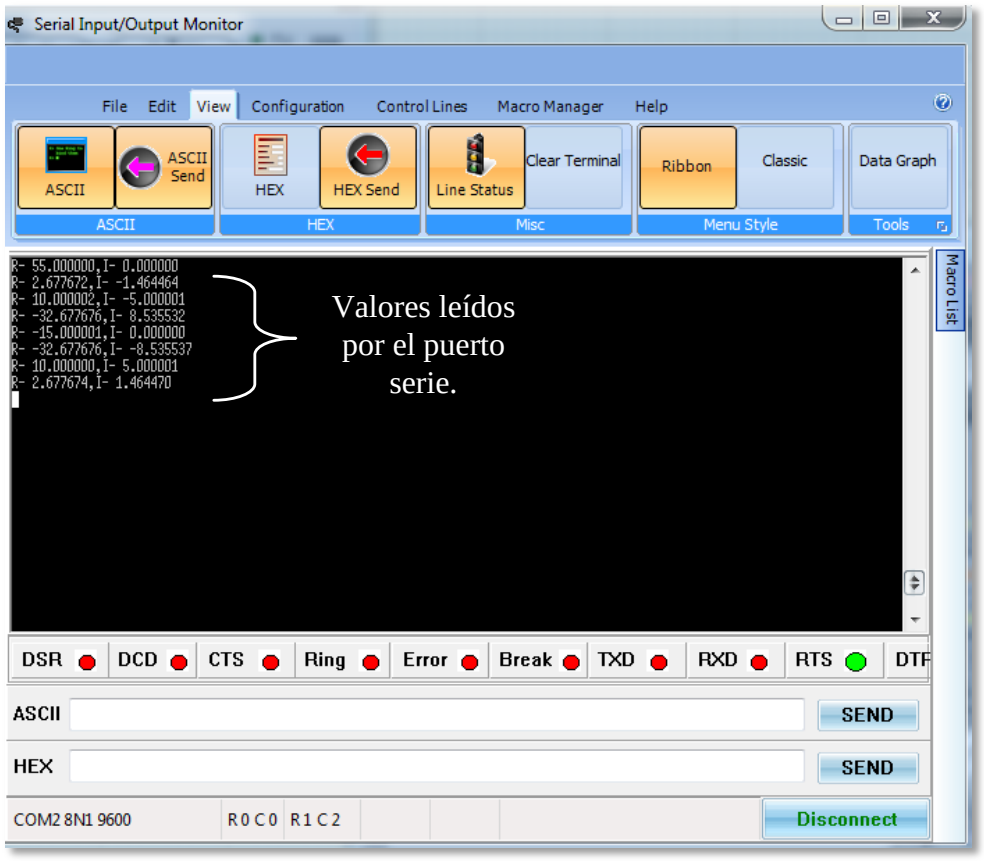

Figura 4.8. Interfaz serial del compilador CCS.

Una de las ventajas de utilizar esta interfaz es que se pueden enviar datos al dispositivo conectado, a diferencia de la aplicación desarrollada en C#, aunque para esta aplicación no se requiere el envió de datos desde la PC.

Para la programación del µC se empleó el software sugerido en el capítulo 3 (CCS), esto por el soporte que proporciona, además de contar con un RTOS.

El programa se implementó inicialmente en un microcontrolador de la familia 18 (PIC18F4620), esto por tener la mayor cantidad de memoria, el programa inicial de prueba solo contenía 8 valores a procesar estos se pueden apreciar en la tabla 4.1, así como también se puede ver el resultado de la Transformada Rápida de Fourier.

Los resultados de la FFT fueron comprobados con el software Matlab (Figura 4.9) para de esta manera tener en cuenta que el código implementado en el microcontrolador sea correcto.

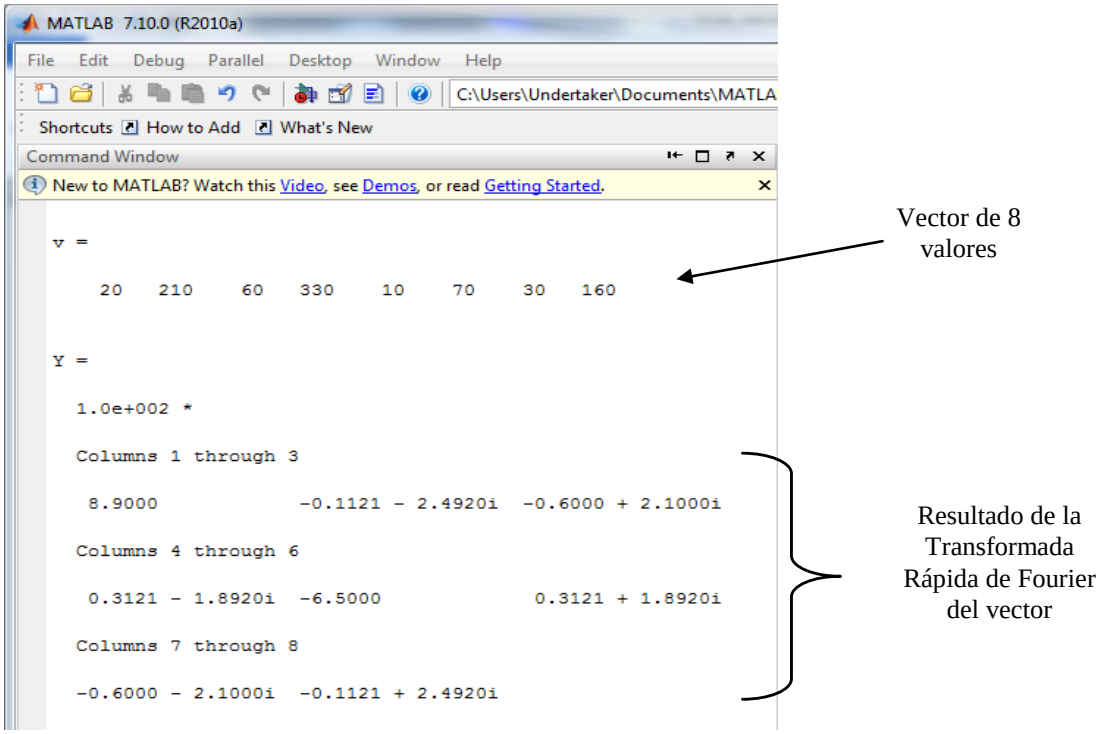

Figura 4.9. Comprobación de la FFT en Matlab.

Comparación entre valores obtenidos								
Valor	20	210	60	330	10	70	30	160
FFT (Matlab)	890 +0i	-11.21 -249i	-60 +210i	31.21 -189.2i	-650 +0i	31.21 +189.2i	-60 -210i	-11.21 +249i
FFT (PIC18)	890 +0i	-11.21 -249i	-60 +210i	31.21 -189.2i	-650 +0i	31.21 +189.2i	-60 -210i	-11.21 +249i
FFT (PIC24)	890 +0i	-11.21 -249i	-60 +210i	-209 +8.78i	-650 -0i	-209 -8.76i	-60 -210i	229 +51.21i

Tabla 4.1. Resultados obtenidos con los microcontroladores usados.

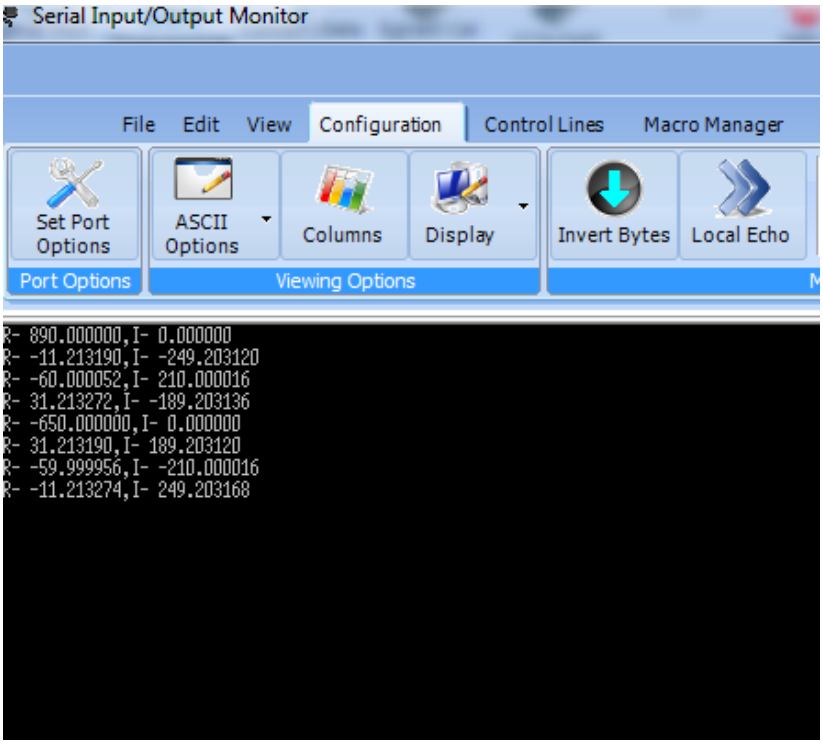

Figura 4.10. Resultados de la FFT implementada en el PIC18F4620.

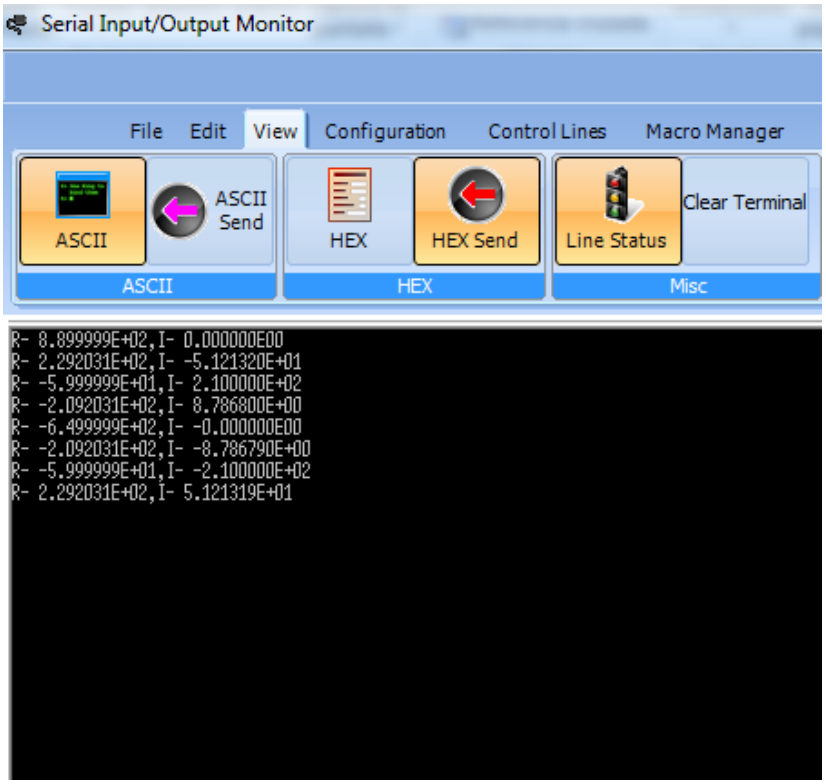

Figura 4.11. Resultados de la FFT implementada en el PIC24HJ256GP206.

Podemos apreciar en la Figura 4.11, que el valor que devuelve la FFT implementada en el PIC24HJ256GP206, no es el valor correcto de acuerdo a lo calculado en Matlab, a pesar que el código sea el mismo a diferencia de los fusibles que se emplean para cada microcontrolador, el resultado sorpresivamente es diferente. Teóricamente no debería tener variaciones puesto que el algoritmo para calcular la FFT es el mismo, esto es un gran problema en vista de lo que se probó en ese momento son solo 8 valores diferentes. Podría pensarse que en vista que ha funcionado con el microcontrolador de la familia 18, lo más factible es eliminar el PIC de la familia 24, pero recapitulando el capítulo 3, lo más apto para el procesamiento del audio, sería un microcontrolador que ofrezca un mayor rendimiento, mayor memoria y mayor cantidad de MIPS (millones de instrucciones por segundo), lo cual era proporcionado por el PIC24HJ256GP206.

Anterior a realizar estas pruebas se comenzó probando programas sencillos en el PIC24, los cuales funcionaron satisfactoriamente. Sería un completo error decir que el PIC no soporta el código o peor aún, decir que el código es erróneo, claramente podemos apreciar que funciona en otros microcontroladores. Al realizar una búsqueda detallada de la falla en el código, se pudo observar que el compilador ejecutaba mal ciertas operaciones sencillas (sumas, restas), después de desglosar de una manera más sencilla cada operación el error seguía presente, este error solo fue eliminado al realizar modificaciones para la corrección de signos negativos, y aunque el programa funcionaba bien dentro del PIC18 en el PIC24 no se obtuvieron los mismos resultados.

Este problema tal vez se deba al compilador, ya que está más enfocado a la programación de los PIC18, no presenta el soporte necesario para una programación más compleja en un PIC24, por lo que definitivamente es mejor utilizar compiladores más apto para la programación de ellos o de los DSPIC.

En vista de que no se pudo utilizar el PIC24HJ256GP206, a pesar de ser lo más adecuado, la implementación del RTOS se manejó con el PIC184620, como los

resultados expresaban el correcto procesamiento de audio, solo basto modificar el código con los comandos del RTOS para que estos a su vez permitieran la adquisición de datos desde el canal analógico.

Cuando se competo el código con el RTOS incluido se obtuvieron resultados inesperados en lo planteado, empezando por que el microcontrolador no ejecutaba infinitamente la captura y procesamiento de datos, en su lugar solo lo realizaba una vez, cabe mencionar que la memoria del microcontrolador estaba ocupada a un 80%, cuestión que generaba que ocurriera este defecto en el RTOS, no podía ejecutar el programa en Tiempo Real por cuestión de exceso de carga en los recursos del microcontrolador (de acuerdo al manual del CCS).
Como se había planteado anteriormente (capitulo 3), si se utilizaba este microcontrolador no se podría solicitar mayor cantidad de puntos de muestreo más haya de 128 variables, esto claro es una desventaja para mayor exactitud de la respuesta, pero la misma cantidad de variables es una ventaja para la demostración de la investigación.

La generación inesperada de estos defectos hace dudar de que el Procesamiento de audio en un microcontrolador no sea algo fiable, sin embargo en el capítulo 3 se planteó desde un principio que lo más ideal era el uso de un microcontrolador apto para el desempeño del procesamiento de audio, precisamente para evitar los problemas anteriormente mencionados.

Esta investigación, como cualquier otra no termina con los resultados expuestos, lo que se ha obtenido es un paso para futuras investigaciones, se entiende que un sistema operativo de tiempo real embebido en un microcontrolador de alto desempeño para procesamiento de audio, si es fiable y se comprobó que es confiable, sin embargo, dependiendo si es para fines prácticos o proyectos más complejos.

Si la aplicación fuese para fines prácticos, lo que se muestra en esta esta investigación es ideal, si la aplicación fuese para proyectos más complejos, esta investigación beneficiara a saber que materiales usar desde un principio.

Si se desea llevar una aplicación más avanzada, al igual que un proyecto más complejo, lo ideal sería tener amplios conocimientos sobre Sistemas operativos en Tiempo Real, para así poder utilizar algún otro compilador que proporcione mayor soporte para la programación de los microcontroladores de la familia 24.

CONCLUSIONES

A lo largo de toda la redacción de esta investigación se intentó mostrar un panorama de lo que se necesitaba para el desarrollo del proyecto, para que así el lector tuviese la idea desde un principio de lo que se necesitaba lograr, y de lo que podría ocurrir si en un momento no se respetaba la metodología.

Los resultados que se obtuvieron fueron los esperados desde un principio, ya que lo que nos proporciona la investigación permitirá realizar pruebas para múltiples aplicaciones, como fuentes bailarinas, afinadores de audio, reconocimiento de voz aplicado a diferentes tareas, generación de filtros, etc.

Los resultados que se obtuvieron permitirán realizar proyectos a futuro mucho más confiables y de manera mucho más apropiada, finalmente el propósito de una investigación no es dejarla hasta el punto en que expone sus resultados, sino aportar esos resultados que proporcione los conocimiento para que sean implementados en otros proyectos, con el objetivo de favorecer a la mejora de nuestra calidad de vida.

A lo largo de la historia se han diseñado múltiples aplicaciones que benefician la calidad de vida humana, desde cosas muy cotidianas, como los celulares, reproductores de audio, accesorios computacionales, etc., hasta aplicaciones médicas (solo por dar un ejemplo). Sin embargo estamos conscientes de que estamos totalmente rodeados de tecnología, que se dio de un día para otro, sabemos que fue el trabajo de investigación de generaciones pasadas. Hace aproximadamente 12 años los teléfonos celulares eran robustos, pesados y tenían fallas, hace 5 años era común ver teléfonos celulares con cámara, hace dos años se generaron múltiples tareas y aplicaciones para los celulares (iPhone). Esto es solo un pequeño ejemplo para demostrar y apoyar lo que se explica.

Es inevitable tener resultados inesperados dentro de cualquier investigación, sin embargo dichos resultados no generan que se retroceda en llegar a la meta, solo generan ligeros desvíos de los cuales se puede obtener otro punto de vista.

Lo que sucedió en la investigación nos muestra que se debe manejar múltiples compiladores para la programación de Microcontroladores, varias pruebas experimentales nos permiten observar que es un mejor compilador que CCS para la programación de microcontroladores de la familia 24, sin embargo no maneja un Sistema Operativo de Tiempo Real. Habría la necesidad de buscar e implementar uno.En conclusión se pudo apreciar que para un correcto uso de un RTOS embebido en un microcontrolador para el procesamiento de audio, se deben de tener en cuenta varios factores:

- Primero la aplicación en la que se va a implementar (fines prácticos o avanzados).
- Si es para fines prácticos, lo que se muestra en esta investigación es lo ideal, pero si es para fines complejos se deberá tener conocimiento de varios compiladores de microcontroladores, así como tener conocimiento de diferentes Sistemas Operativos en Tiempo Real.
- El código de la FFT que se usó en esta aplicación es un aporte ideal que se puede implementar en cualquier otro compilador, por lo que en esa parte no existe ningún problema.

De igual manera se puede apreciar que esta investigación cumple satisfactoriamente su objetivo, al igual que los objetivos planteados en ella misma, pudiendo decir que un Sistema Operativo en Tiempo Real embebido en un Microcontrolador mejora el desempeño de las tareas.

BIBLIOGRAFÍA

[1] Robert Oshana, “DSP Software Development Techniques for Embedded and Real Time System”, Newnes, 2006.

[2] Stuart R., Ball P. E., “Embedded Microprocesor Systems: Real World Design”, Terceraedición,Newnes.

[3] Leon W. Couchi II, “Sistemas de Comunicación Digitales y analógicos”, Séptima edición, Prentice Hall.

[4] Tocci, Widmer, “Sistemas digitales Principios y Aplicaciones”, Octava edición, Prentice Hall.

[5] Roy Blake, “Sistemas Electrónicos de Comunicación”, Séptima edición, Thomson.

[6] Eduardo García Breijo, “Compilador C CCS y Simulador Proteus para Microcontroladores PIC”, Primera edición, Alfaomega.

[7] Lucio Di Jasio, “Programing 32-bit Microcontrollers in C Exploring the PIC 32”, Primera edición, Newnes.

[8] Keith E. Curtis, “Embedded Multitasking with Small Microcontrollers”, Primera edición, Newnes.

[9] Insup Lee, Joseph Y-T. Leung, Sang H. Son, “Handbook of Real-Time and Embedded Systems”, Primera edición, Chapman & Hall/CRC.

[10] Jivan S. Parab, Vinod G. Shelake, Rajanish K. Kamat, Gourish M. Naik, “Exploring C for Mocrocontrollers”, Primera edición, Springer.

[11] Dogan Ibrahim, “Microcontroller Based Applied Digital Control”, Primera edición, Wiley.

[12] Owen Bishop, “Microelectronics System and Devices”, Primera edición, Newnes.

[13] http://www.elai.upm.es/spain/Publicaciones/pub99/intropds.pdf

[14] lc.fie.umich.mx/~jrincon/manual%20PICs%20Ruddy.pdf

[15] http://www.ucontrol.com.ar/wiki/index.php?title=RTOS_para_PIC_de_CSS

[16] http://ael.110mb.com/informatica/Int8051.pdf

APENDICE

Código del sistema operativo en tiempo real para el procesamiento de audio

```
#include <18f4620.h>
#device *=16
#fuses  HS,NOLVP,NOWDT,NOPUT,MCLR,NODEBUG,NOPROTECT
#use delay(clock=40000000)
#use rtos(timer=0,minor_cycle=10us)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
//#int_TIMER1

#ZERO_RAM
#include <math.h>


float32 ReX[8];//={20.0,210.0,60.0,330.0,10.0,70.0,30.0,160.0};
float32 ImX[8]={0.0,0.0 ,0.0,0.0 ,0.0,0.0,0.0,0.0};
float32 abs[8];
float32 mag[8];
float32freq[8];

int8sem;
int16 N=8;
int16 Nmenos1=7;        //Nmenos1=N-1
int16 Ndiv2=4;          //Ndiv2=N/2
int16 j=4;              //j=Ndiv2
int16 k=0;
int16 M=3;              //(int)Log(N, 2)  FFT de 64 puntos Log2(64) = 6
int16elementosSubDFT = 0, elementosSubDFTDiv2 = 0;
```

```
int16 Jmenos1=0,ip=0;

static float32 tiempoR = 0.0, tiempoI = 0.0;
static float32 espectroR = 0.0, espectroI = 0.0;
static float32 Real = 0.0, Imag = 0.0;

unsignedint c=0;

//void FFT(void);

#task(rate=10us,max=10us)
void FFT ();

#task(rate=10us,max=10us)
voidimpresion ( );

void main()
{
setup_adc_ports(all_analog);      //Puerto A analógico
setup_adc(ADC_CLOCK_INTERNAL);    //relojconvertidor AD interno
set_adc_channel(0);
sem=1;
for(;;)
rtos_run ( );
}

void FFT()
{
int16 i=0,w=0,L=0,A=0,B=0;

//Hacer el reordenamiento: decimacion en tiempo
```

//Los elementos inicial y final del arreglo quedan igual despues del ordenamiento,

//no tiene caso modificarlos. La decimació se evalua desde n = 0 hasta N-1. Iniciando en

//n = 1 y finalizando en n = N-2, los elementos inicial y final quedan intactos.

```
for (w=0; w<N; w++)
  {
ReX[w]=read_adc();
//printf("%3.6e\n\r",ReX[w]);
}

for (i=1; i<=(N-2);i++)
  {
if (i<j)              //Para una FFT de 64 ptos, se entra a este if seis veces
    {
tiempoR = ReX[j];
tiempoI = ImX[j];
ReX[j] = ReX[i];
ImX[j] = ImX[i];
ReX[i] = tiempoR;
ImX[i] = tiempoI;
    }
    k = Ndiv2;
while (k<=j)
    {
       j=j-k;
       k=(int16)(k/2);
    }
```

```
        j=j+k;
    }

      //Proceso de sintesis
    for (L=1; L<=M; L++)      //Ciclo para cada etapa (M=6 ciclos para una DFT de 64 ptos)
      {
        //El número de elementos de cada sub DFT va creciendo en potencias de 2 conforme
        //avanzan las etapas de sintesis. elementosSubDFT es el número de elementos que tiene
        //cada sub DFT según sea la estapa actual.
    elementosSubDFT =(int16)Pow(2,L);
        elementosSubDFTDiv2=elementosSubDFT/2;
    Real= 1;
    Imag= 0;
    espectroR =(float32)(Cos(pi/elementosSubDFTDiv2));
    espectroI =(float32)((-1)*(Sin(pi/elementosSubDFTDiv2)));

    for (A=1; A<=elementosSubDFTDiv2; A++) //Ciclo para cada sub DFT
        {
          Jmenos1 = A-1;
    for (B=Jmenos1; B<=Nmenos1; B=B+elementosSubDFT)        //Lazo para cada mariposa
    {
    ip =B + elementosSubDFTDiv2;
    tiempoR=(float32)(ReX[ip]*Real)-(float32)(ImX[ip]*Imag);
    tiempoI=(float32)(ReX[ip]*Imag)+(float32)(ImX[ip]*Real);
    ReX[ip]=ReX[B]-tiempoR;
    ImX[ip]=ImX[B]-tiempoI;
    ReX[B]=ReX[B]+tiempoR;
```

```
ImX[B]=ImX[B]+tiempoI;

        }

tiempoR =(float32)Real;
        Real=(float32)(tiempoR*espectroR)-(float32)(Imag*espectroI);
Imag=(float32)(tiempoR*espectroI)+(float32)(Imag*espectroR);
     }
  }
}

voidimpresion()
{
rtos_wait(sem);

for(c=0;c<N;c++)
    {
printf("R- %3.6g,I- %3.6g\n\r",ReX[c],ImX[c]);
       abs[c]=sqrt((float32)pow(ImX[c],2)+(float32)pow(ReX[c],2));
mag[c]=(float32) (20*(log10(abs[c])));

freq[c]=(c*1000)/N;

       //printf("%3.6e\n\r",mag[c]);
    }

rtos_signal(sem);
rtos_yield();
}
```

Código del PID con RTOS

```
#INCLUDE <16F877.h>
#device adc=10
#use delay(clock=4000000)
#fuses XT,NOWDT
#Byte TRISC = 0x87

#use rtos(timer=0,minor_cycle=1ms)      //Directiva del RTOS

int16 valor;          //lectura de temperatura
int16 control;        //valor del PWM
float a=0.1243;    //constantes del PID para 100ms
float b=0.00006;
float c=62.1514;
float temp_limit=500.0;           //temperatura a alcanzar
float rT,eT,pT,qT,yT,uT;         //variables de ecuaciones
float pT_1=0.0;
float eT_1=0.0;
float max=1000.0;                      //límites máximo y mínimo de control.
float min=0.0;

#task(rate=1ms, max=1ms)          //Indica que la siguiente función es una
Tarea.
void pid ( );                             //que se ejecutará cada 100ms.

void main(){
TRISC=0;
setup_timer_2(t2_div_by_4,249,1);              //periodo de la señal PWM a
1ms
setup_ccp1(ccp_pwm);                          //Módulo CCP a modo PWM
```

```
        setup_adc_ports(all_analog);                  //Puerto A analógico
        setup_adc(ADC_CLOCK_INTERNAL);    //reloj convertidor AD interno
        set_adc_channel(0);                                 //Lectura por el canal 0

        rtos_run ( );                    //Inicia la operación de RTOS
        }

          void pid ( )                   //Tarea PID
        {
         output_bit( PIN_C0, 0);
          valor=read_adc();
         yT=valor*5000.0/1024.0;
         rT=temp_limit;
         eT=rT-yT;
         pT=b*eT+pT_1;
         qT=c*(eT-eT_1);
         uT=pT+a*eT+qT;
         if (uT>max) {
            uT=max;}
         else {
            if (uT<min){
               uT=min;}
               }
         control=uT;
         set_pwm1_duty(control);
         pT_1=pT;
         eT_1=eT;
}
```